Diss. ETH No. 18772
Diss. TIK No. 110

Novel Techniques for Thwarting Communication Jamming in Wireless Networks

A dissertation submitted to
ETH ZURICH

for the degree of
Doctor of Sciences

presented by

MARIO STRASSER

Master of Science ETH in Computer Science
born on April 1, 1978
citizen of Hüttwilen, TG

accepted on the recommendation of
Prof. Dr. Bernhard Plattner
Prof. Dr. Srdjan Čapkun
Prof. Dr. Radha Poovendran
Prof. Dr. Michael Reiter

2009

Abstract

A major challenge in securing wireless applications and services is the inherent vulnerability of radio transmissions to communication jamming Denial-of-Service (DoS) attacks. This vulnerability gains in significance the more one takes the ubiquity of these applications and services for granted and becomes a crucial factor in the context of safety-critical applications. At best, failures of safety-critical systems can result in substantial financial damage—at worst, in loss of life.

In this thesis, we investigate the fundamental primitives that enable jamming-resistant communication and propose novel anti-jamming techniques for scenarios where common anti-jamming techniques cannot be applied. This includes scenarios where network dynamics or lack of trust in the devices prohibits the pre-distribution of shared secrets (a prerequisite for common anti-jamming techniques), or where the use of anti-jamming communication is precluded by the constraints of the employed (e.g., narrowband and single-channel) transceivers.

In the first part of this thesis, we tackle the problem of how devices that do not share any secrets can establish a jamming-resistant communication over a wireless radio channel in the presence of a communication jammer. We address the dependency between anti-jamming spread-spectrum communication and pre-shared keys that is inherent to this problem, and propose Uncoordinated Frequency Hopping (UFH), a novel anti-jamming technique, as a solution to break this dependency. We present and evaluate several UFH-based communication schemes and show their feasibility by means of a prototype implementation. In particular, we illustrate how UFH enables the jamming-resistant execution of (group) key agreement protocols in order to bootstrap common (coordinated) frequency hopping.

In the second part of this thesis, we study the problem of jamming attacks on alarm forwarding in (security- and safety-critical) wireless sensor networks. We argue that common anti-jamming techniques are beyond the capabilities of current sensor nodes and demonstrate the vulnerability to jamming of current forwarding schemes. Prompted by this deficiency, we discuss alternative jamming mitigation techniques and present a novel jamming detection scheme to counter advanced (reactive single bit) jamming attacks. We perform a detailed evaluation of the proposed schemes and validate our findings experimentally. The results show that our solution effectively detects sophisticated jamming attacks and enables the formation of robust sensor networks for the dependable delivery of alarms messages.

Zusammenfassung

Eine wesentliche Herausforderung bei der Realisierung von sicheren drahtlosen Diensten und Applikationen ist die inhärente Verwundbarkeit von Funkkommunikation für Jamming-basierte Denial-of-Service (DoS) Attacken. Diese Verwundbarkeit gewinnt an Bedeutung je mehr man die stetige Verfügbarkeit dieser Anwendungen und Dienste als selbstverständlich betrachtet und wird zu einem kritischen Faktor im Zusammenhang mit sicherheitsrelevanten Anwendungen. Im besten Fall führen Ausfälle von sicherheitsrelevanten Systemen zu erheblichen finanziellen Schäden – im schlimmsten Fall zum Verlust von Menschenleben.

In dieser Arbeit befassen wir uns mit den grundlegenden Prinzipien, welche eine Jamming-resistente Kommunikation ermöglichen und präsentieren neue Techniken für Szenarien, in denen existierende Techniken nicht angewandt werden können. Dies sind insbesondere Szenarien, in denen die Dynamik des Netzwerks oder der Mangel an Vertrauen in die Teilnehmer eine (vorgängige) Verteilung von geheimen Schlüsseln (eine Voraussetzung für gängige Anti-Jamming-Techniken) verhindert, oder die Verwendung von Anti-Jamming-Kommunikation durch die Einschränkunen der eingesetzten Funkmodule (z.B. Schmalband und Einkanal) ausgeschlossen wird.

Im ersten Teil dieser Arbeit befassen wir uns mit dem Problem, wie Kommunikationspartner, die keine geheimen Schlüssel teilen, in der Anwesenheit eines Jammers einen jamming-resistenten Kommunikationskanal über Funk herstellen können. Wir betrachten die Abhängigkeit zwischen Frequenzspreizung-basierter Anti-Jamming-Kommunikation und geteilten geheimen Schlüsseln, welche bezeichnend für dieses Problem ist, und schlagen unkoordiniertes Frequenzspringen (UFH), eine neue Anti-Jamming-Technik, als Mittel zur Auflösung dieser Abhängigkeit vor. Wir präsentieren und evaluieren mehrere auf UFH basierende Kommunikationsverfahren und demonstrieren ihre praktische Realisierbarkeit mit Hilfe eines Prototypen. Des Weiteren veranschaulichen wir, wie UFH die Durchführung eines (Gruppen-)Schlüsselvereinbarungsprotokolls zur Initialisierung von konventionellem (koordiniertem) Frequenzspringen ermöglicht.

Im zweiten Teil dieser Arbeit untersuchen wir das Problem von Jamming-basierten Angriffen auf die Weiterleitung von Alarmmeldungen in (sicherheitsrelevanten) drahtlosen Sensornetzen. Wir argumentieren, dass konventionelle Anti-Jamming-Techniken die Fähigkeiten der gegennwärtigen Sensorknoten übersteigen und zeigen die Angreifbarkeit aktueller

Verfahren zur Nachrichtenweiterleitung durch Jamming. Als Antwort darauf, diskutieren wir alternative Gegenmassnahmen und präsentieren eine neue Technik zur Detektierung von fortgeschrittenen (reaktiven, einzelbit) Jammingattacken. Wir führen eine ausführliche Evaluierung der vorgeschlagenen Techniken durch und validieren unsere Resultate experimentell. Die Ergebnisse zeigen, dass unsere Lösung auch technisch ausgefeilte Jammingattacken effizient detektiert und somit die Erstellung von robusten Sensornetzen für die zuverlässige Vermittlung von Alarmmeldungen ermöglicht.

Acknowledgments

First and foremost, I would like to express my gratitude to Prof. Bernhard Plattner and Prof. Srdjan Čapkun for enabling and supporting this thesis. Both provided assistance in numerous ways, and I sincerely appreciate their invaluable advice and guidance and the time they spent reading drafts and giving honest and constructive feedback. I am also grateful to the co-examiners Prof. Michael Reiter and Prof. Radha Poovendran for their willingness to read and judge this thesis and for providing valuable comments.

Special thanks are due to my colleagues Christina Pöpper and Boris Danev, who co-authored most of the publications that contain the results presented in this thesis, for the numerous (technical and non-technical) discussions that greatly contributed to the successful completion of this thesis. I also want to thank all other people with whom I had the pleasure to collaborate while working on this thesis, notably: Prof. David Basin, Dr. Philipp Blum, Martin Burkhart, Dr. Xenofontas Dimitropoulos, Ghassan O. Karame, Dr. Simon Künzli, Prof. Koen Langendoen, Dr. Martin May, Dr. Andreas Meier, and Dr. Paul E. Sevinç.

I would like to extend my thanks to my office mates Dr. Rainer Baumann, Simon Heimlicher, and Ariane Keller for making our office—with a few construction-noise-related exceptions—a very pleasant place to work. I am also thankful to all the other former and present members of the Communication Systems Group and the System Security Group for the good time and the interesting discussions; thanks to Prof. Eduard Glatz, Prof. Hannes Lubich, Dr. Thomas Dübendorfer, Dr. Ulrich Fiedler, Dr. Stefan Frei, Dr. Merkourios Karaliopoulos, Dr. Franck Legendre, Dr. Vincent Lenders, Dr. Georgios Parissidis, Dr. Lukas Ruf, Dr. Thrasyvoulos Spyropoulos, Dr. Arno Wagner, Dr. Jinyao Yan, Elisa Boschi, Daniela Brauckhoff, Gergely Csúcs, Bernhard Distl, Thomas Heydt-Benjamin, Theus Hossmann, Andreea Picu, Gabriel Popa, Ehud Ben-Porat, Elias Raftopoulos, Kasper B. Rasmussen, Dominik Schatzmann, Bernhard Tellenbach, Nils Ole Tippenhauer, Brian Trammell, and Davide Zanetti. I further thank the TIK technical and administrative staff for providing an excellent service.

I am indebted to my family for their encouragement and support; my mother Anita, my father Georg, my sisters Kristin and Corinna, my brothers in law Antonio and Jean-Marc, my nieces Aurelia, Ann-Cathrin and Josephine, and my nephew Lino. A special thanks goes to my wife Katja for her love, patience, and support.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

The recent advances in computer and communication technology led to the proliferation of wireless mobile devices that enable a multitude of new applications and services. As technology continues to advance so does this trend and more and more wireless applications start to become an integral part of our everyday life. This includes an increasing number of security- and safety-critical applications such as wireless fire or intrusion alarm systems, secure localization and navigation applications, emergency alert broadcasts, and (ad-hoc) communication infrastructures for emergency rescue, law enforcement, or military operations.

A serious threat shared by all these wireless applications and services is that wireless radio communication is inherently insecure against eavesdropping and against communication jamming Denial-of-Service (DoS) attacks. While eavesdropping is a mainly passive attack with the aim to obtain confidential information, jamming is an active attack with the aim to prevent devices from exchanging information by interfering with their communication. Possible communication jamming attacks include signal annihilation, modification (bit-flipping, overshadowing), and jamming as well as the insertion of forged or replayed signals [5, 60, 61]. An attacker who can physically isolate a device by enclosing it in a Faraday's Cage or is powerful enough to jam the whole frequency band in use simultaneously might even be able to completely block all communication from and to a device. To make matters worse, detecting such a jamming attack can be extremely difficult if the attacker pursues a reactive jamming strategy and reduces the amount of jamming to a minimum.

The inherent vulnerability of wireless communication to jamming gains in significance the more one takes the ubiquity of these new wireless applications for granted and becomes a crucial factor in the context of safety-critical applications. At best, failures of safety-critical systems can result in substantial financial damage—at worst, in loss of life. We therefore argue that the availability of jamming-resistant communication is essential for the development of security- and safety-critical applications. Even though one can in general not completely avoid communication jamming, it is nevertheless advisable to make jamming as difficult and expensive for the attacker as possible; that is, to make the communication as jamming-resistant as feasible and desirable.

In this thesis, we investigate the fundamental primitives that enable jamming-resistant communication and propose novel anti-jamming techniques for scenarios where common techniques cannot be applied.

1.1 Jamming Countermeasures

In principle, there are three ways to counter communication jamming: jamming avoidance, jamming detection, and jamming mitigation. The arguably most evident and most effective way is to avoid the jammer by moving out of its range or by switching to a different communication medium (such as a wire) that is not affected by the jamming. But in spite of its effectiveness, avoiding the jammer is almost never possible: most wireless applications and services must be available at a specific location and entirely replacing the wireless communication infrastructure with a wired one is hardly ever a feasible option. The efficiency of jamming detection and localization as a means to counter jamming heavily depends on what the network entities can cause with the obtained information, that is, on whether effective and immediate countermeasures (e.g., the quick deactivation/destruction of the jammer) can be taken. This de facto limits the application of jamming detection to settings where physical intervention is possible (and legal) or where no intermediate actions are required (i.e., where detection of the attacker is sufficient). The third and most common measure against jamming is to mitigate its impact by means of anti-jamming communication techniques that can resist the attack. Possible mitigation techniques include highly directional antennas, forward error-correcting codes, and spread-spectrum communication [5, 60]. Common spread-spectrum anti-jamming communication such as frequency hopping spread spectrum (FHSS) or direct sequence spread spectrum (DSSS) enables the sender to spread a signal (in time and/or frequency) such that its transmission becomes unpredictable for the attacker. Provided that the attacker cannot physically isolate a device, her ability to alter or erase a message is restricted to interfering with the message transmission and is hence limited by the achieved processing gain of the spread-spectrum communication. The processing gain expresses the cost for the attacker to jam such a spread-spectrum transmission in terms of energy or power and is typically in the order of 100 to 1000 times the cost of the sender. The chances of success for such a malicious interference are thus in general sufficiently low—either because the attacker is not powerful enough to achieve more, or because she has no incentive to do so (e.g., if she wants to stay undetected).

However, in order that common anti-jamming communication can be applied, the used radio devices must support some form of spread-spectrum communication and—even more important—a secret spreading code must be shared by the communication partners beforehand. Fulfilling these requirements is not trivial and severely limits the application of common communication anti-jamming techniques. In this thesis, we aim to overcome these limitations and study scenarios in which network dynamics or lack of trust in the devices prohibits the pre-distribution of shared secrets and thus the application of conventional anti-jamming communication; or where the use of anti-jamming communication is precluded by the constraints of the employed (e.g., narrowband and single-channel) transceivers and must be replaced with alternative countermeasures. Specifically, the two problems addressed in this thesis are: (i) anti-jamming communication without pre-shared secrets and (ii) detection of targeted, reactive jamming attacks in (safety-critical) sensor networks. We next discuss the challenges inherent to these problems in more detail and outline our solutions.

1.2 Part I: Anti-jamming Communication without Shared Secrets

In the first part of this thesis, we address the problem of jamming-resistant communication in scenarios in which the communicating parties do not share secret keys. This includes scenarios where the parties are not known in advance or where not all parties can be trusted (e.g., jamming-resistant key establishment or anti-jamming broadcast to a large set of unknown receivers). An inherent challenge in solving this problem is that known anti-jamming communication techniques such as frequency hopping or direct-sequence spread spectrum require that the devices share a secret spreading key (or code) *prior* to the start of their communication. This requirement creates a circular dependency between anti-jamming spread-spectrum communication and key establishment and generally precludes the unanticipated anti-jamming communication between unpaired devices. As a solution to break this dependency, we propose Uncoordinated Frequency Hopping (UFH), a new spread-spectrum anti-jamming technique that does not rely on shared keys. We present and discuss several UFH-based anti-jamming communication schemes and show their usage for various applications, including the establishment of pairwise or group keys in order to bootstrap common coordinated frequency hopping. We thoroughly analyze the performance of our UFH communication schemes analytically and empirically via simulations. We identify an optimal strategy for the

UFH frequency channel selection and show that, although it achieves lower communication throughput, UFH exhibits the same level of anti-jamming protection as common (coordinated) frequency hopping (which, however, cannot be used in scenarios where keys are not pre-shared). We further demonstrate the feasibility of our UFH schemes, in terms of execution time and resource requirements, with a software-radio-based prototype implementation.

1.3 Part II: Detection of Reactive Jamming in Sensor Networks

An integral part of most security- and safety-critical applications is a dependable and timely alarm notification. However, owing to the resource constraints of wireless sensor nodes (i.e., their limited power and spectral diversity), ensuring a timely and jamming-resistant delivery of alarm messages in applications that rely on wireless sensor networks is a challenging task. In order to demonstrate how challenging this task is, we present a state-of-the-art alarm forwarding scheme for wireless sensor networks that is fairly robust against unintentional link failures and investigate its resistance against jamming attacks. We show that in current alarm forwarding schemes blocking alarms by targeted, reactive jamming is not only straightforward, but that this jamming is also very likely to remain unnoticed by existing jamming detection schemes.

In the second part of this thesis we address this problem and propose a novel jamming detection scheme for the identification of such targeted jamming attacks. Our scheme is unique in the sense that it is able to identify the cause of bit errors for individual packets by looking at the received signal strength during the reception of these bits and is well-suited for the protection of reactive alarm systems with very low network traffic. We present three different techniques for the identification of bit errors based on: predetermined knowledge, error-correcting codes, and limited node wiring. We perform a detailed evaluation of the proposed solution and validate our findings experimentally with Chipcon CC1000 radios. The results show that our solution effectively detects sophisticated jamming attacks that cannot be detected with existing techniques and enables the formation of robust sensor networks for the dependable delivery of alarm notifications. Our scheme also meets the high demands on the energy efficiency of reactive surveillance applications as it can operate without introducing additional wireless network traffic.

Part I

Anti-jamming Communication without Shared Secrets

Chapter 2
Introduction

The open nature of wireless radio transmissions makes them particularly vulnerable to communication jamming Denial-of-Service (DoS) attacks. The aim of these attacks is to prevent devices from exchanging any useful information by interfering with their communication. Possible communication jamming attacks include signal annihilation, modification (e.g., bit-flipping or overshadowing) and jamming as well as the insertion of forged or replayed signals [5, 60, 61, 70].

A class of well-known countermeasures against communication jamming attacks are *spread-spectrum* techniques such as frequency hopping, direct-sequence spread spectrum, and chirp spread spectrum [60, 61]. Common to all these techniques is that they rely on secret (spreading) codes that are shared between the communication partners. These secret codes enable the sender to spread the signal (in time and/or frequency) such that its transmission becomes unpredictable for a third party, thus reducing the probability of interference. For these schemes to work, however, the required secret code must be shared between the partners *prior to their communication*, generally precluding (unanticipated) transmissions between unpaired devices or from a sender to an unknown set of receivers. The requirement of a shared code has so far been fulfilled by out-of-band code pre-distribution, which suffers from serious scalability problems.

If pre-sharing the codes is not adequate or even infeasible (e.g., because not all communicating devices are known at the time of deployment or because the devices are not trusted to keep the keys secret) the devices must agree on a secret code (or key) in an ad-hoc manner using the wireless channel. However, the execution of a key-establishment protocol relies on jamming-resistant communication which, in turn, requires the availability of a shared secret code. In other words, the dependency of spread-spectrum techniques on a shared key (or code) and the dependency of key establishment on jamming-resistant communication create a circular dependency, which we call *anti-jamming/key-establishment dependency* (see Figure 2.1). We point out that, even if the devices hold mutual public-key certificates issued by a commonly trusted authority, they still need to communicate in order to establish a secret spreading key (e.g., using an authenticated Diffie-Hellman key-establishment protocol) and to bootstrap common coordinated spread-spectrum communication.

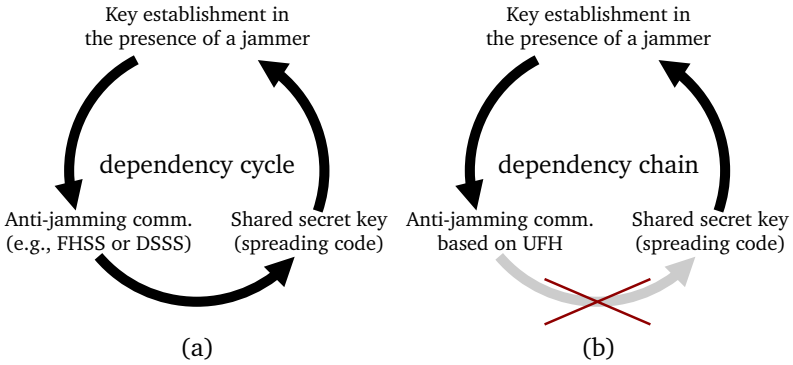

Figure 2.1: *Anti-jamming/Key-establishment dependency graphs. (a) If two devices do not share any secret keys or codes and want to execute a key establishment protocol in the presence of a jammer, they have to use a jamming-resistant communication technique. However, known anti-jamming techniques such as frequency hopping and direct-sequence spread spectrum rely on secret (spreading) codes that are shared between the communication partners prior to the start of their communication. (b) In this work, we break this dependency and propose a novel frequency hopping scheme called Uncoordinated Frequency Hopping (UFH) that enables two parties to execute a key-establishment protocol in the presence of a jammer, even if the parties do not yet share a secret key or code.*

In our present work, we break the dependency between anti-jamming spread-spectrum communication and shared secret keys. We propose a technique called *Uncoordinated Frequency Hopping* (UFH) that enables jamming-resistant (broadcast) communication *without* a pre-shared secret code. We present several UFH-based communication schemes that support the transmission of messages of arbitrary length and show how these schemes enable the execution of (group) key establishment protocols in the presence of a jammer. The established key can then be used by the communication parties to create a secret hopping sequence and to switch to more efficient coordinated frequency hopping for the subsequent communication.

UFH is closely related to coordinated frequency hopping: each message is split into multiple parts and then sent across the air on random hopping frequencies chosen from a fixed frequency band. Like coordinated frequency hopping, UFH is based on the assumption that the attacker cannot

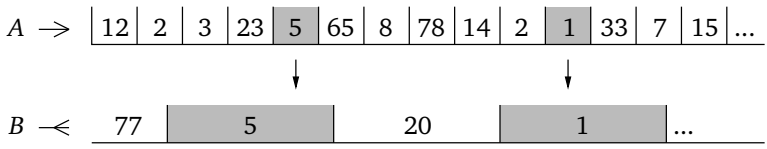

Figure 2.2: *Example of UFH. The numbers indicate the frequency channels where sender A is sending and receiver B is listening over time (here, both send and receive on one frequency at a time). If A and B send and receive simultaneously on the same frequency (5 and 1 in the example), the packet sent on this frequency is successfully transmitted over the undisturbed channel.*

jam all frequency channels on which the devices communicate at the same time so that the sender and receiver can still communicate through the remaining channels. However, unlike in common coordinated frequency hopping, in UFH, the sender and the receiver do not agree on a secret channel sequence but instead transmit and listen on randomly selected channels. Hence, all communication in UFH underlies the observation that, with sufficient transmission attempts, the sender and receiver will send and listen on the same channels in a number of time slots, even if they did not agree on them beforehand (see Figure 2.2). Intuitively, given 200 channels and given a sender hopping among the channels at a high rate of, for instance, 2 kHz, a receiver will be listening on the frequency where the sender is transmitting in average $2000/200 = 10$ times per second (independent of the receiver's choice of the reception channels). Building on this observation, we develop UFH communication schemes that are highly resistant to packet losses, insertions, and active interference by an attacker. They can thus be applied in settings where users want to establish an unanticipated and spontaneous communication without pre-shared keys, which was so far not feasible using coordinated frequency hopping.

2.1 Contributions

In summary, the four main contributions of this part are:

- We address the problem of anti-jamming communication without shared secrets and introduce the anti-jamming/key-establishment circular dependency problem.

- We propose Uncoordinated Frequency Hopping (UFH) as a solution to the addressed problem and develop UFH-based communication schemes that support the transmission of messages of arbitrary length in a jammed environment without relying on shared secrets.
- We develop a comprehensive jammer model for reactive and non-reactive jamming which allows the computation of the number of blocked channels and the probability that a packet is jammed as a function of the packet length and the packet encoding.
- We describe how UFH enables the protection of several applications that could so far not be protected with common spread spectrum techniques. In particular, we show how UFH enables the execution of (group) key establishment protocols in the presence of a jammer; the established key can then be used to support later coordinated frequency hopping communication.

2.2 Outline

The remainder of this part is organized as follows: In Chapter 3, we specify our system and attacker model. We describe our UFH communication schemes in Chapter 4 and analyze their security and performance in Chapter 5 and Chapter 6, respectively. In Chapter 7 we show how our schemes can be used for key establishment and discuss additional applications of UFH. In Chapter 8 we demonstrate the feasibility of the schemes with a prototype implementation. Finally, we discuss related work in Chapter 9.

Chapter 3
System and Attacker Model

In this chapter, we detail our assumptions and prerequisites regarding the considered system and attacker model. If not stated otherwise, denotations introduced in this chapter will be used throughout Part I.

3.1 System Model

We consider a scenario where a set of communication parties which do not share any secret values want to establish a jamming-resistant communication in the presence of a communication jammer. All parties reside within each other's transmission range and are equipped with a full-duplex radio transceiver capable of frequency hopping communication within a set $\mathcal{C}$ of $c = |\mathcal{C}|$ frequency channels. The transceiver can be narrowband or broadband, enabling the parties to send and receive on one or more channels simultaneously; the number of channels on which the transceiver can send and receive on in parallel is denoted by c_t and c_r, respectively. We assume that the transceiver does not leak information about its active reception channels, that is, that the channels on which the transceiver is actively listening cannot be detected by monitoring its radio signal emissions. We further assume that a sender A splits its available transmission power uniformly over its c_t output channels such that it transmits with the same signal strength on all channels. With respect to a specific receiver B, we denote by P_A the strength of A's signal arriving at B and by P_t the minimal required signal strength at B such that B can successfully decode the signal (i.e., the sensitivity of B's receiver). In this context, a transmission between A and B over an undisturbed channel will be successful if $P_A \geq P_t$ and if A sends on a channel on which B is currently listening.

The parties share the same concept of time and their clocks are assumed to be loosely synchronized in the order of seconds (e.g., by means of GPS). Each party A is computationally capable of efficiently performing ECC-based public key cryptography and holds a public/private key pair (K_A, K_A^{-1}), a corresponding public-key certificate σ_A issued by a trusted Certification Authority (CA), and the valid public key K_{CA} of this CA. The keys and certificates were distributed during the system initialization phase (e.g., after the procurement of the devices) and the CA may be off-line or unreachable at the time of communication. To increase the robustness of the message transmissions against interference and jamming, the parties

apply error correcting codes with code rate r_c and resistance ρ to the messages.

3.2 Attacker Model

We consider an omnipresent but computationally bounded adversary that controls the communication channel in the sense that she is able to eavesdrop and insert arbitrary messages but can only modify transmitted messages by adding her own energy-limited signals to the channel. This means that the attacker's ability to alter or erase a message is restricted to interfering with the message transmission and that she cannot disable the communication channel by blocking the propagation of radio signals (e.g., by placing a device in a Faraday's cage).

The attacker's goal is to interfere with the communication of the parties in order to prevent them from exchanging any useful information. That is, the attacker aims at increasing (possibly indefinitely) the time for the message exchange in the most efficient way. In order to achieve this goal, the attacker is not restricted to message jamming only, but can also try to disturb the parties' communication by modifying and inserting messages or by keeping the parties too busy to participate in or proceed with the protocol. More specifically, the attacker can choose among the following actions:

- The attacker can *jam messages* by transmitting signals that cause the original signal to become unreadable by the receiver. The portion of a message the attacker has to interfere with in such a manner depends on the used coding scheme.

- The attacker can *modify messages* by either flipping single message bits or by entirely overshadowing original messages. In the former case, the attacker superimposes a signal on the radio channel that converts one or several bits in the original message from zero to one or vice versa. In the latter, the attacker's signal is of such high power that it entirely covers the original signal at the receiver. As a result, the original signal is reduced to noise in the attacker's signal and the original message is replaced by the attacker's message. In either case, in this attack the signal must remain decodable by the receiver and result in a valid bit sequence.

- The attacker can *insert messages* that she generated by using known (cryptographic) functions and keys as well as by reusing (parts of)

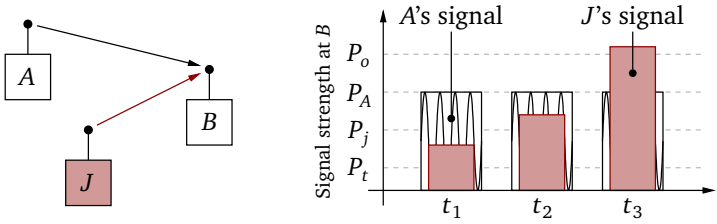

Figure 3.1: *Required signal strengths for different attacker strategies. Let sender A transmit a message to receiver B such that the corresponding signal arrives at B with strength P_A. If an attacker J interferes using a signal that, at B, has lower strength than P_j, then B successfully receives A's message (t_1 in the figure); if, however, J's signal arrives at B with a strength between P_j and P_o, the transmission is jammed, and B receives no message (t_2); finally, if the strength of J's signal at B is even equal or greater than P_o it entirely overshadows A's transmission, and B receives J's message (t_3).*

previously overheard messages (constituting a replay attack). Depending on the signal strength of the inserted messages, these messages might interfere with regular transmissions.

Regarding the interception and jamming of messages, we make the worst case assumption that the attacker is aware of the location and configuration of all devices so that her capabilities are only restricted by the performance of her transceiver. We can therefore abstract away from physical parameters such as device distances, device characteristics (e.g., their antenna gains), and environmental influences, and only consider the power of the original and of the attacker's signal at the receiver. For a given P_A, the strength of the original signal at B, we denote by P_j and P_o the minimal required strength of the attacker's signal at B in order to jam or overshadow a message sent from A to B, respectively (see Figure 3.1). We assume that $P_t < P_A, P_j < P_o$ and that a message from A is successfully received by B if the strength of the attacker's signal at B is less than P_j. In the case of an additive white Gaussian noise (AWGN) channel, for instance, we obtain $P_t = \beta N_B$, $P_j = P_A/\beta - N_B$, and $P_o = \beta(N_B + P_A)$, where N_B is the total noise power at B and β is the minimal required signal-to-noise ratio of B's transceiver to successfully decode the signal. We further assume that the maximal transmission power of the attacker is finite and we denote by P_J the signal strength that the attacker is able

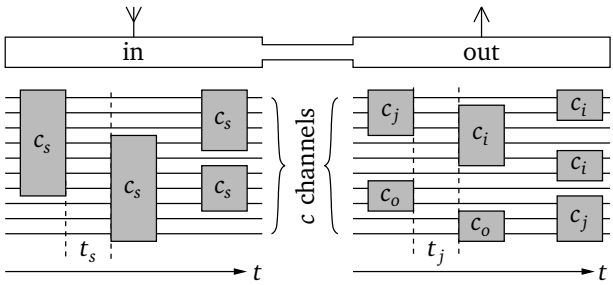

Figure 3.2: *Input and output channel configuration of the attacker. We denote by c_s the number of channels that the attacker can sense in parallel and by t_s (t_j) the required time to switch the frequency of the input (output) channels. The number of channels on which the attacker can send (c_i), jam (c_j), and overshadow (c_o) is bounded by her transmission power.*

to achieve at the receiver B if she transmits with maximal transmission power on a single channel. However, we do not assume any restrictions on the attacker's energy supply, that is, she is considered to be mains-operated. We also assume that the frequency-dependent variance in the signal attenuation is negligible over the communication frequency range of $\mathcal{C}$ and that the attacker can divide her transmission power arbitrarily among all c channels. The only restriction is therefore that for all combinations of output channel assignments in which the attacker inserts on c_i, jams on c_j, and overshadows on c_o channels, $c_i P_t + c_j P_j + c_o P_o \leq P_J$ and $0 \leq c_i, c_j, c_o \leq c$ must hold at all times. Consequently, we can derive $\lfloor \frac{P_J}{P_t} \rfloor$, $\lfloor \frac{P_J}{P_j} \rfloor$, and $\lfloor \frac{P_J}{P_o} \rfloor$ as upper bounds on the number of channels on which the attacker can simultaneously insert, jam or overwrite, respectively. The number of channels that the attacker can concurrently sense is denoted by c_s. We assume that the attacker is able to receive and transmit in parallel, and that the channels on which she receives and transmits can be switched independently of each other (see Figure 3.2). The required time to switch the frequency of the input (output) channels is denoted by t_s (t_j).

Following prior classifications [60], we distinguish between proactive, reactive, and hybrid jammers. Proactive jammers do not sense for ongoing transmissions but permanently jam on c_j channels. *Static* jammers remain on the same channels for much longer than the transmission time of a packet. *Sweep* jammers systematically update the jamming channels in a way that after $\lceil \frac{c}{c_j} \rceil$ jamming cycles all channels have been jammed once (but

they do not have to follow a particular order). *Random* jammers always choose c_j channels at random and might thus jam the same channels several times before having hit all channels. Reactive jammers differ from the above mentioned in that they initially solely sense for ongoing transmissions and enable the jamming channels only when a signal has been detected. *Hybrid* jammers, finally, are a combination of reactive and proactive jammers that have their jamming channels already enabled while scanning for signals (i.e., hybrid jammers can sense and transmit independently). As we shall show, of the introduced jammer types, reactive-sweep jammers are the most general and thus also the most powerful ones.

Chapter 4
Uncoordinated Frequency Hopping Communication

In a Frequency Hopping Spread Spectrum (FHSS) system the sender and the receiver rapidly switch the carrier frequencies of their radio transceivers among a (large) set of frequency channels according to a random hopping sequence. In the case of common, coordinated frequency hopping, this sequence is known to the sender and the receiver and is typically generated by means of a pseudo-random generator which was seeded with a shared secret key. The two main advantages of FHSS communication compared to single carrier communication are a high resistance to (narrowband) interference and a reduced probability of interception.

FHSS communication can further be divided into fast frequency hopping and slow frequency hopping, based on the number of bits sent per hop. The hopping is called fast if there are multiple frequency hops per bit transmission and is called slow if there are multiple bit transmissions per frequency hop. In both cases, the jamming resistance of the scheme is usually expressed by the achieved processing gain, given by the ratio of the width of the whole frequency band in which the channels are located to the bandwith of a single channel. If the channels are orthogonal (i.e., do not overlap) the processing gain is equal to the number of channels among which the sender and the receiver hop.

