Handling Insider Attacks in Wireless Sensor Networks

Vom Fachbereich Informatik der
Technischen Universität Darmstadt genehmigte

Dissertation

in englischer Sprache
zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)
vorgelegt von

Dipl.-Inform.
Christoph Krauß
geboren in Offenbach/Main

Darmstadt 2010
Hochschulkennziffer D17

Erstreferent: Prof. Dr. Claudia Eckert
Koreferent: Prof. Dr. Johannes Buchmann

Tag der Einreichung: 23.03.2010
Tag der Disputation: 10.05.2010

Wissenschaftlicher Werdegang[1]

Seit 03/2009	Wissenschaftlicher Mitarbeiter am Fraunhofer-Institut für Sichere Informationstechnologie SIT
10/2004 - 02/2009	Wissenschaftlicher Mitarbeiter am Fachbereich Informatik der Technischen Universität Darmstadt im Fachgebiet Sicherheit in der Informationstechnik
10/1998 - 09/2004	Studium der Informatik an der TU-Darmstadt

Ehrenwörtliche Erklärung[2]

Hiermit erkläre ich, dass ich die vorliegende Arbeit, mit Ausnahme der in ihr ausdrücklich genannten Hilfen, selbstständig verfasst habe.

[1] gemäß §20 Abs. 3 der Promotionsordnung der TU Darmstadt
[2] gemäß §9 Abs. 1 der Promotionsordnung der TU Darmstadt

Danksagung

Mein besonderer Dank gilt in erster Linie Frau Prof. Dr. Claudia Eckert für die Betreuung meiner Arbeit. Traten Schwierigkeiten auf, so unterstützte sie mich bei der Problemlösung, ihre konstruktiven und kritischen Vorschläge brachten mich weiter und steigerten die Qualität meiner Arbeit. Weiterhin schaffte sie eine äußerst angenehme Arbeitsatmosphäre an ihrem Lehrstuhl für Sicherheit in der Informationstechnik im Fachbereich Informatik an der Technischen Universität Darmstadt, wodurch man motiviert und mit Spaß an die Arbeit ging.

Ebenso danke ich Herrn Prof. Dr. Johannes Buchmann für sein Interesse an meiner Arbeit und die Übernahme des Koreferates.

Mein Dank gilt außerdem allen weiteren ehemaligen Kollegen vom Lehrstuhl für Sicherheit in der Informationstechnik für ihre Unterstützung und Freundschaft. Diese sind – in alphabetischer Reihenfolge – Michael Benz, Thomas Buntrock, Thorsten Clausius, Lars Fischer, Sascha Müller, Taufiq Rochaeli, Patrick Röder, Christian Schneider, Thomas Stibor, Frederic Stumpf, Omid Tafreschi und Henny Walter.

Zu Dank bin ich ebenfalls Dr. Kpatcha Bayarou und Dr. Markus Schneider vom Fraunhofer-Institut für Sichere Informationstechnologie SIT verpflichtet, wobei der Anstoß überhaupt zu promovieren von Dr. Kpatcha Bayarou kam. Den größten Einfluss auf diese Arbeit hatte aber Dr. Markus Schneider. Er unterstütze mich bei allen Fragestellungen und seine konstruktiven Denkanstöße brachten mir neue Sichtweisen und motivierten mich zur Weiterarbeit. Seine Unterstützung und Motivation hat erst dazu geführt, dass ich diese Arbeit fertig geschrieben habe.

Kurzfassung (Deutsch)

Drahtlose Sensornetze stellen eine noch relativ junge Technologie zur Informationsgewinnung und -verarbeitung dar. Ein Sensornetz besteht üblicherweise aus vielen kleinen, in ihren Ressourcen stark beschränkten Sensorknoten. Diese führen Messungen über physikalische Phänomene durch, verarbeiten diese Daten, erzeugen einen Report und senden diesen mittels Hop-zu-Hop Kommunikation zu einer zentralen Datenverarbeitungseinheit, genannt Sink. Je nach Szenario erfolgt die Datenerfassung und -verarbeitung auch durch mehrere Sensorknoten gemeinsam, z.B. zur Bestimmung der mittleren Temperatur in einem größeren Areal.

Durch ihre vielfältigen Einsatzmöglichkeiten erhalten Sensornetze immer größere Aufmerksamkeit. Entstammten die ursprünglichen Einsatzszenarien noch größtenteils dem militärischen Umfeld, z.B. Erfassung von feindlichen Truppenbewegungen, zeichnen sich mittlerweile immer mehr zivile Einsatzszenarien, wie z.B. die Überwachung der strukturellen Integrität von Gebäuden und kritischen Infrastrukturen, ab.

Um die Funktionalität eines Sensornetzes insbesondere unter Angriffsbedingungen zu gewährleisten, müssen Sicherheitsmechanismen integraler Bestandteil jedes Sensornetzes sein. Sensornetze unterscheiden sich jedoch von klassischen (drahtlosen) Netzen, was die Absicherung erheblich erschwert. Gründe hierfür sind die Ressourcenbeschränkung der einzelnen Sensorknoten, die drahtlose Multihop-Kommunikation und die meist vorhandene Möglichkeit der Kompromittierung von Sensorknoten durch einen Angreifer. Knotenkompromittierungen sind in Sensornetzen besonders einfach durchzuführen, da Sensorknoten meist in unkontrollierten oder sogar feindlichen Gebieten ausgelegt sind und aus Kostengründen üblicherweise keine manipulationsresistente Hardware eingesetzt wird. Durch eine Knotenkompromittierung erhält ein Angreifer somit Zugriff auf alle Daten, wie z.B. kryptographische Schlüssel, die auf einem Sensorknoten gespeichert sind. Hierdurch werden Sicherheitsmechanismen, wie z.B. wechselseitige Authentifikation der einzelnen Sensorknoten, wirkungslos. Als Folge dessen kann sich ein Angreifer als berechtigter Sensorknoten ausgeben und als legitimes Mitglied des Sensornetzes Angriffe durchzuführen. Solche Angriffe werden in dieser Arbeit als *Insider Angriffe* bezeichnet und stellen ein ernsthaftes Problem für viele Sensornetze dar.

In dieser Arbeit werden Konzepte und Mechanismen entwickelt, wie man mit Insiderangriffen in Sensornetzen umgehen kann. Den Beitrag dieser Arbeit kann man grob in zwei Teile gliedern: Zunächst wird eine allgemeine Klassifikation von Schutzmaßnahmen gegen Insiderangriffe vorgeschlagen. Im zweiten Teil werden Sicherheitsprotokolle vorgeschlagen, die verschiedene Insiderangriffe abwehren oder deren mögliches Schadenspotential begrenzen.

In der Klassifikation wird zunächst nach den grundsätzlichen Strategien unterschieden, die man verfolgen kann, um mit Insiderangriffen umzugehen. Die Strategien werden anschließend weiter nach den Mechanismen die sie umsetzen unterteilt. Bestehende Arbeiten werden nach den eingesetzten Mechanismen und verfolgten Strategien eingeordnet, um so systematisch offene Probleme und besondere Eigenschaften in den jeweiligen

Bereichen zu identifizieren. Solch eine systematische Betrachtung wurde in Sensornetzen bisher noch nicht durchgeführt. Die erarbeiteten Ergebnisse können als Basis für den Entwurf von neuen Sicherheitsprotokollen verwendet werden.

Die im zweiten Teil dieser Arbeit vorgestellten Protokolle decken verschiedene Bereiche ab. Zunächst wird ein Protokoll zum Schutz gegen einen Denial-of-Service Angriff vorgeschlagen bei dem der Insider viele gefälschte Nachrichten einschleust oder alte Nachrichten wieder einspielt. Diese Nachrichten werden über viele Sensorknoten weitergeleitet, die sowohl hierdurch überlastet werden als auch ihre knappen Energiereserven (vollständig) verschwenden. Im Gegensatz zu bestehenden Arbeiten, die grenzwertbasiert gefälschte Nachrichten probabilistisch ausfiltern, ermöglicht das vorgeschlagene Protokoll gefälschte Nachrichten sofort auszufiltern und toleriert eine beliebige Anzahl an kompromittierten Sensorknoten.

Anschließend werden Protokolle zum Schutz gegen Insiderangriffe bei denen kompromittierte Sensorknoten zur Täuschung des Sinks falsche Reports einschleusen vorgestellt. Bestehende Protokolle verwenden hierzu einen redundanzbasierten Ansatz, bei dem mehrere Sensorknoten gemeinsam einen Report erzeugen müssen, damit er gültig ist. Hierbei ist es jedoch möglich, dass ein einzelner kompromittierter Sensorknoten eine erfolgreiche Reporterzeugung verhindern kann. Bisher wurde eine ausschließlich auf ein Protokoll anwendbare Erweiterung für dieses Problem vorgeschlagen, bei der die angreifenden Sensorknoten nicht identifiziert und ausgeschlossen werden können. In dieser Arbeit werden zwei Protokolle zum Schutz gegen eingeschleuste Reports vorgestellt, die es erstmals ermöglichen solche Angriffe zu erkennen und die verantwortlichen Sensorknoten auszuschließen. Weiterhin können die vorgeschlagenen Protokolle als Erweiterung zu beliebigen anderen Protokollen eingesetzt werden.

Neben diesen Protokollen wird ein grundsätzlicher Ansatz untersucht, wie man in bestimmten Szenarien Insiderangriffe verhindern und versuchte Knotenkompromittierungen erkennen kann. Hierzu wird der Einsatz von manipulationsresistenter (engl. tamper-resistant) Hardware in Form des Trusted Platform Modules (TPM) vorgeschlagen. Aus Kostengründen werden nur einige Sensorknoten, die besondere Aufgaben wie Schlüsselmanagement, Lokalisierung oder Zeitsynchronisierung für andere Sensorknoten durchführen und somit ein lohnenswertes Ziel für einen Angreifer darstellen, mittels TPM geschützt. Zur Erkennung von Manipulationsversuchen an diesen Sensorknoten werden zwei neue effiziente Attestationsprotokolle vorgeschlagen. Diese sind an die Ressourcenbeschränkungen von Sensornetzen angepasst, d.h. der Energieverbrauch für Berechnungen und Kommunikation ist sehr gering, da keine Public Key Operationen auf den verifizierenden Knoten benötigt wird und nur wenige, kurze Nachrichten ausgetauscht werden. Weiterhin hat der vorgeschlagene Ansatz Vorteile gegenüber softwarebasierter Attestation, da nun auch eine erfolgreiche Attestation über mehrere Hops möglich ist, was wichtig für Sensornetze ist. Mit dem vorgeschlagenen Ansatzes ist es nun möglich, die Vertrauenswürdigkeit von bestimmten Sensorknoten auch in unkontrollierten oder feindlichen Gebieten überprüfen zu können und sie so für besondere Aufgaben einzusetzen.

Abstract

Wireless sensor networks are a relatively new technology for information gathering and processing. A sensor network usually consists of many, resource constrained sensor nodes. These nodes perform measurements of some physical phenomena, process data, generate reports, and send these reports via multihop communication to a central information processing unit called sink. Depending on the scenario, information gathering and processing is collaboratively performed by multiple sensor nodes, e.g., to determine the average temperature in a certain area.

Sensor networks can be used in a plethora of application scenarios. Emerging from military research, e.g., sensor networks for target tracking in a battlefield, sensor networks are nowadays used more and more in civil applications such as critical infrastructure monitoring.

For ensuring the functionality of a sensor network, especially in malicious environments, security mechanisms are essential for all sensor networks. However, sensor networks differ from classical (wireless) networks and this consequently makes it harder to secure them. Reasons for this are resource constraints of the sensor nodes, the wireless multihop communication, and the possibility of node compromise. Since sensor nodes are often deployed in unattended or even hostile environments and are usually not equipped with tamper-resistant hardware, it is relatively easy to compromise a sensor node. By compromising a sensor node, an adversary gets access to all data stored on the node, such as cryptographic keys. Thus, deployed security mechanisms such as node-based authentication become ineffective and an adversary is able to perform attacks as a "legitimate" member of the network. Such attacks are denoted as *insider attacks* and pose a serious threat for wireless sensor networks.

In this thesis, we develop concepts and mechanisms to cope with insider attacks in wireless sensor networks. The contribution of this thesis is twofold. First, we propose a new general classification to classify the different approaches to protect against insider attacks. Second, we propose several security protocols to protect against insider attacks.

In our classification, approaches to protect against insider attacks are first distinguished by the implemented security strategy. The respective strategies are further subclassified by the applied mechanisms. Related work is integrated in the classification to systematically identify open problems and specific properties in the respective areas. The results may be a basis for future protocol design.

The protocols, proposed in the second part of this thesis encompass different areas. First, we propose a protocol to protect against a serious Denial-of-Service attack where an adversary injects or replays a large amount of false messages to overload many message forwarding nodes and to (totally) waste their scarce energy resources. Proposed approaches usually apply threshold-based mechanisms to filter such messages out. The drawback of this approach is that messages are not filtered out immediately and if the

threshold of compromised nodes is reached, the attack becomes again possible. Our protocol is able to immediately filter such messages while tolerating an arbitrary number of compromised sensor nodes.

Further mechanisms are required to additionally protect against an insider attack where an adversary injects false reports to deceive the sink. Usually a redundancy-based approach is used where a report is only valid if it has been collaboratively generated by multiple sensor nodes. However, previously proposed protocols are susceptible to an insider attack where an adversary that has compromised only a single node might be able to impede a successful report generation. So far, only one protocol has been proposed to cope with this issue. However, it is a specific enhancement for a particular protocol and the attacking nodes cannot be identified and excluded. In this thesis, we propose two protocols which protect against the injection of false reports and also enable the detection and exclusion of nodes trying to disrupt the collaborative report generation. In addition, our protocols can be used in combination with or as an extension to any other protocol.

In addition, we investigate a general approach to prevent insider attacks and to detect compromised nodes in certain scenarios. We propose to use tamper-resistant hardware in form of the Trusted Platform Module (TPM). Due to cost reasons, the TPM is integrated only in some special sensor nodes that perform some special tasks such as key management, localization or time synchronization in the sensor network. These nodes are a valuable target for an adversary. To detect tampering attempts on these nodes, we propose two efficient attestation protocols. In contrast to attestation protocols proposed for "classical" networks, our protocols have a low communication and computational overhead. They do not require expensive public key operations on the verifying nodes and the few exchanged messages are very short. In addition, compared to software-based attestation, our protocols have the advantage to enable attestation along multiple hops which is of high concern in sensor networks. Using our approach, it is possible to verify the trustworthiness of certain sensor nodes even in unattended or hostile environments making them suitable to perform special tasks.

Contents

1 Introduction

Wireless Sensor Networks (WSNs) provide a technological basis for many different applications. Applications range from battlefield or critical infrastructure surveillance over emergency response to health care scenarios. Depending on the application, the monitored environment can be covered by hundreds or even thousands of sensor nodes. Sensor nodes consist of a sensing, a processing, a transceiver, and a power unit. The sensing unit is used to perform some measurements of some physical phenomena, e.g., temperature, pressure, movements. The processing unit consists of a slow CPU and a restricted memory space. It enables a sensor node to pre-process the measured data before sending it using the transceiver unit. The measurements are sent to a central base station called sink. The power unit usually consists of a battery pack. In addition, a sensor node may be equipped with some additional application-dependent components such as location finding system, actuators etc. Most WSN applications scenarios, especially large-scale WSNs with thousands of sensor nodes, require that an individual sensor node has to be cheap. As a result, the available resources are extremely constrained. The constrained computational power of a sensor node requires the use of only efficient operations which are suitable for the limited CPU and the restricted memory space. Especially the limited energy resources make energy saving of paramount importance to achieve a long lifetime of a WSN. Thus, the overhead for computation and communication must be low. Since communication requires a significant amount of energy, only as few messages as necessary which are as short as possible should be transmitted. Since the range of the wireless transceiver is also limited, messages are sent in a multihop communication, i.e., sensor nodes forward messages hop by hop to the destination.

The focus of this thesis is security in WSNs. More precisely, we investigate the serious threat of insider attacks and how to cope with them. As the above mentioned properties of WSNs indicate, securing WSNs is a challenging task, especially, against inside adversaries. In Section 1.1, we motivate this problem in detail. After that, we present the contribution of this thesis in Section 1.2. In Section 1.3, we give a short outline of the thesis structure.

1.1 Motivation

For ensuring the functionality of a WSN, especially in malicious environments, security mechanisms are essential for all sensor networks. Obviously, scenarios such as emergency response or battlefield surveillance require security mechanisms. These WSNs are valuable targets for an adversary where he can cause serious damage. For example, the functionality and correct operation of a WSN for battlefield surveillance can be disrupted by an adversary that injects bogus location information. However, even WSNs

deployed for security- or safety-uncritical applications, such as wildlife monitoring, require security. Without security, even a very limited adversary can disrupt the whole functionality of the WSN. Thus, security mechanisms, appropriate for the respective security requirements of the scenario, should be deployed in every WSN.

However, securing WSNs is a challenging task [114, 216, 236, 185]. The resource constraints allow only the use of efficient security mechanisms and protocols. For example, only efficient cryptography should be used. The wireless multihop communication enables an adversary to eavesdrop, inject, drop, or alter messages, maliciously participate on a route, or to perform DoS attacks by jamming the wireless channel. Since sensor nodes are often deployed in unattended or even hostile environments, and are usually not equipped with special tamper-resistant hardware, an adversary is able to compromise a sensor node. After compromising a node, an adversary has access to all data stored on the node. By analyzing the code, the adversary is able to identify the applied (security) protocols and by accessing the application data, he gets access to security data such as cryptographic keys. The adversary can misuse this data either to program its own device or to reprogram the compromised sensor node with its own malicious code and perform subsequent attacks as a "legitimate" node of the network. Such attacks, where an adversary uses data of a compromised node to circumvent the applied security mechanisms and appears to be a legitimate node of the WSN are denoted as *insider attacks*.

Examples of insider attacks can be found at nearly all layers. In the following, we name only a few examples. At the link layer, an adversary can compromise a sensor node to get access to the cryptographic keys of the applied link layer security mechanism. This mechanism is used to ensure the authenticity, integrity, and confidentiality of the exchanged messages between neighboring sensor nodes. This enables the adversary for example to perform attacks such as eavesdropping, i.e., decrypting the encrypted messages exchanged between the neighboring nodes and the compromised node. Link layer security mechanisms can be used to additionally protect messages of a higher level. Thus, circumventing the link layer security mechanism also affects these layers, e.g., an adversary is able to inject bogus routing information to disrupt the routing functionality at the network layer. The application layer is ideally suited for insider attacks since an adversary is able to perform sophisticated attacks affecting not only neighboring nodes. For example, an adversary can send false report messages to the sink which results in false alarms. This may considerably disturb the functionality of the WSN.

As the examples have shown, insider attacks can cause serious damage to a WSN. In addition, insider attacks are much easier to perform in WSNs than in classical computer systems. Classically, an inside adversary is a person with extensive knowledge of and privileged access to the target system and is able to attack inside a system's perimeter defense. This restricts the possible number of persons which are able to perform insider attacks, e.g., to former employees of a company or powerful hackers which may be able to break into the system. Security management mechanisms such as "changing passwords after an employee leaves the company" can be used to prevent a former employee to access the system again. Applying resource-consuming security mechanisms such as firewalls or intrusion detection systems (IDS) and regular system updates with security patches minimizes the risk of successful hacking attempts. Even if a system is compromised,

it can still be reinstalled to remove all malicious code and lock out the adversary. In WSNs this is all usually not possible making insider attacks a serious and relatively easy to perform threat.

Thus, it is of paramount importance to secure WSNs against insider attacks. As mentioned before, this is a challenging task. Security mechanisms to cope with insider attacks in WSNs must be designed in a well-considered way. In order to achieve this, it is very helpful to know the general strategies which can be implemented in security mechanisms. Knowing the so far approaches and the open problems in the respective areas further facilitates the development of new security mechanisms. New security mechanisms are required in sundry areas; more than can be addressed in one thesis. One of these areas requiring research is at the application layer relating to secure the report generation against certain insider attacks.

We address these issues in this thesis. In the next section, we describe the contribution of this thesis in relation to the investigated problems.

1.2 Contribution

In this thesis, we focus on the threat of insider attacks in WSNs. We investigate different ways to cope with these attacks and propose new protocols to secure WSNs against inside adversaries. In general, the contribution of this thesis can be divided in two parts: classification and protocol development.

In the classification part, we present a new classification scheme of the different strategies and mechanisms to handle insider attacks in WSNs. In general, we distinguish between the strategies prevention, detection, and recovery. The respective strategies are further subclassified by the applied mechanisms which implement the respective strategy. Related work is classified based on our classification. For all areas of our classification, we identify properties and open problems which should be considered when developing security protocols for WSNs.

The protocol development part is the main part of this thesis. We propose several efficient protocols to cope with different insider attacks.

First, we propose different protocols to enable a secure report generation in the presence of an inside adversary. We consider resource constrained WSNs which perform the principle tasks of a WSN: a query is sent to the WSN, a measurement of the corresponding physical phenomena is performed, a report is generated, and the generated report is transmitted back. The protocols apply detection and recovery strategies and consider problems identified in our classification.

Second, we investigate how tamper-resistant hardware can be used to secure WSNs. The tamper-resistant hardware enables prevention of insider attacks and the detection of tampering attempts.

The major contribution of this thesis is sketched in the following:

Classification A lot of research has already been performed to develop mechanisms and protocols to cope with different types of insider attacks in different application scenarios. For the development of new security mechanisms and protocols it is helpful to know which general strategies can be pursued, how strategies can be implemented by different types of security mechanisms, how related work can be classified, and which open problems exist in the respective areas. For this purpose, we introduce a new *two-tiered classification* [132]. First, we distinguish between the well-known strategies prevention, detection, and recovery. These three categories are the first level of our classification. For each type of strategy, we elaborate the relation to insider attacks in WSNs and identify which types of security mechanisms exist to implement the respective strategy. The different categories of the type of the security mechanism represent the second level of our classification. For each category, we present related work and show properties and open problems which should be considered when developing new security protocols.

Addressing PDoS Attacks We propose a protocol to secure the report generation against Path-based Denial of Service (PDoS) and false data injection attacks [129]. The focus of this work is the protection against PDoS attacks where an adversary replays or injects a large amount of false report messages to drain the energy resources of message forwarding nodes. A false data injection attack has the goal to deceive the sink, e.g., to cause false alarms.

Our proposed protocol implements recovery strategies to cope with these attacks. The recovery strategy addressing PDoS attacks is based on fault tolerance mechanisms. The idea is to limit the impact of the attack to the immediate neighboring nodes while protecting the rest of the network. Our protocol achieves this by tolerating an *arbitrary* number of compromised nodes and using only *efficient* symmetric cryptography to filter messages *immediately* out at the next hop. This protects all subsequent nodes from the PDoS attack and saves their scarce energy resources.

This is an improvement compared to previously proposed protocols which use threshold schemes and probabilistic filtering to cope with PDoS attacks. One disadvantage of these protocols is the use of a threshold scheme which is only secure up to a certain number of compromised sensor nodes. If the threshold is reached, the security totally breaks down. The second disadvantage is that injected messages are not filtered out immediately. Thus, the attack affects much more sensor nodes compared to our protocol.

To protect against false data injection attacks, we use a threshold-based mechanism (with the above mentioned issue) that is commonly used in many other protocols, too. The idea is that multiple sensor nodes must collaboratively generate a report for the sink. To be valid, a report must be endorsed by a certain number of other sensor nodes. This prevents an adversary that has compromised fewer nodes than the threshold value from injecting false reports. However, an adversary that has compromised a single node can invalidate this report by generating a false endorsement to prevent a successful report generation. We introduce the term False-Endorsement-Based DoS (FEDoS) attack

for this attack [130] and propose two additional protocols to address this issue. This contribution is introduced in the next paragraph.

To show the benefit of our protocol, we perform a security and a performance analysis. In the security analysis, we analyze the resilience against false data injection attacks, against PDoS attacks, and the influence of other attacks on our protocol. The performance analysis shows that our protocol has a low storage overhead and is able to significantly reduce the energy consumption compared to an unprotected network. In order to perform simulations of the protocol, a simulation environment is implemented. Simulations can be adjusted by different parameters, e.g., number of nodes of the WSN, number of compromised nodes performing PDoS attacks, protocol activated or not, etc. The simulation environment enables the visualization and evaluation of different scenarios to show the impact on the energy and storage consumption of the protocol. The performed simulations confirm the results of the theoretical analysis.

Addressing False Data Injection and FEDoS Attacks We propose two protocols which protect against false data injection attacks and also address the above mentioned FEDoS attack [130, 131]. Our protocols implement detection and recovery strategies to enable a protection against false data injection attacks without being susceptible to FEDoS attacks.

Previously, a protocol has been proposed in [150] to cope with FEDoS attacks. However, this protocol is a specific extension for an already proposed protocol to cope with PDoS attacks and uses a threshold scheme with the above mentioned issue. In contrast, our protocols are *generic* solutions and can be used in combination with any other protocol. In addition, our protocols enable the *detection and exclusion* of compromised nodes performing a FEDoS attack. The detection process does not require intensive monitoring. This is an advantage compared to usually applied detection mechanisms which require intensive monitoring of neighboring sensor nodes and thus are very energy consuming. In addition, the exclusion process is also very efficient since it does not require to transmit many messages, e.g., to inform the sink. Compromised sensor nodes are immediately after detection locally excluded. This prevents them from successfully continuing the attack and to cause further damage.

For both protocols we perform a security and performance analysis. In the security analysis, we analyze the resilience against false data injection attacks, against FEDoS attacks, and the influence of other attacks on our protocol. The performance analysis shows that the storage requirements of both protocols are feasible for current sensor hardware and the energy consumptions are low. In order to perform simulations of the protocols, the above mentioned simulation environment is extended to additionally support our proposed protocols to cope with FEDoS attacks. The simulation results confirm the low energy overhead and that the storage requirements are feasible on current sensor hardware.

Prevention of Insider Attacks We propose an approach to prevent all types of insider attacks at their root and enable the detection of tampering attempts of sensor nodes.

For this purpose, we introduce the use of tamper-resistant hardware and propose two efficient attestation protocols [133].

Prevention of insider attacks in the usually assumed homogeneous WSNs with extremely resource constrained sensor nodes is virtually impossible. In our approach we use of the Trusted Platform Module (TPM) [231] to prevent successful node compromise. However, because of the cost factor, it is not possible to equip all sensor nodes of a WSN with tamper-resistant hardware. Thus, we assume a hybrid WSN consisting of "ordinary" sensor nodes and a few special nodes which are equipped with a TPM. These special nodes perform and coordinate some special tasks, such as data aggregation, key management, localization, or time synchronization for ordinary sensor nodes. The TPM is used to protect the cryptographic keys. However, an adversary can still try to tamper with the remaining components of a sensor's system to achieve an invalid system state enabling illegal access to the TPM. These tampering attempts have to be detected.

We propose two efficient TPM-based attestation protocols for hybrid WSNs. Previously proposed TPM-based attestation protocols are not suitable for WSNs since they make intensive use of public key cryptography and introduce a high communication overhead. In contrast, our protocols are *adapted to WSNs*, i.e., ordinary sensor nodes need only to perform efficient symmetric (cryptographic) operations and the few exchanged messages are very short. In contrast to previously proposed software-based approaches which require exact time measurement, our protocols enable attestation even if nodes are *multiple hops* away from each other.

Our first proposed protocol runs in fixed time intervals, allowing multiple ordinary nodes to perform a simultaneous attestation. Direct attestation is provided by our second protocol. Both protocols use the sealing concept provided by the TPM. The protocols are not limited to the use in WSNs. They might be also appropriate for scenarios where resource constraints are an issue and efficient attestation along multiple hops is required.

We perform a security and theoretical performance analysis for both protocols. The security analysis shows that our protocols are resistant against the forging of attestations. However, an adversary may try to disrupt attestations. The performance analysis shows that the storage and energy requirements for the ordinary sensor nodes are for both protocols low. To show that both protocols can be realized with current TPMs, a proof of concept implementation is provided.

1.3 Outline

This thesis is organized as follows. In Chapter 2, we present necessary background information of WSNs and security in WSNs. Furthermore, we give a brief overview of resource consumption in WSNs and present values of the energy requirements of the wireless communication and the computation of (cryptographic) operations to secure WSNs. To be able to understand our proposed attestation protocols, we also present the necessary background information of trusted computing and the TPM in this chapter. In Chapter 3, we present the different ways how to handle insider attacks in WSNs. For this purpose, we present our two-tiered classification scheme. In Chapter 4, we introduce

the system model, i.e, the device, network, and adversary model, we assume for our proposed protocols. Afterwards, we describe our protocols. In Chapter 5, we present our protocol to cope with PDoS and false data injection attacks. The two extension protocols, addressing false data injection and FEDoS attacks, are described in Chapter 6. In Chapter 7, we present our approach to prevent insider attacks and two TPM-based attestation protocols to detect tampering attempts. Finally, we conclude this thesis in Chapter 8.

2 Background

In this chapter, we give some general background on Wireless Sensor Networks and security in Wireless Sensor Networks. Furthermore, we give a brief overview of the resource consumption in WSNs and some background on trusted computing which we use in some of our proposed protocols.

2.1 Wireless Sensor Networks

This section gives a brief introduction to Wireless Sensor Networks. Typical hardware platforms, application scenarios, and general properties of Wireless Sensor Networks are presented. For each property we briefly introduce challenges which arise when designing security protocols for Wireless Sensor Networks.

2.1.1 Basics of Wireless Sensor Networks

A *Wireless Sensor Network* (WSN) [3] is a network composed of a large number of low-cost, low-power, multifunctional *sensor nodes* that are deployed for monitoring the physical world. Sensor nodes are also often called *motes*. The term originates from dust motes[1], i.e., a single piece of dust. The main components of a sensor node are a microcontroller, memory, transceiver, power source and one or more sensors to perform measurements of some physical phenomena. The power source is mostly a battery. Sensor nodes are deployed in a sensor field. The deployment can be either done directly, by placing the sensor nodes in specific positions, or randomly, e.g., via aerial scattering in inaccessible terrains or disaster relief operations. Thus, the position of the sensor nodes in the sensor field may not be known in advance. After deployment, the sensor nodes perform some self-organization mechanisms to set up the network, e.g., by determining their neighbors and setting up routing tables. Self-organization is also required to adapt to changes of the network, e.g., caused by node failure due to energy exhaustion.

During operation of the WSN, sensor nodes perform measurements of some physical phenomena, e.g., the temperature at a certain location. This data is sent to one (or more) base station(s), called sink, for further processing. Since the transmission range of a sensor is limited, the sink may not be directly reached. Thus, messages are forwarded in a multihop communication to other sensor nodes which act as routers. Also sensor nodes may perform some operations on the data, e.g., data aggregation to decrease the amount of transmitted data. Since the transmission of data is much more cost-intensive than data processing, this is a commonly used approach to decrease the overall energy consumption. However, this may not be possible in all scenarios.

[1] `http://robotics.eecs.berkeley.edu/~pister/SmartDust/`

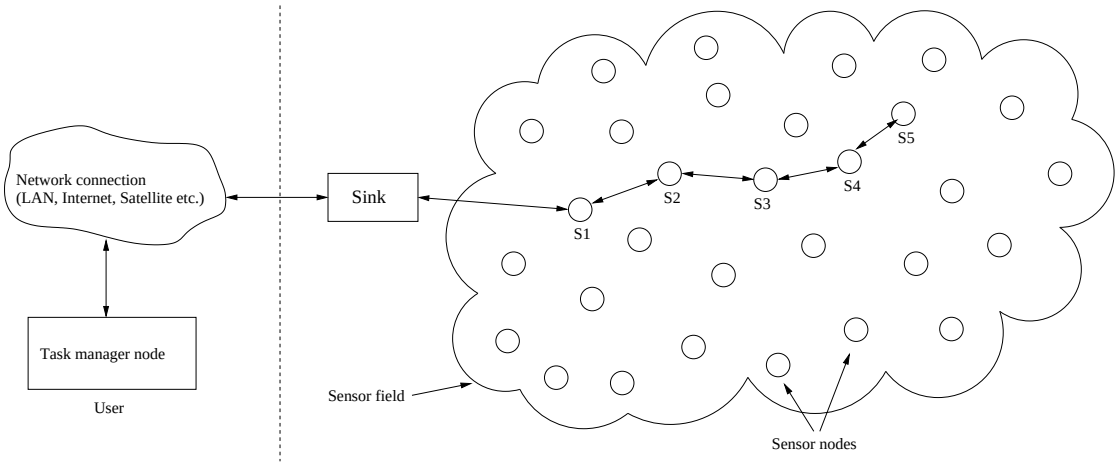

Figure 2.1: Example WSN

Figure 2.1 shows an example WSN. Multiple sensor nodes are deployed in a sensor field. The *user* generates a query, e.g., "What is the temperature at location X?" using the *task manager node*. The query is sent via a *network connection* (e.g., LAN, Internet, satellite ...) to the *sink* which further sends it to the WSN. The query is routed along sensor nodes S_1, S_2, S_3, S_4, to sensor node S_5 which performs the measurement and generates a response, e.g., "The temperature at location X is 23°C.". In this example, the response is routed back along the same route. Alternatively, the WSN may autonomously perform measurements, e.g., following a certain time schedule, and send these measurements in regular intervals to the sink.

WSNs share many properties with wireless ad hoc networks and may require similar techniques such as routing protocols. However, sensor networks differ significantly in certain areas which prohibit the (direct) usage of many protocols proposed for wireless ad hoc networks. To illustrate this issue, the differences between sensor networks and ad hoc networks are summarized [3]:

- The number of sensor nodes in a sensor network can be several orders of magnitude higher than the nodes in an ad hoc network, e.g., hundreds or even thousands of nodes in a WSN.
- Sensor nodes are often densely deployed, i.e., multiple sensor nodes are able to perform a measurement about the same or similar physical phenomena.
- Sensor nodes are prone to failures, e.g., a node fails due to battery exhaustion.
- The topology of a sensor network can change frequently, e.g., caused by node failure, mobile nodes etc.
- Sensor nodes mainly use a broadcast communication paradigm, whereas most ad hoc networks are based on point-to-point communication.

- Sensor nodes are constrained in computational power, memory space, wireless range, and especially in the available energy.
- Sensor nodes may not have global identifiers because of the large amount of overhead and large number of sensors.

The special properties and challenges of WSNs have to be considered when designing (security) protocols and algorithms for WSNs. Before we illustrate these properties and challenges in detail in Section 2.1.5, we first give a brief overview of the protocol stack in Section 2.1.2, some standard sensor network hardware in Section 2.1.3 and some example application scenarios in Section 2.1.4.

2.1.2 Protocol Stack

Nearly all communication and computer network protocols are oriented at the seven layers of the OSI model [119]. Likewise the communication architecture of WSNs can be classified in different layers. However, WSNs do not adhere as closely to the layered architecture of the OSI model as other networks for efficiency reasons. The limited resources of sensor nodes require efficient implementations and low overhead. For example, in practical WSNs, such as those using the TinyOS platform [233], the protocol stack can be roughly broken into only four major layers: physical, link, network, and application layer. Transport, session, and presentation layer are not explicitly considered.

Nevertheless, the layered model is also useful in WSNs for categorizing protocols, attacks, and defenses. Within this paper, we concentrate on the protocol stack for sink and sensor nodes introduced by Akyildiz et al. in [3]. In contrast to the traditional seven layers, the protocol stack is reduced to the following five layers:

- physical layer
- link layer
- network layer
- transport layer
- application layer.

The layered architecture has the advantage that conceptually similar functions are combined at one layer. Each layer provides services to the layer above and receives services from the layer below.

The physical layer should provide robust modulation, transmission, and receiving techniques. Since the wireless channel is susceptible to noise and sensor nodes may be mobile, the medium access control protocol at the link layer must be power-aware and able to minimize collisions. The network layer is responsible for the routing of data supplied by the transport layer. The transport layer is able to maintain the data flow if the WSN application requires it. Depending on the sensing tasks, different types of applications can be implemented at the application layer. Orthogonal to these five layers, Akyildiz et

al. define power, mobility, and task management planes which are responsible for monitoring the power, movement, and task distribution among sensor nodes. These planes help sensor nodes to coordinate sensor tasks and lower the overall power consumption.

2.1.3 Hardware

Sensor hardware can be divided in four groups [103]. *Special-purpose sensor nodes* are purposely designed to sacrifice flexibility in order to be as small and inexpensive as possible. In contrast, *generic sensor nodes* can be flexibly adjusted to different application scenarios. They provide a rich expansion interface where various different types of sensors can be attached. Typical examples are the MICA Motes [104, 105, 53] or the TelosB motes [187, 56]. *High-bandwidth sensor nodes* possess processing and communication capabilities to deal with complex sensor streams, including video and voice processing. An example are the Bluetooth equipped BTnodes [29]. *Gateway nodes* can act as the sink to provide the link between the WSN and backbone infrastructure, i.e., the network connection to the task manager node and the user. The Stargate platform [55] is a typical example for gateway nodes.

Hardware platforms for WSNs are developed by many research groups and commercial companies[2]. Since WSNs are currently in a development stage and there is no true killer application that would decrease the costs, there exists no platform which dominates the market. For many research groups it is often more convenient and even less expensive to build their own WSN devices instead of buying commercial ones.

However, the above mentioned Berkeley MICA Motes [104, 53] and their clones are broader used as other platforms and are accepted as the de facto platform in the research community. Algorithms and protocols proposed for WSNs usually assume hardware resources comparable to the MICA Motes. We make the same assumption for our developed protocols.

Figure 2.2 shows a *MICA2* Mote [104, 53]. The sensor node is equipped with an Atmel ATmega128L [52] processor which is based on a Harvard RISC architecture. The maximum clock signal of the CPU is 16 MHz which provides a throughput up to 16 MIPS. The MICA2 mote provides 128 kbyte Program Flash Memory, 512 kbyte Measurement Flash, and 4 kbyte Configuration EEPROM. The radio is a TI CC1000 (formerly ChipCon) with a data rate of 38.4 kbps. The standard packet size is 36 byte (with a payload of 29 byte). The successor MICAz has similar properties except that the data rate is increased to 250 kbps [54] and the packet size can be up to 128 byte using the ZigBee [273] transceiver CC2420. ZigBee is a short-range, low-power, low-cost, low-data-rate wireless multihop networking technology standard. It is built upon the Physical (PHY) and Medium Access Control (MAC[3]) layers of the IEEE 802.15.4 standard [218]. ZigBee specifies network and application layer, as well as the security service provider. MICA2

[2] `http://wsn.oversigma.com/wiki/index.php?title=WSN_Platforms` lists more than 30 different hardware platforms.

[3] Note: In this context the abbreviation MAC stands for Medium Access Control and not for Message Authentication Code. In the remainder of this thesis we use MAC always for Message Authentication Code.

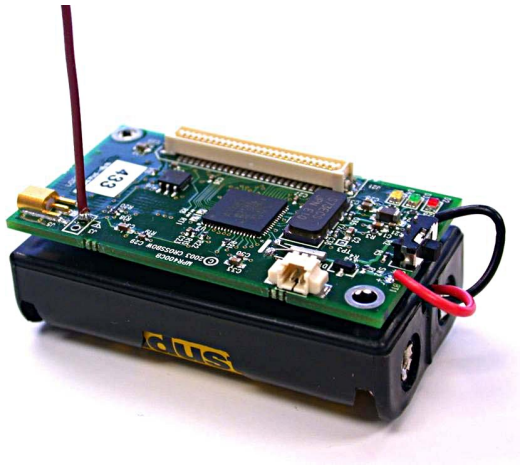

Figure 2.2: Berkeley MICA 2 Mote

Figure 2.3: Sun SPOT Mote

and MICAz nodes are powered with two AA batteries. The energy requirements of the MICA2 motes are stated differently in the literature (see Section 2.3 for details). To save energy, sensor nodes can use different sleep modes. The MICA motes use the TinyOS [233] operating system that was developed for resource constrained sensor nodes. It is written in nesC [94], a programming language for deeply networked systems. Table 2.1 summarizes the properties of the MICA2 sensor node.

Clock Signal	up to 16 MHz
Word Length	8 bit
Program Flash Memory	128 kbyte
Measurement Flash	512 kbyte
Configuration EEPROM	4 kbyte
Data Rate	38.4 kbps

Table 2.1: MICA2 Hardware [53] with Atmel ATmega128L processor [52]

Another hardware platform called *Sun SPOT* [228] is developed by Sun Microsystems. The sensor nodes are equipped with a 180 MHz 32-bit ARM920T processor with 512 kbyte RAM and 4 Mbyte Flash-memory. The radio is a TI CC2420 and is IEEE 802.15.4 compliant. An USB interface is also present. A Sun SPOT node is powered with a 3.7V, 750 mAh lithium-ion battery that is rechargeable via the USB interface. Sun specifies the operation time up to 7 hours with both CPU and the radio active. By having the processor in sleep mode and by turning off the radio the operation time is extended. Figure 2.3 shows a Sun SPOT node.

In this thesis we do not assume sensor nodes with such computational power although it might enable more sophisticated protocols. However, the increased energy consump-

tion would result in such a short lifetime of the WSN, that it might not be practically used in the majority of application scenarios.

2.1.4 Application Scenarios

WSNs can be used in large number of different applications. Originating from military research projects, WSNs are expected to be used in more civil applications. In testbeds, a variety of different application scenarios for WSNs have been investigated, e.g.,

- *habitat monitoring* [161, 229],
- *emergency response* [158],
- *glacier surveillance* [164],
- *volcano surveillance* [245],
- *wildlife monitoring* [161, 120, 157],
- *MarathonNet* [182],
- *traffic monitoring* [50],
- *surveillance missions* [101],
- *structural monitoring* [254],
- *vehicle tracking* [214],
- *mobile countersniper system* [235].

As this wide variety of different applications shows, WSNs may have totally different properties and characteristics. WSNs may vary from small networks to large networks consisting of hundreds or thousands of sensor nodes. The used sensor node hardware may vary from tiny, battery-powered and extremely resource constrained sensor nodes to powerful sensor nodes with permanent energy supply that are able to perform extensive computations. Just as well, the WSN can consist of different types of sensor nodes where the more powerful sensor nodes are able to perform special tasks. These are only a few examples of different characteristics and properties of WSNs. We illustrate these properties of WSNs in detail in Section 2.1.5.

This wide area of application scenarios induces also different security requirements for the respective scenario. For example, the security requirements for a WSN deployed for wildlife monitoring may not be as high as for a WSN used in healthcare. Security certainly is also important for applications such as military or police networks, emergency response, or safety-critical business operations. For example, after a natural disaster like a tornado, hurricane, flood, or earthquake, WSNs could be deployed for real-time safety feedback to assist the rescue teams.

2.1.5 Properties and Challenges

The different application scenarios presented in the previous section point out, that WSNs may have very different properties. As a result, it is complicated to develop general protocols for WSNs. Protocols are often adapted for specific applications and therefore are less or not suitable for different applications. This is also true for security protocols. To illustrate the variety of properties, we present a list of the most relevant properties of WSNs which are also of concern when developing security protocols. The list is a accumulation of the most commonly used properties in the literature. However, the list is not exhaustive since for example future WSN application scenarios may introduce new application specific properties. For each property, we also show the challenges which have to be solved when developing security protocols.

Resource Constraints. As already mentioned in Section 2.1.3, the used sensor node hardware may differ in various ways: computational power, memory space, wireless transmission range, and available energy resources. Furthermore, sensor nodes may be equipped with different sensor boards to perform measurements about different physical phenomena. However, it is generally assumed that sensor nodes are severely resource constrained.

Many application scenarios demand that sensor nodes have to be tiny and cheap. In the envisioned smart dust WSNs, a sensor node will have the size of a dust particle. This indicates that Moore's law is not valid for WSNs, i.e., the available resources remain nearly constant but sensor nodes will be much smaller and cheaper in the future. Thus, although the MICA2 motes, presented in Section 2.1.3, are still a research platform and much larger, we assume that future sensor nodes have similar properties and resources. As a result, security protocols cannot be developed with the assumption that future sensor nodes provide higher resources. Thus, the challenge is the development of security protocols that are working with a small CPU, little memory, short transmission range, and, most important, are energy efficient.

The CPU of a MICA2 mote has very limited computing power. The 8-bit Atmel ATmega128L operates at 16 MHz and provides only 16 MIPS. Thus, complex (cryptographic) operations cannot be reasonably realized. Public key cryptography, for example, may be generally computable on such resource constrained CPUs, as for example [99] shows. However, such computations require much more time than efficient symmetric operations. For real time applications, computationally intensive cryptographic primitives are not suitable. More important, such operations extensively require energy and thus, significantly decrease the lifetime of a sensor node. However, if sparsely used, they may be applicable for infrequent operations, e.g., being used only once in the initialization phase of a WSN.

With a size of only 4 kbyte, the RAM must be efficiently used. Application data, network data, currently processed measurement data, and other middleware data already occupy the most part of the RAM. Thus, there is not much space left to store data required for security protocols such as keys, certificates, etc. However, if certain (security

related) data is currently not required, it may be possible to store this data in the 512 kbyte measurement flash. When needed, this data can be copied to the memory.

Likewise, the range and data rate of the wireless transceiver is very limited. The energy consumption increases quadratic to the transmission distance. Thus, to conserve energy, WSNs communicate in a multihop way using intermediate sensor nodes as routers to transmit only over short distances. The transmission of data is the most cost-intensive factor in WSNs (see Section 2.3 for details), and thus, the amount and size of the transmitted data should be minimized. Therefore, security protocols should not significantly increase the overhead by introducing many large messages.

Finally, the most important constraint is the limited energy of the sensor nodes. The above mentioned constraints (small and energy efficient CPU, little memory, short transmission range) are (except of the cost factor) a result of the constraints in terms of energy. The parts of a sensor node are chosen by a cost to energy-efficiency tradeoff. Mostly, sensor nodes are powered with batteries with a limited capacity and have no permanent power source. Because of the limited size of a sensor node and of cost factors, it is not possible to use arbitrary powerful batteries as power sources. Thus, a major challenge is energy efficiency. This is crucial to prolong the lifetime of the sensor nodes. It may not be possible or may be too expensive to exchange the batteries. Alternative power sources such as solar cells, fuel cells, piezo cells, etc. may be used as additional power sources to extend the lifetime of the sensor nodes. However, the sole usage may not be possible in the majority of application scenarios [201]. As a result, security mechanisms must have low communication and computational overhead to save energy.

In certain scenarios, sensor nodes with more computing power and energy may be deployed. This may be possible for small-scale WSNs with a very limited number of (reusable) sensor nodes. In large-scale WSNs, the increasing costs make this unsuitable. However, heterogeneous WSNs may be deployed, i.e., the WSN consists of different types of sensor nodes. Some sensor nodes may have more resources than others and perform some special tasks. Algorithms and protocols can exploit this heterogeneity to achieve higher levels of security. But also algorithms and protocols designed for homogenous WSNs work in heterogeneous WSNs.

Operational Environment and Physical Topology. The application scenarios presented in Section 2.1.4 have shown that WSNs can be used in different operational environments. WSNs range from small scale WSNs for indoor usage over mobile WSNs to large scale WSNs which are deployed in unattended or even hostile environments. The physical topology of the WSN depends on the size of the network and the operational environment. In static WSNs, sensor nodes may be either specifically deployed, e.g., by manually placing the sensor nodes, or randomly deployed, e.g., via aerial scattering. In this area several important challenges arise.

If sensor nodes are deployed in unattended or even hostile environments, an adversary is able to physically attack the sensor nodes. He can steal them, simply destroy them, or compromise them. If a sensor node is compromised, an adversary can try to read data

out of the node, manipulate the soft- or hardware, or program the sensor node with its own code.

The environmental conditions in the operational environment may severely influence the functionality of the sensor nodes. For example, weather conditions, such as storm and lightning may disturb the operability of the WSN. As a result, sensor nodes may fail or the radio transmission is disturbed. Thus, protocols must adapt to these conditions.

If sensor nodes are mobile or are randomly deployed, the position of the sensor nodes may not be known in advance or even not during the whole lifetime of the network. Since sensor nodes may fail, the topology may also change. The manual maintenance of a WSN may not be possible in the majority of cases. Thus, the WSN should operate autonomously without relying on an infrastructure. Therefore, mechanisms for self-organization are required, i.e., sensor nodes collaborate to achieve their purposes. This must be supported by the applied security protocols.

Communication Channel. WSNs typically communicate using radio waves over a wireless channel. The communication channel can be either bidirectional, or unidirectional. Most homogenous WSN have a bidirectional channel. However, in heterogeneous WSNs some special nodes could be equipped with more powerful transmitters resulting in unidirectional communication channels. Likewise, unidirectional communication channels can exist between the sink with a powerful wireless transmitter and the sensor nodes.

The wireless channel can be easily accessed by an adversary. This enables an adversary to inject, replay, drop, or alter messages. As a result, the sink receives invalid data which could result in false alarms. By injecting or replaying a large amount of messages, the adversary could deplete the scarce energy resources of the sensor nodes and perform a Denial-of-Service (DoS) attack. Thus, a challenging problem is the protection against attacks on the wireless channel, e.g., by using appropriate authentication mechanisms.

Routing Topology. As already mentioned, messages in WSNs are routed along multiple hops. The multihop routing topology can have different forms: star topology (e.g., all sensor nodes send their data directly to the sink), hierarchical or tree organized (e.g., common sensor nodes send their data to aggregation nodes which process the data before sending the aggregate to the sink), or mesh topology (e.g., two sensor nodes exchange data using multiple routing paths). Star and hierarchical topology are commonly used in WSNs.

The mutual dependency of the sensor nodes is a challenging problem in WSNs. A sensor node requires other nodes to forward its messages to the destination. If neighboring sensor nodes fail or are selfish, messages may not reach their destination.

Routing Protocol. WSNs may use different routing protocols such as Directed Diffusion [117, 118], Rumor Routing [26], Geographic routing protocols (e.g., Geographic and energy aware routing (GEAR) [263], and Greedy Perimeter Stateless Routing (GPSR) [123]), Clustering-based routing (e.g., LEACH [102]), etc.

Obviously, the functionality of a WSN may be inhibited by disrupting the routing protocol [122]. We discuss attacks on routing protocols in section 2.2.4. The proposed protocols of this thesis are independent of the underlying routing mechanisms.

Type of Communication. There are three communication forms in WSNs: Many-to-one, One-to-many, and Local Communication. In *Many-to-one* communication, usually multiple sensor nodes send data such as measurements to the sink or to an aggregation node. *One-to-many* communication is used when the sink or a sensor node multicasts or broadcasts a query or some control information to several sensor nodes. *Local Communication* between neighboring sensor nodes can use broadcast, multicast, or unicast to send localized messages to discover and coordinate with each other.

Classical security protocols such as SSL/TLS [67] or IPSec [124] are designed for one-to-one communication. However, the dominant types of communication in WSNs are Many-to-one and One-to-many. This makes such protocols unsuitable for the usage in WSNs[4]. Thus, efficient protocols for this type of communication are required.

Data Aggregation. To reduce the amount of transmitted data, data aggregation is often used in WSNs. Thus, either the WSN performs data aggregation, or the whole data is sent to the sink which analyzes and processes the data.

A challenging problem is the loss of information during the aggregation. For example, after data aggregation, it is not possible to identify malicious nodes which have sent false measurements to manipulate the aggregation value.

Dynamic of the network. In general, WSNs are dynamic networks; sensor nodes can fail, the communication channel can be disrupted, or additional nodes may be added to the network. The result is a frequently changing network topology. Obviously, the dynamic in WSNs with mobile sensor nodes is even higher. The dynamic introduces many challenges.

Self-organization mechanisms are required to cope with dynamically changing topologies. Consequently, security protocols must be suitable for dynamically changing networks. For example, key management and authentication mechanisms have to support the addition of new sensor nodes.

Sink. Typically, a WSN has one sink. However, in certain scenarios multiple sinks are also possible. For example, in a sensor-actuator network, multiple distributed actuators may autonomously react on data received from the sensor nodes.

A challenge is that the sink may be a single-point-of-failure. An adversary jamming the region of the sink may be able to disrupt the whole functionality of the WSN, since the sink neither is able to send queries nor receive messages from the sensor nodes.

[4]Due to the extreme resource requirements, SSL/TLS and IPSec are anyway unsuitable for WSNs.

Localization. Sensor node localization is an important aspect for WSNs to determine the origin of the measurements. We distinguish between three types of localization mechanisms. Sensor nodes can determine their location by themselves (e.g., using a GPS receiver), in a distributed way (e.g., certain nodes are aware of their position and broadcast this information, whereupon their neighboring sensor nodes use this information to calculate their own position), or centralized (e.g., if sensor nodes are deployed at specific locations, it may be possible to pre-program the nodes with their location).

The challenging problem is a reliable and secure localization in an unreliable environment with an active adversary.

Heterogeneity. WSNs can either be homogenous, i.e., all sensor nodes are equipped with the same hardware, or heterogeneous, i.e., the sensor nodes are equipped with different hardware. For example, in hierarchical WSNs, some sensor nodes which perform special tasks may possess higher computational power and a higher transmission range.

The more challenging problem is the development of protocols for homogenous WSNs, consisting of severely resource constrained sensor nodes. Protocols for homogenous WSNs can be also used in heterogeneous networks. However, protocols for heterogeneous networks may exploit the extra resources of the nodes with additional resources. Therefore, protocols must support and exploit this heterogeneity to increase the overall efficiency of the WSN.

Node Identifiers. Even if it is assumed that in most WSNs each sensor node is assigned with a unique identifier (ID), it is also possible that sensor nodes do not possess any identifier or possess only a group identifier. The reason for this is that in WSNs with a large number of sensor nodes it may not be practicable that each single node possesses its own ID.

If sensor nodes are not assigned an ID, it may be difficult to identify and exclude maliciously acting sensor nodes.

Radiation. Usually, radio waves propagate in all directions. However, certain nodes can be equipped with directed antennas to communicate only with certain nodes or to increase the communication range.

Coping with adversaries which use directed antennas is a challenging problem. By increasing the communication range, an adversary can achieve inconsistent states in the network. Furthermore, if an adversary uses a directed antenna, neighboring sensor nodes may not be able to monitor the activities of this node for suspicious actions.

Activity Cycle. Sensor nodes may operate continuously. However, this is very energy consuming. To save energy, sensor nodes can use certain sleep modes to put the CPU in sleep mode or disable the wireless transceiver.

