ETH zürich

ETH Library

Detection of Reactive Jamming in Sensor Networks

Report

Author(s):
Strasser, Mario; Danev, Boris; Capkun, Srdjan

Publication date:
2009

Permanent link:
https://doi.org/10.3929/ethz-a-006835970

Rights / license:
In Copyright - Non-Commercial Use Permitted

Originally published in:
ETH Zurich D-INFK Technical Report 634

This page was generated automatically upon download from the ETH Zurich Research Collection.
For more information, please consult the Terms of use.

Detection of Reactive Jamming in Sensor Networks

ETH Zurich D-INFK Technical Report 634 – August, 2009

Mario Strasser
ETH Zurich, Switzerland
strasser@tik.ee.ethz.ch

Boris Danev
ETH Zurich, Switzerland
boris.danev@inf.ethz.ch

Srdjan Čapkun
ETH Zurich, Switzerland
capkuns@inf.ethz.ch

***Abstract*—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. With current alarm forwarding schemes, blocking of an alarm by jamming is straightforward and jamming is very likely to remain unnoticed. In this work, we propose a novel jamming detection scheme as a solution to this problem. Our scheme 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 and CC2420 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 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.**

I. 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 [11], [13], [9], [21] 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 timely delivery of critical information [23], [25], [22].

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 [18], [25], [17], [16] and have been shown to be well-suited for the detection of *proactive* mid- or long-term jamming [25], [16]. They are, however, not designed to detect *reactive* (packet or single-bit) jamming: To begin with, 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. 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 [18], [10], [17]. 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] and Tmote Sky nodes [3]. 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. In summary, the contributions of this paper are as follows:

- We present a novel jamming detection scheme for countering advanced (reactive single bit) jamming attacks in wireless sensor networks.
- We develop three different techniques for the identification of bit errors based on: predetermined knowledge, error correcting codes, and limited node wiring (n-tuples).
- We evaluate our solution by simulations and experimentally with COTS sensor node platforms (BTnodes and Tmote Sky nodes).
- We analyze the threats on limited wiring and develop a low-power wire compromise detection scheme for the detection of malicious attacks on wires.
- We elaborate an analytical model for random, manual, and airdrop-based deployment of wired n-tuples.

The remainder of this paper is organized as follows: After specifying the system and attacker model in Section II, we discuss the impact and mitigation of reactive jamming in Section III and highlight the need for an efficient detection of reactive jamming. Our novel jamming detection scheme is presented in Sections IV and evaluated in Section V. We present our wire integrity verification protocol in Section VI, discuss related work in Section VII, and conclude in Section VIII.

II. System and Attacker Model

As a typical application scenario for our jamming detection scheme, we consider the following setting: A wireless sensor network system is deployed in area $\mathcal{A}$ (e.g., for the surveillance of a critical infrastructure). 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

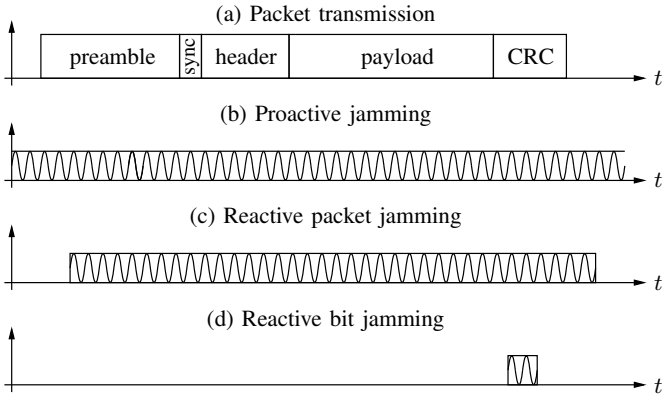

Fig. 1. Jamming types. (b) Proactive jammers keep the channel permanently occupied so that no transmissions are possible whereas reactive jammers only jam once an ongoing transmission has been detected. (c) With reactive packet jamming, the attacker emits the jamming signal as soon as the transmission 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.

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 1(b)) or reactively jam an alarm message once it is sent (Figure 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^i and beam width θ_j^i for a set $\{(P_j^1, \theta_j^1), (P_j^2, \theta_j^2), \ldots, (P_j^k, \theta_j^k)\}$ of 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 [18], [25]: 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.

The purpose of our jamming detection scheme is to detect jamming attacks on the message/alarm forwarding process. 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 actual jamming detection can be composed of several iterative detection steps. However, since the same rules and conditions apply to all these steps, we consider only one such detection step and do not further discuss the jamming alarm reporting.

III. Impact and Mitigation of Reactive Jamming

Before we present our solution for the detection of reactive jamming in Section IV, we first 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.

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 [15]. 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 typical MAC protocols for WSNs. 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 minimal exposure for the attacker. 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 2). Even more important, 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.