4.1 Uncoordinated Frequency Hopping

With UFH, the sender and receiver hop among a set of known frequency channels in an uncoordinated and random manner. Information is transferred whenever the receiver happens to listen on the same frequency channel on which the sender is currently transmitting (Figure 2.2). In order for (coordinated or uncoordinated slow) frequency hopping to be effective against jamming, the time slots during which the sender is transmitting on a specific channel must be kept short (i.e., at most a few hundred bits). Messages—in particular if they are authenticated—thus do typically not fit into the sender's short transmission slots and are split into fragments by the sender and reassembled by the receiver. After the fragmentation, the sender encapsulates each fragment into a packet, encodes the packets

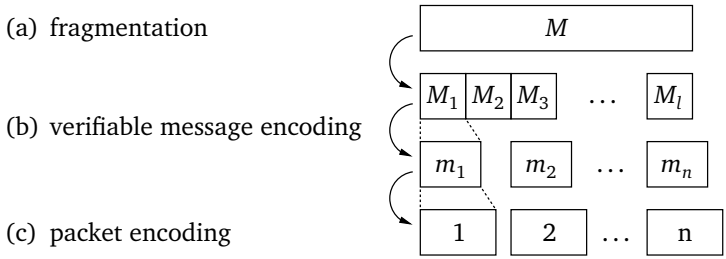

Figure 4.1: *Message fragmentation. For the UFH transmission, the message is (a) split into fragments, then the fragments get message encoded (b), and packet encoded (c). Finally, the sender (repeatedly) transmits the packets on randomly selected frequency channels.*

with error correcting codes, and repetitively transmits the encoded packets one after another on randomly chosen frequency channels (see Figure 4.1).

Receiving a fragment with (coordinated or uncoordinated) frequency hopping requires the receiver to listen on the correct channel for the complete transmission of the fragment. If the sender's and receiver's hopping frequencies were identical (and with it the time that both stay on a channel before hopping to the next), the successful transmission of a fragment would require precisely synchronized transmission and reception slots to avoid partially received fragments. In UFH, we do not require the slots to be synchronized by permitting the receiver to switch the channels less often than the sender (Figure 2.2), thus reducing the number of partially received fragments. Note that this procedure does not affect the jamming resistance of the scheme; in fact, we will show in Chapter 5 that from the attacker's point of view, the probability to jam a transmitted fragment with randomized uncoordinated frequency hopping is equal to the jamming probability in coordinated frequency hopping. The throughput of the communication with UFH is, however, lower than for coordinated hopping: given that the devices did not establish a secret shared key beforehand, the receiver will need numerous reception attempts to receive each fragment.

Whereas UFH message fragmentation is efficient, the reassembly of the message at the receiver is non-trivial and can become very inefficient if the attacker inserts additional fragments or modifies transmitted ones. By way of example, consider that a legitimate message M is divided into l fragments and that z adversarial packets successfully arrive at the receiver

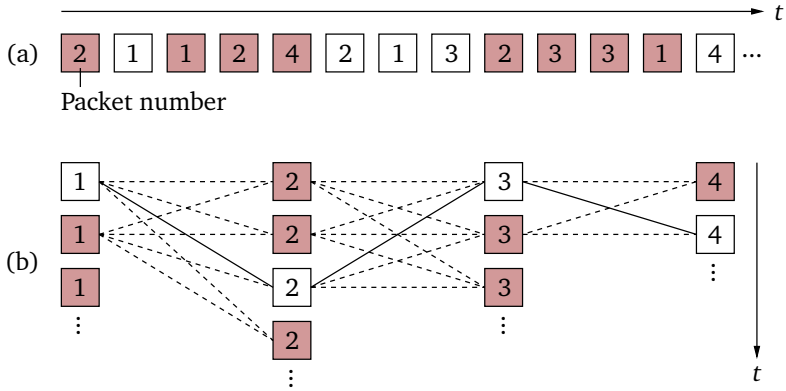

Figure 4.2: *Message reassembly. (a) Example of the packet arrival at the receiver; white packets were sent by the sender, red packets were inserted by the attacker (at the time of reception they cannot be distinguished). (b) The receiver sorts unique packets according to their fragment number (and message id). Without verifiable message coding, the receiver must reassemble and check all combinations of packets (indicated by dashed lines, only a subset is shown). Verifiable message coding enables an efficient reassembly of valid combinations (solid lines).*

during the transmission of M. The number of possible messages that the receiver must reassemble and verify is then in $\mathcal{O}((\frac{z}{l})^l)$. If l is not predefined, adversarial insertions may even lead to an exponential number of message assemblies and verifications at the receiver (i.e., $\mathcal{O}(q^{\lfloor z/q \rfloor})$, where q is the number of unique packets that the attacker inserts per legitimate message fragment, see Figure 4.2). We argue that an attacker can easily insert $z \gg l$ unique packets because the receiver needs l specific packets to reassemble M while the attacker can send any (unique) packets. In a typical setting, where $l = 10$ and where the attacker can insert $z = 100$ unique packets during the legitimate message transmission, the receiver would already need to reassemble and verify 10^{10} messages. This is clearly beyond the capabilities of current devices and would thus constitute a DoS attack on the message reassembly process and consequently block the communication to the receiver. UFH-based communication schemes must therefore take measures that enable the efficient identification of sets of fragments that belong to the same message (without using a shared key) in order to mitigate the impact of maliciously inserted or modified fragments.

We next describe the fragmentation and reassembly process in more detail and present a set of schemes that fulfill this requirement.

4.2 UFH-based Communication Schemes

For a given message size $|M|$ (determined by the application) and a size s of the frequency hopping slots (usually given by the hopping rate of the radio device), the throughput/latency of UFH communication depends not only on the probability that a packet sent by the sender is successfully received by the receiver but also on the number of packets that the receiver must receive to reconstruct the message.

If a message is split into l fragments before the transmission, all fragments must be received so that the message can be reconstructed. The requirement to receive all fragments leads to many redundant receptions, because the more fragments have been received, the less likely it is that the next successfully received fragment will be a new one (coupon collector's problem); after the reception of the second but last fragment, it will take another expected l successful—but redundant and thus useless—receptions to receive the last missing fragment. Erasure codes [50] are a way to relax this requirement and enable message reconstruction if only a subset of all constructed fragments is available. We distinguish:

Erasure Codes Optimal erasure codes encode a message M into n fragments of size $|M|/l$ such that any subset of l fragments can be used to reconstruct M. Near optimal erasure codes are more efficient than optimal codes in terms of coding complexity and memory usage but require a fragment size of $|M|/(l-\varepsilon)$ in order to reconstruct M using l fragments. The constant parameter ε can usually be reduced at the expense of a higher coding complexity. Examples of (near) optimal erasure codes are Reed-Solomon [79] and Tornado [50] codes.

Rateless Erasure Codes Rateless erasure codes (sometimes also called fountain codes) do not generate a finite set of n fragments but a (potentially) infinite fragment sequence. The encoded message can be reconstructed from any set of l different fragments. Examples of efficient near optimal fountain codes are: Online [53], LT [49], and Raptor [66] codes.

In what follows, we denote by $\mathcal{E}(n,l,\varepsilon)$ an erasure code that encodes a message M into n fragments $M_1, M_2, \ldots, M_n$, $|M_i| = |M|/(l-\varepsilon)$, such that

any subset of $l \leq n$ fragments can be used to reconstruct M; for optimal codes $\varepsilon = 0$ and for rateless erasure codes $n = \infty$.

Despite their ability to cope with fragment losses, erasure codes are susceptible to the aforementioned intentional fragment insertions or modifications (pollution attack [41]); that is, they cannot distinguish correct from modified or phony fragments. Hence, the receiver cannot do better than try all possible fragment combinations (Figure 4.2). Limiting the impact of malicious fragments thus requires measures that allow for the efficient identification of sets of fragments that belong to the same message (without shared keys). The therefore required overhead per packet must be kept as low as possible to avoid that the advantage gained by the erasure coding is nullified by a largely increased number of required packet receptions (due to more fragments that the message needs to be split into). Given these constraints, we require a *packet verification technique* to fulfill the following requirements:

Time Efficiency The time to verify (i.e., identify as either belonging to a specific message or being invalid) a packet m_i must be in $\mathcal{O}(N+n)$, where N is the total number of received packets and n is the number of fragments per message.

Space Efficiency The overhead per packet that is required for the verification must be in $\mathcal{O}(n)$.

We define a set of packets $\mathcal{M}$ as being *verifiable* with respect to an erasure code $\mathcal{E}(n,l,\varepsilon)$ and a message M if at least l of the packets in $\mathcal{M}$ can be verified as belonging to the message M. We next present three packet verification techniques that fulfill the above requirements and can be used in combination with (rateless) erasure codes to build *verifiable message coding*.

We point out that the only purpose of these packet verification techniques is to ensure the efficiency of the message reassembly. In particular, they are not intended to provide message authentication or confidentiality (i.e., the attacker may imitate genuine transmissions by inserting self-composed or by replaying overheard messages). These security goals can be achieved on layers running on top of the UFH scheme, for example by using timestamps, message buffers, and public-key cryptography. We assume a security level of k bits for the cryptographic primitives (a strength comparable to a symmetric key of k bits) and denote by $h(\cdot)$ a weak collision-resistant cryptographic hash function with an output length of k bits.

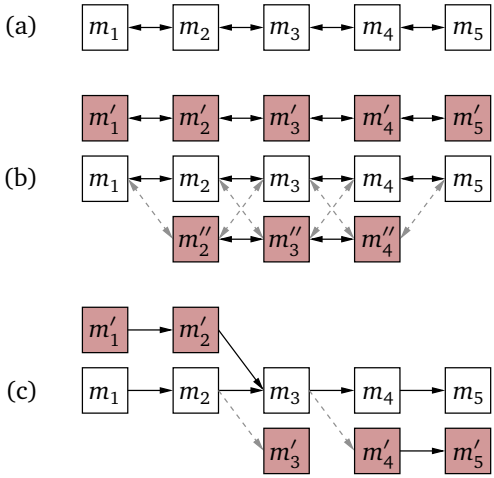

Figure 4.3: *Packet verification. (a) Legitimate packets are logically linked (solid lines) so that they are efficiently identifiable as belonging to the same message. (b) The attacker can insert her own or replayed packets, but she cannot link them to legitimate chains in a way that leads to an amount of possible packet recombinations that grows exponentially with the number of maliciously inserted packets (dashed lines). (c) In the case of the verification based on hash links, for instance, the attacker can create a fusion of her own and a legitimate chain (e.g., by using $h(m_3)$ in m'_2), but she cannot purposefully create a hash link that branches from a legitimate into her own chains (e.g., finding an m'_3 such that $h(m_3) = h(m'_3)$).*

4.2.1 UFH Communication using Hash Links

A straight-forward approach that allows the receiver to efficiently identify fragments of the same message is to arrange all message fragments into a hash chain by linking each fragment to its successor with a hash. In a general solution, each packet is not only linked to its successor but to the next α packets (see Figure 4.4). More precisely, once a message M has been erasure-encoded into the fragments $M_1, M_2, \ldots, M_n$ using $\mathcal{E}(n, l, \varepsilon)$, each fragment M_i is encapsulated into a packet $m_i := id||i||l||M_i||h_{i+1}||\ldots||h_{i+\alpha}$ by adding the message id, the fragment number i, the required number of fragments l, and the hash values of the packets m_{i+1} to $m_{i+\alpha}$. For the non-existing packets m_{n+1} to $m_{n+\alpha-1}$ the hash value of the entire message is used (i.e., $h_i := h(m_i)$ for $1 \leq i \leq n$ and $h_i := h(M)$ otherwise).

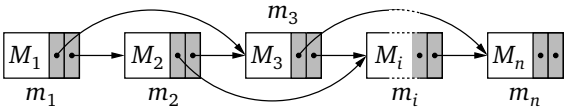

Figure 4.4: *Hash links. Each packet contains the hash value of the next α packets.*

A set $\mathcal{M}$ of $|\mathcal{M}| \geq l$ packets is verifiable if at least l packets in the set are connected (i.e., have no gaps of $\geq \alpha$ missing packets). More formally, the set $\mathcal{M}$ is verifiable if and only if $\exists \mathcal{M}'' \subset \mathcal{M}' \subseteq \mathcal{M}$ such that $|\mathcal{M}''| = l - 1$, $|\mathcal{M}'| = l$ and $\forall m_i \in \mathcal{M}'' : \exists j \in \{i+1, i+2, \ldots, i+\alpha\} : m_j \in \mathcal{M}'$ (i.e., all but the last packet in a chain must have a valid successor to which they link).

Time and Space Efficiency

Upon reception of a new packet m_i, the receiver must identify all packets that link to m_i or to which m_i links. This can be done by traversing all N already received packets once. Note that although each packet links to α other packets, all these α packets are successors of m_i and part of the same chain. Each packet is thus the head of exactly one unique sub-chain and at most $N - 1$ chains join at a packet. These joining chains build a reverse tree that is rooted at the packet. Finding the heads of these chains (i.e., finding the leaves of the reverse three) can be done in $\mathcal{O}(N)$ steps. Hence, the cost to verify m_i (i.e., to find all chains the packet is part of) is in $\mathcal{O}(N)$. The verification-related overhead per packet is αk bits, where k is the length of a hash value.

Security Analysis

In order to violate the integrity of this verification scheme, the attacker must create a branch in a chain by inserting a packet m'_i such that both m_i and m'_i are accepted by the receiver as valid successors of m_{i-1}, that is, the attacker must find a packet $m'_i := id||i||l||M'_i||h'_{i+1}||\ldots||h'_{i+\alpha}$ such that $h(m'_i) = h(m_i)$. However, given that $h(\cdot)$ is a second pre-image resistant hash function, finding such an m'_i is considered infeasible for a computationally bounded attacker. Chains that contain such collisions (i.e., split into branches) will therefore be dropped by the receiver. Note that if $h(\cdot)$ is also collision-resistant, the attacker cannot even find such collisions in the packet chains that she created herself (with non-negligible probability).

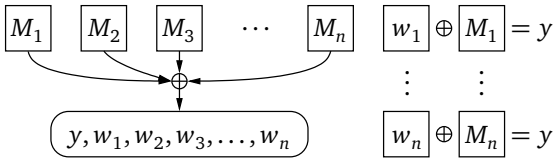

Figure 4.5: *Cryptographic one-way accumulator. The fragments $M_1, M_2, \ldots, M_n$ are accumulated into an accumulator y and the witnesses $w_1, w_2, \ldots, w_n$; the witness w_i proves that M_i was accumulated into y.*

4.2.2 UFH Communication using Accumulators

Cryptographic accumulators [14] combine a (large) set of values U into one (short) accumulator y such that for each value $u_i \in U$ there is a (short) witness w_i that u_i was indeed incorporated into the accumulator (see Figure 4.5). A collision-resistant one-way accumulator is a collision-resistant hash function $f : X \times U \to X$ that is also quasi-commutative [11]. That means,

$$\forall x \in X : \forall u_1, u_2 \in U : f(f(x, u_1), u_2) = f(f(x, u_2), u_1).$$

In order to protect a message M, the sender computes for each fragment M_i generated by $\mathcal{E}(n, l, \varepsilon)$ the witness

$$\begin{aligned} w_i &:= f(id, \mathcal{M} \setminus \{M_i\}) \\ &= f(\ldots f(f(\ldots f(id, M_1) \ldots, M_{i-1}), M_{i+1}) \ldots, M_n), \end{aligned}$$

where id is the message id and $\mathcal{M}$ is the set of all fragments. Fragment M_i is encapsulated into the packet $m_i := id||i||l||M_i||w_i$ by adding the message id, the fragment number i, the number of required fragments l, and the witness w_i.

Time and Space Efficiency

For each new packet m_i, the receiver computes the accumulator

$$\begin{aligned} y &:= f(w_i, M_i) = f(f(id, \mathcal{M} \setminus \{M_i\}), M_i) \\ &= f(id, \mathcal{M}) = f(\ldots f(f(id, M_1), M_2) \ldots, M_n). \end{aligned}$$

Due to the quasi-commutative property of the one-way accumulator, the accumulator y is identical for all fragment/witness pairs of the same message and thus identifies the message the fragment belongs to. Verifying

a packet involves the computation of the accumulator and its comparison with the values of all N already received packets. The cost to verify m_i is thus in $\mathcal{O}(N)$.

The verification-related overhead per packet is given by the size of w_i. Fairly space-efficient one-way accumulators can be constructed based on modular exponentiation [14] or bilinear pairings [55]. For these accumulators, the size of the witness is equal to the prime order of the used group; that is, about $2k$ bits for a security level of k bits.

Another type of space-efficient one-way accumulators are Merkle trees [41]. Here, the hash values of $id||i||M_i$ are the leaves of the hash tree and the accumulator is given by the root value of the tree. The witness w_i for a fragment M_i is the set of the $\log_2(n)$ sibling nodes on the path from M_i to the root and is of size $\log_2(n)k$ bits.

Security Analysis

In order for the receiver to accept a phony inserted fragment M_i', the attacker must find a witness w_i' such that $f(w_i, M_i) = f(w_i', M_i')$. Given that $f(\cdot,\cdot)$ is a collision-resistant hash function, finding such a collision is considered infeasible for a computationally bounded attacker.

4.2.3 UFH Communication using Short Signatures

If the sender and receiver want to take full advantage of rateless erasure coding, the packet verification technique must allow for verifying each packet of a continuous packet stream individually. We propose a scheme based on short signatures that meets these requirements. In our scheme, the sender generates a new public/private key pair (K_M, K_M^{-1}) for every message M that he transmits. The length of these keys must be such that they resist an attack for the duration of the message transmission. Once the message transmission is over, the keys become useless for the attacker. After the keys have been created, the sender encapsulates each fragment M_i generated by $\mathcal{E}(n, l, \varepsilon)$ into a packet $m_i := K_M||i||l||M_i||Sig_{K_M^{-1}}(K_M||i||l||M_i)$ by adding the public key K_M, the fragment number i, the number l of required fragments, and the signature of $K_M||i||l||M_i$.

Time and Space Efficiency

The receiver uses the included public key to verify the signature of each received packet and drops packets with an invalid signature. Packets that are signed with the same private key belong to the same message. Verifying a new packet thus requires to verify the signature once and to compare

the included key with the keys of the N already received packets and is in $\mathcal{O}(N)$.

The verification-related overhead per packet consists of the public key and the signature. For the short signature scheme based on bilinear maps proposed by Boneh et al. [16], the size of the signature and of the public key is equal to the prime order of the used group. The total overhead per packet is thus $4k$ bits for a security level of k bits.

Security Analysis

To violate the integrity of this scheme, the attacker must find a new fragment and signature pair (M_i', σ') such that the signature σ' is accepted by the receiver as a valid signature for the data $K_M||i||l||M_i'$ and the public key K_M. For a secure signature scheme, finding such a pair is equal to generating valid signatures without the help of the private key and considered infeasible for a computationally bounded attacker.

4.3 Authenticated Packet-level Timestamping

A limitation of UFH that might require special treatment in the executed protocols is that messages are usually not received as a whole and that there is in general an unpredictable delay between the reception of the first packet/fragment and the final delivery of the reconstructed message. In a number of applications such as, for example, secure time synchronization or localization (see Chapter 7), the receiver has to obtain an authenticated timestamp that was written into the message right before its transmission. Thus, in order that UFH can be used for the jamming-resistant dissemination of such transmission timestamps, the timestamps must be added to the packets rather than the messages. Alternatively, to reduce the length of the packet timestamps, an absolute timestamp can be included into the message and shorter relative timestamps to the packets. The actual transmission time of a packet is then given by the addition of the absolute and the corresponding relative timestamp. A further overhead reduction can be achieved by adding a timestamp to a fraction (γ) of the packets only. In this case, the probability that the receiver obtained at least one packet with a timestamp during the reception of a message can be computed as $1-(1-\gamma)^L$, where L is the number of received packets.

A received timestamp is, however, useless if it cannot be authenticated. Since the packet verification by itself provides only message integrity, the packet verification must additionally be linked to the message which has to be authenticated (e.g., with a signature). Once the message has

been reconstructed and verified, the link ensures that the packets are also authentic (i.e., originate from the same source). A drawback of this procedure is that an authenticated timestamp is only available after some delay and not right after the reception of the first fragment; however, this additional time is not entirely wasted and has the advantage that, in expectation, not only one but γL timestamps are received per message, which reduces potential measurement errors and increases the confidence in the obtained value.

The required linking of the packet verification to the transferred message is generally a non-trivial process that might even be impossible for some verification schemes. Of the introduced schemes, the linking is simplest for the scheme based on short signatures. Here, the sender simply has to include the public key that is used for the packet verification into the (authenticated) message. If the message is authentic, so is this public key and consequently also all valid packets. Furthermore, any packet can be augmented with a fresh timestamp as each packet is signed individually.

For the verification scheme based on erasure coding and one-way accumulators the linking is slightly more complicated and only possible with some limitations. The reasons are that the witnesses for the n generated packets must be included into the (authenticated) message and thus be computed before the transmission of the packets and that the sender cannot alter the packets once they have been created. Fortunately, the packet transmission (i.e., hopping) schedule is fixed so that the first transmission time—and thus also the timestamp—of each packet can be computed beforehand and be included into the generated packets. In order to obtain a valid timestamp, the receiver must therefore receive at least one packet when it is transmitted for the first time. Assuming that the receiver is already receiving when the message transmission starts, the probability that the receiver obtains at least one authenticated timestamp is $1-(1-p_m\gamma)^n$, the expected number of received timestamps $p_m\gamma n$.

In principle, a similar approach as for the one-way accumulators could be used for the scheme based on hash links. However, because of the small number of packets per message, chances are rather low that a packet is received at its first transmission, which precludes the application of the hash link based schemes in combination with packet timestamps in practice.

Chapter 5
Attacker Strategies and Performance

In this chapter, we investigate the performance of the attacker. We first show that jamming is the best attacker strategy and develop an analytical model for different jammer types. Based on this model, we then express the attacker's strength as the number of channels that she can block during the transmission of a packet. Finally, we present a channel selection strategy for UFH that is optimal with respect to the achieved throughput and that exhibits the same jamming resistance as coordinated frequency hopping.

5.1 Attacker Strategies

As described in Section 3.2, the attacker can use her available transmission power to either insert, modify, or jam packets. We will argue in the following that, due to the fragmentation and packet verification (Section 4.2), the most effective attacker strategy is to use all power for jamming.

Packet Insertion

Maliciously inserted packets are filtered out by the receiver's packet verification or by the application protocol. Their processing is therefore not more expensive (i.e., does not demand more computation or storage) than that of (faulty) legitimate packets. Moreover, by definition of P_j, the minimal required signal strength at the receiver to jam a packet (Section 3.2), malicious packets whose signal strength at the receiver is less than P_j do not interfere with regular packets and thus do not have any impact on their transmission; sending with power greater than P_j, on the other hand, requires at least as much energy as jamming.

Packet Modification

Partial packet modifications such that the packet is still accepted by the receiver are considered infeasible due to the security properties of the packet verification techniques; partially modifying packets is thus an (expensive) form of jamming. Consequently, packets must be entirely replaced (i.e., overwritten) by the attacker with valid alternatives in order to be accepted. The minimal signal strength at the receiver to overwrite a packet is P_o and strictly larger than P_j (Section 3.2). Hence, the number of channels on which the attacker can overwrite packets is $\lfloor \frac{P_J}{P_o} \rfloor < \lfloor \frac{P_J}{P_j} \rfloor$, where P_J is the overall transmission power of the attacker.

Due to the fact that non-interfering packet insertions do not affect the processing of legitimate packets and because jamming a packet requires strictly less power and energy than overwriting a packet and not more power and energy than is required for interfering packet insertions, we conclude that jamming is the most effective and energy-efficient strategy for an attacker whose goal is to prevent communication.

5.2 Impact of the Packet Coding and Length

During their transmission over the wireless channel, the packets resulting from the message encoding (Section 4.2) are protected against jamming attacks and bit errors. A packet coding scheme is parametrized by its jamming resistance ρ, $0 < \rho \leq 1$, and its code rate r_c, $0 < r_c \leq 1$, if it encodes data of length $|m|$ into a packet of length $|m|/r_c$ and if more than $\rho|m|/r_c$ packet bits have to be disrupted so that the receiver cannot correctly decode the packet. We say that a coding scheme is *efficient* if its encoding and decoding times are polynomial in the length of the packet and if the packet length resulting from the encoding remains linear in the data length. In UFH, the length of the packets resulting from the packet encoding must not exceed the size $s = R/f_A$ of the hopping slots (typically in the order of few hundred bits), where R is the data rate and f_A is the hopping frequency of the sender A. In general, it holds that the shorter the slots (i.e., the shorter the packets), the better is the protection against reactive jamming, but the more packets need to be successfully transmitted. On the other hand, the longer the slots, the more redundancy can be added to the packets (allowing to choose codes with smaller r_c and larger ρ) and hence the better is the protection against non-reactive jamming. For a specific ρ and r_c, the transmission time of a packet is $t_m = |m|/r_c/R$ and the minimal duration during which it must be jammed is $\rho\, t_m$.

5.3 Attacker's Jamming Strength

We express the attacker's jamming strength, with respect to a given packet transmission, by the number of frequency channels c_b that she can effectively block during this transmission. Besides being a fairly intuitive and easy to comprehend metric to express the strength of a communication jammer, c_b further allows us to abstract away from the technical realizations of existing jammer types and provides us with a basis for comparing their impact on a given packet transmission and for arguing about optimal defense strategies. We next show how c_b can be computed for the jammer types introduced in Chapter 3.

Recall from Section 3.2 that reactive and hybrid jammers actively search for ongoing transmissions, whereas non-reactive jammers proactively (but blindly) jam the channels in use. Of the presented proactive jammers, static jammers jam the same channels for much longer than the transmission time of a packet, sweep jammers systematically update the jammed channels in a way that after a minimal number of jamming cycles all channels have been jammed once (but they do not have to follow a particular order), and random jammers always choose the channels to jam at random (and might thus jam the same channels several times before having hit all channels).

During the transmission of a packet, the attacker can sense the channels at most $n_s := \frac{t_m - \rho t_m - t_j}{t_s}$ times so that the transmission is detected early enough to still jam the packet for the minimal required time. Let c_s (c_j) be the number of channels on which the attacker can sense (jam) simultaneously and t_s (t_j) be the minimal time that the attacker requires to switch these channels. The maximal number of jamming cycles per packet that the attacker can achieve is $n_j := \frac{t_m}{\rho t_m + t_j}$. Hence, the (expected) number of channels that the attacker can block during the transmission of m is

$$c_b = \begin{cases} c_j & \text{for (proactive) static jammers,} \\ n_j c_j & \text{for (proactive) sweep jammers,} \\ c\left(1 - \left(1 - \frac{c_j}{c}\right)^{n_j}\right) & \text{for (proactive) random jammers,} \\ n_s c_s & \text{for reactive jammers,} \\ c_j + n_s c_s & \text{for reactive-static jammers,} \\ n_j c_j + n_s c_s & \text{for reactive-sweep jammers, and} \\ c\left(1 - \left(1 - \frac{c_j}{c}\right)^{n_j}\right) + n_s c_s & \text{for reactive-random jammers.} \end{cases} \tag{5.1}$$

Lemma 1. *Of the introduced attacker types, reactive-sweep jammers are the most powerful ones.*

Proof. Hybrid jammers are at least as powerful as their non-reactive counterparts as the latter are just a special case of the former. Also, of the hybrid jammers, the reactive-sweep is able to jam the most channels because $c(1-(1-\frac{c_j}{c})^{n_j}) = c(1-\sum_{i=0}^{\infty}\binom{n_j}{i}(\frac{-c_j}{c})^i) \leq c(1-(1-n_j\frac{c_j}{c})) = n_j c_j$. □

5.4 Optimal Channel Selection

In UFH, the frequency channels on which the sender and receiver send and receive are chosen randomly from the set of available channels. In contrast

to coordinated FH, where the jamming resistance and thus the throughput of the communication increases with the number of channels used, using all available channels might not be optimal for UFH: while the jammer's chances to jam the right channel decrease the more channels are used, so do an uncoordinated receiver's chances to listen on the right channel. As an example, consider the case where the sender and receiver can send and receive on one channel (i.e., $c_t = c_r = 1$) and select the frequency channels uniformly at random from a set of c channels. Assuming further that the attacker blocks c_b channels, the probability that a packet is successfully received with UFH is $p_m = \frac{1}{c}(1 - \frac{c_b}{c})$, which is maximized for $c = 2c_b$. We next show that a similar result holds for the general case where the sender and receiver send and receive on more than one channel:

Theorem 1. *In the presence of an attacker that prevents communication on $c_b \leq c$ channels, the probability p_m that a packet is successfully received with UFH is maximized if the sender and receiver choose the c_t (c_r) frequency channels on which they send (receive) uniformly at random from a set of size c^*, where*

$$c^* = \begin{cases} c_b + 1 & \text{if } c_b < c_r \text{ and } c_b < c_t \\ \max\{\approx 2c_b, c_r, c_t\} & \text{otherwise.} \end{cases}$$

To prove Theorem 1, we first introduce three lemmas:

Lemma 2. *If $a_1, \ldots, a_c \in [0,1]$ such that $\sum_{i=1}^{c} a_i \leq c_t$, then $\sum_{i=1}^{c} \frac{1}{a_i} \geq \frac{c^2}{c_t}$.*

Proof. Consider the values $a_i := \frac{c_t}{c} + \varepsilon_i$ and $a_j := \frac{c_t}{c} - \varepsilon_i + \varepsilon_j$. The sum $y := \frac{1}{a_i} + \frac{1}{a_j}$ is minimized if $\frac{dy}{d\varepsilon_i} = -(\frac{c_t}{c} + \varepsilon_i)^{-2} + (\frac{c_t}{c} - \varepsilon_i + \varepsilon_j)^{-2} = 0$; that is, if $\varepsilon_i = \frac{1}{2}\varepsilon_j$ and thus $a_i = a_j$. Since this holds for all pairs a_i and a_j, $\sum_{i=1}^{c} \frac{1}{a_i}$ is minimized if $\forall i, j : a_i = a_j = \frac{c_t}{c}$. □

Lemma 3. *Let c_t (c_r) be the number of channels on which the sender (receiver) sends (receives) in parallel and c_b be the number of channels that the attacker jams. If the sender and receiver select their channels uniformly at random from a set of c channels, the probability that the sender and receiver select at least one common non-jammed channel is*

$$p_m = 1 - \sum_{i=0}^{c_r} \frac{\binom{c-c_t}{c_r-i}\binom{c_t}{i}}{\binom{c}{c_r}} \frac{\binom{c-i}{c_b-i}}{\binom{c}{c_b}}. \tag{5.2}$$

Proof. Let X_i denote the event that there is a transmission on exactly i out of the c_r channels chosen by the receiver. There exist $\binom{c-c_t}{c_r-i}$ possibilities to chose the $c_r - i$ channels without a transmission and $\binom{c_t}{i}$ possibilities to chose the i channels with a transmission. Hence, $\mathsf{P}[X_i] = \binom{c-c_t}{c_r-i}\binom{c_t}{i}/\binom{c}{c_r}$. The attacker, in turn, blocks communication on c_b out of c channels. Given a set of i channels, the probability that the attacker blocks them all is $\mathsf{P}[\text{the } i \text{ channels are jammed}] = \binom{c-i}{c_b-i}/\binom{c}{c_b}$. The proof follows from the observation that $p_m = 1 - \sum_{i=0}^{c_r} \mathsf{P}[X_i]\mathsf{P}[\text{the } i \text{ channels are jammed}]$. □

Lemma 4. *For $u \geq v \geq 0$: $\binom{u}{v} \leq \frac{u^v}{v!}$.*

Proof. By definition we have $\binom{u}{v} = \frac{u(u-1)(u-2)\cdots(u-v+1)}{v!} = (1-\frac{1}{u})(1-\frac{2}{u})\cdots(1-\frac{v-1}{u})\frac{u^v}{v!} \leq \frac{u^v}{v!}$. □

Proof of Theorem 1. First we show that selecting the input and output channels uniformly at random is an optimal strategy for the UFH sender and receiver. We will show that by jamming the channels that are more likely to be selected by the sender more intensively than the rarely selected channels, the attacker can nullify any advantage that the sender and receiver might obtain by using a non-uniform channel selection. Let a_i (b_i) be the probability that the sender (receiver) sends (receives) on channel $i \in \{1, 2, \ldots, c\}$. Let further x_i be the probability that the attacker blocks channel i. Without loss of generality we assume that $a_1 \geq a_2 \geq \ldots \geq a_{c'} > 0$ and $a_{c'+1} = a_{c'+2} = \ldots = a_c = 0$. Now consider the jammer strategy where the attacker jams channel i with probability $x_i = 1 - \frac{c_t(c'-c_b)}{a_i c'^2}$ if $a_i \geq \frac{c_t(c'-c_b)}{c'^2}$ and $x_i = 0$ otherwise. With this strategy, the probability that a packet is not jammed and thus can be successfully received on channel i is equal to $a_i(1-x_i) \leq a_i \frac{c_t(c'-c_b)}{a_i c'^2} = \frac{c_t}{c'}(1-\frac{c_b}{c'})$. This attacker strategy is a valid strategy since $\sum_{i=1}^{c'} a_i \leq c_t$ and $\sum_{i=1}^{c'} \frac{1}{a_i} \geq \frac{c'^2}{c_t}$ (Lemma 2) and thus $\sum_{i=1}^{c} x_i \leq \sum_{i=1}^{c'}(1 - \frac{c_t(c'-c_b)}{a_i c'^2}) = c' - \frac{c_t(c'-c_b)}{c'^2}\sum_{i=1}^{c'}\frac{1}{a_i} \leq c' - \frac{c_t(c'-c_b)}{c'^2}\frac{c'^2}{c_t} = c_b$.