A general challenge is the management of sleep and activity cycles. Certain nodes may not be reachable during their sleep phase. In terms of security, the realization of a

monitoring system to detect malicious nodes is challenging, since nodes may not be able to monitor their neighbors if they are in a sleeping mode.

Size of the Network. The size of a WSN can vary from small scale networks to large scale networks.

The most challenging problem in this area is the scalability of the applied protocols. For example, key management must be applicable for large scale WSNs. It may not be practicable to pre-configure each sensor node with pairwise keys for all other nodes. Therefore, more sophisticated approaches are required.

Adaptability. WSNs can have a long operation time. Thus, it may be required to adapt the WSN to changed conditions. Therefore, WSNs have to support reconfiguration mechanisms or code updates using over-the-air (OTA) programming.

The adaption on new conditions or new code updates must be only performed by authorized entities. Therefore, appropriate authentication mechanisms are required.

Time Synchronization. It is often required to know the time when a measurement has been performed. Thus, sensor nodes require some timing device that must be synchronized between the sensor nodes. Therefore, sensor nodes may be equipped with an accurate clock, or central or distributed time synchronization mechanisms can be used.

If each sensor node is not equipped with an accurate clock, efficient, reliable, and secure time synchronization protocols are required which work in the unreliable environment, supporting multihop communication, and which are resistant against an active adversary.

Robustness. Besides attacks, natural events may have negative influence to the operation of a WSN. For example interferences may jam the wireless channel preventing the transmission of data. To address such events, WSNs must be robust, i.e., they must continue to operate even if abnormal events occur. Thus, robustness is a property which must be addressed in all WSNs; even in non-security-critical WSNs.

2.2 Security in Wireless Sensor Networks

This section gives an overview of security in WSNs. In Section 2.2.1, we introduce the three major properties which make the design of security protocols for WSNs a challenging task. Security protocols can address several security goals. In Section 2.2.2, we discuss the different security goals in the context of WSNs. The goal of an adversary is the violation of one or more of these security goals. The different classes of adversaries are discussed in Section 2.2.3. We reason that handling inside adversaries is the most challenging task in the majority of WSNs. In Section 2.2.4, we give an overview of attacks in WSNs and discuss them from the viewpoint of an outside as well as an inside adversary.

2.2.1 Properties with Major Impact on Security

In Section 2.1.5, we have given an overview of the special properties of WSNs. Some of these properties have rather little effect on the design of security protocols or are only relevant in more exotic application scenarios. However, three of these properties are elementary for nearly all WSN scenarios. These properties differ substantially from properties of classical computer systems and thus have serious influence on the design of security mechanisms and protocols. Therefore, these properties and the related challenges require special attention. These three properties are:

1. Resource Constraints,
2. Operational Environment, and
3. Wireless Multihop Communication.

It is commonly assumed that sensor nodes are highly resource constrained; e.g., the resources are comparable to the Berkeley MICA motes presented in Section 2.1.3. Thus, security protocols for WSNs must be executable on the available hardware and especially must be very efficient in terms of energy consumption and execution time.

The operational environment of most WSNs is assumed to be unattended or even hostile. Since sensor nodes are usually not assumed to be physically protected by some tamper-resistant hardware, an adversary is able to compromise sensor nodes. Thus, even if security mechanisms, such as node-based authentication, are deployed, an adversary is able to participate in the network since he has access to all data, e.g., cryptographic keys stored on the node. Thus, security protocols must be able to operate even if sensor nodes are compromised.

The wireless communication enables an adversary to eavesdrop, inject, drop, or alter messages or to perform DoS attacks by jamming the wireless channel. In contrast to most other wireless networks, the communication is performed in a multihop way. This introduces additional challenges. Compromised nodes may be part of a route, enabling them to modify forwarded messages, or a compromised node injects a large amount of false messages to drain the energy resources of all forwarding nodes.

2.2.2 Security Goals

Computer security relies on security goals which specify the requirements of a secure system. Within this thesis, we focus on the security goals of authenticity, integrity, confidentiality, and availability [81]. The security goals of non-repudiation and privacy [81] are not explicitly considered within this thesis.

In WSNs, authentication mechanisms are required to get confirmation about the identity of sensor nodes or the sink, and the origin of received messages. Integrity covers two aspects: the integrity of the communication in the WSN and the integrity of the system of a sensor node (or the sink). In the first case, mechanisms provide assurance that the received messages are valid, i.e., any unauthorized modification of data will be detected. This also includes the detection of replayed messages, i.e., the freshness of

messages. In the latter case, mechanisms provide assurance that the system of a sensor node is valid, i.e., hardware as well as software modifications will be detected. If sensitive data is transmitted in the WSN, confidentiality mechanisms are required to ensure that information is accessible only to authorized entities. Availability means that the WSN is in a functioning condition and is able to perform its tasks.

Since the security goal non-repudiation cannot be achieved in nearly any WSN, we do not consider it in this thesis. Non-repudiation means that an entity cannot ex post repudiate the validity of a performed action, e.g., a sensor node refuses that it has sent a certain message. To achieve non-repudiation, digital signatures are required, and it must be ensured that no unauthorized entity has access to private keys. Since public key cryptography is usually too resource intensive and sensor nodes can be compromised, this security goal cannot be achieved. The security goal privacy is required if a WSN is used to monitor personal data, e.g., a WSN deployed in health care. Then, it must be ensured that either the monitored entity (in this case, the human) is not identifiable, or the personal data of this entity is not accessible or cannot be associated to the entity. Since we do not explicitly examine privacy-relevant applications and the privacy topic itself suffices for a sole thesis, this security goal is also not considered within this thesis.

In addition to the above mentioned security goals, certain application scenarios may have additional security requirements. For example, some real time monitoring applications require the timeliness of data, i.e., data arrives in time at the destination.

Which security goals have to be achieved to secure a WSN depends on the application scenario. For example, consider a WSN which monitors a forest to detect forest fires. Among the functional requirement actuality, mainly the authenticity and integrity of the exchanged messages is required to guarantee that the messages are indeed from sensor nodes of the deployed WSN and that the messages have not been modified. The confidentiality of these messages might be of minor priority, since no sensitive data is transmitted. In contrast, however, the WSNs deployed in human health care require mechanisms that guarantee the confidentiality and privacy of the measured data.

An adversary can try to attack a WSN, so that one or more of the security goals are violated. Adversaries can be categorized in different classes which we describe in the next section.

2.2.3 Classes of Adversaries

An adversary can attack a WSN in different ways, depending on the effort for the adversary, the knowledge of the adversary about the WSN, and the potential damage the adversary can cause. In this thesis, we basically use the following two general distinctions of types of adversaries.

Mote-class and Laptop-class Adversary

The first distinction of adversaries is made on the basis of the available resources. We distinguish between a *mote-class* and a *laptop-class* adversary [122].

- A *mote-class adversary* has access to one or more sensor nodes with the same or similar capabilities like the sensor nodes deployed in the network.
- A *laptop-class adversary*[5] may have access to more powerful devices such as laptops with more resources, e.g., greater battery power, more powerful CPU, greater memory space, a high power radio transmitter, or a sensitive antenna.

Thus, an adversary with laptop class devices has the advantage of being able to perform more sophisticated attacks than an adversary with ordinary sensor nodes only. For example, an ordinary sensor node might only be able to jam the radio link in its immediate vicinity, while a laptop-class adversary can jam a much larger area. The laptop-class adversary is also able to eavesdrop on messages from a much greater distance than a mote-class adversary.

Outside and Inside Adversary

The second distinction is made based on the knowledge and privileges of the adversary. We distinguish between an *outside adversary* and an *inside adversary* [122].

- An *outside adversary* has no special access to sensor nodes of the WSN, i.e., he has no access to data stored on any sensor node and uses only his own devices to perform attacks.
- An *inside adversary* has access to data stored on one or more sensor nodes and uses this data to perform subsequent attacks. These insider attacks can be launched from either compromised sensor nodes running malicious code or from laptop-class adversaries using stolen key material, code, and data from legitimate nodes.

An outside adversary can be prevented from performing most outsider attacks by applying cryptographic mechanisms. For example, authentication mechanisms prevent an outside adversary from injecting false messages, and encryption mechanisms prevent an adversary from eavesdropping. Therefore, a single network-wide key which is shared by all sensor nodes may be sufficient. Alternatively, sensor nodes establish pairwise keys, e.g., using schemes proposed in [84, 43, 116, 155, 76, 152, 270], and apply mechanisms to ensure authenticity, integrity, and/or confidentiality, e.g., mechanisms proposed in [121, 181]. However, an outside adversary is still able to perform jamming attacks which is a general problem in all wireless communication systems.

The adversary model of an outside adversary is similar to the adversary described in the Dolev-Yao model [70] which is assumed for classical security protocols such as SSL/TLS [67] or IPSec [124]. This adversary is often referred to as Man-in-the-Middle (MitM) adversary. The Dolev-Yao model assumes an adversary who is able to overhear, intercept, and generate any message and is only limited by the cryptographic constraints, i.e., he is only able to encrypt/decrypt messages if he possesses the proper keys. Metaphorically speaking, the adversary sits between two communicating entities and

[5] The case where the adversary modifies the hardware of the compromised sensor node, e.g., by enhancing the memory of the node, can be seen as a laptop-class attack.

intercepts all messages exchanged between these two entities, modifies them or replaces them through new ones, and sends them to the recipient.

However, the Dolev-Yao model is insufficient for WSNs for the following reasons. First, the communication range is limited which may prevent the overhearing, intercepting, and generation of any message. This may even be true for a laptop-class adversary. This property can be exploited by security protocols. For example, sensor nodes detect that a certain region of the WSN is jammed by an adversary and route messages around this region.

The major reason why the Dolev-Yao model is insufficient for WSNs is the possibility of sensor node compromise. Node Compromise is a realistic threat since sensor nodes of WSNs may be deployed in unattended or even hostile environments and usually are not tamper-resistant. After a node compromise, an adversary has knowledge of all data, such as the program code and cryptographic keys, stored on the node. Using this data, an adversary can perform subsequent attacks as an insider. For example, using the cryptographic keys he is able to decrypt and encrypt messages. By reprogramming a sensor node, an adversary is able to launch more sophisticated attacks, such as the injection of false messages to cause false alarms. Furthermore, several compromised sensor nodes may collude and exchange data, e.g., they may exchange keys using an out-of-band channel and then collaboratively perform an attack. This shows that an inside adversary can cause much more damage than an outside adversary. However, the effort for an adversary to perform insider attacks is much higher since it requires the compromise of one or more sensor nodes. In the following, we describe how an adversary can compromise sensor nodes in order to perform insider attacks.

A node compromise can be performed either by physically accessing the sensor node or over the wireless channel. Attacks based on physical access can range from simply connecting a laptop to a programming interface of the sensor node to sophisticated side channel attacks if in certain scenarios (cf. Chapter 7) sensor nodes are equipped with tamper-resistant hardware. Side channel attacks are known from smartcards where an adversary applies techniques such as manual microprobing, laser cutting, focused ion-beam manipulation, glitch attacks, or power analysis, to extract protected software and data from the smartcard processor [81, 9, 10, 128]. To compromise a sensor node over the wireless channel, an adversary can exploit weaknesses in software implementation or in the applied protocols. If sensor nodes support over-the-air (OTA) programming and the code update mechanism is not protected by authentication mechanisms, an adversary may be able to program sensor nodes with its own malicious code. (Security) protocols which are either incorrectly implemented or have general flaws in protocol design, may be also exploited by an adversary. This might enable the adversary to access secret data by sending messages that are non-compliant to the protocol. Likewise, flaws in software implementation of a sensor node may enable buffer overflow (BO) attacks [81]. By sending manipulated packets, the adversary may be able to cause a BO and get access to secret data stored on the node. However, the typical way to compromise a sensor node is by physically accessing the node.

Based on the intention of the adversary, we distinguish between two types of compromise:

1. *Read-only Compromise*

2. *Read-and-Write Compromise.*

In a read-only compromise, the adversary reads out stored secrets, such as cryptographic keys, from the memory of the captured sensor node. Furthermore, the adversary may read out the program code to analyze the implemented protocols to be able to use the obtained keys. Afterwards the adversary can only violate the confidentiality by performing passive attacks such as eavesdropping or traffic analysis (cf. Section 2.2.4).

A read-and-write compromise enables an adversary to perform active insider attacks such as false data injection. Therefore, the adversary not only reads out data but also modifies some existing protocols or writes new malicious code to a sensor node. Since this type of compromise results in a difference in behavior, it may be more easily detected using for example an Intrusion Detection System (IDS) [28].

In both cases, the adversary can perform insider attacks either as a *laptop-class* or as a *mote-class* attack. After a read-only compromise, the adversary may use the obtained keys on a laptop with a much more powerful wireless receiver unit to eavesdrop on sensor nodes which are farther away if for example a single network-wide key is used. However, if the obtained key is only used to encrypt local messages, the attack surface cannot be expanded by using a laptop class device. A read-and-write compromise however, can be exacerbated by using a laptop class device. Using a compromised key, modified code, and a laptop with higher transmission range, an adversary might be able to inject false messages in different areas of the WSN.

Read and read-and-write compromise can be either performed as on-site or off-site compromise. In an on-site compromise, the adversary performs the compromise directly in the deployment area without moving the sensor nodes. An adversary usually performs an off-site compromise if the compromise requires more time and equipment. Therefore, the adversary moves the sensor node to its lab or a safe place to perform the compromise. Such a movement may be detected and immediately reported by the moved or a neighboring sensor node. To cover this, an adversary may perform a jamming attack while moving sensor nodes to prevent the successful transmission of alarm messages.

In addition to the sole compromise of sensor nodes of the WSN, an adversary can also perform a *Node Fabrication Attack* [178, 51, 48] to fabricate new sensor nodes. An adversary uses the keying material stored on compromised sensor nodes to fabricate his own new sensors which are added to the WSN to perform attacks. We distinguish between two types of node fabrication attacks. The first type is known as *Node Clone Attack* or *Node Replication Attack*. After a read-only compromise, an adversary uses the acquired data to generate identical sensor nodes with the same keys, data, and code as the compromised sensor node. In the second type, an adversary actively fabricates sensor nodes that are different from the compromised one. This attack is based on the fact that current key management schemes (e.g., the random key predistribution and pairwise key predistribution schemes [84, 43, 116, 155, 76, 215]) store a key pool on the

sensor nodes and not only the specific keys for this node. Using some of these keys in combination with either an already used node ID or a newly generated valid node ID, an adversary can fabricate nodes. The adversary may guess new valid IDs based on the analysis of the data and code stored on the compromised sensor nodes. The fabricated nodes are now able to participate in the WSN using the ID and the keys from the key pool. This is possible since most sensor nodes accept new sensor nodes as valid new nodes if they can establish pairwise keys with them. The fabricated nodes are able to establish these keys by using their key pool.

The subsequent insider attacks can be then performed at different layers. On the physical layer, an adversary may simply jam the wireless channel; however, this attack is also able as an outsider. Attacks at the medium access control layer may have the goal to starve sensor nodes by manipulating the random backoff mechanism [35]. On the network layer, an adversary may drop, replay, or alter routing packets. Attacks on the application layer include the injection of false data to cause false alarms, manipulation of time synchronization or localization protocols, etc. In the next section, we present the different types of outsider and insider attacks in more detail.

2.2.4 Types of Outsider and Insider Attacks

In this section, we give a wide overview of types of outsider and insider attacks in WSNs. In contrast to previously published work [247, 191, 122, 242, 206, 253, 47, 274], we try to give a more comprehensive overview and also discuss the attacks from the point of view of an inside adversary. For this, we consolidate and extend previous work on, amongst others, routing attacks [122], Denial-of-Service (DoS) attacks [247, 191], various attacks [242, 206, 253, 47, 274] or new types of attacks [130, 131].

First, we give a brief description of the attack and which security goals (cf. Section 2.2.2) are violated. Next, we present possible defenses against an outside adversary. After that, we discuss the impact an inside adversary can cause by performing the respective attack and which issues should be considered.

We present the attacks following the order of the protocol stack (cf. Section 2.1.2). We start by presenting generic attacks that can be performed at different layers. After that we discuss the attacks (and variants of the general attacks) at physical, link, network, transport, and application layer.

Before we describe the different attacks, we first give a short definition of outsider attack and insider attack based on the definition of outside and inside adversary presented in Section 2.2.3.

- We define *insider attacks* as all attacks based on the usage of information, e.g., keying material or code, stored on the compromised sensor node. This enables the adversary to appear as a legitimate node in the network. This differs from the classical view [249], where an inside adversary may be a person with extensive knowledge of and privileged access to the target system and is able to attack inside a system's perimeter defense, e.g., a former employee of a company.

- In contrast, *outsider attacks* are defined as attacks of an adversary who has not compromised any sensor node and is not in possession of any information stored on a sensor node. An adversary can only get information by analyzing the data sent over the wireless channel.

Generic Attacks

In the following, we describe generic attacks that can be performed at different layers. We do not list these attacks again in the respective sections of the individual layers, except variants are worth mentioning.

In a *Spoofing Attack* an adversary successfully masquerades as another sensor node. This type of attack can influence the link, network, and application layer. For example, at the link layer, an adversary can spoof the Media Access Control Address (MAC-Address) of the link layer protocol, e.g., the IEEE 802.15.4 MAC-Address. By claiming a false identity, an adversary can try to disrupt the routing at the network layer. Likewise, the adversary can disrupt the correct operation of applications such as data aggregation. A serious variant of the spoofing attack is the Sybil attack (see Attacks at the network layer) where an adversary forges multiple identities. An outside adversary can be impeded from performing this attack by applying authentication mechanisms such as the link layer authentication mechanisms of IEEE 802.15.4 [205]. An inside adversary who has performed a read-and-write compromise, however, is able to masquerade as the compromised node and can inject malicious data. When considering insider attacks, the applied key management is an important issue. For example, if only one shared network-wide key is used for the whole WSN, an adversary who has compromised only a single node may be able to masquerade as any sensor node of the WSN. Thus, the key management should be robust against node compromise where an adversary cannot impersonate as an uncompromised sensor node.

In a *Data Alteration Attack* an adversary deliberately alters bits, frames, packets, or application data, depending on the layer where the attack is performed. To be able to detect such attacks from an outside adversary, integrity mechanisms such as digital signatures or MACs can be applied. However, an inside adversary can circumvent these mechanisms if he has compromised the corresponding keys. Thus, as with the spoofing attack, key management also plays an important role. In addition, the resilience to node compromise also depends on the communication endpoints of the integrity mechanisms. For example, if the integrity mechanism of a multihop communication is only applied on a link layer level, a compromised en-route node is able to perform a data alteration attack. In contrast, if the integrity mechanism is applied on an end-to-end connection, any modification of an en-route node will be detected by the receiver.

In a *Replay Attack*, an adversary eavesdrops on a legitimate message sent between two sensor nodes and replays it at a later time. At the link layer an adversary can replay data frames, at the network layer data or routing packets, and at the application layer report data. Defenses against an outside adversary are sequence numbers or timestamps in combination with an authentication scheme. For example, if the sender assigns a monotonically increasing sequence number to each packet and a MAC is used to ensure

its authenticity and integrity, then the receiver rejects packets with a smaller sequence number that it has already seen or where the MAC is invalid. This also prevents an inside adversary from replaying messages. However, if an inside adversary is in possession of the required keys, he can alter the sequence number to a higher number. As in the attacks described before, the key management issues also apply.

A *Denial-of-Service (DoS) Attack* is an attack against the availability where an adversary attempts to make the WSN or parts of it unavailable for its intended usage. The functionality of the WSN can be disrupted temporarily or indefinitely. DoS attacks can be performed at all protocol layers. At the physical layer, an adversary can jam the wireless channel or simply destroy sensor nodes, thereby rendering them permanently non-operational. For this purpose, the adversary can perform *Search-based Physical Attacks* [241, 98] where the adversary walks through the WSN using signal detecting equipment to locate active sensors, and then destroys them. At the link layer, an adversary can for example perform a denial-of-sleep attack [27], which prevents the radio from going into sleep mode exhausting the node's energy resources. At the network layer, an adversary can exploit the multihop communication by not forwarding messages or try to disrupt the routing. At the application layer DoS attacks can target for example localization and time synchronization protocols. Defenses against an outside adversary and the impact of insiders are highly dependent on the type of DoS attack and are discussed below.

In an *Eavesdropping Attack*, an adversary violates the confidentiality of data by overhearing the transmitted bits on the wireless channel [250], intercepting messages during multihop communication, or wiretapping directly at a sensor node. Using knowledge of the different layers, an adversary can analyze this data. For example, knowing the applied routing protocols at the network layer enables an adversary to identify and analyze data packets. Likewise, at the application layer, an adversary can identify and analyze application specific data such as report data. An outside adversary can be prevented from eavesdropping by applying encryption mechanisms, e.g., link layer encryption [121]. An inside adversary who has performed a read-only compromise of one or more sensor nodes, gets access to the keys stored on these nodes. Using these keys, he can decrypt data which is encrypted with these keys. Data which is encrypted with different keys, however, cannot be decrypted. Thus, the impact of node compromise depends on the applied key management. If, for example, a single network-wide key is used, an adversary needs only to compromise a single node to be able to eavesdrop on all messages.

Table 2.2 summarizes the generic attacks and shows the violated security goals and which defenses against an outside adversary can be applied.

Attack	**Violated Security Goals**	**Defenses against outsider**
Spoofing	Authenticity	Authentication
Data Alteration	Integrity	Digital signatures, MACs
Replay	Integrity	Sequence numbers, Timestamps
Denial-of-Service	Availability	Depends on attack characteristics
Eavesdropping	Confidentiality	Encryption

Table 2.2: Generic Attacks

Attacks at the Physical Layer

At the physical layer, we distinguish between attacks against the wireless channel and attacks directed at the hardware of a sensor node.

The wireless channel is susceptible to a *Jamming Attack* [247] where an adversary deliberately generates interferences to make sensor nodes unable to communicate. This attack can be performed either by an outside or inside adversary and can have serious consequences by influencing the availability of the wireless channel. In WSNs, jamming attacks cannot be generally prevented. Using mechanisms such as spread-spectrum or frequency hopping may prevent only limited adversaries from performing jamming attacks. In addition, these technologies are usually not feasible for severely resource constrained sensor nodes. One approach is to try to detect and map jammed regions, and to re-route messages around the jammed region [247, 248]. Sensor nodes in the jammed area may go into sleep mode to save energy and sporadically wake up to check if the adversary is still active. Further mitigation strategies are presented in [255, 256]. Jamming attacks must be considered in the design of various security protocols. For example, the functionality and detection rate of an IDS to detect compromised sensor nodes may be significantly influenced by successful jamming attacks.

A direct attack of the sensor node's hardware is possible if an adversary has physical access to sensor nodes. This enables the *Physical Destruction* of sensor nodes. As a result, the destroyed sensor nodes are not available anymore. If a significant number of nodes are destroyed, the availability of the whole WSN may be at stake. This type of attack can be performed either by an outside or an inside adversary. One countermeasure may be the hiding of the sensor nodes or the deployment in protected areas. This would also prevent sensor nodes from becoming compromised; assuming that a compromise over the wireless channel, e.g., by exploiting weaknesses in over-the-air (OTA) programming, is not possible. As a result, insider attacks would be prevented. However, in most WSN application scenarios this is not possible. In this case, additional mechanisms must be taken into account, e.g., when an intruder is detected, mechanisms independent from the WSN are invoked to cope with him.

An adversary can perform a *Tampering Attack* [247] to compromise a sensor node. The type of compromise can either be read-only or read-and-write. The adversary may be able to violate all security goals. Defenses are similar to those discussed for physical destruction and include secure locations, and detection and reaction to intruders which try to tamper with nodes. In the latter case, sensor nodes can, for example, delete their memory to prevent an adversary from accessing the data stored on the node. An alternative approach is the use of tamper-resistant hardware [9] to prevent unauthorized access to a sensor node. In this thesis, we propose a tradeoff approach where tamper-resistant hardware is only used to protect some special sensor nodes (cf. Chapter 7). However, these defenses are not generally applicable since either the WSN cannot be deployed in secure locations or cost-containments forbid the use of additional hardware. In this case, the WSN should provide mechanisms that are resilient to compromised nodes and an inside adversary, i.e., the WSN can tolerate a certain number of compromised nodes or sensor nodes are able to detect and exclude compromised nodes from the WSN.

The detection can be either based on misbehavior detection or on attestation techniques where the code running on the sensor node is validated.

Table 2.3 summarizes the attacks at the physical layer and shows the violated security goals and which defenses against an outside adversary can be applied.

Attack	Violated Security Goals	Defenses against Outsider
Jamming	Availability (communication)	Only mitigation (e.g., spread spectrum, region mapping)
Physical destruction	Availability (sensor node)	Secure locations, detection and reaction to intruders
Tampering	Possibly all	Secure locations, detection and reaction to intruders, tamper-resistant hardware

Table 2.3: Attacks at the physical layer

Attacks at the Link Layer

The link layer provides channel arbitration for neighbor-to-neighbor communication. Protocols mostly require cooperation between sensor nodes making them particularly vulnerable to DoS attacks.

A *Collision Attack* [247, 242, 274] is a type of DoS attack similar to the jamming attack. It can be performed by an outside or inside adversary. When a legitimate sensor node sends a message, the adversary induces a collision by sending his own signal. A collision in only one byte is sufficient to create a CRC error and to invalidate the message. The advantage of the collision attack compared to a jamming attack is the lower energy consumption for the adversary and the difficulty of detecting the attack since the only evidence are incorrect messages. If the collision attack is directed at ACK control messages, it could induce costly exponential backoff in some Media Access Control protocols. The countermeasures against jamming attacks can also be used to mitigate the impact of collision attacks. In addition, error-correcting codes can be applied. However, an inside adversary who has performed a read-and-write compromise can analyze the applied code and can still corrupt more data than can be corrected. Error-correcting codes also induce an expensive communication overheard and require additional processing. Another defense is collision detection. However, an inside adversary may intentionally and repeatedly deny access to the channel, expending much less energy than in fulltime jamming. Currently, there is no completely effective defense known [247].

In an *Exhaustion Attack* [247] an outside or inside adversary exploits some of the properties of the link layer to waste the scarce energy resources of the sensor nodes. For example, an adversary repeatedly generates collisions to cause resource exhaustion. A naive link-layer implementation may continuously attempt to retransmit the corrupted packets and thus the energy reserves of the transmitting node and those surrounding it will be quickly depleted. Possible defenses are rate limits to ignore excessive requests. Furthermore, time-division multiplexing can be used, where each node is allotted a time

slot in which it can transmit. This eliminates the need for arbitration of each frame and can solve the indefinite postponement problem in a backoff algorithm. However, it is still susceptible to collisions.

Another form of exhaustion is an *Interrogation Attack* [247, 191] where an adversary exploits the two-way request-to-send/clear-to-send (RTS/CTS) handshake. The adversary can either use a powerful laptop-class device or sacrifice a mote-class device to exhaust the victim node's resources by constantly sending RTS messages to elicit CTS responses from this node. Applying strong link-layer authentication and replay protection can mitigate the impact an outside adversary can cause since the victim node would not respond to RTS messages. However, the receiving of the bogus RTS messages still consumes energy and network bandwidth. An inside adversary is still able to perform the attack to its full extent.

A *Sleep Deprivation Attack* [221, 220], also known as a *Denial of Sleep Attack* [190, 27, 184], is another form of exhaustion attack. The goal of an adversary is to prevent sensor nodes from entering sleep mode, thereby depleting their energy resources. A clever denial of sleep attack that keeps the transceiver of a sensor node on would drain the batteries in only a few days (assuming nodes described in Section 2.1.3). Since the transceiver consumes more energy than the other components of a sensor node (cf. Section 2.3), Medium Access Control Protocols are a natural focus for denial of sleep attacks. Protocols such as Sensor MAC (S-MAC) [262], Berkeley MAC (B-MAC) [186], or Timeout MAC (T-MAC) [234] are all susceptible to this type of attack [190, 191]. For example, if S-MAC is used, an adversary can repeatedly send malicious synchronization packets causing the victim node to stay awake to maintain synchronization with the adversary. Similar attacks are also possible if T-MAC or B-MAC is used [191]. Mechanisms against an outside adversary include link layer authentication, replay protection, jamming identification and mitigation, and broadcast attack protection [190, 191]. An inside adversary who has performed a read-and-write compromise is able to perform a successful authentication. To cope with an insider, detection and mitigation mechanisms or tamper-resistance can be applied.

Unfairness [247] is a weaker DoS attack, where an outside or inside adversary intermittently performs collision or exhaustion attacks to abuse a cooperative Medium Access Control layer priority scheme to cause unfairness. This attack could result in service degradation, e.g., in a real-time Medium Access Control protocol, sensor nodes miss their deadlines. A possible defense is the use of small frames so that an individual node can capture the channel for only a short time. However, this technique often reduces efficiency and an adversary can still cheat when vying for access, e.g., by instantly retransmitting instead of randomly delaying.

A *Link-layer Jamming Attack* [140] is similar to a jamming attack at the physical layer but an adversary exploits the knowledge of the link layer to perform the attack more energy efficiently. Although the attack is more difficult to perform than a jamming attack at the physical layer, there exist only mitigation defenses such as link layer encryption, use of spread spectrum and TDMA based protocols. Link layer protocol specific defenses are also described in [140]. Since the encryption provides only weak protection, the impact an outsider can cause is nearly similar to that of an inside adversary.

Table 2.4 summarizes the attacks at the link layer and shows the violated security goals and which defenses against an outside adversary can be applied.

Attack	Violated Security Goals	Defenses against Outsider
Collision	Availability (communication, battery), integrity (messages)	Error-correcting codes, Collision detection
Exhaustion	Availability (battery)	Rate limitation
Interrogation	Availability (battery)	Authentication, replay protection
Sleep depravation	Availability (battery)	Authentication, replay protection, detect and sleep, broadcast attack detection
Unfairness	Availability (communication)	Small frames
Link Layer Jamming	Availability (communication)	Encryption, spread spectrum, TDMA

Table 2.4: Attacks at the link layer.

Attacks at the Network Layer

There are several network layer attacks in which an adversary can disturb the routing functionality.

An adversary can directly attack the routing protocol by targeting the routing information exchanged between sensor nodes. By *Spoofing, altering, or replaying routing information* [122] an adversary may create routing loops, attract or repel network traffic, extend or shorten source routes, increase end-to-end latency, partition the network, etc. Defenses against an outside adversary are authentication, replay protection mechanisms, and mechanisms to verify the integrity of messages, e.g., at the link layer. However, an inside adversary who has performed a read-and-write compromise, can perform these attacks by reprogramming the compromised sensor nodes using the acquired keys. Thus, routing protocols must provide resilience to compromised nodes. One general approach is to use multiple path forwarding [90, 59, 144, 145], hoping that at least on path will be unaffected by the adversary. The INSENS routing protocol [60, 61, 65] uses this approach to provide routing between sensor nodes and the sink. Another promising approach to achieving resilience against compromised nodes is the use of geographic routing protocols where sensor nodes know their own location, e.g., as proposed by Wood et al. in [246]. Alternatively, a combination of prevention, detection and recovery, and resilience techniques can be used as proposed in [177].

The multihop communication in WSNs facilitates DoS attacks at the network layer. In a *Selective Forwarding Attack*[6] [122], compromised sensor nodes on the routing path refuse to forward certain messages or simply drop them, ensuring that they are not propagated any further. To reduce the possibility of detection, not all packets are dropped. The adversary selectively forwards packets, e.g., only packets of certain nodes

[6]Similar to the neglect and greed attack described in [247].

are dropped and the remaining traffic is forwarded. If the adversary is able to make itself part of many routes, he can increase the potential damage he can cause. A specific form of this attack is the *Black Hole Attack*, where the adversary simply drops or refuses all packets. However, this attack runs the risk that neighboring sensor nodes will decide that this node has failed or is compromised, and choose another route. Defenses against an outside adversary are link layer authentication and encryption since an adversary is prevented from joining the WSN and cannot selectively drop messages based on their content. An inside adversary who has performed a read-and-write compromise, however, is able to perform this attack at the compromised nodes. Mechanisms against an insider include watchdog mechanisms where neighboring nodes monitor each other, acknowledgement-based schemes [252], and the above mentioned multiple path forwarding of messages. The latter requires the adversary at least to compromise one sensor node on each path to ensure that the message does not reach its destination.

In a *Sinkhole Attack* [122], the adversary's intention is to lure traffic from a particular area through a compromised node, creating a sinkhole with the adversary at the center. Wormhole attacks may be used to generate such a sinkhole. Since routing protocols usually prefer low latency links, the route provided by the wormhole is chosen. Alternatively, an adversary could spoof or replay advertisements for an extremely high-quality route to the sink. Sinkhole attacks may be used as a preparation for other attacks, e.g., selective forwarding attacks. Applying link layer authentication prevents an outside adversary from performing this attack. However, an inside adversary who has performed a read-and-write compromise is able to perform a successful authentication and thus can perform the attack. To cope with this attack, routing protocols are required where this attack is ineffective, e.g., geographic routing protocols such as the above mentioned protocol by Wood et al. [246].

In a *Sybil Attack* [73], a sensor node forges multiple identities. The malicious sensor node behaves as if it is a large number of nodes. Therefore, the node can, for example, impersonate other nodes or simply claim false identities. The Sybil attack may have serious impact on fault tolerance schemes such as distributed storage, dispersity and multipath routing, and topology maintenance [122]. Furthermore, data aggregation, voting, fair resource allocation, and misbehavior detection are susceptible to successful Sybil attacks [170]. Karlof and Wagner point out the threats of the Sybil attack against routing protocols [122]. For example, in geographic routing [4, 123, 263, 25] a malicious node could appear in more than one place at once. In general, applying link layer authentication prevents an outside adversary from performing this attack. After performing a read-and-write compromise, an inside adversary can perform a Sybil attack. However, the adversary can only forge identities of sensor nodes where he possesses the respective keys. The possibilities of an inside adversary can be further restricted by maintaining a list of trusted neighbors, e.g., as in [177]. Further defenses are radio resource testing, verification of key sets for random key predistribution, registration and position verification [170].

In a *Wormhole Attack* [112, 122], an adversary tunnels messages from one part of the network to another using, for example, two laptop class devices with an out-of-band channel providing a high transmission range. Thus, the wormhole provides a lower

latency than the alternatively-used multihop communication. This can be exploited to perform additional attacks, such as sinkhole attacks. Link layer authentication is not sufficient for preventing an outside adversary from performing this type of attack. Although he is prevented from joining the WSN, he is still able to tunnel packets sent by legitimate nodes in one part of the network to legitimate nodes in another part to convince them that they are neighbors or send overheard broadcast packets with a laptop-class device with sufficient power to every node in the WSN. Hu et al. introduced packet leashes to detect and defend against wormhole attacks [112]. Two types of leashes have been introduced: geographical and temporal leashes. Packet leashes and mechanisms using secure positioning, synchronized clocks, or directed antennas [109, 240, 264, 32] can be used to defend against outside and inside adversaries. Similar to the sinkhole attack, new geographic routing protocols are also a promising approach for coping with this attack.

A *HELLO Flood Attack* [122] can be mounted against many routing protocols which use HELLO packets to inform one-hop neighbors of their presence. An adversary can use a laptop-class device with high transmission power to send HELLO packets to nodes in another area to trick these nodes into believing that they are neighbors of the malicious node. If the adversary falsely broadcasts a high quality route to the base station, all of the tricked nodes will attempt to transmit to the attacking node. Since many nodes are outside communication range, many messages are lost. Alternatively, the adversary can replay recorded HELLO packets from other sensor nodes to confuse the network. Defenses against an outside adversary are authentication and the verification of bidirectional links. To perform this attack as an insider, the adversary must authenticate itself to all victim nodes. If the authentication scheme involves the base station, an adversary claiming to be a neighbor of an unusually large number of nodes would raise an alarm [122]. In general, routing protocols such as [246], which do not rely on HELLO messages or do not keep routing tables, are inherently resistant to this attack.

An *Acknowledgement Spoofing Attack* [122] can be mounted against routing protocols which rely on acknowledgements. An adversary can spoof acknowledgements of overheard packets destined for neighboring nodes in order to provide false information to those nodes. For example, to make the sender believe that a weak link is strong or a dead node is alive. This attack can be used as preparation for a selective forwarding attack. Authentication is a defense against an outside adversary. Even an inside adversary is severely restricted since he can only send acknowledgements for nodes where he possesses the cryptographic key.

In addition to the above mentioned attacks, there exist *Routing protocol specific attacks* [122]. An adversary can attract or repel traffic flows, increase end-to-end latency, partition the network, create routing loops, extend or shorten source routes, etc. Details to these attacks can be found for example in [122].

In a *Traffic Analysis Attack* [63] or *Homing Attack* [247], an adversary analyzes packet traffic to deduce the location of critical resources such as the sink or cluster heads. The adversary may then physically attack them or jam these regions to make the WSN useless. One defense against an outside adversary is the encryption of both message header and payload. However, it may still be sufficient for an adversary to analyze the

traffic volume in various parts of the WSN to identify the location of the sink or cluster heads. To address this issue, Deng et al. [62] propose the use of dummy packets to normalize the traffic with the disadvantage of wasted energy. An inside adversary who has performed a read-only compromise is able to decrypt and inspect the packets where he possesses the keys. However, if a resilient key management scheme is deployed, an adversary must compromise a large number of nodes to be successful. This is especially difficult in large scale WSNs.

Table 2.5 summarizes the attacks at the network layer and shows the violated security goals and which defenses against an outside adversary can be applied.

Attack	**Violated Security Goals**	**Defenses against outsider**
Spoofing, altering, or replaying routing information	Authenticity and Integrity (routing information), Availability (communication)	Authentication, replay protection
Selective forwarding	Availability (communication)	Authentication, multipath routing
Black hole	Availability (communication)	Authentication, multipath routing
Sinkhole	Authenticity and Integrity (routing information)	Authentication
Sybil	Authenticity (node identifier)	Authentication
Wormhole	Authenticity and Integrity (routing information)	Packet leashes
HELLO floods	Authenticity and Integrity (routing information)	Authentication, verification of bidirectional links
Acknowledgement spoofing	Authenticity (routing information)	Authentication
Routing protocol specific attacks	Depends on attack type	Depends on attack type
Traffic Analysis	Confidentiality (message flow, message content)	Encryption, dummy packets

Table 2.5: Attacks at the network layer

Attacks at the Transport Layer

The transport layer manages end-to-end connections. In classical computer networks the User Datagram Protocol (UDP) or the Transmission Control Protocol (TCP) are used. However, these protocols have a high resource overhead and are, similar to the Internet Protocol (IP) at the network layer, (usually) not suitable for WSNs. Because of the resource constraints, WSNs often do not explicitly use transport layer protocols although some have been proposed, e.g., in [222] and [237]. In the following, we describe the two attacks which have been identified at the transport layer [247, 191].

A *Flooding Attack* [247, 191] causes memory exhaustion by exploiting the fact that transport layer protocols maintain state at either end of a connection. An adversary

repeatedly sends new connection requests until the resources of the victim node are exhausted or reach a maximum limit. A defense against an outside and an inside adversary are client puzzles. The idea is that a connecting client must demonstrate its commitment to the connection by solving a computationally intensive puzzle. Assuming that an adversary does not have infinite resources, it will be impossible for him to create new connections fast enough to exhaust the resources of the victim node. A disadvantage is the higher energy consumption for legitimate sensor nodes, but it is less costly than wasting wireless transmissions by flooding.

A *Desynchronization Attack* [247, 191] can disrupt an existing connection between two endpoints. An adversary transmits forged packets with bogus sequence numbers or control flags to degrade or prevent the exchange of data. Also sensor nodes may waste energy in an endless synchronization-recovery protocol. Authentication of all packets, including all control fields in the transport header can be used as a defense against an outside adversary. An inside adversary's ability to desynchronize the connection of two uncompromised nodes, depends on the applied key management scheme. However, since the compromised node is certainly able to authenticate itself to some neighboring nodes (using compromised pairwise keys), the adversary can use the desynchronization attack to waste the energy of these nodes.

Table 2.6 summarizes the attacks at the transport layer and shows the violated security goals and which defenses against an outside adversary can be applied.

Attack	**Violated Security Goals**	**Defenses against outsider**
Flooding	Availability (communication, memory)	Client puzzles
Desynchronization	Availability (communication, battery)	Authentication

Table 2.6: Attacks at the transport layer

Attacks at the Application Layer

At the application layer, a plethora of different types of applications can be implemented. Thus, it is virtually impossible to identify and list all attacks in this section. However, there exist certain attacks at the application layer which are a threat for the majority of WSNs. In the following, we describe some more general attacks and attacks on typical WSN applications.

In a *False Data Injection (FDI) Attack* [261], an adversary generates and sends reports for non-existing events. This can result, for example, in false alarms or a data aggregation scheme miscalculates aggregate values. Authentication mechanisms prevent an outside adversary from performing this attack. An inside adversary who has performed a read-and-write compromise, however, can use the compromised key(s) to generate valid report messages. To cope with an inside adversary, usually the redundancy of WSNs is exploited. Instead of relying on the measurements of a single sensor node, multiple sensor nodes perform a measurement. Then, the decision on a report value can be based on

a voting scheme or on resilient aggregation schemes (e.g., using the MEDIAN instead of the AVERAGE). This provides resilience to compromised nodes up to a certain threshold value. Alternatively, detection mechanisms may enable the detection (and exclusion) of false data or compromised nodes.

A *Path-based Denial of Service (PDoS) Attack* [261, 64] shares many similarities with false data injection and replay attacks. However, the goal of the adversary is different. In a PDoS attack, an adversary overwhelms sensor nodes by flooding a multihop end-to-end communication path with either replayed or injected false messages to waste the scarce energy resources. Thus, the energy of en-route sensor nodes is wasted by forwarding falsely injected or replayed messages. Since the transmission of messages is the most cost-intensive factor in WSNs and many nodes may be involved, this is a very severe attack. The defense against an outside adversary is again the use of an authentication scheme. Sensor nodes forward only messages from authenticated neighbors. Thus, an outside adversary can only attack sensor nodes within its transmission range by sending false messages. All subsequent sensor nodes are protected, since these messages are not forwarded. An inside adversary, however, is able to authenticate itself and thus all subsequent nodes also waste their scarce energy resources. En-route filtering schemes [261, 272] have been proposed to cope with an inside adversary. In this thesis, we propose a new highly efficient filtering scheme (cf. Chapter 5) which is optimized for query-response communication [129].

The *False-Endorsement-Based DoS (FEDoS) Attack* [130, 131] is an insider attack which can be performed by an inside adversary who has performed a read-and-write compromise. This attack can be performed against many security schemes addressing the false data injection attack. The idea behind these schemes is that multiple sensor nodes collaboratively generate a report, i.e., one node initiates the report generation and a certain number of neighboring nodes must endorse the report to be valid. This prevents an adversary from injecting false reports if he has compromised less than a certain number of neighboring nodes. However, most of these schemes are susceptible to FEDoS attacks. In a FEDoS attack, an adversary sends a false endorsement to the report generating node to invalidate the report. As a result, the sink does not accept this report, although the information of the report is correct. In this thesis, we present the first general applicable protocols (cf. Chapter 6) to address this issue.

Many WSNs implement some typical applications such as network query, broadcasts from the sink, data aggregation, localization, time synchronization, and over-the-air (OTA) programming which all are susceptible to attacks.

By sending a query, the sink requests certain report data from the WSN. The sink can also send broadcast messages to all sensor nodes, e.g., to distribute new sensing parameters. Data aggregation is used to reduce the amount of transmitted data. Sensor nodes preprocess data, e.g., by calculating the average value of some measurements, and forward only the average value instead of all measurements. Localization and time synchronization are often required to associate measurements with the accurate time and location. Using some OTA programming mechanisms, the sink sends new software to reprogram the sensor nodes.

An adversary attacking these applications may be able to violate all security goals. He can inject new messages, manipulate or replay messages, eavesdrop on messages, or block certain messages (e.g., by jamming or by performing a selective forwarding attack).

Defenses against an outside adversary include the above mentioned (broadcast) authentication, digital signatures, MACs, sequence numbers, timestamps, encryption, and mechanisms to mitigate DoS attacks. Since confidentiality is usually not a problem in these applications, there is often no need for encryption which can lead to energy savings.

Defenses against an inside adversary differ between the applications. The applications network query, sink broadcast, and OTA programming can exploit the property that the sink is assumed to be uncompromised by applying a broadcast authentication scheme addressing attacks against authenticity and integrity. Secure data aggregation in the presence of compromised nodes is a very complex topic and, for example, discussed by Blass in [19]. Time synchronization and localization protocols are often based on some special beacon nodes, e.g., equipped with GPS, which send out accurate time and location values. Thus, the compromise of these nodes may enable an adversary to seriously disrupt these protocols. The design of localization protocols should also consider Distance-Bounding Attacks [49] where an adversary pretends to be closer to a verifier than it is actually the case.

Table 2.7 summarizes the attacks at the application layer and shows the violated security goals and which defenses against an outside adversary can be applied.

Attack	**Violated Security Goals**	**Defenses against outsider**
False data injection	Authenticity and Integrity (report messages)	Digital signatures, MACs
PDoS	Availability (battery), Authenticity and Integrity (report messages)	Authentication
FEDoS	Availability (report messages), Integrity (endorsement)	- (insider attack)
Attacks on typical applications	Possibly all	(Broadcast) Authentication, Digital signatures, MACs, Sequence numbers, Timestamps, Encryption

Table 2.7: Attacks at the application layer

Summary

In this section, we discussed attacks in WSNs from the perspective of an outside as well of an inside adversary. For each attack, we identified the violated security goals and presented defenses against an outside adversary. Furthermore, we discussed the impact an inside adversary can cause by performing the respective attack and which issues should be considered.

The knowledge of possible attacks in WSNs and their impact is required for the development of a security architecture. A security architecture should address the application-

specific threats and risks. By performing a threat and risk analysis, possible attacks and the corresponding security requirements can be identified. This enables the development of a security architecture with the appropriate security level for the scenario. However, we argue that many WSNs have a common basis and that a security architecture should comprise the following components.

A security architecture for a WSN should provide mechanisms to protect against an outside adversary as well as an inside adversary. As basis protection, cryptographic mechanisms such as link layer authentication should be implemented. To minimize the impact of node compromise, it is important to apply a key management scheme that is resilient to node compromise and provides graceful degradation [216]. In addition, a secure broadcast authentication scheme is required to enable the sink to query the network or sensor nodes to perform local broadcast communication. Furthermore, secure routing mechanisms that can tolerate compromised nodes should be applied. For example, multipath routing is preferable over singlepath routing to address attacks such as selective forwarding although it increases the energy costs [200]. Since DoS attacks may have detrimental effects on the WSN, mechanisms should at least mitigate the impact of such attacks. Finally, application specific security mechanisms are required to enable a secure report generation and minimize the possibility of false alarms. These mechanisms should also address attacks such as false data injection, PDoS, and FEDoS since they may have detrimental effects on the functionality of the WSN. For example, a PDoS attack can damage many more sensor nodes than a local sleep deprivation attack.

In this thesis, we focus on the application layer attacks false data injection, PDoS, and FEDoS. We propose several protocols to protect against an outside as well as an inside adversary performing these attacks. Furthermore, we investigate how tamper-resistant hardware can be used to prevent node compromise and propose two attestation protocols for this scenario.

2.3 Resource Consumption in Wireless Sensor Networks

When developing security protocols for WSNs, it is important to know how much energy is consumed by the individual operations. In this section, we give a short overview of resource consumption in WSNs. We give an introduction to research on the resource consumption of a sensor node performing public key cryptography, hash functions, symmetric key cryptography, and message transmission. We use results of these works to define an energy model which is used to analyze the performance of our protocols.

2.3.1 Public Key Cryptography

To achieve a high resilience to node compromise, *public key cryptography* [68] would be the best solution. Each sensor node could be assigned with a unique public-private key pair which is signed by a central authority. If an adversary compromises a sensor node, he can only misuse the individual key pair of this node.

However, public key cryptography has two major drawbacks: the high computational overhead and the large key length. Public key ciphers such as RSA [197] or ElGamal

[82] (detailed descriptions can be found, for example, in [167]) require computationally intensive modular exponentiations and according to the cryptographic key length recommendation [192], the key length should be at least 1008 bit to achieve security level 3 (i.e, short-term protection against medium organizations, medium-term protection against small organizations). Newer cryptosystems based on *Elliptic Curve Cryptography* (ECC) [168], such as NTRU [107] and XTR [148], have been proposed to achieve the same level of security with smaller key sizes and higher computational efficiency. For example, ECC-160 provides comparable security to RSA-1024 and ECC-224 provides comparable security to RSA-2048 [99]. In addition to the key size, signature size of ECC is also less than RSA. For example, ECC-160 has a signature size of 320 bit [58] whereas RSA-1024 has a signature size of 1024 bit [244].

Public key cryptography in WSNs has been intensively investigated, e.g., in [243, 93, 99, 162, 238, 78, 20, 22, 183, 232, 198, 14]. In the following, we present some results on resource consumption of public key cryptography in WSNs.

Gura et al. [99] showed the general applicability of ECC on an Atmel ATmega128 [52] running at 8 MHz. Their implementation of the elliptic curve secp160r1[7] requires 282 byte data memory and 3682 byte code memory. The execution time to perform a point multiplication is 0.81 seconds. This implies that it would take about 1.62 seconds to verify an ECDSA signature on this elliptic curve since the dominant operations in signature verification are two point multiplications.

Wander et al. [238] present energy values for a MICA2dot sensor node. As the MICA2 node, it is also equipped with an Atmel ATmega 128L processor, but the clock speed is only 4 MHz instead of 8-16 MHz. The sign operation of RSA-1024 requires 304 mJ and the verify operation 11.9 mJ. ECDSA-160 requires 22.82 mJ for signing and 45.09 mJ for verifying.

Piotrowski et al. [183] give slightly different values for public key operations. RSA-1024 signature generation requires 359.87 mJ and takes 12.04 seconds. The signature verification requires 14.05 mJ and takes 0.47 seconds. For elliptic curve cryptography they present values for ECC-160. The signature generation requires 26.96 mJ and takes 0.89 seconds. For verifying a signature, 53.42 mJ are required and it takes 1.77 seconds.

Although public key cryptography is generally applicable on WSN hardware, its usage may not be generally reasonable. Encryption times of several seconds [99, 183] or minutes [162] are not tolerable for real-time applications. The dramatically higher energy requirements of public key cryptography render them useless for regular computations. However, a well-considered usage in certain scenarios or applications may be reasonable. For example, each sensor node performs only one public key operation in the initialization phase of a WSN. After that, only efficient symmetric encryption is used to enable a long lifetime of the WSN. Another example is a broadcast authentication scheme where computational and energy intensive signature generations are performed by the sink and the resource constrained sensor nodes need only to perform much cheaper verification operations.

[7] An elliptic curve standardized by NIST/SECG.

2.3.2 Hash Functions and Symmetric Key Cryptography

In contrast to public key cryptography, *hash functions* and *symmetric key cryptography* are much more efficient. This makes them of particular interest for WSNs.

Hash functions such as MD4, MD5, SHA-1, RIPEMD-160 (descriptions can be, for example, found in [167]) are cryptographic primitives which are frequently used in security protocols for WSNs. Commonly used are Message Authentication Codes (MACs) [16], also known as keyed hash functions, for message authenticity and integrity. Furthermore, a hash chain construction is often used in security protocols. A hash chain is a sequence of n hash values, each of some fixed length l generated by a hash function $h : \{0,1\}^l \rightarrow \{0,1\}^l$. The hash function h is successively applied on a seed value c_0 such that $c_{v+1} = h(c_v)$, for $v = 0, 1, \ldots, n-1$. To achieve confidentiality of transmitted messages, encryption algorithms such as DES, 3DES, RC4, RC5, IDEA (descriptions can be, for example, found in [167]) or AES [174] can be used in WSNs.

The resource consumption of hash functions and symmetric key cryptography has been subject of several works. In addition to the energy consumption of public key cryptography, Wander et al. [238] present the energy consumption for calculating SHA-1 hash function and for AES block cipher. The energy consumption for calculating SHA-1 hash values is states with 5.9 μ J/byte. AES-128 encryption/decryption is stated with 1.62/2.49 μJ/byte.

In [261, 121] the calculation of one 64-bit MAC or hash value using RC5 [195] is stated with an energy consumption of about 15 μJ and that a computation takes about 0.5 ms.

In [139], an evaluation framework of block ciphers for WSNs is developed. The framework is used to evaluate Skipjack [173], RC5 [195], RC6 [196], Rijndael/AES [174], Twofish [207], MISTY1 [166], KASUMI [1] and, Camellia [12] according to their security properties and their storage- and energy-efficiency. The result of this evaluation is that the most suitable ciphers for WSNs are Skipjack, MISTY1, and Rijndael/AES depending on the combination of available memory, energy, and required security. The recommended operation mode is Output Feedback Mode for pairwise links (e.g., for mutual authentication between the sink and a sensor node), and Cipher Block Chaining Mode for group communications (e.g., using a group key to encrypt messages for all nodes in a cluster).

In [92], the execution times for different ciphers and hash functions are evaluated for different hardware platforms. The results indicate that the stream cipher RC4 outperforms RC5 on an Atmel ATmega 128; a 128 bit encryption requires 86 μs with RC4 and 413μs using RC5. Furthermore, they showed that the hash functions MD5 and SHA-1 require almost an order of a magnitude higher overhead.

2.3.3 Wireless Transmission

The energy consumption of the wireless interface must not be disregarded. Sending and receiving messages requires a substantial amount of the available energy.

In [261], values for a MICA2 mote with a data rate of 19.2 Kbps are stated. The transmission costs are 16.25 μJ/byte and the receiving costs are 12.5 μJ/byte.

Wander et al. [238] present values for a MICA2dot sensor node which has the same CC1000 transceiver as the MICA2 mote. It operates at 12.4 Kbps. The transmission costs are 59.2 μJ/byte and the receiving costs are 26.6 μJ/byte.

