DEEJAM: Defeating Energy-Efficient Jamming in IEEE 802.15.4-based Wireless Networks

Anthony D. Wood, John A. Stankovic, and Gang Zhou
Department of Computer Science, University of Virginia
Email: {wood | stankovic | gzhou}@cs.virginia.edu

***Abstract*—Jamming is a very effective denial-of-service attack that renders most higher-layer security mechanisms moot—yet it is often ignored in WSN design. We show that an interrupt jamming attack is simple to perpetrate in software using a MICAz mote, is energy efficient and stealthy for the jammer, and completely disrupts communication. Solutions are needed to mitigate this insider threat even if more powerful attackers are not thwarted.**

We present DEEJAM, a novel MAC-layer protocol for defeating stealthy jammers with IEEE 802.15.4-based hardware, to address this problematic area. It layers four defensive mechanisms to hide communication from a jammer, evade its search, and reduce its impact.

Given the difficulty of modeling the physical layer accurately in simulation, we evaluate DEEJAM instead on the MICAz mote. We show the performance of the protocol against successively more complex attacks: interrupt jamming, activity jamming, scan jamming, and pulse jamming. Results show that DEEJAM defeats the otherwise devastating interrupt jammer, and achieves a packet delivery ratio of 88% in the presence of a pulse jammer.

To the best of our knowledge, this work is the first to confront multiple types of jamming on common WSN hardware with solutions that are shown empirically to enable continued communication despite an ongoing attack.

I. Introduction

Denial-of-service from jamming is difficult to prevent with the limited resources available to most ad hoc and wireless sensor network (WSN) nodes [1]. Nodes may be static once deployed, and have fixed energy reserves. Radio transmission is an energy-expensive operation, yet an attacker can easily interfere with it.

The military has long dealt with jamming [2] by using spread-spectrum communication [3]. However, the resources required for traditional defenses and the threats in warfighting are at odds with the constraints in WSNs.

Previous-generation wireless sensor networks used single-frequency radios and are defenseless against narrowband noise, whether unintentional or malicious. For example, the Chipcon CC1000 [4] transceiver on Mica2 and prior motes [5] operates at 433 or 900MHz.

The more recent MICAz [6] and Telos [7] motes use the Chipcon CC2420, which operates at 2.45 GHz, is IEEE 802.15.4 [8] compatible, and uses direct-sequence spread spectrum to reduce vulnerability to noise. The Intel iMote [9] includes Bluetooth [10], which uses frequency-hopping spread spectrum.

These uses of spread spectrum reduce the impact of narrowband noise on communication, such as that from microwave ovens and other wireless networks. However, they do not defeat an adversary with knowledge of the spreading codes or hopping sequence. Since these are either standardized (in IEEE 802.15.4) or derived from node addresses (in Bluetooth), they are not secret.

While it is not likely that resource-constrained WSNs will be able to resist a well-funded, powerful wide-band jammer, we believe the bar has been left intolerably low. We show that an attacker able to compromise a WSN node can soley through software cause a devastating denial-of-service. This interrupt jamming attack is energy efficient and stealthy, since it only jams when necessary. Further, the attacker's microprocessor can sleep until the message is detected via an interrupt.

As WSNs move from the lab and controlled environments into public spaces, their exposure to all kinds of security attacks grows. Defenses against such easily mounted jamming attacks are needed to redress the existing security imbalance, even if solutions are not perfect or do not address all classes of attackers.

We present DEEJAM, a protocol for defeating energy-efficient jamming in networks based on IEEE 802.15.4-compatible hardware. It uses four defensive mechanisms together to defeat or diminish the effectiveness of jamming by attackers in the same capability class as network nodes. Each additional defense addresses a different jamming attack. The end result is a novel protocol that allows network nodes to continue to operate—and communicate—in the presence of a jammer. We evaluate DEEJAM on an embedded platform, the MICAz mote, rather than in the idealized environment of a simulator.

Previous solutions focus on the difficult problem of detecting jamming, make burdensome assumptions about node mobility or capabilities, do not address sporadic jamming, or are evaluated only in simulation. To the best of our knowledge, this work is the first to directly confront multiple types of jamming on common WSN hardware with solutions that are shown empirically to allow nodes to continue to communicate despite an ongoing denial-of-service attack.

Our main contributions of this work include:

- The definition, implementation, and evaluation of four jamming attack classes: interrupt jamming, activity jamming, scan jamming, and pulse jamming. We show their efficacy in disrupting communications in the wireless network, as well as their relative efficiency for the attacker.
- Four complementary solutions—frame masking, channel hopping, packet fragmentation, and redundant encoding—that together significantly reduce the probability of a successful jamming attack. Despite a pulse jammer corrupting an entire channel, DEEJAM maintains a packet delivery ratio of 88%.
- Development and evaluation of an integrated protocol on the MICAz platform. Since radio communication is notoriously difficult to simulate accurately, we show results from a real-world embedded implementation on commonly available WSN hardware.

After describing related work, we give our assumptions about the network and attackers in Section III. Then we describe DEEJAM in detail, incrementally presenting attacks and corresponding defenses to make the rationale for each more clear. We present implementation details followed by results from evaluation on the MICAz mote in Sections V–VI, and finally conclude.

II. Related Work

Xu et al. [11] propose channel hopping and physically moving away from a jammer in Mica2 networks (CC1000 based). However, their focus is on methods for determining when jamming is occurring (as in later work [12]), rather than avoiding it altogether, and they do not address channel hopping overhead or the scanning jammer attack. Also, physical evasion ("spatial retreats") requires node mobility that may be too energy consuming for sensor networks. In contrast, we address an attack that is difficult to detect: interrupt jamming.

Wood et al. described various denial-of-service attacks against WSN nodes, including jamming [13]. One proposal was to cope with jamming, rather than attempt to defeat it. JAM [14] uses a distributed protocol to map a jammed region so the network can avoid it.