Hence, the receiver's chances to successfully receive a packet are the same for the first c' channels and zero for the remaining $c - c'$ channels. It follows that in this case every selection strategy—and thus also selecting the channels uniformly at random—for which $\sum_{i=1}^{c'} b_i = c_r$ is an optimal receiver strategy. We note that jamming channel i with probability $x_i = 1 - \frac{c_t(c-c_b)}{a_i c^2}$ might not be the optimal strategy for the attacker with respect to a particular receiver (e.g., if she knew that the receiver would listen on

one channel only, focusing the jamming on this channel would be optimal). This strategy is, however, optimal for the attacker in the sense that it limits the performance of the sender for any number of receivers with arbitrary channel selection schemes.

Having shown that choosing channels uniformly at random is an optimal strategy for the sender and receiver, we next deduce the optimal set size c^* from which the sender and receiver should uniformly choose their channels. We first consider the special case where the attacker is weaker than the sender and the receiver: If $c_b < c_t$ and $c_b < c_r$, for all $c \in \{c_b + 1, c_b + 2, \ldots, \min\{c_r, c_t\}\}$ there exists at least one non-jammed channel on which the sender is sending and the receiver is receiving. In particular we have $p_m = 1$ for $c^* = c_b + 1$. Otherwise, if either $c_r \leq c_b$ or $c_t \leq c_b$, a second bound on c^* is given by $c^* \geq \max\{c_t, c_r\}$. If $c = c_t \geq c_r$, the receiver always listens on a channel on which a packet is transmitted. Using less than c_t channels would therefore increase the attacker's chances to successfully jam all transmissions without being beneficial for the receiver. Likewise, if $c = c_r \geq c_t$, the receiver receives all transmissions. Using less than c_r channels would again increase the attacker's jamming performance without any benefit for the receiver. Finally, we consider the general case. According to Lemmas 3 and 4, the probability that a packet sent by the sender is successfully received by the receiver is

$$\begin{aligned} p_m &= 1 - \sum_{i=0}^{c_r} \frac{\binom{c-c_t}{c_r-i}\binom{c_t}{i}}{\binom{c}{c_r}} \frac{\binom{c-i}{c_b-i}}{\binom{c}{c_b}} \\ &\gtrapprox 1 - \sum_{i=0}^{c_r} \frac{\frac{(c-c_t)^{c_r-i}}{(c_r-i)!}\frac{c_t^i}{i!}}{\frac{c^{c_r}}{c_r!}} \left(\frac{c_b}{c}\right)^i \\ &= 1 - \left(1 - \frac{c_t}{c}\left(1 - \frac{c_b}{c}\right)\right)^{c_r} =: \hat{p}_m \,. \end{aligned} \tag{5.3}$$

For a given c_t and c_r, $\hat{p}_m$ is maximized if $\frac{1}{c}(1 - \frac{c_b}{c})$ is maximized. We obtain $\frac{d}{dc}\frac{1}{c}(1 - \frac{c_b}{c}) = 0$ if $c = 2b$ and thus $c^* \approx 2b$. For particular values of c_r, c_t and c_b, a more precise result can be obtained with some standard (numerical) optimization technique. However, our simulations showed that the approximation $c^* = 2b$ already leads to very accurate results in practice: for $c_b \leq 100$ the relative error in p_m was always less than 3%. □

Corollary 1. *An UFH communication scheme in which the transmission channels are selected uniformly at random from a set of c^* channels is optimal*

with respect to the achieved throughput and exhibits the same resistance to jamming as coordinated frequency hopping (FH).

Corollary 1 follows from Theorem 1 and the observation that from the attacker's perception, all she can observe are transmissions of corresponding length on arbitrary channels. More precisely, the entropy of the next frequency hop for FH and UFH (for one transmission channel) is $H_{FH} = H_{UFH} = -\sum_{k=1}^{c} \mathsf{P}(c_k) \log_2 \mathsf{P}(c_k) = -\sum_{k=1}^{c} \frac{1}{c} \log_2 \frac{1}{c} = \log_2 c$, where c_k denotes the k-th channel. In other words, any communication jammer that can prevent communication in UFH can also prevent communication of coordinated FH in the same setting, and vice versa.

5.5 Adaptive Channel Selection

Achieving an optimal throughput with UFH requires the sender and receivers to accurately assess c_b and agree on a set of $c = c^*$ frequency channels. Especially in the absence of jamming, any selection of $c \geq 1$ channels will lead to a suboptimal performance. An optimal adaptive scheme in which both the sender and receiver(s) adapt their channels depending on the encountered jamming is, however, not practical. As jamming occurs at the receiver of a transmission, the sender would have to reliably obtain the feedback of the (maybe unknown) receiver(s); since different receivers are likely to observe different jamming strengths, this would further require to adapt to the worst case receiver. To avoid that the attacker can exploit this feedback, the feedback channel would have to be authentic. Providing the sender with the required feedback is therefore the same problem as the one we intend to solve with UFH in the first place.

Consequently, only those adaptive schemes are feasible where the receivers adapt their behavior but not the sender. However, as shown in the proof of Theorem 1, the attacker can nullify any possible advantage of an adaptive scheme that leverages a non-uniform channel selection by jamming more often those channels that are selected by the sender more likely than the rarely selected channels. This result is independent of the receiver's behavior and even holds for multiple senders: if we neglect the impact of the senders' position and of the physical environment, from the attacker's point of view there is no difference between the cases of multiple senders and a single sender that sends on multiple channels in parallel.

In conclusion, the best strategy for the sender is to assess the jamming strength c_b of the expected attacker as accurately as possible and then select all output channels uniformly at random from a set of size $\max\{c^*, c\}$,

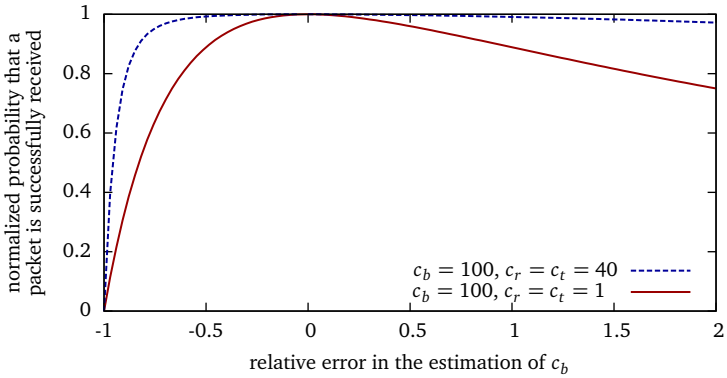

Figure 5.1: *Graceful degradation of UFH performance for an imprecise assessment of the attacker's strength (in terms of c_b). We observe that overestimating the attacker's strength is less harmful than underestimating it. We also observe that the larger c_r and c_t are, the lower is the impact of an imprecise estimation.*

where c is the maximal number of available channels and c^* is the optimal number of channels as specified in Theorem 1. The only exception might be a situation where the attacker (although being strong) is absent (or inactive) most of the time. In such a situation it is advisable to send permanently on one channel and select the other output channels uniformly at random from the remaining channels. The receivers first try to receive the message on the first channel and only switch to UFH if they are not successful. Permanently sending on the first channel ensures that the scheme is as efficient as coordinated frequency hopping if the attacker is absent.

In practice, assessing the attacker's strength means estimating the scanning and sending performance of her jamming device. Given these parameters, the number of channels c_b that the attacker can block can be derived according to Equation (5.1). How accurately the sender can estimate the capabilities of the jamming device depends on the considered attacker model. The impact of an imprecise assessment of the attacker's strength is shown in Figure 5.1. We observe that the performance of UFH gracefully degrades the less accurate the assessment of c_b is and that overestimating the attacker's strength is less harmful that underestimating it. Hence, using all available channels is a reasonable fallback strategy under the threat of a completely unknown attacker.

Chapter 6
Performance Analysis and Evaluation

In this chapter, we evaluate the efficiency of the presented UFH communication schemes (see Chapter 4) in terms of their throughput. As the main metric for their performance, we use the expected number of required packet reception attempts until the receiver can successfully decode the message. Throughout this analysis, we assume that all erasure codes are optimal (i.e., $\varepsilon = 0$) and we neglect differences in the verification and decoding speed of the proposed schemes. This is reasonable since the message reassembly is performed only once per message due to the packet verification and takes significantly less time (in the order of milliseconds [66]) than the time required for the message packets to be acquired at the receiver (order of seconds). The packet verification time of our schemes is also in the order of milliseconds[1] and the verification can be performed in parallel to the packet gathering. We further assume that the hopping frequency of the receiver f_B is much slower than the hopping frequency of the sender f_A (Figure 2.2). We therefore neglect packet losses caused by the lack of synchronization between sender and receiver as they only affect every $\frac{f_A}{f_B}$-th packet and are thus rare events compared to the likelihood that the receiver listens on an incorrect channel (i.e., $\frac{f_B}{f_A} \ll (1 - \frac{1}{c})$).

6.1 Analysis of UFH Schemes based on Rateless Erasure Codes

We first consider the case of a rateless erasure code $\mathcal{E}(\infty, l, \varepsilon)$ in combination with short signatures. This combination is optimal in the sense that there are no duplicate or non-verifiable packets and every successfully received fragment contributes to the message. Hence, a message can be reassembled as soon as l fragments have been received. Let p_m be the probability that a packet sent by the sender is successfully received by a specific receiver (Equation (5.2)). Let further X_i denote the event that

[1] Of the used cryptographic primitives, signatures are the most expensive ones (compared to computing hash values or to verifying accumulators). Verifying a single 160 bit short signature takes about 48 ms on a current general purpose CPU and becomes faster if several packets are verified in a single batch verification (<3 ms per signature for a batch of 200 packets) [30].

the last of the required l packets is received after i (successful and unsuccessful) packet reception attempts. The expected number of required packet reception attempts until a message can be reconstructed by a single receiver is then $N(p_m) = \sum_{i=0}^{\infty} \mathsf{P}[X_i]i$ and thus

$$N(p_m) = \sum_{i=l}^{\infty} \binom{i-1}{l-1} (p_m)^l (1-p_m)^{i-l} i = \frac{l}{p_m} \in \mathcal{O}(l) . \tag{6.1}$$

Note that Equation (6.1) also applies to erasure codes where the number of fragments n is finite but larger than the expected number of required reception attempts $N(p_m)$.

In the general case where a message is broadcasted to a group of $g \geq 1$ receivers, the message transmission is completed once all g receivers have successfully received and reconstructed the message. Let receiver B be the last group member to receive the message and let Y_i be the event that all g receivers have received the message after i (successful and unsuccessful) packet reception attempts of B. The expected number of required packet reception attempts until a message can be reconstructed by the last receiver is then $N_g(p_m) = \sum_{i=0}^{\infty}(1 - \mathsf{P}[Y_i])$, where $\mathsf{P}[Y_i] = (\sum_{j=l}^{i} \mathsf{P}[X_j])^g$; hence,

$$N_g(p_m) = \sum_{i=0}^{\infty} \left(1 - \left(\sum_{j=l}^{i} \binom{j-1}{l-1} (p_m)^l (1-p_m)^{j-l} \right)^g \right) . \tag{6.2}$$

6.2 Analysis of UFH Schemes based on Erasure Codes

If the number of fragments n per message is finite (and smaller than $N(p_m)$), the sender repeatedly sends the sequence of n packets. We observe that after i such sequence transmissions the expected number of missing packets is $n(1-p_m)^i$. For the case that any set of l fragments can be verified we get the approximation

$$N(p_m) \approx \frac{\log(n-l) - \log(n)}{\log(1-p_m)} n \in \mathcal{O}(\log(\tfrac{n}{n-l})n) \tag{6.3}$$

by solving $n(1-p_m)^i = n - l$ for i. More precisely, the probability that a specific receiver has received exactly j out of n fragments after i reception attempts is $\mathsf{P}[X_{ji}] = ((1-p_m)^i)^{n-j}(1-(1-p_m)^i)^j$. In the final round, the last packet will be received after an average of $n/2$ attempts. Let Z_j

denote the event that the set of j fragments is verifiable. For the erasure code $\mathcal{E}(n,l,\varepsilon)$ the expected number of required packet reception attempts is then

$$N(p_m) = \Big(\sum_{i=0}^{\infty}\sum_{j=0}^{n} \mathsf{P}[X_{ji}]\mathsf{P}[\neg Z_j] - 1\Big)n + \frac{n}{2} \tag{6.4}$$
$$= \sum_{i=0}^{\infty}\sum_{j=0}^{n} D(j)\left(\left(1-p_m\right)^i\right)^{n-j}\left(1-\left(1-p_m\right)^i\right)^j n - \frac{n}{2}\,,$$

where $D(j)$ is the number of *non-verifiable* subsets of size j. Clearly, $D(j) = \binom{n}{j}$ if $j < l$ since a message cannot be reconstructed with less than l fragments. Moreover, if an accumulator-based verification is used, then $D(j) = 0$ if $j \geq l$, as each packet can be verified individually, and thus any set of $j \geq l$ genuine fragments is verifiable.

In the case of multiple hash-link verification, a set of $j \geq l$ packets is verifiable if at least l packets in the set are connected without gaps of α or more missing packets between them. In order to compute $D(j)$ (i.e., the number of sets with j packets such that no l packets in the set are connected) we first introduce the following two lemmas:

Lemma 5. *The number of allocations of b indistinguishable balls into r distinguishable bins such that the first m, $0 \leq m \leq r$, bins contain at most k balls is*

$$A(b,r,m,k) = \sum_{i=0}^{\lfloor \frac{b}{k+1} \rfloor} (-1)^i \binom{m}{i} \binom{b-(k+1)i+r-1}{r-1}\,.$$

Proof. A proof can be found in [51]. □

Lemma 6. *The number of sequences of length n with j black and $n-j$ white balls that contain a sub-sequence of $\geq l$, $2l > n$, black and m white balls such that no two subsequent black balls in the sub-sequence are separated by more than b white balls is $A(n-j, j+1, l-1, b)+(j-l)A(n-j-b-1, j+1, l-1, b)$.*

Proof Sketch. We identify each sub-sequence fulfilling the given requirements by the first black ball in the sequence. If the sub-sequence starts at the first black ball in the sequence, according to Lemma 5, there exist $A(n-j, j+1, l-1, b)$ ways to distribute the $n-j$ white balls before, between, or after the j black balls such that there are no more than b white balls between the first l black balls. In the $j-l$ cases where the

wanted sub-sequence starts at the second to $(j-l+1)$-th black ball, the sub-sequence must be preceded by (at least) $b+1$ white balls. Thus, in each of these cases there exist $A(n-j-b-1, j+1, l-1, b)$ ways to distribute the remaining $n-j-b-1$ white balls. □

From Lemma 6 it follows that there exist $D(j) = \binom{n}{j} - A(n-j, j+1, l-1, \alpha-1) - (j-l)A(n-j-\alpha, j+1, l-1, \alpha-1)$ sets of j packets such that no l packets in the set are connected. For the single hash link scheme we have $\alpha = 1$ and $n = l$ and thus:

$$N(p_m) = \sum_{i=0}^{\infty} \left(1 - \left(1-(1-p_m)^i\right)^l\right) l - \frac{l}{2} \tag{6.5}$$
$$\approx \frac{\log(0.5) - \log(l)}{\log(1-p_m)} l \in \mathcal{O}(\log(l)l) \,.$$

For a broadcast transmission in which the message must be received by all g receivers we obtain

$$N_g(p_m) = \sum_{i=0}^{\infty} \left(1 - \left(1 - \sum_{j=0}^{n} \mathsf{P}[X_{ji}]\mathsf{P}[\neg Z_j]\right)^g\right) n - \frac{n}{2} \tag{6.6}$$
$$= \sum_{i=0}^{\infty} \left(1 - \left(1 - \sum_{j=0}^{n} D(j) \left((1-p_m)^i\right)^{n-j} \left(1-(1-p_m)^i\right)^j\right)^g\right) n - \frac{n}{2} \,.$$

6.3 Multiple Concurrent UFH Transmissions

So far, we considered the attacker as the only source of interference. If, however, several neighboring senders engage in different UFH transmissions at the same time, these concurrent transmissions might mutually interfere and increase the effective number of disturbed (i.e., blocked) channels. More precisely, if, in a cluster of g senders, all senders send simultaneously on c_t transmission channels, the expected number of occupied channels with respect to a single transmission under jamming is $c_g = \sum_{i=1}^{c} \mathsf{P}[\text{channel } i \text{ is jammed or used by one of the other transmissions}] = (1-(1-\frac{c_b}{c})(1-\frac{c_t}{c})^{g-1})c$. Hence, the probability that a packet is successfully received by the receiver reduces to

$$p_m = 1 - \sum_{i=0}^{c_r} \frac{\binom{c-c_t}{c_r-i}\binom{c_t}{i}}{\binom{c}{c_r}} \frac{\binom{c-i}{c_g-i}}{\binom{c}{c_g}} \,. \tag{6.7}$$

Apart from this mutual interference, there is no dependency among these transmissions as each receiver selects the reception channels randomly and independent of all other receivers. For evaluation purposes, we can thus consider the concurrent broadcast of g_s senders to g_r receivers as $g_s g_r$ independent transmissions which, in turn, corresponds to a single broadcast to $g_s g_r$ receivers. Hence, the number of required reception attempts until all g_r receivers have received all g_s broadcasts can be computed with Equation (6.2) and (6.6) and the substitution $g := g_s g_r$.

6.4 Performance Comparison and Discussion

We next compare the performance of the presented message coding schemes among each other and with an ideal scheme. In particular we compare the following schemes:

HL1 Simple splitting combined with single hash links (Section 4.2.1). In this case, only l packets are generated and all of them must be received.

HL2 Erasure coding combined with double hash links (Section 4.2.1). Here, a sequence of n packets is generated and any subset of l packets can be used to reconstruct the message. However, in order that the subset can be verified, the packets must form a sequence with at most one missing packet between any two subsequent packets.

HL3 Erasure coding combined with triple hash links (Section 4.2.1). This scheme is similar to HL2, but gaps of up to two packets in the received packet sequence are tolerated.

Acc Erasure coding combined with the one-way authenticator based on modular exponentiation (Section 4.2.2). With this scheme, a sequence of n packets is generated of which each packet can be verified individually. Thus, any subset of l packets can be used to reconstruct the message. Since this accumulator introduces strictly less overhead per packet than a Merkle tree based accumulator, we do not consider the latter in this comparison.

Sig Rateless erasure coding combined with short signatures (Section 4.2.3). Here, an infinite sequence of packets is generated and each packet can be verified individually. Hence, the message can be reconstructed by any set of l distinct packets.

Ideal An ideal scheme that uses rateless erasure coding but has no verification overhead. As for Sig, any set of l distinct packet can be used to reconstruct the message. This scheme constitutes an upper bound on the performance.

In this evaluation, the size of the message ids and the fragment numbers is 48 and 16 bit, respectively. We choose $n = 500$ fragments for Acc and derive the optimal n for the hash-link schemes numerically. The number of required packets is $l = |M|/(|m| - o)$, where o is the overhead per packet (message id, fragment number, verification data). We further choose a (short-term) security level of $k = 70$ bits, which ensures that the cryptographic primitives (hashes, accumulators, signatures) can resist an attack for a period of several weeks to months [3]. We point out that this is more than sufficient since the primitives must resist an attack only for the duration of the message transmission, which is in the order of seconds.

The performance of the considered schemes as a function of the message and packet size is depicted in Figure 6.1. We observe that overall Acc and Sig perform best. The small overhead of HL1 makes it a reasonable choice for small packet sizes whereas the advantage of the second hash link for HL2 is more or less nullified by the larger overhead per packet (i.e., the larger number l of required packets). This is even worse for HL3 which is always less efficient than the basic scheme. We also observe that, if p_m is held constant, the difference between the proposed schemes and the ideal scheme decreases for larger packet sizes (i.e., a smaller number l of required packets per message) and increases for larger message sizes (i.e., a larger number l of required packets per message).

The expected throughput and message transmission times as a function of the message size, the slot size, and the number of receivers are shown in Figure 6.2 and 6.3. The simulation results presented in this figure were obtained with a purpose-built Java application that simulates the message transmissions and uses a simplified communication model for the physical layer. The application assumes a perfect jammer that jams a packet for the minimal required amount of time and whose interference with a packet is always destructive. Hence, a packet is successfully received if the sender and receiver are sending and receiving on at least one unjammed channel, else the packet is dropped. The results show that the transmission time increases linearly with the message size $|M|$ but only logarithmically with the number of receivers g. We further observe from Figure 6.3 that the throughput initially increases for larger slot lengths as less fragments (and thus less reception attempts) per message are needed. Since the

jamming probability also increases for larger slots, the throughput starts decreasing once the advantage of less fragments is outweighed by the increased jamming probability.

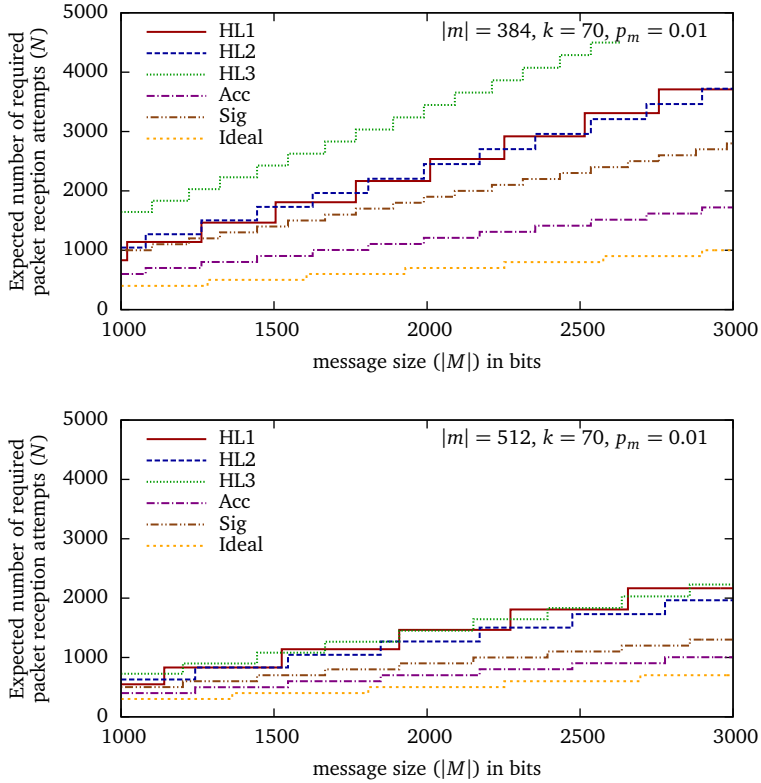

Figure 6.1: *Performance of the presented message coding schemes as a function of message and packet size. We compare the schemes to an ideal scheme without verification overhead. We observe that overall Acc and Sig perform best. HL1 is a reasonable choice for small packet sizes, whereas the advantage provided by two hashes in HL2 is more or less nullified by the larger overhead per packet. This is even worse for HL3, which is always less efficient than HL1. By comparing the plots, we observe that, if p_m is held constant, the difference between the proposed schemes and the ideal scheme decreases for packet sizes and increases for larger message sizes.*

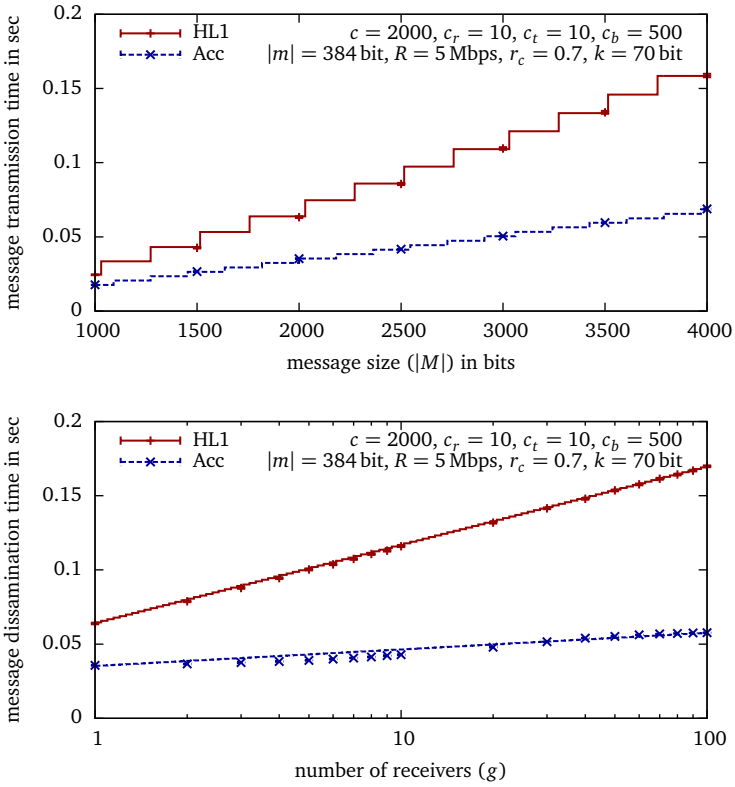

Figure 6.2: *Expected message transmission time as a function of the message and group size. In this plot, the lines show the analytical results, the points and confidence intervals display the findings of our simulations. We observe that the transmission time increases linearly with the message size but only logarithmically with the number of receivers.*

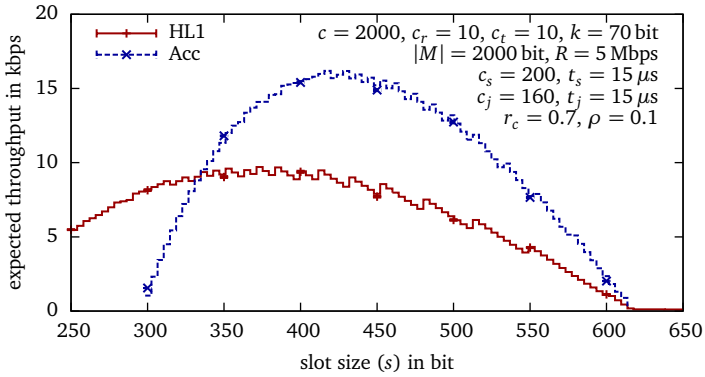

Figure 6.3: *Expected throughput as a function of the slot size. In this plot, the lines show the analytical results, the points and confidence intervals display the findings of our simulations. We observe that the throughput initially increases for larger slot lengths as less fragments (and thus also less reception attempts) per message are needed. Since the jamming probability also increases for larger slots, the throughput finally starts decreasing once the advantage of less fragments is outweighed by the increased jamming probability.*

Chapter 7
Applications of UFH

In this chapter, we first show how UFH enables the bootstrapping of coordinated frequency hopping (FH) by means of a jamming-resistant key establishment. We then present two additional applications that strongly benefit from UFH anti-jamming broadcast communication: emergency alert broadcast and navigation broadcast. Common to these applications is that a sender has to broadcast a message to a dynamic set of unknown and potentially malicious receivers that may want to deprive other receivers from obtaining the information. With coordinated FH, if a sender wants to broadcast a message to a set of receivers, it needs to share a secret code with all receivers and the code needs to be hidden from the attacker. Using coordinated FH in the considered setting is therefore infeasible if the code is held secret, or can easily be disrupted by malicious users if the code is public.

7.1 Bootstrapping Coordinated FH with UFH-based Key Establishment

The bootstrapping of coordinated frequency hopping can be divided into two stages. In the first stage, the parties execute a key-establishment protocol and agree on a shared secret key K using UFH. Various key-establishment protocols can be used in this step and we present the authenticated Diffie-Hellman protocol [7] and the Burmester-Desmedt protocol [17] as typical examples for two-party and group key agreement, respectively. Then, in the second stage, each party transforms the key K into a hopping sequence (using linear feedback shift registers and channel mappers [60]) and switches to coordinated frequency hopping. The first message in the second stage is typically a key confirmation that verifies the successful key agreement and, additionally, is used to synchronize the frequency hopping between the parties. Note that the established key is not used for encrypting or signing sensitive data but exclusively for generating the hopping sequence.

Since our UFH communication schemes do not provide message authentication, all messages that are exchanged during the key establishment are signed in order to prevent the insertion of fake messages. In addition, the protocols use timestamps to preclude replay attacks and a (short-term)

history buffer to detect and drop duplicate messages during the validity of the timestamps. The period during which a message is considered valid is defined by the receiver and is usually in the order of time that is required to successfully transmit the message using UFH. Messages can be received more than once during their validity, either due to replay attacks or due to the repetitive message transmissions which are inherent to our UFH communication schemes. We point out that although an attacker may be able to replay an overheard message within the acceptable time interval in another protocol session, this does still not enable her to deduce the secret hopping sequence from it as the key contribution of the legitimate devices remains secret.

In what follows, let $\mathbb{G}$ be an additive cyclic group of prime order p in which the Decision Diffie-Hellman (DDH) problem is hard and let P be a generator of this group. Because we are more concerned about minimizing the message sizes than the computational overhead, we assume that $\mathbb{G}$ is an elliptic curve group. Let further $\langle\cdot\rangle_X$ be the string in angle brackets concatenated with its signature by party X and let $\{\cdot\}_K$ be the encryption of the string in curly brackets with key K.

7.1.1 Two-party Key Agreement

As an example for a two-party key agreement, we consider the Elliptic Curve Cryptography (ECC) based Station-to-Station (STS) Diffie-Hellman protocol [7]. The STS protocol proceeds in two rounds and is executed as follows (see Figure 7.1): First, party A selects a (pseudo-)random element $r_A \leftarrow_R \mathbb{Z}_p$ and broadcasts a signed message containing its public key certificate σ_A, a timestamp T_A, and the credential $z_A = r_A P$. A party B in the transmission range of A replies with a symmetric message containing its credential $z_B = r_B P$ and A's timestamp T_A. Based on the received messages, both parties then compute the shared key $K = r_A z_B = r_B z_A = r_A r_B P$.

In the second stage, A switches to coordinated frequency hopping and proves its knowledge of K by sending $\{\langle h(r_A P \parallel r_B P)\rangle_A\}_K$ to B. Party B uses this message to synchronize its hopping sequence. Note that the above protocol contains the explicit exchange of the parties' public key certificates; this overhead can be omitted in the case where the parties exchanged or preloaded each other's certificates prior to the protocol execution.

7.1.2 Group Key Agreement

Known Group Key Agreement (GKA) protocols for arbitrary group sizes proceed in several sequential rounds and therefore require some sort of (implicit) synchronization. This synchronization constitutes a major challenge

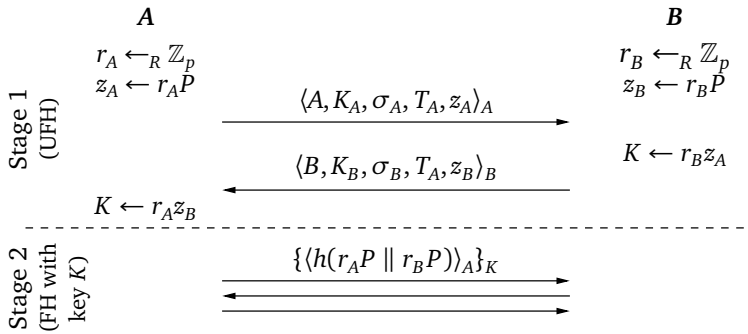

Figure 7.1: *Two-party UFH-based key establishment using an authenticated DH protocol. In stage 1, party A uses UFH to broadcast its certified public key K_A and its key contribution $r_A P$ for the elliptic curve DH protocol. A party B in A's power range answers by sending its own DH-contribution. In stage 2, A transmits a key acknowledgment, then the devices can send arbitrary messages using coordinated frequency hopping.*

for UFH-based group key agreement protocols as common synchronization techniques such as the (explicit) acknowledgment of all unicast and broadcast messages or the exchange of coordination messages should be avoided due to the comparatively high latency of UFH communication. Roughly speaking, the less rounds a GKA protocol comprises and the more message exchanges can be performed in parallel, the better its performance in combination with UFH communication.

In this work, we use an authenticated version of the Burmester-Desmedt (BD) group key agreement protocol [17] as an example for a suitable GKA protocol. What makes the BD GKA protocol particularly suitable for the use with UFH is the fact that it proceeds in only two rounds and that it allows all parties to concurrently send their contributions within a round (see Figure 7.2). Moreover, the protocol does not require all parties to switch simultaneously from the first to the second round. All that must be ensured is that a party stops sending the first contribution only after all intended receivers have received it. To avoid the overhead of an explicit synchronization (e.g., by means of acknowledgment messages), we propose to include the first contribution in the message that is sent in the second round. This simple solution avoids the overhead of additional messages at the cost of a longer second message.