In [183], different values are stated if the CC1000 of a MICA2 mote operates at 433 MHz or 868 MHz. Operating at 433 MHz, the transmission costs are 2.086 μWs/bit = 16.688 μJ/byte using the maximum transmission power. The receiving costs are 0.578 μWs/bit = 4.624 μJ/byte. If the CC1000 operates at 868 MHz, the maximum transmission costs are 1.984 μWs/bit = 15.872 μJ/byte and the receiving costs are 0.750 μWs/bit = 6 μJ/byte.

2.3.4 Comparison

Our short overview has shown that the resource consumption is stated differently in the literature. Reasons for this are, for example, different measurement methods or different assumptions of the current draw or the voltage. Blass discusses this issue in more detail in [19]. However, it can be seen that the most cost intensive factor in terms of energy is the computation of public key cryptography followed by the transmission of messages. The least expensive operations are symmetric operations.

Public key cryptography may require approximately between 3,100 to 21,000 times as much energy than symmetric primitives. Blaß [19] states that the energy consumption of ECC can be up to 22,000 times more than those of symmetric primitives. Using the values from [261], one can see that sending one TinyOS packet with 36 byte requires 39 times as much energy than computing one 64-bit RC5 MAC. In [19], a much larger difference of 190 times as much is stated.

2.4 Trusted Computing

As part of this thesis we developed protocols using tamper resistant hardware, in our case, the *Trusted Platform Module* (TPM)[8]. The TPM is the core of the TCG specifications [231] and is basically a smartcard that serves as a trust anchor for trust establishment. The TPM offers protected storage for cryptographic keys and hardware enhanced calculation engines for random number generation, key-calculation, and hash computation. Although the TPM chip was not specified as necessarily being tamper-resistant, many hardware vendors offer security mechanisms, such as active security sensors, to prevent tampering and the unauthorized extraction of protected keys.

The TPM can generate and store cryptographic keys, both symmetric and asymmetric, and perform asymmetric cryptographic operations. The asymmetric keys can either be marked as migratable or non-migratable, which is specified when the key is generated. Non-migratable keys are always protected by the TPM and must not leave their protected storage.

[8]The TPM is specified only as tamper-evident. However, many real TPM-chips provide tamper-resistance since they are based on smartcards.

The TPM also offers so-called Platform Configuration Registers (PCRs) which are used to store platform-dependant configuration values. These registers are initialized on power up and are used to store software integrity values. Software components (BIOS, bootloader, operating system, applications) are measured by the TPM before execution and the corresponding hash-value is then written to a specific PCR by extending the previous value:

$$Extend(PCR_N, value) = SHA1(PCR_N||value) \tag{2.1}$$

SHA1 refers to the cryptographic hash function used by the TPM and || denotes a concatenation. The trust anchor for a so-called trust-chain is the *Core Root of Trust Measurement* (CRTM), which resides in the BIOS and is first executed when a platform is powered up. The CRTM then measures itself and the BIOS, and hands over control to the next software component in the trust-chain. For every measured component an event is created and stored in the Stored Measurement Log (SML). The PCR values can then be used together with the SML to attest the platform's state to a remote entity. To assure that these values are authentic, they are signed with a non-migratable key, the Attestation Identity Key (AIK). The remote platform can verify the signature and compare these values with reference values to see if the system integrity is trustworthy.

The TPM also offers a concept called *sealing*, which allows a data block to be bound to a specific platform configuration. A sealed message is created by selecting a range of platform configuration registers, a non-migratable key, and the data block which should be sealed. The TPM is then able to decrypt and transfer the sealed data block only if its current platform configuration matches the platform configuration from when the sealing was executed. Sealing provides the assurance that protected messages are only recoverable when the platform is in a known system state.

2.5 Summary

In this chapter, we have shown basic background on WSNs and security in WSNs. Furthermore, we gave a short overview of the resource consumption in WSNs and presented background on trusted computing which we have used in some of our proposed protocols.

Summarizing one can say that securing WSNs is a challenging task. The reasons for this are the special properties of WSNs: resource constraints, different application scenarios, wireless multihop communication, and the possibility of node compromise.

The resource constraints allow only the use of efficient security mechanisms and protocols. As a basic mechanism, symmetric cryptographic primitives are preferred in WSNs. They require significantly less calculation time and energy than public key cryptography. The use of public key cryptography must be well considered.

Since WSNs can be deployed in many different application scenarios, it is hard to develop generic security mechanisms and protocols. For example, a protocol can be suitable for only small-scale WSNs, whereas it cannot be used in large-scale WSNs because the storage requirements on each sensor node increase with the size of the network.

The wireless communication channel enables an adversary to easily perform attacks such as spoofing, message replay, or eavesdropping. This is a general issue that WSNs share with all wireless networks. Since the wireless communication relies on multihop communication, securing WSNs is additionally exacerbated. An adversary that is in possession of a forwarding sensor node is able to perform additional attacks such as data alteration or selective forwarding. However, if security mechanisms are deployed, an adversary must first compromise a sensor node to perform such attacks as an insider.

Node compromise is general possible in WSNs. After compromising a sensor node, an adversary has access to all stored data and is able to perform subsequent attacks as an insider. These insider attacks are a serious threat to WSNs.

Especially the insider attacks false data injection, PDoS, and FEDoS can have detrimental impact on a WSN. Since existing security mechanisms are not sufficient for certain scenarios or yet no generally applicable security mechanisms exist, we focus on these attacks in this thesis. Furthermore, we investigate how tamper-resistant hardware can be used to prevent node compromise and propose two attestation protocols for this scenario.

Before we describe our protocols, we systematically classify different ways to cope with insider attacks. Until now, different ways to cope with insider attacks have not been systematically classified. In the next chapter, we present a new classification how to handle insider attacks in WSNs.

3 Classification

In this chapter, we propose a new two-tier classification how to handle insider attacks in WSNs. We argue that mechanisms and protocols can be first classified by the pursued security strategy. These strategies can be further divided into mechanisms of the same type which implement the respective strategy. For each type of mechanisms, we identify related work and discuss open problems. Related work in this section is discussed on a high level to be able to provide a broad overview. In the subsequent chapters, where we describe our protocols, we provide a more detailed view on related work with regard to our proposed protocols. Our proposed protocols address open problems, presented in this section. They also illustrate all aspects of the different security strategies which can be pursued.

The chapter is organized as follows: In Section 3.1, we introduce the security strategies and briefly introduce the different types of mechanisms. The detailed description of the mechanisms, including related work and open problems, is provided in Section 3.2. The chapter is summarized in Section 3.3.

This chapter shares some material with the previously published work *On Handling Insider Attacks in Wireless Sensor Networks* [132].

3.1 Security Strategies

A well-known classification, which is also used by Bishop in [18] for "classical" computer systems, distinguishes between the security strategies (1) *Prevention*, (2) *Detection*, and (3) *Recovery*. We also use this distinction for our classification w.r.t. WSNs. In the following, we describe these three classes in detail.

3.1.1 Prevention of Insider Attacks

Prevention means that an attack will fail due to implemented mechanisms which an adversary cannot override and which are trusted to be implemented in a correct unaltered way. Mechanisms and protocols which implement a prevention strategy can prevent compromise of parts of the system. Simple preventative examples for "classical" computer systems are passwords (which aim to prevent unauthorized users from accessing the system).

In WSNs, however, sensor nodes can be compromised and the severe resource constraints exacerbate the use of the strong cryptographic mechanisms and protocols. Thus, prevention of insider attacks is hard to achieve or mostly not possible. However, in certain scenarios it might be possible to use mechanisms that implement a prevention strategy to prevent an adversary from successfully performing insider attacks. This implies that

preventative mechanisms protect the keying material stored on a sensor node from being accessed by an adversary. To achieve this, we distinguish between the following two types of mechanisms that implement a prevention strategy:

- Mechanisms increasing the effort for node compromise (see Section 3.2.1)
- Mechanisms making node compromise virtually infeasible (see Section 3.2.2)

Mechanisms of the first type are basic mechanisms which can be applied to increase the effort for an adversary to compromise a sensor node. This prevents only an adversary with rather limited resources from compromising a sensor node and accessing the keying material. However, these mechanisms can be used as a basis for more sophisticated security mechanisms since they increase the time for an adversary to successfully compromise a sensor node. For example, if a security mechanism is secure up to a certain threshold of compromised sensor nodes, these mechanisms can prolong the time until the threshold is reached.

Mechanisms of the second type make a successful node compromise virtually infeasible, i.e., even if an adversary has physical access to a sensor node, he is never able to read keying material out of the node to perform insider attacks. For example, some sensor nodes, which perform special duties, such as key management, localization, time synchronization, etc., are equipped with tamper-resistant hardware. The tamper-resistant hardware protects the keys (or even the whole data) stored on the node from being accessed by an adversary.

3.1.2 Detection of Insider Attacks

Mechanisms that implement a *Detection* strategy have the goal to determine that an attack is underway, or has occurred, and report it. This approach is useful when an attack cannot be prevented, but it can also indicate the effectiveness of preventative measures. Attacks may be monitored to provide information about their nature, severity, and results. An example is a mechanism which monitors the login attempts of a user and reports a warning when a user enters an incorrect password three times. Other examples are Intrusion Detection Systems (IDS) which look for actions or information indicating an attack. One drawback of detection mechanisms is that they do not prevent compromise of parts of the system.

Since it is hard or may be impossible to prevent against insider attacks in WSNs, mechanisms are either based on detection or on recovery (see Section 3.1.3). Mechanisms that implement a detection strategy enable either the sink, or other sensor nodes, to detect that an insider attack is underway, or has occurred. After detection, adequate countermeasures can be applied. Furthermore, detection mechanisms can be used to indicate the effectiveness of prevention or recovery mechanisms. We distinguish between two types of mechanisms:

- Mechanisms to detect manipulated data (see Section 3.2.3)
- Mechanisms to detect compromised nodes (see Section 3.2.4)

The first type enables only to detect that an insider attack has occurred by identifying manipulated data. Compromised sensor nodes are not directly identified. For example, the sink detects that a received aggregated value must be incorrect by performing plausibility checks. The sink may only know that one or more of the aggregating nodes are compromised.

Mechanisms to directly detect compromised nodes are either based on a direct verification that a sensor node has been tampered with, or on misbehavior detection.

To detect manipulated data or to directly identify misbehaving compromised sensor nodes, mechanisms based on *Intrusion Detection Systems (IDSs)* are used. IDS may be classified as either *host-based* or *network-based*, depending on the data collection mechanism [28]. Host-based IDS operate directly on the sensor nodes, whereas network-based IDS operate on packets captured from network traffic. The latter is commonly used in WSNs. In addition, IDS may be classified based on the detection technique. In *signature-based detection*[1], any action that matches the pattern of an attack in a predefined signature is considered as an intrusion. This technique has a low false positive rate, but does not perform well at detecting unknown attacks, and it might be difficult to define the characteristics of an attack. The advantage for the application in WSN is the low computational overhead. For example, a signature to address PDoS attacks (see Section 2.2.4) could be "A node generating more than 20 reports per minute must be reported!". *Anomaly-based detection* defines a profile of normal behavior. Any activity varying from this profile is considered as an intrusion. This profile can be updated to adapt the system to a new "normal" behavior. This technique may detect previously unknown attacks, but may exhibit a high false positive rate and is computationally more intensive. For example, the average packet loss rate is defined as the normal behavior, and if a node drops more packets than the average, this node is assumed to be compromised. Due to the resource constraints and the unpredictable environment where WSNs may be deployed, anomaly-based detection is difficult to implement in WSNs. The environment may change (e.g., the weather conditions) or the conditions in one area of the WSN differ from other areas. Thus, it is hard to define "normal behavior". In *specification-based detection*, a set of constraints is defined that describes the "intended" behavior of a program or protocol. The execution of the program or protocol is monitored with respect to the defined constraints. An alarm is raised if a deviation is detected. This approach enables the detection of known and unknown attacks and exhibits a low false positive rate. The disadvantage is that specifications are usually derived manually from protocol descriptions which may be time-consuming. In addition, specifications are protocol specific which may require specific specifications for each WSN application scenario.

3.1.3 Recovery from Insider Attacks

Recovery can be categorized in two forms [18]. The first is to stop an attack and to assess and repair any damage caused by the attack. An example is a backup system which can

[1] Signature-based detection is also known as Misuse detection.

restore a file that has been deleted by an adversary. In this form, the system's functionality is inhibited by the attack and resumption of correct operation is required. In the second form, the system continues to function correctly while an attack is underway. It relies on fault tolerance and is typically used in safety-critical systems. The difference to the first form of recovery is that the system does not function incorrectly. However, the system may disable nonessential functionality.

In the context of WSNs, the first form of recovery requires the detection of the insider attack, i.e., either the detection that malicious data has been injected, or that sensor nodes are compromised. After that, mechanisms stop the adversary from continuing the attack and/or assess and repair the damage caused by the attack.

The second form of recovery does not necessarily require any detection mechanisms. Mechanisms implementing the second form are designed in such a way that the system automatically recovers itself (within certain boundaries) while an insider attack is underway to maintain the system's functionality. For that, techniques such as fault tolerance [189] and Byzantine fault tolerance [137] as well as security techniques are used. A typical example for WSNs is the use of the MEDIAN instead of the AVERAGE in an aggregation scenario. Assume that one node aggregates the temperature values it receives from a certain number of neighboring sensor nodes. If the AVERAGE is used, a single compromised node can inject a large value which results in a false average value. The MEDIAN, however, is resistant against this attack (up to a certain number of compromised nodes) and provides automatic recovery from such a false data injection attack.

We distinguish between four types of mechanisms which implement one of the two forms of recovery:

- Mechanisms to exclude compromised sensor nodes (see Section 3.2.5)
- Adaptable mechanisms (see Section 3.2.6)
- Mechanisms to reprogram (compromised) sensor nodes (see Section 3.2.7)
- Mechanisms tolerating node compromise (see Section 3.2.8)

An adversary can be stopped from continuing the attack by excluding the compromised sensor nodes from any further participation in the network.

Alternatively, adaptable mechanisms can be used to recover from insider attacks, e.g., the threshold value of a security scheme based on threshold cryptography is increased. Thus, the adversary is stopped from continuing insider attacks unless he has again compromised more nodes than the threshold value.

Likewise, the sensor nodes may be reprogrammed to support additional security mechanisms, e.g., an IDS system is ex post installed on the nodes. In certain cases it may be possible to reprogram compromised nodes to remove the malicious code.

Mechanisms tolerating node compromise can be used to implement the second form of recovery. The above mentioned threshold schemes are typical examples. As long as an adversary has compromised less nodes than the threshold value, the functionality of the WSN still remains. It is also possible to consider these mechanisms as *preventive* since

they prevent the effect of the insider attack. However, the insider attack by itself is not prevented.

Additional mechanisms may be necessary, e.g., new nodes must be added to regain coverage in areas where many nodes have died or have been excluded. Thus, deployed security mechanisms, such as key management, must support the addition of new sensor nodes during runtime of the WSN. Furthermore, it may be necessary that the sink sends new queries after the detection of invalid reports and exclusion of the responsible compromised nodes.

3.2 Detailed Description of Security Mechanisms

In this section, we describe the security mechanisms implementing the above mentioned security strategies in detail, give an overview of related work, and present the most challenging open problems in the respective area.

3.2.1 Mechanisms Increasing the Effort for Node Compromise

Mechanisms increasing the effort an adversary has to spent to successfully compromise sensor nodes can be seen, similar to system hardening [81] of operating systems, as a basic principle in securing WSNs. These mechanisms exacerbate the access to hardware and software components of a sensor. The goal is to maximize the required time, cost, knowledge, technical resources etc.

The required effort to access the sensor node hardware can be increased by some basic mechanisms. First, a cover, which is difficult to remove, protects the sensor board. Next, mechanisms are applied which prevent an adversary from misusing the programming interfaces (e.g., USB, serial, or JTAG (Joint Test Action Group)) to read/write data from/to the node. Therefore, interfaces have to be removed, disabled, or protected, e.g., by passwords. Likewise, the bootstrap loader is protected by mechanisms such as passwords. Another way to access a sensor node is over-the-air (OTA) programming which could also be exploited by an adversary. Thus, OTA has to be disabled if not required, or mechanisms such as authenticated code updates have to be applied. Furthermore, software-based mechanisms can prevent an adversary from learning from a node compromise to facilitate future node compromises. For example, if all nodes store the cryptographic keys at the same location in the memory, then an adversary just has to read out this specific location after identifying it once.

These mechanisms may be sufficient to prevent an adversary with rather limited resources to compromise a sensor node. In addition, even a powerful adversary requires some time to compromise a sensor node. This enables security mechanisms to assume a secure initialization phase [270, 8] and enable a practical use of threshold schemes.

Related Work

Hartung et al. [100] showed that it is possible to compromise a sensor node in less than one minute if no precautions, such as disabling the programming interfaces, are taken.

Becher et al. [15] investigate physical attacks on sensor nodes and present countermeasures, such as disabling the JTAG interface. In [5], a code obfuscation mechanism for WSNs is proposed to improve the resilience to node compromise. The goal is a diversified key protection scheme for WSNs that diversifies data and code segments by creating different and obfuscated data and code segments for each sensor node in the network. Thus, compromising one node does not decrease the time to compromise another node.

Open Problems

Currently available hardware are mostly research platforms and are not designed with security in focus. The nodes provide interfaces which are easy to access and which allow a fast (re-)programming making them unsuitable for many real applications with high security requirements. Future sensor platforms have to be designed with considering node compromise and provide mechanisms that increase the effort for an adversary. The most challenging problem is to realize this at a very low price. Obfuscation techniques are also only able to slightly increase the effort for an adversary [13].

Although sensor nodes should be as cheap as possible, in certain scenarios it may be worthwhile to equip the sensor nodes (or at least a few of them) with tamper-resistant hardware. This might be reasonable in scenarios such as health care, where only a limited number of sensor nodes are used and there is a high demand for security. Further research is required to investigate the applicability of tamper-resistant hardware, such as the TPM (see Section 2.4) in WSNs.

3.2.2 Mechanisms Making Node Compromise Virtually Infeasible

An adversary can be prevented from successfully performing insider attacks by making node compromise virtually infeasible. Without the cryptographic keys stored on a sensor node, an adversary cannot appear as a legitimate node of the WSN. Thus, mechanisms are required which protect these keys.

In general, an adversary must be impeded from recovering the keys either by performing attacks over the wireless channel and/or by physically compromising sensor nodes.

The former requires that the applied software and protocols are not susceptible to malware or buffer overflow attacks. Furthermore, potentially applied OTA programming mechanisms must be secured to prevent unauthorized access to sensor nodes. In addition, the applied cryptographic mechanisms and key length must be appropriate to achieve the desired security level for the application scenario of a WSN.

The latter requires a physical protection of sensor nodes. This could be easily achieved by deploying sensor nodes in a physically protected area. If the sensor nodes cannot be accessed by the adversary, they cannot be compromised. An example is a mobile WSN in a harbor where the sensor nodes are deployed inside locked containers. An adversary needs to break into a container before he is able to compromise a sensor node.

However, this approach is not possible in the majority of cases since sensor nodes are often deployed in unattended and hostile areas where an adversary can physically access the nodes. To prevent an adversary from compromising the sensor nodes, tamper-

resistant hardware, which is resistant to side channel attacks [81, 9, 10, 128], can be used. This hardware can either protect the whole sensor node or only parts of the node. Since the protection of all components of a sensor node is extremely difficult and occasions large costs, only security relevant parts may be protected. For example, the TPM (cf. Section 2.4) can be used to provide secure storage for cryptographic keys, and a secure execution environment for cryptographic operations. Thus, an adversary that has "compromised" a sensor node cannot misuse this node to perform insider attacks since he has no access to the keys. However, since TPMs are too cost-intensive, they cannot be used in many WSN scenarios. Especially in large scale WSNs, it is not reasonable to equip all sensor nodes with a TPM. In certain scenarios, however, it might be reasonable to equip only a few nodes which perform some special tasks and require additional protection with a TPM.

Alternatively, hardware which provides tamper-resistance up to a certain extent may be used in certain scenarios. This may be achieved in scenarios where the lifetime of a WSN is shorter than the time an adversary requires to compromise some unknown proprietary hardware. For example, the MarathonNet [182] presented in Section 2.1.4 is used only for a couple of hours during the marathon. The design and development of the sensor hardware is specifically conducted for this purpose. Thus, an adversary has only the limited time of the marathon to analyze the hard- and software and to create a code image with its malicious code. However, if the same hardware is used again in the future and the adversary steals one of the sensor nodes, the adversary has sufficient time to prepare his attack for the future.

Related Work

Building tamper-resistant hardware, attacks on it, and protection techniques have been intensively investigated especially in the area of smartcards, e.g., [9, 10, 128]. However, only little research has been performed on using tamper-resistant hardware in WSNs. In an unpublished work [87], Ganeriwal et al. propose to equip some of the nodes in a WSN with a TPM for secure key management.

Open Problems

Most cryptosystems used in WSNs are in general resistant to cryptoanalysis. However, the research community currently assumes a key length of 64 bit for symmetric encryption/decryption, MAC computation, and hash calculations. Using 64 bit for symmetric encryption/decryption provides a level 2 protection, i.e., very short-term protection against small organizations, as defined in [192]. To resist until year 2010, one may consider a key size of at least 75 – 80 bit for symmetric systems [192]. Thus, WSNs that have a longer operation time should use longer key sizes as the usually used 64 bit although it increases the energy consumption. Therefore, the recommendations presented in [147, 146] or [192] should be taken into account.

Hash functions are commonly used as security primitives in WSNs. In general, a hash function provides three properties: (1) preimage resistance, (2) second-preimage resis-

tance, and (3) collision resistance [167]. According to [192], the minimum size of a hash in the year 2010 should be 155 bit. This is much higher than the usually assumed 64 bit in WSNs. However, the collision resistance property is often not required. A successful attack on collision resistance does not break the security of the applied mechanisms or protocols. As a result, a shorter bitlength of the hash values can be assumed. Hash functions providing collision resistance have hash values roughly twice the bitlength of one-way function providing only preimage and second-preimage resistance [167]. Thus, if only preimage and second-preimage resistance is required, half of the bitlength as stated in [192] can be used[2].

We argue that the key length depends on the desired level of security and the operation time of the WSN. A WSN for disaster recovery that is deployed only for a few hours may use short key sizes since an adversary may not have enough time to break the keys.

The major problem of tamper-resistant hardware is the higher costs. In large scale WSNs it might be possible to equip only some of the sensor nodes with tamper-resistant hardware which requires new security protocols for heterogeneous WSNs. Likewise; new efficient security protocols for small scale WSNs are required, where all nodes are equipped with tamper-resistant hardware.

Another area of future research is the development of mechanisms that do not require additional tamper-resistant hardware. For example, sensor nodes can detect by themselves that they are removed from the WSN by an adversary and then delete all stored cryptographic keys. However, this does not protect from on-site compromise and is not applicable in mobile WSNs or when sensor nodes can be moved.

3.2.3 Mechanisms to Detect Manipulated Data

In certain scenarios it might not be possible, or desired, to directly detect compromised sensor nodes. Instead, the detection of insider attacks relies on the detection of manipulated data using some IDS techniques (see Section 3.1.2). For example, in an aggregation scenario the information which node has sent which measurement is lost after the aggregation. However, it might still be possible to detect that an insider attack has occurred by performing plausibility checks. Another example is a WSN that is deployed to send alarm messages in case of a certain event. The lifetime of such a WSN is usually very long and it is relatively unlikely that sensor nodes will be compromised since the WSN is not a typical target for an adversary. Thus, the detection of compromised sensor nodes is not (yet) desired since a constant monitoring would be to resource intensive. False alarms are accepted, and if a false alarm occurs, unnecessary reactions, e.g., firefighters react on a fire alarm, are also accepted. Only after the false alarm a detection process is triggered to identify the compromised sensor node(s). After identification, the compromised nodes are excluded.

[2]This assumption should only be made in such resource constrained systems such as WSNs. In non-constrained systems, collision-resistant hash functions with the appropriate bitlength should be used.

Since for most scenarios the detection of compromised nodes is more advantageous, most works in the area of detecting insider attacks concentrate on this aspect (see Section 3.2.4).

Related Work

The detection of manipulated data introduced by adversaries is particularly relevant in data aggregation schemes. Once the sink receives aggregated data, it verifies their validity through mechanisms such as sampling or redundant sensor nodes. Yang et al. [259] propose SDAP: A Secure Hop-by-Hop Data Aggregation Protocol for Sensor Networks. The basic idea is that the sink can detect anomalies in the aggregates. After the detection of an anomaly, the sink identifies the potentially compromised sensor nodes and these nodes must send their measured data and aggregates directly to the sink. So the aggregation is cancelled ex post. The aggregation schemes presented in [188, 44, 75, 108] are additional examples that detect malicious data but cannot identify where the malicious data is introduced and which node is responsible for it.

The Statistical En-Route Filtering Scheme [261], used to cope with PDoS attacks, enables en-route nodes to verify that the data has been tampered with and to filter these manipulated data out.

Open Problems

Since insider attacks are not stopped by mechanisms that only try to detect manipulated data, additional mechanisms are required to identify the compromised nodes and to choose an appropriate recovery strategy.

3.2.4 Mechanisms to Detect Compromised Nodes

In Section 2.2.3 we distinguished between two types of node compromise: (1) Read-only Compromise and (2) Read-and-Write Compromise. The type of compromise strongly affects the difficulty to detect compromised nodes.

The detection of a read-only compromise is very difficult since the node neither has been modified nor seems to behave maliciously. Mechanisms could exploit the actions of the adversary during the compromise, e.g., neighboring nodes detect that a node has been removed, turned off for reprogramming or has been deployed at a different location. However, these approaches may only be applicable in stationary WSNs.

The detection of a read-and-write compromise is more likely. A less practicable method is a direct physical verification of the sensor nodes in the deployment area by verifying if a sensor node has been tampered with. However, this might only be possible for certain scenarios and very small WSNs. Other possibilities are either based on misbehavior detection or on attestation mechanisms. Misbehavior-based detection uses IDS techniques to identify compromised nodes. Attestation techniques address node compromise at its root by directly detecting that the code on a sensor node has been modified. Both approaches could be combined to achieve higher detection accuracy and lower false positive rate, e.g., misbehavior detection may trigger code attestation.

Node compromise detection can be performed by neighboring nodes, the sink, or designated nodes such as cluster heads or aggregation nodes. The designated nodes may also be equipped with tamper-resistant hardware. Since misbehavior detection mostly relies on direct observations, typically neighbors monitor each other to detect node compromise. However, some mechanisms may also enable the sink or a designated node to detect that a certain node is misbehaving. Detection mechanisms based on attestation may be performed by any node. However, some approaches work only reliably if a neighboring node performs the attestation.

Related Work

In the context of attestation in WSNs, a number of software-based approaches have been presented that rely on optimal program code and exact time measurement [211, 210, 209, 208]. A highly optimized program verification routine is executed to verify the memory of a sensor node. The routine is invoked by the attestor by sending a pseudo-random number as a challenge to calculate a hash value by traversing the memory space in a pseudo-random fashion. The probability is very high that a modification in the memory will be detected. The hash value computed on a compromised node will either be wrong, or the execution time of the routine will be longer than normal. Thus, a compromised node is detected. In [213] a similar approach is presented that relies on code obfuscation and time measurement. A randomized attestation routine is sent to a sensor node to traverse its memory. If an adversary analyzes and modifies the routine, this would take much longer and thus, a compromised node would be detected.

Park et al. propose a Program-Integrity Verification (PIV) protocol [176]. It uses PIV servers (PIVS) that verify the integrity of the program of the sensor. For each verification a PIVS creates a new randomized hash function and sends it to the sensor node in a PIV code (PIVC). The PIVC computes a hash value on the program and sends it back to PIVS which verifies the hash value. The protocol also requires a certain time threshold for the attestation. Chang et al. propose an extension to PIV in [45]. They propose a distributed authentication protocol to reduce the communication traffic and the energy consumption for authenticating the PIVSs. Yang et al. propose two schemes for distributed software-based attestation in [260]. Problematic with the approaches presented by Park, Chang, and Yang is the possibility of a simple attack if the sensor hardware is based on a Harvard Architecture[3]. Due to the fact that code and data segment are separated and the data segment is much larger than the code segment, an adversary may easily copy the entire original program to the data segment, add its own malicious code to the code segment and perform a successful attestation using the original program code stored in the data segment. If no exact time measurement is performed, like in the above mentioned works, the adversary is always able to perform a successful attestation.

Misbehavior-based detection has been intensively investigated. Many schemes have been proposed to detect compromised nodes attacking the routing infrastructure in wire-

[3]Nearly all currently available sensor platforms are based on a Harvard Architecture, e.g., the Berkeley Mica Motes.

less ad-hoc networks, e.g., Watchdog [163], Control Messages [143], Neighborhood Watch [30], Statistical Anomaly Detection [268], or Local Intrusion Detection System [6]. Some of these mechanisms are also used in WSNs, especially Watchdog has been often proposed to use in WSNs. IDS in WSNs have been investigated in [57, 199, 165, 175, 2, 85, 11, 171]. Lee et al. propose an alternate path routing scheme that is also able to detect and exclude sensor nodes performing a selective forwarding attack [145]. A reputation-based framework for sensor networks, where nodes maintain reputation for other nodes and use it to evaluate their trustworthiness, is presented by Ganeriwal et al. in [88]. Another general framework for identifying compromised nodes is proposed by Zhang et al. [265]. It utilizes an alert-based detection where sensor nodes report alerts to the sink. The sink evaluates the alerts using an observability graph to decide whether a node is compromised or not. The detection of node replication or node clone attacks is addressed in [178, 51, 48]. A protocol to detect compromised beacon nodes for secure location discovery is proposed in [154]. A possible approach to detect read-only compromise is proposed by Song et al. [219]. They assume that an adversary is not able to deploy a compromised sensor node back to the original position. Thus, the detection of location change will become an indication of a potential node compromise.

Open Problems

Although a lot of research has been performed in detecting compromised sensor nodes, there still exist much room for improvement. The reasons for this are the severe resource constraints of the sensor nodes and the unreliable wireless channel. As a result, detection schemes, especially based on misbehavior detection, are very unreliable. In addition, these schemes usually require intensive monitoring of neighboring nodes which is very energy consuming, e.g., neighboring nodes permanently monitor each other's message forwarding behavior to detect selective forwarding attacks. Thus, new concepts are required to efficiently detect compromised nodes without wasting energy by intensive monitoring. Furthermore, mechanisms for misbehavior detection nearly always address only specific attacks since they are signature-based. Thus, they do not support the detection of unknown attacks. An attestation approach, however, addresses arbitrary node compromise at its root by verifying that a sensor node has been tampered with.

However, the proposed software-based attestation techniques [211, 210, 209, 208, 213] rely on optimal program code and exact time measurement. These approaches are not applicable in multihop WSNs, since they require an authenticated communication channel between the verifier and the attestor, and rely on minimal time fluctuations. The multihop communication and external influences, e.g., weather conditions, may seriously influence the time to send the attestation messages between the entities performing the attestation. The approaches proposed in [176, 45, 260] are additionally susceptible to the above mentioned copy attack. Furthermore, all software-based attestation approaches do not work if the adversary modifies the hardware by, e.g., attaching more memory, or a more powerful CPU. Likewise, all approaches are not resistant against relay attacks, where an adversary relays the attestation requests to a more powerful machine.

The software-based attestation techniques and the direct physical verification have an additional drawback. If an adversary knows the schedule of the verifications, he may reprogram the nodes with the original code just before the verification, and later reprogram the nodes with the malicious code. However, in large scale WSNs this might not be practicable for the adversary.

3.2.5 Mechanisms to Exclude Compromised Sensor Nodes

After a sensor node is identified as being compromised, this node should be excluded from the network to stop this node from causing damage to the WSN. Therefore, mechanisms are required to revoke the (symmetric) cryptographic keys of compromised sensor nodes and remove the nodes from the routing tables.

Key revocation protocols can be divided in two types: *centralized* or *distributed* key revocation protocols [39]. In the centralized approach, upon detection of a compromised node, the sink broadcasts a revocation message to all sensor nodes that need to exclude the compromised node. The revocation message either contains the node identifier of the compromised node, or the entire key ring that have to be removed. In distributed revocation, the revocation decision is made by the neighbors of a compromised node. Therefore, a voting scheme can be applied to decide whether to revoke a given node or not. Advantageous of distributed revocation is the lower overhead, i.e., only local broadcast messages are required which results in a faster exclusion of a compromised node, and the absence of a single point of failure. However, this approach is more complex and error prone. In general, key revocation protocols must solve two major challenges. First, since they are carried out in the presence of active adversaries, the key revocation protocol has to achieve revocation of sensor nodes that are compromised by an adversary despite the active participation of that adversary in the protocol. Second, the protocol must be efficient w.r.t. energy, i.e., rely on very simple cryptographic operations.

After the keys of a sensor node are revoked, the routing tables of the sink and the remaining sensor nodes are updated by removing the node identifier of the compromised node. If necessary, new routes are calculated. Thus, the compromised node is neither chosen as a forwarding node, nor packets from this node are accepted for further processing.

Related Work

In contrast to key distribution, key revocation has not been investigated intensively. Centralized revocation schemes are presented by Eschenauer and Gligor [84], and Dini and Savino [69]. A distributed revocation scheme is proposed by Chan et al. [43]. In [39], Chan et al. present an overview of key-distribution methods in WSNs, e.g., [43, 155, 76, 84, 152], and their salient features to provide context for understanding key and node revocation. They define basic properties that distributed sensor node revocation protocols must satisfy and present a protocol for distributed node revocation that satisfies these properties under general assumptions and a standard attacker model. In [160], several techniques, structures and algorithms for key revocation are presented.

Kim et al. propose a key revocation scheme for mobile WSNs [125]. In [266], Zhang et al. propose a group re-keying scheme for filtering schemes such as [261, 272] to exclude compromised key rings.

Lee et al. propose SeRINS (a Secure alternate path Routing IN Sensor networks) to detect and isolate compromised nodes which try to inject inconsistent routing information [145].

Revocation messages are usually broadcasted. To achieve authenticated broadcasts, an asymmetric mechanism is required, i.e., only the sender can sign messages, and the receiver can only verify messages. This prevents an adversary from impersonating the sink and from successfully injecting false revocation messages. This can be achieved either by digital signatures or by μTESLA [181] and its variations.

Digital signatures are usually generated using public key cryptography such as ECC [127] or RSA [197]. Continues efforts have been performed to improve ECC for WSNs, e.g., [99]. However, as already mentioned in Section 2.3, asymmetric cryptographic mechanisms are significantly more computationally expensive than symmetric ones. In [212], [193] and [194], broadcast authentication schemes based on public key cryptography are proposed that are adapted to the resource constraints of WSNs. However, they still require significantly more energy than symmetric approaches. Alternatively, digital signatures can be based on one-time signature (OTS) schemes. Luk et al. present a broadcast authentication protocol in [159] that enable immediate authentication of broadcast messages and is based on Merkle-Winternitz OTS. Another scheme based on OTS is presented in [46]. The scheme proposed in [42] enables broadcast authentication in hierarchically organized networks by building hash-trees where sensor nodes collaboratively verify broadcast messages by recomputing the hash-tree.

The μTESLA protocol, proposed by Perrig et al. [181], uses only efficient symmetric primitives and achieves asymmetry by a delayed disclosure of cryptographic keys. It is based on TESLA [180, 179] and adapted to the special properties of WSNs. Extensions to μTESLA have been proposed, e.g., in [151, 153, 156] and [106]. A similar scheme based on delayed disclosure is presented in [97]. Schemes for local broadcasts between neighboring sensor nodes based on μTESLA are proposed in [126] and [74]. A variant of μTESLA that enables immediate authentication of the broadcast messages sent at regular and predictable times is proposed in [159].

Protocols to mitigate DoS attacks on broadcast authentication schemes based on digital signature schemes such as ECDSA are proposed in [79, 239, 71]. Ning et al. [172] introduce message-specific puzzle to mitigate DoS attacks on signature-based and μTESLA-based broadcast authentication.

Open Problems

The major problems of key revocation lie in the area of efficient, reliable, and secure broadcast. An adversary who jams or delays broadcast messages may interrupt the revocation messages to some part of the network and remain active there. Simply flooding [91] the WSN in an authenticated way may be a solution, which however, introduces a large communication overhead. This approach may only be possible in scenarios where

revocation events are rare, such that the costs associated with simple flooding are compensated with the security benefits of node revocation.

Already proposed broadcast authentication protocols have all their advantages and disadvantages. Digital signatures such as ECDSA provide a high resilience to node compromise. On the other hand they are computational much more intensive which requires much more energy than mechanisms based on symmetric primitives. They are also susceptible to DoS attacks where an adversary sends false broadcast packets to force sensor nodes to perform expensive signature verifications. Digital signatures based on OTS and symmetric primitives require fewer computations but inherit a large signature size which requires more energy in message transmission. Efficiently, only very short (about 8 bit) messages can be signed. Thus, these schemes cannot be used in every scenario. μTESLA-based broadcast authentication has the advantage of being very efficient. However, it does not support immediate authentication. DoS attacks are also possible where an adversary forces sensor nodes to perform expensive packet forwarding by injecting many false broadcast messages.

The proposed different approaches of key revocation all have their advantages and disadvantages. The centralized approach requires the sink to broadcast a revocation message. The forwarding of the message requires some time and is energy consuming. In addition, the sink may be a single point of failure. In contrast, distributed revocation should be faster, as it requires predominantly only local broadcast messages, and avoids a single point of failure. However, it is inherently more complex and prone to design errors than centralized revocation.

Mechanisms using a centralized approach to exclude compromised nodes have the disadvantage that the exclusion process is not immediately. For example, if the sink receives information about potentially compromised nodes, invokes a decision process, and finally broadcasts a revocation message, this takes some time. In contrast, however, if exclusion of sensor nodes is locally decided, compromised nodes can be immediately excluded.

3.2.6 Adaptable Mechanisms

Recovery can be achieved by using adaptable mechanisms. These mechanisms can be either used to adapt to expected conditions, or to react on different or changed conditions. For example, a threshold-based mechanism is adapted to the expected application scenario and conditions, e.g., the threshold value is adapted to the expected number of node compromises. However, if the conditions change, it may be necessary to adapt the threshold value, e.g., increasing it after the detection of insider attacks. Thus, the adversary might be impeded from continuing the insider attack; at least as long as he has not compromised more nodes than the threshold value again.

Furthermore, it may be necessary to deploy new sensor nodes to react on node failures. Thus, security mechanisms, such as key management, must be adaptable to new or mobile sensor nodes.

Related Work

Most threshold schemes, e.g., [261, 272, 258, 257, 267, 129] may be adapted by broadcasting new parameters using one of the broadcast authentication schemes presented in Section 3.2.5. Key management schemes such as [181, 84, 43, 155, 76] support the addition of new nodes directly.

Open Problems

The major problem is a secure and reliable ex post adaption. Authenticated broadcast mechanisms (with the issues mentioned in Section 3.2.5) are required to prevent an adversary from injecting false reconfiguration messages. However, due to the wireless multihop communication, an adversary may be able to drop messages or jam the wireless channel to prevent nodes from receiving the messages.

3.2.7 Mechanisms to Reprogram (Compromised) Sensor Nodes

In many WSNs the program code of the sensor nodes is pre-configured and not changed during the lifetime of the network. However, in certain scenarios, e.g., health care with only a limited number of expensive sensor nodes, it might be reasonable to perform a code update of the sensor nodes. A manually code update is often not possible for many reasons, e.g., there are too many sensor nodes, some sensor nodes are mobile, or the exact location of the nodes is not known. Thus, sensor nodes are reprogrammed over the air (OTA) by broadcasting the code update to all sensor nodes. Since the code update may be relatively large, this is very energy consuming. However, in WSNs with a long lifetime, a code update may be reasonable to react on new conditions. For example, if the WSN was developed without considering the occurrences of insider attacks and now insider attacks happen, a code update containing an IDS module may be installed on the sensor nodes. The code update mechanisms must be secure, i.e., the authenticity, integrity, and, where required, confidentiality of the code update must be provided. This prevents an adversary from misusing the OTA programming to inject his own malicious code. Furthermore, mechanisms to "heal" sensor nodes by performing a code update to remove malicious code are desired.

Related Work

In the area of code update in WSNs, a number of protocols have been proposed. These protocols include Deluge [115] which is the de facto TinyOS [233] network programming system, Multihop Over-the-Air Programming [230], Trickle [149], Infuse [134], and Multihop Network Reprogramming Service [135]. However, these approaches have been developed in the context of efficiency and reliability and assume a trustworthy environment. Deng et al. attempt to secure code updates by using Merkle hash trees and hash chains to authenticate the code distribution [66]. Sluice, proposed by Lanigan et al. [138], also uses hash chains to secure the code update. In [80], Dutta et al. proposed an approach to secure Deluge. However, these approaches only address the issue

of authenticating the code updates from the sink at the receiver side. Seshadri et al. proposed SCUBA [209] to enable the sink to verify that the receiver indeed applied the code update. SCUBA enables the detection of compromised nodes and either repairs them through code updates, or revokes the compromised nodes. Strasser et al. [224] proposed an autonomous and distributed recovery algorithm. The algorithm enables code update, reset, or shutdown of failed or malicious nodes.

Open Problems

The major issue of all reprogramming mechanisms is the required energy to forward the code update along multiple hops to all nodes in the network and the required time to perform the code update. However, if updates are only performed infrequently, this approach is still reasonable.

Future research is required in the area of healing or self-healing of sensor nodes from node compromise. The SCUBA protocol is a first step in this direction. However, an active adversary who is aware of the protocol may prevent a successful code update with the original code. Thus, new approaches must be developed, e.g., using tamper-resistant hardware.

3.2.8 Mechanisms Tolerating Node Compromise

The term *tolerance* is known from fault tolerance [189] or Byzantine fault tolerance [137, 37]. Fault tolerance describes the property that a system is able to continue operating, possibly at a reduced level (also known as graceful degradation), rather than failing completely, when some part of the system fails. For example, in a classical computer system the throughput of a server is reduced or the response time is increased. Thus, the system as a whole is not stopped due to failure of parts of the system. A Byzantine fault tolerant system is able to defend against a Byzantine failure, i.e., an arbitrary failure. However, failures can be also caused by intruders. In the context of wired networks, intrusion tolerant mechanisms have been studied, e.g., based on threshold public-key cryptography [251, 31]. However, these mechanisms cannot be directly transferred to WSNs because of different properties and threats in WSNs. The scarce resource constraints inhibit intensive usage of public-key cryptography, and the wireless channel enables an adversary to eavesdrop, modify packets, inject packets etc. However, certain properties of WSNs enable efficient mechanisms which tolerate node compromise and the resulting insider attacks.

In general, mechanisms tolerating node compromise have the goal to mitigate the impact of node compromise, i.e., the resulting insider attacks have only little or no impact on the functionality of the WSN. Mechanisms based on threshold schemes are commonly used which exploit the redundancy property of most WSNs. For example, messages are sent along multiple paths to the sink, i.e., even if a compromised en-route node drops certain messages, there is a high probability that the message still reaches the sink using a different route. Thus, even in the presence of an active adversary, the WSN is able to perform its tasks. However, these schemes are only secure up to a

certain threshold of compromised sensor nodes. Alternatively, graceful degradation can be achieved by restricting the possibilities of a sensor node. Thus, an adversary can only exploit these possibilities. For example, if a sensor node is only able to generate valid reports for its local position, an adversary cannot misuse this node to generate reports appearing from other locations in the WSN. Thus, the impact of a node compromise is limited to the region of the compromise.

Another simple example for a threshold-based intrusion tolerant mechanism is the above described use of the MEDIAN instead of the AVERAGE in an aggregation scenario. The MEDIAN provides resistance against false data injection attacks up to a certain threshold of compromised nodes. This threshold depends on how many values are used to generate the MEDIAN.

Toleration mechanisms have the advantages that they are often easier to implement and that they require less resources than more sophisticated techniques such as detection mechanisms. In addition, they might be easily adaptable to different requirements, e.g., the threshold value of a threshold scheme can be adjusted according to the number of compromised sensor nodes. Toleration mechanisms are also a good candidate to be used to extend other more sophisticated mechanisms to increase the overall achieved level of security.

Related Work

Key management is of great importance to build mechanisms tolerating node compromise. If a single network wide key is shared by all sensor nodes, a single node compromise would break the entire security of the WSN. Ideally, a sensor node stores only the keys for its own links. This could be realized either by using public key cryptography (where the adversary gains only access to the private key of that node) or a Key Distribution Center (KDC) based approach, e.g., [181], where the symmetric keys are established through a trusted base station similar to Kerberos [223] (where the adversary gets only access to the symmetric keys related to the compromised node). However, both approaches are very resource intensive. We discussed the resource consumption of public key cryptography in Section 2.3.1 and obviously, the KDC approach introduces a significant communication overhead. Thus, key management schemes make a tradeoff between node compromise tolerance and security by storing additional keys on a node to facilitate key establishment. In PIKE [41], other nodes are used as trusted intermediaries to perform key establishment between neighboring nodes. Many schemes rely on key predistribution [24], i.e., sensor nodes are pre-configured with some keys in such a way that neighboring or each potential group of nodes share a common key. Deterministic key predistribution schemes are proposed in [142, 38]. The random key pre-distribution protocols [84, 43, 116, 155, 76, 215] rely on probabilistic key sharing among nodes enabling a shared-key connectivity with high probability. However, an adversary compromising one node also gets access to keys from other nodes and links in different areas of the WSN. Thus, the adversary can perform attacks, e.g., eavesdropping, also on these links. Nevertheless, these schemes still provide intrusion tolerance, but at a reduced level. A different way to tolerate node compromise is used in the LEAP protocol proposed by

Zhu et al. [270, 271]. Multiple keying mechanisms are used to establish four different keys. Depending on whom the sensor node is communicating with, the appropriate key is used. The protocol mitigates the impact of a node compromise to the immediate network neighborhood of the compromised node, i.e., an adversary may get only access to keys which a node shares with neighboring sensor nodes. Overviews of key management can be found for example in [36] and [39].

To address false data injection and PDoS attacks many schemes have been proposed [261, 272, 258, 257, 267]. A threshold scheme is used in [261, 272, 258] where [258] and [267] additionally mitigate the impact of a node compromise to the location of the compromise.

The One-Time Sensors [17, 86] concept also addresses false data injection attacks to deceive the sink. The basic idea of this concept is to allow a sensor node to send only one single report to the sink. After that, this node acts only as a router and forwards messages of other nodes. Thus, an adversary compromising one node is only able to inject one false message.

Secure aggregation is addressed for example in [188, 21, 23, 19]. The framework presented in [188] has the goal to ensure that the aggregated result is a good approximation to the true value in the presence of a small number of compromised nodes. In [21], ESAWN is proposed which implements a security-energy trade-off and is able to tolerate multiple compromised nodes.

Compromised sensor nodes can easily affect the routing in WSNs. Thus, routing mechanisms should be able to tolerate a certain number of compromised sensor nodes. For example, to address selective forwarding attacks, [90], [62] and [59] use multipath routing where a message is sent along multiple paths to achieve a high probability that the message reaches the sink. Likewise, INSENS [60, 61, 65] uses redundant multipath routing and bypasses malicious nodes.

Open Problems

Key management is still the key issue in building mechanisms tolerating node compromise. The contradictory terms, efficiency and resilience to node compromise need to be harmonized. Public-key cryptography would provide a perfect resilience to node compromise but is (still) too inefficient. Currently proposed mechanisms sacrifice resilience to node compromise to achieve greater efficiency.

One major drawback of many protocols is the requirement of threshold schemes. The security of these schemes totally breaks down if an adversary compromises greater or equal nodes than the threshold value. Thus, new mechanisms are required that minimize the benefit for an adversary for a successful node compromise without relying on a threshold scheme. For example as realized in the One-Time Sensor scheme [17, 86] addressing false data injection attack.

3.3 Summary

Insider attacks are a serious threat in WSNs and require sophisticated approaches to cope with them. In this section, we have systematically classified the different ways to cope with insider attacks in WSNs. For this purpose, we have introduced a new two-tiered classification. The approaches have been classified based on the pursued security strategy. We distinguish between prevention, detection, or recovery strategy. The strategies can be implemented by different types of mechanisms. We have classified these types of mechanisms based on their key characteristics. The current state of research has been integrated in our classification to identify challenging open problems and areas of future research.

The major problem of mechanisms implementing a prevention strategy is that resource constraints, the possibility of node compromise, and cost factors make preventative mechanisms very difficult to realize in WSNs. Mechanisms increasing the effort for node compromise are only basic mechanisms and can only prevent a very limited adversary from compromising a sensor node. To prevent an adversary from successfully performing insider attacks, he must be impeded from accessing the cryptographic keys stored on the node. The only reliable way to achieve this are mechanisms that make node compromise infeasible, e.g., using tamper-resistant hardware. However, because of the increasing cost, this approach is not possible in the majority of applications. In large-scale WSNs it is not possible to equip all sensor nodes with tamper-resistant hardware. In general one can state that in WSNs prevention is hard to achieve and predominantly mechanisms implementing a Detection or Recovery strategy are used in WSNs. However, some form of Prevention can be achieved by Detection and Recovery mechanisms. For example, if a compromised node performing an insider attack is detected and excluded by its neighboring nodes, further insider attacks are prevented. But the node compromise and the initial insider attacks are not prevented a priori and thus, we classify this as Recovery since a "healing" process is invoked.

The properties of WSNs also complicate the development of mechanisms implementing a detection strategy. Misbehavior-based detection usually requires intensive monitoring of neighboring sensor nodes. This is very energy consuming since sensor nodes must overhear many messages and cannot go into sleep mode during monitoring. Thus, more sophisticated mechanisms are required enabling detection without intensive monitoring. An additional problem of misbehavior-based detection is that in WSN usually only known attacks are detected. To overcome this shortcoming, attestation-based mechanisms can be used. However, current software-based attestation techniques are not suitable for multihop communication and are also susceptible to certain easy to perform attacks. Thus, new attestation techniques for WSNs are required.

Recovery is often achieved by implementing mechanisms tolerating node compromise. For this purpose, threshold schemes are used that are secure up to a certain number of compromised sensor nodes. If the threshold is reached, the security totally breaks down. Furthermore, many recovery mechanisms to exclude compromised nodes do not provide immediate exclusion of compromised nodes. Thus, efficient exclusion mechanisms based on local decisions are required.

Additional research is required in the area of efficient, reliable, authenticated broadcast. Current mechanisms require too much energy, are susceptible against certain DoS attacks, or are only applicable in certain application scenarios.

In the next chapters, we present the main contributions of this thesis. Based on the results of our classification, we propose several protocols addressing identified problems. We consider two areas in this thesis.

First, we consider resource constrained WSNs which perform the principle application of a WSN: a query is sent to the WSN, according to the query is a measurement of some physical phenomena performed, a report is generated, and the generated report is transmitted back to the sink. For this scenario, we propose several protocols to address the application layer insider attacks false data injection, PDoS, and FEDoS (cf. Section 2.2.4) while considering the above identified problems. The proposed protocols implement mechanisms which pursue detection and recovery strategies. The protocols are presented in Chapter 5 and Chapter 6.

Second, we examine how to achieve prevention (and detection) of insider attacks for high-security WSN scenarios with lower cost constraints. To achieve prevention, we propose to equip a minority of sensor nodes with tamper-resistant hardware in hybrid WSNs. These specially protected sensor nodes are used to perform special tasks such as key management, localization, time synchronization, etc. Furthermore, we propose two efficient attestation protocols for these hybrid WSNs enabling "ordinary" sensor nodes (which can be multiple hops away) to detect that an adversary has tried to tamper with a special node. Just like the software-based attestation protocols, our approach also addresses node compromise detection at its root by directly detecting software manipulations on the sensor node. In contrast to software-based attestation, our protocols are also able to perform detection if sensor nodes are multiple hops away. Our results are presented in Chapter 7.

Before we describe our proposed protocols in detail, we present the system model we assume for our proposed protocols in Chapter 4.

4 System Model

In this chapter, we describe the system model which we assume for our proposed protocols when not stated otherwise. The assumptions of our system model are similar to related work to enable a comparison of our protocols with existing protocols. First, we describe the assumed hardware devices for sink and sensor nodes and their respective properties. After that, we describe the assumption of the network which consists of one sink and several sensor nodes. Finally, we describe the different types of adversaries in our adversary model. The adversary model is used to evaluate the security of our protocols provide against these different types of adversaries.

The chapter is organized as follows: In Section 4.1, we describe the device model and in Section 4.2 the network model. The adversary model is introduced in Section 4.3. Finally, the chapter is summarized in Section 4.4.

4.1 Device Model

In the device model, we distinguish between a model for sensor nodes and for the sink. We refer to user, task manager node, and sink (cf. Section 2.1.1) collectively as sink.

Sensor Node Model We assume sensor nodes with comparable properties and resources as the MICA motes described in Section 2.1.3. Although it has been shown that public key is in principle applicable in WSNs [99, 238], we assume that such operations cannot be generally performed to enable a long lifetime of the WSN. However, certain efficient public key-based broadcast authentication schemes can be used where the signature verification operations for sensor nodes are cheap and only signature generation for the sink is resource-intensive. In general, sensor nodes are able to perform some basic operations, like computing hash functions, symmetric encryption, XOR operation, etc.

Security protocols can occupy a fraction of the available 4 kbyte EEPROM and the 512 kbyte flash memory. Frequently used data can be stored in the EEPROM whereas rarely used data can be stored in the flash memory.

The communication is done via bidirectional wireless links, i.e., all sensor nodes have the same transmission range. Communication with nodes outside the transmission range is performed in a multihop fashion.

Sensor nodes are battery-powered and have limited energy resources. Our assumptions of the energy consumption of (cryptographic) operations are based on results presented in Section 2.3. We introduce the necessary details on energy consumption and available energy resources in the respective sections of our energy analyses.

Sensor nodes are not equipped with tamper-resistant hardware and can be compromised. However, we assume that certain mechanisms increasing the effort for node

compromise (cf. Section 3.2.1) are applied. This enables the assumption, similar as for instance in [270, 8, 258], that the system bootstrapping phase is secure and nodes cannot be compromised during this phase. In this phase, a sensor node can securely perform some initial procedures. It is assumed that the time to perform the initial procedures T_{est} is less than the lower bound to compromise a sensor node T_{min}, i.e., $T_{est} < T_{min}$. To protect against eavesdropping, a single network wide key can be used to protect the initially exchanged messages. This key is erased before T_{min} is reached.

