Dodge-Jam: Anti-Jamming Technique for Low-power and Lossy Wireless Networks

Jeongyoon Heo, Jung-Jun Kim, and Saewoong Bahk*
Department of Electrical Engineering
Seoul National University and INMC
Seoul, Republic of Korea
E-mail: {jrheo, jjkim, sbahk}@netlab.snu.ac.kr

Jeongyeup Paek
School of Computer Science & Engineering
Chung-Ang University
Seoul, Republic of Korea
E-mail: jpaek@cau.ac.kr

***Abstract*—Jamming is one of the most famous and powerful attacks in wireless networks, and is advancing to be more stealthy and long-lasting with limited energy. Stealthy attackers transmit short jamming signals to become less detectable with less energy, and yet powerful enough to ruin the entire packet transmission procedures. For this study, we deal with three types of stealthy attacks: 'reactive jamming', 'jamming ACK', and 'fake ACK' attacks. These attacks are fatal to Low-power and Lossy wireless Network (LLN) applications because they not only interfere with communication, but also cause LLN devices to quickly drain their batteries.**

In this paper, we present *Dodge-Jam*, a light-weight anti-jamming technique suitable for LLN environments to address the stealthy jamming attacks with small overhead. It protects ACK exchange by switching the ACK channel calculated based on the content of a data packet. Moreover, by partitioning a packet into multiple small blocks and performing logical shifts of the blocks when retransmitting the packet, it helps the receiver recover the original packet from multiple erroneous packets. We implement *Dodge-Jam* on practical embedded devices, and evaluate its performance through experiments on a multihop LLN testbed. Our results show that *Dodge-Jam* successfully avoids many jamming attacks, recovers packets that have been jammed, and improves packet delivery performance of both singlehop and multihop networks significantly.

***Index Terms*—Low-power Lossy Network (LLN), IEEE 802.15.4, Jamming, Security, Wireless Sensor Network**

I. Introduction

Low-power and Lossy Networks (LLNs) have been used widely in many areas including smart grids [1], wireless sensor networks (WSN) [2][3][4], and Internet of Things (IoT). However, LLNs, often comprised of resource-constrained battery-operated devices, are vulnerable to jamming attacks that cause Denial-of-Service (DoS) [5]. When a jammer sends jamming signals in order to disrupt communication, the target node under attack repeatedly fails in transmission attempts and retransmits the packets, which will severely decrease network performance and quickly exhaust the battery of the devices. Jamming is critical to applications that are sensitive to delay or loss. For example, if a medical network which monitors patients' physiological data over an LLN [6] is under attack, serious problems may arise because they are directly linked to patients' life and death. Home security systems using WSN [7][8] also have a similar problem. An intruder may use jamming attacks to break into a house by disturbing transmissions of alarm messages to the home control center. Jamming attack can also affect smart grids and smart factories to make misjudgments and cause erroneous operation [9].

LLN devices may be helpless against well-funded powerful wideband jamming attackers [5]. However, if a battery-operated jammer transmits jamming signal continuously at high power, that will drain jammer's energy quickly resulting in short overall jamming lifetime. If the jammer has a sufficient amount of energy to be powerful and long-lasting, then it can be easily detected by legitimate defenders [10]. On the other hand, there are more advanced attacks that are less detectable and long-lasting while consuming minimal energy. In these kinds of stealthy attacks, attackers only send jamming signals as needed, just enough to disrupt communication. For example, in 'reactive jamming', an attacker sends jamming signals only when it detects on-going packet transmissions.

'Jamming ACK' attack is also one of such attacks. The jammer detects (or predicts) an ACK transmission, and upon detection, sends a short jamming signal to jam only the ACK frame in order to make the legitimate sender keep retransmitting the original packet. Furthermore, the work in [11] introduces 'fake ACK' attack which disturbs communication by sending a jamming signal during a packet transmission, and then sends a fake ACK to make the sender believe that the original packet transmission has succeeded. There have been several prior works that address these attacks [5][11][12][13][14][15], but most of these efforts either require expensive pre-shared secrets, resulting in limited effect, or incur excessive overhead to resource and power limited LLN devices.