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

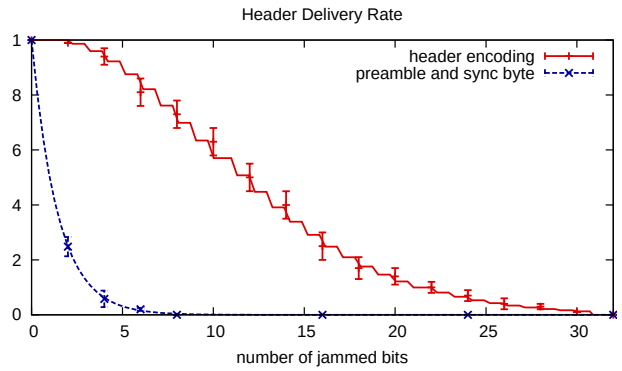

Fig. 2. Header delivery rate for our coding-based header detection algorithm and a common preamble/sync-byte-based approach. The lines show the expected theoretical results, the points the measured results; for each transmission the bit shuffling was based on a new (secret) seed. According to these results, 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.

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 [19]. 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 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 the next section, even such a significantly longer jamming duration does not suffice to recognize jamming by current jamming detection schemes.

IV. 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 Section VII). 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: Firstly, 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. Secondly, assessing an accurate PDR is not practical in a reactive forwarding scheme as 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 [18], [10], [17]. 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.

[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 Section V.

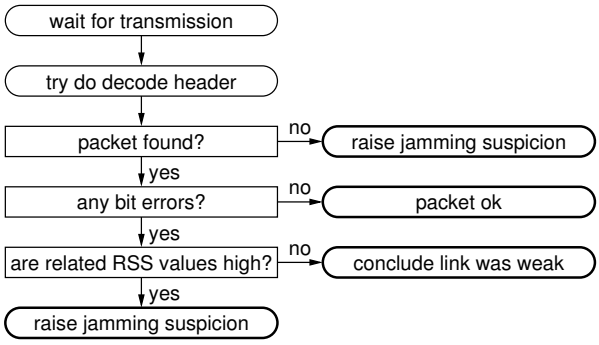

Fig. 3. Enhanced packet reception and jamming detection. Once a transmission signal is detected, the receiver tries to decode the presumed packet header. If it fails (i.e., if there is no transmission although the channel is busy) this might indicate (proactive) jamming and the sequential jamming test is updated. If the received packet contains bit errors, the root cause of this errors is analyzed and either the jamming test updated or the packet ignored.

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, 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 1): (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 3.

Algorithm 1 Jamming detection algorithm

```
RESETJAMMINGTEST()
while true do
    (e, s) := GETERRORSAMPLE()                    (A)
    x := DETECTINTERFERENCE(e, s)                 (B)
    y := UPDATEJAMMINGTEST(x)                     (C)
    if y = jamming then
        raise jamming suspicion
    else if y = no jamming then
        RESETJAMMINGTEST()
    else
        do nothing as we need more evidence
    end if
end while
```

A. 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, 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 most 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 detect 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 errorcorrecting 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 II. This method leverages the link redundancy (wired/wireless) provided by these n-tuples. Note that for simplicity, we assume 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.

Algorithm 2 Error Sample Acquisition

```
function GETERRORSAMPLE()
    while true do
        receive (m_1, s_1) by wire
        if the related packet (m_2, s_2) has already been received by radio then
            ∀i : e[i] := m_1[i] ⊕ m_2[i]
            ∀i : s[i] := min{s_1[i], s_2[i]}
            return (e, s)
        else if neighbor in the tuple will send it next then
            receive m_2 by radio and record RSS into s_2
            ∀i : e[i] := m_1[i] ⊕ m_2[i]
            ∀i : s[i] := min{s_1[i], s_2[i]}
            return (e, s)
        end if
    end while
end function
```

B. 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. The main intuition behind this approach 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.

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

[2]Since all traffic is encrypted, the attacker cannot distinguish probe from alarm messages and has to jam every message it detects.

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

Algorithm 3 Interference Detection

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

C. 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) [26] 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 [27]. 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., $0 \leq p \leq p_c$, hypothesis H_0) is then $p_0(k) := \int_{p=0}^{p_c} \binom{k}{q}p^q(1-p)^{k-q}\,\mathsf{d}p$; the marginal likelihood that there was jamming (i.e., $p_c \leq p \leq 1$, hypothesis H_1) is $p_1(k) := \int_{p=p_c}^{1} \binom{k}{q}p^q(1-p)^{k-q}\,\mathsf{d}p$. Hence, the log-likelihood ratio for H_0 and H_1 after k samples is

$$\eta(k) = \log\frac{p_1(k)}{p_0(k)} = \log\frac{\int_{p=p_c}^{1} p^q(1-p)^{k-q}\,\mathsf{d}p}{\int_{p=0}^{p_c} p^q(1-p)^{k-q}\,\mathsf{d}p}. \quad (1)$$

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 4 Jamming Test

```
function RESETJAMMINGTEST
    k := 0; q := 0
end function

function UPDATEJAMMINGTEST(x)
    k := k + 1; q := q + x;
    η(k) := SPRT(k, q)
    if η(k) ≥ log (1−τ_FN)/τ_FP then
        return jamming
    else if η(k) ≤ log τ_FN/(1−τ_FP) then
        return no jamming
    else
        return undefined
    end if
end function
```

V. Performance Evaluation

In this section, we evaluate the performance of the proposed jamming detection techniques. We therefore implemented them and conducted a series of experiments using COTS BTnodes (Atmel ATmega 128L microcontroller @ 8 MHz, Chipcon CC1000 radio) and Tmote Sky sensornodes (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 $\mathrm{I}^2\mathrm{C}$ bus, forming a two-tuple; a detailed performance and cost analysis of the wired communication is presented in Section VI-C.