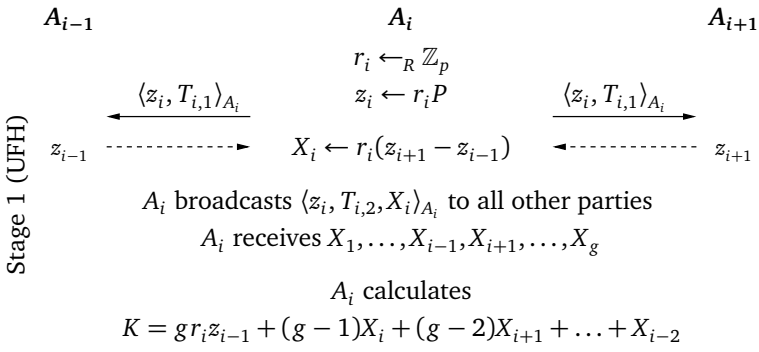

Figure 7.2: *UFH-based Burmester-Desmedt group key agreement protocol, depicted from device A_i's perspective. In the first round, A_i transmits the message $\langle z_i, T_{1,i} \rangle_{A_i}$ which needs to be received by its two logical neighbors A_{i-1} and A_{i+1}. Once it has received the two key contributions of its neighbors, A_i starts the second round and broadcasts the message $\langle z_i, T_{i,2}, X_i \rangle_{A_i}$. After having received all contributions X_i, A_i computes the group key $K = g r_i z_{i-1} + (g-1)X_i + \ldots + X_{i-2}$, where g is the group size.*

Let $A_1, A_2, \ldots, A_g$ be the set of parties that participate in the GKA. For simplicity, we assume that the set is known to each party and that the indices form a logical ring from 1 to g (i.e., A_{g+1} is A_1 and A_0 is A_g). The BD GKA protocol proceeds as follows: In the first round, each party A_i selects a (pseudo-)random element $r_i \leftarrow_R \mathbb{Z}_p$ and broadcasts a signed message containing a timestamp $T_{i,1}$ and the credential $z_i = r_i P$. In the second round, each A_i broadcasts a second signed message containing a timestamp $T_{i,2}$, the credential z_i, and the value $X_i = r_i(z_{i+1} - z_{i-1}) = (r_{i+1} - r_{i-1}) r_i P$. Given the values $X_1, X_2, \ldots, X_g$, each participant finally computes the shared key $K = g r_i z_{i-1} + (g-1)X_i + (g-2)X_{i+1} + \ldots + X_{i-2} = (r_1 r_2 + r_2 r_3 + \ldots + r_g r_1)P$. In the second stage, the parties switch to coordinated frequency hopping and use A_1's broadcast of $\{\langle h(X_1 \,\|\, X_2 \,\|\, \ldots \,\|\, X_g) \rangle_{A_1}\}_K$ to synchronize their hopping sequences.

7.2 Anti-jamming Emergency Alerts

Two typical examples where a jamming-resistant dissemination of emergency alerts is required are (1) if a central (governmental) authority needs to inform the public about the threat of an imminent or ongoing (terrorist) attack while minimizing the risk that the attackers can jam the alert trans-

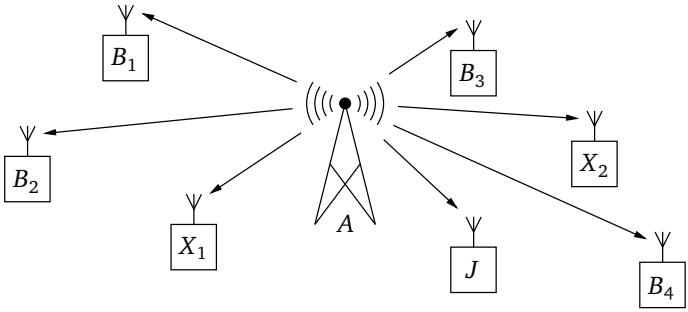

Figure 7.3: *UFH-based emergency alert broadcast: Using UFH, a sender is able to disseminate a message to a set of unknown or untrusted receivers in an ad-hoc and jamming-resistant manner.*

mission, or (2) if a distress call in high sea operations (nautics) needs to be undertaken in face of an (imminent) adverse invasion (see Figure 7.3).

Even under jamming, information dissemination in these settings is crucial. Being able to disseminate the information within a delay (even of seconds) under jamming is clearly preferred over not being able to communicate any information at all. Once the information has been received by some entities, other communication means (e.g., speech or landline) may additionally support its dissemination to more people or authorities concerned.

In addition to the single-hop broadcast scenarios given above, the anti-jamming emergency alert property of UFH communication can also be used for (multi-hop) jammer alarm forwarding in mobile ad-hoc or mesh networks. Jamming is a menacing threat to wireless networks because it deactivates the communication channel and thus, apart from disrupting normal network communication, also disables the transmission of jamming alerts and communication targeting to counteract the ongoing jamming. Here, UFH can be used for the delivery of short warning messages outside of the jammed region in an ad-hoc manner (i.e., without the need for any previous coordination among the nodes) from where external countermeasures can be taken.

7.3 Anti-jamming Navigation Broadcast

Another well-suited application for UFH communications is the broadcast of navigation signals which are primarily used for time synchronization, lo-

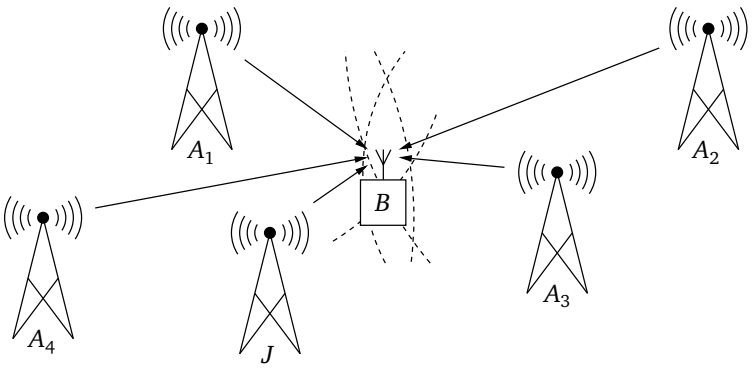

Figure 7.4: *UFH-based Navigation Broadcast: jamming-resistant reception of navigation signals used for positioning and/or time-synchronization. The receiver receives the landmark transmissions of multiple senders simultaneously and uses them to compute its position and/or local time.*

calization, and navigation. Examples of navigation systems include satellite navigation (e.g., GPS [33]) and terrestrial systems such as Loran [8] (based on TDoA) and DME-VOR [15] (based on distance/angle measurements). Localization and time-synchronization systems require the reception of navigation signals from multiple base stations; in general, three or four different signals are necessary for most localization methods [15]. The broadcast stations are precisely time-synchronized (e.g., via wired links) and located at static or predetermined positions. Each broadcast station transmits navigation signals either continuously due to a fixed schedule (GPS, Loran-C) or sends replies to individual localization requests (DME-VOR, WLAN-localization), based on which the localized device determines its position.

Without appropriate protection, navigation signals are vulnerable to signal spoofing, signal synthesis, and jamming attacks [43, 64]. Even though current civilian implementations using GPS satellite signals [33] or terrestrial WLAN signals [9] apply spreading to make the transmissions resistant to *unintentional* interference, they do not provide any means to counteract targeted DoS attacks because their spreading codes are public and can thus easily be misused for jamming.

Whereas data integrity (and confidentiality) can be achieved by cryptographic functions, the precise (relative) arrival times of navigation signals

at the receiver are also security-critical for many localization methods [43]. An attacker should thus not be able to block or delay the signals.

UFH communication offers an enhancement to the dissemination of navigation signals that counters targeted jamming. The fact that UFH does not require the sender and the receiver to be coordinated enables the receivers to receive several navigation signals in one reception phase (see Figure 7.4). The receivers each hold the authentic public key of the base stations (although they do not share secret keys with them) and can receive three (or more) individual messages, verify their authenticity, and therefrom derive position and/or time information. In order for the time information to be accurate enough, authenticated packet-level time-stamps must be used (see Section 4.3). Depending on the implementation and underlying hardware, the delay in the reception of UFH messages may vary. However, even if UFH causes a delay, the computed position and time are accurate since the aforementioned timestamping still enables the receiver to deduce the exact transmission and arrival times of the packets he receives.

Chapter 8
Prototype Implementation

To demonstrate the feasibility of the proposed uncoordinated frequency hopping schemes, we created a prototype implementation based on Universal Software Radio Peripherals (USRPs) [48] and GnuRadio [2]. In this chapter, we first outline the characteristics and configuration of the employed USRP hardware platform, subsequently describe our software implement in more detail, and finally present the experimental results that we obtained with this prototype.

8.1 USRP Platform Specification

The USRPs consist of a mainboard for the baseband processing and two exchangeable daughterboards for the down- and up-mixing from and to the radio frequency band. This modular design allows the USRPs to be used on different radio frequency bands. The mainboard includes four analog-to-digital (A/D) and digital-to-analog (D/A) converters of type AD9862 that provide an A/D (D/A) sampling rate of 64 Mb/s (12 Mb/s) and a sample resolution of 12 bits (14 bits). The sample conversion is coordinated by an Altera Cyclone EP1C12Q240C8 FPGA which is connected to a Cypress USB 2.0 controller.

In our setup, each USRP was equipped with two RFX2400 daughterboards (covering a carrier frequency range from 2.3 to 2.9 GHz) and was connected via a 480 Mb/s USB 2.0 link to a Lenovo T61 ThinkPad (Intel Core 2 Duo CPU @ 2.20 GHz) running Linux (kernel 2.6) and GnuRadio version 3.0.3 (see Figure 8.1). In the basic USRP configuration one daughterboard was used in reception, the other in transmission mode, thereby enabling full duplex communication on each device. For the experiments with a single sender, all but one daughterboard were set into reception mode so that one USRP device could act as two receivers.

8.2 Implementation Overview

For performance reasons and for ease of deployment, our sender and receiver applications were written entirely in C++, which—at the time—required porting some GnuRadio libraries from Python to C++. Moreover, to achieve a synchronization between writing the signal data to the USRP

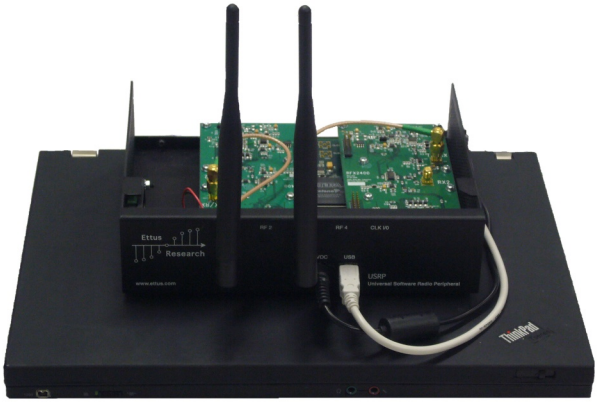

Figure 8.1: *Hardware setup of our UFH prototype, consisting of a Universal Software Radio Peripheral (USRP) and a Lenovo T61 ThinkPad.*

and the actual signal generation that is accurate enough to enable frequency hopping, the USRP drivers had to be adapted in way such that a call-back handler is called whenever the data for a single hop was completely received by the USRP.

Altogether, our implementation comprises tree parts: the low-level drivers to access the USRP device, the core UFH implementation, and the broadcast and key establishment applications. The structure of the core UFH implementation is further outlined in Figure 8.2, a schematic description of the sending and receiving process in Figure 8.3.

In our UFH implementation, we use LT erasure codes [49] with optimal degree distributions [39] for the message fragmentation; for these codes the number of non-verifiable sets (see Section 6.2) after the reception of j out of n packets is $D(j) = \binom{n}{j}$ if $j < l$ and $D(j) \approx \frac{1}{60} \frac{\sqrt{l} \log^2 l}{j-l+1}$ otherwise [49]. LT erasure codes are easy to implement and provide a reasonable tradeoff between encoding/decoding performance and correction capability. All fragments are mutually linked with either a single hash of 72 bits or an accumulator witness of 144 bit. The sender encapsulates the fragments into packets, encodes them with a (8,4)-Hamming code, and scrambles (interleaves) the bits according to a public pseudo-random permutation. Packets are transmitted on a randomly chosen frequency from the set {2.301 GHz, 2.303 GHz, ..., 2.700 GHz} of 200 channels using Gaussian Minimum Shift Keying (GMSK) at a bitrate of $R = 1$ Mb/s. After the last bit of a packet has

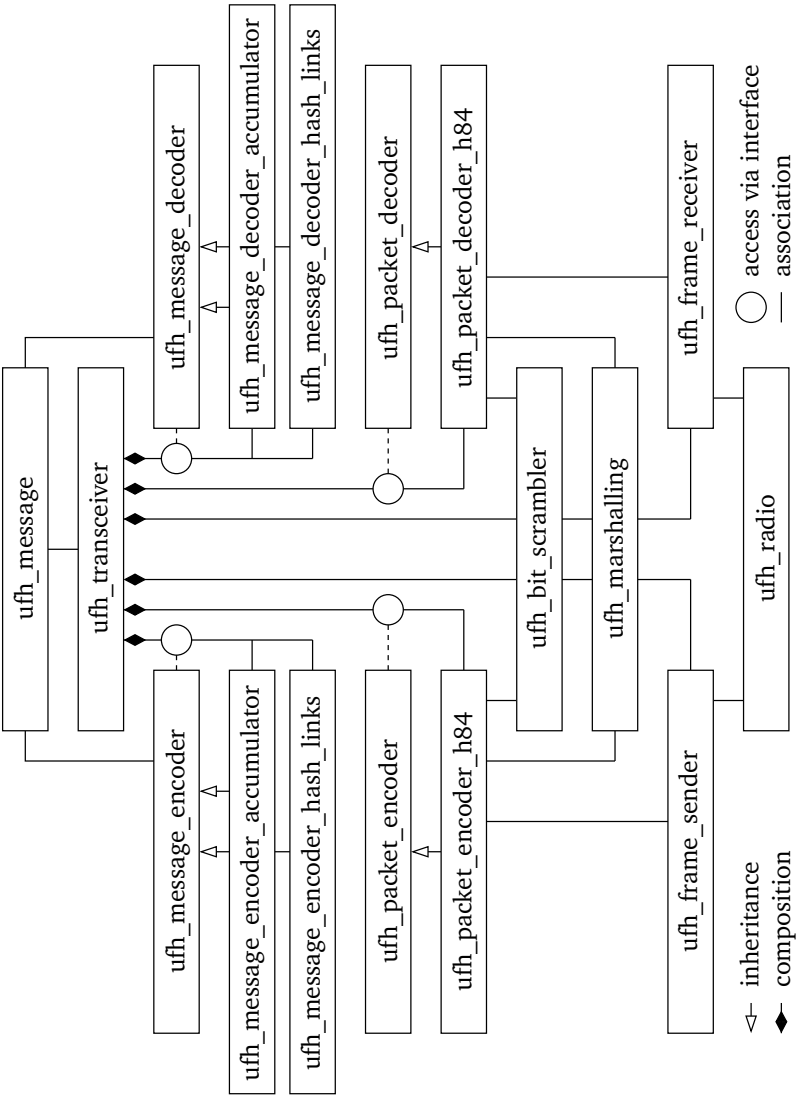

Figure 8.2: *Class diagram of the core UFH implementation.*

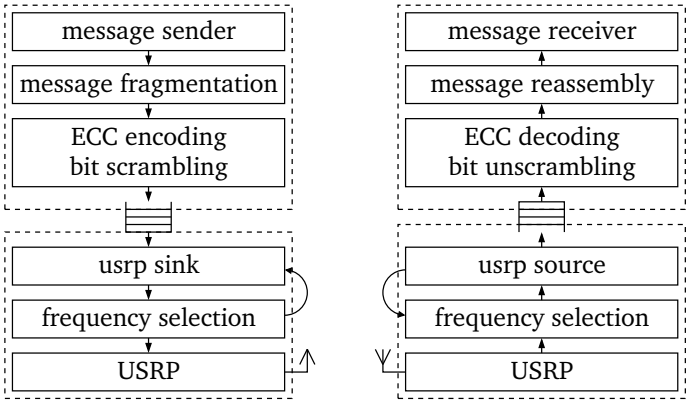

Figure 8.3: *Schematic description of our UFH sender and receiver implementation.*

been transmitted, the USRP driver switches to the new frequency channel and waits until the switching procedure has completed. With the used USRPs, switching a frequency channel took up to 500 μs. Given this rather long switching time, the output signal of the transceiver is set to zero during the transition to avoid that a signal is emitted before the actual packet transmission (what would help the attacker in detecting the transmission). As a result, each frequency slot consists of a first phase in which the actual packet transmission takes place and a second phase in which the transmitter is reconfigured and no signal is emitted (see Figure 8.4). The overall packet transmission time is thus $t_m \approx s/R + 500\,\mu$s, where s is the length of an encoded packet. Note that purpose-built frequency hopping transceivers can provide much faster frequency switching times in the order of one microsecond and much higher data rates of several tens of Mb/s.

The UFH receivers switch the input channels at a rate of about 100 Hz. If a signal is detected, they continuously try to decode the received data. Successfully received fragments are verified and stored in the corresponding message buffer. Once enough fragments are available, the message is reassembled and the used packets are removed. If the message is further accepted by the application as being authentic (i.e., if the signature is valid), the whole message buffer is discarded and the message gets appended to the message history buffer.

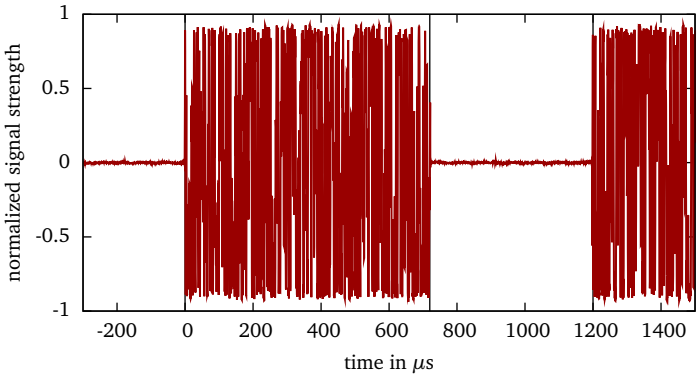

Figure 8.4: *Sampled baseband signal of an UFH transmission. Each frequency hop consists of two phases: In the first phase (about 720 μs in the example), the packet is transmitted whereas in the second phase (about 480 μs in the example), no signal is emitted while the transmitter is reconfigured.*

Our implementation of the Diffie-Hellman (DH) and Burmester-Desmedt (BD) key agreement protocols uses 256-bit prime fields for the elliptic curves and 32 bit values for identities and timestamps. This results in a message size for the DH key agreement of 1144 bits if the certificates are pre-shared and about 3360 bits if the certificates must be included in the messages. For the BD key agreement, the message sizes are 1000 bit for the first and 1512 bit for the second message.

In addition to the UFH sender and the UFH receiver, we also used the USRP platform for implementing a reactive-sweep jammer. In this implementation, the first of the two daughterboards is used for scanning, the second for jamming. Unfortunately, the high channel-switching times of the USRP platform severely limit the jamming efficiency and do not allow high jamming probabilities. For this reason and for the sake of accuracy, we simulated the jamming in our experiments in software by discarding a received packet with probability p_j.

8.3 Experimental Results

In our first experiment, we performed a series of message broadcasts from a single sender to a set of up to three receivers and measured the time after which the last receiver was able to reconstruct the message. The

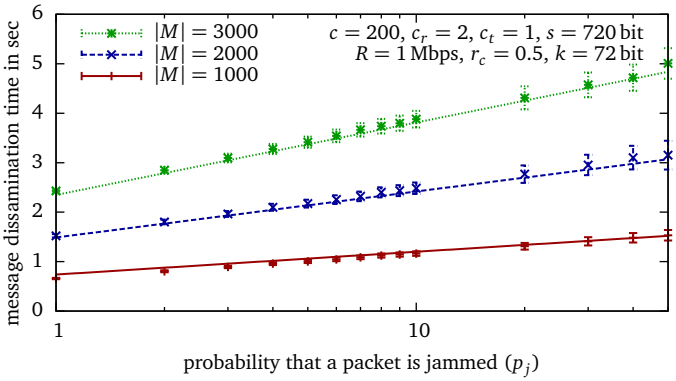

Figure 8.5: *Time to broadcast a message with our USRP-based prototype implementation. The lines show the expected theoretical results, the points and confidence intervals the findings of our experiments. We observe that the message transmission time increases linearly with the message size but only logarithmically with the number of receivers.*

presented measurement results for more than three receivers were obtained by combining multiple runs for the same parametrization; this procedure is reasonable given the inherent randomness of the reception process (i.e., a sample that is obtained by combining the results of two receivers and five measurements is statistically equivalent to a sample that is obtained by a single measurement with ten receivers). In our second experiment, we evaluated the performance of the DH and BD key agreement protocols. We performed several DH and BD key establishments and measured the time after which the last participant was able to compute the shared key.

The average time to broadcast a message with our USRP-based prototype implementation is shown in Figure 8.5, the average time to establish a shared key in Figure 8.6. We observe from Figure 8.5 that the message transmission time increases linearly with the message size but only logarithmically with the number of receivers. The results depicted in Figure 8.6 further show that even if the processing gain of 23 dB allows the attacker to jam up to 50% of the packets, a key can be established between two devices in less than 4 s and among a group of 10 devices in less than 7 s (assuming that certificates have been pre-shared). We point out that purpose-built frequency hopping transceivers can provide frequency switching times in the order of one microsecond and bit-rates of several tens of Mb/s. Realistic times of purpose-built UFH transceivers are thus likely

to be 10 to 100 times lower than what we achieve with the presented implementation. Nevertheless, our results clearly show that the proposed UFH-based communication is feasible in practice.

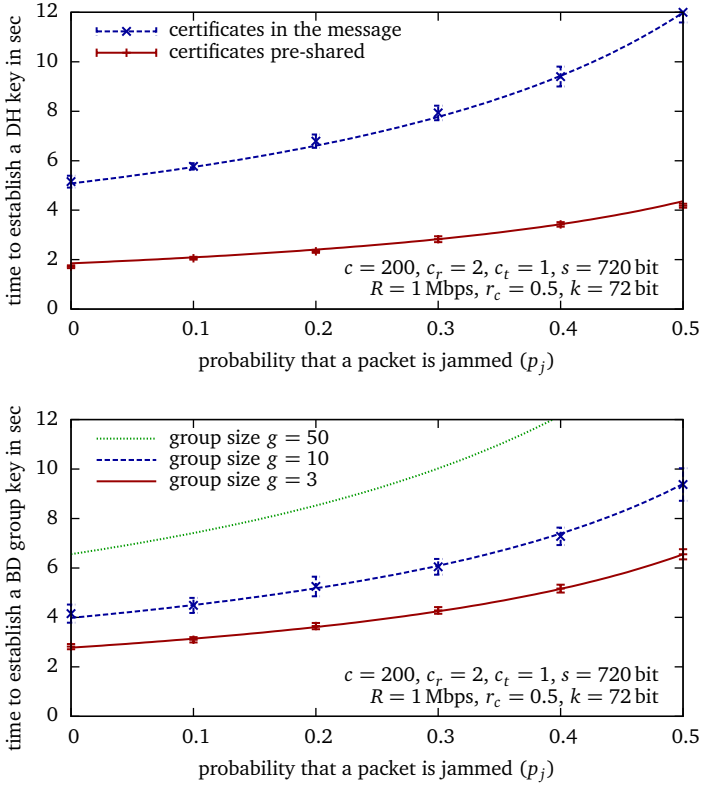

Figure 8.6: *Time to establish a shared key with our USRP-based prototype implementation of the Diffie-Hellman and the Burmester-Desmedt key-agreement protocols. The lines show the expected theoretical results, the points and confidence intervals the findings of our experiments. We point out that purpose-built UFH transceivers enable to decrease these times significantly.*

Chapter 9
Related Work

In this chapter, we discuss the work that is related to Part I of this thesis. For the most part, we focus on work that addresses the impact and mitigation of jamming and refer to Chapter 15 for a discussion of related work that reviews the detection of jamming.

9.1 Impact of Jamming

Due to the fact that communication jamming is not a novel threat, the impact of jammers according to their capabilities (e.g., broadband or narrowband) and behavior (e.g., constant, random, reactive) has been widely studied [4, 5, 13, 21, 22, 25, 31, 47, 56, 60, 61].

Commander et al. investigated the so-called wireless network jamming problem, that is, the minimum number of required jamming devices and their optimal placement to suppress all communication in a network [21, 22]. In their work, they present several optimization formulations for covering the communicating nodes with jamming devices and for limiting the connectivity index of the network nodes (i.e., the probability that there exists a path between two nodes). They further incorporate percentile constraints to develop formulations which provide solutions that require less jamming devices at the cost of a reduced solution quality.

The impact of jamming on the connectivity in ad-hoc networks was studied by Noubir [56]. He investigated both the case where the jammers can be optimally placed by the attacker and the case where the jammers are randomly located within the deployment area. His findings show that even a small number of jammers can drastically reduce the network connectivity when the nodes communicate using omnidirectional antennas. They also show that directional antennas significantly improve the network connectivity at the expense of an increased average number of hops and that mobility allows further resiliency to jamming.

A class of jamming attacks, referred to as flow-jamming attacks, in which an adversary with multiple jammers in the network jams packets to reduce traffic flow were analyzed by Tague et al. [73, 74]. In their work, they propose the evaluation metrics jamming impact (i.e., average fraction of jammed flow rate over all flows), jamming efficiency (i.e., ratio of jamming impact to the average jamming resource expenditure), and jamming resource variation (i.e., relative difference between the maximum

and minimum jamming resource expenditure) and formulate optimal flow-jamming attacks with respect to these metrics using linear programming. They further show that even in the absence of a centralized jammer coordination, efficient flow-jamming attacks can be performed using distributed algorithms.

A second jamming-related problem considered by Tague et al. is jamming-aware source routing in which the source node performs traffic allocation based on empirical jamming statistics at individual network nodes [72]. They formulate the traffic allocation as a network flow optimization problem and show that in multi-source networks, this centralized optimization problem can be solved using a distributed algorithm.

Gilbert et al. established bounds for several classical multiplayer problems such as leader election or binary consensus under the influence of malicious interference [31]. Specifically, they show that a jammer that is allowed to block up to β messages can delay the communication of two players for $2\beta + \lg|V|/2$ rounds, where V is the set of possible values that the players communicate. Via reduction to this bound they further derive a lower bound of $2\beta + \Theta(\lg|V|)$ rounds for reliable broadcast, $2\beta + \Omega(\log \frac{n}{k})$ rounds for leader election, and $2\beta + \Omega(k \lg \frac{|V|}{k})$ rounds for static k-selection. In the latter two bounds, k represents the number of participants which contend to become leader and to transmit their initial value, respectively.

The performance of IEEE 802.11 under jamming was evaluated by Thuente and Acharya [76] and Bayraktaroglu et al. [13]. In their studies they consider proactive (periodic) jammers as well as reactive jammers that jam only non-colliding transmissions and might optimally adjusts their strategy according to the current states of the participating nodes. The evaluations by Thuente and Acharya are solely based on OPTNET simulations whereas the study by Bayraktaroglu et al. comprises a theoretical analysis of the saturation throughput of 802.11 based on discrete-time Markov chains, extensive Qualnet simulations, and real world experiments using GNU Radio and the USRP platform. Their findings show that even a simple periodic jammer is highly damaging when the network is saturated and that reactive jammers can drastically reduce the throughput of IEEE 802.11 networks with only a limited energy cost for the attacker.

9.2 Jamming Mitigation

Spread-spectrum (SS) techniques such as FHSS, DSSS, and chirp SS [5, 60, 61] achieve frequency diversity over the communication channel and are well-studied countermeasures against communication jamming attacks.

Jamming and (unintentional) interference are further thwarted by highly directional antennas [5] and by the application of forward error-correcting codes and special coding strategies [47, 79]. Over the last years, various additional countermeasures have been proposed [19, 23, 71, 77, 82]: Desmedt et al. and Tague et al. studied the problem of compromised insiders and proposed efficient methods to assign channels to receivers in a way such that the capability of colluding groups to jam the reception of other receivers is minimized [23, 71]; Tague et al. additionally proposed an algorithm for the identification of compromised users [71]; Xu et al. examined the ability of sensor networks to cope with radio interference and proposed a scheme called channel surfing, whereby (groups of) sensor nodes successively switch their communication to a backup channel if they assume that the current channel is jammed [82]; Čagalj et al. showed how wormholes based on wires or on channel hopping can be used to establish communication out of a jammed area [77]; and Chiang et al. proposed a scheme for broadcast jamming mitigation in which the spreading codes are organized in a binary tree and which enables the detection of compromised nodes with redundant transmissions using test codes (i.e., codes at the leaves of the tree that belong to a single node) [19]. However, all these countermeasures rely on secrets that were pre-established between the sender and the receiver(s) or are very specific regarding the attacker's capabilities and can thus not be applied in the considered scenarios.

Recently, the theory and systems community became aware of the shortcoming of non-existing methods for jamming-resistant communication without shared secrets and started to propose solutions for this problem.

Dolev et al. presented f-AME [25], a round-based, randomized protocol to set up group keys in the presence of message collisions and insertions. In f-AME, nodes recruit surrogates that relay messages on their behalf and in each round $t+1$ honest nodes simultaneously broadcast on $t+1$ channels, where t is the number of channels that the attacker can jam, so that at least one transmission succeeds. A major limitation of f-AME is its requirement of a (fully connected) group of size $> 3(t+1)^2 + 2(t+1)$; usually t is in the order of tens or even hundreds of channels, requiring a group of hundreds or even thousands of nodes.

The solution proposed by Baird et al. [10] uses concurrent codes in combination with UWB pulse transmissions. A concurrent code is a set of bit vectors such that no vector is a subset of a bitwise or of a small number of other vectors. To send a vector, each bit of a vector is assigned to a timeslot and an UWB pulse is transmitted if the bit is set. The receiver records the received pulses and tries to identify the sent vector by eliminating those

vectors from the set of possible vectors for which an expected pulse was not received. The basic assumption behind this approach is that the attacker cannot remove UWB pulses from the radio channel but only insert additional ones. The jamming resistance achieved by this scheme is, however, not one-to-one comparable to common spread-spectrum-based techniques: While the successful jamming of a spread-spectrum communication requires that the attacker has enough transmission power to overcome the processing gain, with concurrent codes the limiting factor is the number of pulses that the attacker can insert, that is, the energy of the attacker.

Jin et al. introduced intractable forward-decoding along with efficient backward-decoding as a new concept to enable DSSS communication without shared keys [40]. Their solution is based on the observation that if a seed of size k is large enough to ensure that the attacker cannot identify the therefrom generated spreading sequence on time (i.e., before the message has been transmitted), then the scheme remains secure even if we use a seed of size k for the first half of the message an a seed of size $k-1$ for the second half of the message (although the attacker must try only half as many keys for the second half of the message as for the first half, she also has only has half as much time left). In the proposed scheme, the sender first selects a random seed of size k and divides the message into l parts. To generate the spreading sequence for the transmission of the i-th part, the i most significant bits of the seed are replaced with the i most significant bits of the receiver's MAC address, thereby reducing the entropy in the seed to $k-i$ bits. The receiver exploits this fact by first recording the entire message transmission and then decoding the message in reverse order, starting with the last part. Once the $k-l+1$ bits that were used to spread the last part have been identified, only one additional bit must be guessed for each additional part. The presented approach is very efficient in terms of energy costs but targets at pairwise communication.

Pöpper et al. proposed Uncoordinated DSSS (UDSSS) as an alternative approach to enable DSSS-based (broadcast) communication without a shared key [63]. UDSSS follows the principle of DSSS in that the transmitted data is spread using spreading sequences. However, as opposed to DSSS, where the spreading sequence is known to the sender and the receiver, in UDSSS the sender uses a randomly chosen spreading sequence from a publicly known code set. The receivers—unaware of the right spreading sequence—first record the entire message transmission and then decode the message by trying all sequences from the code set. The required decoding time therefore depends on the size of the code set and on the receivers' computing power. The size of the code set must further be large

enough to prevent that the attacker can identify the right sequence before the message has been transmitted. As a result, the jamming resistance of UDSSS depends not only on the achieved processing gain (i.e., on the attacker's transmission power) but also on the attacker's computing power.

From a conceptual point of view, the two main advantages of UDSSS over UFH are that in UDSSS messages must not be fragmented and that UDSSS usually provides the better LPI characteristics. The two main advantages of UFH over UDSSS are that UFH achieves the same jamming resistance as the related coordinated scheme and that UFH does not impose additional requirements on the FH transceivers (and thus can easily be applied to existing systems). The absolute performance of the two schemes heavily depends on the capabilities of the employed hardware; that is, on whether FH or DSSS is better supported.