To address these challenges, we design *Dodge-Jam*, a light-weight anti-jamming technique for LLNs without large overhead. *Dodge-Jam* confronts attackers that have the same capability level (in terms of processing, memory, transmission power, and battery) with legitimate LLN devices. It is composed of three main techniques: 'ACK channel hopping', 'multi-ACK channel hopping' and 'multi-block data shift'. To address the fake ACK attack, ACK channel hopping changes the channel on which to send ACK frames from the channel

Saewoong Bahk is the corresponding author.

This work was supported in part by the ICT R&D program of MSIP/IITP Korea (B0717-16-0026) and the National Research Foundation of Korea (NRF-2015R1A2A2A01008240), and in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF-2014R1A1A2056626).

on which data packets have been received. There can be a channel-scanning attacker that scans channels to find on which channel a legitimate packet is transmitted, and to jam that channel [5]. ACK channel hopping is free from such an attack because ACK transmission is too short to be found by the attacker using channel scanning. Multi-ACK channel hopping applies a random rendezvous technique to the ACK channel hopping in order to avoid the jamming ACK attack opportunistically.

Multi-block data shift is designed to recover jammed and corrupted packets. The sender partitions a packet into several small blocks and adds a CRC for each block. When retransmitting the packet due to transmission failure (upon no ACK reception due to reactive jamming attack, jamming ACK attack, or natural link loss), it performs a logical shift to the packet with the expectations that some blocks have been successfully received at the receiver and the attacker jams similar part(s) of the packet. After a few retransmissions, the receiver can recover the original jammed packet from multiple erroneous packets.

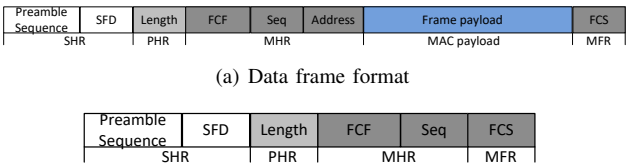
The contributions of this work are threefold:

- We study three stealthy jamming attacks in LLNs: 'reactive jamming', 'jamming ACK', and 'fake ACK' attacks, and experimentally measure how destructive they are to communication between LLN devices.
- We design 'Dodge-Jam', a lightweight anti-jamming technique for LLNs, that combines 'ACK channel hopping', 'multi-ACK channel hopping', and 'multi-block data shift' defense mechanisms to defeat the aforementioned attacks. We then show how each of the components in *Dodge-Jam* contributes to defeating these attacks.
- We implement *Dodge-Jam* TelosB [16] wireless sensor network platform and experimentally evaluate its performance through real testbed experiments on both single-hop and multi-hop topologies. Our evaluation shows that *Dodge-Jam* achieves significantly better packet delivery ratio compared to the network without *Dodge-Jam*. In some cases, it allows the network under attack to continue to operate and communicate, which would otherwise have been useless without *Dodge-Jam*.

The remainder of this paper is structured as follows. Section II discusses related work, and Section III describes three stealthy jamming attacks in LLNs that are energy-efficient and effective. Then, we propose *Dodge-Jam* to defeat these jamming attacks in Section IV. In Section V, we describe how we implement the attacks and *Dodge-Jam*. Section VI evaluates both the impact of the attacks and the performance of *Dodge-Jam* on how it tackles those attacks. Finally, Section VII concludes the paper.

II. Related Work

Two pieces of work that are the closest to ours are Jam-Buster [17] and DEEJAM [5]. These works target jamming attackers in the same capability class as network nodes, and their goal is to increase the cost of jammers and force them to transmit more jamming signals to achieve effective jamming, which will reduce the lifetime and increase the detectability of the jammers.