A compilation of exemplary measurements for an undisturbed, jammed, and weak link between A and C is shown in Figure 4. 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 III). 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 (an 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, we focus in the remainder of this section on bit or byte oriented radios and present 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.

TABLE I

JAMMING DETECTION PERFORMANCE FOR A STRONG LINK: TRUE POSITIVES / FALSE NEGATIVES — FALSE POSITIVES / TRUE NEGATIVES

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

TABLE II

JAMMING DETECTION PERFORMANCE FOR A WEAK LINK: TRUE POSITIVES / FALSE NEGATIVES — FALSE POSITIVES / TRUE NEGATIVES

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

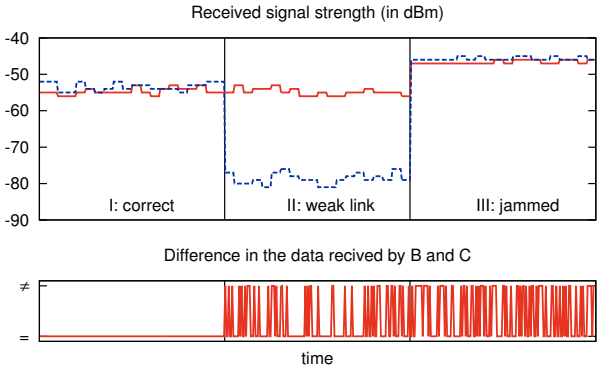

Fig. 4. Sample results obtained with our implementation and a CC1000 radio for three cases. I (adequate links and no jamming): both receivers are able to decode the packets and the packets do not differ. II (weak link from A to C): node C receives incorrect bits and thus the packets do not match; however, 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 III. 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 IV-A. Each technique was evaluated with a series of 1000 undisturbed packet transmissions, five times 2000 packet transmissions were a fraction of 2, 4, 8, 16, or 24 bit was jammed, and three times 2000 transmissions were 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 I and II. The second column shows the results for the case where the received packet is already known by the receiver, that is, the two techniques were 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 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.

A. 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 (1) and solving the equation

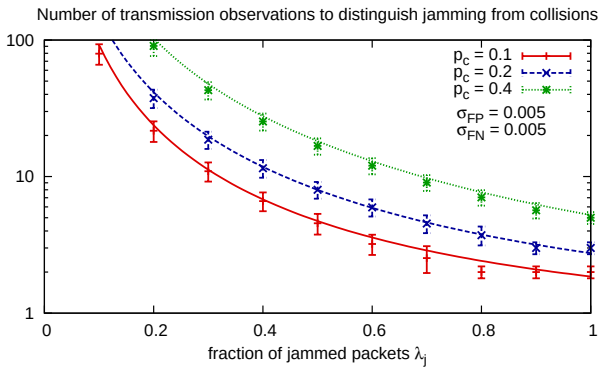

Fig. 5. Performance of the sequential jamming test. We observe that the larger the collision probability p_c and the lower the fraction of packets λ_j that the jammer jams, the longer it takes to detect the jammer; however, the lesser is also the impact of the jammer. If the attacker blocks an alarm (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.

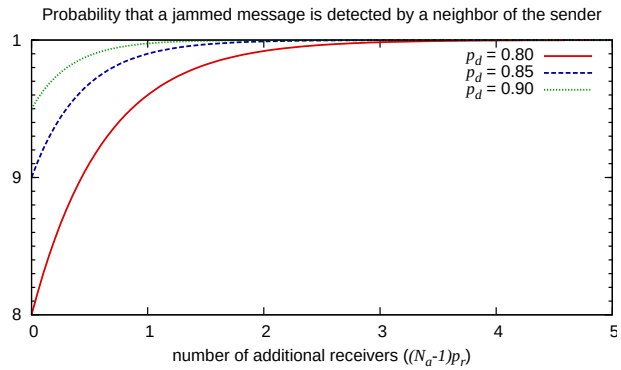

Fig. 6. Probability that a jammed packet is detected by at least one of the nodes in the proximity of the sender. Since the accuracy of the local jamming detection is already fairly high (i.e., $p_d \gtrapprox 0.9$) an overall detection probability of ≥ 0.99 is achieved with only one additional receiver (i.e., if $(N_a - 1)p_r \geq 1$).