9.3 Secure Erasure Coding and Hash Links

The distillation codes introduced by Karlof et al. [41] have a similar goal as the verification techniques presented in this thesis. However, because UFH communication relies on short packets, the distillation codes based on Merkle trees proposed in their work are not well suited for our purposes.

In the context of distributed data storage, Hendricks et al. proposed the application of homomorphic fingerprinting [37, 38] to ensure that all erasure code fragments belong to the same file. The presented fingerprints preserve the structure of the erasure code and allow that each fragment can be verified independently. However, unlike in the data storage setting, the (comparatively large) size of the fingerprints can no longer be neglected in the considered anti-jamming scenario and severely limits their application to our schemes.

Independent of our work, Slater et al. [67] proposed two additional packet verification techniques (Hashcluster and Merkleleaf) and one similar to our scheme based on cryptographic accumulators (Witnesscode). Their results confirm our finding that cryptographic accumulators are an excellent building block for creating efficient message coding schemes.

The concept of hash chains was fist proposed by Lamport [44] and has since been extensively used in various cryptographic systems. In particular, the use of hash chains as an efficient means to protect lossy packet streams was proposed by Perrig et al. [59] and extended to bidirectional, multiply-linked hash graphs by Golle and Modadugu [32] and Miner and Staddon [54]. As opposed to our work, these works focus on authentication and assume the availability of shared keys.

Part II

Detection of Reactive Jamming in Wireless Sensor Networks

Chapter 10
Introduction

Initially motivated by battlefield intelligence, wireless sensor networks (WSNs) have expanded into a number of security and safety critical civilian applications including emergency response support, fire and burglar alarm systems, and the protection of critical infrastructures. Common to these applications is that they rely on dependable and timely delivery of alarm notifications. These alarms are typically raised by sensor nodes upon the detection of a sensed event (e.g., presence of an intruder) and must subsequently be forwarded to the network authority in a hop-by-hop manner. A sensor network that supports these applications must therefore guarantee the timely delivery of alarms even under jamming attacks.

The expected lifespan of such sensor network applications ranges from months to years and, given the limited power supply of sensor nodes, places high demands on the energy efficiency of the running algorithms. To meet these demands, existing surveillance applications [26, 34, 36, 69] combine low duty-cycling with reactive notification. Here, alarms are only transmitted upon detection of an event (i.e., for the network authority "no news is good news"). While such behavior is highly desirable in energy-constraint sensor networks, in conjunction with the low output power and limited spectral diversity of sensor node transceivers, it makes the alarm forwarding highly vulnerable to jamming-based denial-of-service attacks (i.e., alarm masking); these attacks have been shown to come at a low cost for the attacker while being particularly harmful to the timely delivery of critical information [77, 80, 83].

In principle, there are two solutions to counter jamming attacks on alarm forwarding: jamming mitigation and jamming detection. However, common spread-spectrum-based jamming mitigation techniques such as FHSS or DSSS are beyond the capabilities of current sensor nodes and existing jamming detection techniques for sensor networks do not suffice to protect the considered reactive message forwarding. Existing jamming detection techniques rely on the packet-delivery-ratio (PDR) and/or the received ambient signal strength as their main decision criteria [46, 57, 60, 83] and have been shown to be well-suited for the detection of *proactive* mid- or long-term jamming [46, 83]. They are, however, not designed to detect *reactive* (packet or single-bit) jamming: Firstly, existing jamming detection techniques rely only on the CRC of a packet to decide whether

it was received correctly and therefore can (in general) not distinguish between packet failures due to weak radio links and interference. Secondly, assessing an accurate PDR is not practical in a reactive forwarding scheme for messages are sent very rarely. Thirdly, jamming does not necessarily cause a steady and high received signal strength (RSS) value, as only a small fraction of a packet has to be interfered with in order for the packet to be invalid [5, 57, 60]. A reactive jammer can thus keep the increase in the effective RSS value very low and hence avoid being detected with current approaches.

In this work, we propose a novel jamming detection scheme as a solution to these problems. Our scheme is able to identify the cause of bit errors for individual packets with high probability by looking at the received signal strength (RSS) during the reception of these bits; bit errors are detected either based on predetermined knowledge, error correcting/detecting codes, or limited node wiring in the form of wired node chains (n-tuples). The intuition behind this process is that if there was a bit error although the RSS value was high, this indicates external interference (intentional or unintentional); if the bit error was due to a weak signal (e.g., due to fast fading or shadowing), the RSS value should be low. This additional information allows an accurate differentiation of packet errors due to intentional interference from errors due to weak links, even in the case of a sophisticated (reactive) attacker that jams only a small portion of a packet.

We discuss the strengths and weaknesses of the proposed bit-error identification techniques and evaluate their jamming-detection performance analytically, by simulations, and experimentally with an implementation on BTnodes [1]. The evaluation results confirm that our solution meets the performance and accuracy requirements of (reactive) alarm forwarding protocols and enables the detection of advanced jamming attacks in which the attacker can freely choose the duration, strength, and beam width of the jamming signal. To the best of our knowledge, this work is the first to present a jamming detection scheme for sensor networks that enables the detection of reactive (single bit) jamming or overshadowing on a per-packet basis. We further believe that this work provides useful insights into the utility of limited wiring as a means for securing wireless sensor networks. In particular, our work shows how the combination of limited node wiring and low-power wire integrity verification enables the formation of robust sensor networks for the timely and dependable delivery of alarm notifications.

10.1 Contributions

In summary, the three main contributions of this part are:

- We demonstrate the susceptibility to jamming attacks of current state-of-the-art forwarding schemes for WSNs.
- We discuss possible techniques for the mitigation of jamming based on forward error correction and limited node wiring (n-tuples).
- We present a novel jamming detection scheme for countering advanced (reactive single bit) jamming attacks in wireless sensor networks and develop three different techniques for the identification of bit errors based on: predetermined knowledge, error correcting codes, and limited node wiring (n-tuples).

10.2 Outline

The remainder of this part is organized as follows: In Chapter 11 and 12 we motivate our work by demonstrating the severe impact of reactive jamming on (safety-critical) alarm forwarding in WSNs. We present our novel jamming detection scheme in Chapter 13 and evaluate it in Chapter 14. Finally, we discuss related work in Chapter 15.

Chapter 11
Motivating Example: Alarm Forwarding in Safety-critical Sensor Networks

In this chapter, we present a state-of-the-art alarm forwarding protocol for safety-critical wireless sensor network (WSN) applications, such as fire and burglar alarm systems. Using WSN technology for implementing safety-critical surveillance applications is a challenging task: Alarms detected by sensor nodes have to be reported reliably and within a few seconds to at least one sink node, even in case that some of the nodes and communication links fail. A complicating factor is that maintenance costs have to be very low should the application be economically feasible. This requires an energy-efficient operation of the sensor network for batteries should not be replaced more often than once every two to three years.

With current generation sensor node hardware built out of COTS components, the radio consumes the most power, and the above lifetime requirement translates into a duty cycle of less than 1% and asks for a reactive forwarding scheme where no notifications are sent in the absence of a noteworthy event. The three fundamental requirements associated with safety-critical applications are thus: reliable data delivery, low latency, and low energy consumption. To jointly address these three requirements, we present Dwarf, a Delay-aWAre Robust Forwarding algorithm that is based on the following observations and assumptions:

a) One of the most robust, yet simple forwarding algorithms is flooding because it ensures that a message will eventually reach its destination as long as the network remains connected.

b) Traditional flooding is very expensive (with regard to energy consumption and transfer costs) and does not consider the message delivery time.

c) Nodes duty-cycle their radio to increase network lifetime and spend most of their time in sleep mode. Moreover, to minimize overheads and reduce protocol complexity, nodes do not synchronize globally and wake up independently of each other.

d) The (worst-case) node-to-sink notification time is determined by the relatively long sleep periods of the forwarding nodes along the path.

The fundamental idea of Dwarf is to perform a unicast-based partial flooding towards the sink in combination with a (greedy) delay-aware node selection strategy to overcome the drawbacks mentioned above. More precisely, the number of neighbors k to which an alarm is forwarded determines the degree of introduced redundancy, thus making the algorithm more robust at the expense of an increase in the number of messages and the associated complexity in handling peak loads (e.g., collisions). The selection of the destination nodes according to their wake-up times and relative positions aims at reducing the overall alarm notification time. That is, neighbors that wake up first and are closer to a sink are favored over nodes that wake up later or are not on the shortest path towards the sink.

11.1 Application Scenario and Requirements

The concrete application scenario for which Dwarf was designed is a distributed indoor wireless alarm system. Each sensor node consists of a micro controller (e.g., an ATMEL ATMega128), a communication unit (e.g., a CC1000 transceiver), a power supply (in the order of 2 AA batteries) and a sensor for detecting a specific alarm condition. All nodes are manually deployed at fixed locations in a building as with ordinary, wired sensors. In addition, there is at least one mains-powered sink node that is connected to a central control station. Domain specific regulations [28] require that an alarm raised by a sensor is reported at the control station (sink) *within 10 seconds*, which leaves little room for per-hop delays in typical office buildings with long corridors and one control station per floor. It is also strongly advised that wireless alarm systems use the frequency band from 868.6-868.7 MHz, which is exclusively to be used for wireless alarm systems.

Further requirements are that failed nodes must be replaced and that the integrity of the restored system is asserted by a qualified technician. The latter is a costly operation and for financial reasons it therefore makes sense to replace the batteries of all nodes as soon as the first one runs out of energy. To reduce operational costs, such a grand replacement procedure should, however, not occur more often than every two to three years. This consideration requires that the proposed Dwarf protocol minimizes and equalizes the power consumption of all sensor nodes and that it operates with a very low duty cycle.

In order to achieve such a low duty cycle, the employed MAC protocol has to be carefully chosen. One class of MAC protocols that achieve very

low duty cycles, are the low-power listening (LPL) protocols, of which B-MAC [62] is the best-known exponent. A big advantage of these protocols is that they do not require a central synchronization, and hence do not burden the communication budget with an extra synchronization overhead. As a basis for our alarm forwarding protocol, we chose WiseMAC [27], an elaborate member of the class of LPL protocols.

Similar to the operation of B-MAC, with WiseMAC a node wakes up periodically and checks if there is any activity on the radio channel. If no activity is detected, the node goes right back to sleep, otherwise it keeps on listening to receive a potential message. To ensure that messages are not missed, each message is prepended with a wake-up preamble that is slightly longer than the wake-up period so that the intended receiver will sense an active channel and receive the entire transmission. A major advantage of WiseMAC over other LPL protocols is its ability to learn the receiver's wake-up schedule with every message exchange. Specifically, the receiver tells the sender in the acknowledgment for how long it has been receiving the preamble before the actual message transmission started. This additional knowledge allows the sender to shorten the preambles and to start a transmission right before the intended receivers wake up. The gain of this procedure is not only a significantly reduced energy consumption but also a lower utilization of the wireless channel. The exact length of a preamble depends on the estimated clock drift of the intended receiver and the time that passed since it was last contacted.

11.1.1 Definitions

Throughout this chapter, we represent the sensor network by the graph $G := (V, E)$ consisting of the set of sink nodes $S \subset V$, the set of sensor nodes $V \setminus S$, and the set of edges E. All communication links are considered to be bidirectional and two nodes $u, v \in V$ can directly communicate with each other (i.e., are neighbors) if and only if $\{u, v\} \in E$. All sensor nodes are organized according to their distance to the nearest sink; nodes with the same distance are said to be on the same level:

Definition 1. *Let $d(u, v)$ be the distance (i.e., the length of the shortest path) between two nodes u and v. A node u is said to be on level $l(u) = i$ with respect to the set of sink nodes S if and only if $\min\{d(u, s) : s \in S\} = i$. The set $L_i := \{u : u \in V \wedge l(u) = i\}$ contains all nodes on level i and the maximal level is denoted by $\hat{l} := \max\{l(u) : u \in V\}$.*

Based on Definition 1, the neighbors of a node are divided into parents, siblings, and children:

Definition 2. *We denote the set* $N_u^P := \{v : \{u,v\} \in E \wedge l(v) = l(u) - 1\}$ *as the parents of a node u, the set* $N_u^S := \{v : \{u,v\} \in E \wedge l(v) = l(u)\}$ *as its siblings, and the set* $N_u^C := \{v : \{u,v\} \in E \wedge l(v) = l(u) + 1\}$ *as its children, respectively.*

As already mentioned, we assume that all nodes but the sinks sleep most of the time in order to save energy, and only wake up periodically. The wake-up period of the nodes is denoted by T_w, $0 < T_w(u) < \infty$.

Definition 3. *Let* $\tau_{u,i}$, $u \in V \setminus S$, $i \in \{0,1,2,\ldots\}$ *be the wake-up times of node u and* $\tau_{u,i+1} = \tau_{u,i} + T_w$. *The duration until the upcoming wake-up time relative to the current time t is denoted by* $\tau(u) := \min\{\tau_{u,i} : \tau_{u,i} > t\} - t$. *Sink nodes are assumed to be always listening, hence,* $\forall s \in S : \tau(s) = 0$.

Finally, we assume that there exists a constant upper bound $t_m \in O(1)$ on the transmission time of any message m, and use k to denote the (constant) upper bound on the number of neighbors to which a message is forwarded.

11.2 Delay-aware Alarm Forwarding

The objectives of the Dwarf algorithm are to ensure that (i) an alarm message finally reaches a sink node and that (ii) it does so as quick as possible. To this end, each node keeps track of the (estimated) wake-up times of its parents and siblings and forwards a message to the k parents and siblings that wake up next; neighbors which are known to have already received the alarm are not considered in this process. Forwarding the message to the parent that is the first to wake up minimizes the local forwarding delay, whereas sending the message to more than one neighbor decreases the probability that the alarm is lost. In detail, the forwarding algorithm works as follows (see Algorithm 1):

For each newly created or received alarm message m, the node maintains a set of parent and sibling candidates, denoted as C_m^P and C_m^S, respectively. These sets contain all the parents and siblings which are assumed to have not yet received the message m (i.e., from which neither an acknowledgment of m nor the message m itself has been received). As long as the parent candidate set is not empty, the parent that wakes up first is chosen as the next forwarding destination and subsequently removed from the candidate set. If there are no more parents to chose from (i.e., if $C_m^P = \emptyset$), the sibling that wakes up next is used instead. Once both sets are empty, they are reinitialized with $N_u^P \setminus B_m$ and $N_u^S \setminus B_m$, respectively,

Algorithm 1 Alarm forwarding for node u

```
1:  var H ← ∅
2:
3:  function INITCANDIDATES(m)
4:      C_m^P ← N_u^P \ B_m
5:      C_m^S ← N_u^S \ B_m
6:  end function
7:
8:  function GETNEXTHOP(m)
9:      if C_m^P = ∅ and C_m^S = ∅ then
10:         INITCANDIDATES(m)
11:     end if
12:     if C_m^P ≠ ∅ then
13:         select v ∈ C_m^P such that τ(v) = min{τ(v) : v ∈ C_m^P}
14:         C_m^P ← C_m^P \ {v}
15:         return v
16:     else if C_m^S ≠ ∅ then
17:         select v ∈ C_m^S such that τ(v) = min{τ(v) : v ∈ C_m^S}
18:         C_m^S ← C_m^S \ {v}
19:         return v
20:     else
21:         return ⊥
22:     end if
23: end function
24:
25: function SENDALARM(m)
26:     if r_m < r_a then
27:         r_m ← r_m + 1
28:         v ← GETNEXTHOP(m)
29:         if v ≠ ⊥ then
30:             send alarm message m to node v
31:         end if
32:     end if
33: end function
```

Algorithm 1 (continued) Alarm forwarding for node u

```
34: function FORWARDALARM(m)
35:     INITCANDIDATES(m)
36:     r_m ← 0
37:     a_m ← 0
38:     SENDALARM(m)
39: end function
40:
41: upon acknowledgment of alarm message m sent to w
42:     a_m ← a_m + 1
43:     B_m ← B_m ∪ {w}
44:     if a_m < k and w is not a sink then
45:         SENDALARM(m)
46:     end if
47: end upon
48:
49: upon drop of alarm message m sent to w
50:     SENDALARM(m)
51: end upon
52:
53: upon reception of alarm message m from v
54:     if m ∉ H then
55:         B_m ← {v}
56:         H ← H ∪ {m}
57:         FORWARDALARM(m)
58:     else if v ∉ B_m then
59:         B_m ← B_m ∪ {v}
60:     end if
61: end upon
62:
63: upon detection of an alarm
64:     create alarm message m
65:     B_m ← {}
66:     H ← H ∪ {m}
67:     FORWARDALARM(m)
68: end upon
```

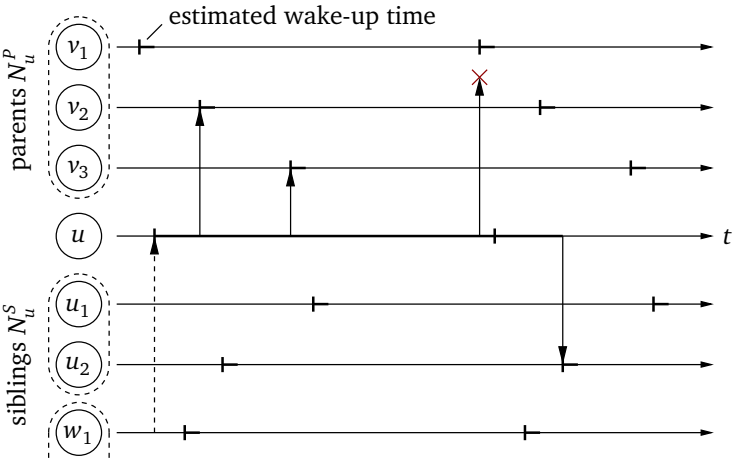

Figure 11.1: *Dwarf forwarding example for $k = 3$. Node u receives an alarm from its child w_1 and subsequently forwards the alarm to k of its parents $N_u^P := \{v_1, v_2, v_3\}$ and siblings $N_u^S := \{u_1, u_2\}$. The destination nodes are selected according to their distance from the sink (parents come first) and their estimated wake-up times.*

where B_m is the set of neighbors that have already received the message m. A node aborts the forwarding process as soon as the message has been successfully forwarded to k neighbors or to a sink node. The forwarding process is also aborted if there are no more parents and siblings to forward the message to (i.e., if $(|N_u^P| \cup |N_u^S|) \setminus B_m = \emptyset$). In any case, overall at most $r_a \geq k$ attempts are made to forward the message. Consequently, an alarm is dropped without having been forwarded after r_a unsuccessful transmission attempts.

Upon reception of an alarm message m, a node first verifies that the alarm has not already been forwarded (i.e., that m is not in the message history H). New messages are appended to the message history and then forwarded in the same manner as a newly generated alarm.

A forwarding example for $k = 3$ is shown in Figure 11.1. In this example, node u receives an alarm message from its child w_1 which it then forwards to the parents v_2, v_3, and v_1 according to their estimated wake-up times. After the unsuccessful transmission to parent v_1, the parent set is empty and the message is forwarded to the sibling u_2.

11.3 Robustness Properties

In the following, we analyze the maximal number of link and node failures that can be tolerated by our algorithm, compute an upper bound on the required hop count, and show that the proposed (greedy) destination selection algorithm is only a constant factor worse than an optimal solution.

Lemma 7. *The proposed alarm forwarding algorithm (Algorithm 1) can tolerate up to* $t_l := \min(k, \delta) - 1$ *link failures, where* $\delta = \min\{|N_u^P| + |N_u^S| : u \in V\}$ *and* k *is the number of forwarding destinations.*

Proof. Let us assume that u is the first node on level $i > 0$ to receive or create an alarm which cannot be forwarded to a node on level $i-1$. Consequently, all links to u's parents must be broken. In addition, for $\geq \min(k, |N_u^S|)$ of u's siblings either the links from u to these siblings or from these siblings to their parents must be broken. Per definition, each node has at least one parent. Thus, in total $\geq |N_u^P| + \min(k, |N_u^S|) \geq \min(k, \delta) > t_l$ links must be down, contradicting the assumption that there are at most t_l link failures. The threshold is tight for $k \geq \delta$ as a node u with $|N_u^P| + |N_u^S| = \delta$, which exists per definition, can be isolated from all its parents and siblings if $\geq \delta = t_l + 1$ links fail. □

Lemma 8. *If at most* t_l *links fail (see Lemma 7), an alarm initiated by node* u *on level* $l(u)$ *will reach the nearest sink after at most* $l(u) + \min(t_l, l(u)) \leq 2l(u)$ *hops.*

Proof. In order to extend the number of required hops by one, the links to all parents of a node v must be broken. However, as a message is forwarded to $\min(k, |N_v^S|)$ siblings, at least $\min(k, |N_v^S|) - (t_l - |N_v^P|) = \min(k + |N_v^P|, |N_v^S| + |N_v^P|) - t_l \geq \min(k, \delta) - t_l = 1$ of them are able to forward it to one of their parents. As a result, the number of required hops can be extended by only one for each level and requires that at least one link is down. The maximal number of additional hops is thus $\min(t_l, l(u))$. □

Definition 4. *For a node u, we denote by* $|N_u^Z|$ *the maximal number of siblings such that: (i) for each sibling there exists a path to a node on level* $l(u) - 1$ *that is not a parent of u; (ii) on each path, all but the last node are on level* $l(u)$*; and (iii) all paths are mutually node-disjoint. More formally,* $|N_u^Z| = \max\{|a| \mid a \subseteq N_u^S \wedge \forall v \in a : \exists$ *path* $(v, v_1, v_2, v_3, \ldots, v_q)$ *such that* $\forall v_i, 1 \leq i < q : v_i \in L_{l(u)}$ *and* $v_q \in L_{l(u)-1} \setminus N_u^P \wedge \forall v, w \in a, v \neq w : \forall i, j : v_i \neq w_j\}$

Lemma 9. *The proposed alarm forwarding algorithm (Algorithm 1) can tolerate up to* $t_p := \min(k, \gamma) - 1$ *node failures, where* $\gamma = \min\{|N_u^P| + |N_u^Z| : u \in V\}$ *and* k *is the number of forwarding destinations.*

Proof. Let us assume that u is the first node on level $i > 0$ to receive or create an alarm which cannot be forwarded to a node on level $i - 1$. Consequently, all parents of u must have failed. In addition, for $\geq \min(k, |N_u^Z|)$ of u's siblings, they, a node on the corresponding node disjoint path, or the corresponding node in the next level must have failed. Thus, in total $\geq |N_u^P| + \min(k, |N_u^Z|) \geq \min(k, \gamma) > t_p$ nodes must be down, contradicting the assumption that there are at most t_p node failures. The threshold is tight for $k \geq \gamma$ as a node u with $|N_u^P| + |N_u^Z| = \gamma$, which exists per definition, can be isolated from all nodes in the next level if we allow $\geq \gamma = t_p + 1$ node failures. □

Lemma 10. *If at most* t_p *nodes fail (see Lemma 9), an alarm initiated by node* u *on level* $l(u)$ *will reach the nearest sink after at most* $l(u) + \beta t_p$ *hops, where* $\beta = \max\{|N_u^C| : u \in V\}$.

Proof. Given that at most t_p nodes fail, there exists a path $(u, u_1, u_2, \ldots, u_m)$ which connects a node u with a node u_m in the next lower level. A node v that fails has at most $|N_v^C|$ children and thus can prevent at most $|N_v^C|$ nodes on level i from forwarding an alarm to level $i - 1$. Consequently, after at least $|N_v^C|$ hops in the same level, a node with a different parent is reached. As a result, each failed node v can extend the number of required hops by at most $|N_v^C| \leq \beta$ and the maximal number of additional hops is bounded by βt_p. □

Lemma 11. *The presented (greedy) destination selection algorithm selects a route that is at most* $1 + \frac{T_w}{t_m} \in O(1)$ *times slower than the optimal route.*

Proof. If there are no link failures, an alarm m initiated by node u on level $l(u)$ will reach the nearest sink in time $t_b = l(u)t_m$ in the best case and in time $t_w = l(u)(T_w + t_m)$ in the worst case. Thus, if the algorithm prefers parent v over w because $\tau(v, t_0) = T < \tau(w, t_0) = T + \varepsilon$ we get a worst case ratio of

$$c = \frac{T + (l(u) - 1)(T_w + t_m)}{T + \varepsilon + (l(u) - 1)t_m} \leq \frac{T_w + t_m}{t_m} = 1 + \frac{T_w}{t_m}.$$

□

11.4 Experimental Evaluation

To evaluate the proposed forwarding algorithm under real conditions we developed a prototype implementation for a sensor node platform based on the MSP 430F148 microprocessor and the Chipcon CC1000 radio transceiver and conducted a series of experiments in two different testbeds:

The first testbed was a typical office environment in which 16 sensor nodes and one sink node were deployed on one floor of an office building. The nodes were mounted on the ceiling at positions at which wired fire detectors would be placed. Due to the distributed nature of the deployment, the signal quality was degraded by walls and other obstacles, resulting in an increased chance for random packet losses.

The second testbed was a dense tabletop setting and represented an open-space scenario that can be encountered in factories and airport halls where nodes are typically in line of sight to each other and where the number of neighbors is rather high. In this setting, 32 nodes were arranged in a 8 x 4 grid and each node was assigned to a specific level (i.e., distance to the sink): Five nodes were assigned to level 1 and were allowed to directly communicate with the sink, 10 nodes were assigned to level 2, and 16 nodes to level 3.

In both settings, all nodes were wired to a central control station collecting performance data for the various tests. It was possible to trigger fire alarms at specific nodes and to trace the path and delay of the alarm message to the sink. The two main metrics that were evaluated are alarm latency and energy consumption.

11.4.1 Alarm Latency

The alarm forwarding performance of the Dwarf algorithm was evaluated by triggering a sequence of alarms on different nodes and by measuring the time until the alarms reached the sink node. In the office setting, a total of 4950 alarms was triggered on all nodes in a round robin fashion. Altogether, all but three alarm messages reached the sink, resulting in delivery rate of 99.94%. We later identified that two out of the three lost messages were dropped due to an implementation error and cannot be attributed to the alarm-forwarding algorithm. In the tabletop setting, we triggered 4609 alarms and achieved a delivery rate of 99.98%, with only one alarm being lost.

The latency distribution of the measured alarms for the office and tabletop scenario are depicted in Figure 11.2 and 11.3, respectively. We

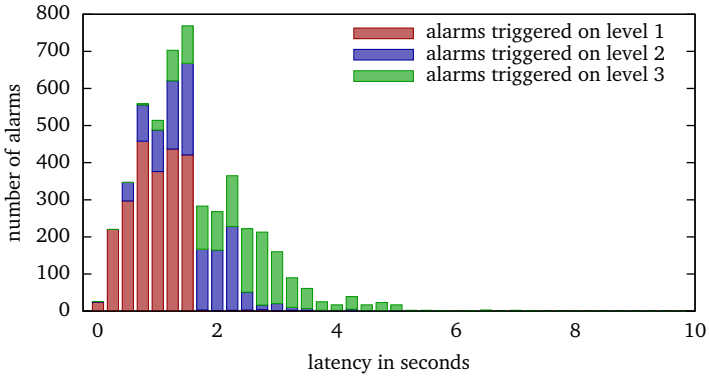

Figure 11.2: *Alarm latency for the office testbed.*

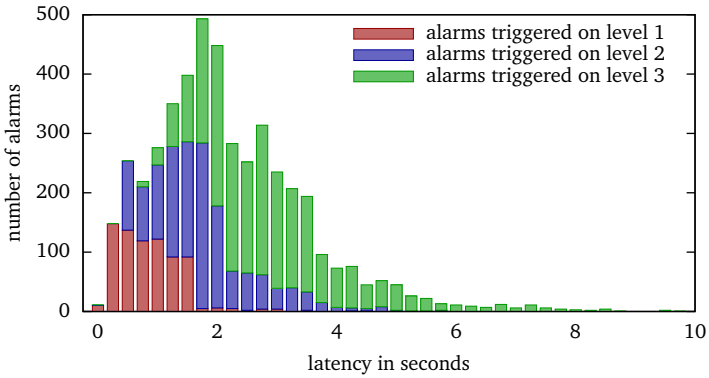

Figure 11.3: *Alarm latency for the tabletop testbed.*

observe that although there is clear correlation between higher levels and increased message latencies, no alarm was delayed beyond the 10 s limit.

11.4.2 Energy Consumption

Given the rare occurrence of alarms, energy is mainly consumed by the status monitoring and by the network initialization and maintenance. The nodes' average current consumptions in the steady state in the office and tabletop setting are depicted in Figure 11.4 and 11.5. The results show a fairly low average current consumption of at most 130 μA, which—even more important—is very well balanced.

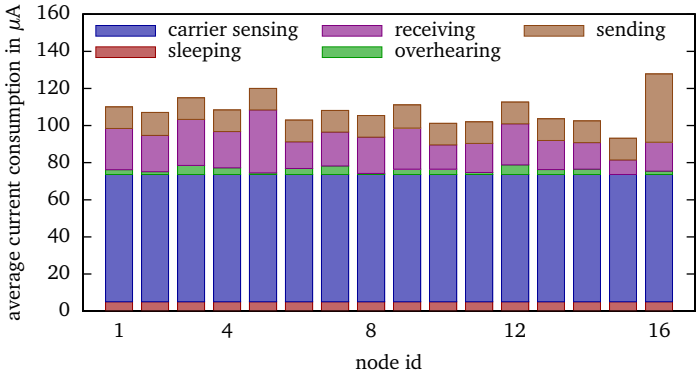

Figure 11.4: *Energy consumption without alarms for the office testbed.*

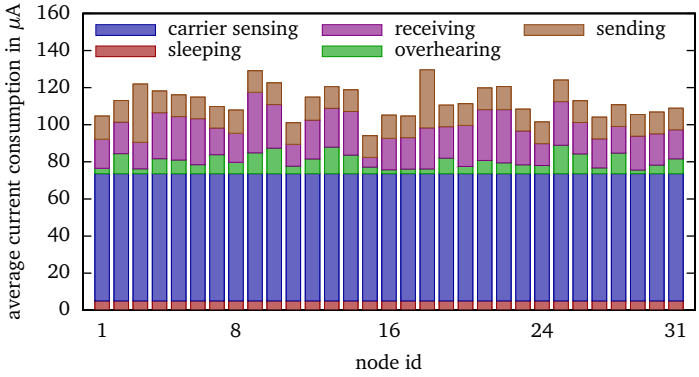

Figure 11.5: *Energy consumption without alarms for the tabletop testbed.*

In the tabletop setting, we additionally measured the energy demands for sending alarm messages. In these experiments, each node sent 42 alarm messages in a round robin fashion so that a total of 1302 alarms were forwarded. This number corresponds to approximately one alarm per day over a period of over three years and is clearly above the rate at which alarms are expected to be triggered. It is worth mentioning that all triggered alarms were received at the sink.

Figure 11.6 shows the obtained results, normalized to 1000 alarms and after the subtraction of the steady energy consumption. Most notably, the

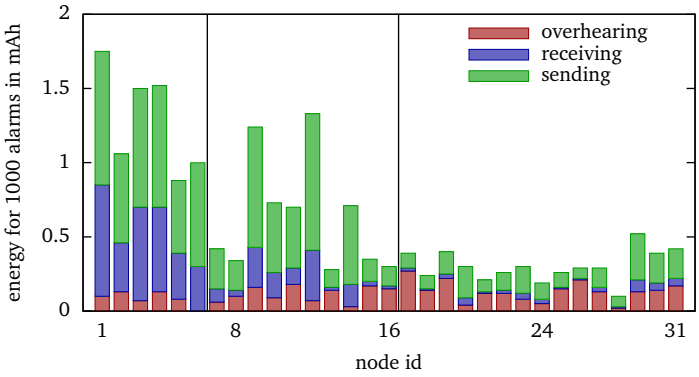

Figure 11.6: *Energy consumption in the tabletop testbed for sending 1000 alarms. The nodes are sorted according to their distance to the sink: nodes 1 to 5 are on the first level, nodes 6 to 15 on the second, and nodes 7 to 31 on the third level.*

1000 alarm messages reduced the battery capacity by at most 1.76 mAh, which is less than 0.1% of the capacity of a single alkaline AA battery.

Based on these results, a battery capacity of about 3400 mAh is required to achieve the targeted network lifetime of three years. Two alkaline batteries with a capacity of 2800 mAh each would therefore be more than enough and leave an ample surplus for the sensor readings and the network initialization.

11.5 Summary

In conclusion, we identify five main characteristics that are typical for current alarm forwarding schemes in WSNs:

1. **Energy-efficient operation of the sensor nodes.** For economic reasons, batteries should typically not be replaced more often than once every two to three years. The energy consumption of the nodes should additionally be as balanced as possible, for usually all batteries are replaced as soon as the first node runs out of energy.

2. **Nodes duty-cycle their radio.** To minimize the energy consumption of the radio transceiver, the nodes turn their radio only on when necessary (i.e., if they want to send a message or expect one).