(a) Data frame format

(b) ACK frame format

Fig. 1. Frame formats of IEEE 802.15.4.

In DEEJAM [5], the authors first describe the effectiveness and stealthiness of interrupt jamming attack (and its variants), and how vulnerable an IEEE 802.15.4 based wireless network can be under such energy-efficient jamming. Then they propose four defensive mechanisms: frame masking, channel hopping, packet fragmentation, and redundant encoding. Both the attacks and defense mechanisms are evaluated on MicaZ motes to show that their proposal is effective in mitigating such jamming attacks. However, their solutions are based on pre-shared keys, and rely on tight synchronization between legitimate senders and receivers, resulting in high overhead even in the absence of jamming attacks.

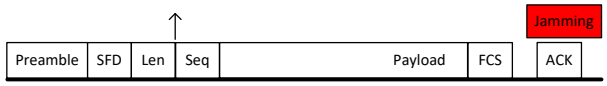
In Jam-Buster [17], the authors point out that low resilience and easy differentiability of protocol control messages, as well as high predictability of node wakeup schedules, are what makes LLNs vulnerable to jamming, and propose to eliminate such vulnerability by using multi-block payloads, equal size packets, and randomized wakeup times of network nodes. The idea of using multi-block payloads is similar to one of our schemes. However, it targets only duty-cycled MAC, increases the packet transmission overhead by making all packet sizes larger even in the absence of jamming, and does not handle stealthier jamming ACK or fake ACK attacks. Furthermore, unlike our work, both Jam-Buster [17] and DEEJAM [5] evaluate their schemes only on a 1-hop network setup without considering multihop scenarios.

To handle jamming ACK attack, Zhang *et al*. propose JACK which applies a random backoff for sending an ACK [12]. JACK starts with sending an ACK at a randomly chosen time between $[0, (\mathrm{R}-1)\cdot T_{ACK}]$, where T_{ACK} is the time for sending the ACK. They found 7.5 as an optimal value for R which means that JACK takes 7.5 times longer compared to the original ACK sending. To disable JACK, an attacker may simply send a longer jamming signal which only makes the attacker spend a little more energy. Therefore, it has a limited effect with large overhead. To handle the fake ACK attack, the works in [11][13][14][15] make efforts of authenticating ACK packets, but all of which need expensive pre-shared secrets.

III. Attack models

In this paper, we study and tackle three kinds of stealthy and energy-efficient attacks: 'reactive jamming', 'jamming ACK', and 'fake ACK' attacks. This section describes each of these attacks.

(a) Reactive jamming attack.



(b) Jamming ACK attack.

(c) Fake ACK attack.

Fig. 2. An operation example of reactive jammer, jamming ACK attacker and Fake ACK attacker.

A. Reactive jamming

A reactive jammer jams a packet 'reactively' only when it detects an on-going packet transmission to minimize the energy usage and the risk of getting detected while jamming. If a jammer sends jamming signals continuously, not only it will drain its energy quickly resulting in short jamming lifetime, but it will also be easily detected. Therefore, the reactive jammer ensures that it sends a jamming signal just long enough to drop a packet, but at the same time short enough to be not easily detected while minimizing energy usage.

The attacker can detect the beginning of an on-going packet transmission by sensing a start-of-frame-delimiter (SFD) signal. Fig. 1(a) shows the data frame format of IEEE 802.15.4 [18]. The synchronization header (SHR) consists of a preamble sequence and the SFD, and always precedes a transmission. After a node receives the SFD field, the SFD pin goes active and interrupts the microcontroller to start reading data from the packet buffer. Fig. 2(a) shows the procedures of reactive jamming. When the attacker detects the SFD, it switches its state from receiving mode to transmission mode, which requires the Rx-Tx switching delay of roughly 192μs on our platform [19]. After the Rx-Tx switching, it starts jamming to interfere with legitimate transmissions.