Sink Model We assume that the sink is unconstrained in its computational, storage and energy resources. The sink is equipped with sufficient storage space to store any required data, e.g., unique symmetric keys with each sensor node. Furthermore, the computational resources enable the sink to perform resource-intensive operations such as public-key cryptography. However, the communication capabilities are similar to the sensor nodes, i.e., the sink cannot use a unidirectional link to reach sensor nodes located farther away. In contrast to the sensor nodes, the sink is assumed to be trustworthy and that it cannot be compromised.

4.2 Network Model

We assume a large scale, static WSN consisting of one sink and a great number of sensor nodes with the aforementioned properties. The network is densely deployed, so that the same or similar physical phenomena can be detected by multiple sensor nodes. Sensor nodes do not know their immediate neighboring nodes in advance, i.e., they can be deployed randomly (e.g., via aerial scattering). Because of the limited communication capabilities, the devices communicate in a multihop fashion using bidirectional wireless links.

After the deployment of the sensor nodes, we assume that each pair of neighboring nodes establishes a pairwise key in a secure bootstrapping phase using existing techniques. These keys are used by security mechanisms as basic protection against an outside adversary to ensure authenticity, integrity, and freshness of the exchanged messages between neighboring nodes.

We assume that an adversary has physical access to the deployment area of the WSN and is able to compromise sensor nodes. Thus, an adversary is able to perform both outsider and insider attacks and can perform them either as mote-class or laptop-class adversary. In the following section, we discuss our considered adversary model in detail.

4.3 Adversary Model

To evaluate the security of a protocol, an adversary model must be defined which is used as the basis of the security analysis. The Dolev-Yao model [70], which we briefly introduced in Section 2.2.3, is widely accepted as the standard by which cryptographic protocols should be evaluated. However, this model is too rigid to be used to evaluate security protocols for WSNs. In the Dolev-Yao model, the adversary is assumed to be a

legitimate user of the network and has full control over all messages that are sent through the network. Thus, the adversary is able to overhear, analyze, or modify messages, as well as to generate new messages and sent these messages to any user of the network. The adversary is only bound by the applied cryptographic mechanisms which he cannot circumvent without knowing the required keys. However, the Dolev-Yao model does not involve entity compromise. Furthermore, in most WSNs it is unrealistic that an adversary has such communication power to intercept all messages of the whole network. Therefore, our adversary model consists of different types of adversaries with different capabilities as we will explain later. This enables an adaption to different scenarios and different security requirements.

Before we describe our adversary model in detail, we briefly introduce two adversary models ([111] and [7]) that have been proposed in the context of routing in ad-hoc networks. We argue that these models are not generally applicable in WSNs. However, our adversary model shares some parts with these models.

The adversary model presented in [111] formalizes an *active-n-m* adversary, where n is the number of compromised insiders that hold keying material, and m is the total number of adversary nodes in the network. All adversary nodes have the same capabilities as non-malicious nodes. In addition, the nodes are able to distribute compromised keys to all other *m-1* nodes. This approach has the disadvantage that the number of nodes that are able to perform insider attacks is hard to determine since it is unclear how many new nodes of the adversary are successfully added to the network using compromised keying material. Furthermore, an adversary is able to deploy more powerful nodes with additional resources. Thus, our adversary model comprises only a number of compromised nodes. This number includes potentially fabricated or cloned nodes where an adversary compromises only a few nodes and uses the captured keys to fabricate new "legitimate" nodes with new identities. To mitigate this issue, key management schemes that are resilient to node fabrication attacks, such as [113], should be used. In addition, our model considers malicious nodes that are equipped with additional resources, e.g., more energy capacity or a transceiver with higher transmission range.

In [7], the authors propose an adaptive threat model for secure ad-hoc routing protocols. They consider the strength of an attacker, communication capability, insider or outsider, and the goal of an attacker on route integrity. This model is appropriate for evaluating routing protocols for ad-hoc networks. However, to evaluate security protocols for WSNs additional aspects must be considered. For example, many security mechanisms for WSN use threshold schemes which must be considered in the assumed adversary model.

Thus, we define our own adversary model for WSNs to evaluate our proposed security protocols. Our adversary model is based on the different classes of adversaries described in Section 2.2.3. Table 4.1 gives an overview of our adversary classification for WSNs.

We first make a distinction between outsider (type I) and insider (type II) adversaries. Although the main focus of our work are inside adversaries, for the sake of completeness we also consider the damage an outside adversary can cause. An outside or inside adversary can try to perform the attacks described in Section 2.2.4. However, as already mentioned above, an outside adversary can be easily impeded from performing most of

these attacks by cryptographic means. The relevant attacks, such as a jamming attack, which an outsider is still able to perform, are considered in all analyzes.

We further refine outsider and insider adversaries by their computational and communication capabilities, i.e., we distinguish between mote and laptop class. Thus, we have the following four sub-categories: I.1 (Outsider, Mote Class), I.2 (Outsider, Laptop Class), II.1 (Insider, Mote Class), and II.2 (Insider, Laptop Class). A mote class adversary has the same or similar resources as an already deployed sensor node. The resources of a laptop class adversary can have different characteristics. The laptop class device can have only slightly increased resources, e.g., only increased energy resources or an increased receive radius of the wireless transceiver. However, a laptop class adversary can have resources comparable to the Dolev-Yao adversary, i.e., he might be able to receive and transmit messages in any part of the network. Such a powerful adversary, even if he is an outsider, may be able to jam the whole WSN. However, such an adversary is unrealistic in most scenarios since a wireless signal quickly drops to low levels with growing distances. Thus, an adversary must either be close to the target or use extreme high levels of transmission power.

Each of the four types of adversaries are further refined by the number of nodes m the adversary owns. In the case of a type I adversary (Outsider), we distinguish in each case (type I.1 or type I.2) if the adversary owns only one node or owns $m > 1$ nodes; i.e., type I.1.1 (Outsider, Mote Class, 1 Node), I.1.2 (Outsider, Mote Class, $m > 1$ Nodes), I.2.1 (Outsider, Laptop Class, 1 Node), I.2.2 (Outsider, Laptop Class, $m > 1$ Nodes). For example, a type I.1.1 adversary can jam only a very limted area of the network whereas a type I.2.2 adversary may be able to jam the whole WSN.

The type II.1 (Insider, Mote Class) and II.2 (Insider, Laptop Class) adversaries are each further refined in three subcategories specifying how many sensor nodes the adversary has compromised. We distinguish between 1, $1 < m < \mathcal{T}$, and $m \geq \mathcal{T}$, where $\mathcal{T}$ specifies a threshold value if a threshold scheme is applied. Thus, we distinguish between the following types of adversaries: type II.1.1 (Insider, Mote Class, 1 Node), type II.1.2 (Insider, Mote Class, $1 < m < \mathcal{T}$ Nodes), type II.1.3 (Insider, Mote Class, m $\geq \mathcal{T}$), type II.2.1 (Insider, Laptop Class, $m = 1$ Node), type II.2.2 (Insider, Laptop Class, $1 < m < \mathcal{T}$ Nodes), and type II.2.3 (Insider, Laptop Class, m $\geq \mathcal{T}$ Nodes).

In addition, we make the following general assumption of an adversary:

- As in the Dolev-Yao model [70], we assume perfect cryptography, i.e., an adversary cannot break any cryptographic mechanism if he is not in possession of the cryptographic keys. This includes the inability of encrypting/decrypting messages, generating/verifying MACs, finding preimages[1] or second preimages[2] of one-way functions[3], etc.

[1] Given a one-way function h, and $y = h(x)$ it is computational impossible to find any preimage $\tilde{x}$ such that $h(\tilde{x}) = y$.

[2] It is computational infeasible to find a second preimage $\tilde{x} \neq x$ such that $h(x) = h(\tilde{x})$

[3] Note that our protocols do not require collision resistance, i.e., it is computational infeasible to find any two disjoint inputs $x, \tilde{x}$ which hash to the same output such that $h(x) = h(\tilde{x})$.

Outsider/ Insider	**Available Resources**	**Number of nodes m**	**Adversary Category**
Outsider	Mote Class	$m = 1$	I.1.1
		$m > 1$	I.1.2
	Laptop Class	$m = 1$	I.2.1
		$m > 1$	I.2.2
Insider	Mote Class	$m = 1$	II.1.1
		$1 < m < \mathcal{T}$	II.1.2
		m $\geq \mathcal{T}$	II.1.3
	Laptop Class	$m = 1$	II.2.1
		$1 < m < \mathcal{T}$	II.2.2
		m $\geq \mathcal{T}$	II.2.3

Table 4.1: Adversary Classification

- By misbehaving in the execution of the security mechanisms applied between neighboring nodes, an adversary cannot obtain the used cryptographic keys.
- If a sensor node is compromised, all data, such as the cryptographic keys, can be accessed by the adversary.
- An adversary is able to combine cryptographic keys of multiple compromised sensor nodes. Therefore, the nodes can exchange this data either locally (e.g., a mote class adversary that has compromised neighboring sensor nodes) or globally (e.g., a laptop class adversary that establishes a wormhole to another compromised node farther away).

4.4 Summary

In this chapter, we have described the system model consisting of device, network, and adversary model for our proposed protocols. The device model specifies the properties and assumptions for sink and sensor nodes whereas the network model specifies the general assumptions of the network itself. The adversary model is used to evaluate the security of our proposed protocols.

5 Addressing PDoS Attacks

In Section 3.2.8 we have argued that recovery strategies using mechanisms that tolerate node compromise are usually based on threshold schemes. Thus, if an adversary compromises more nodes than some threshold value, the security is broken. It is a challenging problem to design a security protocol for WSNs that provides recovery for an arbitrary number of compromised nodes.

In this chapter, we present a protocol that applies the second form of recovery by using a mechanism that tolerates compromised nodes and adaptable mechanisms to protect against PDoS and false data injection attacks. The protection against PDoS attacks does not rely on a threshold scheme and is able to tolerate an arbitrary number of compromised sensor nodes. A threshold scheme is used to protect against false data injection attacks. However, if the threshold scheme is broken, the adversary is limited to the immediate region of the compromised nodes. Focus of this protocol is the protection against PDoS attacks whereas the protocols presented in Chapter 6 focus on false data injection (and FEDoS) attacks.

This chapter shares some material with the previously published work *STEF: A Secure Ticket-Based En-route Filtering Scheme for Wireless Sensor Networks* [129].

The chapter is organized as follows. In Section 5.1, we introduce and motivate the problem which we address in the proposed protocol. We distinguish our protocol from related work in Section 5.2. In contrast to Chapter 3, we provide more details specific for this context. The specific assumptions are presented in Section 5.3 before we describe our proposed protocol in detail in Section 5.4. The security of our protocol is discussed in Section 5.5, and a theoretical performance analysis is presented in Section 5.6. In Section 5.7, we give a short overview of the implementation and the performed simulations of the performance of our proposed protocol. Finally, the chapter is summarized in Section 5.8.

5.1 Introduction

PDoS and False Data Injection Attacks (see Section 2.2.4) are two serious types of attacks. A successful PDoS Attack may prevent a large number of sensor nodes from performing their tasks for a certain time period by overloading them with a large amount of false messages. Much worse, a successful PDoS Attack can result in a totally exhaustion of the energy resources of the nodes which makes them unusable. No less serious are false data injection attacks which may enable an adversary to deceive the sink. This can for example result in false alarms.

To protect against such attacks, it is necessary to filter most of the false messages as early as possible, because sending and receiving messages is the most cost-intensive

factor in WSNs. The few overlooked false messages should be further rejected at the sink. Therefore, en-route nodes must be able to detect false or replayed messages, and drop them. Several security approaches [261, 272] have recently been proposed for this purpose. To achieve the en-route filtering, a *symmetric key sharing* approach is used, where the keys for generating and verifying Message Authentication Codes (MACs) are the same. As a consequence, verification keys can be misused to generate reports. This limits the resilience to node compromises. In [257], an approach based on commutative ciphers is presented which eliminates the problem. However, appropriate commutative ciphers are all based on public key cryptography, e.g., RSA [197], which requires more resources than available in most WSNs.

We propose a Secure Ticket-Based En-route Filtering Scheme (STEF) that is resistant against false data injection and PDoS attack. The protocol neither requires symmetric key sharing among sensor nodes nor expensive cryptographic operations.

STEF exploits the typical operation mode of query-response in WSNs. Many application scenarios expect WSNs that process, store and provide the sensed data to the network users upon their demands. Thus, as a common communication paradigm, the network users are expected to issue queries to the WSNs through the sink before obtaining the information of their interest. In this scenario, our protocol provides protection against PDoS attacks even if an adversary is able to compromise an arbitrary number of sensor nodes. However, in scenarios where sensor nodes are expected to autonomously generate messages (e.g., alarm messages if a certain temperature value is reached), STEF is not applicable. As we already stated, it is often not possible to develop generic security protocols for WSNs which are applicable in all application scenarios. During the work on this thesis, we also developed a generic variant of STEF where sensor nodes are able to autonomously generate messages [83]. In contrast to STEF, however, the resilience to node compromise is limited in this approach.

The main idea of STEF is to forward messages to the sink only if they contain a valid *ticket*. A ticket is issued by the sink, within a query, to a specific sensor node which enables the node to generate a valid response. A query message is authenticated and cannot be forged or replayed by an adversary. The response message contains the ticket which is verified by en-route nodes. Only response messages which contain a valid and fresh (i.e., it has not been used before) ticket are forwarded. False or replayed messages are filtered out. The requirement of a valid and fresh ticket for each message prevents an adversary from injecting as many messages as desired to perform a successful PDoS attack even if he has compromised several nodes. Furthermore, the impact of node compromises is limited to the region of the compromise. Unlike other protocols (e.g., [261, 272]), an adversary cannot generate valid reports appearing from arbitrary locations within the network if he has compromised a certain threshold of nodes. The analyses show that our proposed protocol can achieve a good level of security, and is very effective in filtering false messages. Due to the fact that false messages are filtered after one hop and the low processing overhead, STEF shows remarkable energy savings.

5.2 Related Work

In this section, we present related work in the area of filtering false data. These works have already been briefly introduced in Chapter 3. In contrast, we provide now a more detailed view and distinguish our work from these works.

The first proposals for filtering false data in WSNs are SEF [261] and IHA [272]. The basic idea of these two proposals is that en-route nodes share symmetric keys with member nodes in a sensor node group or cluster. Multiple member nodes endorse reports by generating a MAC for the reports using these keys. En-route nodes can verify MACs before forwarding packets. In SEF, the member nodes and en-route nodes use randomly pre-distributed keys from different key partitions to generate and verify MACs. There is a high probability that en-route nodes share keys with the member nodes, and thus can verify endorsements to filter false messages. However, there are several problems with SEF. First, as SEF uses a probabilistic approach, it cannot guarantee that every false packet will be filtered out on its way to the sink. Additionally, a false packet will be forwarded for a certain number of hops before it is filtered out which wastes resources. Second, SEF remains only safe if an adversary cannot compromise more keys than a certain threshold value. If the threshold is exceeded, the adversary can forge reports appearing from anywhere in the WSN. In IHA, member nodes and en-route nodes set up interleaved keys, using randomly pre-distributed keys. These interleaved keys and the hop-by-hop authentication ensure that the sink will detect any false data when no more than a certain number of nodes are compromised. Otherwise, an adversary can forge arbitrary data reports.

In [258], the authors of SEF present an improved protocol, where the keys are bound to geographical locations, thus limiting the scope for which an adversary can misuse compromised keys. A report is forwarded inside a beam from the source to the sink. However, the protocol is still probabilistic, with the problems mentioned above. Furthermore, an adversary compromising a certain number of nodes within a beam may forge reports appearing from arbitrary locations within the beam.

In [257], a commutative cipher-based filtering mechanism is presented that drops fabricated reports en-route without symmetric key sharing. A source node establishes a secret association with the base station on a per-session basis, while the intermediate forwarding nodes are equipped with a witness key which is used to verify the authenticity of reports without knowing the original session key. However, all appropriate commutative ciphers are based on public key cryptography. We adopt the idea of using the query response operation mode and the separation of verification and generation means in our protocol. In contrast to the proposed mechanism, STEF uses only lightweight one-way functions, which is much more efficient.

In [267], a comprehensive set of location-based compromise-tolerant security mechanisms for WSNs is presented. The idea of location-based keys (LBKs), which bind private keys of individual nodes to their ID and geographic location, is proposed. An LBK-based neighborhood authentication protocol and a method to establish pairwise keys is presented. Furthermore, a location-based threshold-endorsement (LTE) protocol

is proposed to filter bogus messages. In this approach, sensor nodes are assumed to have much more computing power and are able to perform public key cryptography.

The protocol presented in [266] uses SEF to filter false messages. Furthermore, a method is presented for excluding compromised nodes, after they have been identified, by updating a group key on the uncompromised nodes. A family of predistribution and local collaboration-based group rekeying protocols is proposed. The design of these protocols is based on the ideas that future group keys can be preloaded to the sensor nodes before deployment, and neighbors can collaborate to protect and appropriately use the preloaded keys.

In [269], two protocols for false report filtering are presented, which separate the generating keys and verification keys by using one-way hash chains (OHCs). In the first protocol, sensor nodes signal events using OHCs, which enables en-route nodes to verify the authenticity of reports based on commitments of detecting sensor nodes, but prevents them from forging events. This protocol is not suitable for high density networks, since the storage requirements for OHC and commitments become excessive. The second protocol extends the first protocol to a collaborative filtering protocol by using commitment predistribution, which makes it more adaptable for mobile and high-density sensor networks. Both protocols use a probabilistic approach, meaning there is no guarantee that every false packet will be filtered out on the path to the sink. Additionally, a false packet will be forwarded for a certain number of hops before it is filtered out.

The work in [64] presents a solution to prevent PDoS attacks using OHCs to protect end-to-end communications. This work focuses on preventing en-route nodes or outsiders from performing PDoS attacks. Insider attacks from report generating nodes are not considered.

5.3 Setting

We assume the system model described in Chapter 4. In addition, we make the following assumptions.

We assume a reactive setting of the WSN, i.e., the network is queried by the sink and sensor nodes do not autonomously generate messages. Query messages are sent from the sink to sensor nodes in the area of interest. We assume that the response message is forwarded along the reversed path that the query message traverses. This assumption is reasonable, because the probability that the route changes is very small since the time between query and response is rather short. In cases where the route for a response message is different from the route for the corresponding query message (e.g. because of en-route node failure), the sink has to send a new query message after a certain period of time.

A (broadcast) authentication scheme is used to prevent an adversary from injecting or replaying query messages. Sensor nodes are able to immediately verify the authenticity of query messages and drop false or replayed query messages. Appropriate schemes are, for example, proposed in [99, 194, 42]. In [58], we proposed an efficient signature

scheme which is also appropriate for this purpose in certain scenarios. In addition to the authentication scheme, mechanisms to protect against DoS attacks on signature verification can be used [172, 79, 239, 71].

Furthermore, we assume that appropriate localization mechanisms exist to enable each node to obtain its location after deployment, e.g., [141, 204, 40, 33, 34, 77]. Most monitoring applications require such location-awareness to determine the location of events.

5.4 Protocol Description

In this section, we present STEF. First, we describe the basic protocol for en-route filtering supporting authenticity and integrity of transmitted data. We then show how our protocol can be easily modified to support the confidentiality of queries and reports in response messages.

5.4.1 Basic Protocol

The main idea of STEF is that reports from sensor nodes are forwarded towards the sink only if they contain a valid ticket. The ticket concept is realized with a query-response communication which is a typical operational mode in sensor networks. Sensor nodes can act in three different roles: (1) En-route node (*EN*), (2) cluster head (*CH*), and (3) cluster node (*CN*). The sink randomly selects a node in the area of interest which acts as the current *CH* for this query-response communication and sends the query including the ticket via the *EN*s to it. *CH* builds a dynamic cluster with its *CN*s, i.e., its direct neighbors. The ticket is specific for *CH*, i.e., it can be used by this *CH* only. Before sending a response to the query, *CH* generates a report according to the query which must be endorsed by multiple *CN*s and attaches the ticket to the report. The report is sent back towards the sink, and the *EN*s are able to verify the correctness of the ticket. Messages including no or invalid tickets are immediately dropped.

STEF consists of five phases: *Initialization*, *Queries from Sink*, *Report Generation*, *En-route Filtering*, and *Sink Verification*. These five phases are presented below.

Initialization

The initialization phase is performed only once to configure the sensor nodes before deployment, and to execute some initialization procedures directly after deployment. This phase is assumed to be secure as mentioned above.

Each sensor node S_i for $i = 1, \ldots, n$ has a unique identifier ID_{S_i} and is preloaded with a unique key K_{S_i} shared with the sink, henceforth called *personal key*. An error range ε is preconfigured within which *CN*s accept reports from *CH* and generate endorsements.

After the nodes are deployed, they obtain their location and report it to the sink. To decrease the communication overhead, the location reports can be aggregated or piggybacked in other messages. Furthermore, each sensor node establishes pairwise keys

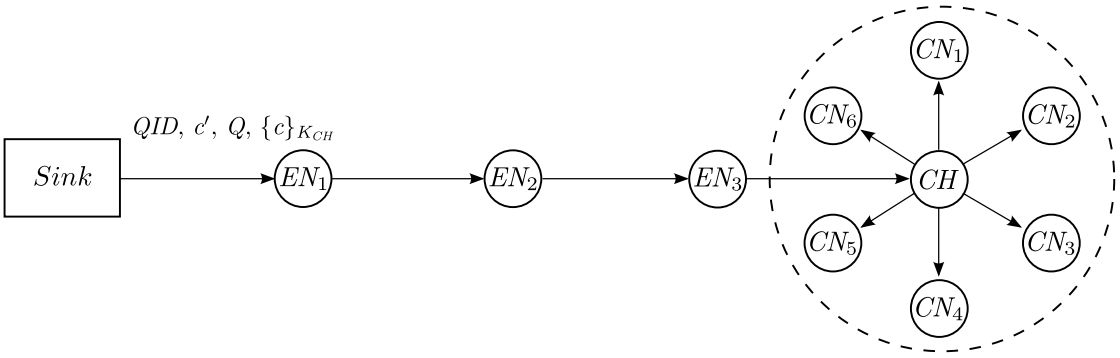

Figure 5.1: Query message sent from the sink to *CH*

with its one-hop neighbors as stated in the network model described in Section 4.2. After the initialization phase, the network can be queried by the sink.

Example 5.1 *Figure 5.1 shows an example WSN consisting of ten sensor nodes. The three en-route sensor nodes EN_1, EN_2, and EN_3 forward messages from the sink to the cluster head CH and vice versa. CH initiates a report generation and generates a report with the help of the neighboring cluster nodes $CN_1, \ldots, CN_6$.*

Queries from Sink

After the initialization phase, the WSN is initialized and the sink is able to send queries. Each query message from the sink includes a ticket. The ticket concept is realized as follows: The sink generates a random value $c \in_R \{0,1\}^l$ of a certain length, e.g., $l = 64$ bit. Next, a *one-way function* $h : \{0,1\}^l \rightarrow \{0,1\}^l$ that provides preimage and second preimage resistance, e.g., a hash function [167], is applied to the value c, so that $c' = h(c)$. Preimage resistance means that for any given x, it is easy to compute $y = h(x)$, but given a value y, it is not feasible to compute a value $\tilde{x}$ such that $y = h(\tilde{x})$. Second preimage resistance means that it is computational infeasible to find a second preimage $\tilde{x} \neq x$ such that $h(x) = h(\tilde{x})$. Note that our protocol does not require collision resistance[1] which enables a much shorter bit length of the hash values[2].

The sink knows the location of all nodes in the area of interest, and randomly selects one node as *CH* and sends a query message to it. The message contains a unique query identifier *QID*, the value c', the query Q representing the interest of the user expressed in multiple attribute-value pairs, and the value c encrypted with the personal key K_{CH} shared between the sink and *CH* representing the ticket. Thus, the query message has the following form:

$$Sink \rightarrow CH : QID, c', Q, \{c\}_{K_{CH}} \tag{5.1}$$

[1] Collision resistance means that it is computational infeasible to find any two disjoint inputs $x, \tilde{x}$ which hash to the same output, i.e., such that $h(x) = h(\tilde{x})$.

[2] Compare also Section 3.2.2.

The sink stores QID, c, c', Q, ID_{CH} for further verification purposes. En-route nodes can verify the ticket in the response message later in the en-route filtering phase using the value c'. For each query message, a new value for c is randomly chosen and the appropriate c' value is calculated. The message is authenticated by an authentication scheme which supports immediate authentication as mentioned above.

Example 5.2 *In the example shown in Figure 5.1, the sink sends out a query message. A query Q might look like "What is the temperature at location X?". The whole query message is forwarded hop-by-hop to CH. All en-route nodes EN_1, EN_2, and EN_3 store the tuple (QID, c') for future verification purposes.*

Algorithm 5.1 describes the actions of an EN when receiving a query message. EN stores the tuple (QID, c') in the data structure $QList$.

Algorithm 5.1 ENQuery(QID, c', $QList$)

```
1: if received Query is authentic then
2:     add (QID, c') to QList
3:     forward message
4: end if
```

Report Generation

CH performs the initial measurement of the physical phenomena and initiates the report generation. However, *CH* cannot generate a valid report for the sink by itself. *CH* requires the help of t neighboring *CN*s which endorse the measurement. An endorsement is only sent from a *CN* if it agrees on the measurement within a certain error range. A *CN* generates an endorsement by computing a MAC on the received report R using its personal key.

First, we explain the report generation by continuing Example 5.2. Afterwards, we describe the actions of *CH* and the *CN*s in general.

Example 5.3 *When CH receives the query message, it generates a report R according to the query; e.g., "The temperature at location X is 23° C." and sends this report to its neighbors. In Figure 5.1, CH sends R to nodes CN_1, ..., CN_6. Each sensor node that agrees on the report within a certain error range, uses its personal key to generate a MAC (the endorsement) on R, and sends the MAC to CH. The authenticity and integrity of these messages is ensured using the pairwise keys shared between the nodes and CH. In Figure 5.2, all cluster nodes CN_1, ..., CN_6 agree on the report and calculate the MAC as follows for $i = 1, \ldots, 6$*

$$MAC_{CN_i} = MAC(R, K_{CN_i}) \tag{5.2}$$

and send the generated MACs to CH. CH also generates a MAC, and additionally, chooses t different MACs randomly, where t is a system parameter, and compresses them

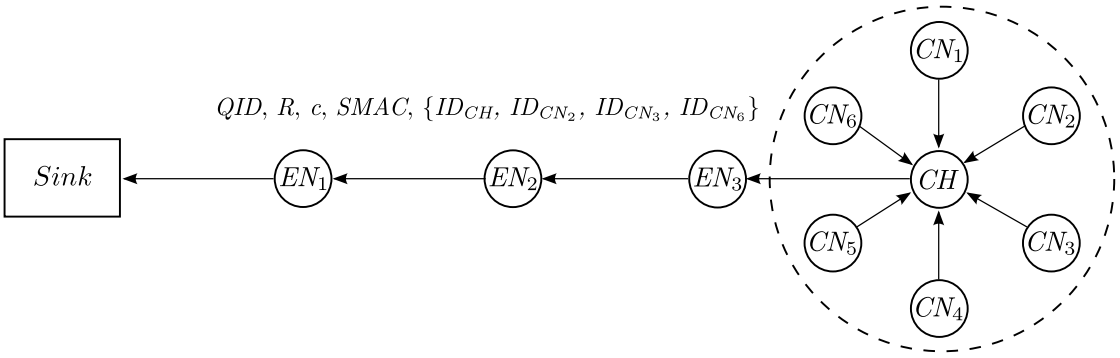

Figure 5.2: Response message sent back from CH to the sink

to one SMAC by bitwise XOR operation. Assume that $t = 3$*, thus SMAC is generated by* $t + 1 = 4$ *different MACs. CH chooses the MACs from* CN_2*,* CN_3*, and* CN_6 *and calculates SMAC which can be later verified by the sink as follows:*

$$SMAC = MAC(R, K_{CH}) \oplus MAC(R, K_{CN_2}) \oplus MAC(R, K_{CN_3}) \oplus MAC(R, K_{CN_6}) \quad (5.3)$$

Next, CH decrypts c from the query message and generates the final report containing QID, R, c, SMAC, its own identifier ID_{CH} *and the identifiers of the t endorsing nodes* ID_{CN_2}*,* ID_{CN_3}*, and* ID_{CN_6}*. Figure 5.2 shows the response message sent by CH to the sink:*

$$CH \rightarrow Sink : QID, R, c, SMAC, \{ID_{CH}, ID_{CN_2}, ID_{CN_3}, ID_{CN_6}\} \quad (5.4)$$

The actions of CH are shown in Algorithm 5.2. To simplify matters, we assume that CH receives $s \geq t$ MACs from CNs, where t is the number of required endorsements.

Algorithm 5.2 CHRepGen(t, K_{CH})

1: **if** received Query is authentic **then**
2: Generate R
3: Send R to all CNs
4: $SMAC = MAC(R, K_{CH})$
5: $V = \{ID_{CH}\}$
6: randomly select t distinct MACs $(MAC_{f_1}, \ldots, MAC_{f_t})$ from s received MACs
7: **for** $i = 1$ to t **do**
8: $SMAC = SMAC \oplus MAC_{f_i}$
9: add node identifier f_i to V
10: **end for**
11: $D(\{c\}_{K_{CH}}, K_{CH}) = c$
12: `Sendto`$(Sink) : QID, R, c, SMAC, V$
13: **end if**

First, CH verifies the authenticity of the query, generates the appropriate report R for the query, and sends R to all CNs. Next, CH calculates $MAC(R, K_{CH})$ and receives

s MACs from *CN*s which agree on the report R. *CH* randomly selects t MACs and compresses them together with its own MAC to one *SMAC*. The node identifiers of *CH* and the t *CN*s whose endorsements have been used to generate *SMAC* are stored in a data structure V. *CH* decrypts the ticket value c and generates the response message consisting of $QID, R, c, SMAC$, and V.

Algorithm 5.3 describes the actions of a *CN* when it receives R from *CH* to endorse (or not to endorse) a report. *CN* first performs its own measurement R'. If R' matches R within the preconfigured error range ε, *CN* generates a MAC and sends it to *CH*.

Algorithm 5.3 CNRepEnd(K_{CN}, R, ε)

1: Generate R'
2: **if** $R' - \varepsilon \leq R \leq R' + \varepsilon$ **then**
3: $MAC_{CN} = MAC(R, K_{CN})$
4: `Sendto`$(CH) : MAC_{CN}$
5: **end if**

En-route Filtering

To protect against PDoS attacks, each en-route node verifies the ticket value included in the response message. This enables false messages to be filtered out immediately at the next hop. Since the verification involves only one hash computation, this can be performed very efficiently. We first continue the examples from above and then present an algorithm which describes the actions of the en-route nodes.

Example 5.4 *The response is forwarded along the reversed query message path, i.e., from CH over EN_3, EN_2, and EN_1 to the sink. Each EN checks if it has stored the appropriate QID and the value c'. If not, the report is dropped. Otherwise, the node checks if $c' \stackrel{?}{=} h(c)$. If the equation holds, the ticket is valid and the message is forwarded to the next hop and the tuple (QID, c') is deleted.*

Algorithm 5.4 describes the actions of an *EN*. First, *EN* checks if its *QList* contains a tuple (QID, c') where the *QID* matches the *QID* in the response message. If there exists such an entry, *EN* verifies if the ticket value c is valid by calculating $c' \stackrel{?}{=} h(c)$. If this is the case, the message is forwarded to the next hop and the tuple (QID, c') is deleted from *QList*. If any of the verifications fail, the message is dropped.

Not considered in the shown algorithm is the case that query or response messages might have been lost during transmission. In this case, *EN* will never receive the response message to a stored tuple and waste memory. To cope with that, an *EN* deletes a tuple if a response message to a query does not arrive within a certain period of time. The length of the time period can be preconfigured and, if necessary, adjusted by the sink by broadcasting new values.

Algorithm 5.4 ENResponse(*QID*, *c*, *QList*)

```
1: if QID ∈ QList then
2:    if c' = h(c) then
3:       forward message
4:       delete (QID, c') from QList
5:    else
6:       drop message
7:    end if
8: else
9:    drop message
10: end if
```

Sink Verification

The sink performs the final verification. We first continue the examples from above and then present an algorithm which describes the actions of the sink.

Example 5.5 *When the sink receives the message, it first checks if QID matches a recently sent query message and if the value c is valid. Next, the sink verifies that the message contains the node identifiers of CH and t CNs and that the CNs are indeed in the local neighborhood of CH and their location matches the location of the report. Finally, the sink verifies the correctness of the SMAC by calculating* MAC'_{CH}, MAC'_{CN_2}, MAC'_{CN_3}, *and* MAC'_{CN_6} *over R using the respective personal keys of CH,* CN_2, CN_3, *and* CN_6. *These MACs are XOR-ed to calculate* $SMAC'$ *which is compared to the received SMAC. If all verifications have passed, the report is accepted. Otherwise, the sink can send out a new query message to another node in this area.*

Algorithm 5.5 describes the actions of the sink to verify a received response message.

Algorithm 5.5 SinkVerify($QID, R, c, SMAC, V$)

```
1: result = reject
2: if QID and c are valid then
3:    if V contains ID_CH and t additional node identifiers then
4:       if location verifications pass then
5:          SMAC' = MAC(R, K_CH)
6:          for i = 1 to t do
7:             SMAC' = SMAC' ⊕ MAC(R, K_CN_i)
8:          end for
9:          if SMAC' = SMAC then
10:            result = accept
11:         end if
12:      end if
13:   end if
14: end if
```

The sink checks if *QID* and c are valid, the set V contains ID_{CH} and t additional node identifiers, and the known locations of these nodes are nearby the stated location of the report. Then, the sink uses the locally stored personal keys of *CH* and the endorsing *CN*s to calculate the reference $SMAC'$ and compares it with the received *SMAC*.

The whole protocol is illustrated in Protocol 5.1. CN_j are all *CN*s of *CH* and CN_i are all nodes that agree on the report and send a *MAC* back to *CH*.

Protocol 5.1 The STEF protocol

1. Initialization:
   - All sensor nodes have a unique identifier and shared key with the sink.
   - Each sensor node has established pairwise keys with its neighbors.
   - The sink is aware of the approximate locations of the sensor nodes.
2. Protocol steps:

1a	*Sink*	$\rightarrow$	$EN^{(1)}$:	$QID, c', Q, \{c\}_{K_{CH}}$
1b	EN:			verify authenticity
1c	EN:			store (QID, c')
1d	EN	$\rightarrow$	CH:	$QID, c', Q, \{c\}_{K_{CH}}$
2a	CH:			generate R
2b	CH	$\overset{*}{\rightarrow}$	CN_j:	send R to all CN_j
2c	CN_j:			generate R'
2d	CN_j:			verify $R' - \varepsilon \leq R \leq R' + \varepsilon$
2e	CN_i:			calculate $MAC_{CN_i} = MAC(R, K_{CN_i})$
2f	CN_i	$\rightarrow$	CH:	send MAC_{CN_i}
2g	CH:			calculate $MAC_{CH} = MAC(R, K_{CH})$
2h	CH:			calculate $SMAC$ using MAC_{CH} and t distinct MAC_{CN_i}
2i	CH:			generate V
2j	CH:			decrypt $D(\{c\}_{K_{CH}}, K_{CH}) = c$
2k	CH	$\rightarrow$	EN:	$QID, R, c, SMAC, V$
3a	EN:			verify $c' \overset{?}{=} h(c)$ related to QID
3b	EN	$\rightarrow$	*Sink*:	$QID, R, c, SMAC, V$
3c	EN			delete (QID, c')
4a	*Sink*:			verify $c' \overset{?}{=} h(c)$ related to QID
4b	*Sink*:			verify V contains ID_{CH} and t additional node identifiers
4c	*Sink*:			verify locations
4d	*Sink*:			calculate $SMAC'$, verify $SMAC' = SMAC$

(1) To simplify matters, *EN* represents all en-route nodes forwarding the messages.
The symbol $\overset{*}{\rightarrow}$ denotes that a message is sent to all CN_j of the cluster.

5.4.2 Confidentiality Enhancement

Some sensor network applications may need confidentiality of the queries and responses. Since en-route nodes do not need to know the contents of Q or R to filter false messages, these values can be encrypted before transmission. In the query message, Q is also encrypted with the shared key between the sink and the *CH*. Thus, the query message has the following form:

$$Sink \rightarrow CH : QID, c', \{Q, c\}_{K_{CH}} \tag{5.5}$$

Before *CH* sends the response to the sink, it encrypts the report R using its pairwise key shared with the sink. For the example shown in Figure 5.2, *CH* sends the following message to the sink:

$$CH \rightarrow Sink : QID, \{R\}_{K_{CH}}, c, SMAC, \{ID_{CH}, ID_{CN_2}, ID_{CN_3}, ID_{CN_6}\} \tag{5.6}$$

This enhancement introduces only marginally increased overhead for the *CH* by performing one symmetric encryption operation. En-route nodes are not affected by this enhancement.

5.5 Security Analysis

In this section, we discuss the security of STEF. We perform our analysis according to our adversary model presented in Section 4.3 and refer to the protocol steps listed in Protocol 5.1. As our work focuses on the security threat of false data injection and PDoS attacks to either deceive the sink by forging events or to waste the scarce energy resources, we first show the efficacy of STEF against these attacks. In addition, we discuss additional attacks an adversary can perform to prevent the sink from receiving a (valid) response message.

5.5.1 Resilience against False Data Injection Attacks to Deceive the Sink

To successfully perform a false data injection attack to deceive the sink, an adversary has to generate a valid response message. The sink accepts only a response message containing a valid ticket value c and a valid $SMAC$ generated by $t+1$ nodes (*CH* and t *CN*s), and where the locations of the sensor nodes match the location stated in R (cf. steps 4a - 4c). Furthermore, the ticket value c must be valid such that *EN*s forward the message towards the sink (cf. steps 3a and 3b). Thus, an adversary has to generate a valid message of the following form (cf. steps 2k and 3b):

$$CH \rightarrow EN \rightarrow Sink : QID, R, c, SMAC, V.$$

and send it along the reversed path that the query message traverses or directly to the sink. In the following, we discuss how the different types of adversaries specified in the adversary model can try to perform a false data injection attack to deceive the sink.

Any **type I adversary (Outsider)** cannot successfully perform a false data injection attack, since he is not in possession of pairwise keys to authenticate himself. Messages from an outside adversary are immediately dropped. To be able to send valid messages to neighboring nodes, an adversary requires valid pairwise keys. Since an outside adversary has no access to keys stored on sensor nodes, he can only analyze overheared messages or misbehave in the execution of the security mechanisms to expose keys. The former violates the assumptions that the applied cryptosystem cannot be broken, the latter the assumption that misbehaving in the execution of the security mechanism does not result in the exposure of keys (cf. Section 4.3).

Now we assume that no security mechanisms are applied to ensure authenticity, integrity, and freshness of the exchanged messages between neighboring nodes. Even then, an outside adversary cannot successfully deceive the sink. We show this by discussing type I.1 and I.2 adversaries separately.

We assume a **type I.1 adversary (Outsider, Mote Class)** who cannot reach the sink directly but is able to send a valid response message which is forwarded hop by hop towards the sink. Thus, according to protocol step 3a

$$EN\text{: verify } c' \stackrel{?}{=} h(c) \text{ related to } QID.$$

this message must contain a valid ticket value c to be forwarded by *EN*s in step 3b. There are two alternatives for the adversary to get in possession of a valid ticket value.

First, we assume that the adversary has intercepted a query message and has calculated c (or a second preimage) from c' or by decrypting the encrypted value of c. The first case violates our assumption that h is a preimage and second preimage resistant one-way function wheras the latter violates the assumption that the adversary is unable to break cryptography.

Second, we assume that the adversary replays a beforehand recorded ticket value from a valid response message. Since a ticket is only valid once (cf. step 3c), the message of the adversary would be immediately dropped. Now we assume that the adversary is able to intercept a currently sent response message while preventing the subsequent *EN*s from receiving this message. In this case, the adversary can use the ticket from the valid response message in his own response message which is then forwarded to the sink. However, the sink would not accept this message since it does not contain a valid *SMAC*. We discuss this in the following.

We assume a **type I.2 adversary (Outsider, Laptop class)** who is able to reach the sink directly[3] and possesses a valid ticket value c. According to protocol step 4d

$$Sink\text{: calculate } SMAC', \text{ verify } SMAC' = SMAC,$$

the response message is only accepted when it contains a valid *SMAC* collaboratively generated by *CH* and t *CN*s (cf. step 4b). Thus, the adversary requires either to forge $t+1$ *MAC*s or convince *CH* and t *CN*s to generate *MAC*s for his false event. Forging *MAC*s violates our assumption that the adversary is unable to break cryptography. Since

[3] We also consider a type I.1 adversary that is only one hop away from the sink.

*CN*s (and *CH* believing to be a *CN* for another *CH*) generate only a *MAC* if the received report R matches their own measurement within a certain error range (cf. step 2d), they do not generate a *MAC* for false events. In case that security mechanisms are applied between neighboring nodes, no *CN* would send a *MAC* to the adversary because the authentication would fail.

Now we consider adversaries of **type II.1.1 and II.1.2 (Insider, Mote Class, $\mathbf{1 \leq m < \mathcal{T}}$ Nodes)**, which have compromised $m < t{+}1$ nodes in one region. We directly discuss the case that the adversary is in possession of a valid ticket value. For example, the sink has chosen one of the compromised nodes as the actual *CH* (cf. step 1a). Assume the adversary has generated a valid response message containing a valid *SMAC* generated with $t{+}1$ distinct personal keys (cf. step 2h). The adversary is in possession of $m < t{+}1$ keys. Thus, he requires $t{+}1{-}m$ additional keys or must guess valid *MAC*s. However, this is in contradiction to our assumption the adversary is unable to break cryptography. Thus, even if a false response message is not filtered out because of an invalid ticket value, the sink will eventually detect the invalid *SMAC* and will not accept the false message (cf. step 4d).

Adversaries of **type II.2.1 or II.2.2 (Insider, Laptop Class, $\mathbf{1 \leq m < \mathcal{T}}$ Nodes)** are just as the type II.1.1 and II.1.2 adversaries unable to deceive the sink since they are unable to generate a valid *SMAC*.

Now we consider adversaries of **type II.1.3 or II.2.3 (Insider, Mote or Laptop Class, $\mathbf{m \geq \mathcal{T}}$ Nodes)**, which have compromised $m \geq t{+}1$ sensor nodes in one region. We distinguish between the case where *CH* is uncompromised and the case where *CH* is one of the compromised nodes.

In the former case, we assume that the adversary has intercepted a currently sent response message and was able to prevent the subsequent *EN*s from receiving this message. Although this attack is difficult to perform and requires exact timing of the adversary, we consider it, since it represents the worst case. As a result of the attack, the adversary is in possession of a valid ticket value and he can inject his own response message which is forwarded towards the sink. In addition, the adversary is able to generate a *SMAC* from $t{+}1$ *MAC*s. However, in step 4b, the sink verifies that the *SMAC* contains the *MAC* generated by *CH*. Since forging this *MAC* is in contradiction to our assumption the adversary is unable to break cryptography, he is not able to successfully deceive the sink.

In the latter case, the adversary can inject one false response message which would be accepted by the sink for each received query. Thus, the number of false response messages the adversary can inject is limited by the number of received queries. In addition, since in step 4c the sink verifies that the location stated in R matches the location of the sensor nodes, the adversary cannot generate reports appearing from arbitrary locations in the WSN. The impact of node compromise is limited to this specific region. Furthermore, if the sink detects that reports originating from this *CH* or region are not plausible, it can easily exclude this *CH* or region by no longer sending queries there. If the adversary does not receive any queries, he cannot successfully inject false messages to deceive the sink even if he has compromised $m \geq t{+}1$ sensor nodes in one region.

5.5.2 Resilience against PDoS Attacks

In this section, we describe the resilience of our protocol against PDoS attacks. Note that PDoS attacks are performed at the application layer (cf. Section 2.2.4 and [191]). STEF does not directly address attacks at lower levels such as denial-of-sleep attacks which prevents a victim node from going into sleep mode.

To perform a PDoS attack an adversary can either try to inject a large amount of false query messages or false response messages. In our work, we concentrate on the latter and rely on existing authentication schemes which support immediate authentication and replay protection to secure query messages. Such authentication schemes exploit the property that the sink cannot be compromised and is not resource constrained. Using such a scheme to protect the query message prevents an adversary of any type from performing a successful PDoS attack. Each node is able to verify that a query message originates from the sink, and immediately filters false queries out. However, an adversary could exploit some inherent properties of the authentication schemes to perform some general attacks which we briefly discuss in the following.

In principle, a sensor node must receive and verify a message to decide if it is valid or not. Thus, an inside adversary may repeatedly send a query message to force the next *EN* to waste energy by receiving (step 1a) and verifying (step 1b) the message. By sending a large amount of such messages, the adversary may drain the energy resources of this node. However, all subsequent sensor nodes are protected since false queries are immediately filtered out. Furthermore, the energy consumption of the verifying node is lower than in an unprotected network since the total energy required for reception and verification is lower than the energy for reception and transmission (see Section 2.3).

The authentication schemes have the property that all nodes in the WSN can verify the authenticity of a query message originating from the sink. When receiving a query for the first time, a sensor node usually accepts and forwards it (unless additional verifications such as invalid timestamps preclude it from doing so). Thus, an adversary could try to distribute a valid query message to a large amount of sensor nodes preparing multiple disjoint return paths. On each path, the adversary could inject one single response message. Hence, in the worst case, the adversary is able to distribute the query once to all nodes of the WSN and inject one response message to all nodes. However, since sensor nodes accept and forward only messages from authenticated neighbors, an adversary must compromise a large number of sensor nodes to be successful. In this case, the adversary has compromised so many nodes, that the WSN is presumably unusable anyway. Even if the adversary could perform this attack without compromising many nodes, the impact of the attack is rather limited since the adversary can distribute each query only once.

The protection against an adversary that tries to perform a PDoS attack by injecting false response messages is realized in STEF with the ticket concept. A valid response message must contain a valid ticket c (cf. steps 2k and 3b). The response message is

forwarded along the reversed path that the query message traverses. Each *EN* has stored a verification tuple (QID, c') and verifies c according to protocol step 3a:

$$EN\text{: verify } c' \stackrel{?}{=} h(c) \text{ related to } QID.$$

before forwarding it (cf. step 3b). After a successful verification *EN* deletes the verification tuple (QID, c') (cf. step 3c) and any replayed message containing an old ticket is immediately filtered out. Messages injected in other areas of the WSN, are likewise filtered out since nodes in this area of the WSN are not in possession of an appropriate (QID, c') tuple. To perform a successful PDoS attack, an adversary would require a large amount of valid tickets since a ticket can be used only once. In the following, we discuss how the different types of adversaries specified in the adversary model could try to perform a PDoS attack.

Any **type I adversary (Outsider)** cannot successfully perform a PDoS attack, since he is not in possession of pairwise keys to authenticate himself. Messages from an outside adversary are immediately dropped. To be able to send valid messages to neighboring nodes, an adversary requires valid pairwise keys. Since an outside adversary has no access to keys stored on sensor nodes, he can only analyze overheared messages or misbehave in the execution of the security mechanisms to expose keys. The former violates the assumptions that the applied cryptosystem cannot be broken, the latter the assumption that misbehaving in the execution of the security mechanism does not result in the exposure of keys (cf. Section 4.3).

We consider an adversary of **type II.1 (Insider, Mote Class)** that has compromised one or more sensor nodes. We first consider the case where neither of the compromised nodes is chosen as *CH*. Thus, the adversary does not receive any ticket values. Furthermore, the adversary cannot find preimages or second preimages and cannot decrypt encrypted messages and thus, is not able to access any valid ticket values currently used in the WSN. As a result, any message sent by the adversary is filtered out at the next hop by the next *EN* (cf. step 3a). All other *EN* behind the filtering node are protected and do not receive any injected message. However, the adversary can attack the filtering *EN* by sending a large amount of false response messages. The filtering *EN* receives and verifies each message from the compromised node. This consumes energy for message reception and ticket verification. However, all other *EN*s on the route behind the filtering *EN* are not affected. Furthermore, the resources of the filtering *EN* decrease slower than in an unprotected network where the node simply forwards messages. The reason is the much lower energy cost for computing hash functions to verify the ticket values compared to the energy required to send the message to the next sensor node. Thus, the energy resources of the attacked *EN* will last much longer than in an unprotected network. We show this in our theoretical performance analysis (see Section 5.6) and in our simulations (see Section 5.7).

Even if one or more of the compromised sensor nodes are chosen as *CH*, the adversary cannot perform a successful PDoS attack. A successful PDoS attack requires the adversary to inject a large amount of messages. In contrast, however, the adversary is only able to sent exactly one message for each received ticket value. Thus, the number

of injected messages depends on the number of queries the sink sends to one of the compromised *CH*. As a result, an adversary cannot increase the global overall normal network traffic. However, he is able to attack the next *EN* as described above.

An adversary of **type II.2 (Insider, Laptop Class)** may be able to use a higher transmission power to send messages to sensor nodes farther away. However, since sensor nodes have no corresponding verification tuple (QID, c'), such messages are immediately dropped. Anyway, sensor nodes accept only messages received from neighboring nodes with which they have established pairwise link keys. Thus, messages are directly discarded at the link layer. An adversary could only try to use two or more compromised nodes to establish a wormhole and attack intermediate nodes. For example, assume the setting shown in Figure 5.1 and that nodes EN_1 and *CH* have established a wormhole and collaborate with the goal to drain the energy resources of nodes EN_2 and EN_3. After *CH* has received a valid query from the sink, it generates a valid response message that EN_3 and EN_2 forward to EN_1. After forwarding, they delete the verification tuple. Thus, no state is saved about already processed responses. EN_1 and *CH* could try to replay the query and the response again and again to drain the energy of EN_2 and EN_3. However, since the authentication scheme used to protect the query message provides replay protection, e.g., by using sequence numbers or timestamps, the replayed query message is dropped by EN_2 and the subsequent nodes (in this example only EN_3) are protected. More promising than this attack are attacks at lower levels, e.g., a denial-of sleep attack by sending packets to all nodes in its communication range. However, as we already stated above, STEF does not address such attacks. Therefore, additional measures must be taken into account.

In this section, we showed the resilience of STEF against PDoS attacks. STEF is independent of any threshold value of compromised nodes, i.e., the filtering power remains, even if an adversary has compromised an arbitrary number of sensor nodes.

5.5.3 Additional Attacks

The STEF protocol is designed to provide protection against false data injection and PDoS attacks to enable a secure report generation and transmission in WSNs which use query-response communication between sink and sensor nodes. In the previous sections, we have discussed the resilience of our protocols against these two attacks. We showed that only a type II.1.3 or II.2.3 (Insider, Mote or Laptop Class, $m \geq \mathcal{T}$ Nodes) adversary who has compromised the actual *CH* can perform a false data injection attack. In addition, we showed that STEF provides resilience against PDoS attack against any type of adversary. However, an adversary can try to perform attacks with the goal of preventing the sink from receiving (valid) reports. In this section, we discuss how an adversary could achieve this by performing one or more of the attacks described in Section 2.2.4. However, we do not address these attacks directly since these are well-known attacks on wireless multihop communication. We discuss these attacks in the context of the STEF protocol for the sake of completeness and briefly introduce possible countermeasures.

To prevent the sink from receiving (valid) reports, an adversary can either disrupt the receiving of messages or invalidate messages.

Message receipt can be disrupted by DoS attacks that prevent query, response, or endorsement messages from arriving at their destination. By performing a jamming attack, an adversary can prevent that a query, response, or endorsement message reaches its destination. Thus, either *CH* cannot generate a valid report since it does not receive sufficient endorsements, a query does not reach *CH*, or the response message does not reach the sink. In either case, the sink does not receive a report. Depending on the number of nodes and the range of the wireless transceiver (Mote Class or Laptop Class adversary), the affected area of the WSN might vary. Jamming attacks are a general problem of all wireless communication systems and can only be mitigated (cf. Section 2.2.4). On the STEF protocol itself, DoS attacks have no negative impact since non-receiving of messages has no negative consequences, e.g., a *CH* is not excluded from the network if the sink does not receive a response message. However, a DoS attack may have negative consequences for the WSN application if for example time-critical alarm messages do not arrive (in time) at the sink.

Alternatively, an adversary can try to perform attacks on the routing protocol to force route changes during a query-response communication. As a result, the message is dropped and does not reach the sink. Also an *EN* can perform a selective forwarding or blackhole attack and drop query or response messages. To cope with attacks on the routing protocol, multipath routing schemes (cf. Section 2.2.4) could be used in combination with STEF. By setting up multiple paths where each query-response is sent along, one can tolerate a certain number of en-route nodes performing a selective forwarding or blackhole attack. Alternatively, protocols based on IDS such as [144, 145] can be used to detect nodes performing selective forwarding or blackhole attacks. However, since *EN*s, that apply the STEF protocol, filter injected false messages out, an IDS may falsely identify a filtering *EN* as a node performing a selective forwarding attack. This issue must be considered when using such protocols.

An adversary can also try to invalidate the report the sink receives. A malicious *EN* can perform a data alteration attack and modify query or response messages before forwarding them. A modification of the query message, however, would be immediately detected by the next *EN* which would drop the message. As a result no response message would be generated. An *EN* can also flip some bit in the response message. As long as *QID* and c have not been modified, the message is forwarded to the sink but not accepted since at least one of the verifications fail. Likewise, a malicious *CH* can send an invalid response message. Multiple path routing and detection and exclusion of malicious nodes are typical countermeasures against such an adversary.

A *CN* is able to invalidate the response message by performing a FEDoS attack. The mechanism used by the STEF protocol to protect against false data injection attacks enables FEDoS attacks. However, this is not a specific problem of STEF; other protocols, e.g., [261, 272, 258, 257], are also susceptible to this attack. In STEF, *CH* receives t endorsements of neighboring *CN*s in form of a MAC. However, *CH* is not able to verify the MACs since they are generated using a shared key known only by the *CN* and the sink. Thus, a malicious *CN* can send a false MAC value to invalidate the collaboratively

generated report. The calculated *SMAC* is invalid, since one of the MACs is false. As a result, the sink does not accept the received report. Due to the XOR operation, the sink is not able to distinguish if the report generating node has tried to perform a false data injection attack or if one of the endorsing nodes has sent a false MAC. As part of this thesis, we developed protocols to address this issue. These protocols are presented in Chapter 6.