3. **Reactive alarm notification.** To minimize communication, messages are only transmitted if a specific event is detected. With respect to alarm messages, this usually means that “no news is good news”.

4. **Narrow band, single channel receivers.** Due to existing regulations and economic considerations, wireless alarm systems use narrow band, single channel radio transceivers. Even if multiple channels are supported, the nodes change them very infrequently and never during the transmission of a message.

5. **Focus on random, non-malicious errors**. The employed techniques to ensure fault-tolerance and robustness such as multicast, forward error correction, and automatic repeat-request are parametrized for random, non-malicious errors.

In the next chapter, we will show that these characteristics make current alarm forwarding schemes highly vulnerable to (targeted) jamming attacks and discuss possible countermeasures and mitigation strategies.

Chapter 12
Impact of Reactive Jamming and Mitigation Strategies

Before we present our solution for the detection of reactive jamming, we first want to highlight the importance of such a detection scheme by demonstrating how a reactive jammer can block communication in current wireless sensor networks with minimal exposure. We additionally present two techniques based on forward error correction and limited node wiring, respectively, that help to alleviate the impact of jamming attacks but do not suffice to adequately counter them in practice.

12.1 System and Attacker Model

The scenario that we consider in the remainder of Part II is a generalization of the indoor fire alarm detection system presented in Chapter 11: A wireless sensor network system is deployed in area $\mathcal{A}$ for the surveillance of this area and the infrastructure therein. The main purpose of the network is to, upon the detection of an exceptional event (e.g., presence of an intruder), raise an alarm and forward it to the network authority. The network behavior is reactive, that is, alarms are sent when an (exceptional) event is sensed, and they are resent if they are not acknowledged by the intended receivers. We assume that the node deployment is dense enough to ensure that alarm messages reach several neighbors and that, for security reasons, all traffic is encrypted and authenticated.

In this system, the attacker's goal is to interrupt or delay the alarm notification process by means of jamming. We assume that the attacker is in control of one/several static/mobile jamming devices but is unable to destroy or deactivate nodes without being noticed (e.g., tamper-responsive packaging triggers alarm upon misuse); otherwise she could simply disable all nodes. To achieve her goal, the attacker can either proactively jam the intrusion area (Figure 12.1(b)) or reactively jam an alarm message once it is sent (Figure 12.1(c) and (d)). More specifically, we consider an attacker J that can freely choose its jamming location, frequency, rate, and strategy. We further assume that the maximal transmission power P_J of the attacker is finite, but we do not impose any restrictions on the attacker's energy supply. At each point in time, the attacker can freely choose the power P_j and beam width θ_j for a set $\{(P_j^1, \theta_j^1), (P_j^2, \theta_j^2), \ldots, (P_j^k, \theta_j^k)\}$ of

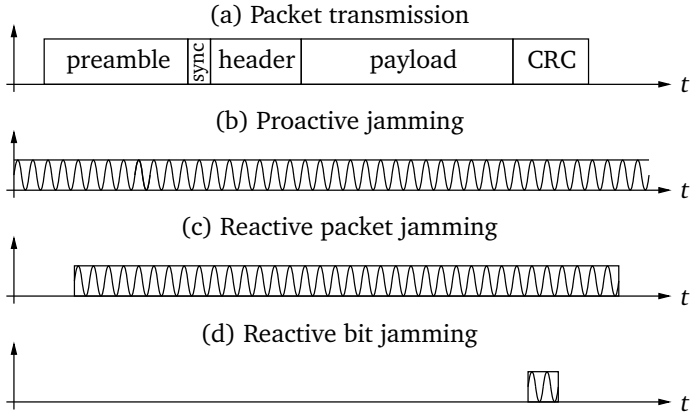

Figure 12.1: *Jamming types. (b) Proactive jammers keep the channel permanently occupied so that no transmissions are possible, whereas reactive jammers only jam once an ongoing transmission has been detected. (c) With reactive packet jamming, the attacker emits the jamming signal as soon as the transmission is detected and typically jams for an entire packet length. (d) With reactive bit jamming, the attacker targets its jamming signal at a specific part of the packet and keeps the jamming duration to a minimum.*

emitted jamming signals, provided that $\frac{1}{2\pi}\sum_{i=1}^{k} P_j^i \theta_j^i \leq P_J$. The jammer can either be proactive or reactive [60, 83]: Proactive jammers do not sense for ongoing transmissions but jam the channel permanently, whereas reactive jammers initially solely sense for ongoing transmissions and start jamming only when a packet transfer has been detected. In order to remain undetected for as long as possible, the attacker might decide to jam only a certain fraction λ_j of all packets, to vary the beam direction and width, and/or to move between individual attacks.

12.2 Impact of Reactive Jamming

The reason for the weak jamming resistance of current MAC protocols for WSNs roots in the their dependency on a preamble/sync-byte header to mark the start of the packet header [45]. This dependency makes the protocols extremely vulnerable to bit errors in the preamble, sync-byte, or packet header.

To demonstrate this vulnerability, we investigated the impact of reactive bit jamming on the performance of the above introduced alarm forwarding scheme (or, more precisely, on the WiseMAC and B-MAC protocols). The experiments were conducted with BTnode sensor nodes that use an Atmel ATmega 128L microcontroller running at 8 MHz and a Chipcon CC1000 radio [1], the preamble and header length were set to 96 and 8 bytes, respectively. In order to enable the jamming of single bits without having to use expensive hardware, the transmission rate of the sender and receiver was reduced to 2.4 kBaud. We then implemented the jammer using an additional BTnode sending random data at a rate of 38.4 kBaud.

Our results clearly show that reactive bit jamming can efficiently block communication that uses current sensor network protocols with very low exposure for the attacker (i.e., a very short time during which the attacker has to send a jamming signal). Specifically, in our experiments, we were able to achieve a jamming success rate of 90% by jamming only three bits in the packet header or in the sync byte (see Figure 12.2). Even more importantly, if the jamming was targeted at the sync-byte, the jammed packet transmissions were not even recognized as such by the network stack and were thus completely ignored by the nodes and not counted as packet losses.

As a first step towards a better jamming resistance, we therefore introduce a more robust packet header detection technique that significantly increases the minimal duration during which the jammer must interfere with a packet to block it. In addition, we discuss a jamming mitigation technique that harnesses limited node wiring in the form of wired node chains to forward messages out of the jammed region. We show that even though both schemes help to diminish the impact of the jammer, they cannot entirely counter jamming attacks. In the absence of (broadband) anti-jamming communication, an effective jamming detection thus remains an essential countermeasure against such attacks.

12.3 Robust Packet Detection

With our header detection technique, before a packet is transmitted, the sender applies error correcting codes to the header and shuffles the encoded bits according to a pseudo random sequence based on a secret key shared by the sender and the receiver. As we shall see, this process ensures that a substantial part of the packet header must be jammed to prevent being decoded. Note that otherwise the MAC protocol in use is not modified; in particular a possibly required preamble—for example to account for

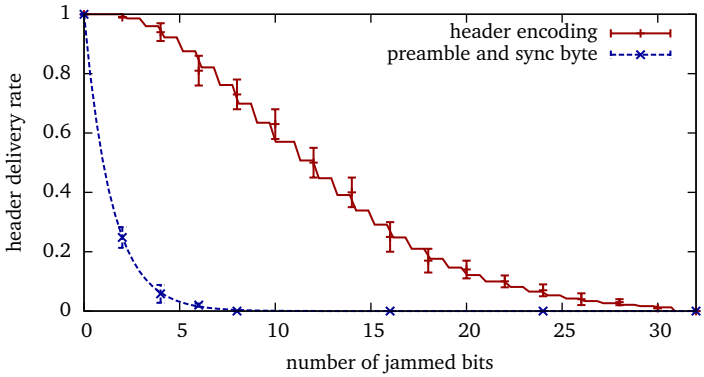

Figure 12.2: *Header delivery rate for our coding-based header detection algorithm and a common preamble/sync-byte-based approach. The lines show the expected theoretical results, the points the measured results; for each transmission the bit shuffling was based on a new (secret) seed. According to these results, a jammer that wants to mask a packet transmission with a probability of* $> 90\%$ *would have to jam only 3 bits if the common preamble/sync-byte method is used and more than 21 bits if our coding-based detection is used. Hence, the coding-based header detection is not only much more robust than common approaches, but also facilitates the detection of a jammer as it enforces a longer jamming duration.*

imprecision in the nodes synchronization or to announce a transmission if low power listening is used—is still transmitted.

The receiver (periodically) samples the channel according to the schedule of the employed MAC protocol and uses the received signal strength to assess whether a transmission is taking place. If a transmission is detected, the sender starts receiving it. However, as opposed to common practices, the sender will not wait for a predefined sync-byte to mark the start of the packet but will try do decode the header itself. Specifically, the receiver will receive a complete header length, unscramble the data, and try to decode it. Upon success, he receives the remainder of the packet; otherwise, he drops the first (i.e., oldest) bit, appends the newest received bit, and tries to decode the new input. This process is repeated until a packet is detected or the transmission ends (i.e., the received signal strength drops to the noise level for some time).

The main drawback of this packet detection algorithm is that the error correcting codes increase the header length and thus the required energy

for a packet transmission. Whether this increase has a noticeable impact on the overall energy consumption of the sender and/or receiver depends on the application and the MAC protocol in use. The well-known B-MAC protocol, for instance, uses preamble lengths which are several times longer than the packet header [62]. A coding rate of up to 0.5 (i.e., an encoded header length that is twice as long as the unencoded header) has thus only a small impact on the duty-cycle and can usually be neglected.

We evaluated the performance of our proposed header detection scheme experimentally with the aforementioned setup based on BTnode sensor nodes. For the header encoding we used a Hamming (8,4) code that allows for correcting single bit errors, detecting all two bit errors, and detecting some three bit errors; the bit shuffling was performed with a linear feedback shift register.

The efficiency of our coding-based header detection algorithm compared to a preamble/sync-byte-based approach is shown in Figure 12.2. Assuming a flipping probability of 0.5 for the jammed bits and a jamming duration of x bits, in theory, the expected header delivery rate is 2^{-x} for the preamble/sync-byte method and about $\sum_{i=0}^{x} \binom{x}{i} 2^{-x} (1 - \frac{8-1}{8 \cdot 16})^{\binom{i}{2}}$ for the coding-based approach. This has been confirmed by our experiments: A jammer that wants to block a packet transmission with a probability of $> 90\%$ has to jam only 3 bits if the common preamble/sync-byte method is used but more than 21 bits if our coding-based detection is used. The results thus show that the coding-based header detection is not only much more robust than a preamble/sync-byte-based detection, but also increases the detection probability of a potential jammer as it enforces a longer jamming duration. However, as we shall see in Chapter 13, even such a significantly longer jamming duration does not suffice to recognize jamming with current jamming detection schemes.

12.4 Limited Node Wiring

Limited node wiring as an alternative jamming mitigation technique in sensor networks was initially proposed by Čagalj et al. [77]. In their solution, a regular wireless sensor network is augmented with a number of wired node pairs. The purpose of these wired pairs is to establish a jamming-resistant link from a jammed to an unjammed node in order to forward alarm messages out of a jammed region (see Figure 12.3(a)). The authors analyze the probability that such a forwarding link exists (i.e., that at least for one wired node pair the first node is inside the jammed area while the second node is outside) as a function of the wire length

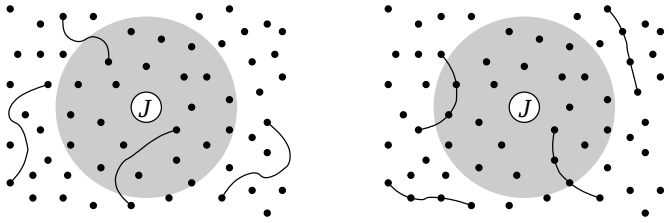

Figure 12.3: *Limited node wiring (a) in the form of node pairs and (b) its generalization to wired node chains (n-tuples).*

and the total number of wired pairs, but do not further evaluate whether the therefore required wiring is feasible and ignore possible attacks on the wires. In this section, we generalize the proposed pair-wise wiring to limited (short-range) node wiring in the form of wired node chains (n-tuples) that link $n \geq 2$ nodes (see Figure 12.3(b)). We present an analytical model for the random, manual, and airdrop-based deployment of such n-tuples and evaluate the forwarding performance of the resulting hybrid network analytically and with simulations. In this evaluation, we focus on jamming attacks and assume that the wires in the n-tuples are protected against attackers that have physical access to the network by means of a low-power wire compromise detection protocol [68]; typical physical attacks that must be detected by this protocol are disconnection (node unplugging or wire cutting) and bridging (insertion of a rogue node between two wired nodes).

12.4.1 Deployment of Pre-wired n-tuples

One of the objections that could be risen against the proposed limited sensor wiring is the seemingly difficult node deployment. In this section, we argue that the introduced wired n-tuples can be efficiently deployed in a number of scenarios, and we present an analytical model for the three most popular deployment techniques: random, manual, and airdrop-based deployment.

For our purposes (i.e., to escape the jammed region), a desirable property of an n-tuple is that its wired nodes do not cluster at a single point, but are spread over a large area, preferably in a straight line (the deployment direction ϕ). In order to see to what extent this can be achieved, we conducted some rudimentary experiments using a dummy 3-tuple of

20 m length. Not surprisingly, our experiments showed that in a manual deployment, arranging the nodes in a (almost) straight line can easily be achieved by unrolling the n-tuple (e.g., from a cable reel carried by a person or mounted on a vehicle). For simulating airdrop-based deployment, we dragged the n-tuple through the air and then released it from a small ropeway at a low altitude ($\sim$3 m). We observed that the expected length of the deployed n-tuple was about 16-19 m and that its nodes deviated from the straight line less than 2 m; this deviation depends of course on the rigidity and length of the wires. We admit that these findings are not conclusive and that additional, more extensive studies are required in order to obtain statistically significant results. However, they illustrate that pre-wired n-tuples can be deployed in a low-cost manner and with an effort that is comparable to the one needed for the (manual) deployment of wireless sensor nodes.

12.4.2 Deployment Model

Let $u_i := (u_{i,1}, u_{i,2}, \ldots, u_{i,n})$ denote a n-tuple that is deployed in the deployment area $\mathcal{A}$. The order of the nodes in a tuple also determines their wiring: that is, for a n-tuple u_i and $1 \leq j < n$, node $u_{i,j}$ is connected to node $u_{i,j+1}$. For simplicity, we assume that all nodes are connected with wires of the same length l_w. The position of a node $u_{i,j}$ in the deployment area is denoted by $q_{i,j} \in \mathcal{A}$.

Given the deployment direction ϕ and the position $q_{i,1}$ of the first node of the n-tuple u_i, the deployment of the remaining nodes $u_{i,2}$ to $u_{i,n}$ can be modeled as follows: For each node $u_{i,j}$, $2 \leq j \leq n$, imagine a disc $\mathcal{D}_{j-1} \subset \mathcal{A}$ centered at $u_{i,j-1}$ and of radius l_w. The position of $u_{i,j}$ is then chosen from $\mathcal{D}_{j-1}$ according to a random distribution defined by the (conditional) probability density function $f_D(q_{i,j}|\phi, q_{i,j-1})$. More formal, let $r_{i,j}$ be the euclidean distance between node $u_{i,j-1}$ and $u_{i,j}$, and $\alpha_{i,j}$ be the deviation of $u_{i,j}$'s position with respect to the deployment direction ϕ. To each n-tuple $u_i = (u_{i,1}, u_{i,2}, \ldots, u_{i,n})$ we can then associate a $(2n-1)$-dimensional (continuous) random variable $Q_i := (Q_{i,1}, R_{i,2}, \Lambda_{i,2}, \ldots, R_{i,n}, \Lambda_{i,n})$ taking values from the set $\{(q_{i,1}, r_{i,2}, \alpha_{i,2}, \ldots, r_{i,n}, \alpha_{i,n}) | q_{i,1} \in \mathcal{A} \wedge \forall j, 2 \leq j \leq n : 0 < r_{i,j} \leq l_w \wedge \forall j, 2 \leq j < n : -\pi \leq \alpha_{i,j} \leq \pi\}$ according to a random distribution defined by the (joint) probability density function $f_i(q_{i,1}, r_{i,2}, \alpha_{i,2}, \ldots, r_{i,n}, \alpha_{i,n}) = f_Q(q_1) f_\Phi(\phi) f_{R,\Lambda}(r_2, \alpha_2) f_{R,\Lambda}(r_3, \alpha_3) \cdots$ $f_{R,\Lambda}(r_n, \alpha_n)$. Here, $f_Q(q_1)$ and $f_\Phi(\phi)$ represent the distributions on $\mathcal{A}$ and the interval $[0, 2\pi)$, respectively, and $f_{R,\Lambda}(r_j, \alpha_j)$ is the (joint) probability density function for the distance $r_{i,j}$ between node $u_{i,j-1}$ and node $u_{i,j}$ as well as the deviation α_j of $u_{i,j}$ from the deployment direction ϕ.

The probability density function $f_{R,\Lambda}(r_j, \alpha_j)$ reflects the actual deployment conditions and depends on the kind of deployment (random, manual, or airdrop-based) and on several physical parameters (e.g., the local terrain conditions or the rigidity of the wires). It can be approximated with appropriately parametrized (two-dimensional) Beta distributions (scaled to the interval $[0, l_w]$ and $[-\pi, +\pi]$). If $f_Q(\cdot)$, $f_\Phi(\cdot)$, and $f_{R,\Lambda}(\cdot)$ represent uniform distributions on $\mathcal{A}$, the interval $[0, 2\pi)$, and a disc of radius l_w, respectively, the resulting deployment is truly random. In any case, once the probability density functions are determined, the deployment of any set of n-tuples $\{u_1, u_2, \ldots, u_m\}$ can be formally described by the set of random variables $\{Q_1, Q_2, \ldots, Q_m\}$.

Simulation Parameters

In our simulations, the deployment area $\mathcal{A}$ is a square with a side length of $a = 500\,\text{m}$. The position of the nodes in the tuples is chosen according to the probability density function $f_{R,\Lambda}(r_j, \alpha_j) = f_R(r) f_\Lambda(\alpha)$, where $f_R(r) = B(a_r, b_r)^{-1} (\frac{r}{l_w})^{a_r - 1} (1 - (\frac{r}{l_w}))^{b_r - 1}$ and $f_\Lambda(\alpha) = B(a_\alpha, b_\alpha)^{-1} (\frac{\alpha + \pi}{2\pi})^{a_\alpha - 1} (1 - (\frac{\alpha + \pi}{2\pi}))^{b_\alpha - 1}$ are two beta distributions and $B(a, b) = \int_0^1 t^{a-1} (1-t)^{b-1}\,\mathrm{d}t$. Based on our (admittedly limited) experimental results, we chose the parameters $a_r = 10$, $b_r = 1.76$ and $a_\alpha = b_\alpha = 124$. For a given wire length l_w, this results in an expected n-tuple length of $0.85(n-1)l_w$, $\sigma \approx 0.1(n-1)l_w$ and an expected deviation of $\alpha = 0$, $\sigma \approx 0.14\pi$.

12.4.3 Forwarding Performance

Once an alarm has escaped the jammed area, any (alarm) forwarding scheme (which, in turn, might again leverage on the wired links) can be used to inform the sink about the jamming. In this analysis, we therefore focus on computing the probability that there exists an n-tuple that connects the jammed with the non-jammed area. We first consider the case of an omnidirectional jammer and afterwards generalize this result for an arbitrary jammer. For simplicity, we assume that all nodes of an n-tuple lie on a straight line and that any two consecutive nodes in a tuple have the same distance.

Let X denote the event that at least one node of an n-tuple lies inside a disc of radius r that is centered at the jammer, and by Y the event that at least one node of an n-tuple lies outside the disc. Let further l be the expected length of a tuple and z be the distance between the first node of the n-tuple and the center of the disc. The probability that one node of a

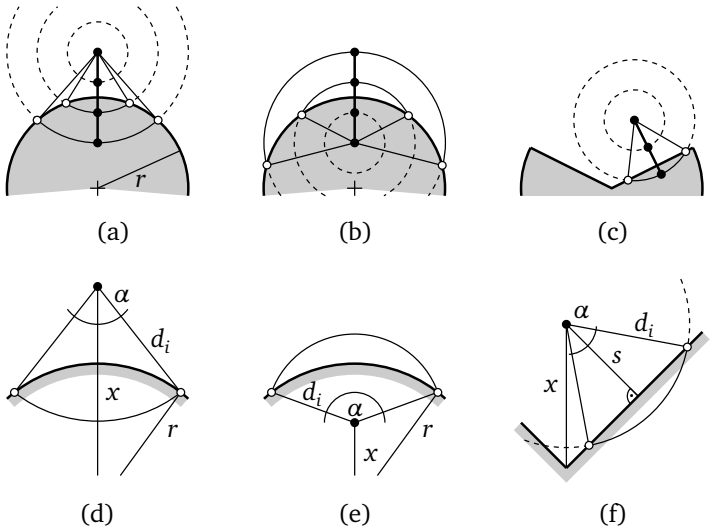

Figure 12.4: *Geometric relations between the position/orientation of an n-tuple and its possible intersections with a disc or a circular sector of radius r; the figures are not true to scale. Drawings (a) and (d) show the situation where the i-th node of the n-tuple lies inside a disc, drawings (b) and (e) where it lies outside. Drawings (c) and (f) show the situation where the n-tuple crosses a radius of a circular sector.*

tuple lies inside the disc while another one lies outside is then

$$p_{XY}(r,l) = \int_{x=0}^{\infty} \mathsf{P}[X \wedge Y | z = x]\mathsf{P}[z = x] \tag{12.1}$$

$$= \int_{x=0}^{r} \mathsf{P}[Y|z = x]\frac{2x\pi \mathrm{d}x}{|\mathcal{A}|} + \int_{x=r}^{r+l} \mathsf{P}[X|z = x]\frac{2x\pi \mathrm{d}x}{|\mathcal{A}|}\,.$$

The probabilities $\mathsf{P}[Y|z = x]$ and $\mathsf{P}[Y|z = x]$ can be computed as follows: Let $d_i = \frac{i-1}{n-1}l$ be the distance between the first and the i-th node in the n-tuple. Now imagine a circle of radius d_i centered at the first node in the tuple. Given that the direction of a tuple is chosen uniformly at random, the probability that the i-th node lies within the disc is proportional to the central angle subtended by the two intersection points of this circle

with the perimeter of the disc to the first node (see Figure 12.4(a)). As illustrated in Figure 12.4(d), this angle is

$$\alpha_X(d_i,x) := \begin{cases} 0 & \text{if } x<0 \text{ or } x>r+d_i \text{ or } x+r<d_i, \\ 2\pi & \text{if } 0\le x+d_i\le r, \\ 2\arccos\left(\frac{x^2+d_i^2-r^2}{2d_i x}\right) & \text{otherwise.} \end{cases} \tag{12.2}$$

Likewise, the probability that the i-th node lies outside of the disc is proportional to the central angle subtended by the two intersection points of the circle with the perimeter of the disc to the first node (see Figure 12.4(b)). According to Figure 12.4(e), this angle is

$$\alpha_Y(d_i,x) := \begin{cases} 0 & \text{if } x<0 \text{ or } d_i+x<r, \\ 2\pi & \text{if } x>r+d_i \text{ or } x+r>d_i, \\ 2\pi-2\arccos\left(\frac{x^2+d_i^2-r^2}{2d_i x}\right) & \text{otherwise.} \end{cases} \tag{12.3}$$

Hence, we obtain

$$\mathrm{P}[X|z=x] = \max_{1\le i<n} \alpha_X\left(\tfrac{i}{n-1}l,x\right)\frac{1}{2\pi}, \tag{12.4}$$

$$\mathrm{P}[Y|z=x] = \max_{1\le i<n} \alpha_Y\left(\tfrac{i}{n-1}l,x\right)\frac{1}{2\pi}, \tag{12.5}$$

and thus

$$p_{XY}(r,l) = \int_{x=0}^{r} \frac{\max_{1\le i<n} \alpha_Y\left(\frac{i}{n-1}l,x\right)}{2\pi}\frac{2x\pi \mathrm{d}x}{|\mathcal{A}|} + \int_{x=r}^{r+l} \frac{\max_{1\le i<n} \alpha_X\left(\frac{i}{n-1}l,x\right)}{2\pi}\frac{2x\pi \mathrm{d}x}{|\mathcal{A}|}. \tag{12.6}$$

For the general case, in which the attacker does not emit a single omnidirectional jamming signal but several directed signals specified by the set $\{(\theta_j^1,P_j^1),(\theta_j^2,P_j^2),\ldots,(\theta_j^k,P_j^k)\}$ (see Section 12.1), we also have to consider the possibility that a tuple does not cross the arc but the two radii of the circular sector (Figure 12.4(c)). Let

$$s(x) := \min(l_w, x\sin(\min(2\pi-\theta_j^i,\pi/2))) \tag{12.7}$$

be the (upper bounded) distance from the first node in the tuple to a radius of the circular sector and

$$\alpha_{XY}(d_i, y) := \begin{cases} 0 & \text{if } y < 0 \text{ or } y > d_i, \\ 2\arccos(y/d_i) & \text{otherwise} \end{cases} \tag{12.8}$$

be the central angle subtended by the two intersection points of the circle centered at the first node of the n-tuple with this radius to the first node (see Figure 12.4(f)). The probability that an n-tuple crosses the radii of a circular sector can then be approximated as

$$p'_{XY}(r,l) \gtrapprox 2 \int_{x=0}^{r} \int_{y=0}^{s(x)} \frac{\max\limits_{1\le i<n} \alpha_{XY}\left(\frac{i}{n-1}l, y\right)}{2\pi} \frac{\mathrm{d}y\mathrm{d}x}{|\mathcal{A}|} . \tag{12.9}$$

For a general attacker, the probability that there exists at least one n-tuple that connects the jammed with the non-jammed area can thus be lower bounded by

$$p_l \gtrapprox 1 - \prod_{i=1}^{k} \left(1 - \left(1 - \frac{\theta_j^i}{2\pi} p_{XY}(R(P_j^i), l) - p'_{XY}(R(P_j^i), l)\right)^m\right) , \tag{12.10}$$

where m is the number of deployed n-tuples and $R(P_j^1)$ is the radius of the jammed area.

The influence of the number of nodes n per tuple, the wire length l_w, the number of deployed tuples m, and of the size of the jammed area (i.e., $R(P_j^i)$ and θ_j^i) on the probability p_l that there exists a wired link out of the jammed area is depicted in Figure 12.5. We observe that longer n-tuples can significantly increase this probability, whereas even very long tuples cannot benefit from more than three wired nodes per tuple. As the probability p_l is proportional to the perimeter of the jammed area, it increases approximately linear with the jamming range and beam width. However, our results also show that for a practical tuple length of 20 m, over 400 n-tuples must be deployed per km^2 to ensure that a message can be forwarded out of the jammed region in at least 95% of the cases. This rather high number clearly limits the application of limited wiring as a means to mitigate jamming attacks.

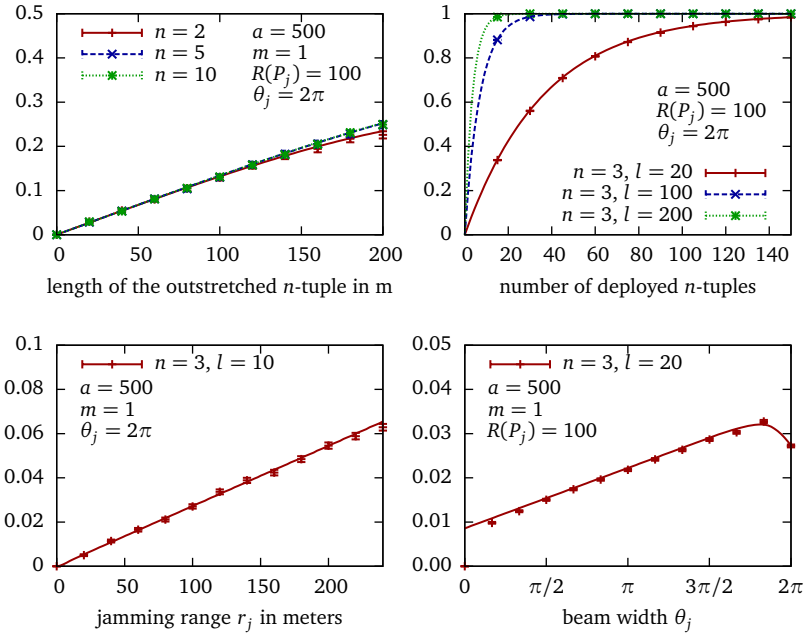

Figure 12.5: *Probability p_l that there exists a wired link out of the jammed area. The lines show the expected result according to our analysis, the points and confidence intervals the simulation results. We observe that even very long tuples with a length of over hundred meters cannot benefit from more than three wired nodes per tuple. Increasing the length of a tuple, on the other hand, results in a linear increase of p_l. As the probability p_l is proportional to the perimeter of the jammed area, it increases approximately linear with the jamming range and beam width; the offset for 0 and 2π is due to the change from a closed circle to a circular sector.*

Chapter 13
Detection of Reactive Jamming

Traditional approaches for the detection of jamming in wireless sensor networks use the packet-delivery-ratio (PDR) and the received ambient signal strength as the main decision criteria. Jamming is detected as soon as the (averaged) PDR and/or the ambient signal strength exceeds a pre-defined threshold (see Chapter 15). Although these approaches are well-suited for the detection of proactive (long-term) jamming, they are not sufficient to protect the considered applications against targeted reactive jamming. To begin with, existing schemes rely only on the CRC of a packet to decide whether it was received correctly and thus can (in general) not distinguish between packet failures due to weak radio links and interference. Furthermore, assessing an accurate PDR is not practical in a reactive forwarding scheme as messages are sent very rarely. Finally, jamming does not necessarily cause a steady and high received signal strength (RSS) value, as only a small fraction of a packet has to be interfered with in order for the packet to be invalid [5,57,60]. A (reactive) jammer can thus keep the increase in the effective RSS value very low and can hence avoid being detected with current approaches.

Our novel jamming detection scheme does not suffer from these limitations. The central idea of our approach is to identify the cause of individual bit errors within a packet and to deduce therefrom whether the packet was jammed or just sent over a weak link. This is achieved as follows: Whenever a node receives a packet transmission, it not only receives the packet, but also records the RSS for each received bit of the packet[1]. Given a bit error, a node then deduces the root cause of this error by looking at the respective RSS value that was sampled during the reception of this bit. The intuition behind this process is that if there was a bit error although the RSS value was high, this indicates external interference (intentional or unintentional); if the error was due to a weak signal (e.g., due to fast fading or shadowing), the RSS value should be low. This additional information allows an accurate differentiation of packet errors that are caused by (un)intentional interference from errors that are caused by weak links,

[1] Some radios do not provide this accuracy but compute the averaged RSS value over a sequence of k bits (e.g. one byte). In these cases the algorithm as described might not detected jamming that affects less than k bits. This issue and possible mitigation strategies are further discussed in Chapter 14.

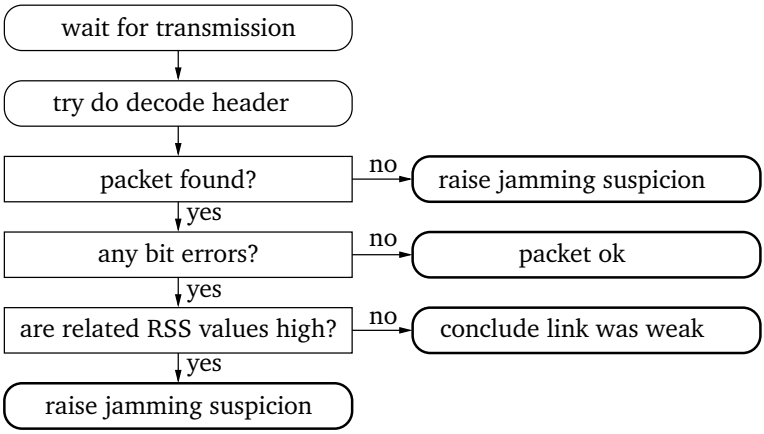

Figure 13.1: *Packet reception and jamming detection. Once a transmission signal is detected, the receiver tries to decode the presumed packet header. If it fails (i.e., if there is no transmission although the channel is busy) this can indicate proactive jamming, and the sequential jamming test is updated. If the received packet contains bit errors, the root cause of these errors is analyzed and either the jamming test updated or the packet ignored.*

even in the case of a sophisticated (reactive) attacker that jams only a small portion of a packet.

Our jamming detection algorithm comprises three steps (see Algorithm 2): (A) error sample acquisition (i.e., packet reception with RSS recording and identification of bit errors), (B) interference detection (i.e., error cause analysis), and (C) sequential jamming test. The function of the last step is to decide whether a detected interference was malicious or due to an unintentional packet collision and is only required if the probability of such a collision cannot be neglected. Next, we describe each of these steps in detail; the overall packet reception and jamming detection process that also considers proactive jamming is outlined in Figure 13.1.