B. Jamming ACK attack

Jamming an ACK frame is another kind of attack that is energy efficient and not easily detectable. A jamming ACK attacker disturbs communication by sending a jamming signal that collides with an ACK frame. Normally, when a receiver successfully receives a packet, it sends an ACK frame back to the sender. If the sender does not receive the ACK frame for a specific ACK waiting time, it supposes that the transmission has failed. By hampering the ACK, the jamming ACK attacker ruins the packet transmission and makes the sender keep retransmitting the entire packet. Fig. 2(b) shows an example of jamming ACK attack. After a jamming ACK attacker detects a packet transmission by sensing the SFD, it gets its length value which is located right after the SFD. From the length value, it calculates the time when the receiver is going to send the ACK, and sends the jamming signal at the exact time to disturb it.

C. Fake ACK attack

Fake ACK attack is stealthier than the aforementioned attacks. If a packet transmission keeps failing, the sender assumes that the link quality has degraded and thus updates its neighbor table. Then, the routing layer of the sender may change its routing path to send packets on. This recovery process at the routing layer may successfully avoid the jamming area and send packets on a new routing path.

A fake ACK attacker jams a data packet, and then sends a legitimate-looking fake ACK to make the sender believe that the receiver has successfully received the packet. When a node receives an ACK, it cannot know who sent the ACK because, as shown in the Fig. 1(b), the ACK frame of IEEE 802.15.4 does not include the address of the ACK sender [18]. Therefore, the attacker can fool the sender by making a fake ACK just with the proper sequence number. Fig. 2(c) shows an operation example of a fake ACK attacker. When the attacker detects the packet transmission, it gets the length value and the sequence number from the beginning of the packet to construct and send ACK properly. Then, it jams the data packet and creates an ACK with the corresponding sequence number. Using the length value, the attacker sends the ACK frame at an appropriate time in which the legitimate receiver sends the ACK when it receives the packet successfully.

IV. Proposed scheme: *Dodge-Jam*

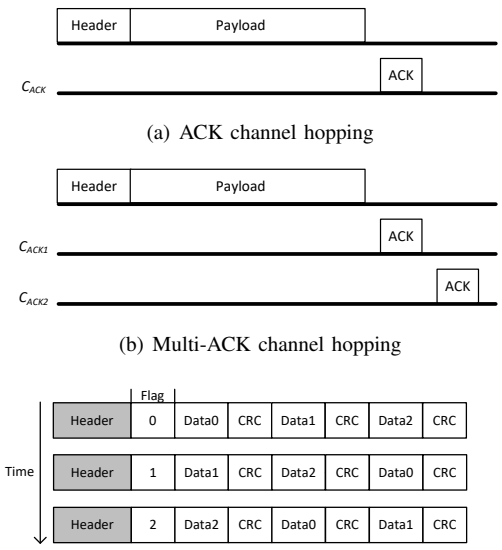
To evade jamming attacks and recover jammed packets from those attacks described in Section III, we present three defense mechanisms. ACK channel hopping addresses the fake ACK attack by making sure that a sender is not fooled by fake ACKs, and multi-ACK channel hopping uses multiple channels when sending ACKs to cope with the jamming ACK attack. Multi-block data shift helps the receiver recover a jammed packet by reconstructing the original packet from multiple block-shifted retransmissions when some blocks are transmitted in error. Lastly, we propose *Dodge-Jam* which adaptively combines the three aforementioned schemes to provide a comprehensive defense against the jamming attacks.