$\eta(k) = \log \frac{1-\sigma_{FN}}{\sigma_{FP}}$ for k then yields the expected 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 5. The lines show the theoretical value, the points and σ-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.

B. 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.

1) Monitoring by Unwired Nodes: Ideally, a node would receive and analyze every packet it overhears. However, given the stringent energy constaint 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 [12], 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 6 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 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.

2) Monitoring by n-tuples: As mentioned in Section IV, the jamming detection performance depends not only on the n-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 case of a manual or airdrop-based node deployment where the position (orientation) of the n-tuples is chosen uniformly at random from the deployment area $\mathcal{A}$ (the interval $[0, 2\pi)$). More precisely, the considered n-tuple deployment is as follows:

Let $u_i := (u_{i,1}, u_{i,2}, \ldots, u_{i,n})$ denote an 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 an n-tuple u_i and $1 \leq j < n$, node $u_{i,j}$ is connected to node $u_{i,j+1}$. 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 $p_{i,j} \in \mathcal{A}$.

Given the deployment direction ϕ and the position $p_{i,1}$ 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 $f_D(p_{i,j}|\phi, p_{i,j-1})$. 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 $P_i := (P_{i,1}, R_{i,2}, \Lambda_{i,2}, \ldots, R_{i,n}, \Lambda_{i,n})$ taking values from the set $\{(p_{i,1}, r_{i,2}, \alpha_{i,2}, \ldots, r_{i,n}, \alpha_{i,n}) \,|\, p_{i,1} \in \mathcal{A} \wedge \forall j, 2 \leq j \leq n : 0 < r_{i,j} \leq l_w \wedge \forall j, 2 \leq j < n : -\pi \leq \alpha_{i,j} \leq \pi\}$ according to a random distribution defined by the (joint) probability density function $f_i(p_{i,1}, r_{i,2}, \alpha_{i,2}, \ldots, r_{i,n}, \alpha_{i,n}) = f_P(p_1) f_\Phi(\phi) f_{R,\Lambda}(r_2, \alpha_2) f_{R,\Lambda}(r_3, \alpha_3) \cdots f_{R,\Lambda}(r_n, \alpha_n)$. Here, $f_P(p_1)$ 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). Usually the density function can be approximated using appropriately parametrized (two-dimensional) Beta distributions (scaled to the interval $[0, l_w]$ and $[-\pi, +\pi]$). If $f_P(\cdot)$, $f_\Phi(\cdot)$, 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 $\{P_1, P_2, \ldots, P_m\}$.

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 $B(a, b) = \int_0^1 t^{a-1}(1-t)^{b-1}\,\mathsf{d}t$. In order to assess a realistic parametrization, we conducted experiments using a dummy 3-tuple of 20 m length. 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 $\alpha = 0$, $\sigma \approx 0.14\pi$. In the analytical evaluation we assume for simplicity 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 $l/(n-1)$, where l is the expected length of the n-tuple.

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 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

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

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 8(a)). As illustrated in Figure 8(b), this angle is

$$\alpha_X(d_i, x) := \begin{cases} 0 \text{ if } x < 0 \text{ or } x > r + d_i \text{ or } x + r < d_i, \\ 2\pi \qquad\qquad\qquad\qquad \text{if } 0 \leq x + d_i \leq r, \\ 2 \arccos\left(\frac{x^2 + d_i^2 - r^2}{2 d_i x}\right) \qquad \text{otherwise.} \end{cases} \qquad (3)$$

Hence,

$$\mathsf{P}[X|z = x] = \max_{1 \leq i < n} \alpha_X\left(\tfrac{i}{n-1} l, x\right) \frac{1}{2\pi} \qquad (4)$$

and thus

$$p_X(r, l) = \frac{r^2 \pi}{|\mathcal{A}|} + \int_{x=r}^{r+l} \frac{\max_{1 \leq i < n} \alpha_X\left(\frac{i}{n-1} l, x\right)}{2\pi} \frac{2x\pi \mathrm{d}x}{|\mathcal{A}|} \,. \qquad (5)$$

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

$$p_a \geq 1 - \left(1 - p_X(R(P_j^1), l)\right)^m \,, \qquad (6)$$

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 II), 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

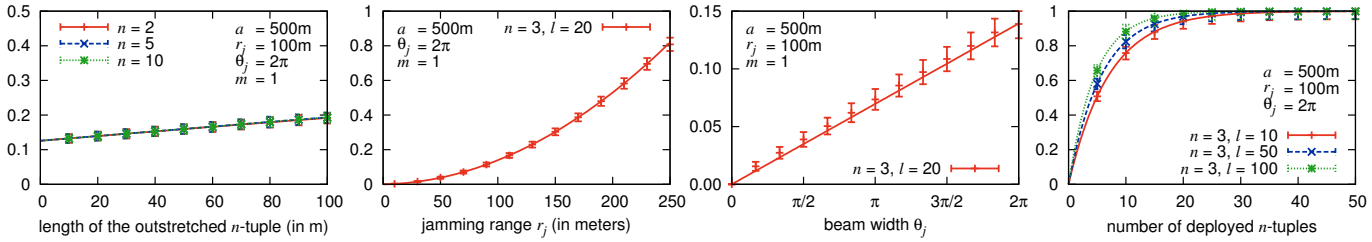

Fig. 7. Probability p_a that the jammed area contains at least one wired node. Full lines show the expected result according to our analysis, the points and σ-confidence intervals the simulation results. We observe that even very long tuples cannot benefit from more than 3 wired nodes per tuple whereas 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 1 km^2 in order that the jamming is monitored by at least one wired node with p_a >95%.