Law et al. evaluate the effect of link-layer jamming on the S-MAC protocol [15]. They propose data blurting and schedule switching as countermeasures. Evaluation is primarily through simulation, whereas we have an implementation on mote-class hardware.

Bluetooth [10] uses frequency hopping across 79 channels in the ISM band. The sequence and phase are determined by the master's address and clock. Discoverable devices' addresses are provided to inquirers, and so are vulnerable to jamming attacks, even if higher-layer authentication and pairing operations are not possible. Other work uses multiple frequencies [16], [17], but does not address the impact of jamming attacks.

Čagalj et al. [18] propose using wormholes to exfiltrate events during jamming and give analytical and simulation results for wired wormholes, frequency-hopping pairs, and uncoordinated channel hopping.

Error correction schemes are numerous, though many assume errors are independent, random, and sparse, not the case with an active adversary. Low density parity codes (LDPC) [19] have been proposed for jamming countermeasures in 802.11 wireless LANs, but may not be suitable for WSNs with short packet lengths.

III. Assumptions and Attack Model

We assume that multiple, non-interfering (orthogonal) radio channels are available for dynamic selection by software. IEEE 802.15.4 provisions 16 channels separated by 5 MHz, and the Chipcon CC2420 allows dynamic selection of the same.

Jamming is a denial-of-service attack at the physical layer that uses intentionally interfering radio communication to disrupt the reception of messages at another node (see Figure 1).

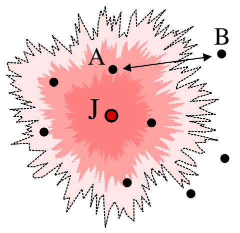

Fig. 1. Node J jams reception at neighbor A.

Our attacker uses the same or similar hardware as WSN nodes in terms of capability, energy capacity, and complexity. More powerful attackers with specialized hardware can certainly jam WSN nodes—perhaps the entire network. In that case, other approaches may be more appropriate, such as detecting the affected area and avoiding it [14], [11].

An attacker is therefore power limited and wishes to avoid jamming continuously, which quickly drains power. We call this on-demand jammer "energy-efficient" if it is successful. The attacker further desires to avoid detection when possible, so it uses the least intrusive (most stealthy) jamming method that is effective.

We assume a mechanism exists for pairwise secret key agreement, whether pre- or post-deployment [20], [21]. We use the pairwise shared key K_N between sender and receiver to generate other keys, which are used with a cipher in ECB[1] mode to generate pseudo-random sequences.

A secure time synchronization service, such as TinySeRSync [22], is necessary for coordinated frame masking and channel hopping, as discussed by So et al. [23]

In this paper, we focus on the operation of DEEJAM between a single sender and receiver, such as between a cluster leader and sensor node in a neighborhood.

IV. DEEJAM: Defeating Jamming

The goal of DEEJAM is to reduce the impact of a jammer on packet loss/corruption so the network can operate while an attack is ongoing. Even without an attacker, communication in wireless networks suffers from signal fading and reflections, interference from other networks, congestion, and environmental noise [24], [25], [26]. Adding a malicious attacker makes this situation worse, and we do not expect to be able to achieve perfect reception with any protocol.

Despite these difficulties, DEEJAM is able to recover from much of the packet loss caused by jamming. Our general design approach for the protocol is to hide messages from a jammer, evade its search, and reduce the impact of messages that are corrupted anyway.

We believe the overhead of our defenses is worth the resulting performance it gives, which allows continued operation. Our protocol requires that the jammer increase its effort substantially to continue to cause disruption, which also increases the opportunity for finding and removing it by external means. By contrast, we show that the neighborhood of a jammer is rendered completely unusable when no defenses are used.

Security is often an escalating scenario of attack and defense, with each side adapting to the strategy of the other. Hence, we start with a very stealthy jamming attack, then incrementally describe defenses and counterattacks. Each builds upon the previous defenses/attacks, until the final DEEJAM protocol combines all the defenses described, and addresses all the attacks.

[1]In Electronic Code Book (ECB) mode, each block is encrypted separately. Its use for message encryption is not recommended; however, we use it only to generate a pseudo-random sequence.

A. Attack 1: Interrupt Jamming

Since radio transmission is an expensive operation, a continuously transmitting jammer will rapidly run out of energy. Although this will completely disrupt communication locally, it may not last long enough to impair the network over the long-term, and is also not stealthy. Network nodes can use energy-conserving modes to attempt to outlast the jammer, or can locate and isolate the affected area.

A better strategy for the attacker is *Interrupt Jamming*, in which it transmits only when valid radio activity is signaled from its radio hardware. At other times, the attacking device enters sleep states while its radio passively listens.

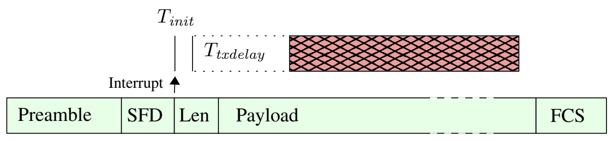

Fig. 2. Interrupt jamming of packet, triggered by SFD reception.

Figure 2 shows a typical physical-layer frame being transmitted (bottom). A multi-byte preamble and Start of Frame Delimeter (SFD) sequence precede the PHY frame. The length of the payload, the payload itself, and a frame check sequence follow.

During normal reception, the radio constantly scans for a preamble and SFD, which indicate that a packet follows. When the SFD is detected, an interrupt is raised in the microcontroller to trigger reading of packet contents from the radio's buffer.

In the interrupt jamming attack, after initialization of necessary state or radio chip registers (taking T_{init} time), the SFD interrupt handler sends a transmit command to the radio. A jamming packet is transmitted to interfere with the legitimate packet in the local area, after a delay of $T_{txdelay}$ imposed by the radio hardware for switching on transmitter circuits and oscillator stabilization.