A. ACK channel hopping

To evade the fake ACK attack, we propose ACK channel hopping in which a sender and a receiver change the channel to exchange ACK dynamically on a per-packet basis. A channel on which to send an ACK, C_{ACK}, is calculated as,

$$C_{ACK} = rand(hash(data)) \bmod 16 + 11 \quad (1)$$

where `rand()` is a pseudo random function which is used to generate a channel randomly. We use a `hash()` function to generate a 16 bits input for the random function based on the input data. In practice, this can be also a `crc16()` function for simplicity. The input *data* of Eq. (1) includes both the MAC header and the MAC payload. Since the MAC header



(a) ACK channel hopping

(b) Multi-ACK channel hopping

(c) Multi-block data shift

Fig. 3. An operation example of ACK channel hopping, multi-ACK channel hopping, multi-block data shift.

includes a sequence number for each message, including the header ensures that the two consecutive packets with the same payload will have different ACK channels.

From the ACK channel hopping, only a node who receives the whole packet can calculate and send ACK properly. Fig. 3(a) illustrates an example of ACK channel hopping. After the sender sends a packet, it calculates the ACK exchange channel from the data it sent and switches the listening channel to receive the ACK. When the receiver receives the packet, it calculates the channel to send the ACK from the received data. Since the attacker sends jamming signals during the packet transmission, it cannot receive the whole packet as is and compute the proper ACK channel. Therefore, if we use ACK channel hopping, we can defeat the fake ACK attack.

B. Multi-ACK channel hopping

ACK channel hopping is also effective for preventing jamming ACK attack if the ACK channel selection procedures (i.e. Eq. (1)) are not known to the attacker. In this scenario, the best alternative that the attacker can use is a brute-force method in which it randomly picks a channel to attack, whose probability of success reaches only $1/(\text{total number of channels})$ i.e. $1/16$. However, if the ACK channel hopping procedures are known to the jamming ACK attacker who overhears the whole packet, then the attacker can calculate the ACK channel in the same way as a legitimate receiver and successfully jam the ACK. We call this attack 'channel hopping jamming ACK attack'. The ACK channel hopping also works against this channel hopping jamming ACK attacker by making it spend more energy in jamming the ACK frames, although it has a limitation that it cannot completely prevent the attack itself.

To deal with this issue, we propose multi-ACK channel hopping to avoid the channel hopping jamming ACK attack

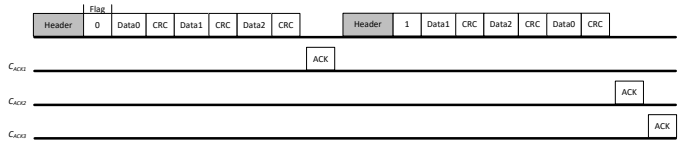


Fig. 4. An operation example of Dodge-Jam.

stochastically. When the receiver receives a packet, it obtains n channels from the received data by repeatedly using Eq. (1). That is, it repeats the calculation of Eq. (1) until it obtains n different channels. Then, it sends n ACKs in a random sequence of the n channels. The sender waits for the ACK at a channel that is randomly selected from the calculated n channels. As the attacker cannot know neither the ACK channel of the sender nor the channel sequence of the receiver, it can only jam a randomly selected channel among the n channels. From this procedure, we lower the probability of successful channel hopping jamming ACK attack to $1/n$.

Fig. 3(b) shows an example of multi-ACK channel hopping using 2 channels. From a received packet, the receiver computes two ACK channels, C_{ACK1} and C_{ACK2}, and decides to send ACKs over C_{ACK1} and C_{ACK2} in order. The sender decides the channel on which to wait for the ACK as C_{ACK2}. Then the sender receives the ACK on C_{ACK2} when the receiver sends the second ACK, given that there is no attack on C_{ACK2}. In this example, we avoid jamming ACK attack with probability 1/2. From retransmissions, the probability of successfully avoiding jamming increases rapidly. If the receiver continuously uses two channels to send ACKs, the probability of avoiding jamming is $1 - \left(\frac{1}{2}\right)^k$, where k is the number of total transmissions of the packet. It is 87.5% after the 2 retransmissions and about 94% after the 3 retransmissions.