5.6 Theoretical Performance Analysis

In this section, we theoretically evaluate the performance of STEF. First, we evaluate the storage requirements on the sensor nodes. Second, we perform an energy saving analysis for the main energy consumers to show how much energy STEF could save by filtering false reports. For the sake of completeness, we finally show the additional overhead for STEF which affects the energy consumption only marginally. In all our analyses the resources of the sink are assumed to be unlimited and are therefore not considered. The results of this analysis are verified by implementing the STEF protocol and performing simulations which is discussed in Section 5.7.

5.6.1 Storage Requirements

The storage requirements introduced by STEF can be divided into two parts: (1) the storage overhead generated directly by STEF, and (2) the overhead generated by usage of existing techniques.

STEF requires each sensor node S_i to store its node identifier ID_{S_i} and the key K_{S_i} shared with the sink. To verify the correctness of received response messages, each sensor node stores the tuple (QID, c') for each query message. These tuples are deleted after the appropriate response message has been forwarded or if the response message does not arrive within a certain period of time.

STEF uses existing authentication and key establishment schemes. The authentication scheme is used to authenticate the query messages sent by the sink. Such a scheme can also be used to authenticate periodical, necessary broadcast messages. Since the storage requirements depend on the particular authentication scheme and usage scenario, we neglect the storage requirements for the authentication scheme in the analysis. Furthermore, each node establishes pairwise keys with its neighboring nodes to authenticate the exchanged messages during the report generation phase.

To quantify the storage requirements let L_n, L_k, L_q, and L_h denote the bit length of a *node identifier*, a *symmetric key*, a *QID*, and a *hash value* respectively. Let the average number of stored (QID, c') tuples be v and the average number of neighbors of a node be w. A sensor node S_i has to store its own *node identifier* ID_{S_i}, its *personal key* K_{S_i}, w pairwise keys shared with its neighbors, and v tuples (QID, c'). Thus, the storage requirements *SR* for a sensor node are:

$$SR = L_n + L_k \cdot (w + 1) + v \cdot (L_q + L_h). \tag{5.7}$$

Example 5.6 *For example, suppose each sensor node has 6 neighboring nodes on average, and has to store 5 (QID, c') tuples on average. The bit length of each key and hash value is 64 bit and the length of a node identifier is 10 bit. A sensor node has to store 6 node identifiers, 7 keys and 5 hash values, resulting in a total of 103.5 byte on average. The Berkeley Mica2 Motes offer 4KB of SRAM. Therefore, the storage requirements of STEF are suitable for current sensor nodes.*

5.6.2 Energy Savings

We analyze how much energy can be saved by filtering false messages using STEF. In our protocol, sensor nodes consume energy in two ways: (1) to send and receive messages, and (2) to verify response messages by computing hash values. Although computing hash values increases energy consumption only marginally [181], we consider the computational overhead, since in STEF, every node verifies each response message.

We use an analysis model similar to that in [261] and [257] to quantify the energy consumption. In addition to the notion used in Section 5.6.1 let L_r and L_m denote the length of a regular report, and a MAC respectively. A response message in STEF contains the regular report (including the node identifier of the *CH*), the *QID*, the ticket c, the *SMAC*, and the identifiers of the t endorsing nodes. Thus, the length of a response message in STEF is $L'_r = L_r + L_q + L_h + L_m + t \cdot L_n$. Let H be the number of hops a report travels, and the normalized legitimate traffic and malicious traffic be 1, and β respectively. We use $e_1 = e_{1s} + e_{1r}$ to denote the energy consumed in sending and receiving 1 byte, and e_2 to denote the energy for one hash computation to verify a response message.

Without STEF, the legitimate traffic and the injected traffic is sent (and received) along all H hops. This results in an energy consumption:

$$E_r = H \cdot L_r \cdot e_1 \cdot (1 + \beta) \tag{5.8}$$

Using STEF, a legitimate response message is never dropped and is forwarded along all H hops. The ticket within this message is verified at each hop. A fabricated response message injected by a compromised node is received from an uncompromised node at the next hop. This node verifies the included ticket and drops the message. The injected message is filtered out after one hop. This results in an energy consumption:

$$E'_r = H \cdot (L'_r \cdot e_1 + e_2) + \beta \cdot (L'_r \cdot e_{1r} + e_2) \tag{5.9}$$

Example 5.7 *Figure 5.3 compares the energy consumption for message transmission with and without STEF, when the length of a regular report is $L_r = 24$ byte, of a QID or* node identifier *is $L_q = L_n = 10$ bit, of a hash value or MAC is $L_h = L_m = 64$ bit, and $H = 100$ Hops. As the overhead affected by the choice of t is only marginal (resulting only in a slightly larger message containing more small node identifiers), we plot the energy consumption only for different values for β and hold t constant to 5 for the sake of simplicity. We use the results presented in [261] to quantify $e_{1s} = 16.25\ \mu J$*

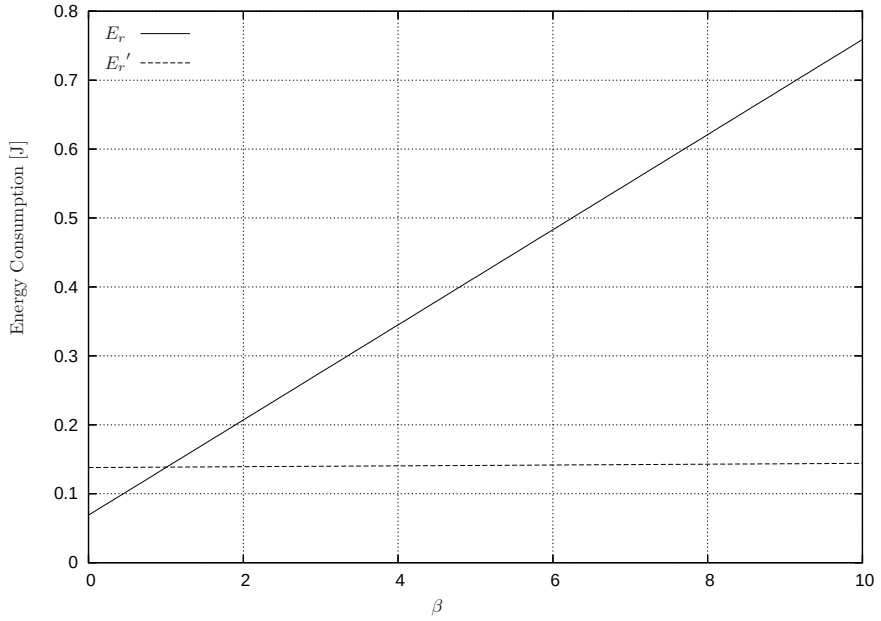

Figure 5.3: Energy consumption depending on the injected traffic ratio β.

for sending, $e_{1r} = 12.5$ μJ for receiving, and $e_1 = 28.75$ μJ for sending and receiving one byte using Mica2 nodes. The energy consumed for performing one hash (or MAC) computation using RC5 [195] block cipher is $e_2 = 15$ μJ and takes about 0.5 ms. We can see that the dashed curve, showing the energy consumption E_r' of STEF, is extremely slowly growing. At the intersection of both curves STEF begins to save energy, i.e., STEF saves energy when the amount of malicious traffic starts to exceed the legitimate traffic and demonstrates increasingly remarkable energy savings. For example, when $\beta = 2$ or 5, STEF saves more than 32% or 65% of energy, respectively.

In most WSN applications, legitimate traffic occurs only when some events of interest appear in the sensor network. In contrast, to increase the impact of their attacks, adversaries often inject a large amount of bogus traffic into the network, which is often several orders of magnitude greater than that of legitimate traffic [261]. STEF is particular useful for these scenarios in saving a great deal of energy by early filtering of false messages.

The average path length also has impact of the energy-saving performance of STEF. The further the false messages are injected away from the sink, the greater the energy savings that can be achieved by STEF. Injected messages distant from the sink are more detrimental than those injected in the sink's vicinity because their transmission involves many more intermediate nodes. Therefore, it is reasonable for an adversary to inject false messages far away from the sink.

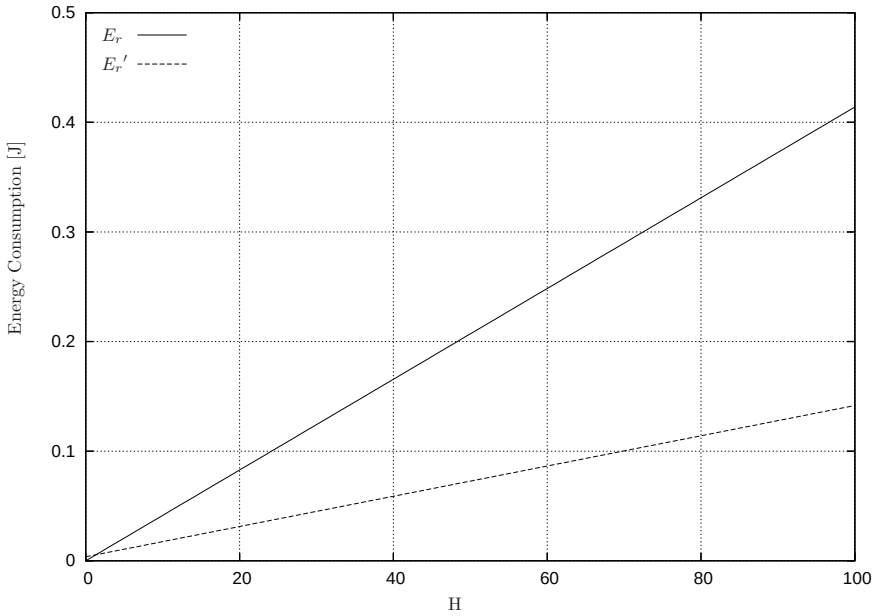

Figure 5.4: Energy Consumption depending on the average number of hops for $\beta = 5$.

Example 5.8 *Figure 5.4 illustrates the impact of the path length for a fixed value of* $\beta = 5$*. The remaining variables are set according to Example 5.7.*

5.6.3 Additional Overhead

STEF introduces some additional overhead which affects the energy consumption only marginally. For the sake of completeness we present this overhead in this section.

When *CH* receives a query message, it generates a report which must be endorsed by t neighboring nodes. Therefore, *CH* sends the report to its w neighbors. If a neighboring node agrees on this report, it endorses the report by generating a MAC as described in Section 5.4.1 and sends it back to *CH*. Assume all w neighboring nodes agree on the report. *CH* has to send one local broadcast message containing the report which all w neighboring nodes receive. Each of the w nodes (and *CH*) computes a MAC on the report. *CH* has to receive w MACs which are sent by the w neighboring nodes. This results in a local communication and computational overhead:

$$E_l = (e_{1s} + w \cdot e_{1r}) \cdot L_r + (e_{1r} + e_{1s}) \cdot w \cdot L_m + (w + 1) \cdot e_2 \tag{5.10}$$

Example 5.9 *Assuming the values from the previous examples this results in an energy consumption of* $E_l = 3.675\ mJ$ *which adds an additional overhead of about 2.6%.*

STEF produces some additional overhead in the query message by adding the value c' and the encrypted value c to the message.

Example 5.10 *Continuing the example from above, this results in an overhead of 128 bit in this message. Therefore, the energy overhead for one node forwarding the query message is 460 μJ which is negligibly small.*

Furthermore, the query message is authenticated using authentication schemes supporting immediate authentication. Such an authentication is necessary in almost all scenarios and is therefore neglected in our analysis. Evaluations of the additional overhead can be found in the respective publications.

5.7 Implementation and Simulation

To perform a deeper analysis of the performance of the STEF protocol, a simulation environment called STEF-Sim has been implemented [169]. STEF-Sim enables the visualization and evaluation of different scenarios with and without STEF. Simulations can be adjusted by a set of parameters to show the impact on the energy and storage consumption of the protocol. The parameters include *number of sensor nodes of the network*, *number of regular query-responses*, *number of compromised nodes*, *behavior of a compromised node*, *required energy for sending and receiving*, *required energy for cryptographic computations*, *transmission range* etc. Sensor nodes and the sink can be generated with the desired properties and placed at the desired locations. Figure 5.5 shows the main window of the simulation environment running a simulation with 200 sensor nodes and one attacking sensor node.

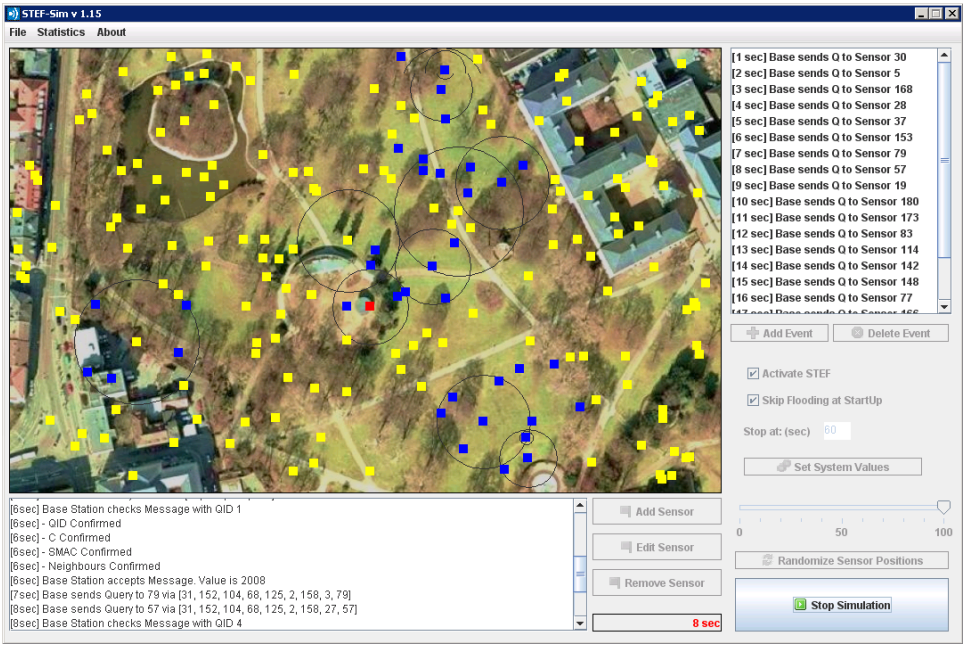

Figure 5.5: Main Window of STEF-Sim

The main focus of the simulation environment is the evaluation of PDoS attacks. Nevertheless, the collaborative report generation is considered in the energy and storage consumption. To evaluate the performance of STEF under active attacks, the adversarial behavior of a sensor node can be adjusted in different ways. The time when a sensor node gets malicious can be set and how a compromised node performs a PDoS attack. Based on the different settings, simulations can be performed with or without the use of STEF to get indications of the effectiveness of STEF. The energy and storage consumption of the whole network and each individual sensor node can be visualized and exported in Gnuplot [95] files.

Simulations have been performed for different scenarios ranging from networks with 200 sensor nodes and one compromised sensor node to networks with 1000 sensor nodes and 100 compromised nodes. Regular traffic is simulated by setting different values for the number of regular queries (and the related responses). Injected traffic is simulated by setting different values for the number of adversarial nodes which inject one message per second. The simulation time is set to 60 seconds and each sensor node is configured with a fixed amount of available energy. Each setting is simulated with STEF activated and STEF deactivated. The simulations have been evaluated and the energy consumptions have been plotted. The evaluations include the number of nodes which died from energy exhaustion, the overall number of sent messages, the overall remaining energy, and the maximum storage requirements. These values have been compared to show the number of dead nodes, the number of filtered messages, and the saved energy. The results have shown that even a single compromised node can cause serious damage and force many sensor nodes to waste energy if STEF is not used.

In Figure 5.6 and Figure 5.7 we illustrate this issue by exemplarily showing simulation results of a network consisting of 200 sensor nodes and one adversarial node. The figures show the energy consumption of sensor nodes 31 and 152. Each node forwards different parts of the regular traffic of 20 queries and responses. Furthermore, they forward injected traffic by the adversarial node. This node is in direct range of node 31 but can reach node 152 only via node 31. Figure 5.6 shows the change of the available energy of nodes 31 and 152 during the simulation when STEF is not used. At the beginning of the simulation, the available energy of each sensor is 1000 mJ. Both sensor nodes receive and forward the same amount of injected messages, but since node 31 forwards more queries and responses than node 152, its energy consumption is higher. Nearly at the end of the simulation time, the energy resources of node 31 are totally exhausted. Figure 5.7 shows the energy of the two nodes if STEF is used. Since node 31 filters the injected messages of the adversarial node, the available energy of node 152 is much higher at the end of the simulation time. Furthermore, since node 31 saves the energy for sending the injected messages, it still has remaining energy at the end of the simulation.

Further information on the developed simulation environment STEF-Sim and the performed simulations can be found in [169].

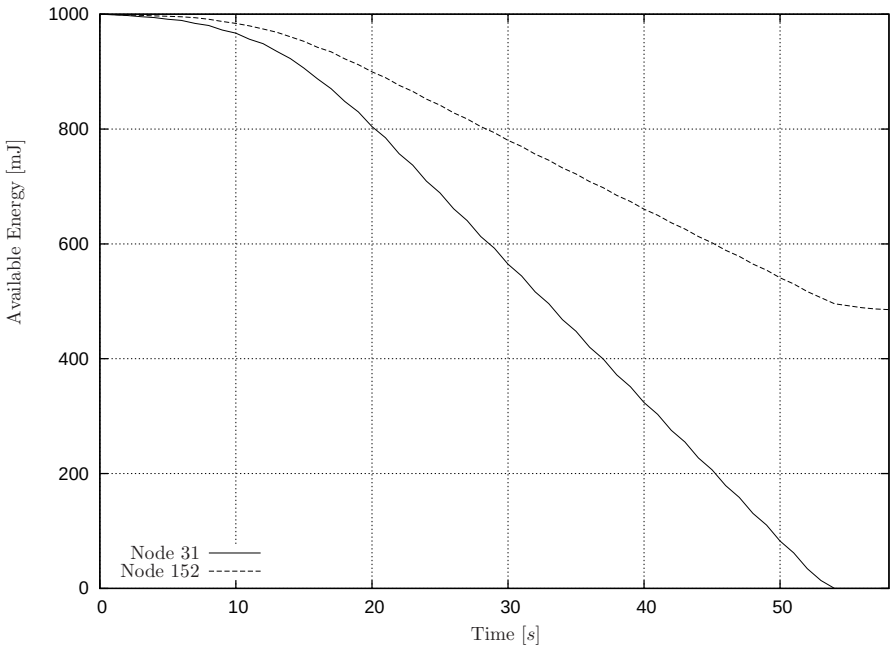

Figure 5.6: Available Energy without STEF

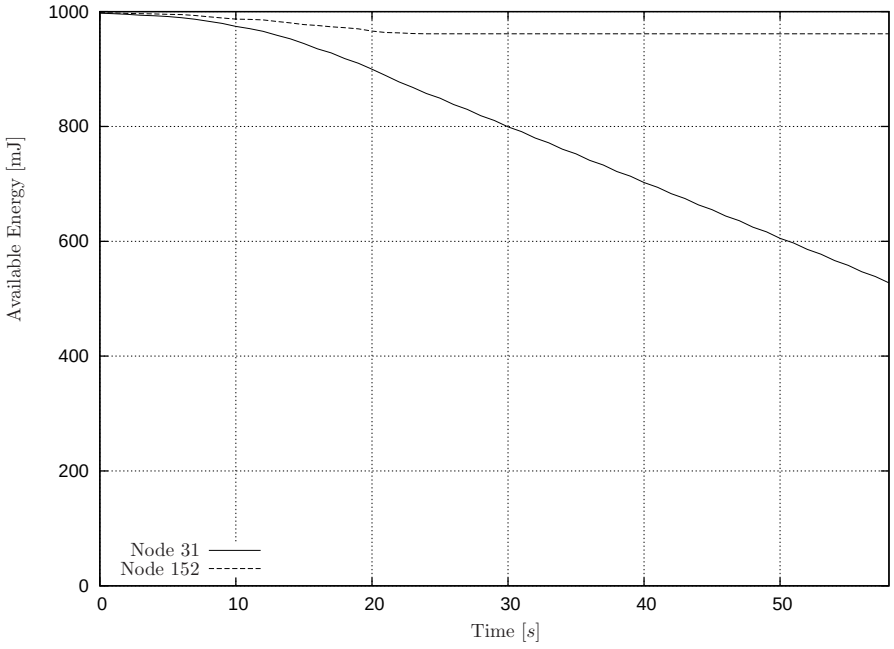

Figure 5.7: Available Energy with STEF

5.8 Summary

In this chapter, we presented STEF, a protocol that applies the second form of recovery by using a mechanism that tolerates compromised nodes and adaptable mechanisms to protect against PDoS and false data injection attacks. Exploiting the typical query-response communication paradigm of many WSNs, STEF achieves recovery from PDoS attacks without relying on a threshold scheme, i.e., the security of STEF is not broken if a certain threshold number of nodes are compromised. To protect against false data injection attacks, a typical threshold-based approach is applied.

The STEF protocol relies only on lightweight cryptographic operations which are applicable in resource constrained WSNs. The ticket concept enables a the filtering of injected false messages of an adversary that tries to perform a PDoS attack. Messages to the sink are only valid if they contain a valid ticket. Each en-route node which forwards a message is able to verify the validity of the ticket and drops the message if the ticket is invalid. Hence, a false message can be filtered out immediately. The ticket concept enables the separation of report generation with sink verification, and the en-route filtering, without the need for symmetric key sharing between sensor nodes. This results in a high resiliency against node compromise. Even if an adversary compromises several nodes, he is not able to inject as many messages as desired to perform a successful PDoS attack because he does not possess the necessary tickets. If a region is under suspicion to be compromised, it can be easily excluded by simply not sending query messages containing valid tickets there. Furthermore, node compromises are limited to the immediate vicinity of the compromised nodes and do not affect the whole network.

The theoretical performance analysis and the simulation results show that STEF is able to significantly reduce the energy consumption by immediate filtering of false reports. STEF's energy savings increase with the number of injected false messages and with the distance to the sink where an adversary injects false messages. Furthermore, the storage requirements in the sensor nodes is very low, and thus, STEF is applicable in high density networks, and leaves room for further security mechanisms.

6 Addressing False Data Injection and FEDoS Attacks

In Section 5.5.3, we have argued that many protocols (including our proposed STEF protocol) are susceptible to *False-Endorsement-Based Denial of Service (FEDoS)* attacks. By compromising a single sensor node, an adversary may be able to invalidate many valid reports.

In this chapter, we present a basic and an enhanced protocol to address both false data injection and FEDoS attacks. The protocols apply a detection strategy by detecting compromised sensor nodes that perform a FEDoS attack. Our protocols do not require intensive monitoring of neighboring sensor nodes to detect the FEDoS attack. After detection, the first form of recovery (see Section 3.1.3) is applied by using mechanisms to exclude the compromised nodes. Immediately after detection, compromised sensor nodes are locally excluded to prevent them from continuing the attack.

This chapter shares some material with previously published work *Defending against False-Endorsement-Based DoS Attacks in Wireless Sensor Networks* [130] and *An Enhanced Scheme to Defend against False-Endorsement-Based DoS Attacks in WSNs* [131].

The chapter is organized as follows. In Section 6.1 we introduce the problem of FEDoS attacks. In Section 6.2, we present related work. Setting and notation are described in sections 6.3 and 6.4. Our proposed basic endorsement protocol is described in Section 6.5. The security of our proposed protocol is analyzed in Section 6.6 and the performance is evaluated in Section 6.7. Next, we describe and analyze our enhanced endorsement protocol in Section 6.8. Implementation and simulation results of both protocols are shown in Section 6.9. Finally, we summarize the chapter in Section 6.10.

6.1 Introduction

In Chapter 5, we have described our proposed STEF protocol and showed in Section 5.5.3 that STEF and many other protocols using similar mechanisms to cope with false data injection attacks are susceptible to *False-Endorsement-Based Denial of Service (FEDoS)* attacks (cf. Section 2.2.4). In a FEDoS attack, an adversary sends a false endorsement to the report generating node, to invalidate the message for the sink. As a result, the sink does not accept this message, although the information of the message is correct. For example, in STEF and the protocols proposed in [272, 269, 257], the adversary can send a false MAC to the report generating node. This MAC is compressed together with the MACs of the other endorsing nodes using bitwise XOR. The sink does not accept this message since the verification of the endorsements fails. Due to the XOR operation, the sink is not able to distinguish whether the report generating node has tried to perform

a false data injection attack or one of the endorsing nodes has sent a false endorsement. Furthermore, the report generating node cannot verify the received endorsements since it does not possess the cryptographic keys to verify the MACs.

The first approach to handle FEDoS attacks has been proposed in [150]. Their probabilistic voting-based filtering scheme (PVFS) is an extension of [261] and requires that all endorsements (MACs) are attached to the message for the sink. The scheme is resistant against FEDoS attacks up to a certain threshold of compromised nodes but is not capable to identify a false endorsing node.

In this chapter, we present a basic and enhanced protocol to cope with FEDoS attacks. They can be used as an extension of STEF or for example the protocols proposed in [272, 269, 257] to cope with all three types of attacks: *false data injection*, *PDoS*, and *FEDoS*. The protocols prevent an adversary compromising less than a certain threshold of sensor nodes, to inject false reports. Furthermore, the protocols enable the report generating node to verify if a neighboring node has sent a false endorsement. Therefore, an endorsing node has to prove at a later point in time, that the sent endorsement was correct. If the proof fails or the node does not perform the proof, the malicious node is locally excluded to prevent further damage by the report generating node and if necessary, a new report for the sink is generated. In addition, our protocols induce a low processing and transmission overhead. Only cheap and fast to perform operations are required, and the local communication overhead is significantly low. In particular, the message sent to the sink is very short since it does not require attaching multiple MACs.

6.2 Related Work

The protocols addressing false data injection and PDoS attacks presented in Section 5.2 are all susceptible to FEDoS attacks.

The first work considering FEDoS attacks is presented by Li et al. in [150]. They refer to the FEDoS attack as *false vote on real event* attack. A probabilistic voting-based filtering scheme is proposed as an extension to the Statistical En-Route Filtering Scheme proposed in [261]. A node endorses a report by sending a vote, which is basically a MAC generated with a symmetric key. Each node is randomly assigned with a certain number of keys from a key pool. The message sent to the sink contains multiple MACs. If an en-route node possesses one (or more) key(s) used to generate a MAC, it can verify the MAC. Two binary sequences are used where the verifying en-route nodes record the total number of verified MACs and the verified valid MACs. If a predefined threshold of invalid MACs has been reached, the message is dropped. Thus, if a malicious node tries to inject a false message, then multiple MACs have to be forged. This will be detected in a probabilistically manner and the message is filtered out on the way to the sink. Furthermore, if a malicious node tries to send a false vote, then the message can still reach the sink, since only one MAC is wrong and the threshold has not been reached. The proposed scheme has the drawback that it requires to attach all MACs to the message sent to the sink. Since this communication involves multiple nodes this

is not very efficient. Furthermore, a distinction between FEDoS or false data injection attack cannot be made and the report generating node cannot detect and exclude a false endorsing node. In addition, the scheme does not consider malicious en-route nodes which try to invalidate the message by rating some MACs as false.

6.3 Setting

We assume the system model described in Chapter 4. In addition, we make the following assumptions.

We assume that our protocols are used in combination with a protocol that addresses PDoS attacks as mentioned above. To simplify matters, we assume that our protocols are used in combination with the STEF protocol presented in Chapter 5.

Furthermore, we assume that the sensor nodes are loosely time synchronized as in μTESLA [181]. To achieve this time synchronization, protocols such as [89, 226, 227, 203] can be used. Time synchronization is necessary in many scenarios to have a link between the measured data and the time of measurement.

The WSN is organized in clusters. One node within each cluster acts as cluster head (*CH*) for the other cluster nodes (*CN*s). The report generation for the sink is initiated by the *CH* by performing the initial measurement of the physical phenomena. In our protocol, this measurement must be endorsed by t neighboring *CN*s which sense the same or similar physical phenomena, where t is a system parameter. The parameter t can be adjusted according to the density of the network, the resistance to node compromise etc. Clusters can either be static (meaning the *CH*s do not change, e.g., as in [133]) or dynamic (meaning the nodes acting as CHs can change during the lifetime of the network, e.g., as in the STEF protocol). Reports can be generated either reactively (by reacting to queries from the sink as in the STEF protocol) or actively if a pre-defined event (e.g., the temperature exceeds a certain threshold) occurs (as in [261]). Broadcast messages from *CH* can be overheard by all *CN*s.

6.4 Notation

We present our protocol by means of a cluster $\mathfrak{C}$ consisting of one cluster head *CH* and several cluster nodes CN_j, $j = 1, \ldots, u$.

Applying a cryptographic *hash function* h on data m is denoted with $h(m)$. A one-way *hash chain* [136] stored on *CH*, or CN_j is denoted with $C^{CH} = c_0^{CH}, \ldots, c_\tau^{CH}$, or $C^{CN_j} = c_0^{CN_j}, \ldots, c_\tau^{CN_j}$ respectively. The hash chain is a sequence of τ hash values, each of fixed length l, generated by a hash function $h : \{0,1\}^l \rightarrow \{0,1\}^l$ by applying the hash function h successively on a seed value c_τ, such that $c_\nu = h(c_{\nu+1})$, for $\nu = \tau - 1, \tau - 2, \ldots, 1, 0$. In other words,

$$c_0 = h^\tau(c_\tau) = \underbrace{h \circ h \circ \ldots \circ h}_{\tau \text{ times}}(c_\tau)$$

As in the STEF protocol, the hash function must provide only preimage and second preimage resistance; collision resistance is not required (compare Sections 3.2.2 and

5.4.1). Note that the hash chains are generated from c_τ to c_0 but used in reversed order. For the sake of simplicity, our protocol is presented by means of such simple hash chains. However, more efficient hash chain constructions should be used, such as Sandwich-chain or Comb Skipchain [110]. These constructions requires less storage space and enable a fast resynchronization, i.e., a verifier can verify a chain value on a distant trusted chain value efficiently without requiring to apply the hash function on each single intermediate hash value.

6.5 Endorsement Protocol

Our proposed protocol enables a collaborative report generation. It is resistant against false data injection up to *t+1* compromised sensor nodes and is resistant against compromised *CN*s which perform a FEDoS attack. By using our protocol as an extension to STEF or one of the protocols presented in [272, 269, 257], one obtains the resistance against PDoS attacks.

The main idea of our protocol is that after a certain time span an endorsing *CN* must prove to *CH* that the sent endorsement was correct. If *CH* detects that one or more *CN*s do not prove the correctness of their endorsement or the prove fails, then it excludes this node(s) from its communication. If required, a new report for the sink can be generated without the malicious node(s). All subsequent reports are also generated without the malicious node(s).

Endorsements are generated by using values of a hash chain stored on each node. A value is only valid for a certain time span. Like in μTESLA [181], the hash values are disclosed at a later point in time, which enables *CH* to verify the previously received endorsements. Since the hash values are invalid at this point in time, a malicious *CH* cannot misuse them to inject a false report.

The protocol is divided in three phases, *Initialization*, *Report Generation*, and *Verification*. The initialization phase is performed only once whereas report generation and verification phase are always performed when a report for the sink is generated.

6.5.1 Initialization

The initialization phase is performed to configure the sensor nodes before deployment, and to execute some initialization procedures directly after deployment. This phase is assumed to be secure as mentioned above.

CH and $CN_1, \ldots, CN_u$ are assigned with a unique identifier ID_{CH} and $ID_{CN_1}, \ldots, ID_{CN_u}$, and are preloaded with a *hash chain* C^{CH} and $C^{CN_1}, \ldots, C^{CN_u}$.

After the nodes are deployed, each sensor node establishes pairwise keys with its one-hop neighbors using some existing protocols and exchanges the verification values c_0^{CH} and $c_0^{CN_1}, \ldots, c_0^{CN_u}$, respectively. Since we assume that our protocol is used in combination with the STEF protocol where each sensor node could be chosen as a *CH*, all nodes store the verification values of all other nodes in the cluster and establish pairwise keys with all other nodes in the cluster. In scenarios with a static setting where

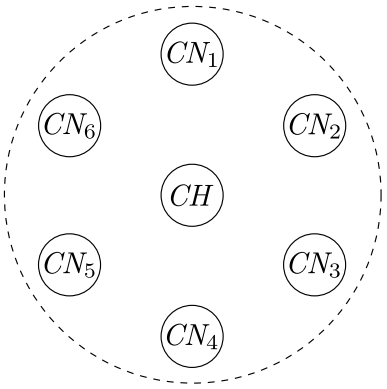

Figure 6.1: Cluster $\mathfrak{C}$ consisting of CH, CN_1, CN_2, CN_3, CN_4, CN_5, and CN_6

CH does not change, it is sufficient that only pairwise keys are established between CH and each CN, and that only CH stores the verification values of all CNs.

Adding new nodes at a later point in time is not integral part of our proposed protocols. However, an additional mechanism similar to Kerberos (cf. related work in Section 3.2.8) where the sink acts as a trusted intermediary could be used to distribute verification values and to establish pairwise keys. The sink stores all hash values of all hash chains and is able to distribute the required verification values for the specific time to newly deployed sensor nodes and their neighbors. Likewise, pairwise keys can be established.

Example 6.1 *Figure 6.1 shows a cluster $\mathfrak{C}$ consisting of CH and CN_1, CN_2, CN_3, CN_4, CN_5, and CN_6. The nodes have exchanged the verification values and have established a symmetric key with each other node. Thus, CN_1 stores c_0^{CH}, $c_0^{CN_2}$, $c_0^{CN_3}$, $c_0^{CN_4}$, $c_0^{CN_5}$, $c_0^{CN_6}$ and the pairwise keys. According to this, CH and the other CNs store verification values and keys.*

6.5.2 Report Generation

CH performs the initial measurement of the physical phenomena and initiates the report generation. However, CH cannot generate a valid report for the sink by itself. CH requires the help of t neighboring CNs which endorse the measurement. An endorsement is only sent from a CN if it agrees on the measurement within a certain error range.

An endorsement is generated by using the values of the hash chain stored on each CN. A hash value is only valid within a certain time interval. For this purpose, we introduce time intervals I_λ, $\lambda = 1, \ldots, \tau$. In interval I_λ, the hash values $c_\lambda^{CN_j}$ of each CN_j are valid to endorse a report. Thus, in interval I_1 hash values $c_1^{CN_j}$, in interval I_2 hash values $c_2^{CN_j}$, and so on, are used to endorse a report. CH uses the values c_λ^{CH} from its own hash chain to generate its own "endorsement". The parameter τ can be either adjusted to the lifetime of the WSN or the size of the hash chains. In the latter case, new hash chains need to be generated which are valid from interval $I_{\tau+1}$.

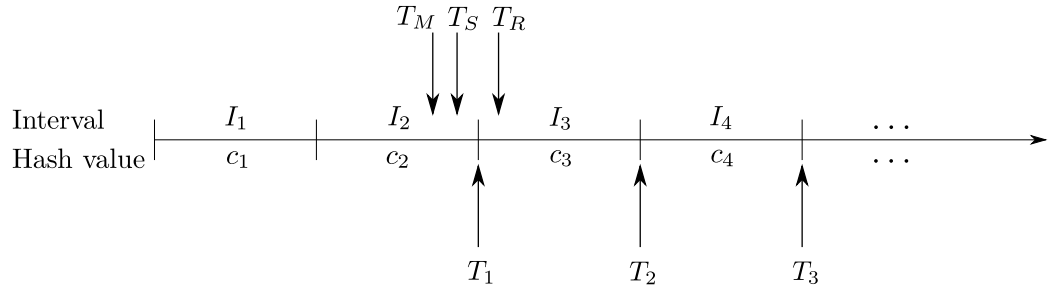

Figure 6.2: Exemplary time bar: *CH* performs a measurement at time T_M in interval I_2. The report generation ends by sending the report message to the sink at time T_S. This message is received by the sink at time T_R in interval I_3. The hash values c_2, which are used to generate the endorsements in interval I_2 are disclosed at time T_2 at the beginning of interval I_4.

As in μTESLA [181], the hash values are disclosed at a later point in time to enable the detection of false endorsing *CN*s in the verification phase.

For the sake of simplicity, we assume that the whole report generation is completed before the end of one interval. This is the common case, since our protocol only involves local one-hop communications and operations are cheap and fast to execute. If *CH* detects that the interval is nearly at its end, *CH* waits until the beginning of the next interval. Furthermore, we assume that the used hash chains last for the whole lifetime of the network, i.e., in the last interval I_τ the seed values c_τ^{CH} and $c_\tau^{CN_j}$ are valid.

First, we explain the report generation by continuing Example 6.1. Afterwards, we describe the detailed protocol steps in general by means of two algorithms describing the actions of *CH* and the *CN*s.

Example 6.2 *We consider the cluster shown in Figure 6.1 and the chronological order shown in Figure 6.2. The report generation is initiated by CH in interval I_2 with the measurement of the physical phenomena at time T_M. We set $t = 2$, i.e., CH requires endorsements from two CNs to generate a valid report for the sink.*

At time T_M, CH performs a measurement of some physical phenomena and generates the related report R and associates the time of measurement T_M with the report. CH sends a local broadcast message[1] to all $CN_j, j = 1, \ldots, 6$ in the cluster containing R and T_M

$$CH \xrightarrow{*} CN_j : R, T_M \tag{6.1}$$

which is also shown in Figure 6.3.

After a CN has received this message, it checks the interval of validity of T_M by verifying if the measurement has been performed in the current interval I_2, and if the report R received from CH matches its own measurement R' within a certain range ε. We assume that these verifications fail on nodes CN_5 and CN_6. However, on nodes CN_1, CN_2, CN_3, and CN_4 the verifications pass, i.e., these nodes generate an endorsement. CN_1, CN_2, and CN_3 calculate

$$End_{CN_i} = h(c_2^{CN_i} || R || T_M) \tag{6.2}$$

[1]The symbol $\xrightarrow{*}$ denotes a local broadcast message.

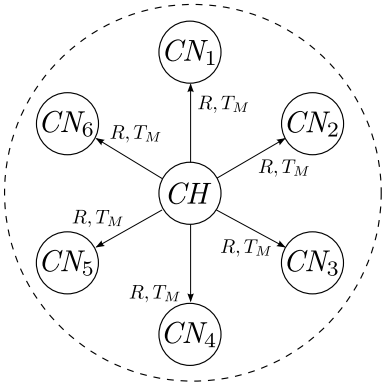

Figure 6.3: CH broadcasts R and T_M to all $CN_j, j = 1, \ldots, 6$.

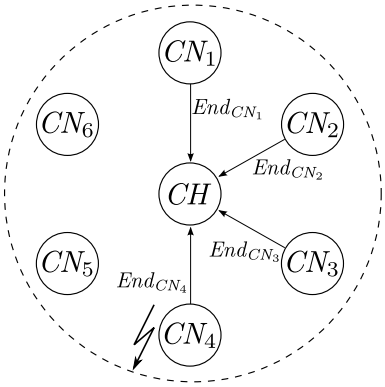

Figure 6.4: CN_1, CN_2, CN_3, and CN_4 send endorsements to CH. The endorsement of CN_4 is invalid.

for $i = 1, \ldots, 3$ and send endorsements End_{CN_i} to CH. We assume that CN_4 is compromised and tries to perform a FEDoS attack by sending $End_{CN_4} \neq h(c_2^{CN_4}||R||T_M)$. Figure 6.4 illustrates this. CH temporarily stores all received endorsements for future verification purposes.

Likewise, CH calculates End_{CH}. It chooses two of the four received endorsements, e.g., End_{CN_1} and End_{CN_4}, and calculates

$$SEnd = End_{CH} \oplus End_{CN_1} \oplus End_{CN_4}. \tag{6.3}$$

Then, CH sends the following message to the sink

$$R, T_M, SEnd, \{ID_{CH}, ID_{CN_1}, ID_{CN_4}\} \tag{6.4}$$

containing the report R, the time of measurement T_M, the compressed endorsements $SEnd$, and the set of node identifiers of the endorsing nodes.

In Algorithm 6.1 and Algorithm 6.2, we describe the general protocol steps of *CH* and the *CN*s in detail.

Algorithm 6.1 shows the actions of CH. First, CH generates R and T_M and sends these values to all CN_j. Each CN_k where the verifications pass sends an endorsement back to CH. The remaining CNs do not send an endorsement. CH maintains a set F, containing all node identifiers of trusted CN_j whose endorsements are accepted. Directly after the initialization phase, F contains the node identifiers of all CN_j in the cluster $\mathfrak{C}$. Each endorsement, received from a trusted CN_k, is temporarily stored for future verification. CH calculates $h(c_\lambda^{CH}||R||T_M)$, selects t endorsements it has received from trusted CNs, and compresses them to one $SEnd$. The node identifiers of CH and the t CNs whose endorsements have been used to generate $SEnd$ are stored in a data structure V. The final message to the sink consists of R, T_M, End, and V.

Algorithm 6.1 CHRepGen(t, c_λ^{CH})

1: Generate R and T_M
2: Send R, T_M to all CNs
3: **while** Receiving Endorsements **do**
4: **if** `getID`(End_{CN_k}) $\in F$ **then**
5: store End_{CN_k}
6: **end if**
7: **end while**
8: $End_{CH} = h(c_\lambda^{CH}||R||T_M)$
9: $End = End_{CH}$
10: $V = \{ID_{CH}\}$
11: randomly select t distinct endorsements $End_{f_1}, \ldots, End_{f_t}$
12: **for** $i = 1$ to t **do**
13: $End = End \oplus End_{f_i}$
14: add node identifier f_i to V
15: **end for**
16: `Sendto`($Sink$) : R, T_M, End, V

Algorithm 6.2 describes the actions of a CN_j when it receives R and T_M from CH to endorse (or not to endorse) a report. CN_j first checks if the measurement has been performed in the current interval. Next, it performs its own measurement R'. If R' matches R within a certain error range ε, CN_j generates an endorsement and sends it to CH.

Algorithm 6.2 CNRepEnd($c_\lambda^{CN_j}$, I_λ, R, T_M, ε)

1: **if** $T_M \in I_\lambda$ **then**
2: Generate R'
3: **if** $R' - \varepsilon \leq R \leq R' + \varepsilon$ **then**
4: $End_{CN_j} = h(c_\lambda^{CN_j}||R||T_M)$
5: `Sendto`(CH) : End_{CN_j}
6: **end if**
7: **end if**

6.5.3 Verification

The verification phase is twofold. In the *Sink Verification* the sink verifies that *CH* and t of the neighboring CN_j have collaboratively generated the report. This prevents an adversary from performing a false data injection attack to deceive the sink, if he has compromised less than *t+1* nodes. The *CH Verification* enables *CH* to verify if one or more CN_k have sent a false endorsement in the report generation phase.

Sink Verification

Since the sink knows all values of each node's hash chain, it can easily verify the received report. We first continue the examples from above and then present an algorithm which describes the actions of the sink.

Example 6.3 *The sink receives the message at time T_R in interval I_3 (see Figure 6.2). First, the sink verifies that the message is fresh, i.e., $R, T_M, SEnd, V$ has not been replayed. Then the sink verifies that the message contains the identifier of CH and t=2 additional node identifiers. The verification passes since the message contains the set of identifiers $\{ID_{CH}, ID_{CN_1}, ID_{CN_4}\}$. Next, the sink identifies the interval I_λ in which the message was generated by means of T_M; in this case I_2. Now the sink verifies that the used hash values c_2^{CH}, $c_2^{CN_1}$, and $c_2^{CN_4}$ have not been disclosed yet by verifying that*

$$T_R + T_\delta < T_2 \tag{6.5}$$

where T_δ is the maximum synchronization error and T_2 is the time where the hash values used in interval I_2 are disclosed (see Figure 6.2). This verification also passes and thus, the sink verifies SEnd by calculating

$$SEnd\,' = h(c_\lambda^{CH}||R||T_M) \oplus h(c_\lambda^{CN_1}||R||T_M) \oplus h(c_\lambda^{CN_4}||R||T_M) \tag{6.6}$$

and compares $SEnd\,'$ with the received SEnd

$$SEnd\,' \stackrel{?}{=} SEnd. \tag{6.7}$$

Since CN_4 has sent a false endorsement, this verification fails and the sink does not accept the report. The sink is not able to distinguish whether a compromised CN has sent a false endorsement or a compromised CH has tried to perform a false data injection attack by guessing some endorsements. In contrast, however, CH is able to verify the received endorsements and to detect the false endorsing CN_4 which CH excludes from any of its further report generations. Since the protocol is used in combination with the STEF protocol, reports are generated reactively and the sink sends a new query to CH. The new report generation without the malicious CN_4 is initiated after CH has performed the CH Verification. If our protocol is used in scenarios where reports are generated actively, CH autonomously initiates a new report generation after detecting that a false endorsement has been used to generate the message for the sink.

Algorithm 6.3 describes the actions of the sink to verify the received message. V denotes the set of the node identifiers of *CH* and the t endorsing nodes. The first verification ensures that the message is fresh using T_M as a unique timestamp for R and V. Then it is verified that V contains the identifier of *CH* and t additional node identifiers. Next, the function `getInterval` returns the interval in which the report was generated. The following verification ensures, that the hash values used to generate the report have not been disclosed yet. If the verifications pass, the reference $SEnd'$ is calculated using the hash values stored at the sink. The function `getHash` returns the corresponding hash values from the memory of the sink. Finally, the calculated $SEnd'$ is compared with the received *SEnd* and if they are equal, the report is accepted.

Algorithm 6.3 SinkVerify($R, T_M, SEnd, V, t, T_R, T_\delta, T_\lambda$)

```
1:  result = reject
2:  if R, T_M, V fresh then
3:    if V contains ID_CH and t additional node identifiers then
4:      I_λ = getInterval(T_M)
5:      if T_R + T_δ < T_λ then
6:        for i = 0 to t do
7:          c_λ = getHash(V[i], I_λ)
8:          SEnd' = h(c_λ||R||T_M) ⊕ SEnd'
9:        end for
10:       if SEnd' = SEnd then
11:         result = accept
12:       end if
13:     end if
14:   end if
15: end if
```

CH Verification

To detect false endorsing CN_k, a second verification is performed when the endorsing CN_k discloses their used hash values $c_\lambda^{CN_k}$. The values are disclosed at time $T_\lambda = (\lambda + \Delta) \cdot T_L$ where T_L denotes the length of an interval and Δ specifies the delay before the hash values are disclosed. To prevent a powerful adversary from performing a false data injection attack, Δ should be adjusted in such a way that the sink has already received the message before the hash values are disclosed. This prevents an adversary from collecting hash values to generate a false message and transmitting it to the sink using a faster communication channel, e.g., a wormhole [122]. If a CN_k does not disclose the used hash value, it is assumed that the node has been compromised and is excluded by *CH* from any further report generation. In this case, a new message for the sink is generated by performing the report generation again, but without the malicious CN.

We first continue our examples and afterwards present an algorithm which describes the actions of *CH* to detect a FEDoS attack.

Example 6.4 *At time T_2 (see Figure 6.2) CN_1, CN_2, CN_3, and CN_4 disclose the hash values $c_2^{CN_1}$, $c_2^{CN_2}$, $c_2^{CN_3}$, and $c_2^{CN_4}$. First, CH checks if the hash values are valid by verifying for $k = 1, \ldots, 4$*

$$h(c_2^{CN_k}) \stackrel{?}{=} c_1^{CN_k}. \tag{6.8}$$

If CH has missed some of the previously disclosed hash values or no reports have been generated and therefore, some hash values have not been disclosed, CH can apply the hash function multiple times to resynchronize; i.e., given a trusted value c_p, it can verify the validity of c_q, where $q > p$, by verifying that $H^{q-p}(c_q) = c_p$. As mentioned in Section 6.4, we propose to use efficient hash chain constructions which enable efficient resynchronization without the need to apply the hash function $q - p$ times. Especially if $q - p$ is large, this is important to save energy and to enable fast verifications.

We assume that all endorsing nodes disclose the correct hash value (even the false endorsing node CN_4). Since CH has stored all received endorsements in the report generation phase, it is able to verify them now. CH calculates for $k = 1, \ldots, 4$

$$End\,'_{CN_k} = h(c_2^{CN_k}||R||T_M) \tag{6.9}$$

and compares it with the already stored End_{CN_k}, $k = 1, \ldots, 4$

$$End\,'_{CN_k} \stackrel{?}{=} End_{CN_k} \tag{6.10}$$

The verification of End_{CN_4} fails, since CN_4 has sent a false endorsement in the report generation phase. Thus, CH excludes CN_4 from any of its future report generations. Since the message sent to the sink contains the endorsement of CN_4, a new message for the sink needs to be generated. Since the protocol is used together with the STEF protocol, CH waits for a new query from the sink. If reports are generated actively, CH autonomously starts a new report generation.

If the endorsement from CN_4 has not been used in the generation of the SEnd, the fraud is anyway detected by CH but no new report needs to be generated.

Algorithm 6.4 describes the verifications of CH. Before the algorithm is executed, CH removes all CN_k which do not disclose the hash values used to generate an endorsement from the set F. If a CN is removed from F, CH can delete the pairwise key and the verification value of this CN to save memory space. If an endorsement of any of these CN_k has been used to generate $SEnd$, a reaction is necessary since presumably the report has been invalidated by these nodes. In case that only query-response communications between sink and CH is used, CH waits for a new query from the sink. In an active setting, CH initiates a new report generation. For each of the remaining CNs Algorithm 6.4 is executed.

First, the disclosed hash values are verified on their validity. If the verification fails, CN_k is removed from F. In case that an endorsement of one of these nodes has been used to generate the report message, appropriate reactions as described above are performed. If the verification passes, CH calculates the endorsement of CN_k and compares it with the temporarily stored endorsement, received in the report generation phase. Again, if this verification fails CN_k is removed from F and if necessary appropriate reactions are performed.

Algorithm 6.4 CHVerify$(R, T_M, V, ID_{CN_k}, End_{CN_k}, c_\lambda^{CN_k}, c_{\lambda-1}^{CN_k}, G)$

```
1: if h(c_λ^{CN_k}) ≠ c_{λ-1}^{CN_k} then
2:    F = F \ ID_{CN_k}
3:    if ID_{CN_k} ∈ V then
4:       reaction
5:    end if
6: else
7:    End'_{CN_k} = h(c_λ^{CN_k} || R || T_M)
8:    if End'_{CN_k} ≠ End_{CN_k} then
9:       F = F \ ID_{CN_k}
10:      if ID_{CN_k} ∈ V then
11:         reaction
12:      end if
13:   end if
14: end if
```

In the listing of Protocol 6.1, we summarize the whole protocol for our cluster $\mathfrak{C}$. If the role of CH changes, it is required that all nodes establish pairwise keys and exchange the verification values. Furthermore, we show only the relevant aspects of our protocol. To use the protocol in combination with the STEF protocol additionally requires that each node stores a symmetric key shared with the sink. The possibly required reaction depends on the setting. In a reactive setting (as in the STEF protocol), reactions are initiated by the sink by sending new queries. In an active setting, CH initiates a reaction by initiating a new report generation after detecting that a FEDoS attack has been performed and the endorsement has been used to generate the sink message.

The listing contains the respective numbers of the protocol steps, the interval or point in time where a protocol step is performed, the involved sensor nodes, and the transmitted message or the performed action.

Protocol 6.1 Protocol to detect FEDoS attacks

1. Initialization:
   - Each sensor node has a unique identifier.
   - Each sensor node stores a hash chain.
   - CH has established pairwise keys with its neighboring CN_j.
   - CH stores verification value(s) $c_0^{CN_j}$ for each CN_j.
   - CH maintains a set F of trusted CN_j.

2. Protocol steps:

1a	$T_M \in I_\lambda$	CH:			generate R
1b	I_λ	CH	$\xrightarrow{*}$	CN_j:	R, T_M
1c	I_λ	CN_j:			verify T_M
1d	I_λ	CN_j:			generate R'
1e	I_λ	CN_j:			verify $R' - \varepsilon \leq R \leq R' + \varepsilon$
1f	I_λ	CN_k:			calculate $End_{CN_k} = h(c_\lambda^{CN_k} \| R \| T_M)$
1g	I_λ	CN_k	$\rightarrow$	CH:	End_{CN_k}
1h	I_λ	CH:			store all End_{CN_k} to R, T_M
1i	I_λ	CH:			calculate $End_{CH} = h(c_\lambda^{CH} \| R \| T_M)$
1j	I_λ	CH:			calculate $SEnd$ using End_{CH} and t End_{CN_k}
1k	I_λ	CH:			generate V
1l	$T_S \in I_\lambda$	CH	$\rightarrow$	$Sink$:	$R, T_M, SEnd, V$
2a	T_R	$Sink$:			verify message is fresh
2b	T_R	$Sink$:			verify V contains ID_{CH} and t additional node identifiers
2c	T_R	$Sink$:			verify hash values have not been disclosed
2d	T_R	$Sink$:			calculate $SEnd\,'$, verify $SEnd\,' = SEnd$
2e	T_R	$Sink$:			if necessary perform reaction
3a	T_λ	CN_k	$\rightarrow$	CH:	$c_\lambda^{CN_k}$
3b	T_λ	CH:			remove non-disclosing CN_k from F
3c	T_λ	CH:			if $h(c_\lambda^{CN_k}) \neq c_{\lambda-1}^{CN_k}$ remove CN_k from F
3d	T_λ	CH:			calculate all $End\,'_{CN_k}$, if $End\,'_{CN_k} \neq End_{CN_k}$ remove CN_k from F
3e	T_λ	CH:			if necessary perform reaction

The symbol $\xrightarrow{*}$ denotes that a message is sent to all CN_j of the cluster.

6.6 Security Analysis

In this section, we discuss the security of our protocol. We perform our analysis according to our adversary model presented in Section 4.3 and refer to the protocol steps listed in Protocol 6.1.

Since our protocol focuses on the security threats of false data injection attacks and FEDoS attacks to either deceive the sink by forging events or to invalidate the collaboratively generated report, we show the efficacy of our protocol against such attacks in this section. In addition, we note the impact of the STEF protocol in the relevant text passages since we assume that our protocol is used in combination with the STEF protocol presented in Chapter 5. Furthermore, we discuss additional attacks an adversary can perform to disrupt the functionality of our protocol.