The attacker continues to jam for the duration of a normal packet for the given network. If messages in the network use an integrity checksum (CRC, FCS, or MIC),[2] the jamming transmission can be much shorter than an entire packet length. Only the minimum amount necessary to overcome encoding redundancy and to corrupt the checksum is required for other receivers to discard the entire packet.

[2]Common MAC-level integrity checks include a Cyclic Redundancy Check (CRC), Frame Check Sequence (FCS), and cryptographic Message Integrity Code (MIC).

Interrupt jamming causes other receivers in the neighborhood of the attacker to receive corrupted messages. Instead of a constant energy drain, the jammer uses energy proportional to the number of messages overheard in the local neighborhood. If a FCS is used, energy savings are even greater. Whereas legitimate senders must transmit an entire packet, a jammer can transmit only enough bits to invalidate the FCS.

The attacker can further save energy by selectively jamming messages of interest. Instead of immediately transmitting, it reads the message header and decides whether to jam based on its contents.

Current WSNs are vulnerable to this interrupt jamming attack, which is simple to implement, stealthy, energy efficient, and completely effective in the neighborhood of a jammer.

B. Defense 1: Frame Masking

A receiving radio normally searches for symbols that conform to the spreading code and modulation used in the network. A packet's multi-byte preamble (see Figure 2) provides time for the receiver to synchronize on 4-bit symbols. A specific byte-pattern, the Start of Frame Delimeter (SFD) provides byte synchronization for the receiver and indicates that message contents follow, starting with the length byte.

In the *Frame Masking* defense, a sender and receiver agree on a secret pseudo-random sequence for the SFD in each packet. Unless the attacker's radio is configured to search for the correct SFD, no interrupt will be signaled to begin jamming.

We use a pairwise shared key K_N to generate an SFD key $K_S = E_{K_N}(0)$. This is used to generate a pseudo-random SFD sequence $\mathcal{SS} = \{E_{K_S}(i) \bmod 2^l\},\ i \geq 0$, where l is the length in bits of the SFD code.

To use efficient interrupt jamming, an attacker must therefore guess the proper SFD. A 16-bit SFD, reserving two special values from each 4-bit symbol,[3] gives $14^4 = 38,416$ choices, or less than a 1 in 2^{15} chance of guessing correctly. A single-byte SFD gives 196 choices, or less than 1 in 2^7 chance of random selection.

It is not sufficient for sender and receiver to merely choose a single SFD, since the attacker may discover it by searching the relatively small space of codes. The pseudo-random sequence guarantees that even if the attacker detects one message, the next will use a different SFD. Hence, the attacker cannot rely on the radio hardware to provide a low-cost, interrupt-based notification of radio activity.

[3]On the Chipcon CC2420, SFD nibbles of 0 and 15 have special significance, so we exclude them from calculations to be conservative.

C. Attack 2: Activity Jamming

Since the attacker's radio no longer detects the unknown SFDs, it will continue searching but will not receive the message. To detect activity the attacker must periodically sample the radio signal strength indicator (RSSI) or, if available, the radio's clear-channel assessment (CCA) output. CCA output indicates when the RSSI is above a programmable threshold, and/or when the modulation and spreading characteristics of the received signal are compatible with those of the network. The RSSI or CCA reported by the radio is typically the average of received radio energy over a short period (8 symbols, or 4 bytes on the Chipcon CC2420).

In contrast to interrupt jamming, this *Activity Jamming* attack has higher false positives from channel noise[4] and greater latency. Since detection is less accurate, the jammer must transmit more often, using energy and incurring additional risk of detection.

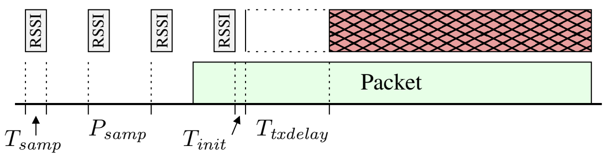

Fig. 3. Activity jamming attack, where RSSI in a single channel is periodically sampled and jamming begins upon packet detection.

Figure 3 shows an attacker sampling RSSI every P_{samp} units. Each sample takes T_{samp} time units. When activity is detected, jamming is initiated as described in Attack 1, with a full or partial transmission. The attacker then returns to activity sampling, waiting on channel activity to be detected before jamming again. Communication between sender and receiver is disrupted, though at a higher cost to the jammer.

D. Defense 2: Channel Hopping

The 2.45 GHz Industrial, Scientific, and Medical (ISM) band is 80 MHz wide, and IEEE 802.15.4 specifies 16 channels with 5 MHz separation. Since an attacker can only sample RSSI for a channel on which its radio is listening, in the *Channel Hopping* defense, the sender and receiver change channels to evade the jammer.

[4]Empirical observations indicate that CCA is frequently triggered by background noise, as well as by competing uses of the channel by WLAN devices, cordless phones, and microwaves. This may be mitigated by adjusting the RSSI threshold, but it is difficult to accurately tune for both near and far transmitters.

The parties use their shared key K_N to create a channel key $K_C = E_{K_N}(1)$, which generates a pseudo-random channel sequence $\mathcal{CS} = \{E_{K_C}(i) \bmod C\}$, $i \geq 0$, where C is the number of channels available in the band. Message M_i is transmitted on channel C_i, which is unknown to any but the two parties involved.

The probability of an attacker selecting the correct channel at random is $1/C$ or $1/16$ in IEEE 802.15.4. By itself, this gives a relatively high chance of success to the attacker. However, successful jamming of one message gives no advantage for the next, since it will be on a different, unknown channel.

Also, using multiple frequencies allows for greater transmission multiplexing, which increases network throughput and diminishes the effect of a single jammer.

In our implementation, we use a time-based hopping scheme, so that a network node that desires to send a message to a neighbor knows what channel to use (where the receiver will be listening).