C. Multi-block data shift

To reconstruct and recover a data packet under reactive jamming or fake ACK attack, we propose multi-block data shift. In this scheme, a sender partitions a packet into multiple small blocks, and adds a CRC for each block at the end of that block. When the receiver receives the packet, it checks the CRC of each block as well as the CRC of the entire packet. If all the blocks pass the CRC check, it sends an ACK back to the sender. Otherwise, it only saves those blocks which pass the CRC check and waits for next retransmission. On the sender side, the sender performs logical shift of the blocks every time it needs to retransmit the packet. Since stealthy attackers jam packets only for a short duration for energy and detectability reasons, there can be some blocks which are not jammed. Also, since a reactive attacker or a fake ACK attacker who jams the data packet sends jamming signals after it senses the SFD, it is highly probable that jamming signals are inserted at a similar position of the data packet for each transmission. Hence, if the sender transmits shifted packets for retransmissions, the block at which the packet is jammed at the previous transmission may not be jammed when it is retransmitted. Based on this

intuition, the receiver recovers the jammed packet from several erroneous retransmissions of shifted packets.

Fig. 3(c) shows an example of multi-block data shift which separates a packet into three blocks. Without loss of generality, we give a number starting from 0 to each block according to the position of the block in the original packet. The sender inserts a 1-byte CRC at the end of each block. In front of the payload, it adds 1 byte flag which is the number of the first block in the packet to notify the order of the blocks. For example, if a flag is 1, the order of blocks is 1, 2, and 0. If the sender does not receive an ACK after a transmission, it performs the logical shift and retransmits the packet until it receives an ACK for the packet. Adding the CRC and the flag is the overhead of applying the multi-block data shift. If we partition a packet into three blocks, the overhead takes up about 3% of the packet when the size of the packet is 127 bytes.

D. Dodge-Jam

Finally, *Dodge-Jam* combines the three mechanisms described above to handle all the aforementioned attacks. Fig. 4 shows an operation example of *Dodge-Jam* which partitions a packet into three blocks. First, a sender sends a multi-blocked packet, and the receiver computes and changes the channel to send ACK on. If there is a fake ACK attacker, we cannot recover the packet by using multi-block data shift alone because the attacker sends a fake ACK which prevents the sender from knowing the transmission failure and retransmitting. Combining multi-block data shift and ACK channel hopping lets the sender successfully recover the packet even when there is a fake ACK attacker. For the first packet transmission, we use only one ACK channel to decrease overhead when the network is not under attack. If the sender does not receive an ACK, it will shift the packet blocks and retransmits the packet. When retransmitting the packet, we apply the multi-ACK channel hopping and increase the number of ACK channel by 1 for each retransmission. While doing this, we limit the maximum number of ACK channels to M to put a bound on the overhead ($M = 3$ in our experiments).

However, there is a challenging issue. If a sent packet is lost, the number of sent packets at the sender and the number of received packets at the receiver may become different. This inconsistency makes the sender and receiver to use different size of channel pools. To address the issue, we use the flag field which is used in multi-block data shift in Section IV-C. In multi-block data shift, the flag field is added to notify the order of the blocks. If we increase this value by 1 for each retransmission, the receiver knows both the exact number of sent packets and the first block which is $flag \bmod \#$ of blocks.

V. Implementation

Each LLN node, for both attackers and legitimate nodes, is a TelosB clone device [16] with an MSP430 microcontroller and a CC2420 radio operating at 2.4GHz ISM band with 16 channels. We implement the attacks and *Dodge-Jam* in ContikiOS 2.7, and the transmission power of each node is configured to -7dBm.

A. Implementing the Attackers

Each attacker described in Section III tries to respond quickly to on-going legitimate transmissions to perform its attack. In particular, the reactive jammer and the fake ACK attacker should start sending jamming signals fast enough to corrupt ongoing packet transmissions, and all three attackers need some time to prepare for jamming which includes Rx-Tx switching delay and the time to construct the jamming signal. For this reason, all the attackers terminate their packet reception process as soon as they gain the necessary information to attack the packet transmission.