6.6.1 Resilience against False Data Injection Attacks

To successfully deceive the sink, an adversary has to generate a valid response message. The sink accepts only a fresh response message containing a valid *SEnd* generated by *t+1* nodes (*CH* and *t CN*s) where the used hash values to generate *SEnd* have not been disclosed yet (cf. steps 2a -2e). Thus, an adversary has to generate a response message according to step 1l:

$$T_S \in I_\lambda,\ CH \rightarrow Sink : R, T_M, SEnd, V$$

containing report R, time of measurement T_M, *SEnd*, and the set V containing the identifiers of *CH* and the *t CN*s. *SEnd* must be generated using only hash values valid for the current interval I_λ. In the following, we discuss how the different types of adversaries specified in the adversary model can try to perform a false data injection attack to deceive the sink.

Assume that a **type I adversary (Outsider)** successfully generates a valid response message in interval I_λ for a false event. Thus, the adversary must be in possession of *t+1* valid endorsements for this event (cf. step 1j). To get *t+1* valid endorsements the adversary can try to convince neighboring sensor nodes to send valid endorsements for the false event, misuse old messages, or try to generate valid endorsements by himself.

In the first case, the adversary sends his false event to neighboring sensor nodes (cf. step 1b) to get endorsements from these nodes. However, this violates our assumption that neighboring sensor nodes send only endorsements to nodes which authenticate themself using the pairwise keys established in the initialization phase. Even if no authentication mechanisms are applied, the adversary would not receive any endorsements from neighboring nodes. An uncompromised sensor node first verifies the validity of the event (cf. step 1c and 1e) before generating an endorsement.

In the second case, the adversary eavesdrops on transmitted messages and then tries to misuse them. The adversary can record a valid response message sent in step 1l and replay it at a later point in time to make the sink believe an event happened later on again. However, since the sink keeps record of received messages, a message containing the same timestamp T_M for R and V is not accepted again (cf. step 2a). Alternatively,

the adversary can record endorsements sent in step 1g and misuse them to generate a new response message. However, endorsements are specific for R and T_M (cf. step 1f) and only valid as long as the hash values has not been disclosed (cf. step 2c). Thus, to inject a message describing a false event (within this interval I_λ), the adversary would have to invert the hash function to get in possession of a valid hash value. However, this is in contradiction to our assumption that this is not possible (cf. Section 4.3). Finally, the adversary can try to misuse disclosed hash values to generate endorsements for his false event. However, the sink accepts only a message if the used hash values have not been disclosed (cf. step 2c). Since we assume that sensor nodes are loosely time synchronized (cf. Section 6.3), the adversary cannot misuse these hash values.

In the latter case, the adversary needs a hash value valid for the current interval to generate a valid endorsement by himself. As stated above, he cannot extract a valid hash value from an endorsement. The only alternative is to calculate preimages to already disclosed hash values which is a contradiction to our assumption that the hash function provides preimage and second preimage resistance.

In addition, since we assume that the protocol is used in combination with the STEF protocol, false injected messages from a type I.1 adversary (Outsider, Mote Class) are immediately filtered out at the next hop since the message does not contain a valid ticket value (compare Section 5.5.1).

Now we consider **type II.1.1, II.1.2, II.2.1 and II.2.2 adversaries (Insider, Mote or Laptop Class,** $1 \leq m < \mathcal{T}$ **Nodes)**, i.e., an adversary has compromised $m < t{+}1$ nodes. The adversary is in possession of $m < t{+}1$ valid hash values and can generate m valid endorsements. However, to generate a valid *SEnd*, $t{+}1{-}m$ additional endorsements are required since *SEnd* must be generated by $t{+}1$ different nodes using the appropriate values of their hash chains (cf. step 2b). As mentioned in the discussion of type I adversaries, an adversary cannot generate these endorsements since this would be in contradiction to our assumptions. Thus, the adversary is unable to generate a valid *SEnd*. Furthermore, the applied STEF protocol, requires the adversary to compromise the actual *CH* since the response message must include a valid ticket value.

A **type II.1.3 or II.2.3 adversary (Insider, Mote or Laptop Class,** $m \geq \mathcal{T}$ **Nodes)** has compromised $m \geq t{+}1$ sensor nodes. Using the hash values stored on the compromised sensor nodes, the adversary can generate endorsements and calculate a valid *SEnd* for a false event. Thus, the adversary can generate a valid response message according to step 1l and deceive the sink. However, the impact of these node compromises is mitigated by the STEF protocol. It prevents an adversary from performing PDoS attacks and limits the node compromise to the region of the compromise, i.e., an adversary cannot inject false reports appearing from arbitrary locations. In addition, the adversary can only inject messages if one of the compromised nodes is choosen as *CH* (compare Section 5.5.1).

6.6.2 Resilience against FEDoS Attacks

A FEDoS attack can be only performed by endorsing *CN*s which send false endorsements to *CH* to invalidate the collaboratively generated report for the sink. The goal of our proposed protocol is to detect and exclude a *CN* performing a FEDoS attack.

To perform a FEDoS attack in interval I_λ, an adversary uses at least one *CN* to send an invalid endorsement according to step 1g to *CH*:

$$I_\lambda,\ CN \rightarrow CH : End_{CN} \neq h(c_\lambda^{CN}||R||T_M).$$

However, this attack is only successful when the endorsement is used to generate *SEnd*. To elude detection, the adversary must disclose a valid hash value at time T_λ which passes the verifications in steps 3c and 3d.

Any **type I adversary (Outsider)** cannot successfully perform a FEDoS attack, since he is not in possession of pairwise keys to authenticate himself. False endorsements or replayed old endorsements are immediately dropped. To be able to send endorsements to *CH*, an adversary requires valid pairwise keys. Since an outside adversary has no access to keys stored on sensor nodes, he can only analyze overheared messages or misbehave in the execution of the security mechanisms to expose keys. The former violates the assumptions that the applied cryptosystem cannot be broken, the latter the assumption that misbehaving in the execution of the security mechanism does not result in the exposure of keys (cf. Section 4.3).

Now we consider a **type II adversary (Insider)** who has compromised one or more *CN*s. Since the adversary can access the pairwise keys, he can send authentic messages which are accepted by *CH*.

We assume that a compromised *CN* sends a false endorsement $\overline{End_{CN}}$ in step 1g. Since *CH* randomly uses t endorsements to generate *SEnd* (cf. step 1j), the attack is successful with a certain probability. Now we discuss how the adversary can try to elude detection to continue his attacks. To get not excluded, *CN* must disclose a hash value $\overline{c_\lambda^{CN}}$ at time T_λ (cf. step 2b). This hash value must pass the verification in step 3c. *CH* verifies that $h(\overline{c_\lambda^{CN}}) = c_{\lambda-1}^{CN}$. Since the hash function provides preimage and second preimage resistance, the adversary must disclose $\overline{c_\lambda^{CN}} = c_\lambda^{CN}$ to pass the verification. If the adversary discloses a different value, *CH* immediately excludes *CN*. We assume that the adversary has disclosed the correct hash value. In step 3d, *CH* uses this hash value to calculate $End\,'_{CN}$ and compares it with the received $\overline{End_{CN}}$. Since the endorsements are calculated using a preimage and second preimage resistant hash function, the false endorsement $\overline{End_{CN}}$ must be equal to the valid endorsement End_{CN} to pass the verification $End\,'_{CN} = End_{CN}$. This is in contradiction to our assumption that *CN* has sent a false endorsement. Thus, the adversary can either send only a valid endorsement or gets detected and excluded.

When a false endorsement has been used to generate *SEnd*, a new report generation is initiated but without the compromised node. Likewise, if a node does not disclose the hash value in order to prevent detection of a FEDoS attack, this node is also excluded from *CH* and a new report is generated.

A compromised *CN* performing a FEDoS attack is detected at the time of disclosure of the hash value used to generate the first false endorsement. The compromised *CN* can send false endorsements until this point in time. The number of false endorsements that can be sent depends on the number of regular reports generated for the sink and on the setting of the hash value disclosure. If the parameters are well adjusted, then an adversary can invalidate only one report and is then excluded.

If our protocol is used in a dynamic system where the role of a *CH* changes, then each node must gain its own experiences which other nodes behave maliciously. Thus, if a *CN* is excluded by the current *CH* it might be possible that the malicious *CN* can again send a false endorsement to a successor *CH* which then excludes the malicious node.

6.6.3 Additional Attacks

In this section, we discuss how an adversary could try to disrupt the functionality of our proposed protocol by performing one or more of the attacks described in Section 2.2.4.

An adversary can try to disrupt the collaborative report generation to prevent the sink from receiving a response message. Since the report generation involves only local one hop communications, attacks such as selective forwarding or data alteration are not suitable. However, an adversary can perform a jamming attack to prevent sensor nodes from receiving certain messages. Jamming the broadcast message sent by *CH* to its *CN*s, can prevent some of the *CN*s from receiving R, T_M. As a result, they do not generate endorsements and *CH* might not receive sufficient endorsements to generate a valid sink message. Likewise, the adversary can jam the wireless channel at the time where *CN*s send endorsements or when *CH* sends the sink message. As already discussed in Section 2.2.4, jamming attacks are a general problem of all wireless communication systems and can only be mitigated. Similar to the STEF protocol, non-receiving of these message has no negative impact on the protocol itself (cf. Section 5.5.3) but may influence the WSN application when messages do not arrive (in time).

Alternatively, an adversary can try to disrupt the detection process of false endorsing nodes. As discussed in the previous section, an adversary cannot prevent that a false endorsing *CN* is excluded. However, an adversary can try to mislead *CH* to falsely exclude an uncompromised *CN* by jamming the wireless channel at the time of hash value disclosure. This is possible since *CN*s that do not disclose the verification hash values are excluded by *CH*. To address this issue, we propose an enhanced version of our protocol in Section 6.8 which is not susceptible to this attack. This version of the protocol can be used in scenarios where jamming attacks are likely and no jamming countermeasures such as proposed in [255, 256] are taken.

6.7 Theoretical Performance Analysis

In this section, we theoretically evaluate the performance of our protocol. First, we evaluate the storage requirements on the sensor nodes. Second, we analyze the energy consumption of our protocol by evaluating the impact of computational and communica-

tion overhead. In all our analyses the resources of the sink are assumed to be unlimited and are therefore not considered.

6.7.1 Storage Requirements

We consider the storage requirements of a *CH* in the analysis since *CH* has the highest requirements. *CH* needs to store a hash chain, verification values for all nodes in the cluster, pairwise keys for all nodes in the cluster, and temporarily R, T_M, received endorsements and respective node identifiers of the endorsing nodes. If the role of *CH* does not change, a *CN* needs only to store a hash chain and one pairwise key shared with *CH*. In a dynamic system the storage requirements for *CH* can be transferred to all nodes in the cluster.

Most of the storage space is required by the hash chain. As already mentioned above, efficient constructions for one-way hash chains should be used. In [110] two efficient constructions for one-way hash chains have been proposed called Sandwich-chain and Comb Skipchain. An example for a WSN with resource constrained sensor nodes is presented. The WSN is assumed to have a lifetime of 10 years, where hash values are disclosed every second, thus, requiring a total of 315 million values. Storing each single value in one hash chain is impossible. Using the Sandwich-chain, the storage requirements are still too large for current sensor hardware (more than 1.5 Mbyte) whereas the storage requirements using Comb Skipchain (1188 byte) are feasible for this scenario. However, the Sandwich-chain construction has the advantage that a verifier can "catch up", i.e., verify a hash chain value based on a distant trusted chain value, faster. Summarizing, the storage requirements for the hash chain depend on the scenario (lifetime of the network, disclosure schedule of the hash values) and on the specific construction of the hash chain. In addition, the storage requirements for the hash chain decrease since old already disclosed hash values can be deleted to free storage space.

To verify the disclosed hash values, *CH* needs to store one (or more) verification hash values. Using simple hash chain constructions [136] requires *CH* to store one hash value for each *CN*. The Comb Skipchain is a hierarchical one way chain consisting of two levels of chains where a primary chain act as root of a set of secondary chains. The verification can be either realized by storing either only one value of the primary chain or by storing one value of the primary and additionally one value of the secondary chain. Storing two values has the advantage that the verification is faster and cheaper, since the verifier has to perform only one hash computation. In the initialization phase *CH* is configured with the initial verification values $c_0^{CN_j}$, $j = 1, \ldots, u$ where the respective value represent either one or two effective values depending on the implementation of the hash chain. These values are successively replaced with afterwards disclosed values. Hence, the storage requirements for the verification values remain constant. If sensor nodes are excluded from the communication (or fail because of energy failure), then *CH* deletes the hash values of the respective nodes to free storage space.

CH establishes a pairwise key with each *CN* in the cluster to provide some basic authentication and integrity measures. As stated in our general setting, for example ZigBee link layer keys can be established in the initialization phase.

If a report for the sink is generated, *CH* has to store R, T_M, and the endorsements and the related *ID*s of the endorsing *CN*s temporarily until the disclosure of the necessary verification hash values. After the endorsements have been verified, this data can be deleted.

To quantify the storage requirements SR let SR_H, SR_V, L_n, L_k L_r, L_t, and L_h denote the *storage requirements for the hash chain*, the *storage requirements for the verification value(s) for one node*, the *length of an ID*, the *length of a symmetric key*, the *length of a report*, the *length needed for the time of measurement*, and the *length of an endorsement* (i.e., the length of a hash value), respectively. Let $u+1$ be the number of nodes in the cluster, v the average number of endorsement sets a node stores, i.e., the number of reports for which endorsements are temporarily stored, and w the average number of endorsements for one specific report. Thus, the storage requirements for a sensor node are:

$$SR = SR_H + u \cdot (SR_V + L_n + L_k) + v \cdot (L_r + L_t + w \cdot (L_h + L_n)) \tag{6.11}$$

Example 6.5 *Suppose a lifetime of 10 years where hash values are disclosed every second. Using the Comb Skipchain construction for the hash chain requires 1188 byte (9504 bit) as mentioned above. Realizing the verification by storing two hash values requires a verifying node to store one hash value of length 80 bit of the primary chain and one hash value of length 64 bit of the secondary chain [110]. Let the length of an ID be 10 bit, the length of a symmetric key be 64 bit, the length of a report be 24 byte, the length of* T_M *be 29 bit, the length of an endorsement be 64 bit,* $u = 6$*,* $v = 2$*, and* $w = 5$*. Thus, the storage requirements are:*

$$\begin{aligned} SR &= 9504 + 6 \cdot ((80 + 64) + 10 + 64) + 2 \cdot (192 + 29 + 5 \cdot (64 + 10)) \\ &= 11994bit = 1499.25byte \end{aligned} \tag{6.12}$$

The example clearly shows that the major storage space is required by the hash chain. Since only few values are currently needed and most of the values are only needed in the far future, the main part can be stored in the 512 kbyte flash memory of a Berkeley Mica2 mote. The currently needed space for hash values, temporarily stored reports etc. requires only little space in the 4 kbyte RAM of the node. If new values of the hash chain are required, they can be moved from the flash memory to the RAM.

To apply the STEF protocol together with our proposed protocol, additionally requires that *CH* stores a shared key with the sink. This enables the sink to send encrypted ticket values to *CH*.

6.7.2 Energy Consumption

In this section, we evaluate the energy consumption of our protocol. The energy consumption of the sensor nodes can be divided into two parts, (1) the energy required for the cluster $\mathfrak{C}$ for report generation and the endorsement verification, and (2) the energy to forward the message along multiple hops to the sink. We evaluate the energy requirements of our protocol solely and in combination with STEF. The combination of both

protocols is compared with the PVFS scheme proposed by Li et al. in [150]. We consider the worst case with the maximum energy consumption where all nodes in the cluster $\mathfrak{C}$ send endorsements and no PDoS attack is performed, i.e., no messages are filtered out. However, we neglect the energy costs for comparison operations and the XOR operation since these costs are negligibly small.

Energy Consumption of Cluster $\mathfrak{C}$

The report generation requires CH to broadcast R and T_M to all u nodes CN_j which all u CN_j receive. CH and all u CN_j generate an endorsement by performing one hash computation. Each CN_j sends the endorsement back to CH. Thus, CH receives u endorsements. CH generates the message for the sink, and sends it to the sink. Each CN_j discloses the used hash value and sends it to CH. The hash values[2] and endorsements are verified by CH. Equation 6.13 shows the required energy E_{l1} for the local communication and computational overhead of the cluster. We use $e_1 = e_{1s} + e_{1r}$ to denote the energy consumed in sending and receiving one byte, and e_2 to denote the energy for one hash computation.

$$\begin{aligned} E_{l1} = {} & e_{1s} \cdot L_r \cdot L_t \\ & + u \cdot e_{1r} \cdot L_r \cdot L_t \\ & + (u+1) \cdot e_2 \\ & + u \cdot e_{1s} \cdot L_h \\ & + u \cdot e_{1r} \cdot L_h \\ & + (L_r + L_t + L_h + (t+1) \cdot L_n) \cdot e_{1s} \\ & + u \cdot e_{1s} \cdot L_h \\ & + u \cdot e_{1r} \cdot L_h \\ & + 2u \cdot e_2 \end{aligned} \tag{6.13}$$

Example 6.6 *We use the results presented in [261] to quantify $e_{1s} = 16.25\ \mu J$ for sending, $e_{1r} = 12.5\ \mu J$ for receiving, and $e_1 = 28.75\ \mu J$ for sending and receiving one byte using Berkeley Mica2 Motes. The energy consumed for performing one hash or MAC computation using RC5 [195] block cipher is $e_2 = 15\ \mu J$ and takes about $0.5ms$. In addition to the values used in Example 6.5 let $t = 4$. This results in a total energy consumption for the cluster $\mathfrak{C}$ of $E_{l1} = 11.66mJ$. Thus, the average energy consumption for one node is $1.67mJ$.*

The Mica2 Motes are powered with two 1.5 V AA batteries in series connection. We assume a total capacity of 2750 mAh using standard AA batteries which results in 29700 J. Thus, the average ratio of energy consumed for one node and one report generation including the CH verification if a false endorsement has been sent is 0.0056% of the total available energy.

[2]Depending on the the interval of the last previously verified hash value, and the used hash chain construction, it might be necessary to compute more than one hash computation. We neglect this in our theoretical analysis. In the implementation however, this issue is considered.

Energy Consumption for Message Forwarding

Since transmitting messages is the most cost intensive factor in WSNs, it is crucial that the transmitted messages are short. In particular, the message sent to the sink should be short since the transmission of this message involves multiple forwarding en-route sensor nodes. Thus, one goal in the design of our protocol was to let this message be as short as possible.

The message sent to the sink consists of the report R, the time of measurement T_M, the compressed endorsements *SEnd* and the *t+1* node identifiers. Equation 6.14 shows the energy consumption to forward the message along H hops to the sink.

$$E_{f1} = H \cdot (L_r + L_t + L_h + (t+1) \cdot L_n) \cdot e_1 \tag{6.14}$$

Example 6.7 *Assuming that the message generated in Example 6.6 is forwarded along 100 hops, we have an energy consumption of* $E_{f1} = 120.4mJ$. *Compared to the energy consumed in the cluster* $\mathfrak{C}$ *this is more than 6.68 times as much. Thus, the total energy requirements are* $E_1 = E_{l1} + E_{f1} = 132.06mJ$.

Energy Consumption in Combination with STEF

The report generation and verification is performed as described in Section 6.5, except that the message sent to the sink additionally contains a query identifier of length L_n and a ticket value of length L_h which enables the resistance against PDoS attacks. The ticket value is basically a hash value of length L_h.

The local communication and computational overhead is shown in Equation 6.15.

$$\begin{aligned} E_{l2} = {} & e_{1s} \cdot L_r \cdot L_t \\ & + u \cdot e_{1r} \cdot L_r \cdot L_t \\ & + (u+1) \cdot e_2 \\ & + u \cdot e_{1s} \cdot L_h \\ & + u \cdot e_{1r} \cdot L_h \\ & + (L_r + L_t + 2L_h + (t+2) \cdot L_n) \cdot e_{1s} \\ & + u \cdot e_{1s} \cdot L_h \\ & + u \cdot e_{1r} \cdot L_h \\ & + 2u \cdot e_2 \end{aligned} \tag{6.15}$$

Equation 6.16 shows the energy consumption to forward the message along H hops, where at each hop a hash computation is performed to verify if the ticket within the message is valid to prevent PDoS attacks.

$$E_{f2} = H \cdot ((L_r + L_t + 2L_h + (t+2) \cdot L_n) \cdot e_1 + e_2) \tag{6.16}$$

Example 6.8 *We use the values of the previous examples. The energy consumption for a combination of both protocols is* $E_2 = E_{l2} + E_{f2} = 11.81 + 148.48 = 160.29mJ$.

Comparison to PVFS

The scheme proposed by Li et al. [150] addresses all three attacks. However, it is a probabilistic scheme which tolerates a certain number of false endorsements but it does not enable *CH* to detect and exclude a false endorsing *CN*.

The scheme requires *CH* to generate a report which is broadcasted to the *CN*s. Each of the *CN*s and *CH* generate a MAC on the report using a symmetric key. Each *CN* sends the MAC together with the key identifier of the used key back to *CH*. *CH* generates and sends the message for the sink. The message contains an identifier of the cluster, the report, and *t+1* MACs with the respective key identifier. Let L_{kid}, and L_{cid} denote the *length of a key identifier*, and the *length of a cluster identifier* respectively. The energy for the cluster to generate and send the message is shown in Equation 6.17.

$$\begin{aligned} E_{l3} = & \, e_{1s} \cdot L_r \\ & + u \cdot e_{1r} \cdot L_r \\ & + (u+1) \cdot e_2 \\ & + u \cdot e_{1s} \cdot (L_h + L_{kid}) \\ & + u \cdot e_{1r} \cdot (L_h + L_{kid}) \\ & + (L_{cid} + L_r + (t+1) \cdot (L_h + L_{kid})) \cdot e_{1s} \end{aligned} \tag{6.17}$$

Other CHs, which forward the message to the sink, share a symmetric key used to generate a MAC with a certain probability. If a *CH* holds a corresponding verification key, it verifies the MAC and records the verification result in two binary sequences. One binary sequence records the verified MACs, and one binary sequence records the verified valid MACs. The length of these two binary sequences depends on the length of *t+1*. Each node which has performed a verification, appends a signature (in the form of a MAC), generated with a symmetric key shared with the sink, to the message. In addition, an accepted flag is set if a certain number of valid verifications has been performed. If this bit is set, the message is forwarded to the sink without further verifications. This also limits the maximum message size, since no further signatures are appended. For a comparison with similar properties, we set the number of verifications (and appended signatures) to *t+1*.

Equation 6.18 shows the energy consumption for forwarding the message including the energy required to verify the MACs and to generate the signatures. The first part (lines 1-2) of the equation represents the constant part (containing cluster identifier, report, *t+1* MACs and the respective key identifier, the two binary sequences and the accepted flag). The second part of the equation (lines 3-5) represents the energy required for the dynamic part of the message, i.e., the transmission energy for the appended signatures. We use the σ function[3] to distinguish between two cases. Altogether, *t+1* signatures are attached at maximum. Therefore, it must be distinguished between the case $H \leq (t+1)$

[3] $\sigma(x) = \begin{cases} 0 & x \leq 0 \\ 1 & x > 0 \end{cases}$

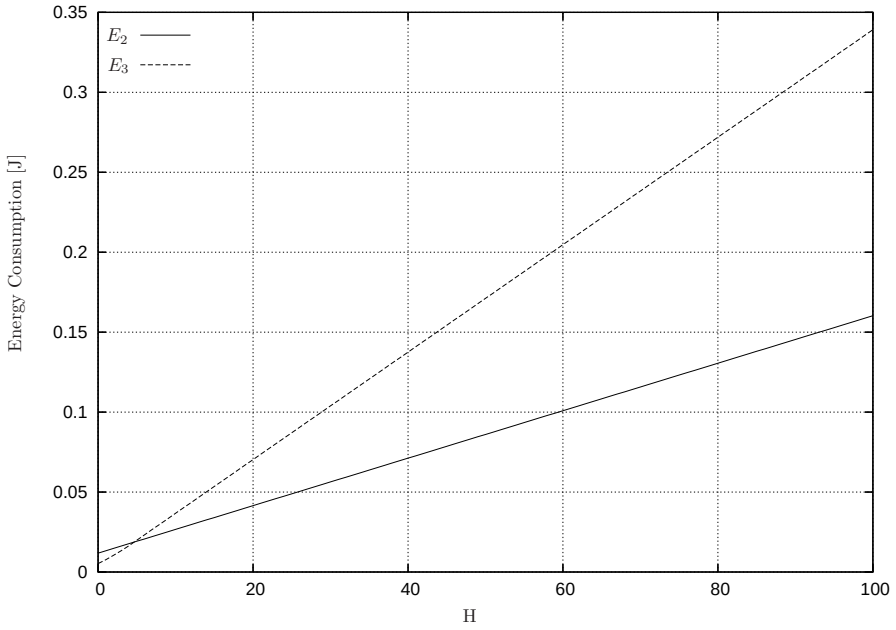

Figure 6.5: Energy Consumption depending on the average number of hops for $u = 6$ and $t = 4$

and $H > (t + 1)$. The last part (lines 6-7) represents the energy required to verify and generate the signatures where we also must distinguish between the two cases.

$$\begin{aligned} E_{f3} = H \cdot (&L_{cid} + L_r + (t+1) \cdot (L_h + L_{kid}) + \\ &2\lceil log_2(t+1) \rceil / 8 + 1/8) \cdot e_1 \\ &+ \sigma(t + 2 - H) \cdot \Sigma_{k=0}^{H} k \cdot L_h \cdot e_1 + \\ &\sigma(H - t - 1) \cdot (\Sigma_{k=0}^{(t+1)} k \cdot L_h \cdot e_1 + \\ &(t+1)(H - (t+1)) \cdot L_h \cdot e_1) \\ &+ \sigma(t + 2 - H) \cdot 2 \cdot H \cdot e_2 + \\ &\sigma(H - t - 1) \cdot 2 \cdot (t+1) \cdot e_2 \end{aligned} \tag{6.18}$$

Example 6.9 *We use the same values as in the previous examples and let the length of a key identifier L_{kid} and the length of a cluster identifier L_{cid} be 16 bit, respectively. The energy requirement is $E_3 = E_{l3} + E_{f3} = 5.3 + 333.87 = 339.17mJ$.*

Figure 6.5 plots the energy consumption depending on the average number of hops for $u = 6$ and $t = 4$. E_2 represents the energy consumption of our protocol in combination with the STEF protocol and E_3 the energy consumption of PVFS. It can be seen that the local overhead of the combination of our protocol and STEF is slightly higher than PVFS, but compensates this by reducing the energy required to forward messages.

6.8 Enhanced Endorsement Protocol

The above described basic protocol can be used in scenarios where jamming attacks are unlikely or measures, e.g., [255, 256], are taken to cope with jamming attacks. However, as described in Section 6.6.3, if an adversary is able to successfully perform a jamming attack, uncompromised sensor nodes may be falsely excluded by *CH*.

In this section, we discuss several possible solutions and extend the above proposed protocol to additionally cope with this jamming attack. For this, we introduce a greylisting approach. If *CH* does not receive the proof of a *CN*, either an active adversary might have performed a jamming attack or the *CN* indeed did not perform the proof. Thus, *CH* does not immediately exclude this *CN*, but rather greylists it. *CN* is able to perform the proof within the next endorsement it sends when no jamming attack occurs. The enhancement introduces only a marginal increase of energy consumption compared to the basic protocol.

6.8.1 Addressing the Jamming Attack

Jamming attacks are a general problem in wireless networks. The shared wireless channel can be easily blocked by an adversary, resulting in a DoS of transmission and reception functionalities.

Security protocols should not enable an adversary to cause additional damage (other than successful transmission or reception of messages) by performing a jamming attack. In the basic protocol, however, this is possible. It is assumed that a *CN* is compromised if it does not disclose the hash chain value used to generate an endorsement. As a result, this *CN* is excluded by *CH*. However, if *CH* does not receive the hash chain values because of a jamming attack, the *CN* is falsely classified as compromised and excluded from the report generation process.

One way to cope with the jamming attack might be using jamming detection mechanisms. If there are indications of a jamming attempt, *CH* could not immediately exclude the *CN* from which it does not receive the hash chain value. However, detection of jamming alone is not sufficient, since even if we were able to detect that a jamming attack has been performed, we are not able to verify the previously received endorsement. Furthermore, current jamming detection mechanisms are very unreliable.

Another approach would be a random variation in the disclosure schedule of the hash chain values. Thus, the adversary does not know the exact point in time he should perform the jamming attack. The problem with this approach is that we cannot extend the variation to an arbitrary long time span, since the verification of received endorsements should be as fast as possible to enable a quick reaction on false endorsements, i.e., exclusion of false endorsing nodes and if necessary new report generation. Thus, the adversary just has to jam a short period of time to accomplish the goal that innocent *CN*s are falsely excluded.

Alternatively, *CH* can perform a challenge-response like verification with the *CN* from which it does not receive the hash chain value. *CH* might request the hash chain value

if it does not receive the value at the specified point in time. However, we still cannot distinguish whether *CN* does not respond or a jamming attack occurs.

One approach that does not require any additional mechanisms, such as jamming detection, introduces greylists and requires only slight modifications to the original protocol. This approach is based on the following considerations. As long as an adversary continues the jamming attack, *CH* does not receive endorsements anyway. Thus, *CH* requires to verify previously received endorsements not until the receiving of a new endorsement. This can be realized by integrating a second verification process into the endorsement messages. After verifying the old endorsement, *CH* can immediately use the new one.

We propose to realize this approach as follows. Each *CH* maintains a greylist. *CH* adds a *CN* to the greylist if it does not receive the hash chain value from *CN* to verify the previously received endorsement at the specified point in time. *CH* does not use subsequent received endorsements from a *CN* that is listed in the greylist, until it receives a valid hash chain value to successfully verify the unverified endorsement. *CH* completely excludes a *CN* if a verification fails or if a specified threshold is reached, e.g., maximum time span without a successful verification or maximum entries in the greylist is reached. To enable a *CN* to be trusted again, it appends the last hash chain value which is allowed to be disclosed to the next endorsement sent to *CH*. This enables *CH* to verify the old unverified endorsement. If the verification of the old endorsement passes, *CN* is trusted again and the currently received endorsement can be used to generate the current report. If an adversary still performs a jamming attack, *CH* would not receive the message anyway and *CN* remains in the greylist. Generally, we cannot protect against jamming attacks that prevent reception of messages. However, applying this modification to the protocol prevents an adversary from performing a jamming attack that affects the protocol in such a way that an innocent *CN* is falsely excluded. We describe the enhanced protocol in the next section.

6.8.2 Description of the Enhanced Protocol

In the enhanced protocol, the initialization phase remains the same as in the original protocol. However, the report generation and verification phase are modified to cope with the jamming attack. Each *CH* maintains a greylist that stores the node identifiers of *CN*s from which *CH* does not receive verification hash chain values. Furthermore, a *CN* sends the last verification hash chain value that is allowed to be disclosed together with each endorsement it sends to *CH*.

Before we describe the modified report generation and verification phase in detail, we illustrate the jamming attack in the following example.

Example 6.10 *We assume that a report generation has been performed according to Examples 6.1, 6.2, 6.3, and 6.4. However, we assume that an adversary has performed a jamming attack at time T_2 and successfully prevented CH from receiving the verification hash values from CN_1 and CN_2. Figure 6.6 illustrates this. In the basic protocol, CH*

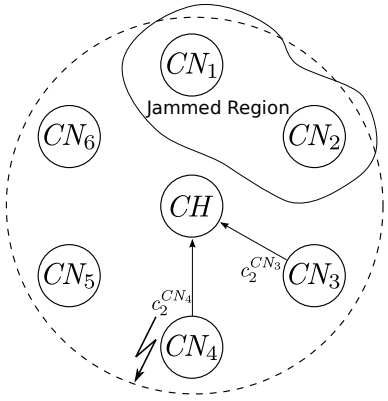

Figure 6.6: Jamming Attack at time T_2: CH does not receive $c_2^{CN_1}$ and $c_2^{CN_2}$

falsely excludes both CNs. However, in the enhanced protocol, CH adds CN_1 *and* CN_2 *to its greylist and does not immediately exclude them.*

In the following, we describe the modified report generation and verification phase in detail. Each phase is first illustrated by means of an example. Afterwards, we describe the respective modified algorithms.

Report Generation

We first continue Example 6.10 to describe how the uncompromised CN_1 and CN_2 can prove with the next endorsement that they, indeed, have sent a correct endorsement in interval I_2.

Example 6.11 *We assume that a new report is generated in interval* I_4*. Since CH did not receive hash values from* CN_1 *and* CN_2*, they are listed in its greylist. Furthermore, we assume that the adversary that has performed the jamming attack has moved to a different location, i.e., no jamming attack is performed anymore.*

As in the basic protocol, CH generates R *and* T_M *and sends these values to the neighboring CNs. After a CN has received this message, it checks the interval of validity of* T_M *by verifying if the measurement has been performed in the current interval* I_4*, and if the report* R *received from CH matches its own measurement* R' *within a certain range* ε*. We assume that these verifications fail on nodes* CN_3 *and* CN_5*. However, on nodes* CN_1*,* CN_2*, and* CN_6 *the verifications pass, i.e., these nodes generate an endorsement.* CN_1*,* CN_2*, and* CN_6 *calculate*

$$End_{CN_k} = h(c_4^{CN_k}||R||T_M) \tag{6.19}$$

for $k = 1, 2, 6$*. In contrast to the basic protocol, each* CN_k *sends the endorsement* End_{CN_k} *together with the last hash value that is allowed to be disclosed to CH. In this case,* $c_2^{CN_k}$*. Figure 6.7 illustrates this. For each received endorsement, CH checks if the endorsing CN is listed in its greylist. In this case,* CN_1 *and* CN_2 *are listed, and CH is now able*

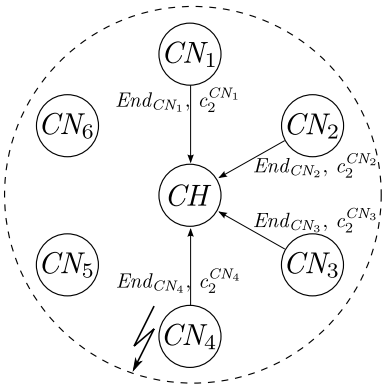

Figure 6.7: CN_k, $k = 1, \ldots, 4$ send endorsements End_{CN_k} and the last hash value that is allowed to be disclosed $c_2^{CN_k}$ to CH. The endorsement of CN_4 is invalid.

to verify the old endorsements generated in interval I_2. The detailed description of the verification is described in Example 6.12 and Algorithm 6.7. Since, indeed, an adversary has performed a jamming attack, and the verifications of the old endorsements of CN_1 and CN_2 pass, CN_1 and CN_2 are trusted again, i.e., the currently received endorsements can be used. CH temporarily stores all received endorsements, R, and T_M for future verification purposes.

CH calculates its own "endorsement" End_{CH}, chooses two of the four received endorsements, e.g., End_{CN_1} and End_{CN_6}, and calculates

$$SEnd = End_{CH} \oplus End_{CN_1} \oplus End_{CN_6}. \tag{6.20}$$

Then, CH sends the following message to the sink

$$R, T_M, SEnd, \{ID_{CH}, ID_{CN_1}, ID_{CN_6}\} \tag{6.21}$$

containing the report R, the time of measurement T_M, the compressed endorsements $SEnd$, and the set of node identifiers of the endorsing nodes.

Now, we describe the modified report generation phase in general by means of two algorithms. Algorithm 6.5 specifies the actions of CH. As in the original protocol, CH generates R and T_M and sends these values to all CN_j in the cluster. Each CN_k that agrees on R and T_M generates an endorsement End_{CN_k} and sends it together with the last hash chain value $c_{\lambda-\Delta}^{CN_k}$ that is allowed to be disclosed to CH (see Algorithm 6.6). CH maintains a set F, containing all node identifiers of trusted CNs whose endorsements are accepted. Initially, after the initialization phase, F contains the node identifiers of all CN_j in the cluster. Furthermore, CH maintains a set G, containing all the node identifiers of greylisted CNs, i.e., those CNs from which CH did not receive a hash value to verify a previously sent endorsement. For each CN_k from which CH receives the tuple $(End_{CN_k}, c_{\lambda-\Delta}^{CN_k})$, it first checks if this CN_k is listed in its greylist G. If so, CH verifies the old endorsement for which it has not received a hash chain value at the specified point in time, using $c_{\lambda-\Delta}^{CN_k}$ according to Algorithm 6.7. The node identifier of

CN_k is removed from G and added back to the set of trusted nodes F if the verification passes. Otherwise, CN_k is excluded from any further report generation. The detailed description of the verification is described below in Example 6.12 and Algorithm 6.7. Next, CH temporarily stores each endorsement End_{CN_k} it receives from a CN listed in the set F for future verification. After CH has received the endorsements, it calculates $h(c_\lambda^{CH}||R||T_M)$ and selects t endorsements received from trusted CNs, and compresses them to one $SEnd$ using bitwise XOR. The node identifiers of CH and the t CNs whose endorsements have been used to generate $SEnd$, are stored in a data structure V. The final message to the sink consists of R, T_M, End, and V.

Algorithm 6.5 CHRepGen(t, c_λ^{CH})

```
1: Generate R and T_M
2: Send R, T_M to all CNs
3: while Receiving (End_{CN_k}, c_{λ−Δ}^{CN_k}) from CN_k do
4:    if ID_{CN_k} ∈ G then
5:       CHVerifyGreylist(c_{λ−Δ}^{CN_k})
6:    end if
7:    if ID_{CN_k} ∈ F then
8:       store End_{CN_k}
9:    end if
10: end while
11: End_CH = h(c_λ^CH||R||T_M)
12: End = End_CH
13: V = {ID_CH}
14: randomly select t distinct endorsements End_{f_1}, ..., End_{f_t}
15: for i = 1 to t do
16:    End = End ⊕ End_{f_i}
17:    add node identifier f_i to V
18: end for
19: Sendto(Sink) : R, T_M, End, V
```

Algorithm 6.6 describes the actions of a CN when it receives R and T_M from CH to endorse (or not to endorse) a report. CN first checks that the measurement has been performed in the current interval. Next, it performs its own measurement R'. If R' matches R within a certain error range ε, CN generates an endorsement and sends it together with the last hash chain value $c_{\lambda-\Delta}^{CN}$ that is allowed to be disclosed to CH.

Verification

The sink verification remains the same as in the original protocol. However, to address the jamming attack, the verification performed by CH is modified. We distinguish between two cases, (1) when CH directly verifies the endorsements of the CNs when they disclose their hash chain values at time T_λ, and (2) the greylisting-based verification when CH did not receive the value at T_λ.

Algorithm 6.6 CNRepEnd(c_λ^{CN}, $c_{\lambda-\Delta}^{CN}$, I_λ, R, T_M, ε)

1: **if** $T_M \in I_\lambda$ **then**
2: Generate R'
3: **if** $R' - \varepsilon \leq R \leq R' + \varepsilon$ **then**
4: $End = h(c_\lambda^{CN}||R||T_M)$
5: `Sendto`$(CH) : End, c_{\lambda-\Delta}^{CN}$
6: **end if**
7: **end if**

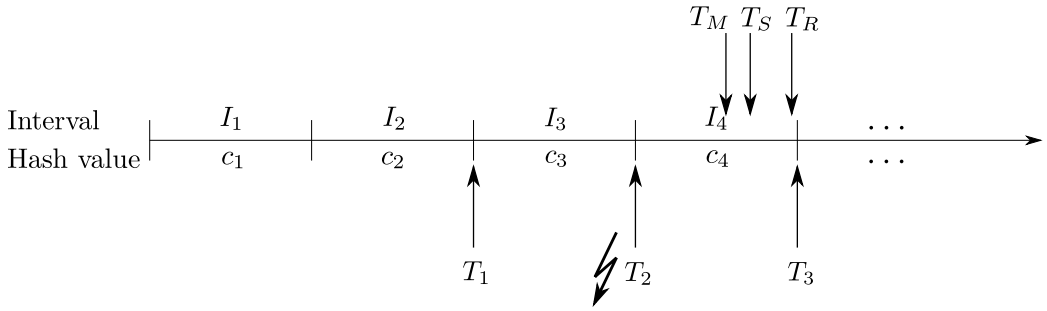

Figure 6.8: An adversary performs a jamming attack at time T_2. During the report generation in interval I_4 no jamming attack is performed anymore.

The first case is identical to the basic protocol except that before Algorithm 6.4 is executed, *CH* removes all *CN*s from which it does not receive a verification hash chain value from the set F and adds them to the greylist G.

In the following, we describe the second case in detail. First, we continue Example 6.11 and afterwards present an algorithm describing the greylist verification.

Example 6.12 *An adversary has successfully performed a jamming attack at time T_2, i.e., CH did not receive hash chain values from CN_1 and CN_2 and thus, cannot verify the endorsements it has received in interval I_2 (see Figure 6.8). Thus, CH has added ID_{CN_1} and ID_{CN_2} to its greylist G and has generated a new message for the sink without the nodes in the greylist. In the current report generation in interval I_4, CH receives $c_2^{CN_1}$ and $c_2^{CN_2}$ together with the endorsements End_{CN_1} and End_{CN_2}. Using these hash values, CH verifies the old endorsements. In this example, the verifications pass since a jamming attack has, indeed, prevented CH from receiving the verification hash chain values. Thus, the jamming attack has no affect in the modified protocol, i.e., an adversary cannot blame innocent CNs so that they are excluded by CH.*

Now we describe the case where *CH* does not receive the hash chain value from a *CN* to verify a previously sent endorsement in general. *CH* executes Algorithm 6.7 in the next report generation phase when *CN* sends a new endorsement together with the last hash chain value that is allowed to be disclosed. First, *CH* removes the identifies of *CN* from G. Next, *CH* verifies that the received hash chain value is correct. If it is correct, *CH* recomputes the endorsement and compares it with the stored old endorsement. If

both verifications pass, *CH* re-inserts the node identifier of *CN* into its set of trusted nodes F and accepts endorsements from this node again. Otherwise, this *CN* is excluded, i.e., endorsements from this node are not accepted and *CH* can delete the pairwise key and the verification value of this *CN*.

Algorithm 6.7 CHVerifyGreylist($R^{old}, T_M^{old}, V, ID_{CN}, End_{CN}^{old}, c_{\lambda-\Delta}^{CN}, c_{\lambda-\Delta-1}^{CN}, F, G$)

1: $G = G \setminus \{ID_{CN}\}$
2: **if** $h(c_{\lambda-\Delta}^{CN}) = c_{\lambda-\Delta-1}^{CN}$ **then**
3: calculate c_{old}^{CN} used to generate End_{CN}^{old}
4: $End_{CN}^{'old} = h(c_{old}^{CN}||R^{old}||T_M^{old})$
5: **if** $End_{CN}^{'old} = End_{CN}^{old}$ **then**
6: $F = F \cup \{ID_{CN}\}$
7: **end if**
8: **end if**

The listing of Protocol 6.2 summarizes the enhanced protocol for the cluster $\mathfrak{C}$. For the sake of clarity, we show only the modified protocol steps compared to Protocol 6.1. Protocol steps 1a - 1f are identical to the original protocol. In step 1g, an endorsing CN_k sends not only the endorsement End_{CN_k} but also the last verification value $c_{\lambda-\Delta}^{CN_k}$ that is allowed to be disclosed. A new step 1h is introduced after step 1g to verify previously greylisted *CN*s. These *CN*s have sent an endorsement in a previous report generation but did not disclose a verification value. Starting from step 1i the subsequent steps of *CH* remain the same as in the original protocol. Since the sink verification is unchanged, protocol steps 2a - 2e also remain unchanged. Protocol step 3a where endorsing *CN*s disclose their used hash values remains unchanged as well. The last modified protocol step is step 3b. Instead of immediately excluding a CN_k which does not disclose the verification hash value, such a node is greylisted. Steps 3c - 3e again remain unchanged.

In the description, we assume that the role of *CH* does not change. In case that the role of *CH* can dynamically change between the nodes of the cluster, it is required that all nodes establish pairwise keys and exchange the verification values.

As in the listing of Protocol 6.1, we do show the combination with the STEF protocol. To use the protocol in combination with the STEF protocol, requires each node to store a symmetric key shared with the sink.

The possibly required reaction depends on the setting. In a reactive setting (as in the STEF protocol), reactions are initiated by the sink by sending new queries. In an active setting, *CH* initiates a reaction by initiating a new report generation after detecting that a FEDoS attack has been performed and the endorsement has been used to generate the sink message.

Protocol 6.2 Enhanced protocol to detect FEDoS attacks

1. Initialization:
   - Each sensor node has a unique identifier.
   - Each sensor node stores a hash chain.
   - CH has established pairwise keys with its neighboring CN_j.
   - CH stores verification value(s) $c_0^{CN_j}$ for each CN_j.
   - CH maintains a set F of trusted nodes.
   - CH maintains a set G of greylisted nodes.

2. Protocol steps:

...	...	...			...
1g	I_λ	CN_k	$\rightarrow$	CH:	$End_{CN_k}, c_{\lambda-\Delta}^{CN_k}$
1h	I_λ	CH:			if $ID_{CN_k} \in G$, remove ID_{CN_k} from G verify $h(c_{\lambda-\Delta}^{CN_k}) = c_{\lambda-\Delta-1}^{CN_k}$ calculate c_{old}^{CN} used to generate End_{CN}^{old} if $End_{CN}^{'old} = End_{CN}^{old}$ add ID_{CN_k} to F
1i	I_λ	CH:			store all End_{CN_k} to R, T_M
...	...	...			...
3b	T_λ	CH:			move ID_{CN_k} of non-disclosing CN_k from F to G
...	...	...			...

6.8.3 Security Analysis

In Section 6.6 we have analyzed the security of the basic protocol and showed that innocent CNs can be falsely excluded if an adversary performs a jamming attack at the time of disclosure of the hash chain values. The enhanced protocol is, except for the greylist concept, identical to the original protocol. Thus, we briefly discuss only how the enhancement addresses the jamming attack. We refer to the protocol steps listed in Protocol 6.2.

The original protocol is modified by introducing a greylist. CH adds a CN to this list if it does not receive the verification hash value to verify the previously received endorsement (cf. step 3b). The next endorsement message, CH receives from CN, includes a hash chain value that enables CH to verify the old endorsement (cf. step 1g). If the verification of the old endorsement passes (cf. step 1h), the currently received endorsement can be used to generate the current report. It is sufficient to send the verification hash chain values together with the endorsements, since the information that an old endorsement was correct or not is only required if we receive a new endorsement

to decide whether this endorsement can be used for the current report generation. At an earlier point in time, the adversary could still perform the jamming attack. If the verification of the old endorsement fails, *CN* is excluded from any further report generation, since it has indeed performed a FEDoS attack. Thus, as in the original protocol, an adversary can only send false endorsements until the point in time where verification hash chain values have to be disclosed. In addition, a jamming attack with the goal that *CH* falsely excludes innocent *CN*s is not possible anymore. However, constant jamming attacks with the goal of disrupting the communication can prevent a successful report generation since either *CH* does not receive t endorsements or the message for the sink is blocked by the jammer. But, as soon as the jammer leaves the region, new reports can be generated and old endorsements can be verified.

6.8.4 Theoretical Performance Analysis

In this section, we evaluate the performance of the enhanced protocol and compare it with the original protocol.

Storage Requirements

The only additional storage space required in the enhanced protocol compared to the original protocol is required for the greylist. In addition, the respective endorsements of greylisted *CN*s have to be stored together with R and T_M until they can be verified or are removed since the predefined threshold of stored values is reached. We use the same notation as used in equation 6.11, i.e., we let SR_H, SR_V, L_n, L_k, L_r, L_t, and L_h respectively denote the *storage requirements for the hash chain*, the *storage requirements for the verification hash chain value(s) for one node*, the *length of an ID*, the *length of a symmetric key*, the *length of a report*, the *length needed for the time of measurement*, and the *length of an endorsement* (i.e., the length of a hash value). Likewise, we let $u + 1$ be the number of nodes in the cluster, v the average number of endorsement sets *CH* stores, i.e., the number of reports for which endorsements are temporarily stored, and w the average number of endorsements for one specific report. In addition, we let SR_G denote the *storage requirements for the greylist*, i.e., IDs of greylisted nodes and the respective endorsements and if required respective R and T_M. Thus, the storage requirements SR_{enh} for *CH* are:

$$SR_{enh} = SR_G + SR_H + u \cdot (SR_V + L_n + L_k) + v \cdot (L_r + L_t + w \cdot (L_h + L_n)) \quad (6.22)$$

Example 6.13 *Suppose a lifetime of 10 years where hash chain values are valid for one second. Using an efficient hash chain construction proposed in [110] requires 1188 byte (9504 bit). Let SR_V be 144 bit, the length of an ID be 10 bit, the length of a symmetric key be 64 bit, the length of a report be 24 byte, the length of T_M be 29 bit, the length of an endorsement be 64 bit, $u = 6$, $v = 2$, and $w = 5$. The storage space for the greylist is zero if no node is greylisted. In Example 6.10, CN_1 and CN_2 have been greylisted. Thus, CH stores the IDs and endorsements of these nodes together with R and T_M in the greylist. This results in a storage requirement $SR_G = 20bit + 128bit + 24byte + 29bit = 369bit$.*

The total storage requirements are $SR_{enh} = 12363bit \approx 1545byte$*. Compared to Example 6.5, the enhanced scheme requires approximately 3% more storage space.*

As in the original protocol, the main storage is required by the hash chain where the majority can be stored in the 512 kbyte flash memory of a Mica2 mote. Currently needed values occupy only a small fraction of the 4 kbyte RAM. The storage space for the greylist depends on the number of greylisted nodes. Depending on the number of nodes in a cluster, it might be necessary to limit the maximum size of the greylist. Using an appropriate replacement strategy, such as least recently used (LRU), *CN*s are excluded by removing them from the greylist. However, this is only necessary in scenarios where memory space is extremely scarce. Usually the worst case of storage requirements for the greylist should be reasonable for most scenarios. In the worst case, *CH* needs to store for each *CN* the ID, one endorsement and different values for R and T_M. Since each new endorsement enables the verification of the old endorsement, no more than one endorsement and respective R and T_M for each *CN* needs to be stored. For Example 6.13, the worst case storage requirements are as follows.

Example 6.14 *In the worst case, CH requires to store the ID, one endorsement,* R *and* T_M *for each of the 6 CNs. Thus, the storage requirements for the greylist are* $SR_G = 1770bit \approx 221byte$*. Compared to Example 6.5 of the basic scheme, the storage requirements are approximately 15% higher.*

Energy Consumption

As in the basic scheme, the energy consumption of the sensor nodes can be divided into two parts, (1) the energy required for the cluster for report generation and endorsement verification, and (2) the energy to forward the message along multiple hops to the sink (compare Section 6.7). The energy consumption of the sensor nodes can be divided into two parts, (1) the energy required for the cluster for report generation and endorsement verification, and (2) the energy to forward the message along multiple hops to the sink. The enhanced protocol differs only slightly in the local communication overhead for the cluster. The energy for sending and receiving the endorsement message is slightly higher since a verification hash chain value is included in the message. Equation 6.23 shows the required energy E_{l4} for the local communication and computational overhead of the cluster.

$$\begin{aligned} E_{l4} = {} & e_{1s} \cdot L_r \cdot L_t \\ & + u \cdot e_{1r} \cdot L_r \cdot L_t \\ & + (u+1) \cdot e_2 \\ & + u \cdot e_{1s} \cdot 2L_h \\ & + u \cdot e_{1r} \cdot 2L_h \\ & + (L_r + L_t + L_h + (t+1) \cdot L_n) \cdot e_{1s} \\ & + u \cdot e_{1s} \cdot L_h \\ & + u \cdot e_{1r} \cdot L_h \\ & + 2u \cdot e_2 \end{aligned} \tag{6.23}$$

The energy for message forwarding remains the same as defined in equation 6.14, i.e., $E_{f4} = E_{f1}$.

Example 6.15 *Compared to Example 6.6, the local energy consumption increases about 11.8% from* $E_{l1} = 11.66mJ$ *to* $E_{l4} = 13.04mJ$ *(*$13.19mJ$ *if used in combination with STEF). The energy for message forwarding remains the same, i.e.,* $E_{f4} = E_{f1} = 120.4$ *(*$148.48mJ$ *if used in combination with STEF).*

Now we consider the case that the basic and enhanced protocol are used in combination with the STEF protocol. The total energy consumption presented in Example 6.8 is $E_2 = 11.81 + 148.48 = 160.29mJ$*. For the enhanced protocol the energy consumption is* $E_4 = E_{l4} + E_{f4} = 13.19 + 148.48 = 161.67mJ$*. This is an increase of 0.86%.*

6.9 Implementation and Simulation

The STEF-Sim simulation environment developed to analyze the performance of the STEF protocol presented in Section 5.7 has been extended in [96] to additionally support our proposed protocols to cope with FEDoS attacks. Figure 6.9 shows the new main window of STEF-Sim 2.0. In addition to false data injection and PDoS attacks, STEF-Sim 2.0 supports FEDoS and jamming attacks. STEF-Sim 2.0 enables the analysis of the energy and storage requirements of the STEF protocol, the STEF protocol in combination with the basic or the enhanced protocol, or solely the basic or the enhanced protocol. The results can also be exported in Gnuplot [95] files.