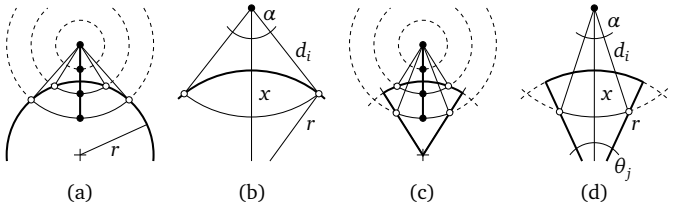

Fig. 8. Geometric relations between the position/orientation of an n-tuple and its possible intersections with a disc or a circular sector of radius r; the figures are not true to scale.

sectors or their extension to a radius of length $R(P_j^i)+l$, this probability can be approximated as

$$p_w \gtrapprox 1 - \prod_{i=1}^{k} \left(1 - \left(1 - \frac{\theta_j^i}{2\pi} p_X(R(P_j^i), l)\right)^m\right) . \quad (7)$$

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 8(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

$$\alpha'_X(d_i, x) := 4 \arcsin(\sqrt{y}/(2d_i)), \quad (8)$$
$$y = r^2 + (x - d_i)^2 - 2\cos(\theta_j^i/2) r (x - d_i)$$

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 8(d)).

2. 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

$$p_Z(r, l) = \int_{x=0}^{\infty} \mathsf{P}[Z|z = x]\mathsf{P}[z = x] \quad (9)$$
$$= \int_{x=0}^{r+l} \mathsf{P}[Z|z = x] \frac{2x\pi \mathrm{d}x}{|\mathcal{A}|} .$$

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

$$\mathsf{P}[Z|z = x] = \overset{2}{\max}_{0 \le i < n} \alpha_X\left(\tfrac{i}{n-1} l, x\right) \frac{1}{2\pi} \quad (10)$$

and thus

$$p_Z(r, l) = \int_{x=0}^{r+l} \frac{\overset{2}{\max}_{0 \le i < n} \alpha_X\left(\tfrac{i}{n-1} l, x\right)}{2\pi} \frac{2x\pi \mathrm{d}x}{|\mathcal{A}|} . \quad (11)$$

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

$$p_p \ge 1 - (1 - p_Z(R(P_a), l))^m , \quad (12)$$

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 7 and 9, 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.

VI. Wire Integrity Protection

Our jamming detection scheme can handle sophisticated jamming attacks as demonstrated in the previous sections. However, it could be compromised by tampering the wired

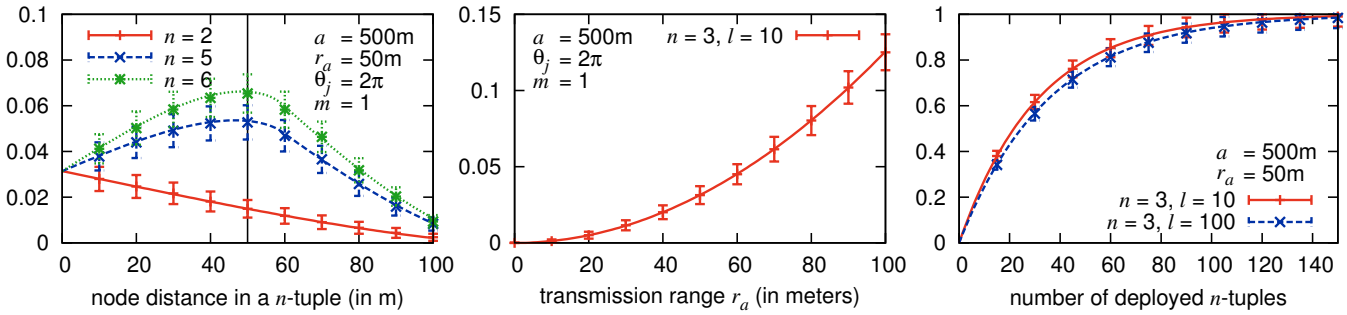

Fig. 9. Probability p_p that the neighborhood of a node is monitored by an n-tuple. We observe that for 2-tuples p_p decreases for longer wires and becomes zero if $l_w >$ twice the transmission range $r_a = R(P_a)$, whereas for $n > 2$ the probability p_p becomes maximal if the distance between two 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%).

connectivity of n-tuples. For example, an attacker with physical access to the network could disconnect the wires, even if the sensor nodes themselves are tamper-resistant or not accessible. In such a way, she could increase her chances to successfully jam the system alarms without being detected.