E. Attack 3: Scan Jamming

To defeat channel hopping, the attacker must hop faster than normal network nodes, scanning for activity on every channel. If the attacker can scan all channels in the entire band in less time than it takes to transmit a single packet, it can jam the bandwidth equivalent of an entire channel. The other $C - 1$ channels (or their equivalent) are unmolested.

In this *Scan Jamming* attack, the attacker samples each channel as briefly as possible (T_{samp}), determining whether activity is present (as in activity jamming). If so, the node begins jamming; otherwise it hops to the next channel, continuing the scan. The time to scan each channel is therefore: $T_{scan} = (T_{hop} + T_{samp})$.

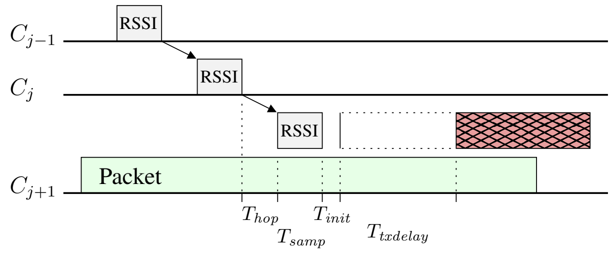

Fig. 4. Channel scanning for activity, followed by jamming of the message found on C_{j+1}.

Figure 4 shows an attacker scanning channels C_{j-1} to C_{j+1}, sampling RSSI at each. In this example, activity is found at C_{j+1}, so the attacker begins jamming after the requisite delay ($T_{init} + T_{txdelay}$). The jammer's transmission overlaps the legitimate message, likely causing nearby nodes to receive a partially corrupted frame.

Messages are always jammable if:

$$\frac{T_{pkt} - (T_{init} + T_{txdelay})}{T_{scan}} > C \tag{1}$$

That is, the attacker can scan the available channels C quickly enough to find and jam a message before the sender is through transmitting it (which takes T_{pkt} time). Whether it is possible depends on the number C, and the time required to scan and initiate a jamming transmission.

Since nodes choose the channel uniformly at random from C, we can also calculate the probability $\mathcal{P}$ of successful scan jamming:

$$\mathcal{P} = \min\left(\frac{T_{pkt} - (T_{init} + T_{txdelay})}{C \cdot T_{scan}},\ 1\right) \tag{2}$$

Activity detection depends critically on the sensitivity and RSSI or CCA thresholds used by the attacker. If activity detection is less than perfect, the jamming probability of Equation 2 is scaled by a detection probability D. The final proportion of jammed messages is expected to be $D \cdot \mathcal{P}$, somewhat less than ideal.

F. Defense 3: Packet Fragmentation

From Equation 2, we see that scan jamming is most successful if messages are long enough to be found and jammed before they are completely transmitted. In the *Packet Fragmentation* defense, a node breaks an outgoing application payload into fragments to be transmitted separately, on different channels and with different SFDs. The last fragment contains a FCS for the entire payload.

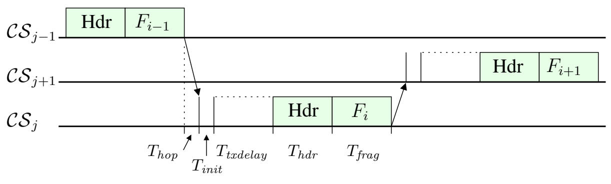

Fig. 5. Channel hopping, with packet fragments transmitted on channels $\{\mathcal{CS}_{j-1}, \mathcal{CS}_j, \mathcal{CS}_{j+1}\}$ known only to the sender and receiver.

Figure 5 shows how fragments are transmitted. The transmitter and receiver use the channel and SFD sequences as before. Fragment F_i is transmitted with the requisite PHY header on channel $\mathcal{CS}_i$ using SFD $\mathcal{SS}_i$.

To transmit fragment F_i, the sender hops to $\mathcal{CS}_i$, fills the transmit FIFO with F_i, sets SFD to $\mathcal{SS}_i$, and issues the transmit command. After a delay, the radio sends the

PHY header and F_i. The time to transmit the fragment is therefore $T_{hop} + T_{init} + T_{txdelay} + T_{hdr} + T_{frag}$.

If the fragments are short enough, an attacker's reactive jamming message does not start until after the sender has finished transmitting and hopped to another channel.

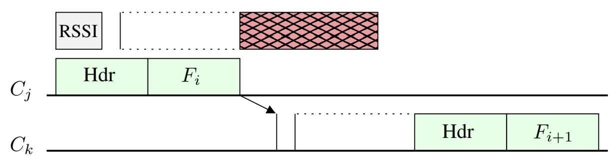

Fig. 6. Jammer detects packet on channel C_j, but sender hops to C_k before jamming can begin, evading the attack.

Figure 6 shows a sender transmitting on channel C_j. Simultaneously, the attacker samples the RSSI of the channel, detects activity, and initiates jamming. Just as the transmission begins, the sender hops to channel $C_k \neq C_j$, and transmits the next fragment F_{i+1}.

If the jammer is on channel C_i and begins to sample RSSI at precisely the moment when the PHY header begins transmission, the minimum jam latency is $T_{jam} = T_{samp} + T_{init} + T_{txdelay}$. Whenever $T_{jam} \geq T_{hdr} + T_{frag}$, the sender has completely defeated on-demand activity and scan jamming. There is not enough time for the jammer to react before the sender hops to another channel. Since the channel sequence $\mathcal{CS}$ is unknown, the attacker must begin scanning again.

G. Attack 4: Pulse Jamming

The attacker is left with few good options for effective jamming if the network uses frame masking, channel hopping, and packet fragmentation together. Since SFD interrupts are very unlikely (due to masking), sampling RSSI is necessary to detect activity. This only works if the attacker knows which channel to sample (defeated by channel hopping), or can quickly scan to find the sender. Even if a packet is detected, with fragmentation the attacker cannot react quickly enough to jam it.