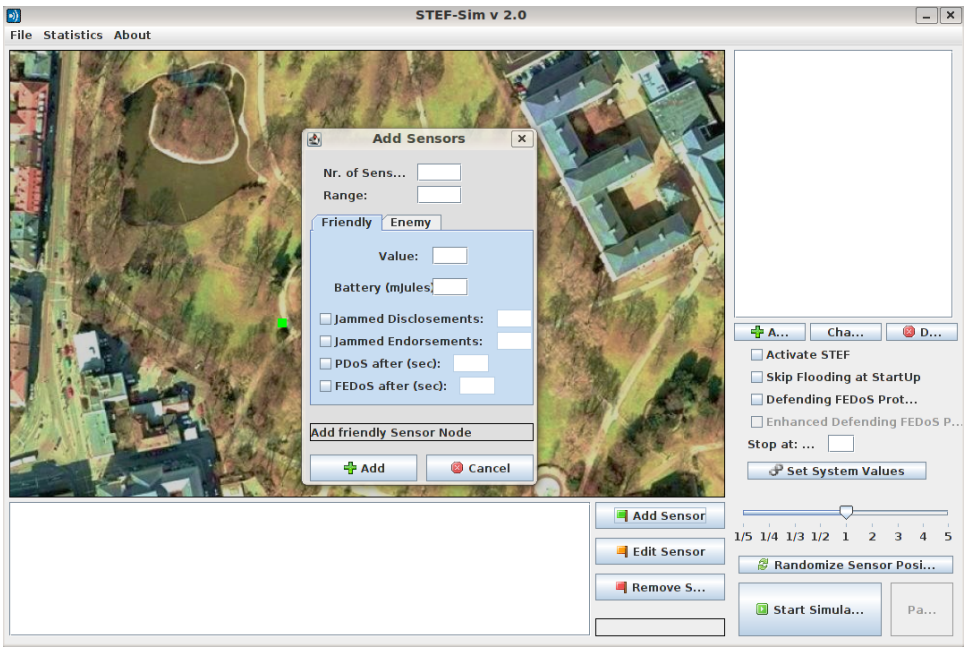

Figure 6.9: Main Window of STEF-Sim 2.0

Several simulations have been performed using STEF-Sim 2.0 to evaluate the impact of different parameters. The simulations use the STEF protocol as a reference to analyze the overhead introduced by the basic and the enhanced protocol. Simulations with different parameters have been performed for the STEF protocol, the basic protocol, the enhanced protocol, the basic protocol in combination with STEF, and the enhanced protocol in combination with STEF. The following parameters have been used[4]:

- number of sensor nodes in a cluster (including *CH*): 5, 10, 20, 30
- number of regular queries and responses per interval: 1, 3, 6
- number of required endorsements: 3, 6, 12, 24
- number of false endorsing nodes: 1, 5, 10, 20
- number of jammed nodes (verification values): 1, 5, 10, 20

The number of nodes in a cluster has the major impact on energy consumption and required storage space. Since the number of endorsements *CH* receives increases with the size of the cluster, *CH* consumes more energy. The progression of the energy consumption of the basic and the enhanced protocol is similar to the STEF protocol. However, both require slightly more energy wherein the enhanced protocol requires more energy than the basic protocol. For example, in a simulation with 30 sensor nodes, 6 queries per interval, 20 queries totally, and no adversary, the energy consumption for a *CN* is 1.3% (4.1%) higher for the basic (enhanced) protocol compared to the STEF protocol. For *CH*, the energy consumption is 3.2% (7.1%) higher. The required storage space for *CH* is also higher since *CH* has to store more endorsements in a larger cluster. For the just mentioned example, the required storage space can be up to ten times higher compared to the STEF protocol. However, even in this worst case scenario, the storage requirements are still feasible on current sensor hardware. For WSNs with a large cluster size, it is reasonable to change the role of *CH* to distribute the energy consumption to the nodes of the cluster.

It is obvious that less queries result in a lower energy consumption and lower storage requirements since less messages are sent and *CH* needs to store less endorsements.

The number of required endorsements has only a negligible impact on the energy consumption since it influences only the size of the response message which contains more node identifiers.

The number of false endorsing nodes has minimal impact on energy consumption and storage requirements. Since in the simulations excluded sensor nodes are not considered anymore, *CH* does not receive endorsements from these nodes. As a result, the energy consumption of *CH* is slightly lower when more nodes are excluded. Similarly, the storage requirements are slightly lower since *CH* has to store less endorsements, pairwise keys, and verification values.

[4]Note that not all parameter combinations are applicable for all simulations.

The number of jammed verification values has low impact on the energy consumption. Since the number of messages CH receives decreases with the number of jammed messages, the energy consumption decreases also. However, the storage requirements are slightly higher since CH has to store more endorsements for future verifications.

In summary, the simulation results confirm the low energy overhead and the feasibility of the storage requirements on current sensor node hardware. All details of the implementation and the performed simulations can be found in [96].

6.10 Summary

In this chapter, we presented a basic and an enhanced protocol to address FEDoS attacks. They are based on a detection and recovery strategy. Compromised sensor nodes that perform a FEDoS attack are detected and excluded. To achieve this, sensor nodes must prove at a later point in time that they did not perform a FEDoS attack. In the basic protocol, sensor nodes are immediately excluded if the proof fails or a node does not perform the proof. However, if jamming attacks are possible, an adversary may prevent a sensor node to perform the proofs. In this case, the enhanced scheme can be used that is resistant against this attack by introducing a greylist approach.

The security analyses show that the scheme can efficiently detect and exclude compromised nodes performing a FEDoS attack. Our performance analyses show that both protocols are feasible for current sensor hardware. The storage requirements are even for large sensor networks with a long operation time feasible. Furthermore, the actual required storage in the RAM of the node is very small. The energy consumption of both protocols is low since it requires only cheap operations and the exchange of few, short messages. In particular, the message for the sink is short compared to other protocols. The additional overhead of the enhanced protocol is only marginal compared to the original protocol.

Both protocols can be used to extend existing protocols addressing false data injection and PDoS attacks to additionally cope with FEDoS attacks.

In addition, both protocols do not restrict the normal operation of the WSN by, e.g., introducing a delay in the report transmission to the sink. The report to the sink is sent without delay (except for the local generation) to the sink. Thus, in the (common) case where no attack is performed, the network can operate normally. If an attack occurs, the attack is detected ex post but then the malicious node is locally excluded.

7 Preventing Insider Attacks

In Sections 3.2.1, 3.2.2, and 3.3 we argued that prevention of insider attacks is difficult to realize in WSNs. In this chapter, we present a new approach to prevent insider attacks and to detect compromised sensor nodes by using tamper-resistant hardware and efficient attestation protocols.

This chapter shares some material with previously published work *Detecting Node Compromise in Hybrid Wireless Sensor Networks Using Attestation Techniques* [133].

The chapter is organized as follows. In Section 7.1 we introduce and motivate our approach to use tamper-resistant hardware and attestation protocols to prevent insider attacks and to detect compromised sensor nodes. The aspects of different attestation techniques are further illustrated in Section 7.2. We present setting and notation in Sections 7.3 and 7.4. Our proposed protocols are described in Section 7.5. The security of our protocols is discussed in Section 7.6, and a theoretical performance analysis is presented in Section 7.7. In Section 7.8, we give a brief overview of the implementation. Finally, the chapter is summarized in Section 7.9.

7.1 Introduction

Without deploying sensor nodes in physically secured areas, sensor nodes are susceptible to node compromise. An adversary can access a sensor node and can read out all data, such as the cryptographic keys or implemented algorithms and protocols, stored on the node. Using this information, the adversary is able to perform insider attacks.

To prevent an adversary from performing insider attacks, he must be impeded from accessing the cryptographic keys stored on a sensor node. We propose to use the TPM (see Section 2.4) for this purpose. However, due to the large scale and the desired low-cost of WSNs, it is not feasible to integrate a TPM in each individual sensor node. Fortunately, many WSNs are organized in clusters where a minority of special nodes performs some special functions, such as data aggregation, key management, localization, or time synchronization for other sensor nodes. The special nodes usually store shared keys with all other sensor nodes in their vicinity. If an adversary is able to compromise a special node, he gets access to all stored keys. As a result, he is able to frame the sensor nodes in his vicinity, inject false reports, etc. Since special nodes are a valuable target for an adversary, it might be reasonable to equip them with a TPM in scenarios where a high level of security is desired. However, an adversary can still try to tamper with the remaining components of a sensors' system to perform insider attacks. Such tampering attempts have to be detected by neighboring sensor nodes.

One approach to detect compromised nodes is based on attestation techniques, where sensor nodes must prove that their system has not been modified by an adversary. Attes-

tation techniques that have already been proposed for WSNs [211, 210, 209] are software-based and rely on relatively accurate time measurement. These techniques are unsuitable for attestation along multiple hops and when static interferences delay message transmissions, which prevents an exact time measurement.

To overcome these shortcomings, we propose to use the TPM as the trust anchor for attestation protocols. The trust anchor is responsible for providing assurance of delivered attestation values. The TPM provides such a hardware-based trust anchor. It also offers certain cryptographic functions which provide the foundation for attesting the configuration of the local platform to a remote platform. Unfortunately, already proposed TPM-based attestation protocols require asymmetric cryptography and have a high communication and computational overhead [225, 217]. Such complex attestation protocols are unsuitable for resource constrained WSNs.

We propose two efficient TPM-based attestation protocols for hybrid WSNs organized in clusters. Networks consist of low-cost cluster nodes (*CN*s) and more expensive TPM-equipped cluster heads (*CH*s).

The first attestation protocol is highly efficient and allows a number of *CN*s to simultaneously validate the trustworthiness of a *CH* in regular intervals. This protocol is suitable for a continuous monitoring scenario. For example, several *CN*s perform a measurement each minute in parallel and send their data to an aggregating *CH* which sends the aggregate via a direct connection to the sink. A compromised *CH* could seriously interfere with this process by sending for example false aggregates or by refusing to send any aggregates. In case *CH* has been compromised, *CN*s want to know this, to be able to send their measurements to another uncompromised *CH*. Our attestation protocol efficiently ensures all *CN*s that *CH* has not been successfully tampered with, without requiring each *CN* to individually perform an attestation.

The second attestation protocol enables an individual *CN* (or the sink) to verify the trustworthiness of a *CH* at any time. This protocol is suitable for a scenario where *CH* provides services such as key management, localization, or time synchronization for *CN*s. For example, when new nodes have been deployed or mobile nodes enter the vicinity of *CH*, *CH* provides accurate location and time information using an integrated GPS or supports *CN*s to securely establish pairwise keys.

Both protocols do not require expensive public key cryptography on the *CN*s and the exchanged messages are very short. The attestation can be performed even when sensor nodes are multiple hops away from each other. However, due to the unreliable, multihop communication, we can only prove the trustworthiness of the *CH*s, but not untrustworthiness. In addition, these protocols are not limited to cluster-based scenarios. For example the attestation protocols can be used in WSNs where many (mobile) TPM-equipped sinks exist, which are deployed in insecure locations. The network operator can verify if the data received from these sinks is still trustworthy.

7.2 Attestation Techniques

In this section, we compare different attestation techniques and evaluate their applicability for WSNs.

7.2.1 TPM-Based Attestation

Existing attestation protocols [225, 217, 202] are based on the TPM's ability to report the system configuration to a remote party. These approaches are mainly developed for non-resource constrained computer systems. They require extensive computational power from each communication partner to perform public-key cryptography and the transmission of large messages, making these approaches not usable in WSNs (cf. Section 2.4). The complete system configuration, as denoted in the PCRs of the attesting entity, must be transmitted to the verifying entity. The verifying entity evaluates the trustworthiness of the attested entity by comparing the received SML and PCR values with given reference values. Since the verifying entity receives the current platform configuration directly, we refer to this as *explicit attestation*. However, in hybrid WSNs most sensor nodes do not possess enough resources to perform public key cryptography and the transmission of large messages increases the energy consumption significantly. This causes explicit attestation to be inapplicable in WSNs.

To perform an attestation in WSNs, computation intensive operations must be transferred to nodes which possess sufficient computational power, and resource constrained sensor nodes need only to perform minimal verification computations. The sealing concept of the TPM enables an attestation without directly transferring the platform configuration (PCR values and SML). We refer to this as *implicit attestation*. This approach minimizes the amount of transmitted data and does not require public key cryptography on resource constrained nodes. Sealing provides the functionality to bind data to a certain platform configuration. The TPM releases, i.e., decrypts this data only if the current platform configuration is valid. The disadvantage of this approach is that software updates change the values inside the PCRs. Since this results in inaccessible sealed data, this approach is not very applicable in non-resource constrained computer systems, where software configurations change very often through legitimate system updates. Fortunately, the software configuration of sensor nodes often does not change during the whole lifetime of a WSN. Therefore, the attested entity is only able to decrypt a sealed data structure if the current platform configuration matches its initial platform configuration. Our protocols smartly exploit this property to enable a lightweight attestation of the trustworthiness of the attested entity.

7.2.2 Software-Based Attestation

The main disadvantage of TPM-based attestation is that the platform configuration only reflects the initial load-time configuration. Therefore, memory modifications during the runtime cannot be detected, e.g., buffer-overflows. To overcome this shortcoming, attestation software may measure the memory and report the values to a remote party. In

this case, the attestation software forms the trust anchor which must be protected against tampering. In [211, 210, 209, 213], approaches based on measuring the execution time of an optimal attestation routine is introduced. The routine verifies the memory of a sensor node by calculating hash values of randomly selected memory regions. The verifications of the routine cannot be optimized further, i.e., the execution time cannot be made faster, which prevents an adversary from injecting malicious code without detection. However, the success of this approach relies critically on the optimality of the attestation routine and on minimal time fluctuations of the expected responses. Particularly in WSNs with multihop verification and external influences, time intervals for responses can vary. The multihop communication also requires an authenticated communication channel between the verifier and the attestor to reliably identify the number of hops. In scenarios where attestation along multiple hops is required or external interferences prevent an exact time measurement, timing-based software attestation techniques are not applicable since attestations would fail, even though a sensor node is in a trustworthy system state.

7.3 Setting

We assume a hybrid WSN consisting of low-cost cluster nodes (*CN*s) and more expensive TPM-equipped cluster heads (*CH*s). *CN*s are comparable to the sensor nodes described in the device model in Section 4.1. However, *CH*s are assumed to possess much more computing power, memory space, and energy resources, e.g., comparable to the resources of the Stargate platform [55]. The TPM, integrated in the *CH*s, is used to protect keys and other security related data and cannot be circumvented. We do not require any modification of the TPM, such as adding support for symmetric encryption with external data. Since present TPMs only support internal symmetric encryption, some data must be stored temporarily in the Random Access Memory (RAM) of a *CH* for further processing. We assume that access to this temporarily stored data is not possible. For example, in Linux-based systems, such as the Stargate platform, all driver modules and applications that enable direct access to the RAM should be disabled. For this purpose, a static kernel that prohibits the dynamic reloading of modules has to be configured, and all other possibilities to generate a memory dump, such as the `/proc/kcore` file system, has to be disabled. As soon as future TPMs support symmetric encryption with external data this assumption can be revoked. To subvert a *CH*, an adversary must re-program and reboot the node to either modify the system so that access to the RAM is possible or to access the security related data directly. After a reboot with a modified system, the platform configuration is changed and the access to sealed data is no longer possible. Thus, this data is neither accessible directly to the adversary nor loaded into the RAM. To achieve the binding of cryptographic keys to a specific platform configuration, which subsequently prevents rebooting in a compromised system configuration, we assume that we have a reduced measurement architecture, such as IBM's IMA [202], that extends the trust chain specified by the TCG up to the firmware and therefore includes integrity measurement of the kernel and operating system of the *CH*. To aggravate tampering

attempts, we assume mechanisms increasing the effort for an adversary as described in Section 3.2.1 are applied and that OTA programming is disabled.

We assume a slightly different adversary model compared to the one described in Section 4.3. The adversary tries to compromise a *CH* to access stored information, e.g., keying material, and misuse the node to perform insider attacks, e.g., injecting false reports to cause false alarms. Therefore, the adversary can try to read out data or re-program the node to behave according to the purposes of the adversary. However, since the TPM prevents access to the cryptographic keys, our definition of the inside adversary does not exactly fit. In this chapter, an inside adversary tries to tamper with a *CH* but is not able to circumvent the protection of the TPM.

7.4 Notation

CHs are denoted as $CH_i, i = 1, \ldots, a$ and the *CN*s are denoted as $CN_j, j = 1, \ldots, b$, where $b \gg a$.

$E(m, e)$ denotes the *encryption* of data m using an encryption function E and encryption key e. *Encrypted data* m using the key e is denoted with $\{m\}_e$. The *decryption* of $\{m\}_e$ using a decryption function D and the decryption key d is denoted with $D(\{m\}_e, d)$.

Similar to the notation provided in Chapter 6.4, applying a hash function h on data m is denoted with $h(m)$, and a one-way hash chain stored on CH_i, or CN_j is denoted with $C^{CH_i} = c_0^{CH_i}, \ldots, c_\tau^{CH_i}$, or $C^{CN_j} = c_0^{CN_j}, \ldots, c_\tau^{CN_j}$ respectively. The values of the hash chains are generated by applying the hash function h successively on a seed value c_τ, such that $c_\nu = h(c_{\nu+1})$, for $\nu = \tau - 1, \tau - 2, \ldots, 1, 0$.

A specific state of the system of a CH_i is referred to as *platform configuration* $P_{CH_i} := (PCR_0, \ldots, PCR_p)$ and is stored in the appropriate PCRs of the TPM. Data m can be cryptographically bound to a certain platform configuration P_{CH_i} by using the `TPM_Seal` command. Using the `TPM_Unseal` command, the TPM releases, i.e., decrypts m only if the platform configuration has not been modified. This concept allows an implicit attestation to be performed without a direct validation of the PCRs by a *CN*. Since we are abstracting the `TPM_Seal` and `TPM_Unseal` commands, we denote our commands with `Seal` and `Unseal`. Given a non-migratable asymmetric key pair (e_{CH_i}, d_{CH_i}) we denote the *sealing* of data m for the platform configuration P_{CH_i} with $\{m\}_{P_{CH_i}}^{e_{CH_i}} = \texttt{Seal}(P_{CH_i}, e_{CH_i}, m)$. To *unseal* data m it is necessary that the current platform configuration P'_{CH_i} is equal to P_{CH_i}: $m = \texttt{Unseal}(P'_{CH_i} = P_{CH_i}, d_{CH_i}, \{m\}_{P_{CH_i}}^{e_{CH_i}})$.

7.5 Attestation Protocols

In this section, we describe our proposed protocols which enable a CN_j to verify the platform configuration of a CH_i. These protocols represent some basic primitives which can be used in conjunction with or in more complex protocols. Our proposed protocols enable only CN_j to verify the platform configuration of CH_i. To verify the trustworthiness of received data from CN_j, a CH_i has to perform additional mechanisms like redundancy checks or voting schemes.

We have adapted the sealing technique provided by the TPM to realize the implicit attestation (see Section 2.4). We assume, that in the initialization phase the platform configuration of a CH_i is trustworthy. Data needed to perform a successful attestation is sealed in this phase to this platform configuration. Access to this sealed data is only possible if the CH_i is in the initially specified platform configuration. Compromising a CH_i results in a different platform configuration where access to this data is not possible. Thus, a successful attestation is no longer possible.

The first proposed protocol enables a broadcast attestation, where a CH_i broadcasts its platform configuration to its CN_j in periodic intervals. This enables CN_j to verify the platform configuration of the CH_i simultaneously. The second protocol enables a single CN_j (or the sink), to either individually verify the platform configuration of a CH_i using a challenge response protocol or to send data to a CH_i and receive a confirmation that the data has been received correctly and that the CH_i is trustworthy.

7.5.1 Periodic Broadcast Attestation Protocol (PBAP)

In some scenarios, many CN_j perform measurements in parallel and in regular intervals. For example, a couple of CN_j monitor the temperature in a specific region of the WSN. The measurement is performed every 10 minutes to see the change over time. Therefore, the CN_j report their measurement nearly in parallel in specific time intervals to their CH_i. If each CN_j performs an individual attestation of the CH_i, this results in an avoidable overhead. It might be desirable that all CN_j are able to nearly simultaneously verify if their CH_i is still trustworthy using an efficient mechanism. For this purpose, we propose the Periodic Broadcast Attestation Protocol (PBAP).

The PBAP adapts the idea of μTESLA [181] to use one-way hash chains for authentication and extends it to enable attestation in hybrid WSNs. The sealing function of the TPM is used to bind a one-way hash chain to the platform configuration of a CH_i. A CH_i sends the values of the hash chain in periodic intervals and its CN_j can verify these values. The proof of trustworthiness of a CH_i is only possible while its platform configuration has not been modified.

The protocol is divided into two phases. In the *initialization* phase CH_i and CN_j are preconfigured before deployment. In the *attestation* phase, CH_i periodically broadcasts an attestation message. This phase normally lasts for the whole lifetime of the CH_i.

Initialization Before CH_i is deployed, it is preconfigured with a non-migratable public key pair (e_{CH_i}, d_{CH_i}) and a hash chain C^{CH_i}. The seed value $c_\tau^{CH_i}$ of the hash chain is generated on CH_i using the TPM's physical random number generator and used by the CPU to perform the additional computations. CH_i is assumed to possess only one valid platform configuration, denoted as P_{CH_i}. After CH_i is powered up, a measurement about each component (BIOS, bootloader, operating system, applications) is performed, and the related values are stored in the corresponding PCR registers. Each value of the hash chain C^{CH_i} is sealed to this platform configuration P_{CH_i}:

$$\{c_0^{CH_i}\}_{P_{CH_i}}^{e_{CH_i}}, \ldots, \{c_\tau^{CH_i}\}_{P_{CH_i}}^{e_{CH_i}} = \texttt{Seal}(P_{CH_i}, e_{CH_i}, c_0^{CH_i}), \ldots, \texttt{Seal}(P_{CH_i}, e_{CH_i}, c_\tau^{CH_i}).$$

Each CN_j which interacts with a CH_i is configured with the last value $c_0^{CH_i}$ of the hash chain C^{CH_i}. Since the number of CHs is very small compared to the number of CNs, the CNs could be preprogrammed with the values of all CHs. After deployment, the CNs can only keep the values for its CH and another certain number of CHs in their vicinity to save memory.

Attestation CH_i and the associated CNs (denoted as CN_*) are loosely time synchronized. The time is divided into intervals I_λ, $\lambda = 1, \ldots, \tau$. At the beginning of each interval, CH_i sends a broadcast attestation message to CN_*. The attestation messages consist of the values of the hash chain disclosed in reversed order of the generation and the identifier I_λ of the current interval. If the platform configuration of CH_i has not been modified, it can unseal the values of the hash chain C^{CH_i}. In the first interval I_1, CH_i unseals the hash value $c_1^{CH_i}$ and transmits it together with the interval identifier. In the second interval $c_2^{CH_i}$ is unsealed and transmitted and so on. CN_* check if the interval I_1 stated within the message matches their local interval counter I_1' within a certain error range. If they match, CN_* verify whether $h(c_1^{CH_i}) = c_0^{CH_i}$. If the equation holds, CH_i is considered trustworthy and the value $c_0^{CH_i}$ is overwritten with the value $c_1^{CH_i}$. In the next interval, CH_i discloses $c_2^{CH_i}$ and so on, which are similarly checked. The complete protocol is shown in Protocol 7.1 whereas the protocol steps are repeated for each interval I_λ, $\lambda = 1$ to τ.

Protocol 7.1 Periodic Broadcast Attestation Protocol

1. Initialization:
   - CH_i is preconfigured with non-migratable public key pair (e_{CH_i}, d_{CH_i}) and a hash chain C^{CH_i}.
   - The hash chain is sealed to the platform configuration P_{CH_i}.
   - Each interacting CN_j is configured with $c_0^{CH_i}$.
2. Protocol steps:

Interval	Node(s)			Message or Action
I_λ	CH_i:			$\texttt{Unseal}(P_{CH_i}, d_{CH_i}, \{c_\lambda^{CH_i}\}_{P_{CH_i}}^{e_{CH_i}}) = c_\lambda^{CH_i}$
I_λ	CH_i	$\rightarrow$	CN_*:	$c_\lambda^{CH_i}, I_\lambda$
I_λ	CN_*:			$I_\lambda \stackrel{?}{=} I_\lambda'$
I_λ	CN_*:			if $h(c_\lambda^{CH_i}) \stackrel{?}{=} c_{\lambda-1}^{CH_i}$, state of CH_i is valid
I_λ	CN_*:			overwrite $c_{\lambda-1}^{CH_i}$ with $c_\lambda^{CH_i}$
...	...			...

Due to unreliable communication, a CN_j could miss some messages. Thus, CN_j should not immediately declare a CH_i as being untrustworthy but wait for a certain threshold

of time. If a CN_j receives messages again, it can resynchronize by applying the hash function multiple times. For example, assume that a CN_j does not receive the disclosed hash values $c_3^{CH_i}$ and $c_4^{CH_i}$ in interval I_3 and I_4. Receiving the value $c_5^{CH_i}$ in interval I_5, CN_j can resynchronize and verify the platform configuration of CH_i by applying the hash function three times on $c_5^{CH_i}$ and verify if $h(h(h(c_5^{CH_i}))) \stackrel{?}{=} c_2^{CH_i}$. But if CN_j does not receive any attestation message after a certain number of intervals and/or the individual attestation fails, CN_j should react by declaring CH_i as not trusted anymore and choose another CH_i as its CH.

We illustrate PBAP by the following data aggregation scenario where multiple CN_j want to verify in parallel that they can still trust a CH_i.

Example 7.1 *Assume a cluster consisting of CH_1 and several CNs. The CNs are configured with the verification value $c_0^{CH_1}$ for CH_1 and the verification value $c_0^{CH_2}$ of the nearby CH_2. Measurements are performed in regular intervals. PBAP is adapted to these intervals, i.e., attestation messages are sent from CH_1 (and CH_2) shortly before the CNs perform their measurements. After receiving the attestation messages, the CNs perform their measurement and send this data to CH_1 which performs the data aggregation and sends a short message to the sink.*

If the attestation of CH_1 fails or repeatedly no attestation message is received, CNs switch to CH_2 (which is still sending correct attestation messages) and from now on send messages only to this CH. Alternatively, a CN can immediately switch to a backup CH if it does not receive an attestation message if a high reliabilty is desired.

7.5.2 Individual Attestation Protocol (IAP)

Using the Individual Attestation Protocol (IAP), a *CN* (or the sink) can individually verify the platform configuration of a *CH* at any time. Alternatively a *CN* can send data to a *CH* and receive a confirmation that the data has been received correctly and that the *CH* is trustworthy. A *CN* needs only to perform symmetric operations and two short messages need to be exchanged. The messages are very small, because no long public key primitives, such as keys or signatures need to be transmitted. Since transmitting messages is the most cost intensive factor in WSNs [261], this is of particular interest, especially if the sink wants to verify the platform configuration of a *CH*. In this case, messages are transferred along several hops.

The protocol we propose is again divided in *initialization* phase and *attestation* phase. The initialization phase is performed only once after deployment of the sensor nodes while the attestation phase can be performed every time a *CN* (or the sink) wants to verify the platform configuration of a *CH*.

Initialization Each CN_j establishes a shared, symmetric key K_{CN_j,CH_i} with its CH_i. Therefore, existing (non TPM-based) techniques, e.g., [270], might be used. However, we recommend using the key establishment protocol presented in [87], which also assumes a hybrid WSN with TPM-equipped *CH*s and resource constraint *CN*s. This approach has the advantage that key generation within a TPM is inherently more secure than key

generation on off-the-shelf embedded WSN platforms. As in [270], we assume that this short period of time to establish pairwise keys is secure and nodes cannot be compromised. The keys K_{CN_j,CH_i} are sealed on CH_i to its valid platform configuration P_{CH_i}. Thus, CH_i can access these keys only if it is in its valid state.

To enable the sink to perform the attestation with CH_i, a shared symmetric key K_{Sink,CH_i} is preconfigured on CH_i before deployment and sealed likewise.

Attestation A CN_j can verify the platform configuration of CH_i as follows. First, CN_j sends a challenge to CH_i. The challenge consists of an encrypted block containing a *Nonce* and the identifier ID_{CN_j} of CN_j, and additionally ID_{CN_j} in cleartext. K_{CN_j,CH_i} is used for encryption. After receiving the challenge, CH_i unseals K_{CN_j,CH_i} related to ID_{CN_j}. This is only possible if the platform configuration P_{CH_i} is valid. Using this key, CH_i decrypts the encrypted block and verifies if the decrypted identifier is equal to the identifier received in cleartext. If they match, CH_i knows that this message originates from CN_j, encrypts the *Nonce'* using K_{CN_j,CH_i}, and sends it back.[1] Otherwise, CH_i aborts. CH_i then deletes K_{CN_j,CH_i} from the RAM. CN_j decrypts the received response message and checks if the decrypted *Nonce"* matches the *Nonce* it has sent in the first step. If they match, CH_i is declared trustworthy and CN_j can send data to CH_i. This data is encrypted using K_{CN_j,CH_i}. The attestation of CH_i by the sink is performed analog, using the key K_{Sink,CH_i}. The whole protocol is shown in Protocol 7.2.

Protocol 7.2 Individual Attestation Protocol

1. Initialization:
   - CN_j establishes a shared, symmetric key K_{CN_j,CH_i} with its CH_i.
   - Keys K_{CN_j,CH_i} are sealed on CH_i to its platform configuration P_{CH_i}.
   - K_{Sink,CH_i} is preconfigured and sealed on CH_i.
2. Protocol steps:

1	$CN_j \rightarrow CH_i$:	$ID_{CN_j}, \{Nonce, ID_{CN_j}\}_{K_{CN_j,CH_i}}$
2a	CH_i:	$\texttt{Unseal}(P_{CH_i}, d_{CH_i}, \{K_{CN_j,CH_i}\}^{e_{CH_i}}_{P_{CH_i}}) = K_{CN_j,CH_i}$
2b	CH_i:	$D(\{Nonce, ID_{CN_j}\}_{K_{CN_j,CH_i}}, K_{CN_j,CH_i}) = (Nonce', ID'_{CN_j})$
2c	CH_i:	check $ID'_{CN_j} \stackrel{?}{=} ID_{CN_j}$
2d	CH_i:	$E(\{Nonce', ID_{CH_i}\}, K_{CN_j,CH_i}) = \{Nonce', ID_{CH_i}\}_{K_{CN_j,CH_i}}$
2e	$CH_i \rightarrow CN_j$:	$ID_{CH_i}, \{Nonce', ID_{CH_i}\}_{K_{CN_j,CH_i}}$
2f	CH_i:	delete K_{CN_j,CH_i} from RAM
3a	CN_j:	$D(\{Nonce', ID_{CH_i}\}_{K_{CN_j,CH_i}}, K_{CN_j,CH_i}) = (Nonce'', ID'_{CH_i})$
3b	CN_j:	if $Nonce'' \stackrel{?}{=} Nonce$, state of CH_i is valid

[1] However, the trustworthiness of CN_j cannot be assumed, because the node could be potentially compromised and the key is not protected by a TPM.

IAP is useful in scenarios where a single *CN* or the sink wants to verify that no adversary has tampered with a *CH*. We illustrate this by the following example.

Example 7.2 *As in Example 7.1, we assume a data aggregation scenario where CH_1 aggregates measurements from several CNs and sends the aggregated data to the sink. However, measurements are not sent in regular intervals to the sink but only if the measured values exceed a certain threshold. Now we assume that the sink does not receive data from CH_1 for a long period of time. One reason for this could be the compromise of CH_1. IAP can be used to verify that CH_1 can be still trusted. Performing a successful attestation ensures the sink that nobody has tampered with CH_1 and something else is the reason for not receiving data. For example, there is just nothing to report or a number of compromised CNs do not send measurements to CH_1 impeding a successful report generation for the sink.*

However, if the attestation has failed, one cannot conclude that CH_1 is compromised. As already stated in the introduction, attestation along multiple hops is only able to prove trustworthiness, but not untrustworthiness. A compromised en-route sensor node could have dropped the attestation messages which results in a failed attestation. Thus, additional mechanisms are required, e.g., performing a new attestation by using a different route. This issue is investigated in [72].

In certain scenarios it is preferable to transmit data directly within the attestation messages. Therefore, we introduce the optional data fields (shown in squared brackets) *data1* and *data2*. Protocol 7.3 shows the modified IAP, including the optional data fields in squared brackets. Modified IAP can be used in two different ways: (1) using only *data1*, and (2) using both *data1* and *data2*.

The first case can be used in scenarios where an immediate receipt of data is important or where *CN*s send data very infrequently to a *CH*. Therefore data is transmitted directly within the challenge. Thus, in protocol step 1, CN_j sends data in data field *data1* to CH_i within the encrypted block. CH_i can only decrypt this message in step 2b if its platform configuration is valid and access the data. Thus, if CN_j receives the message in step 2e and the checks in steps 3a and 3b succeed, CN_j can be assured that CH_i has successfully received the data and is still trusted.

Example 7.3 *Assume that CHs act more as a second type of sink. CHs have a direct link to the main sink using a powerful transmitter. Thus, CNs send their data to the nearest CH that relays the data directly to the main sink. By applying the modified IAP, CN can receive a confirmation that CH is not compromised and has correctly forwarded the received data to the main sink.*

The latter case is useful in scenarios where a CH_i performs some tasks for a CN_j that requires a feedback back to CN_j. In this case, data field *data1* contains a query from CN_j and data field *data2* contains the response of CH_i. We illustrate this in the following example.

Example 7.4 *An example application for this setting is time synchronization. CHs are equipped with an accurate clock whereas CNs are only equipped with a cheap clock that*

may deviate after a certain time span. To synchronize its local clock, a CN initiates the modified IAP where data1 *contains the query for the actual time. CH generates a response that contains the time of the accurate clock of CH and sends the response back to CN using data field* data2*. By considering roundtrip time etc. (compare time synchronization protocols proposed in [89, 226, 227, 203]), CN calculates the correct time. Since the time synchronization is included in the modified IAP, CN is assured that CH is not compromised and that the received time is, indeed, correct.*

Protocol 7.3 Modified Individual Attestation Protocol

1. Initialization:
   - CN_j establishes a shared, symmetric key K_{CN_j,CH_i} with its CH_i.
   - Keys K_{CN_j,CH_i} are sealed on CH_i to its platform configuration P_{CH_i}.
   - K_{Sink,CH_i} is preconfigured and sealed on CH_i.
2. Protocol steps:

1	$CN_j \rightarrow CH_i$:	$ID_{CN_j}, \{Nonce, ID_{CN_j}, [data1]\}_{K_{CN_j,CH_i}}$
2a	CH_i:	$\texttt{Unseal}(P_{CH_i}, d_{CH_i}, \{K_{CN_j,CH_i}\}^{e_{CH_i}}_{P_{CH_i}}) = K_{CN_j,CH_i}$
2b	CH_i:	$D(\{Nonce, ID_{CN_j}, [data1]\}_{K_{CN_j,CH_i}}, K_{CN_j,CH_i}) = (Nonce', ID'_{CN_j}, [data1])$
2c	CH_i:	check $ID'_{CN_j} \stackrel{?}{=} ID_{CN_j}$
2d	CH_i:	$E(\{Nonce', ID_{CH_i}, [data2]\}, K_{CN_j,CH_i}) = \{Nonce', ID_{CH_i}, [data2]\}_{K_{CN_j,CH_i}}$
2e	$CH_i \rightarrow CN_j$:	$ID_{CH_i}, \{Nonce', ID_{CH_i}, [data2]\}_{K_{CN_j,CH_i}}$
2f	CH_i:	delete K_{CN_j,CH_i} from RAM
3a	CN_j:	$D(\{Nonce', ID_{CH_i}, [data2]\}_{K_{CN_j,CH_i}}, K_{CN_j,CH_i}) = (Nonce'', ID'_{CH_i}, [data2])$
3b	CN_j:	if $Nonce'' \stackrel{?}{=} Nonce$, state of CH_i is valid

7.6 Security Analysis

In this section, we discuss the security of our two proposed attestation protocols. The goal of both protocols is that *CN*s can verify the trustworthiness of *CH*s. We distinguish between the following two strategies an adversary can pursue to circumvent the functionality of our protocols:

1. compromise a *CH* and forge a trusted platform configuration or
2. prevent successful attestations.

In the first case, the adversary wants to deceive *CN*s (and to perform insider attacks) in a compromised environment by forging attestations. In the latter case, the adversary performs attacks to prevent successful attestations although *CH* is not compromised. These attacks can be either performed as an outside adversary or as in inside adversary by compromising en-route *CN*s if the attestation involves multiple hops.

In this analysis, we show that our protocols are resistant against the forging of attestations. However, the wireless multihop communication may enable an adversary to disrupt valid attestations. Thus, our protocols can prove only the trustworthiness of *CH*s. However, untrustworthiness cannot be proven since one cannot distinguish whether a "compromised" *CH*, a compromised en-route *CN*, or transmission errors in the unreliable multihop communication are reasons for invalid, modified, or dropped attestation messages. Thus, additional mechanisms are required to distinguish between this cases when an attestation fails.

As already mentioned in Section 7.5, our proposed protocols enable only *CN*s to verify the platform configuration of *CH*s. Since *CN*s are not protected by a TPM, an adversary is able to compromise them to perform insider attacks. To verify the trustworthiness of received data from *CN*s, a *CH* has to perform additional mechanisms like redundancy checks or voting schemes.

In the following, we separately discuss the security of our protocols. We use our adversary model presented in Section 4.3 to discuss an adversary attacking en-route *CN*s to prevent successful attestations. However, we do not use the adversary model to discuss the adversary attacking a *CH* since the TPM prevents the classical inside adversary. In this case, we consider an adversary that tries to tamper with the remaining components of a *CH*, and/or tries to circumvent the functionality of the TPM.

7.6.1 Security of the PBAP

First, we consider the case where an adversary tries to perform a successful attestation although he has tampered with a *CH* and forge a trustworthy platform configuration. Before trying to access the sealed hash chain directly, an adversary can try alternative ways to forge a trustworthy platform configuration. The adversary could try to perform a replay attack by first blocking the forwarding of legitimate hash values to collect them, then tamper with a *CH*, and finally use this hash values to perform the attestation. However, an adversary would not be successful since hash values are only valid for a specific interval, which is validated by each *CN*. Alternatively, after the adversary has eavesdropped a hash value sent by *CH*, he can try to invert the hash chain to get valid hash values. However, this is a contradiction to the pre-image resistance of the hash function (compare Section 4.3). The only remaining way to get access to valid hash values is to circumvent the functionality of the TPM.

Let's assume that the adversary has successfully tampered with the TPM and has access to the values of the hash chain. Thus, the adversary must have performed one of the following three actions: (1) executed the unseal command under a compromised platform configuration, (2) accessed the key used to seal the hash chain with physical at-

tacks, or (3) decrypted the sealed values of the hash chain by breaking the cryptographic mechanism.

The first case violates our assumption that access to the sealed hash chain is only possible if the platform configuration has not been modified (compare Section 7.3). This prevents the unauthorized extraction of the values of the hash chain in a compromised system environment. The second case violates our assumption that the security mechanisms of the TPM can be circumvented. As described in Section 2.4 the TPM is basically a smartcard and offers high security mechanisms for preventing unauthorized extraction of protected keys. Finally, the third case violates our assumption that the adversary is able to break cryptographic mechanisms (compare Section 4.3). Even if an adversary could access the RAM of a sensor node, he cannot retrieve other hash values, because for each attestation only the actual hash value is unsealed and loaded into the RAM. Thus, an adversary cannot forge a valid platform configuration and perform successful attestations after tampering with a *CH*. Thus, if an attestation of a *CH* was successful, this *CH* can be declared as trustworthy.

However, our approach cannot handle runtime attacks caused by buffer overflows, since we report the platform configuration measured in the initialization phase, i.e., when the software is first executed. Such attacks would result in a (malicious) modified system configuration, but the platform configuration stored in the PCRs is still the valid configuration.

Now, we consider the case where an adversary tries to prevent successful attestations. A **type I adversary (Outsider)** might be able to prevent successful attestations by performing a jamming attack. In the worst case, the adversary is located nearby *CH*, and is able to constantly jam all (attestation) messages of *CH*. In this case, *CH* might be anyway useless and the *CN*s could switch to a new *CH*. However, if the jamming attack stops and *CN*s receive attestation messages from the old *CH* again, they can switch back to the old *CH*. Such jamming attacks are a general problem in wireless communications. To mitigate their impact, additional mechanisms as discussed in Section 2.2.4 can be applied.

An **en-route type II adversary (Insider)** is able to perform Man-in-the-Middle attacks if the attestation is performed between nodes which are multiple hops away from each other. To prevent successful attestations, an adversary can either alter or drop attestation messages.

The altering of an attestation message results in an unsuccessful attestation. To cope with that, a *CN* should possess an additional mechanism which enables the *CN* to reach its *CH* using a different communication path or change to a different *CH*. The *CN* can then use an alternative path and perform the IAP with the *CH* to make a clear statement, whether the route, or the node has been compromised. If the *CH* is compromised, a *CN* could, for example, switch to another *CH* where the communication paths and the new *CH* may not have been compromised.

An adversary is able to drop messages by performing a blackhole or a selective forwarding attack. Since the PBAP is performed in cleartext an adversary can distinguish between attestation and data messages, and therefore, perform a selective forwarding

attack by forwarding attestation messages, but blocking data messages. That kind of attack is a general problem in WSNs and shows that the PBAP is not resistant against all attacks in a multihop scenario with malicious en-route *CN*s. This issue can be mitigated by using multipath routing schemes, as discussed in Section 3.2.8, to forward attestation messages along multiple paths. If the adversary has not compromised nodes on each path to a *CN*, the attestation messages can still reach *CN*.

Alternatively, an adversary could perform further DoS attacks (cf. Section 2.2.4) to prevent attestation messages from reaching their destination.

Finally, note that spoofing or relay attacks are not relevant, because PBAP is not an authentication protocol. It gives an assertion about the trustworthiness of the specific *CH* and not which node has relayed the message.

7.6.2 Security of the IAP

Analogue to the sealing of the hash chain in PBAP, the security of IAP relies on the sealing of the symmetric keys to the valid platform configuration. Thus, an adversary that wants to perform a successful attestation although he has tampered with a *CH* requires access to the symmetric keys. Just as described above, tampering with the TPM will not enable an adversary to access the sealed symmetric keys. Alternatively, an adversary may record some individual attestations before tampering with a *CH* and try to perform a replay attack. However, since a new Nonce is used in each message, such an attempt will be detected. All the adversary can try now is breaking the cryptographic mechanism by analyzing recorded attestations to get access to the symmetric keys. However, as already mentioned above, this is a contradiction to our assumption that this is not possible (compare Section 4.3).

Furthermore, in contrast to WSNs where *CH*s are not equipped with a TPM, a single compromise of a *CH* does not result in the compromise of all shared keys stored on this node. Even using the TPM in only a few sensor nodes, results in a higher resilience to node compromise.

Now, we consider the case where an adversary tries to prevent successful attestations. Analogue to PBAP, a **type I adversary (Outsider)** can perform a jamming attack to prevent a *CN* (and/or *CH*) to receive the attestation messages.

An **en-route type II adversary (Insider)** can perform a blackhole or a selective forwarding attack to prevent attestation messages to reach their destination. However, since attestation messages and data messages have the same form (identifier plus encrypted data block), an adversary cannot distinguish between them to perform a sophisticated selective forwarding attack by blocking only certain messages[2]. Furthermore, the adversary could perform other DoS attacks to prevent attestation messages from reaching their destination as discussed above.

[2]However, an adversary may analyze the different length of the messages to get indications of the message type, e.g., data message, IAP message, modified IAP message. To cope with that, padding could be used to set all messages to the same length.

Furthermore, an adversary can alter a response message sent to a *CN* which results in a failed attestation. *CN* cannot distinguish if either *CH* is compromised, or if the message has been altered by an en-route adversary. Thus, if an attestation fails, a *CN* should first try to perform a new attestation of the same *CH* using another communication path, if possible. If this is not possible or the attestation fails again, either the *CH* or a node on the communication path is compromised. The *CN* should then select a new *CH*, since messages sent to the old one might be susceptible to attacks.

Note that IAP includes an authentication protocol. An en-route adversary acting as a man-in-the-middle cannot spoof the identifier of a *CN* (and/or a *CH*). A *CH* (and/or *CN*) detects the modification of message 1 (and/or message 2e) (see Protocol 7.2) by an en-route adversary, since the included identifier does not match the identifier sent in cleartext. Since a new Nonce is used in each message, replay attacks are not possible.

7.7 Theoretical Performance Analysis

Since we assume that *CH*s possess sufficient resources, we perform our analysis only for the *CN*s. First, we analyze the additional storage requirements. Next, we estimate the additional energy consumption by evaluating the computational and communication overhead.

7.7.1 Storage Requirements

For the PBAP, a *CN* must store one hash value and the identifier for the corresponding *CH*. Depending on the network configuration, it might also store hash values (and identifiers) for other *CH*s in its vicinity. Let L_n, and L_h denote the length of a *node identifier* and a *hash value*, respectively. Let the number of *CH*s for which a *CN* stores values be v. Thus, the storage requirements SR_{PBAP} for a *CN* are:

$$SR_{PBAP} = v \cdot (L_n + L_h) \tag{7.1}$$

Example 7.5 *Suppose a CN stores values for 5 CHs. The length of each hash value is 64 bit and the length of a node identifier is 10 bit. This results in a storage requirements of* $SR_{PBAP} = 46.25byte$.

For the IAP, a *CN* must store one symmetric key for each *CH* with which it wants to perform an attestation. Let this number be denoted by w and the length of a key denoted by L_k. Thus, the storage requirements SR_{IAP} for a *CN* are:

$$SR_{IAP} = w \cdot L_k \tag{7.2}$$

Example 7.6 *Suppose a CN stores keys for 5 CHs. The length of each key is 64 bit. Thus, the storage requirements are* $SR_{IAP} = 40byte$.

The low storage requirements of both PBAP and IAP, enables that the required data can be permanently stored in the 4kbyte SRAM of a Berkeley Mica2 Mote. Both protocols are suitable for current sensor nodes, even if both protocols are used in conjunction.

7.7.2 Energy Consumption

The PBAP requires a *CN* to receive one attestation message and to perform one hash computation at each time interval. An attestation message consists of a hash value and an identifier of the interval, e.g., a counter. Although computing hash values only marginally increases energy consumption [181], we consider the computational overhead, since a hash computation is performed in each time interval.

We use $e_1 = e_{1s} + e_{1r}$ to denote the energy consumed in sending and receiving one byte, and e_2 to denote the energy for one hash computation. In addition to the notation used above, let L_t denote the length needed for the interval identifier. The total number of intervals in the whole lifetime of the network is denoted with t. This results in an additional energy consumption:

$$E_{PBAP} = t \cdot ((L_t + L_h) \cdot e_{1r} + e_2) \tag{7.3}$$

Example 7.7 *Suppose the lifetime of the network is one year and broadcast messages are sent every 10 minutes. Therefore, a 16 bit counter is sufficient for numbering each interval. We use the results presented in [261] to quantify* $e_{1s} = 16.25\ \mu J$ *for sending,* $e_{1r} = 12.5\ \mu J$ *for receiving, and* $e_1 = 28.75\ \mu J$ *for sending and receiving one byte using Berkeley Mica2 Motes. The energy consumed for performing one hash computation using RC5 [195] block cipher is* $e_2 = 15\ \mu J$*. This results in a total energy consumption of 7358.4 mJ. The Mica2 Motes are powered with two 1.5 V AA batteries in series connection. We assume a total capacity of 2750 mAh using standard AA batteries which results in 29700 J. Thus, the ratio of energy consumed in one year by the PBAP is about 0.025% of the total available energy which is negligibly small.*

Just as in our proposed protocols to cope with FEDoS attacks, efficient hash chain constructions (compare Section 6.4) could be used. This would decrease the storage requirements on *CH* and would enable a faster resynchronization if a *CN* has missed multiple hash values. However, since we assume that *CH* possess sufficient resources and the case that *CN* misses many hash values is rare, we consider only the usage of "normal" hash chains.

The IAP requires a *CN* to generate and send a challenge[3], and the verification of the response (see Protocol 7.2, steps 1., 2e., 3a. and 3b.). The challenge requires one Nonce generation, one encryption and one transmission, while the response verification requires the receipt of one message, one decryption and one comparison of two values. Thus, the additional energy consumption is:

$$E_{IAP} = 3 \cdot e_2 + (e_{1s} + e_{1r}) \cdot (2 \cdot L_n + L_h) \tag{7.4}$$

Example 7.8 *We assume the values from above. Additionally, we assume that as in [181], the Nonce is generated using a Message Authentication Code (MAC) as pseudo-random number generator with a generator key* $K_{CN_j}^{rand}$*. Using RC5 for MAC generation*

[3] We do not consider the case where data is sent within the challenge, because we estimate only the additional overhead introduced by our protocol.

requires $e_2 = 15\ \mu J$. The encryption cost using RC5 are also 15 μJ. We neglect the energy cost for the comparison of two values since they are negligibly small. This results in a total energy consumption on a CN for one individual attestation of about 347 μJ which is $1.17 \cdot 10^{-6}$ % of the total available energy.

7.8 Implementation

A proof of concept implementation has been provided using Java and the tpm4java[4] library to show that PBAP and IAP can be realized with current TPMs [72]. The implementation supports the three different types of nodes: *CH*, *CN*, and en-route *CN*. On each *CH* an attestation service runs that is responsible for the communication with the TPM. Depending on a parameter, the attestation service can run either PBAP or IAP. For PBAP, the TPM creates a non migratable asymmetric key pair, generates the seed of the hash chain using the random number generator of the TPM, and generates the hash chain using the SHA-1 module of the TPM. The generated hash chain is sealed using the asymmetric key pair to the platform configuration stored in the PCRs. For IAP, symmetric keys are generated using the javax.crypto package. Another non-migratable asymmetric key pair is generated to seal the symmetric keys. During an attestation, *CH*, *CN*, and en-route *CN* communicate to each other using sockets. The implementation shows that our protocols can be realized with current TPMs.

In addition, to run simulations of an attestation-enhanced WSN in the presence of an adversary, the J-SIM[5] simulation environment has been extended to support our protocols. Furthermore, a routing concept is developed to search for an alternative route after a failed attestation. Details about how these protocols have been implemented can be found in [72].

7.9 Summary

In this chapter, we presented an approach to prevent and detect insider attacks in hybrid WSNs. The WSNs consist of resource constrained *CN*s and TPM-equipped CHs. The TPM chip acts as a trust anchor and prevents unauthorized extraction of cryptographic keys. Since an adversary can still try to tamper with a CH, we propose two attestation protocols to detect such tampering attempts. Both protocols allow *CN*s to verify whether the platform configuration of a *CH* is trustworthy or not, even if they are multiple hops away. The PBAP runs in fixed time intervals, allowing multiple nodes to verify the trustworthiness simultaneously, while the IAP enables a direct attestation. In contrast to WSNs where *CH*s are not equipped with a TPM, a single compromise of a *CH* does not result in the compromise of all shared keys stored on this node. Even using the TPM in only a few sensor nodes, results in a higher resiliency to node compromise.

In contrast to attestation protocols proposed for non-resource constrained systems, our protocols are adapted to WSNs and have a low overhead in terms of storage and

[4] http://tpm4java.datenzone.de
[5] http://www.j-sim.org/

energy consumption. Furthermore, our protocols are applicable in multihop attestation in contrast to timing-based software attestation.

In addition, our proposed protocols are not limited to be used only in WSNs. They are also applicable in other environments where efficiency is also essential, e.g., in embedded systems.

8 Conclusion

In this thesis, we proposed techniques and mechanisms to improve the security of WSNs against insider attacks. Since insider attacks can seriously disrupt the functionality of a WSN and nearly all WSNs are susceptible to insider attacks, appropriate security mechanisms and protocols are required. For a deeper understanding of the different aspects of insider attacks in WSNs, we first proposed a new classification scheme. Moreover, we proposed several protocols to protect against certain types of insider attacks and a general approach to protect against all types of insider attacks.

Our proposed classification enables a systematical categorization of mechanisms and protocols to cope with insider attacks in WSNs. Classifications are based on the applied security strategy, namely: prevention, detection, or recovery. These strategies are further subdivided into mechanisms that apply the respective strategy. We categorized related work and identified properties as well as open problems in the respective areas. We argued that a prevention strategy is hard or even impossible to realize when sensor nodes can be compromised. Detection strategies, especially when based on misbehavior detection, are difficult to realize because of the resource constraints and unreliable wireless communication. In addition, usually only known attacks can be detected. Recovery strategies are often realized with threshold schemes which are only secure when an adversary has not compromised more than a certain number of sensor nodes. Furthermore, recovery is achieved by excluding compromised nodes which requires an efficient mechanism to immediately exclude such nodes. In addition, our proposed classification can serve further purposes. The classification can be used as an introduction to the topic of insider attacks in WSNs by providing a broad overview of the different aspects and related work in the respective areas. Furthermore, the classification can support a secure WSN deployment and make WSN users aware of security related properties and open problems. Before deploying a WSN, a threat analysis should be performed. Our classification can be used as a basis for a systematical threat analysis to identify new threats or threats depending on the application scenario. Appropriate countermeasures may be found in the related work sections of our classification. Likewise, the classification can expedite the development of new or modified security protocols. The modification or combination of existing security protocols can be used to address application specific scenarios or different types of attacks. In addition, the classification may help to secure areas other than WSNs which share similar properties. For example, the classification may be adapted to computational constrained embedded systems used in a car with real-time requirements.

We proposed several protocols to cope with insider attacks in WSNs addressing open problems identified in our classification. The protocols encompass two areas: (1) securing

the report generation process, and (2) developing a general approach to prevent insider attacks and to detect compromised nodes.

Since energy saving is crucial for the operation of a WSN, the report generation process must be protected against PDoS attacks in which an adversary injects a large amount of false report messages to drain the energy resources of message forwarding nodes. Thus, we proposed the STEF protocol to address this issue. We pursued three major goals in the design of STEF. First, false messages should be filtered out as early as possible since the transmission of messages requires a large amount of energy. Second, only efficient cryptography should be used. Third, no threshold scheme should be applied to provide a high resilience to node compromise. Since previously proposed protocols to cope with PDoS attacks use probabilistic filtering and apply threshold schemes, false messages are not filtered out immediately and security breaks down when an adversary compromises a certain number of sensor nodes. In contrast, STEF is able to detect and filter false messages immediately at the next hop and does not rely on a threshold scheme. Thus, even if an adversary compromises an arbitrary number of sensor nodes, he is not able to perform a successful PDoS attack. STEF achieves the second form of recovery by using a mechanism that tolerates compromised nodes and adaptable mechanisms. The typical operation mode of query-response is exploited and verifying nodes need only to perform efficient symmetric cryptography to detect false messages. The theoretical performance analysis and the simulation results showed that STEF is able to significantly reduce the energy consumption through the immediate filtering of false reports while having low storage requirements. In addition, STEF applies a commonly used threshold-based mechanism where multiple sensor nodes collaboratively generate a report to protect against an adversary performing a false data injection attack with the goal to deceive the sink. However, this approach has the drawback of relying on a threshold scheme and is susceptible to FEDoS attacks. FEDoS attacks are a serious threat since only a single compromised node may prevent a successful report generation.