In particular, our scheme is vulnerable to four different attacks: disconnection, bridging, wiretapping and wire removal. In a disconnection attack the attacker achieves permanent disruption by cutting or unplugging. In a bridging attack, the attacker inserts a rogue (sensor) node between two wired sensor nodes. She therefore can control the wire and stay transparent to the system. In a wiretapping attack, the attacker instruments a wiretap (e.g., by direct electrical connection or by induction) and eavesdrops the communication. It can be used by the attacker to monitor if her malicious activity has been detected. A wire removal attack consists of removing wires or capturing entire n-tuples which is similar to a disconnection and classical node capture attack [7] respectively.

Given the vulnerability of our system to attacks on the wire connectivity, we propose a wire integrity verification protocol suitable for energy-constraint devices and analyze its security implications. In Section VI-C, we demonstrate the energy efficiency of our protocol in a real-life implementation.

A. Stream Cipher-based Wire Integrity Verification (SC-WIV)

The SC-WIV protocol provides mutual authentication by exchanging a sequence c of size l bits generated by a synchronized stream cipher. Node/wire compromise is detected by using appropriate local timeouts. The protocol is summarized in Figure 10 where A and B denote the two sensor nodes.

1. Setup: Prior to deployment, A and B are preloaded with a shared symmetric key K_{AB} and an initialization vector IV_{AB}. These values are used to initialize the respective stream ciphers. The local timeouts for wire/node compromise detection, T^A and T^B, respectively, are calibrated before deployment based on measured round-trip times of messages and session frequency. We assume that both nodes A and B enter a special preemptive mode when sending and receiving bits. This mode postpones all other tasks in order to process the protocol messages as quickly as possible. We further assume

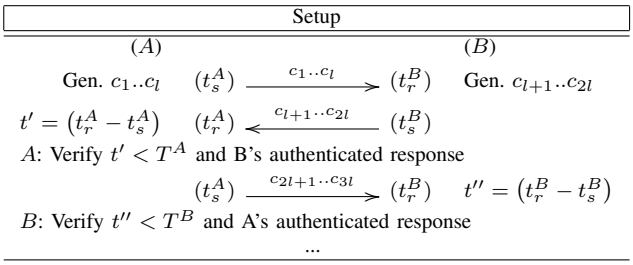

Fig. 10. Single session mutual authentication between A and B based on synchronous stream cipher. Note that the protocol is executed repetitively at regular intervals and that A and B alternate as protocol initiators.

that the stream cipher can generate one random bit upon request for a very long time. Trivium [8] stream cipher can be used for such a purpose: it can be efficiently implemented in hardware and generate one bit per clock cycle (up to 2^{64} bits) from an 80-bit key and an 80-bit initialization vector.

2. Protocol description: A starts the first authentication session as follows: A sends a sequence of l bits to B and records the sending time t_s^A. Upon reception, B sends its response bits immediately and also records the sending time t_s^B. When A receives B's response, it calculates the elapsed time $t' = \left(t_r^A - t_s^A\right)$ and verifies that the sequence of bits is from B and is within the acceptable timeout $t' \leq T^A$. If the verification succeeds, A sends immediately the next sequence of bits to B. On timeout or if the authentication fails, an alarm is raised. Similarly, B receives A's response, calculates the elapsed time $t'' = \left(t_r^B - t_s^B\right)$, verifies that the sequence of bits is correct and with time $t'' \leq T^B$. Again, if the verification succeeds, B schedules to perform the next mutual authentication session after time T^F, where T^F is a shared time between successive protocol rounds.

The SC-WIV protocol is also used to signal whether application data will be transferred between the current and next protocol round: if the received protocol bit sequence c is inverted then data will be transmitted; if the received sequence is regular, there will be no data transmission.