13.1 Error Sample Acquisition

Whenever a node receives a packet transmission by radio, it receives the packet (even if it is not the intended receiver) and also records the RSS value for each received bit of the packet. That is, a node associates to each packet m a sequence s of RSS values corresponding to packet bits. Hence,

Algorithm 2 Jamming detection algorithm

```
 1: resetJammingTest()
 2: while true do
 3:     (e,s) := getErrorSample()                    (A)
 4:     x := detectInterference(e, s)                (B)
 5:     y := updateJammingTest(x)                    (C)
 6:     if y = jamming then
 7:         raise jamming suspicion
 8:     else if y = no jamming then
 9:         resetJammingTest()
10:     else
11:         do nothing as we need more evidence
12:     end if
13: end while
```

for each bit in a packet the RSS at the time of its reception is also known; we denote by $m[i]$ and $s[i]$ the i-th bit in the packet m and the i-th RSS value in the RSS sequence s, respectively.

The next and generally more challenging task is the identification of bit errors. In the simplest case, the content of m is predetermined (at least to a large extent) and known by the receiver. Finding bit errors thus reduces to comparing m with the reference packet $\hat{m}$. More formally, the error vector e can be computed as $e[i] := m[i] \oplus \hat{m}[i]$, where $\oplus$ is the exclusive or and $e[i] = 1$ if and only if the i-th bit is false. This error vector and the RSS sequence are combined to an error sample (e,s) which is then used as the base for the interference detection. The main drawback of this approach is that, because the packet content must be known to the receiver, the information conveyed by the packet is virtually limited to at best a few bits.

This limitation can be overcome by means of error detecting/correcting codes. These codes allow for detecting or even correcting bit errors in arbitrary messages. Where the original data can be recovered, the bit errors can be precisely detected by comparing the received and recovered data. If, however, a code word can be identified as being faulty but cannot be corrected, all bits in the word might equally be wrong and are thus marked as false. In addition to this possible loss of precision, the second drawback of using error correcting codes is the overhead that the codes introduce. Depending on the strength of the code, the packet length (and thus the energy required for its transmission) might be several times higher than the length of the original packet.

A third, more elaborate way to acquire error samples is based on limited, short-range sensor node wiring in the form of wired node chains (n-tuples), as introduced in Section 12.4. This method leverages the link redundancy (wired/wireless) provided by these n-tuples. In what follows, we assume for simplicity that errors on the wired links can be neglected or are corrected (e.g., by forward error correction). If two nodes of a n-tuple are in the transmission range of a sending node, they will both receive the same packet transmission and record the corresponding RSS values. Two such independently received packet/RSS-sequence pairs (m_1, s_1) and (m_2, s_2) from the same packet transmission are then combined into an error sample (e, s), where $e[i] := m_1[i] \oplus m_2[i]$ and $s[i] := \min\{s_1[i], s_2[i]\}$. In general, error samples can be obtained in two ways:

In active monitoring, a node in an n-tuple sends a packet first over the wired and then over the wireless channel. The other nodes in the tuple receive both packets and record the RSS values of the packet received by radio; the RSS values of the (faultless) packet received by wire are set to infinity. Note that here the nodes in the tuple know when a packet is being transmitted and thus can still try to receive and compare its (payload) data, even if they fail to decode (part of) the header or payload. In passive monitoring, whenever a node in an n-tuple receives a packet that does not originate from a node in the tuple and that has not yet been received by wire, the node broadcasts the packet over the wire together with its respective RSS value sequence to all the nodes in the tuple. Each node in the n-tuple that receives a packet over the wired and over the wireless channel can then combine them to an error sample. Note that since the algorithm does not make any assumptions regarding the content of the packets, any regular application packet can be used to form a sample.

Being able to work with passive and active monitoring, our scheme allows to trade off n-tuple deployment density against energy consumption: In a passive system where the n-tuples do not introduce any additional network traffic, at least two nodes of an n-tuple must be in the transmission range of the sending node to detect jamming. In a (partially) active system where (some of) the wired nodes periodically exchange probe messages[2], only one node of such an n-tuple must be included in the jammed region. Moreover, as opposed to existing solutions, signal overshadowing or cases where the packet transmission is not (or only partially) recognized by the receiver's radio can also be detected.

[2] Since all traffic is encrypted, the attacker cannot distinguish probe from regular messages.

Algorithm 3 Error Sample Acquisition

```
1:  function GETERRORSAMPLE()
2:      while true do
3:          receive (m_1, s_1) by wire
4:          if the related packet (m_2, s_2) has already been received then
5:              ∀i : e[i] := m_1[i] ⊕ m_2[i]
6:              ∀i : s[i] := min{s_1[i], s_2[i]}
7:              return (e, s)
8:          else if neighbor in the tuple will send it next then
9:              receive m_2 by radio and record RSS into s_2
10:             ∀i : e[i] := m_1[i] ⊕ m_2[i]
11:             ∀i : s[i] := min{s_1[i], s_2[i]}
12:             return (e, s)
13:         end if
14:     end while
15: end function
```

13.2 Interference Detection

If a received packet contains at least one bit error, a node uses the measured RSS values to decide whether the identified errors are due to interference or due to a weak signal. If there was a bit error although the respective RSS value was high (i.e., although the link appeared to be strong), we conclude that the error must have been caused by external (intentional or unintentional) interference. If, on the other hand, the respective RSS value was low, we conclude that the error was most likely due to a low signal-to-noise ratio.

Here, we present a simple threshold-based mechanism to decide whether a packet error is due to interference. Let q be the counter for the number of recently observed packet errors due to interference. For each bit error in a packet, the respective RSS value is compared with a threshold S. If for at least one such case the RSS value is above the threshold S, q is increased, otherwise it is left unchanged. More formally, given an error sample (e,s), if $\exists i : e[i] = 1 \wedge s[i] > S$ then $q := q + 1$. The choice of (an optimal) S depends on the used radio and modulation scheme; it can be predefined (e.g., as the result of experiments) or be computed on-the-fly (e.g., as a function of the RSS values of correctly received bits). If only links of poor quality are available, more sophisticated (but also more expensive) decision methods such as likelihood-ratio tests or Bayes factors can also be used [12]. In

our experiments we achieved good results by adaptively changing S to the average signal strength of the last 10 successfully received packets.

Algorithm 4 Interference Detection

```
1: function DETECTINTERFERENCE(e, s)
2:     if ∃i : e[i] = 1 and s[i] > S then
3:         return 1
4:     else
5:         return 0
6:     end if
7: end function
```

13.3 Sequential Jamming Test

If the probability of packet collisions can be neglected, a node rises an alarm whenever it detects bit errors that were caused by interference. Otherwise, the result of the interference detection is taken as an input to a sequential probability ratio test (SPRT) [85] which is used to decide whether the recent packet errors (if any) were due to unintentional packet collisions or due to jamming. We assume that the nodes can assess the expected local interference (which is supposed to be low if the MAC works properly), either based on their knowledge about the used MAC and neighborhood or by using more sophisticated procedures such as those proposed in [86]. Let p_c be an upper-bound on the expected collision probability, τ_{FP} (τ_{FN}) be the targeted probability for a false alarm (missed attack), and q be the number of identified packet errors that were due to interference during the last k error samples. Given the probability p that the transmission of a packet fails, the probability that q out of k transmissions fail is $\binom{k}{q}p^q(1-p)^{k-q}$. The marginal likelihood that the observed packet errors were solely due to unintentional collisions (i.e., $0 \le p \le p_c$, hypothesis H_0) is then $p_{H_0}(k) := \int_{p=0}^{p_c} \binom{k}{q}p^q(1-p)^{k-q}\,\mathrm{d}p$; the marginal likelihood that there was jamming (i.e., $p_c \le p \le 1$, hypothesis H_1) is $p_{H_1}(k) := \int_{p=p_c}^{1} \binom{k}{q}p^q(1-p)^{k-q}\,\mathrm{d}p$. Hence, the log-likelihood ratio for H_0 and H_1 after k samples is

$$\eta(k) = \log\frac{p_{H_1}(k)}{p_{H_0}(k)} = \log\frac{\int_{p=p_c}^{1} p^q(1-p)^{k-q}\,\mathrm{d}p}{\int_{p=0}^{p_c} p^q(1-p)^{k-q}\,\mathrm{d}p}. \tag{13.1}$$

Now, if $\eta(k) \leq \log \frac{\tau_{FN}}{1-\tau_{FP}}$ the nodes decide that there is no jamming and reset the sequence (i.e., set k and q to zero), if $\eta(k) \geq \log \frac{1-\tau_{FN}}{\tau_{FP}}$ jamming is detected and the nodes raise an alarm, finally if $\log \frac{\tau_{FN}}{1-\tau_{FP}} < \eta(k) < \log \frac{1-\tau_{FN}}{\tau_{FP}}$ no conclusive decision can be made yet and is deferred until there is more conclusive evidence available.

Algorithm 5 Jamming Test

```
1:  function resetJammingTest
2:      k := 0; q := 0
3:  end function
4:
5:  function updateJammingTest(x)
6:      k := k + 1; q := q + x;
7:      η(k) := SPRT(k, q)
8:      if η(k) ≥ log (1−τ_FN)/τ_FP then
9:          return jamming
10:     else if η(k) ≤ log τ_FN/(1−τ_FP) then
11:         return no jamming
12:     else
13:         return undefined
14:     end if
15: end function
```

Chapter 14
Performance Evaluation

In this chapter, we evaluate the performance of the proposed jamming detection techniques. To that end, we implemented them and conducted a series of experiments using COTS BTnodes (Atmel ATmega 128L microcontroller @ 8 MHz, Chipcon CC1000 radio) and Tmote Sky sensor nodes (TI MSP430F1611 microcontroller @ 8 MHz, Chipcon CC2420 radio). The experimental setup consisted of four nodes: one sender (node A), two receivers (node B and C), and one jammer (node J). For the wire-aided jamming detection, node B and C were connected over the I^2C bus, forming a two-tuple.

A compilation of exemplary measurements for an undisturbed, jammed, and weak link between A and C is shown in Figure 14.1. The results confirm the validity or our approach and show that decoding errors caused by jamming can clearly be distinguished from errors caused by a weak radio signal by looking at the corresponding RSS values. However, these initial experiments also revealed some hardware constraints that limit the accuracy of the proposed detection techniques: In cases where the used radios do not provide an RSS value per bit but instead provide an averaged RSS value for a set of k bits, the algorithm might not be able to detected jamming that affects less than k bits. To overcome this limitation, error correcting codes that enforce a minimal required jamming duration of $> k$ bits can be applied (see Section 12.3). Furthermore, packet based radio transceivers such as the Chipcon CC2420 typically rely on a particular synchronization preamble or training sequence to detect packet transmissions. If this preamble or training sequence is jammed, the corresponding transmission is simply ignored (the automatic CRC verification is usually not an issue as it can mostly be disabled). Simple bit or byte oriented radio transceivers such as the Chipcon CC1000 that provide a continuous data demodulation and RSS estimation are thus better suited for our purposes. Therefore, the remainder of this section focuses on bit or byte oriented radios and presents the results obtained with our implementation for the CC1000 radio (i.e., the BTnodes) only. Nevertheless, we would like to point out that our basic considerations apply in general and thus our detection techniques in principle also work with packet-based radios.

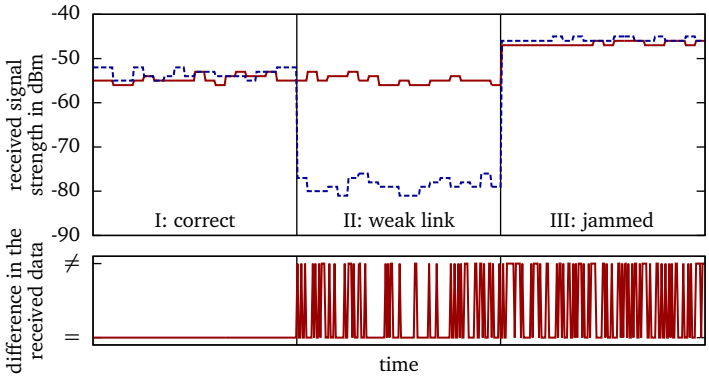

Figure 14.1: *Sample results obtained with our implementation and a CC1000 radio for three cases. I (adequate links and no jamming): both receivers are able to decode the packets and the packets do not differ. II (weak link from A to C): node C receives incorrect bits and thus the packets do not match; however, since the RSS of node C associated with the bit errors is low, the errors are correctly identified as non jamming related. III (with jamming): the RSS values for the observed bit errors are high for both receivers and thus the interference is correctly detected.*

The BTnode implementation uses our advanced header detection introduced in Section 12.3. To allow for the jamming of single bits with the jammer node J, the transmission rate of the sender and receiver was reduced to 2.4 kBaud, whereas the jammer was sending random data at a rate of 38.4 kBaud.

We performed our experiments in two different scenarios: In the fist scenario the wireless connection between the sender and the receivers was fairly good, that is, the RSS of A's signal at B and C was about -55 dBm; in the second scenario the connection between A and B was rather weak, that is, the RSS of A's signal at B and C was about -70 dBm. To make the jamming detection most challenging, the transmission power of the jammer was set to the lowest possible value for which the jamming was still effective (i.e., >1%), which was 3 dBm for the scenario with the strong links and -5 dBm for the scenario with the weak link.

In both scenarios, we measured the performance of the four bit error detection techniques introduced in Section 13.1. Each technique was evaluated with a series of 1000 undisturbed packet transmissions, five times 2000 packet transmissions where a fraction of 2, 4, 8, 16, or 24 bit was

number of jammed bits	message known or predetermined		
2	100% / 0%	—	0% / 100%
4	100% / 0%	—	0% / 100%
8	100% / 0%	—	0% / 100%
≥16	100% / 0%	—	0% / 100%
number of jammed bits	message encoded with ECCs		
2	100% / 0%	—	0% / 100%
4	100% / 0%	—	0% / 100%
8	99.9% / 0.1%	—	0% / 100%
≥16	100% / 0%	—	0% / 100%
number of jammed bits	comparison of two receptions		
2	84.9% / 15.1%	—	0% / 100%
4	94.1% / 5.9%	—	0% / 100%
8	98.8% / 1.2%	—	0% / 100%
≥16	100% / 0%	—	0% / 100%

Table 14.1: *Jamming detection performance for a strong link: true positives / false negatives — false positives / true negatives*

jammed, and three times 2000 transmissions where a fraction of 8, 16, or 24 bit was suppressed (i.e., the transmission power at the sender was reduced to the minimum during their transmission in order to simulate a temporarily weak signal). The obtained results are summarized in Table 14.1 and 14.2. The second column shows the results for the case where the received packet is already known by the receiver, that is, the two techniques where the content of a packet is either predetermined or was first transmitted over the wire (active probing). The results in the third column represent the bit error location technique based error correcting codes and were obtained with a Hamming (8,4) code that allows for correcting single bit errors, detecting all two bit errors, and detecting some three bit errors. The results in the fourth column, finally, show the results for the case in which two wired nodes exchange their individual receptions.

First of all we notice that throughout our extensive experiments, no single false positive occurred (i.e., no bit error was erroneously identified as being caused by jamming). Furthermore, all false negatives (i.e., jamming-caused errors that were not identified as such) were due to inaccuracies

number of jammed bits	message known or predetermined		
2	100% / 0%	—	0% / 100%
4	100% / 0%	—	0% / 100%
8	100% / 0%	—	0% / 100%
≥16	100% / 0%	—	0% / 100%
number of jammed bits	message encoded with ECCs		
2	100% / 0%	—	0% / 100%
4	100% / 0%	—	0% / 100%
8	99.9% / 0.1%	—	0% / 100%
≥16	100% / 0%	—	0% / 100%
number of jammed bits	comparison of two receptions		
2	85.2% / 14.8%	—	0% / 100%
4	94.1% / 5.9%	—	0% / 100%
8	98.7% / 1.3%	—	0% / 100%
≥16	100% / 0%	—	0% / 100%

Table 14.2: *Jamming detection performance for a weak link: true positives / false negatives — false positives / true negatives*

in the bit error localization. More precisely, with some small probability it happens that the bit errors result again in a valid code word or that the two wired nodes observe exactly the same bit flips, respectively. We point out that in this respective, the measured results for the 2-tuple are actually worst case results because the more nodes are connected by wire, the less likely it is that all observe exactly the same bit flips.

14.1 Sequential Jamming Test

We next analyze the performance of the sequential testing which is required in cases where unintentional packet collisions cannot be neglected. Let λ_j be the fraction of all transmissions within the attacker's jamming range that she actually jams (i.e., the aggressiveness of the attacker) and p_c be the expected collision that a node observes. The expected number of channel samples that is faulty due to interference after k samples is thus $q = (1-(1-\lambda_j)(1-p_c))k$. Inserting this expression into Equation (13.1) and solving the equation $\eta(k) = \log \frac{1-\sigma_{FN}}{\sigma_{FP}}$ for k then yields the expected

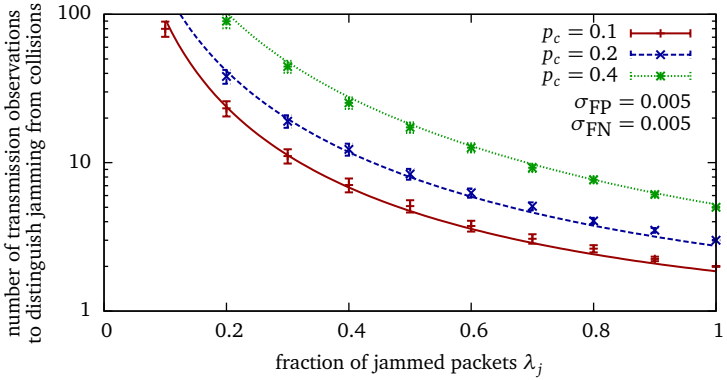

Figure 14.2: *Performance of the sequential jamming test. We observe that the larger the collision probability p_c and the lower the fraction of packets λ_j that the jammer jams, the longer it takes to detect the jammer; however, the lesser is also the impact of the jammer. If the attacker blocks an alarm (i.e., if $\lambda_j = 1$) the jamming will be detected after only five channel samples (for $p_c \leq 0.4$). Since alarm packets are immediately repeated if not acknowledged and because the attacker has to jam all alarms, this number will usually be reached after only a few seconds.*

number of channel samples that must be processed before the jamming is detected. The resulting jamming detection performance as a function of p_c and λ_j is shown in Figure 14.2. The lines show the theoretical value, the points and confidence intervals the results of our experiments. In a typical alarm forwarding scenario, the most relevant situation is one where the attacker intends to mask an alarm (i.e., $\lambda_j = 1$). We observe that in this case the jamming will be detected after only five channel samples (for reasonable collision probabilities $p_c \leq 0.4$). Due to the fact that alarm packets are repeated if not acknowledged and because the attacker has to jam all alarms, we argue that this number will usually be reached after only a few seconds.

14.2 Impact of the Node Density

Having evaluated the detection performance of our scheme if run on a node or n-tuple, we finally analyze the probability that the attacker's jamming activities are observed by a node or by an n-tuple in her proximity. We assume that at least a fraction of all deployed nodes runs our detection

algorithm and participates in the jamming detection. Upon the detection of a jamming attack, the nodes raise a jamming alert which is then either locally stored (e.g., as evidence for later investigations) or reported to the network authority by means of the existing alarm forwarding scheme. In the latter case, the attacker might extend the jammed region during her attack in order to also block these new alarms caused by her jamming. Consequently, if the jamming alarms raised by the initial set of nodes cannot escape the jammed area, these blocked jamming alarms must in turn also be detected by neighboring nodes, and so forth. This means that the entire jamming notification process can be composed of several iterative detection steps. However, since the same rules and conditions apply to all these steps, we focus on one such detection step and not further discuss the jamming alarm reporting in the following.

14.2.1 Monitoring by Unwired Nodes

Ideally, a node would receive and analyze every packet it overhears. However, given the stringent energy constraint of current sensor nodes, not all nodes in the transmission range of a sender usually receive a packet, but only the set of intended receivers.

Let N_a be the average number of neighbors of a node, p_r be the probability that a neighbor which is not an intended receiver of a packet still receives and analyzes it, and p_d be the probability that potential jamming is correctly detected. As each packet has at least one receiver, the probability that the jammer is detected by the neighbors of the sender is $\geq 1-(1-p_d)^{1+(N_a-1)p_r}$. Let further N be the total number of nodes deployed in the deployment area $\mathcal{A}$ and $R(\cdot)$ be a function that maps transmission power levels to distances. The function $R(\cdot)$ depends on the nodes' radios and the environment they are deployed in. For the well-known physical communication model [35], for instance, we have $R(P) := \sqrt[\alpha]{\frac{P}{\beta N_0}}$, where α, $2 < \alpha \leq 6$, is the so-called path-loss exponent, N_0 is the ambient noise power level, and β is the minimal required signal-to-noise-ratio to receive a packet. Given the transmission power P_a of a node and assuming (roughly) uniform node density, the node's expected number of neighbors N_a can then be estimated as $N_a \approx N\frac{R(P_a)^2\pi}{\mathcal{A}}$.

Figure 14.3 depicts the probability that the jammer is detected by the neighbors as a function of p_d, N_a and p_r. We observe that given the fairly high accuracy of typically $p_d \gtrapprox 0.9$ for the nodes' jamming detection (see above) an overall detection probability of ≥ 0.99 is already achieved with a single additional receiver (i.e., if $(N_a-1)p_r \geq 1$). Note that this result

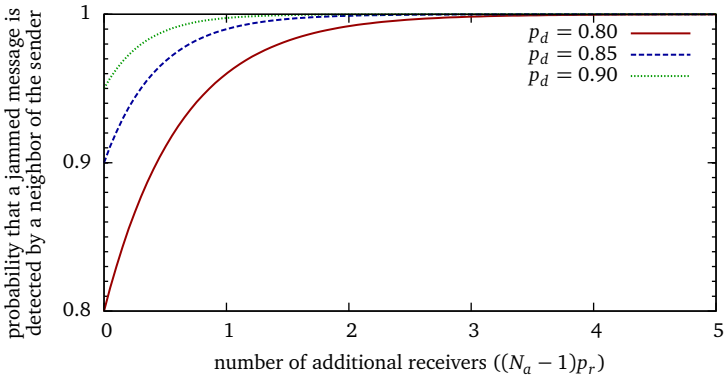

Figure 14.3: *Probability that a jammed packet is detected by at least one of the nodes in the proximity of the sender. Since the accuracy of the local jamming detection is already fairly high (i.e., $p_d \gtrapprox 0.9$) an overall detection probability of ≥ 0.99 is achieved with only one additional receiver (i.e., if $(N_a - 1)p_r \geq 1$).*

applies to a single transmission. Since alarm messages are repeated if not acknowledged, all blocked alarms will eventually be detected by at least one neighbor in practice.

14.2.2 Monitoring by *n*-tuples

As mentioned in Section 13.1, the detection performance of the *n*-tuples depends not only on the tuple density, but also on whether the monitoring is passive or active. In a passive system, at least *two nodes* of an *n*-tuple must be *in the transmission range of the sending node* for a minimum of two independent packet receptions are required. In an active system, the sender is part of the *n*-tuple and thus only *one (additional) node* of an *n*-tuple must be included *in the jammed region*. We next evaluate both scenarios for the node deployment models and simulation parameters introduced in Section 12.4.1.

Active Monitoring

In order to determine the probability p_a that the jammed area is monitored by an *n*-tuple, we first compute the probability of the event X that at least one node of an *n*-tuple lies within a disc of radius r that is centered at the jammer. In a second step, this result is then generalized to the case

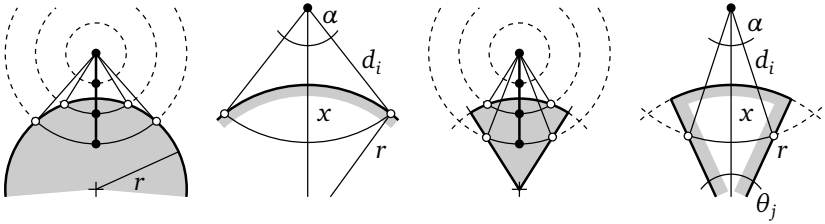

Figure 14.4: *Geometric relations between the position/orientation of an n-tuple and its possible intersections with a disc or a circular sector of radius r; the figures are not true to scale. Drawings (a) and (b) show the situation where the i-th node of the n-tuple lies inside a disc, drawings (c) and (d) where it lies inside a circular sector.*

where the attacker does not emit a single, omnidirectional signal but a set of directional signals. For simplicity, we assume that all nodes of an n-tuple lie on a straight line and that any two consecutive nodes in a tuple have the same distance.

Given the disc with radius r around the jammer, let z be the distance between the first node of an n-tuple and the center of the disc. The probability that at least one node of the tuple lies within the disc is then

$$p_X(r,l) = \int_{x=0}^{\infty} \mathsf{P}[X|z=x]\mathsf{P}[z=x] = \int_{x=0}^{r+l} \mathsf{P}[X|z=x]\frac{2x\pi \mathrm{d}x}{|\mathcal{A}|}\,. \qquad (14.1)$$

Let $d_i = \frac{i-1}{n-1}l$ be the distance between the first and the i-th node in the n-tuple. Now imagine a circle of radius d_i centered at the first node in the tuple. Given that the direction of a tuple is chosen uniformly at random, the probability that the i-th node lies within the disc is then proportional to the central angle subtended by the two intersection points of this circle with the perimeter of the disc to the first node (see Figure 14.4(a)). As illustrated in Figure 14.4(b), this angle is

$$\alpha_X(d_i,x) := \begin{cases} 0 & \text{if } x<0 \text{ or } x>r+d_i \text{ or } x+r<d_i, \\ 2\pi & \text{if } 0 \le x+d_i \le r, \\ 2\arccos\left(\frac{x^2+d_i^2-r^2}{2d_i x}\right) & \text{otherwise.} \end{cases} \qquad (14.2)$$

Hence,

$$\mathrm{P}[X|z=x] = \max_{1\le i<n} \alpha_X\left(\tfrac{i}{n-1}l, x\right)\frac{1}{2\pi} \tag{14.3}$$

and thus

$$p_X(r,l) = \frac{r^2\pi}{|\mathcal{A}|} + \int_{x=r}^{r+l} \frac{\max\limits_{1\le i<n} \alpha_X\left(\tfrac{i}{n-1}l, x\right)}{2\pi}\frac{2x\pi \mathrm{d}x}{|\mathcal{A}|}\,. \tag{14.4}$$

For an omnidirectional jammer the probability that the jammed area is monitored by a wired node is

$$p_a \ge 1 - \left(1 - p_X(R(P_j^1), l)\right)^m\,, \tag{14.5}$$

where m is the number of deployed n-tuples and $R(P_j^1)$ is the radius of the jammed area (i.e., the area in which jamming is effective and thus also detectable).

In the case of a general attacker that emits several directional signals specified by the set $\{(\theta_j^1, P_j^1), (\theta_j^2, P_j^2), \ldots, (\theta_j^k, P_j^k)\}$ (see Section 12.1), the probability that the jammed area is monitored by a wired node is equal to the probability that at least one node of an n-tuple lies within one of the respective circular sectors of central angle θ_j^i and radius $R(P_j^i)$. Considering only those n-tuples whose first node is enclosed by one of these sectors or their extension to a radius of length $R(P_j^i)+l$, this probability can be approximated as

$$p_w \gtrapprox 1 - \prod_{i=1}^{k}\left(1 - \left(1 - \frac{\theta_j^i}{2\pi}p_X(R(P_j^i), l)\right)^m\right)\,. \tag{14.6}$$

In order to account for those cases where the intersection of the (virtual) circles with radii d_i and r are outside of the circular sector given by θ_j^i (i.e., if $\theta_j^i < \pi$, see Figure 14.4(c))) the function $\alpha_X(d_i, x)$ has additionally to be substituted with $\min(\alpha_X(d_i, x), \alpha'_X(d_i, x))$, where

$$\alpha'_X(d_i, x) := 4\arcsin(\sqrt{y}/(2d_i)), \qquad y = r^2 + (x-d_i)^2 - 2\cos(\theta_j^i/2)r(x-d_i) \tag{14.7}$$

is the angle subtended by the two intersection points of the circle with radius d_i and the radii of the circular sector to the first node (see Figure 14.4(d)).

Passive Monitoring

Recall that with passive monitoring at least two nodes of an n-tuple must be in the transmission range of the sending node to detect a jamming attack. Let Z denote the event that at least two nodes of an n-tuple lie within a disc of radius r centered at the sender and let z denote the distance between the first node of an n-tuple and the center of this disc. The probability that at least two nodes of the tuple lie within the disc is then

$$p_Z(r,l) = \int_{x=0}^{\infty} \mathsf{P}[Z|z=x]\mathsf{P}[z=x] = \int_{x=0}^{r+l} \mathsf{P}[Z|z=x]\frac{2x\pi \mathrm{d}x}{|\mathcal{A}|} \,. \quad (14.8)$$

The probability that two nodes lie within the disc is proportional to the smaller of the two respective center angels. Let $\overset{2}{\max}$ be a function that returns the second largest value. We obtain

$$\mathsf{P}[Z|z=x] = \overset{2}{\underset{0\le i<n}{\max}} \alpha_X\left(\tfrac{i}{n-1}l, x\right)\frac{1}{2\pi} \quad (14.9)$$

and thus

$$p_Z(r,l) = \int_{x=0}^{r+l} \frac{\overset{2}{\underset{0\le i<n}{\max}} \alpha_X\left(\tfrac{i}{n-1}l, x\right)}{2\pi}\frac{2x\pi \mathrm{d}x}{|\mathcal{A}|} \,. \quad (14.10)$$

The probability that the neighborhood of a node is (passively) monitored by an n-tuple is

$$p_p \ge 1 - \left(1 - p_Z(R(P_a), l)\right)^m \,, \quad (14.11)$$

where m is the number of deployed n-tuples and $R(P_a)$ the nodes' transmission range.

The influence of the number of nodes n per tuple, the wire length l_w, the number of deployed tuples m, the node's transmission power P_a, and the size of the jammed area (i.e., P_j^i and θ_j^i) on the jamming detection performance of active and passive monitoring is depicted in Figure 14.5 and 14.6, respectively. The results show that even for short wires of about 3 m and a moderate jamming range of 100 m, only 80 3-tuples must be deployed per 1 km^2 in order that the jamming is (actively) monitored by at least one wired node with $p_a > 95\%$. In a purely passive scenario, about $(R(P_j)/R(P_a))^2$ times as many n-tuples have to be deployed to achieve the same protection as in an active scenario.

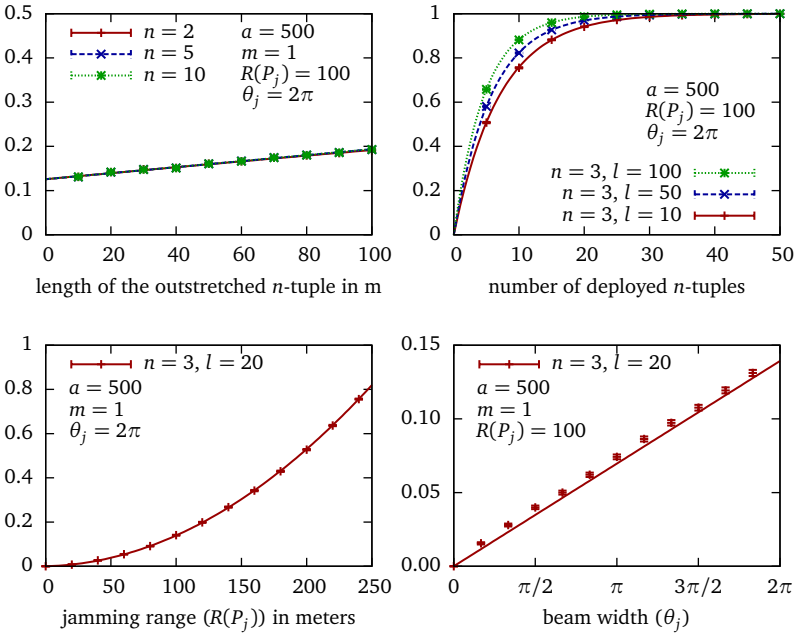

Figure 14.5: *Probability p_a that the jammed area contains at least one wired node. The lines show the expected result according to our analysis, the points and confidence intervals the simulation results. We observe that even very long tuples cannot benefit from more than 3 wired nodes per tuple, whereas increasing the length of a tuple results in a linear increase of p_a. As p_a is proportional to the size of the jammed area, it increases exponentially with the jamming range and approximately linear with the beam width. Our findings also show that even for short wires with a length of about 3 m and a moderate jamming range of 100 m, only 80 3-tuples must be deployed per km^2 in order that the jamming is monitored by at least one wired node with $p_a > 95\%$.*

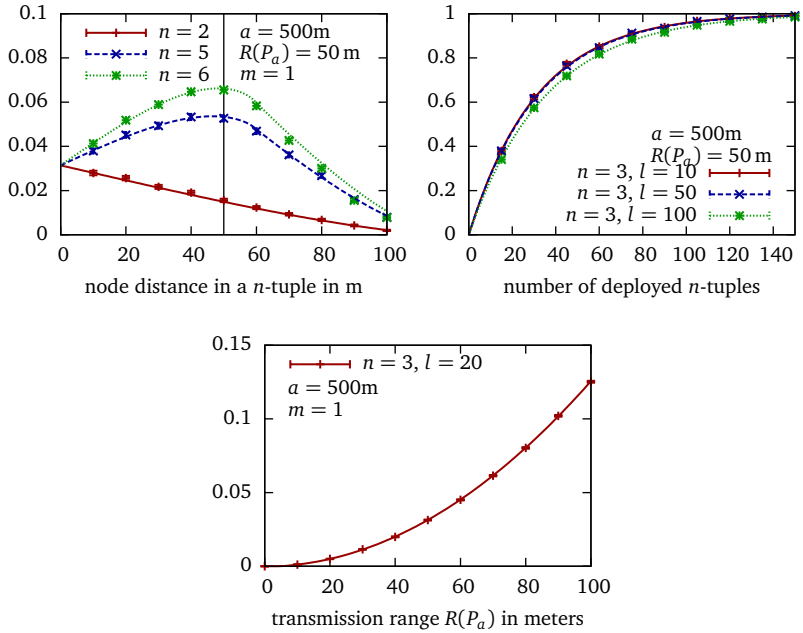

Figure 14.6: *Probability p_p that the neighborhood of a node is monitored by an n-tuple. We observe that for 2-tuples p_p decreases for longer wires and becomes zero if $l_w >$ twice the transmission range $R(P_a)$, whereas for $n > 2$ the probability p_p becomes maximal if the distance between two wired nodes in a tuple is $\approx r$. As p_p is proportional to the size of a node's neighborhood, it increases exponentially with the transmission range. Also, for $R(P_a) = R(P_j)/k$, in a passive scenario about k^2 times as many n-tuples have to be deployed than in an active scenario to achieve the same protection (e.g., $p_p >$95%).*

Chapter 15
Related Work

In this chapter, we discuss work that is related to Part II of this thesis, focusing on sensor networks. We refer to Chapter 9 for a discussion of related work that reviews the impact and mitigation of jamming at large.