The best remaining strategy for the attacker seems to be jamming continuously or intermittently on a single channel. This avoids the (small) time spent hopping between channels (T_{hop}), during which no jamming would occur. Since fragments must be shorter than $T_{hdr} + T_{frag}$, for small C there is a good chance that any given channel is used by one or more fragments of a packet. Corrupting one of them is sufficient to cause the entire packet to be dropped, since the FCS of the assembled packet will be incorrect at the receiver.

In the *Pulse Jamming* attack, the jammer remains on a single channel, hoping to disrupt any fragment that may be transmitted there (though it does not have time to listen to verify this). Since packets cannot be detected quickly enough to selectively jam, the attacker transmits blindly in short pulses. The jamming pulses must occur no less frequently than $T_{hdr} + T_{frag}$ to prevent any fragments from slipping through.

The shortest transmission possible for the jammer is T_{hdr}, the time for the PHY header only with no payload. This is more energy-efficient than continuous jamming, and requires a transmit duty cycle of $T_{hdr}/(2T_{hdr} + T_{frag})$. Chances of being detected are much higher than if the jammer could use a reactive, rather than blind attack.

H. Defense 4: Redundant Encoding

This final defense mitigates the risk from a pulse jammer disrupting one or more fragments of a packet, since the loss of any one results in the loss of the entire packet. When fragmenting the header, payload, and FCS, a *Redundant Encoding* scheme is used that allows the receiver to recover from one or more corrupted fragments. At the cost of transmission redundancy, network nodes gain robustness to corruptions on a few channels.

For simplicity of implementation and evaluation, we used fragment replication to achieve the desired redundancy. The receiver compares each fragment pair to determine whether they are different. If so, it must recalculate the FCS for the assembled packet using each fragment copy until a correct FCS is found. In the worst case (all fragment pairs differ) this requires 2^F calculations (for F fragments per packet), however corruption of a single fragment on a channel by a jammer often results in the receiver missing the fragment altogether, since the preamble or SFD are corrupt. This reduces the choice to the one good copy, and in practice we expect the number of calculations to be much less than 2^F.

The redundancy and correction overhead may be reduced by using near-optimal erasure codes, which can be efficiently implemented in hardware or software [27]. Our fragment duplication has the equivalent transmission overhead as an erasure code of rate one-half.

We impose a small additional constraint on the channel sequence $\mathcal{CS}$: $\forall i, C_i \neq C_{i+1}$. This ensures that consecutive fragment copies are not transmitted on the same channel, since both are lost if an attacker is pulse jamming there.

I. DEEJAM Summary

At the culmination of the attack, defense, and counterattack scenarios described, the adversary is already pulse jamming an entire channel. Therefore, only the addition of new attack nodes causes further degradation. Enough attackers will of course be able to completely deny service by jamming all channels in the band, but this requires more resources and certainly draws more attention.

Each of the four defense mechanisms layer atop one another to allow operation even in the presence of an ongoing jamming attack. In the final combined protocol operating at the MAC layer,

1) a FCS is computed in software for the entire header and application payload,
2) the combined header, application payload, and FCS are divided into fragments small enough to avoid or reduce jamming,
3) fragments are redundantly encoded with rate R,
4) each fragment is assigned an SFD from the random sequence $\mathcal{SS}$, and
5) each fragment is transmitted on a channel in the band from the random sequence $\mathcal{CS}$.

The incremental presentation of attacks and defenses shows that *channel hopping by itself is not sufficient to defeat jamming*. The attacker can quickly scan the band to find and jam transmissions. For the same reason, we cannot a priori assume that an attacker will pulse jam on a single channel, and simply avoid transmitting there. Rather, the jammer chooses this mechanism only after the defenses we describe preclude other attacks.

V. Implementation

Since radio communication and jamming are inherently stochastic and difficult to model accurately [26], we evaluated DEEJAM on a real embedded platform, rather than in a PC-based simulator. We implemented it in nesC for TinyOS, on a Crossbow MICAz mote. The MICAz has a Chipcon CC2420 radio transceiver that is IEEE 802.15.4 compliant. Table I summarizes terms used in the exposition, their meanings, and values typical in the Chipcon CC2420 and/or TinyOS.

Short fragment lengths are crucial for avoiding scan jamming, so we shortened the PHY frame to the extent possible on the CC2420 by using a four-byte $T_{txdelay}$ (six-byte default) and one-byte preamble (three-byte default). We also trimmed the MAC header, saving five bytes, at the expense of IEEE 802.15.4 compatibility.

Interrupt jamming uses byte-serial receive mode and the FIFOP signal since it is tied to an interrupt line on the microprocessor (the SFD signal must be polled). The Chipcon CC2420's FIFOP signal is normally used to indicate when a programmable amount of data has been received in the radio buffer. Setting its threshold to zero causes it to behave the same as the SFD signal, but with the benefit of triggering an interrupt.

TABLE I
Symbols, meanings, and values typical for TinyOS and/or the Chipcon CC2420 radio. The byte-time unit (B) = 32 μs in IEEE 802.15.4, which operates at 250 Kbps.

Symbol	**Value**	**Meaning**
C	16	number of channels in the 802.15.4 band
T_{pkt}	39B	time to transmit a normal (29-byte payload) TinyOS message, including PHY and shortened MAC headers, and FCS
T_{hop}	132 μs	time to change to another channel's frequency and stabilize (empirical)
T_{samp}	4B	time to sample RSSI or CCA
T_{init}	$>10us$	processing and command-issue overhead
$T_{txdelay}$	4 or 6B	on-radio-chip delay between issuing command and actual transmission start
T_{hdr}	4B	time to transmit a minimal PHY header (preamble, SFD, length)
T_{frag}	5B	time to transmit a fragment payload
T_{jam}	$\approx$ 9B	measured latency for activity jamming

Empirical measurements of the MICAz showed that when scan jamming, CCA always falsely indicates a busy channel for about 132 μs after hopping. Therefore, we use $T_{hop} = 132\,\mu s$ in the scan jamming analysis.