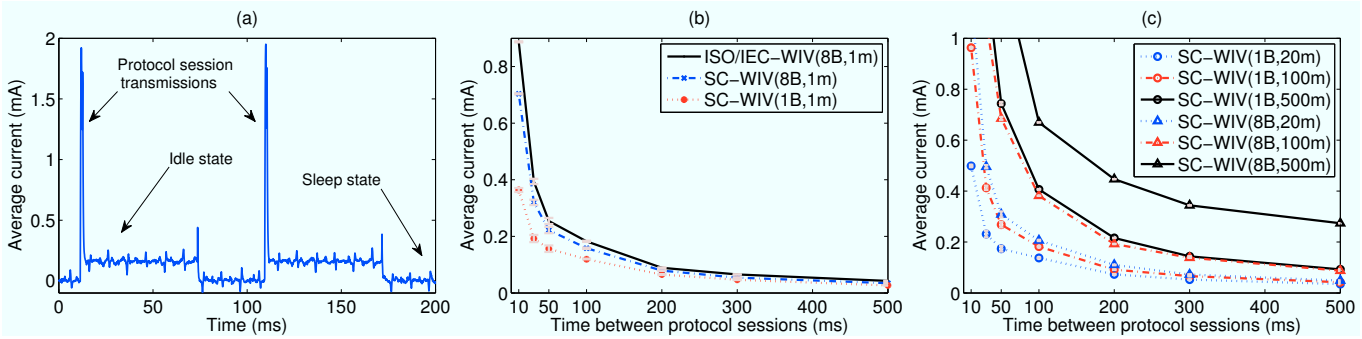

Fig. 11. Energy consumption: a) A typical I^{2}C communication for SC-WIV. Current draw gradually returns to idle and sleep state after a session execution. b) A comparison of the energy consumption of SC-WIV vs. ISO/IEC-WIV. SC-WIV is more efficient as it does not require encryption operations. c) The SC-WIV energy efficiency for different wire lengths and bit sequence sizes.

B. Security Analysis

The presented SC-WIV protocol achieves disconnection and bridging attack. Some types of bridging attacks can still be possible, but will force the attacker to forward the protocol bit sequences. Wiretapping cannot be detected by our protocol but encryption of packets on the application level is an appropriate mechanism in this case.

In the SC-WIV protocol, if the cipher cannot generate bits for a sufficiently long time, an initialization vector renewal procedure must be devised. We acknowledge that the protocol gives the attacker the possibility to guess the sequence of bits and thus to delay detection. More precisely, the attacker must guess one sequence of bits if the protocol session is initiated by the other entity and two sequences of bits when she initiates the protocol session. Therefore, the probability to delay detection for i runs (1 bit exchange) is $2^{-(i+\lfloor \frac{i}{2} \rfloor)}$ and thus decreases exponentially for successive protocol runs.

To summarize, there is a trade-off between energy consumption and timely detection of wire compromise. Optimizing the energy consumption by reducing the number of bits increases the attacker's chances to delay detection. The actual delay gain depends on the frequency of successive protocol runs.

C. Implementation Results

We implemented the SC-WIV protocol using Tmote Sky devices running TinyOS 2.x [2]. We used the pseudo-random number function in TinyOS with shared initialization vector to generate bits, emulating a stream cipher functionality. The I^{2}C bus on the Tmote Sky platform was used for wired communication with shielded and foiled Ethernet cables.

For comparison, we also implemented a standard challenge-response protocol for mutual entity authentication as described in ISO/IEC 9798-2. In this protocol two sensor nodes A and B mutually authenticate themselves at regular intervals and alternate as session initiators. For the random nonce generation, we again used the pseudo-random number function in TinyOS. Symmetric key encryption/decryption was achieved with Skipjack block-cipher adapted from TinySec [14] for the TinyOS platform. We refer to that solution as ISO/IEC-WIV.

TABLE III
WIRE LENGTH VS. CLOCK SPEED

Wire length	Clock frequency	SC-WIV Session Tx Time	
		1 byte	8 bytes
1 m	95 kHz	1 ms	2.2 ms
20 m	39 kHz	2.1 ms	7.5 ms
50 m	21 kHz	3.5 ms	14 ms
100 m	12 kHz	6 ms	22.5 ms
500 m	3 kHz	25 ms	90 ms

Power consumption was measured at 3V using an Agilent 34411A multimeter. Figure 11(b) compares the ISO/IEC and SC-WIV protocols. As ISO/IEC-WIV uses Skipjack block cipher, the minimum size of an authenticated message is the minimum block size (8 bytes). So, we compared 8 bytes payload for both protocols and the SC-WIV protocol with 1 byte—the minimum possible payload in the I^{2}C implementation of the Tmote Sky platform.

From the results, we can see that SC-WIV is more efficient than the ISO/IEC-WIV protocol. This is especially seen at high frequencies (10 to 100 ms intervals). Furthermore, SC-WIV with 1 byte dramatically reduces the power consumption. Decreasing the ping frequency makes the average power consumption to converge towards the sensor idle/sleeping state power consumption. This is consistent as the sleeping state will be longer for lower frequencies and thus will significantly affect the average consumption (see Figure 11(a)).

It should be noted that the current I^{2}C specification minimum packet size consists of 7 bits address + 1 byte payload. Thus, SC-WIV protocol could be optimized to transmit one bit by changing the I^{2}C protocol. This would further improve the power consumption of the protocol.

Additionally, we tested SC-WIV over wires of 5, 20, 50, 100, and 500 meters length. (see Figure 11(c)). We found that I^{2}C can effectively be used for communication over longer wires. However, long distance wires significantly increase the energy consumption for both high (10 to 100 ms) and low (300 to 500 ms) frequencies. These findings also motivate the design of protocols requiring single bit transmissions. The higher power consumption for longer wires is due to the

decrease of clock speeds in order to compensate the resistance and capacitance in longer wires. Table III summarizes the clock speeds and transmission times required in the SC-WIV protocol session execution for 1 and 8 bytes.