15.1 Jamming detection

The detection and mapping of jammed areas in the realm of wireless sensor networks was studied by Wood, Stankovic, and Son [81]. They propose to use the utility of a link for jamming detection but do not further evaluate its performance as a detection metric. Instead, they focus on the problem of mapping the jammed area by identifying and grouping nodes with at least one jammed neighbor.

Xu et al. advocate the usage of packet delivery ratio (PDR) along with either signal strength at the receiver (RSS) or location information as a consistency check for jamming detection [83]. In the former case, jamming is detected if the PDR is low although the RSS value is high, in the latter case if the PDR is low although the senders are close. We point out that unlike our work, their scheme does not correlate the RSS measurements on a per-bit basis, but compares an averaged RSS value with a threshold once the PDR drops below a specified level.

Çakiroğlu and Özcerit proposed two jamming detection schemes based on the PDR, the bad packet ratio (BPR), and the energy consumption amount (ECA) of the radio [18]. In the basic scheme, jamming is detected if the PDR, BPR, or ECA values rise above or fall below specified thresholds. Altogether, five rules are specified, each focusing on a different set of jammer types. In the extended scheme, the nodes base their decision not only on their local view but exchange query and alarm messages with their neighbors in order to reduce the number of false positives at the expense of an increased communication overhead.

A major drawback shared by the aforementioned schemes is that assessing an accurate PDR is not practical for a reactive forwarding scheme, as messages are sent very rarely. Moreover, as argued in this and previous work [5, 60, 76], jamming does not necessarily cause a steady and high RSS value, as only a small fraction of a packet has to be interfered with in order for the packet to be invalid [57]. A (reactive) jammer can thus keep the increase in the effective RSS value very low and can hence avoid being

detected by these schemes. Also, the proposed detection algorithms cannot distinguish between intentional and unintentional interference, and timely delivery of alarm notifications is not considered.

A sequential jamming detection technique based on the number of erroneously received messages was presented by Li, Koutsopoulos, and Poovendran [46]. The key idea of the proposed algorithm is that an increased number of observed message collisions during an observation window compared to the learned long-term average indicates a jamming attack. Using Wald's Sequential Probability Ratio Test, they identify optimal jamming attacks as well as network defense policies with respect to detection and notification time. Unlike our algorithm, that approach cannot distinguish between packet failures due to weak links and collisions. Thus, it is sensitive to changes in nodes' environment that influence the observed PDR and to (temporary) link failures or unanticipated changes in the traffic pattern which are likely to cause false alarms. Finally, none of the above mentioned schemes considers overshadowing, where the original packet is covered by a (maliciously inserted) second message.

15.2 Wired Infrastructure

The application of additional infrastructure in the form of wired shortcuts was proposed before by Chitradurga and Helmy [20] and by Sharma and Mazumdar [65]. The objective of these works is to improve the energy efficiency of wireless networks by identifying optimal wirings; the dependability and jamming resistance of the hybrid network is not considered. Čagalj at al. show how wired node pairs can be used to build wormholes in order to establish communication out of a jammed area [77]. The two main differences between their and our work are that they do not consider the use of wired tuples for jamming detection but exclusively for alarm forwarding and that they focus on wired node pairs, whereas we also investigate wired node chains of more than two nodes. Furthermore, their work does neither consider the security of the wired links nor evaluate to what extent the proposed wirings are feasible.

15.3 Alarm Forwarding

Resilience to message failure is an important functionality required by safety critical systems. However, most transport protocols for sensor networks that do address link failures (e.g., ESRT [6] and PSFQ [78]) only raise the reliability, for example by utilizing path diversity to overcome

errors on individual links, but do not provide end-to-end guarantees. At best, advanced protocols like MMSPEED [29] provide probabilistic bounds on end-to-end delivery ratios but ignore energy consumption. End-to-end acknowledgment-based schemes such as [24], Flush [42], or RCRT [58] increase the reliability, but suffer greatly from high costs and long (multi-hop) latencies, making them unsuitable for time-critical applications.

An effective, but expensive, approach to handle communication errors is to use (partial) flooding [75]. The GRAB protocol [84], for instance, uses a credit mechanism to specify how many additional hops may be made to reach the destination, effectively creating a "wide-path". The DFRF framework [52] generalizes this idea and allows for easy creation of tailor-made partial-flooding. Unfortunately, DFRF and GRAB do not integrate well with the MAC layer below, making it hard to control latency and energy consumption. Like Dwarf [69], Dynamic Switch-based Forwarding (DSF) [34] is based on the observation that the (end-to-end) latency can be significantly reduced by forwarding a message to the neighbor that wakes up next instead of waiting until a specific, predetermined node becomes active. The main difference between the two schemes is that Dwarf selects the next hop in an ad-hoc manner, whereas DSF proactively computes the forwarding sequence at regular intervals.

Chapter 16
Conclusions

A steadily growing number of wireless devices that penetrate our everyday life and environment brings about a multitude of new applications and services that need to be appropriately secured. A major challenge in reaching this goal is the inherent vulnerability of wireless communication to communication jamming DoS attacks. The problem becomes even more significant the more one takes the ubiquity of these emerging applications and services for granted. In this thesis, we addressed this problem, focusing on scenarios where common anti-jamming techniques (such as FHSS and DSSS) cannot be applied.

More specifically, in the first part of this thesis, we tackled the problem of how devices that *do not share any secrets* can establish jamming-resistant communication over a wireless radio channel in the presence of a communication jammer. We addressed the dependency between anti-jamming spread-spectrum communication and pre-shared keys that is inherent to this problem, and proposed a novel anti-jamming technique called Uncoordinated Frequency Hopping (UFH) as a solution to break this dependency. We presented a number of UFH-based communication schemes and showed their use in several applications. In particular, we illustrated on the examples of the authenticated Diffie-Hellman and the Burmester-Desmedt key agreement protocols how our schemes can be used to establish a shared secret (group) key to bootstrap common (coordinated) frequency hopping.

We performed a detailed analysis of our UFH-based schemes and showed their feasibility by means of a prototype implementation. Our evaluation results show that even with our simple prototype, the average time to establish a pairwise or group key is in the order of a few seconds (for a processing gain of 23 dB). This time is reasonably short, given that the much shorter channel switching times and the higher data rates of purpose-built hardware allow to decrease this time significantly, and that with common anti-jamming techniques the devices would not be able to communicate and thus could not execute a key establishment protocol. We modeled and analyzed the impact of different attacker types and strategies on UFH communication and presented optimal channel selection strategies to counter these attacks. Our analysis also showed that, although our UFH scheme has lower communication throughput, it achieves the same level of anti-jamming protection as common frequency hopping.

In the second part of this thesis, we addressed the problem of jamming attacks on alarm forwarding in sensor networks. We demonstrated the susceptibility to jamming attacks of current state-of-the-art forwarding schemes for WSNs and discussed possible techniques to mitigate the impact of jamming. We further presented a novel jamming detection scheme for countering advanced (reactive single bit) jamming attacks in sensor networks. Our detection scheme is able to identify the root cause of bit errors for individual packets by looking at the received signal strength during the reception of these bits. The scheme is thus well-suited for the protection of reactive alarm systems with very low network traffic. We presented and discussed three different techniques for the detection and localization of bit errors based on: predetermined knowledge, error correcting/detecting codes, and limited node wiring in the form of wired node chains (n-tuples). The presented protocols and algorithm were evaluated analytically, by simulations, and experimentally on COTS BTnodes. The results showed that our solution effectively detects sophisticated jamming attacks that cannot be detected with existing techniques and enables the formation of robust sensor networks for the dependable delivery of alarm notifications. Since our scheme can operate without introducing additional wireless network traffic, it also meets the high energy efficiency demand of reactive surveillance applications.

16.1 Contributions

In summary, the main contributions of this thesis are:

Part I: Anti-jamming Communication without Pre-shared Secrets

- the identification of the problem of how to achieve anti-jamming communication without shared secrets and the introduction of the anti-jamming/key-establishment circular dependency problem;
- UFH as a novel anti-jamming technique along with several UFH-based communication schemes that support the transmission of messages of arbitrary length in a jammed environment without relying on shared secrets;
- a comprehensive jammer model for reactive and non-reactive jamming which allows the computation of the number of blocked channels and the probability that a packet is jammed as a function of the packet length and the packet encoding;

- the description of several applications that strongly benefit from the UFH communication schemes, as they could so far not be protected with common spread spectrum techniques, notably the execution of (group) key establishment protocols in the presence of a jammer;

Part II: Detection of Reactive Jamming in Sensor Networks

- the demonstration of the susceptibility of current state-of-the-art forwarding schemes for WSNs to jamming attacks;
- jamming mitigation techniques for sensor networks based on forward error correction and limited node wiring;
- a novel jamming detection scheme for countering advanced (reactive single bit) jamming attacks in sensor networks along with three different techniques for the identification of bit errors based on: predetermined knowledge, error correcting codes, and limited node wiring.

16.2 Remaining Issues and Future Work

An unavoidable constraint of UFH is that the frequency hops (i.e., packets) cannot be arbitrarily small but comprehend at least a message and packet id, one data bit, and the packet verification data. Altogether, this results in a minimum hop length of roughly 80-100 bit; so-called fast frequency hopping, where the frequency channel is switched for each bit, is thus not possible. A second drawback of UFH is that the jamming resistance and the achievable throughput are tightly linked to each other: the more channels are used, the lower is the probability that the receiver is listening on the right one. With other spread-spectrum techniques this might not necessarily be the case. A promising direction for future work is thus the investigation of alternative uncoordinated (spread-spectrum) techniques as well as combinations thereof. Another topic for future work would be to identify additional applications for the introduced packet verification techniques which provide message integrity without the need for shared secrets or sender authentication. These techniques might, to give an example, help in countering DoS attacks on the fragment assembly process of distributed file sharing applications or application level firewalls. Related to this, it would further be interesting to find additional, more efficient packet verification techniques.

What might be criticized concerning the work presented in Part II of this thesis is that detecting a jamming attack is not equally helpful in different

WSN surveillance applications. In a burglar alarm system, for example, there is no difference in the response to a jamming or intrusion alert, and thus no relevant information is lost if an intrusion alert is replaced with a jamming alert as consequence of a jamming attack. In a fire alarm system, on the other hand, the alarm messages carry important information (e.g., about the smoke density and whether there was only smoke or also fire) and a jamming alert, although still better than nothing, is less informative than the original alert. In other words, detecting jamming cannot always make up for a message loss, that is, for the lack of anti-jamming communication.

Regarding the second part of this thesis, future work includes the development of additional techniques for the detection of bit errors and the evaluation of alternative methods to distinguish unintentional from intentional interference. We further believe that this work provides useful insights into the utility of limited wiring as a means for securing wireless sensor networks, but future work is required to identify additional fields of application.

A more general direction for future work consists in exploring the impact of jamming, not only on the physical layer but on all layers of the network stack. Above all, one needs a better understanding of (distributed) jamming attacks that target particular, application-specific traffic characteristics and exploit the distributed nature of most applications. Future work is also required to get more insight into what kind of attacks are feasible on the physical layer, under which conditions, and at which costs, in order to improve existing attacker and jammer models.

16.3 Publications

The work presented in this thesis is based on the following publications:

P1 Mario Strasser, Andreas F. Meier, Koen Langendoen, and Philipp Blum. Dwarf: Delay-aWAre Robust Forwarding for Energy-Constrained Wireless Sensor Networks. In *Proceedings of the IEEE Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2007

P2 Mario Strasser, Christina Pöpper, Srdjan Čapkun, and Mario Čagalj. Jamming-resistant Key Establishment using Uncoordinated Frequency Hopping. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2008.

P3 Mario Strasser, Christina Pöpper, and Srdjan Čapkun. Efficient Uncoordinated FHSS Communication. In *Proceedings of the ACM Inter-*

national Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), 2009.

P4 Christina Pöpper, Mario Strasser, and Srdjan Čapkun. Jamming-resistant Broadcast Communication without Shared Keys. In *Proceedings of the USENIX Security Symposium*, 2009.

P5 Christina Pöpper, Mario Strasser, and Srdjan Čapkun. Anti-jamming Broadcast Communication using Uncoordinated Spread Spectrum Techniques. To appear in *IEEE Journal on Selected Areas in Communications (J-SAC), Special Issue Mission Critical Networking*, 2010.

P6 Mario Strasser, Boris Danev, and Srdjan Čapkun. Detection of Reactive Jamming in Sensor Networks. To appear in *ACM Transactions on Sensor Networks (TOSN)*, 2010.

P7 Andreas Meier, Mario Strasser, Simon Künzli, and Severin Hafner. Safety-Critical Event Monitoring with Wireless Sensor Networks. *Under submission*.

P8 Mario Strasser, Christina Pöpper, and Srdjan Čapkun. Anti-jamming Communication without Shared Keys using Uncoordinated Frequency Hopping. *Under submission*.

Bibliography

[1] BTnodes – A Distributed Environment for Prototyping Ad Hoc Networks. http://www.btnode.ethz.ch/.

[2] GNU Radio Software. http://gnuradio.org/trac.

[3] ECRYPT Yearly Report on Algorithms and Keysize. D.SPA.28, July 2008. IST-2002-507932.

[4] Imad Aad, Jean-Pierre Hubaux, and Edward W. Knightly. Denial of Service Resilience in Ad hoc Networks. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, pages 202–215. ACM, 2004.

[5] David Adamy. *A First Course in Electronic Warfare*. Artech House, 2001.

[6] Özgür B. Akan and Ian F. Akyildiz. Event-to-sink Reliable Transport in Wireless Sensor Networks. *IEEE/ACM Transaction on Networking*, 13(5):1003–1016, 2005.

[7] ANSI. X9.63-2001: Key Agreement and Key Transport Using Elliptical Curve Cryptography. American National Standards Institute, 2001.

[8] International Loran Association. LORAN: LOng Range Aid to Navigation. http://www.loran.org.

[9] Paramvir Bahl and Venkata N. Padmanabhan. RADAR: An In-building RF-based User Location and Tracking System. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 775–784. IEEE Communications Society, 2000.

[10] Leemon C. Baird, William L. Bahn, Michael D. Collins, Martin C. Carlisle, and Sean Butler. Keyless Jam Resistance. In *Proceedings of the IEEE Information Assurance and Security Workshop (IAW)*, pages 143–150. IEEE, 2007.

[11] Niko Bari and Birgit Pfitzmann. Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees. In *Advances in Cryptology EUROCRYPT*, volume 1233/1997 of *Lecture Notes in Computer Science*, pages 480–494. Springer Berlin / Heidelberg, 1997.

[12] Michael Baron. *Probability and Statistics for Computer Scientists*. Chapman & Hall/CRC, 2007.

[13] Emrah Bayraktaroglu, Christopher King, Xin Liu, Guevara Noubir, Rajmohan Rajaraman, and Bishal Thapa. On the Performance of IEEE 802.11 under Jamming. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 1265–1273. IEEE Communications Society, 2008.

[14] Josh Benaloh and Michael de Mare. One-Way Accumulators: A Decentralized Alternative to Digital Signatures. In *Advances in Cryptology EUROCRYPT*, volume 765/1994 of *Lecture Notes in Computer Science*, pages 274–285. Springer Berlin / Heidelberg, 1994.

[15] Alan Bensky. *Wireless Positioning Technologies and Applications*. Artech House, 2008.

[16] Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signatures from the Weil Pairing. *Journal of Cryptology*, 17(4):297–319, 2004.

[17] Mike Burmester and Yvo Desmedt. A Secure and Efficient Conference Key Distribution System. In *Advances in Cryptology EUROCRYPT*, volume 950 of *Lecture Notes in Computer Science*, pages 275–286. Springer Berlin / Heidelberg, 1995.

[18] Murat Çakiroğlu and Ahmet Turan Özcerit. Jamming Detection Mechanisms for Wireless Sensor Networks. In *Proceedings of the International Conference on Scalable Information Systems (InfoScale)*, pages 1–8. ICST, 2008.

[19] J.T. Chiang and Yih-Chun Hu. Dynamic Jamming Mitigation for Wireless Broadcast Networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 1211–1219. IEEE Communications Society, 2008.

[20] Rohan Chitradurga and Ahmed Helmy. Analysis of Wired Short Cuts in Wireless Sensor Networks. In *Proceedings of the IEEE/ACS International Conference on Pervasive Services (ICPS)*, pages 167–176. IEEE Computer Society, 2004.

[21] Clayton W. Commander, Panos M. Pardalos, Valeriy Ryabchenko, Oleg Shylo, Stan Uryasev, and Grigoriy Zrazhevsky. Jamming Communication Networks under Complete Uncertainty. *Optimization Letters*, 2(1):53–70, 2008.

[22] Clayton W. Commander, Panos M. Pardalos, Valeriy Ryabchenko, Stan Uryasev, and Grigoriy Zrazhevsky. The Wireless Network Jamming Problem. *Journal of Combinatorial Optimization*, 14(4):481–498, 2007.

[23] Yvo Desmedt, Rei Safavi-Naini, Huaxiong Wang, Lynn Batten, Chris Charnes, and Josef Pieprzyk. Broadcast Anti-Jamming Systems. In *Proceedings of the IEEE International Conference on Networks (ICON)*, pages 349–355. IEEE Computer Society, 1999.

[24] Tuan Le Dinh, Wen Hu, Pavan Sikka, Peter Corke, Leslie Overs, and Stephen Brosnan. Design and Deployment of a Remote Robust Sensor Network: Experiences from an Outdoor Water Quality Monitoring Network. In *Proceedings of the IEEE Conference on Local Computer Networks (LCN)*, pages 799–806. IEEE Computer Society, 2007.

[25] Shlomi Dolev, Seth Gilbert, Rachid Guerraoui, and Calvin Newport. Secure Communication over Radio Channels. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 105–114. ACM, 2008.

[26] Prabal Dutta, Mike Grimmer, Anish Arora, Steven Bibyk, and David Culler. Design of a Wireless Sensor Network Platform for Detecting Rare, Random, and Ephemeral Events. In *Proceedings of the IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 70–75. IEEE, 2005.

[27] Amre El-Hoiydi and Jean-Dominique Decotignie. WiseMAC: An Ultra Low Power MAC Protocol for Multi-hop Wireless Sensor Networks. In *Algorithmic Aspects of Wireless Sensor Networks*, volume 3121 of *Lecture Notes in Computer Science*, pages 18–31. Springer Berlin / Heidelberg, 2004.

[28] Fire detection and fire alarm systems – Part 25: Components using radio links. European Norm (EN) 54-25:2008-06, 2008.

[29] Emad Felemban, Chang-Gun Lee, and Eylem Ekici. MMSPEED: Multipath Multi-SPEED Protocol for QoS Guarantee of Reliability and Timeliness in Wireless Sensor Networks. *IEEE Transactions on Mobile Computing*, 5(6):738–754, 2006.

[30] Anna Lisa Ferrara, Matthew Green, Susan Hohenberger, and Michael Østergaard Pedersen. Practical Short Signature Batch Verification. 5473:309–324, 2009.

[31] Seth Gilbert, Rachid Guerraoui, and Calvin Newport. Of Malicious Motes and Suspicious Sensors. *Theoretical Computer Science*, 410(6-7):546–569, 2009.

[32] Philippe Golle and Nagendra Modadugu. Authenticating Streamed Data in the Presence of Random Packet Loss. In *Proceedings of the Symposium on Network and Distributed Systems Security (NDSS)*, pages 13–22. Internet Society, 2001.

[33] U.S. Government. Global positioning system. http://www.gps.gov, 2009.

[34] Yu Gu and Tian He. Data Forwarding in Extremely Low Duty-cycle Sensor Networks with Unreliable Communication Links. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 321–334. ACM, 2007.

[35] Piyush Gupta and P. R. Kumar. The Capacity of Wireless Networks. *IEEE Transactions on Information Theory*, 46(2):388–404, 2000.

[36] 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 Krogh. VigilNet: An Integrated Sensor Network System for Energy-efficient Surveillance. *ACM Transactions on Sensor Networks*, 2(1):1–38, 2006.

[37] James Hendricks, Gregory R. Ganger, and Michael K. Reiter. Low-overhead Byzantine Fault-tolerant Storage. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, pages 73–86. ACM, 2007.

[38] James Hendricks, Gregory R. Ganger, and Michael K. Reiter. Verifying Distributed Erasure-coded Data. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 139–146. ACM, 2007.

[39] Esa Hyytiä, Tuomas Tirronen, and Jorma T. Virtamo. Optimal Degree Distribution for LT Codes with Small Message Length. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 2576–2580. IEEE Communications Society, 2007.

[40] Tao Jin, Guevara Noubir, and Bishal Thapa. Zero Pre-shared Secret Key Establishment in the Presence of Jammers. In *Proceedings of*

the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), pages 219–228. ACM, 2009.

[41] Chris Karlof, Naveen Sastry, Yaping Li, Adrian Perrig, and J. D. Tygar. Distillation Codes and Applications to DoS Resistant Multicast Authentication. In *Proceedings of the Symposium on Network and Distributed Systems Security (NDSS)*. The Internet Society, 2004.

[42] Sukun Kim, Rodrigo Fonseca, Prabal Dutta, Arsalan Tavakoli, David Culler, Philip Levis, Scott Shenker, and Ion Stoica. Flush: A Reliable Bulk Transport Protocol for Multihop Wireless Networks. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 351–365. ACM, 2007.

[43] Markus Kuhn. An asymmetric security mechanism for navigation signals. In *Information Hiding*, volume 3200 of *Lecture Notes in Computer Science*, pages 239–252. Springer Berlin / Heidelberg, 2005.

[44] Leslie Lamport. Password Authentication with Insecure Communication. *Communications of the ACM*, 24(11):770–772, 1981.

[45] Koen Langendoen. *Medium Access Control in Wireless Networks*, chapter Medium Access Control in Wireless Sensor Networks. Nova Science Publishers, 2008.

[46] Mingyan Li, I. Koutsopoulos, and R. Poovendran. Optimal Jamming Attacks and Network Defense Policies in Wireless Sensor Networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 1307–1315. IEEE Communications Society, 2007.

[47] Guolong Lin and Guevara Noubir. On link layer denial of service in data wireless LANs. *Wireless Communications & Mobile Computing*, 5(3):273–284, 2005.

[48] Ettus Research LLC. Universal Software Radio Peripheral (USRP). http://www.ettus.com.

[49] Michael Luby. LT Codes. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 271–280. IEEE, 2002.

[50] Michael G. Luby, Michael Mitzenmacher, M. Amin Shokrollahi, Daniel A. Spielman, and Volker Stemann. Practical Loss-resilient Codes. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 150–159. ACM, 1997.

[51] Frosso S. Makri, Andreas N. Philippou, and Zaharias M. Psillakis. Success Run Statistics Defined on an Urn Model. *Advances in Applied Probability*, 39(4):991–1019, 2007.

[52] Miklós Maróti. Directed Flood-Routing Framework for Wireless Sensor Networks. In *Middleware*, volume 3231 of *Lecture Notes in Computer Science*, pages 99–114. Springer Berlin / Heidelberg, 2004.

[53] Petar Maymounkov. Online Codes. Technical Report TR2002-833, New York Univeristy, 2002.

[54] Sara Miner and Jessica Staddon. Graph-Based Authentication of Digital Streams. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 232–246. IEEE Computer Society, 2001.

[55] Lan Nguyen. Accumulators from Bilinear Pairings and Applications. In *Topics in Cryptology CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 275–292. Springer Berlin / Heidelberg, 2005.

[56] Guevara Noubir. On Connectivity in Ad Hoc Networks under Jamming Using Directional Antennas and Mobility. In *Wired/Wireless Internet Communications*, volume 2957 of *Lecture Notes in Computer Science*, pages 521–532. Springer Berlin / Heidelberg, 2004.

[57] Guevara Noubir and Guolong Lin. Low-power DoS Attacks in Data Wireless LANs and Countermeasures. *Mobile Computing and Communications Review*, 7(3):29–30, 2003.

[58] Jeongyeup Paek and Ramesh Govindan. RCRT: Rate-controlled Reliable Transport for Wireless Sensor Networks. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 305–319. ACM, 2007.

[59] Adrian Perrig, J. D. Tygar, Dawn Song, and Ran Canetti. Efficient Authentication and Signing of Multicast Streams over Lossy Channels. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 56–73. IEEE Computer Society, 2000.

[60] Richard A. Poisel. *Modern Communications Jamming Principles and Techniques*. Artech House, 2004.

[61] Richard A. Poisel. *Foundations of Communications Electronic Warfare*. Artech House, 2008.

[62] Joseph Polastre, Jason Hill, and David Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 95–107. ACM, 2004.

[63] Christina Pöpper, Mario Strasser, and Srdjan Čapkun. Jamming-resistant Broadcast Communication without Shared Keys. In *Proceedings of the USENIX Security Symposium*, pages 231–247. USENIX Association, 2009.

[64] Kasper B. Rasmussen, Srdjan Capkun, and Mario Cagalj. Extended Abstract: SecNav: Secure Broadcast Localization and Time Synchronization in Wireless Networks. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, pages 310–313. ACM, 2007.

[65] Gaurav Sharma and Ravi Mazumdar. Hybrid Sensor Networks: A Small World. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 366–377. ACM, 2005.

[66] A. Shokrollahi. Raptor Codes. *IEEE Transactions on Information Theory*, 52(6):2551–2567, 2006.

[67] David Slater, Patrick Tague, Radha Poovendran, and Brian J. Matt. A Coding-Theoretic Approach for Efficient Message Verification Over Insecure Channels. In *Proceedings of the ACM Conference on Wireless Network Security (WiSec)*, pages 151–160. ACM, 2009.

[68] Mario Strasser, Boris Danev, and Srdjan Čapkun. Detection of Reactive Jamming in Sensor Networks. Technical Report D-INFK 634, ETH Zurich, 2009.

[69] Mario Strasser, Andreas Meier, Koen Langendoen, and Philipp Blum. Dwarf: Delay-aWAre Robust Forwarding for Energy-Constrained Wireless Sensor Networks. In *Distributed Computing in Sensor Systems*, volume 4549 of *Lecture Notes in Computer Science*, pages 64–81. Springer Berlin / Heidelberg, 2007.

[70] Symantec. Securing Enterprise Wireless Networks. White Paper, 2003.

[71] Patrick Tague, Mingyan Li, and Radha Poovendran. Mitigation of Control Channel Jamming under Node Capture Attacks. *IEEE Transactions on Mobile Computing*, 8(9):1221–1234, 2009.

[72] Patrick Tague, Sidharth Nabar, Jim Ritcey, and Radha Poovendran. Jamming-Aware Traffic Allocation for Multiple Path Routing using Portfolio Selection. Technical Report NSL 004, University of Washington, 2009.

[73] Patrick Tague, David Slater, Guevara Noubir, and Radha Poovendran. Linear Programming Models for Jamming Attacks on Network Traffic Flows. In *Proceedings of the International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pages 207–216. IEEE, 2008.

[74] Patrick Tague, David Slater, Guevara Noubir, and Radha Poovendran. Quantifying the Impact of Efficient Cross-Layer Jamming Attacks via Network Traffic Flows. Technical Report NSL 005, University of Washington, 2009.

[75] Ken Tang and Mario Gerla. MAC Reliable Broadcast in Ad Hoc Networks. In *Proceedings of the IEEE International Military Communications Conference (MILCOM)*, volume 2, pages 1008–1013. IEEE, 2001.

[76] David Thuente and Mithun Acharya. Intelligent Jamming in Wireless Networks with Applications to 802.11b and Other Networks. In *Proceedings of the IEEE International Military Communications Conference (MILCOM)*. IEEE Communications Society, 2006.

[77] Mario Čagalj, Srdjan Čapkun, and Jean-Pierre Hubaux. Wormhole-Based Antijamming Techniques in Sensor Networks. *IEEE Transactions on Mobile Computing*, 6(1):100–114, 2007.

[78] Chieh-Yih Wan, Andrew T. Campbell, and Lakshman Krishnamurthy. PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks. In *Proceedings of the ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, pages 1–11. ACM, 2002.

[79] Stephen G. Wilson. *Digital Modulation and Coding*. Prentice-Hall, 1996.

[80] Anthony D. Wood and John A. Stankovic. Denial of Service in Sensor Networks. *IEEE Computer*, 35(10):54–62, 2002.

[81] Anthony D. Wood, John A. Stankovic, and Sang Son. JAM: A Jammed-area Mapping Service for Sensor Networks. In *Proceedings of the IEEE Real-Time Systems Symposium (RTSS)*, pages 286–297. IEEE, 2003.

[82] Wenyuan Xu, Wade Trappe, and Yanyong Zhang. Channel Surfing: Defending Wireless Sensor Networks from Interference. In *Proceedings of the IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 499–508. ACM, 2007.

[83] Wenyuan Xu, Wade Trappe, Yanyong Zhang, and Timothy Wood. The Feasibility of Launching and Detecting Jamming Attacks in Wireless Networks. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 46–57. ACM, 2005.

[84] Fan Ye, Gary Zhong, Songwu Lu, and Lixia Zhang. GRAdient Broadcast: A Robust Data Delivery Protocol for Large Scale Sensor Networks. *Wireless Networks*, 11(3):285–298, 2005.

[85] X. J. Zhang. *Auxiliary Signal Design in Fault Detection and Diagnosis*. Springer-Verlag, 1989.

[86] G. Zhou, T. He, J.A. Stankovic, and T. Abdelzaher. RID: Radio Interference Detection in Wireless Sensor Networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 891–901. IEEE Communications Society, 2005.

Resume

MARIO STRASSER

Born in Münsterlingen, TG, Switzerland on April 1, 1978
Citizen of Hüttwilen, TG, Switzerland

Educational Curriculum Vitae

2005 – 2009	**Doctor of Sciences** Swiss Federal Institute of Technology (ETH) Zurich Advisors: Prof. Bernhard Plattner, Prof. Srdjan Čapkun
2003 – 2005	**Master of Science in Computer Science** Swiss Federal Institute of Technology (ETH) Zurich
2002	**Postgraduate Course: Advanced Mathematics** Zurich University of Applied Sciences (ZHW)
1998 – 2001	**Bachelor of Science in Information Technology** Zurich University of Applied Sciences (ZHW)
1994 – 1998	**Advanced Technical College Entrance Qualification** Berufsmittelschule St. Gallen (GBSG)
1994 – 1998	**Eid. Dipl. Elektroniker** Metrohm AG, Herisau

Working and Teaching Experience

2007 – present	Freelance security and strategy consultant/engineer
2005 – 2009	Teaching assistant at ETH Zurich
9 – 12, 2006	Internship at the NEC Lab, Princeton, US
2004 – present	Founder and project leader of the TPM Emulator project
2001 – 2003	Research associate with the Network Security Group of Prof. A. Steffen at ZHW
1994 – 1998	Apprenticeship as an electronics technician at Methrohm AG, Herisau