Protection against false data injection attacks should be part of any security mechanism in WSNs. However, since the usually applied collaborative report generation mechanism is susceptible to FEDoS attacks, a new security mechanism is required. Thus, we proposed a basic and an enhanced protocol to cope with false data injection attacks while being resistant against FEDoS attacks. Before our work, only one protocol has been proposed to cope with false data injection and FEDoS attacks. However, this protocol is a specific extension for an already proposed protocol to cope with PDoS attacks and is not able to detect and exclude sensor nodes that perform a FEDoS attack. Our proposed protocols are based on detection and recovery strategies and are able to detect and exclude compromised nodes performing a FEDoS attack. FEDoS attacks are detected without using a resource consuming monitoring mechanism. Instead, sensor nodes have to prove that they have not performed a FEDoS attack, and if the proof fails, they can be immediately excluded. The idea is inspired by the μTESLA broadcast authentication protocol. We modified and adapted the idea to be able to address this new type of attack. The theoretical performance analysis and the simulation results showed the feasibility of our protocols on current sensor hardware. The energy consumption of both protocols is low since only cheap operations and the exchange of few short messages

is required. The storage requirements are feasible even for large sensor networks with a long operation time. Our protocols can also be used in combination with any other protocol. For example, the combination with the STEF protocol provides protection against all three types of attacks: false data injection, PDoS, and FEDoS.

Since in the usually assumed WSNs (just as in our previously proposed protocols) sensor nodes can be compromised, the prevention of insider attacks is hard or even impossible. To address this issue, we investigated mechanisms to prevent sensor nodes from being compromised. The obvious approach of deploying sensor nodes in physically secured areas is often not possible. We proposed to use tamper-resistant hardware in the form of the TPM to prevent node compromise while still being able to deploy sensor nodes in hostile environments. Our approach is designed for the widely-used hybrid WSNs where some special nodes perform some tasks for ordinary sensor nodes. The TPM is used only to protect the special nodes since these nodes are a valuable target for an adversary. The TPM, which has properties similar to a smartcard, protects the cryptographic keys stored on a node. However, an adversary can still try to tamper with the remaining components of a sensor's system to achieve an invalid system state enabling illegal access to the TPM. These tampering attempts have to be detected by neighboring sensor nodes. To achieve this, we proposed two efficient attestation protocols for hybrid WSNs. Attestation enables the detection of all types of insider attacks since it addresses the problem at its root by directly detecting tampering attempts in contrast to unreliable mechanisms such as misbehavior detection. Previously proposed TPM-based attestation protocols typically require asymmetric cryptography and introduce a high communication cost. This makes these protocols unsuitable for resource constrained WSNs. Our proposed protocols, however, are adapted to WSNs and efficiently enable ordinary sensor nodes to verify the trustworthiness of a TPM-equipped node. Ordinary sensor nodes need only to perform efficient operations such as computing hash functions and symmetric cryptography and the few exchanged messages are very short. Furthermore, in contrast to software-based attestation which requires exact time measurement, our protocols enable attestation even if sensor nodes are multiple hops away from each other. We evaluated the performance of our protocols and showed that the overhead for storage and the energy consumption are negligibly small. In addition, our proposed protocols are not limited to use in WSNs. They are also applicable in other environments where efficiency is essential, such as the above mentioned embedded systems.

In summary, this thesis provides an additional step to secure WSNs against insider attacks. However, there still exist open problems or application scenario specific problems. Since the different application scenarios have different security requirements, it is impossible to specify a generic security architecture for all scenarios. Further research could develop guidelines for the development of scenario-specific security architectures. These guidelines could help to systematically identify possible threats and respective mechanisms to cope with these threats. Using the results of our classification could facilitate the development of such guidelines and our proposed protocols to cope with false data injection, PDoS, and FEDoS attacks are suitable candidates for security mechanisms. Another promising area for further research is the use of tamper-resistant hardware in

WSNs. Although tamper-resistant hardware cannot be used in the majority of application scenarios, it is indispensable for most high-security scenarios to prevent insider attacks. Since these scenarios may also have different requirements, new application-specific protocols are needed in addition to our attestation protocols.

Bibliography

[1] 3GPP. Specification of the 3GPP Confidentiality and Integrity Algorithms Document 2: KASUMI Specification. ETSI/SAGE Specification Version: 1.0, 1999.

[2] Afrand Agah, Sajal K. Das, Kalyan Basu, and Mehran Asadi. Intrusion detection in sensor networks: A non-cooperative game approach. In *Proceedings of the Third IEEE International Symposium Network Computing and Applications (NCA)*, pages 343–346. IEEE, 2004.

[3] Ian Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. A survey on sensor networks. *IEEE Communications Magazine 40*, 8:102–114, 2002.

[4] Jamal N. Al-Karaki and Ahmed E. Kamal. Routing techniques in wireless sensor networks: A survey. *Wireless Communications*, 11:6 – 28, 2004.

[5] Abdulrahman Alarifi and Wenliang Du. Diversify sensor nodes to improve resilience against node compromise. In *Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks (SASN)*, pages 101–112. ACM, 2006.

[6] Patrick Albers, Olivier Camp, Jean-Marc Percher, Bernard Jouga, Ludovic Mé, and Ricardo Staciarini Puttini. Security in ad hoc networks: a general intrusion detection architecture enhancing trust based approaches. In *Wireless Information Systems*, pages 1–12, 2002.

[7] Todd R. Andel and Alec Yasinsac. Adaptive threat modeling for secure ad hoc routing protocols. *Electronic Notes in Theoretical Computer Science*, 197 (2):3–14, 2008.

[8] Ross Anderson, Haowen Chan, and Adrian Perrig. Key infection: Smart trust for smart dust. In *Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP)*, pages 206–215. IEEE, 2004.

[9] Ross Anderson and Markus Kuhn. Tamper resistance: a cautionary note. In *Proceedings of the 2nd USENIX Workshop on Electronic Commerce Proceedings*. USENIX Association, 1996.

[10] Ross J. Anderson and Markus G. Kuhn. Low cost attacks on tamper resistant devices. In *Proceedings of the 5th International Workshop on Security Protocols*, pages 125–136. Springer, 1998.

[11] Farooq Anjum, Dhanant Subhadrabandhu, Saswati Sarkar, and Rahul Shetty. On optimal placement of intrusion detection modules in sensor networks. In *Proceedings of the First International Conference on Broadband Networks (BROADNETS)*, pages 690–699. IEEE, 2004.

[12] Kazumaro Aoki, Tetsuya Ichikawa, Masayuki Kanda, Mitsuru Matsui, Shiho Moriai, Junko Nakajima, and Toshio Tokita. Camellia: A 128-bit block cipher suitable for multiple platforms - design and analysis. In *Proceedings of the 7th Annual International Workshop on Selected Areas in Cryptography (SAC)*, pages 39–56. Springer, 2001.

[13] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology (CRYPTO)*, pages 1–18. Springer, 2001.

[14] Lejla Batina, Nele Mentens, Kazuo Sakiyama, Bart Preneel, and Ingrid Verbauwhede. Low-cost elliptic curve cryptography for wireless sensor networks. In *Third European Workshop on Security and Privacy in Ad-Hoc and Sensor Networks (ESAS)*, pages 6–17, 2006.

[15] Alexander Becher, Zinaida Benenson, and Maximillian Dornseif. Tampering with motes: Real-world physical attacks on wireless sensor networks. In *Security in Pervasive Computing*, volume 3934 of *Lecture Notes in Computer Science*, pages 104–118. Springer, 2006.

[16] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Message authentication using hash functions: the HMAC construction. *CryptoBytes*, 2(1):12–15, Spring 1996.

[17] Kemal Bicakci, Chandana Gamage, Bruno Crispo, and Andrew S. Tanenbaum. One-time sensors: A novel concept to mitigate node-capture attacks. In *2nd European Workshop on Security and Privacy in Ad-hoc and Sensor Networks (ESAS)*, pages 80–90. Springer, 2005.

[18] Matt Bishop. *Computer Security: Art and Science*. Addison Wesley, 2003.

[19] Erik-Oliver Blaß. *Sicherer, aggregierender Datentransport in drahtlosen Sensornetzen*. Dissertation, Universitätsverlag Karlsruhe, Karlsruhe, Germany, June 2007. ISBN: 978-3-86644-142-2.

[20] Erik-Oliver Blaß, Holger Junker, and Martina Zitterbart. Effiziente implementierung von public-key algorithmen für sensornetze. In *GI Jahrestagung (2)*, pages 140–144, 2005.

[21] Erik-Oliver Blaß, Joachim Wilke, and Martina Zitterbart. A security–energy tradeoff for authentic aggregation in sensor networks. In *IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2006.

[22] Erik-Oliver Blaß and Martina Zitterbart. Towards acceptable public-key encryption in sensor networks. In *IWUC*, pages 88–93, 2005.

[23] Erik-Oliver Blaßand Martina Zitterbart. An efficient key establishment scheme for secure aggregating sensor networks. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security (ASIACCS)*, pages 303–310. ACM, 2006.

[24] Carlo Blundo, Alfredo De Santis, Amir Herzberg, Shay Kutten, Ugo Vaccaro, and Moti Yung. Perfectly-secure key distribution for dynamic conferences. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO)*, pages 471–486. Springer, 1993.

[25] Prosenjit Bose, Pat Morin, Ivan Stojmenovic, and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001.

[26] David Braginsky and Deborah Estrin. Rumor routing algorthim for sensor networks. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications (WSNA)*, pages 22–31. ACM, 2002.

[27] Michael Brownfield, Yatharth Gupta, and Nathaniel Davis. Wireless sensor network denial of sleep attack. In *Information Assurance Workshop*. IEEE, 2006.

[28] Paul Brutch and Calvin Ko. Challenges in intrusion detection for wireless ad-hoc networks. In *Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT-W)*. IEEE, 2003.

[29] BTnode Platform, 2007. `http://www.btnode.ethz.ch/Main/Overview`.

[30] Sonja Buchegger and Jean-Yves Le Boudec. Performance analysis of the CONFIDANT protocol. In *MobiHoc*, pages 226–236, 2002.

[31] Christian Cachin and Jonathan A. Poritz. Secure intrusion-tolerant replication on the internet. In *Proceedings of the 2002 International Conference on Dependable Systems and Networks (DSN)*, pages 167–176. IEEE, 2002.

[32] Mario Cagalj, Srdjan Capkun, and Jean-Pierre Hubaux. Wormhole-based antijamming techniques in sensor networks. *IEEE Transactions on Mobile Computing*, 6(1):100–114, 2007.

[33] Srdjan Capkun and Jean-Pierre Hubaux. Secure positioning of wireless devices with application to sensor networks. In *IEEE INFOCOM*, pages 1917–1928, 2005.

[34] Srdjan Capkun and Jean-Pierre Hubaux. Secure positioning in wireless networks. *IEEE Journal on Selected Areas in Communications*, 24(2):221–232, 2006.

[35] Alvaro A. Cárdenas, Svetlana Radosavac, and John S. Baras. Detection and prevention of mac layer misbehavior in ad hoc networks. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks (SASN)*, pages 17–22. ACM, 2004.

[36] David W. Carman, Peter S. Kruus, and Brian J. Matt. Constraints and approaches for distributed sensor network security. Technical report, NAI Labs, 2000.

[37] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proceedings of the third symposium on Operating systems design and implementation (OSDI)*, pages 173–186. USENIX Association, 1999.

[38] Seyit A. Çamtepe and Bülent Yener. Combinatorial design of key distribution mechanisms for wireless sensor networks. *IEEE/ACM Trans. Netw.*, 15(2):346–358, 2007.

[39] Haowen Chan, Virgil D. Gligor, Adrian Perrig, and Gautam Muralidharan. On the distribution and revocation of cryptographic keys in sensor networks. *IEEE Trans. Dependable Secur. Comput.*, 2(3):233–247, 2005.

[40] Haowen Chan, Mark Luk, and Adrian Perrig. Using clustering information for sensor network localization. In *Proceedings of IEEE Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2005.

[41] Haowen Chan and Adrian Perrig. PIKE: Peer intermediaries for key establishment in sensor networks. In *Proceedings of IEEE Infocom*, 2005.

[42] Haowen Chan and Adrian Perrig. Efficient security primitives derived from a secure aggregation algorithm. In *Conference on Computer and Communications Security (CCS)*, 2008.

[43] 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 (SP)*, page 197. IEEE, 2003.

[44] Haowen Chan, Adrian Perrig, and Dawn Song. Secure hierarchical in-network aggregation for sensor networks. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS)*, 2006.

[45] Katharine Chang and Kang G. Shine. Distributed authentication of program integrity verification in wireless sensor networks. *ACM Trans. Inf. Syst. Secur.*, 11(3):1–35, 2008.

[46] Shang-Ming Chang, Shiuhpyng Shieh, Warren W. Lin, and Chih-Ming Hsieh. An efficient broadcast authentication scheme in wireless sensor networks. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security (ASIACCS)*, pages 311–320. ACM, 2006.

[47] Xiangqian Chen, Kia Makki, Kang Yen, and Niki Pissinou. Attack distribution modeling and its applications in sensor network security. *EURASIP Journal on Wireless Communications and Networking*, 2008:11 pages, 2008.

[48] Heesook Choi, Sencun Zhu, and Thomas F. La Porta. SET: Detecting node clones in sensor networks. In *SecureComm*, 2007.

[49] Jolyon Clulow, Gerhard P. Hancke, Markus G. Kuhn, and Tyler Moore. So near and yet so far: Distance-bounding attacks in wireless networks. In *Third European Workshop on Security and Privacy in Ad-Hoc and Sensor Networks (ESAS)*, pages 83–97, 2006.

[50] Sinem Coleri, Sing Yiu Cheung, and Pravin Varaiya. Sensor networks for monitoring traffic. In *Forty-Second Annual Allerton Conference on Communication, Control, and Computing, Univ. of Illinois*, 2004.

[51] Mauro Conti, Roberto Di Pietro, Luigi Vincenzo Mancini, and Alessandro Mei. A randomized, efficient, and distributed protocol for the detection of node replication attacks in wireless sensor networks. In *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, pages 80–89. ACM, 2007.

[52] ATMEL Corporation. Atmel atmega128 datasheet, 2007. `www.atmel.com/dyn/resources/prod_documents/doc2467.pdf`.

[53] Crossbow Technology Inc. MICA2 datasheet, 2007. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf`.

[54] Crossbow Technology Inc. MICAz datasheet, 2007. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf`.

[55] Crossbow Technology Inc. Stargate datasheet, 2007. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/Stargate_Datasheet.pdf`.

[56] Crossbow Technology Inc. TelosB Mote, 2007. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf`.

[57] Ana Paula R. da Silva, Marcelo H. T. Martins, Bruno P. S. Rocha, Antonio A. F. Loureiro, Linnyer B. Ruiz, and Hao Chi Wong. Decentralized intrusion detection in wireless sensor networks. In *Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks (Q2SWinet)*, pages 16–23. ACM, 2005.

[58] Erik Dahmen and Christoph Krauß. Short hash-based signatures for wireless sensor networks. In *The 8th International Conference on Cryptology and Network Security (CANS)*, 2009.

[59] Budhaditya Deb, Sudeept Bhatnagar, and Badri Nath. ReInForM: Reliable information forwarding using multiple paths in sensor networks. In *Proceedings of the 28th Annual IEEE International Conference on Local Computer Networks (LCN)*. IEEE, 2003.

[60] Jing Deng, Richard Han, and Shivakant Mishra. INSENS: Intrusion-tolerant routing for wireless sensor networks. Technical Report CU-CS-939-02, Department of Computer Science, University of Colorado, 2002.

[61] Jing Deng, Richard Han, and Shivakant Mishra. A performance evaluation of intrusion-tolerant routing in wireless sensor networks. In *Second International Workshop on Information Processing in Sensor Networks (IPSN)*, pages 349–364, 2003.

[62] Jing Deng, Richard Han, and Shivakant Mishra. Intrusion tolerance and anti-traffic analysis strategies for wireless sensor networks. In *DSN*, 2004.

[63] Jing Deng, Richard Han, and Shivakant Mishra. Countermeasures against traffic analysis attacks in wireless sensor networks. In *Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM)*, pages 113–126. IEEE, 2005.

[64] Jing Deng, Richard Han, and Shivakant Mishra. Defending against path-based DoS attacks in wireless sensor networks. In *Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, pages 89–96. ACM, 2005.

[65] Jing Deng, Richard Han, and Shivakant Mishra. Insens: Intrusion-tolerant routing for wireless sensor networks. *Computer Communications*, 29(2):216–230, 2006.

[66] Jing Deng, Richard Han, and Shivakant Mishra. Secure code distribution in dynamically programmable wireless sensor networks. In *Proceedings of the fifth international conference on Information processing in sensor networks (IPSN)*, pages 292–300. ACM, 2006.

[67] Tim Dierks and Eric Rescorla. The transport layer security (TLS) protocol version 1.1. Network Working Group, Request for Comments: 4346, 2006.

[68] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[69] Gianluca Dini and Ida Maria Savino. An efficient key revocation protocol for wireless sensor networks. In *Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks (WOWMOM)*, pages 450–452. IEEE, 2006.

[70] Danny Dolev and Andrew Chi-Chih Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.

[71] Qi Dong, Donggang Liu, and Peng Ning. Pre-authentication filters: providing dos resistance for signature-based broadcast authentication in sensor networks. In *Proceedings of the first ACM conference on Wireless network security (WiSec)*, pages 2–12. ACM, 2008.

[72] Xin Dong. Attestation over multiple hops in hybrid Wireless Sensor Networks. Diploma Thesis, TU-Darmstadt, 2008.

[73] John R. Douceur. The sybil attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems (IPTPS)*, pages 251–260. Springer, 2002.

[74] Jawad Drissi and Qijun Gu. Localized broadcast authentication in large sensor networks. In *Proceedings of the International conference on Networking and Services (ICNS)*. IEEE, 2006.

[75] W. Du, J. Deng, Y. Han, and P. Varshney. A witness-based approach for data fusion assurance in wireless sensor networks. In *Global Communications Conference*, pages 1435–1439. ACM, 2003.

[76] 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 communications security (CCS)*, pages 42–51. ACM, 2003.

[77] Wenliang Du, Lei Fang, and Peng Ning. Lad: localization anomaly detection for wireless sensor networks. *J. Parallel Distrib. Comput.*, 66(7):874–886, 2006.

[78] Wenliang Du, Ronghua Wang, and Peng Ning. An efficient scheme for authenticating public keys in sensor networks. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, pages 58–67. ACM, 2005.

[79] Xiaojiang Du, Mohsen Guizani, Yang Xiao, and Hsiao-Hwa Chen. Defending dos attacks on broadcast authentication in wireless sensor networks. In *IEEE International Conference on Communications (ICC)*, 2008.

[80] Prabal K. Dutta, Jonathan W. Hui, David C. Chu, and David E. Culler. Securing the deluge network programming system. In *Proceedings of the fifth international conference on Information processing in sensor networks (IPSN)*, pages 326–333. ACM, 2006.

[81] Claudia Eckert. *IT-Sicherheit: Konzepte, Verfahren, Protokolle*. R. Oldenbourg-Verlag, 2007.

[82] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469 – 472, 1985.

[83] Matthias Enzmann, Christoph Krauß, and Claudia Eckert. PDoS-resilient push protocols for sensor networks. In *Third International Conference on Sensor Technologies and Applications (SENSORCOMM)*. IEEE, 2009.

[84] Laurent Eschenauer and Virgil D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM conference on Computer and communications security (CCS)*, pages 41–47. ACM, 2002.

[85] Chih fan Hsin and Mingyan Liu. A distributed monitoring mechanism for wireless sensor networks. In *Proceedings of the 1st ACM workshop on Wireless security (WiSE)*, pages 57–66. ACM, 2002.

[86] Chandana Gamage, Jussipekka Leiwo, Kemal Bicakci, Crispo Bruno, and Andrew S. Tanenbaum. A cost-efficient counter-intrusion scheme for one-time sensor networks. In *2nd International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, 2005.

[87] Saurabh Ganeriwal, Srivaths Ravi, and Anand Raghunathan. Trusted platform based key establishment and management for sensor networks. Unpublished.

[88] Saurabh Ganeriwal and Mani B. Srivastava. Reputation-based framework for high integrity sensor networks. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks (SASN)*, pages 66–77. ACM, 2004.

[89] Saurabh Ganeriwal, Srdjan Čapkun, Chih-Chieh Han, and Mani B. Srivastava. Secure time synchronization service for sensor networks. In *Proceedings of the 4th ACM workshop on Wireless security (WiSe)*, pages 97–106. ACM, 2005.

[90] Deepak Ganesan, Ramesh Govindan, Scott Shenker, and Deborah Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(4):11–25, 2001.

[91] Deepak Ganesan, Bhaskar Krishnamachari, Alec Woo, David Culler, Deborah Estrin, and Stephen Wicker. An empirical study of epidemic algorithms in large scale multihop wireless networks. Technical Report IRB-TR-02-003, Intel Research, 2002.

[92] Prasanth Ganesan, Ramnath Venugopalan, Pushkin Peddabachagari, Alexander Dean, Frank Mueller, and Mihail Sichitiu. Analyzing and modeling encryption overhead for sensor network nodes. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications (WSNA)*, pages 151–159. ACM, 2003.

[93] Gunnar Gaubatz, Jens-Peter Kaps, and Berk Sunar. Public key cryptography in sensor networks - revisited. In *Security in Ad-hoc and Sensor Networks (ESAS)*, 2004.

[94] David Gay, Philip Levis, Robert von Behren, Matt Welsh, Eric Brewer, and David Culler. The nesC language: A holistic approach to networked embedded systems. In *Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation (PLDI)*, pages 1–11. ACM, 2003.

[95] Gnuplot Website, 2007. `http://www.gnuplot.info/`.

[96] Maik Görtz. Implementierung und Analyse von Protokollen gegen FEDoS Angriffe. Bachelor Thesis, TU-Darmstadt, 2008.

[97] Bogdan Groza. Broadcast authentication protocol with time synchronization and quadratic residues chain. In *Proceedings of the The Second International Conference on Availability, Reliability and Security (ARES)*, pages 550–557. IEEE, 2007.

[98] Wenjun Gu, Xun Wang, Sriram Chellappan, Dong Xuan, and Ten H. Lai. Defending against search-based physical attacks in sensor networks. In *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference*, 2005.

[99] 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)*, volume 3156/2004 of *Lecture Notes in Computer Science*, pages 119–132. Springer, 2004.

[100] Carl Hartung, James Balasalle, and Richard Han. Node compromise in sensor networks: The need for secure systems. Technical Report CU-CS-990-05, Department of Computer Science, University of Colorado, Boulder, January 2005.

[101] Tian He, Sudha Krishnamurthy, Liqian Luo, Ting Yan, Lin Gu, Radu Stoleru, Gang Zhou, Qing Cao, Pascal Vicaire, John A. Stankovic, Tarek F. Abdelzaher, Jonathan Hui, and Bruce H. Krogh. Vigilnet: An integrated sensor network system for energy-efficient surveillance. *TOSN*, 2(1):1–38, 2006.

[102] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8 (HICSS)*, page 8020. IEEE, 2000.

[103] Jason Hill, Mike Horton, Ralph Kling, and Lakshman Krishnamurthy. The platforms enabling wireless sensor networks. *Commun. ACM*, 47(6):41–46, 2004.

[104] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors. *SIGPLAN Not.*, 35(11):93–104, 2000.

[105] Jason L. Hill and David E. Culler. Mica: A wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, 2002.

[106] Sang ho Park and Taekyoung Kwon. An efficient message broadcast authentication scheme for sensor networks. In *International Conference on Computational Intelligence and Security (CIS)*, pages 427–432, 2005.

[107] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *Third International Symposium on Algorithmic Number Theory*, volume 1423/1998 of *Lecture Notes in Computer Science*. Springer, 1998.

[108] Lingxuan Hu and David Evans. Secure aggregation for wireless networks. In *Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT-W)*. IEEE, 2003.

[109] Lingxuan Hu and David Evans. Using directional antennas to prevent wormhole attacks. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2004.

[110] Yih-Chun Hu, Markus Jakobsson, and Adrian Perrig. Efficient constructions for one-way hash chains. In *Applied Cryptography and Network Security (ACNS)*, 2005.

[111] Yih-Chun Hu, Adrian Perrig, and David Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. *Wireless Networks Journal*, 11:21–38, 2005.

[112] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Packet leashes: A defense against wormhole attacks in wireless networks. In *IEEE INFOCOMM 2003*, 2003.

[113] Dijiang Huang and Deep Medhi. Secure pairwise key establishment in large-scale sensor networks: An area partitioning and multigroup key predistribution approach. *ACM Trans. Sen. Netw.*, 3(3):16, 2007.

[114] Jean-Pierre Hubaux, Levente Buttyán, and Srdan Capkun. The quest for security in mobile ad hoc networks. In *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing (MobiHoc)*, pages 146–155. ACM, 2001.

[115] Jonathan W. Hui and David Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys)*, pages 81–94. ACM, 2004.

[116] Joengmin Hwang and Yongdae Kim. Revisiting random key pre-distribution schemes for wireless sensor networks. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks (SASN)*, pages 43–52. ACM, 2004.

[117] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Sixth Annual International Conference on Mobile Computing and Networking (MobiCOM)*, 2000.

[118] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Netw.*, 11(1):2–16, 2003.

[119] ISO. Information technology - open systems interconnection - basic reference model: The basic model. ISO/IEC 7498-1, 1994.

[120] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li-Shiuan Peh, and Daniel Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet. In *ASPLOS*, pages 96–107, 2002.

[121] Chris Karlof, Naveen Sastry, and David Wagner. TinySec: A link layer security architecture for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys)*, pages 162–175. ACM, 2004.

[122] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: attacks and countermeasures. In *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*, pages 113–127, 2003.

[123] Brad Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking (MobiCom)*, pages 243–254. ACM, 2000.

[124] Stephen Kent and Karen Seo. Security architecture for the internet protocol. Network Working Group, Request for Comments: 4301, 2005.

[125] Dong Seong Kim, Mohammed Golam Sadi, and Jong Sou Park. A key revocation scheme for mobile sensor networks. In *Frontiers of High Performance Computing and Networking ISPA 2007 Workshops*, volume 4743/2007 of *Lecture Notes in Computer Science*, pages 41–49. Springer, 2007.

[126] Joon Wan Kim, Yong Ho Kim, Hwaseong Lee, and Dong Hoon Lee. A practical inter-sensor broadcast authentication scheme. In *HCI (5)*, pages 399–405, 2007.

[127] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[128] Oliver Kömmerling and Markus G. Kuhn. Design principles for tamper-resistant smartcard processors. In *Proceedings of the USENIX Workshop on Smartcard Technology on USENIX Workshop on Smartcard Technology (WOST)*. USENIX Association, 1999.

[129] Christoph Krauß, Markus Schneider, Kpatcha Bayarou, and Claudia Eckert. STEF: A secure ticket-based en-route filtering scheme for wireless sensor networks. In *2nd International Conference on Availability, Reliability and Security (ARES) 2007*. IEEE, 2007.

[130] Christoph Krauß, Markus Schneider, and Claudia Eckert. Defending against false-endorsement-based DoS attacks in wireless sensor networks. In *Proceedings of the First ACM Conference on Wireless Network Security (WiSec)*. ACM, 2008.

[131] Christoph Krauß, Markus Schneider, and Claudia Eckert. An enhanced scheme to defend against false-endorsement-based DoS attacks in WSNs. In *International Workshop on Security and Privacy in Wireless and Mobile Computing, Networking and Communications (SecPri WiMob)*. IEEE, 2008.

[132] Christoph Krauß, Markus Schneider, and Claudia Eckert. On handling insider attacks in wireless sensor networks. *Information Security Technical Report*, 13:165–172, 2008. Elsevier.

[133] Christoph Krauß, Frederic Stumpf, and Claudia Eckert. Detecting node compromise in hybrid wireless sensor networks using attestation techniques. In *Fourth European Workshop on Security and Privacy in Ad hoc and Sensor Networks (ESAS)*, volume 4572 of *Lecture Notes in Computer Science*, pages 203–217. Springer, 2007.

[134] Sandeep S. Kulkarni and Mahesh Arumugam. INFUSE: A TDMA based data dissemination protocol for sensor networks. Technical report, Michigan State Univ., East Lansing, MI, USA, 2004.

[135] Sandeep S. Kulkarni and Limin Wang. MNP: Multihop network reprogramming service for sensor networks. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 7–16. IEEE, 2005.

[136] Leslie Lamport. Password authentication with insecure communication. *Commun. ACM*, 24(11):770–772, 1981.

[137] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[138] Patrick E. Lanigan, Rajeev Gandhi, and Priya Narasimhan. Sluice: Secure dissemination of code updates in sensor networks. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2006.

[139] Yee Wei Law, Jeroen Doumen, and Pieter Hartel. Survey and benchmark of block ciphers for wireless sensor networks. *ACM Trans. Sen. Netw.*, 2(1):65–93, 2006.

[140] Yee Wei Law, Lodewijk van Hoesel, Jeroen Doumen, Pieter Hartel, and Paul Havinga. Energy-efficient link-layer jamming attacks against wireless sensor network mac protocols. In *Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks (SASN)*, pages 76–88, 2005.

[141] Loukas Lazos and Radha Poovendran. SeRLoc: Robust localization for wireless sensor networks. *ACM Trans. Sen. Netw.*, 1(1):73–100, 2005.

[142] Jooyoung Lee and Douglas R. Stinson. Deterministic key predistribution schemes for distributed sensor networks. In *Selected Areas in Cryptography (SAC)*, pages 294–307, 2004.

[143] Seungjoon Lee, Bohyung Han, and Minho Shin. Robust routing in wireless ad hoc networks. In *Proceedings of the 2002 International Conference on Parallel Processing Workshops (ICPPW)*. IEEE, 2002.

[144] Suk-Bok Lee and Yoon-Hwa Choi. A resilient packet-forwarding scheme against maliciously packet-dropping nodes in sensor networks. In *Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks (SASN)*, pages 59–70. ACM, 2006.

[145] Suk-Bok Lee and Yoon-Hwa Choi. A secure alternate path routing in sensor networks. *Comput. Commun.*, 30(1):153–165, 2006.

[146] Arjen K. Lenstra. Unbelievable security. Matching AES security using public key systems. In *ASIACRYPT '01: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security*, pages 67–86. Springer, 2001.

[147] Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. In *Proceedings of the Third International Workshop on Practice and Theory in Public Key Cryptography (PKC)*, pages 446–465. Springer, 2000.

[148] Arjen K. Lenstra and Eric R. Verheul. The XTR public key system. In *Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO)*, pages 1–19. Springer, 2000.

[149] Philip Levis, Neil Patel, David Culler, and Scott Shenker. Trickle: a self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, 2004.

[150] Feng Li and Jie Wu. A probabilistic voting-based filtering scheme in wireless sensor networks. In *Proceeding of the 2006 international conference on Communications and mobile computing (IWCMC)*, pages 27–32. ACM, 2006.

[151] Donggang Liu and Peng Ning. Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks. In *Network and Distributed System Security Symposium (NDSS)*, 2003.

[152] Donggang Liu and Peng Ning. Location-based pairwise key establishments for static sensor networks. In *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks (SASN)*, pages 72–82. ACM, 2003.

[153] Donggang Liu and Peng Ning. Multilevel μTESLA: Broadcast authentication for distributed sensor networks. *Trans. on Embedded Computing Sys.*, 3(4):800–836, 2004.

[154] Donggang Liu, Peng Ning, and Wenliang Du. Detecting malicious beacon nodes for secure location discovery in wireless sensor networks. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 609–619. IEEE, 2005.

[155] Donggang Liu, Peng Ning, and Rongfang Li. Establishing pairwise keys in distributed sensor networks. *ACM Trans. Inf. Syst. Secur.*, 8(1):41–77, 2005.

[156] Donggang Liu, Peng Ning, Sencun Zhu, and Sushil Jajodia. Practical broadcast authentication in sensor networks. In *Proceedings of the The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS)*, pages 118–132. IEEE, 2005.

[157] Ting Liu, Christopher M. Sadler, Pei Zhang, and Margaret Martonosi. Implementing software on resource-constrained mobile sensors: experiences with impala and zebranet. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services (MobiSys)*, pages 256–269. ACM, 2004.

[158] Konrad Lorincz, David J. Malan, Thaddeus R. F. Fulford-Jones, Alan Nawoj, Antony Clavel, Victor Shnayder, Geoffrey Mainland, Matt Welsh, and Steve Moulton. Sensor networks for emergency response: Challenges and opportunities. *IEEE Pervasive Computing*, 3(4):16–23, 2004.

[159] Mark Luk, Adrian Perrig, and Bram Whillock. Seven cardinal properties of sensor network broadcast authentication. In *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, 2006.

[160] YoungJae Maeng, Abedelaziz Mohaisen, and DaeHun Nyang. Secret key revocation in sensor networks. In *4th International Conference Ubiquitous Intelligence and Computing (UIC)*, volume 4611 of *Lecture Notes in Computer Science*, pages 1222–1232. Springer, 2007.

[161] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications (WSNA)*, pages 88–97. ACM, 2002.

[162] David J. Malan, Matt Welsh, and Michael D. Smith. A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography. In *First IEEE International Conference on Sensor and Ad Hoc Communications and Networks*, 2004.

[163] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking (MobiCom)*, pages 255–265. ACM, 2000.

[164] K. Martinez, P. Padhy, A. Elsaify, G. Zou, A. Riddoch, J. K. Hart, and H. L. R. Ong. Deploying a sensor network in an extreme environment. In *Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing -Vol 1 (SUTC)*, pages 186–193. IEEE, 2006.

[165] Dmitriy Martynov, Jason Roman, Samir Vaidya, and Huirong Fu. Design and implementation of an intrusion detection system for wireless sensor networks. In *IEEE International Conference on Electro/Information Technology*, 2007.

[166] Mitsuru Matsui. New block encryption algorithm MISTY. In *Proceedings of the 4th International Workshop on Fast Software Encryption (FSE)*, pages 54–68. Springer, 1997.

[167] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001.

[168] Victor S Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology (CRYPTO 1985)*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer, 1986.

[169] Walther Müller. Implementierung und Analyse des STEF Protokolls für drahtlose Sensornetze. Bachelor Thesis, TU-Darmstadt, 2008.

[170] James Newsome, Elaine Shi, Dawn Song, and Adrian Perrig. The sybil attack in sensor networks: analysis & defenses. In *Proceedings of the third international symposium on Information processing in sensor networks (IPSN)*, pages 259–268. ACM, 2004.

[171] Edith C. H. Ngai, Jiangchuan Liu, and Michael R. Lyu. On the intruder detection for sinkhole attack in wireless sensor networks. In *IEEE International Conference on Communications*, 2006.

[172] Peng Ning, An Liu, and Wenliang Du. Mitigating DoS attacks against broadcast authentication in wireless sensor networks. *ACM Trans. Sen. Netw.*, 4(1):1–35, 2008.

[173] NIST. SKIPJACK and KEA algorithm specifications, version 2.0, 1998.

[174] National Institute of Standards and Technology (NIST). Advanced Encryption Standard (AES) (FIPS PUB 197), 2001. `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`.

[175] Ilker Onat and Ali Miri. An intrusion detection system for wireless sensor networks. In *IEEE International Conference on Wireless And Mobile Computing, Networking And Communications (WiMob)*, 2005.

[176] Taejoon Park and Kang G. Shin. Soft tamper-proofing via program integrity verification in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 4(3):297–309, 2005.

[177] Bryan Parno, Mark Luk, Evan Gaustad, and Adrian Perrig. Secure sensor network routing: a clean-slate approach. In *Proceedings of the 2006 ACM CoNEXT conference*, pages 1–13. ACM, 2006.

[178] Bryan Parno, Adrian Perrig, and Virgil Gligor. Distributed detection of node replication attacks in sensor networks. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy (SP)*, pages 49–63. IEEE, 2005.

[179] Adrian Perrig, Ran Canetti, Dawn Song, and J. D. Tygar. Efficient and secure source authentication for multicast. In *Network and Distributed System Security Symposium, NDSS '01*, pages 35–46, February 2001.

[180] Adrian Perrig, Ran Canetti, J.D. Tygar, and Dawn Xiaodong Song. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Symposium on Security and Privacy*, pages 56–73, May 2000.

[181] Adrian Perrig, Robert Szewczyk, J. D. Tygar, Victor Wen, and David E. Culler. SPINS: security protocols for sensor networks. *Wirel. Netw.*, 8(5):521–534, 2002.

[182] Dennis Pfisterer, Martin Lipphardt, Carsten Buschmann, Horst Hellbrueck, Stefan Fischer, and Jan Hendrik Sauselin. MarathonNet: adding value to large scale sport events - a connectivity analysis. In *Proceedings of the first international conference on Integrated internet ad hoc and sensor networks (InterSense)*. ACM, 2006.

[183] Krzysztof Piotrowski, Peter Langendoerfer, and Steffen Peter. How public key cryptography influences wireless sensor node lifetime. In *Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks (SASN)*, pages 169–176. ACM, 2006.

[184] S. Zhu-V. Narayanan P. McDaniel M. Kandemir R. Brooks Pirretti, M. The sleep deprivation attack in sensor networks: Analysis and methods of defense. *International Journal of Distributed Sensor Networks*, 2(3):267–287, 2006.

[185] Eric Platon and Yuichi Sei. Security software engineering in wireless sensor networks. *Progress in Informatics*, 5:49–64, 2008.

[186] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys)*, pages 95–107. ACM, 2004.

[187] Joseph Polastre, Robert Szewczyk, and David Culler. Telos: enabling ultra-low power wireless research. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*. IEEE, 2005.

[188] 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)*, pages 255–265. ACM, 2003.

[189] B. Randell, P. Lee, and P. C. Treleaven. Reliability issues in computing system design. *ACM Comput. Surv.*, 10(2):123–165, 1978.

[190] David Raymond, Randy Marchany, Michael Brownfield, and Scott Midkiff. Effects of denial of sleep attacks on wireless sensor network mac protocols. In *Information Assurance Workshop*, 2006.

[191] David R. Raymond and Scott F. Midkiff. Denial-of-service in wireless sensor networks: Attacks and defenses. *IEEE Pervasive Computing*, 7(1):74–81, 2008.

[192] Cryptographic Key Length Recommendation. http://www.keylength.com/.

[193] Kui Ren, Wenjing Lou, Kai Zeng, and Patrick J. Moran. On broadcast authentication in wireless sensor networks. *IEEE Transactions on Wireless Communications*, 6(11):4136–4144, 2007.

[194] Kui Ren, Wenjing Lou, and Yanchao Zhang. Multi-user broadcast authentication in wireless sensor networks. In *IEEE SECON*, 2007.

[195] Ronald L. Rivest. The RC5 Encryption Algorithm. In *Proceedings of the 1994 Leuven Workshop on Fast Software Encryption*, pages 86–96. Springer, 1995.

[196] Ronald L. Rivest, M.J.B. Robshaw, R. Sidney, and Y.L. Yin. The RC6 block cipher, version 1.1, 1998.

[197] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 26(1):96–99, 1983.

[198] Rodrigo Roman, Cristina Alcaraz, and Javier Lopez. A survey of cryptographic primitives and implementations for hardware-constrained sensor network nodes. *Mob. Netw. Appl.*, 12(4):231–244, 2007.

[199] Rodrigo Roman, Jianying Zhou, and Javier Lopez. Applying intrusion detection systems to wireless sensor networks. In *3rd IEEE Consumer Communications and Networking Conference (CCNC)*, 2006.

[200] Tanya Roosta, Sameer Pai, Phoebus Chen, Shankar Sastry, and Stephen Wicker. Inherent security of routing protocols in ad-hoc and sensor networks. In *IEEE Global Telecommunications Conference (GLOBECOM)*, 2007.

[201] Shad Roundy, Dan Steingart, Luc Frechette, Paul Wright, and Jan Rabaey. Power sources for wireless sensor networks. In *European Workshop on Wireless Sensor Networks*, 2004.

[202] Reiner Sailer, Xiaolan Zhang, Trent Jaeger, and Leendert van Doorn. Design and Implementation of a TCG-based Integrity Measurement Architecture. In *13th USENIX Security Symposium*. IBM T. J. Watson Research Center, August 2004.

[203] David Sanchez. Secure, accurate and precise time synchronization for wireless sensor networks. In *Proceedings of the 3rd ACM workshop on QoS and security for wireless and mobile networks (Q2SWinet)*, pages 105–112. ACM, 2007.

[204] Naveen Sastry, Umesh Shankar, and David Wagner. Secure verification of location claims. In *Proceedings of the 2003 ACM workshop on Wireless security (WiSe)*, pages 1–10. ACM, 2003.

[205] Naveen Sastry and David Wagner. Security considerations for IEEE 802.15.4 networks. In *Proceedings of the 3rd ACM workshop on Wireless security (WiSe)*, pages 32–42. ACM, 2004.

[206] Mohit Saxena. Security in wireless sensor networks - a layer based classification. Technical report, Purdue University, 2007.

[207] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. Twofish: A 128-bit block cipher, 1998.

[208] Arvind Seshadri, Mark Luk, and Adrian Perrig. SAKE: Software attestation for key establishment in sensor networks. In *Proceedings of the 2008 International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2008.

[209] Arvind Seshadri, Mark Luk, Adrian Perrig, Leendert van Doorn, and Pradeep Khosla. SCUBA: Secure Code Update By Attestation in Sensor Networks. In *Proceedings of the 5th ACM workshop on Wireless security (WiSe)*, pages 85–94. ACM, 2006.

[210] Arvind Seshadri, Mark Luk, Elaine Shi, Adrian Perrig, Leendert van Doorn, and Pradeep Khosla. Pioneer: verifying code integrity and enforcing untampered code execution on legacy systems. In *Proceedings of the twentieth ACM symposium on Operating systems principles (SOSP)*, pages 1–16. ACM, 2005.

[211] Arvind Seshadri, Adrian Perrig, Leendert van Doorn, and Pradeep Khosla. SWATT: SoftWare-based ATTestation for Embedded Devices. In *IEEE Symposium on Security and Privacy*, 2004.

[212] Stefaan Seys and Bart Preneel. Efficient cooperative signatures: A novel authentication scheme for sensor networks. In *Security in Pervasive Computing*, volume 3450/2005 of *LNCS*, pages 86–100, 2005.

[213] Mark Shaneck, Karthikeyan Mahadevan, Vishal Kher, and Yongdae Kim. Remote software-based attestation for wireless sensors. In *Second European Workshop on Security and Privacy in Ad-hoc and Sensor Networks (ESAS)*, pages 27–41, 2005.

[214] Cory Sharp, Shawn Schaffert, Alec Woo, Naveen Sastry, Chris Karlof, Shankar Sastry, and David Culler. Design and implementation of a sensor network system for vehicle tracking and autonomous interception. In *Proceeedings of the Second European Workshop on Wireless Sensor Networks*, 2005.

[215] Yulong Shen, Jianfeng Ma, and Qingqi Pei. Research on the resilience of key management in sensor networks. In *Proceedings of the 2007 International Conference on Computational Intelligence and Security (CIS)*, pages 716–720. IEEE, 2007.

[216] Elaine Shi and Adrian Perrig. Designing secure sensor networks. *Wireless Communication Magazine*, 11(6):38–43, December 2004.

[217] Elaine Shi, Adrian Perrig, and Leendert Van Doorn. BIND: A Fine-Grained Attestation Service for Secure Distributed Systems. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy (SP)*, pages 154–168, 2005.

[218] IEEE Computer Society. IEEE standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks, part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs), 2006. `http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf`.

[219] Hui Song, Liang Xie, Sencun Zhu, and Guohong Cao. Sensor node compromise detection: the location perspective. In *Proceedings of the 2007 international conference on Wireless communications and mobile computing (IWCMC)*, pages 242–247. ACM, 2007.

[220] Frank Stajano. *Security for Ubiquitous Computing*. John Wiley and Sons, February 2002.

[221] Frank Stajano and Ross J. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Proceedings of the 7th International Workshop on Security Protocols*, pages 172–194. Springer, 2000.

[222] Fred Stann and John Heidemann. RMST: Reliable data transport in sensor networks. In *First International Workshop on Sensor Network Protocols and Applications*, 2003.

[223] Jennifer G. Steiner, B. Clifford Neuman, and Jeffrey I. Schiller. Kerberos: An authentication service for open network systems. In *USENIX Winter*, pages 191–202, 1988.

[224] Mario Strasser and Harald Vogt. Autonomous and distributed node recovery in wireless sensor networks. In *Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks (SASN)*, pages 113–122. ACM, 2006.

[225] Frederic Stumpf, Omid Tafreschi, Patrick Röder, and Claudia Eckert. A Robust Integrity Reporting Protocol for Remote Attestation. In *Proceedings of the Second Workshop on Advances in Trusted Computing (WATC)*, 2006.

[226] Kun Sun, Peng Ning, and Cliff Wang. Secure and resilient clock synchronization in wireless sensor networks. *IEEE Journal on Selected Areas in Communications*, 24(2):395–408, 2006.

[227] Kun Sun, Peng Ning, and Cliff Wang. TinySeRSync: secure and resilient time synchronization in wireless sensor networks. In *Proceedings of the 13th ACM conference on Computer and communications security (CCS)*, pages 264–277. ACM, 2006.

[228] Sun Microsystems, Inc. Project Sun SPOT. `http://www.sunspotworld.com`.

[229] Robert Szewczyk, Alan Mainwaring, Joseph Polastre, John Anderson, and David Culler. An analysis of a large scale habitat monitoring application. In *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys)*, pages 214–226. ACM, 2004.

[230] Deborah Estrin Thanos Stathopoulos, John Heidemann. A remote code update mechanism for wireless sensor networks. Technical Report CENS-TR-30, UCLA: Center for Embedded Networked Sensing, November 26. 2003.

[231] Trusted Computing Group. Trusted Platform Module (TPM) specifications. Technical report, `https://www.trustedcomputinggroup.org/specs/TPM`, 2006.

[232] Leif Uhsadel, Axel Poschmann, and Christof Paar. Enabling full-size public-key algorithms on 8-bit sensor nodes. In *Security and Privacy in Ad-hoc and Sensor Networks (ESAS)*, 2007.

[233] University of California Berkeley: TinyOS. http://www.tinyos.net/.

[234] Tijs van Dam and Koen Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys)*, pages 171–180. ACM, 2003.

[235] Peter Volgyesi, Gyorgy Balogh, Andras Nadas, Christopher B. Nash, and Akos Ledeczi. Shooter localization and weapon classification with soldier-wearable networked sensors. In *Proceedings of the 5th international conference on Mobile systems, applications and services (MobiSys)*, pages 113–126. ACM, 2007.

[236] J. P. Walters, Z. Liang, W. Shi, and V. Chaudhary. *Wireless Sensor Network Security: A Survey*, chapter 17. Auerbach Publications, CRC Press, 2006.

[237] Chieh-Yih Wan, Andrew T. Campbell, and Lakshman Krishnamurthy. PSFQ: A reliable transport protocol for wireless sensor networks. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications (WSNA)*, pages 1–11. ACM, 2002.

[238] 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)*, pages 324–328. IEEE, 2005.

[239] Ronghua Wang, Wenliang Du, and Peng Ning. Containing denial-of-service attacks in broadcast authentication in sensor networks. In *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, pages 71–79. ACM, 2007.

[240] Weichao Wang and Bharat Bhargava. Visualization of wormholes in sensor networks. In *Proceedings of the 3rd ACM workshop on Wireless security (WiSe)*, pages 51–60. ACM, 2004.

[241] Xun Wang, Sriram Chellappan, Wenjun Gu, Wei Yu, and Dong Xuan. Search-based physical attacks in sensor networks. In *14th International Conference on Computer Communications and Networks (ICCCN)*, 2005.

[242] Yong Wang, Garhan Attebury, and Byrav Ramamurthy. A survey of security issues in wireless sensor networks. *IEEE Communications Surveys & Tutorials*, 8 (2):2–23, 2006.

[243] 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)*, pages 59–64. ACM, 2004.

[244] André Weimerskirch. *Authentication in Ad-hoc and Sensor Networks*. PhD thesis, Ruhr-University Bochum, 2004.

[245] Geoffrey Werner-Allen, Konrad Lorincz, Matt Welsh, Omar Marcillo, Jeff Johnson, Mario Ruiz, and Jonathan Lees. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, 10(2):18–25, 2006.

[246] Anthony D. Wood, Lei Fang, John A. Stankovic, and Tian He. SIGF: A family of configurable, secure routing protocols for wireless sensor networks. In *Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks (SASN)*, pages 35–48. ACM, 2006.

[247] Anthony D. Wood and John A. Stankovic. Denial of service in sensor networks. *IEEE Computer*, 35(10):54–62, 2002.

[248] Anthony D. Wood, John A. Stankovic, and Sang H. Son. JAM: A jammed-area mapping service for sensor networks. In *Proceedings of the 24th IEEE International Real-Time Systems Symposium RTSS*, page 286. IEEE, 2003.

[249] Bradley J. Wood. An insider threat model for adversary simulation, 2000.

[250] Bing Wu, Jianmin Chen, Jie Wu, and Mihaela Cardei. A survey on attacks and countermeasures in mobile ad hoc networks. Wireless Mobile Networks Security, ch. 5, pp. 103–135, Springer, 2007.

[251] Thomas Wu, Michael Malkin, and Dan Boneh. Building intrusion tolerant applications. In *Proceedings of the 8th conference on USENIX Security Symposium (SSYM)*. USENIX Association, 1999.

[252] Abhishek Jain Xin Zhang, Haowen Chan and Adrian Perrig. Bounding packet dropping and injection attacks in sensor networks. Technical Report CMU-CyLab-07-019, CyLab, Carnegie Mellon University, 2007.

[253] Kai Xing, Shyaam Sundhar Rajamadam Srinivasan, Manny Rivera, Jiang Li, and Xiuzhen Cheng. *Network Security*, chapter Attacks and Countermeasures in Sensor Networks: A Survey. Springer, 2007.

[254] Ning Xu, Sumit Rangwala, Krishna Kant Chintalapudi, Deepak Ganesan, Alan Broad, Ramesh Govindan, and Deborah Estrin. A wireless sensor network for structural monitoring. In *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys)*, pages 13–24. ACM, 2004.

[255] Wenyuan Xu, Ke Ma, Wade Trappe, and Yanyong Zhang. Jamming sensor networks: Attack and defense strategies. *IEEE Network*, 20(3):41–47, 2006.

[256] Wenyuan Xu, Wade Trappe, and Yanyong Zhang. Anti-jamming timing channels for wireless networks. In *Proceedings of the first ACM conference on Wireless network security (WiSec)*, pages 203–213. ACM, 2008.

[257] Hao Yang and Songwu Lu. Commutative cipher based en-route filtering in wireless sensor networks. In *IEEE VTC Wireless Security Symposium*, 2004.

[258] Hao Yang, Fan Ye, Yuan Yuan, Songwu Lu, and William Arbaugh. Toward resilient security in wireless sensor networks. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, pages 34–45. ACM, 2005.

[259] Yi Yang, Xinran Wang, Sencun Zhu, and Guohong Cao. SDAP: a secure hop-by-hop data aggregation protocol for sensor networks. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, pages 356–367. ACM, 2006.

[260] Yi Yang, Xinran Wang, Sencun Zhu, and Guohong Cao. Distributed software-based attestation for node compromise detection in sensor networks. In *Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems (SRDS)*, pages 219–230. IEEE, 2007.

[261] Fan Ye, Haiyun Luo, Songwu Lu, and Lixia Zhang. Statistical en-route filtering of injected false data in sensor networks. In *IEEE INFOCOM*, 2004.

[262] Wei Ye, John Heidemann, and Deborah Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Trans. Netw.*, 12(3):493–506, 2004.

[263] Yan Yua, Ramesh Govindan, and Deborah Estrin. Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks. Technical Report UCLA-CSD TR-01-0023, UCLA Computer Science Department, May 2001.

[264] Ji-Hoon Yun, Il-Hwan Kim, Jae-Han Lim, and Seung-Woo Seo. WODEM: Wormhole attack defense mechanism in wireless sensor networks. In *Ubiquitous Convergence Technology*, volume 4412 of *Lecture Notes in Computer Science*, pages 200–209. Springer, 2006.

[265] Qing Zhang, Ting Yu, and Peng Ning. A framework for identifying compromised nodes in wireless sensor networks. *ACM Trans. Inf. Syst. Secur.*, 11(3):1–37, 2008.

[266] Wensheng Zhang and Guohong Cao. Group rekeying for filtering false data in sensor networks: A predistribution and local collaboration-based approach. In *IEEE INFOCOM*, 2005.

[267] Yanchao Zhang, Wei Liu, Wenjing Lou, and Yuguang Fang. Location-based compromise-tolerant security mechanisms for wireless sensor networks. *IEEE Journal on Selected Areas in Communications*, 24, Issue 2:247 – 260, 2006.

[268] Yongguang Zhang and Wenke Lee. Intrusion detection in wireless ad-hoc networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking (MobiCom)*, pages 275–283. ACM, 2000.

[269] Li Zhou and Chinya Ravishankar. A fault localized scheme for false report filtering in sensor networks. In *IEEE International Conference on Pervasive Services (ICPS)*, 2005.

[270] 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, 2003.

[271] Sencun Zhu, Sanjeev Setia, and Sushil Jajodia. LEAP+: Efficient security mechanisms for large-scale distributed sensor networks. *ACM Trans. Sen. Netw.*, 2(4):500–528, 2006.

[272] Sencun Zhu, Sanjeev Setia, Sushil Jajodia, and Peng Ning. An interleaved hop-by hop authentication scheme for filtering false data in sensor networks. In *IEEE Symposium on Security and Privacy*, 2004.

[273] ZigBee Alliance. ZigBee Specification, January 2008. `http://www.zigbee.org`.

[274] Wassim Znaidi, Marine Minier, and Jean-Philippe Babau. An Ontology for Attacks in Wireless Sensor Networks. Research Report RR-6704, INRIA, 2008.