The shortest message that can be transmitted on the Chipcon CC2420 is $T_{hdr} + 1 = 5$ bytes. As a result, the transmit duty cycle for pulse jamming was $(T_{hdr} + 1)/(2T_{hdr} + 1 + T_{frag}) = 35.7\,\%$ with five-byte fragment payloads used in the evaluation.

VI. Evaluation

A triad of MICAz motes were placed in an indoor environment, each separated by two meters from the others and elevated 9 cm from the floor. At the beginning of each test run, the motes were time synchronized. One mote (the "sender") transmitted packets to another mote (the "receiver"), using the defense mechanism under test. A third mote (the "jammer") implemented the attack under test. At the conclusion of each test, all motes reported the results to a PC via a base station.

Each test run lasted 60 seconds, with the sender transmitting 32 messages per second with 29 bytes of payload (39 bytes total). Tests were repeated five times each for a total of 9595 messages, and the averages and 90% confidence intervals were computed. All motes used a transmission power level of -7 dBm. A CCA threshold of -27 dBm was used, and the number of channels was 16, except where noted otherwise.

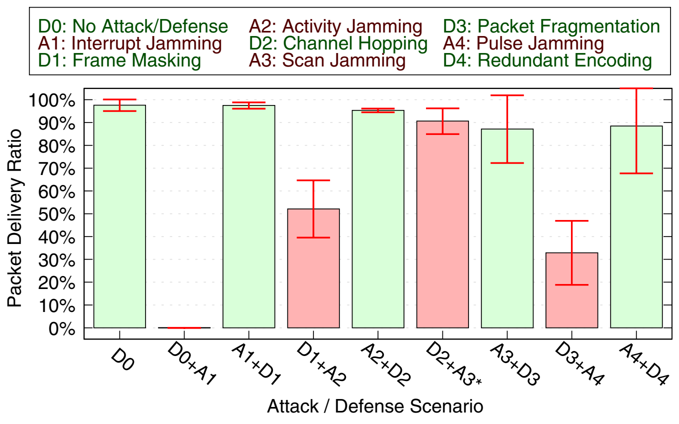

Fig. 7. Performance of DEEJAM for defense and attack scenarios.

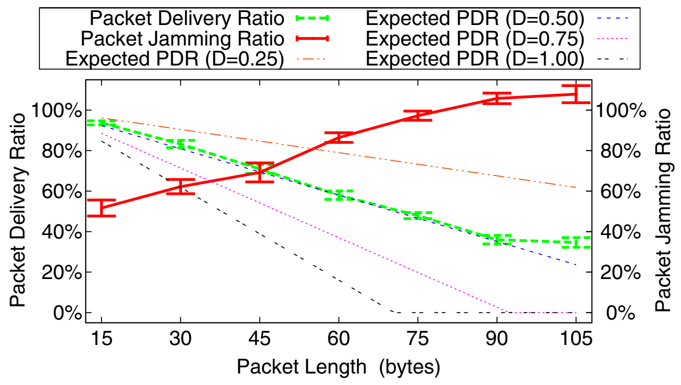

Fig. 8. PDR, PJR, and predicted PDR for a Channel Hopping defense (D2) and Scan Jamming attack (A3), limited to eight channels. 98% confidence intervals are shown.

We measured the packet delivery ratio (PDR) of DEEJAM with combinations of attacks and defenses to show its performance. The packet jamming ratio (PJR) shows the jammer's effort. Packet delivery ratio in the absence of attacks shows the overhead of DEEJAM mechanisms. Overhead from message expansion caused by packet fragmentation and encoding is calculated analytically.

A. Experimental Results: DEEJAM Performance

The primary goal of DEEJAM is to be able to communicate despite an ongoing jamming attack. To demonstrate success, we measured the ratio of the number of packets successfully delivered to the number that were sent (PDR). The results are shown in Figure 7, in the order of attack and defense as described in Section IV.

Baseline PDR with no attack or defense (D0) is shown here for comparison at 98%. The addition of an interrupt jamming attack results in complete disruption of communication between sender and receiver (0% PDR). With a frame masking defense (D1), PDR recovers to nearly 98%, showing that without knowledge of the SFD, the jammer cannot use the efficient interrupt jamming and that our solution is very effective.

When the attacker therefore uses activity jamming (A2), PDR drops to 52%. This may vary as a function of the RSSI or CCA threshold (here -27 dBm) the jammer uses to detect activity. A lower threshold may result in lower PDR (a better attack), but also results in a higher jamming rate due to false positives from noise. Channel hopping (D2) by the sender and receiver recovers a PDR of 95%, since the jammer remains on a single channel.

Figure 7 shows that a scan jamming attack (A3) lowers the PDR to 91%, only slightly less than without the attack. We empirically determined that the CCA pin on the MICAz mote is not valid until 262 μs after changing channels. Using Equation 2 with timing values shown in Table I and $T_{scan} = 262\,\mu s$, we predict the scan jamming attack has a small effect for the default packet size of 39 bytes and with 16 channels. This is because the relatively slow scanning rarely finds an active channel before it is too late to jam it.

To validate this prediction, we halved the number of channels used in this D2+A3 scenario to eight. We varied the packet length from 15 to 105 bytes, and measured the PDR and PJR (see Figure 8). Also plotted is the expected PDR, from Equation 2 for four activity detection probabilities D, from 0.25 to 1.0. We find that the measured PDR ranges from 94% to 35% and tracks the expected PDR very closely when $D = 0.5$. This is consistent with the activity jamming result already discussed, where PDR is 52%. Note that PJR may exceed 100% due to channel noise, i.e., the jammer transmits more frequently than a legitimate sender.