VII. Related Work

The detection and mapping of jammed areas in the realm of wireless sensor networks has been studied by Wood, Stankovic, and Son [24]. 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 [25]. 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 propose two jamming detection schems based on the PDR, the bad packet ratio (BPR), and the energy consumption amount (ECA) of the radio [5]. 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 [18], [10], 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 [17]. 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 has been presented by Li, Koutsopoulos, and Poovendran [16]. 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 present optimal jamming attacks as well as network defense policies with respect to detection and notification time. Other than 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, other than our work none of the above mentioned schemes considers overshadowing, where the original packet is covered by a (maliciously inserted) second message.

The application of additional infrastructure in the form of wired short-cuts has been proposed before by Chitradurga and Helmy [6] as well as by Sharma and Mazumdar [20] with an objective to improve the energy efficiency of wireless networks. Čagalj, Čapkun, and Hubaux showed how wired node pairs can be used to build a wormhole in order to establish communication out of a jammed area [22]. The main difference between our and their work is that theirs does not consider the use of wired tuples for jamming detection; they did neither consider the security of the wired links nor evaluated to what extent the proposed wirings are feasible.

VIII. Conclusion

In this work, we 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 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). We further analyzed the threats on limited wiring and developed a low-power wire compromise detection scheme for the detection of malicious attacks on wires. The presented protocols and algorithm were evaluated analytically, by simulations, and experimentally on COTS BTnodes and Tmote Sky nodes. Since our scheme can operate without introducing additional wireless network traffic, it also meets the high energy efficiency demand of reactive surveillance applications. To the best of our knowledge, this work is the first to present a jamming detection scheme for sensor networks that allows for the detection of advanced (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.

References

[1] BTnodes. http://www.btnode.ethz.ch/.

[2] Tiny OS 2.x. http://www.tinyos.net/.

[3] Tmote Sky. http://www.sentilla.com/moteiv-endoflife.html.

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

[5] Murat Çakiroğlu and Ahmet Turan Özcerit. Jamming Detection Mechanisms for Wireless Sensor Networks. In *Proceedings of the 3rd International Conference on Scalable Information Systems (InfoScale)*, pages 1–8, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

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

[7] Mauro Conti, Roberto Di Pietro, Luigi Vincenzo Mancini, and Alessandro Mei. Emergent Properties: Detection of the Node-capture Attack in Mobile Wireless Sensor Networks. In *Proceedings of the first ACM Conference on Wireless Network Security (WiSec)*, pages 214–219, New York, NY, USA, 2008. ACM.

[8] C. De Cannière. Trivium: A Stream Cipher Construction Inspired by Block Cipher Design Principles. In *Proceedings of the 9th International Conference (ISC)*, 2006.

[9] 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 4th International Symposium on Information Processing in Sensor Networks (IPSN)*, page 70, Piscataway, NJ, USA, 2005. IEEE Press.

[10] D. Gamma. *EW101: A First Course in Electronic Warfare*. Artech House, 2001.

[11] Yu Gu and Tian He. Data Forwarding in Extremely Low Duty-cycle Sensor Networks with Unreliable Communication Links. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 321–334, New York, NY, USA, 2007. ACM.

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

[13] 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 (TOSN)*, 2(1):1–38, 2006.

[14] Chris Karlof, Naveen Sastry, and David Wagner. TinySec: A Link Layer Security Architecture for Wireless Sensor Networks. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 162–175, New York, NY, USA, 2004. ACM.

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

[16] Mingyan Li, I. Koutsopoulos, and R. Poovendran. Optimal Jamming Attacks and Network Defense Policies in Wireless Sensor Networks. In *Proceedings of the 26th IEEE Conference on Computer Communications (INFOCOM)*, pages 1307–1315, May 2007.

[17] Guevara Noubir and Guolong Lin. Low-power DoS attacks in data wireless LANs and countermeasures. *SIGMOBILE Mobile Computing and Communications Review*, 7(3):29–30, 2003.

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

[19] Joseph Polastre, Jason Hill, and David Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *Proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 95–107, New York, NY, USA, 2004. ACM.

[20] Gaurav Sharma and Ravi Mazumdar. Hybrid Sensor Networks: A small World. In *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 366–377, New York, NY, USA, 2005. ACM.

[21] 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 (DCOSS)*, volume 4549/2007 of *Lecture Notes in Computer Science*, pages 64–81. Springer Berlin / Heidelberg, 2007.

[22] Mario Čagalj, Srdjan Čapkun, and Jean-Pierre Hubaux. Wormhole-Based Antijamming Techniques in Sensor Networks. *IEEE Transactions on Mobile Computing*, 2007.

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

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

[25] Wenyuan Xu, Wade Trappe, Yanyong Zhang, and Timothy Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, pages 46–57, New York, NY, USA, 2005. ACM.

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

[27] G. Zhou, T. He, J.A. Stankovic, and T. Abdelzaher. RID: Radio Interference Detection in Wireless Sensor Networks. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 891–901, 2005.