Returning to Figure 7, we see that the use of packet fragmentation (D3) results in a PDR of 87%. With 29-byte message payloads, each packet requires $F = 7$ five-byte fragments, each transmitted on the next channel in sequence. In this experiment, we maintain a transmission rate of 32/second, so the test traffic throughput drops from 1248 Bps to 288 Bps. We predict a stationary pulse jammer (A4) should be able to corrupt almost $F/C = 7/16$ of the packets; the measured PDR is 33%.

Adding the final defense, redundant encoding (D4), allows the receiver to recover from a dropped or corrupted fragment within each pair. PDR improves drastically to 88%, despite the presence of a pulse jammer in the band.

B. Experimental Results: Overhead and Costs

Next, we consider the effort that a jammer must expend to mount the attacks described. Figure 9 shows the

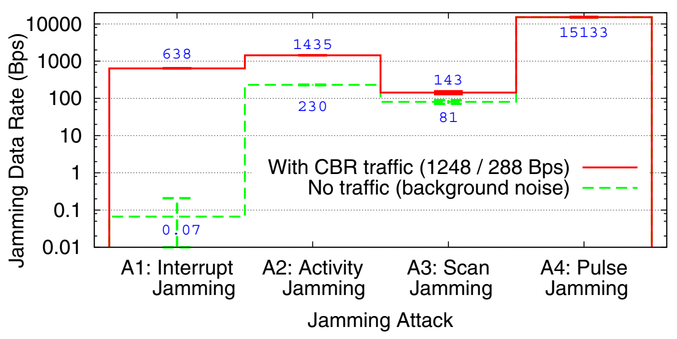

Fig. 9. Effort required for jammer to mount the attacks, with and without traffic, as a jamming data rate in bytes per second (logscale).

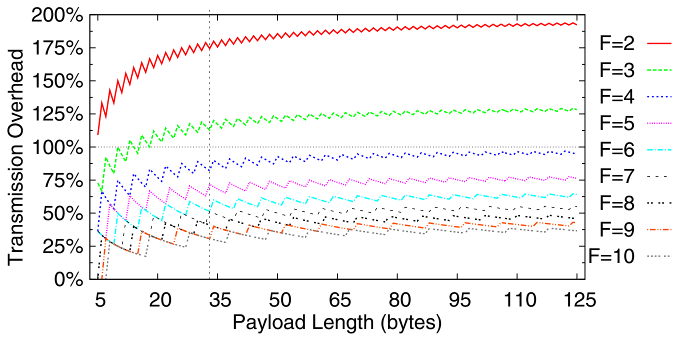

Fig. 11. Overhead due to fragmentation for varying original message payload size (P=33 shown), calculated analytically. Fragment payload size is F. Assumes PHY header is 4 B, FCS is 2 B.

jamming data rate in bytes per second for each attack. We measured the rate both without traffic (attacker responds to background noise) and with the CBR traffic as before.

Interrupt jamming is most efficient, since without traffic an average of .07 Bps are transmitted (due to only one false detection among all tests). The sensitivity (−27 dBm) required to mount activity and scan jamming attacks causes jamming rates of 230 and 81 Bps, respectively, even without traffic due to background noise. Finally, pulse jamming had by far the highest transmission rate of 15133 Bps, a duty cycle of 50%.[5]

Each of the defenses described, though effective for recovering communication in the presence of an attack, affects the maximum PDR achievable. To determine this cost, we measured the PDR for each defense with no attacker present. Results are shown in Figure 10.

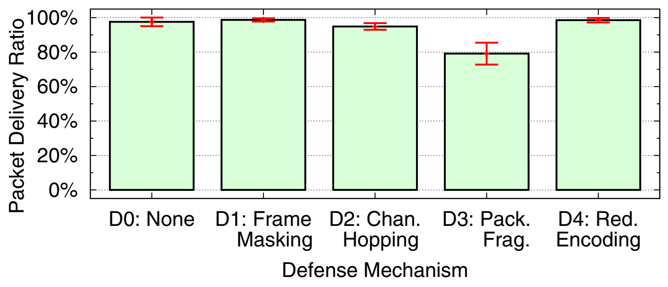

Fig. 10. Performance of DEEJAM defenses with no attacker present.

Packet fragmentation has the most noticeable effect, lowering the PDR to 79%. This is due to increased losses from channel noise, and is exacerbated by message expansion, since each fragment requires its own PHY header. The length of the fragments affects the transmission overhead, which we calculate next.

[5]The pulse jamming implementation transmitted five byte (160 μs) packets repeatedly; with a $T_{txdelay} = 128$ and overhead $\approx 160\,\mu s$, the actual transmit duty cycle was 50%.

Figure 11 shows the analytical message transmission overhead resulting from fragmentation. For PHY header length H, PHY footer length T, original message payload length P, fragment payload length F, and redundancy factor R, the overhead is calculated as:

$$\text{Overhead} = R \cdot \left(\frac{\left\lceil \frac{P+T}{F} \right\rceil \cdot (H+F)}{H+P+T} \right) \tag{3}$$

As expected, using very short fragments to transmit long payloads results in greater than 190% overhead (F=2). Our evaluation uses five-byte fragments (F=5) to transmit 33-byte payloads (P=33), for an overhead of 72%. Duplication of fragments, as in our redundant encoding implementation, doubles these values.

Though high when compared with performance-maximizing protocols, these overheads are acceptable when under attack. The ability to achieve from 87–98% PDR despite the presence of a jammer outweighs the additional transmission cost, especially when the alternative is zero PDR (for the interrupt jamming attack).

C. Discussion

With no defense against jamming, as in most WSNs, a simple interrupt jamming attack is completely effective (zero PDR) and costs little to the jammer. By adding defenses incrementally, we force a jammer to adapt its strategy. More effort is required at each step, culminating in a high-cost pulse jamming attack which only degrades performance by about 11%.

Note that if the attacker had started with single-channel pulse jamming, nodes could simply use a different channel. Then the attack would be both expensive and ineffective. We have shown how, with the accumulation of defenses and counter-attacks, the energy-conserving jammer nevertheless seems to have no better strategy than pulse jamming.

We believe the recovery of performance during otherwise devastating jamming attacks is well worth the overhead required. With DEEJAM, network nodes can continue to operate despite the attack, significantly improving robustness and reliability for successful deployment in medical, military, or industrial networks.

Some classes of attackers will of course be able to mount stronger jamming attacks than DEEJAM can defend against. They require more compromised or deployed nodes, higher power transmitters, or equipment with faster scanning and shorter transmission delays than the MICAz (shown in Table I). If anti-jamming countermeasures are not used, however, none of those costs are necessary and anyone able to reprogram a single mote can cause a denial-of-service.

VII. Conclusion

We presented DEEJAM, a protocol for defeating jamming with IEEE 802.15.4-based hardware. It uses frame masking, channel hopping, packet fragmentation, and redundant encoding to eliminate most of the impact of jamming by a mote-class attacker.

Using embedded WSN hardware we evaluated the effectiveness of DEEJAM against successively more complex jamming attacks: interrupt jamming, activity jamming, scan jamming, and pulse jamming. Ultimately, a pulse jammer corrupting an entire channel causes packet delivery ratios to drop by only 11%, allowing the network to continue operating.

In the future we will evaluate runtime selection of a subset of defenses to reduce the overhead based on what kind of attack is currently detected.

Acknowledgment

This work has been supported in part by ARO grant W911NF–06–1–0204, and NSF grants CNS–0614773 and CNS–0626616.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley Computer Publishing, 2001, pp. 326–331.

[3] R. L. Pickholtz, D. L. Schilling, and L. B. Milstein, "Theory of spread spectrum communications—a tutorial," *IEEE Transactions on Communications*, vol. 20, no. 5, pp. 855–884, 1982.

[4] Chipcon AS, subsidiary of Texas Instruments, CC1000 and CC2420 Radio Transceiver Products, URL: www.chipcon.com.

[5] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister, "System architecture directions for networked sensors," in *Proc. of ASPLOS*, 2000, pp. 93–104.

[6] Crossbow Inc., MICAz Wireless Sensor Network Hardware, URL: http://www.xbow.com.

[7] J. Polastre, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless research," in *IPSN*, 2005, pp. 364–369.

[8] *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)*, IEEE 802.15.4-2003 Standard for Information Technology, 2003.

[9] R. M. Kling, "Intel mote: an enhanced sensor network node," in *Int'l Workshop on Advanced Sensors, Structural Health Monitoring, and Smart Structures*, 2003.

[10] *Wireless MAC and PHY Specifications for Wireless Personal Area Networks (WPANs)*, IEEE 802.15.1-2005 Standard for Information Technology, 2005, (Bluetooth).

[11] W. Xu, T. Wood, W. Trappe, and Y. Zhang, "Channel surfing and spatial retreats: defenses against wireless denial of service," in *Proc. of WiSe*, 2004, pp. 80–89.

[12] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The feasibility of launching and detecting jamming attacks in wireless networks," in *Proc. of MobiHoc*. ACM Press, 2005, pp. 46–57.

[13] A. D. Wood and J. A. Stankovic, "Denial of service in sensor networks," *IEEE Computer*, vol. 35, no. 10, pp. 54–62, 2002.

[14] A. D. Wood, J. A. Stankovic, and S. H. Son, "JAM: A jammed-area mapping service for sensor networks," in *Proc. of RTSS*, 2003.

[15] Y. W. Law, P. H. Hartel, J. I. den Hartog, and P. J. M. Havinga, "Link-layer jamming attacks on S-MAC," in *Proc. of EWSN*, 2005, pp. 217–225.

[16] Z. Tang and J. J. Garcia-Luna-Aceves, "Hop-reservation multiple access (HRMA) for ad hoc networks," in *Proc. of INFOCOM*, 1999, pp. 194–201.

[17] G. Zhou, C. Huang, T. Yan, T. He, J. A. Stankovic, and T. F. Abdelzaher, "MMSN: Multi-frequency media access control for wireless sensor networks," in *Proc. of INFOCOM*, 2006.

[18] M. Čagalj, S. Čapkun, and J.-P. Hubaux, "Wormhole-based antijamming techniques in sensor networks," *IEEE TMC*, vol. 6, no. 1, pp. 100–114, 2007.

[19] G. Noubir and G. Lin, "Low-power DoS attacks in data wireless LANs and countermeasures," in *Proc. of MobiHoc*, 2003, pp. 29–30.

[20] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *Proc. of CCS*, 2003.

[21] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *IEEE Symposium on Research in Security and Privacy*, 2003, pp. 197–213.

[22] K. Sun, P. Ning, and C. Wang, "TinySeRSync: secure and resilient time synchronization in wireless sensor networks," in *Proc. of CCS*, 2006, pp. 264–277.

[23] H.-S. W. So, G. Nguyen, and J. Walrand, "Practical synchronization techniques for multi-channel MAC," in *Proc. of MOBICOM*, 2006, pp. 134–145.

[24] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *Proc. of SenSys*, Los Angeles, CA, 2003.

[25] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *Proc. of SenSys*, Los Angeles, CA, 2003.

[26] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic, "Impact of radio iregularity on wireless sensor networks," in *Proc. of MobiSys*, 2004, pp. 125–138.

[27] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," *ACM CCR*, vol. 27, no. 2, pp. 24–36, 1997.