To achieve this promptness, the attackers change their receive mode from buffered to unbuffered mode. If a node is in the buffered mode, it accesses the packet data after it buffers the data in the RXFIFO buffer. However, in the unbuffered mode, when a receiver senses a SFD signal, the CC2420 chip of the receiver forwards the data directly to the microcontroller without buffering. Therefore, the attackers use the unbuffered mode to get the necessary data in real-time and change their state to the transmission mode to jam the ongoing packet. By default, CC2420 transceiver uses clear channel assessment (CCA) to send a packet only when the channel is idle. To interfere with an ongoing packet, we turn off the default CCA of the attackers. Furthermore, we also turn off the default autoACK feature since it is unnecessary for the jammer.

B. Implementing Dodge-Jam

There are three main components in *Dodge-Jam*, which are all implemented at an *Dodge-Jam*-layer between the link layer and the CC2420 chip. First, to implement the multi-block data shift, we disable the default CRC check in CC2420. Then, we make the sender partition a packet into multiple blocks, and add a CRC of each block at the end of the block. We also add a 1-byte flag to indicate the partitioning sequence of the packet.

To implement the ACK channel hopping and the multi-ACK channel hopping, we first disable the default autoACK feature and handle the ACK creation and transmission in the software at the *Dodge-Jam*-layer. When a data packet is delivered to the *Dodge-Jam*-layer from the physical layer, the receiver checks the flag value to decide the number of channels to send ACKs on, and then selects the ACK channels by using the content of the frame. The sender also chooses the ACK receiving channels in the same way as the receiver. The number of channels that the sender and the receiver compute depends on the number of transmissions. The receiver sends ACKs in a random sequence of the computed channels and the sender randomly picks one of the computed channels to wait for the ACK. After the ACK exchange, they return to the original channel.

VI. Evaluation

In this section, we provide and analyze the measurement results for each of the three attacks, and evaluate how *Dodge-*

TABLE I
SHORTHAND NOTATIONS FOR ATTACKS AND DEFENSE SCHEMES

N0	Baseline (without any attack)
A1	Reactive jamming attack
A2	Fake ACK attack
A3	Jamming ACK attack
A3*	Channel hopping jamming ACK attack
D1	Multi-block data shift
D2	ACK channel hopping
D3	Multi-ACK channel hopping

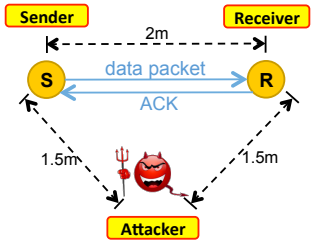
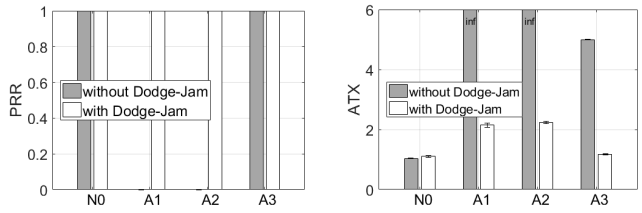


Fig. 5. Single-hop scenario with one sender, one receiver, and one jammer.

TABLE II
EFFECT OF ATTACKS WITHOUT ANY DEFENSE

Attack	Average ATX	Average PRR
N0	1.038	1.0
A1	infinite	0.0
A2	infinite	0.0
A3	5	1.0



(a) PRR (b) ATX

Fig. 6. Packet delivery performance comparison in a single-hop scenario under three different attacks, with and without *Dodge-Jam*.

Jam defeats or diminishes the effect of those attacks. We experiment on both single-hop and multihop testbed setups, and compare the results for three different jammer locations in the multihop case. We also investigate the effect of various components and parameter settings in *Dodge-Jam*.

Table I lists the shorthand notations of the three attack models that we study, and also the defense mechanisms/components of *Dodge-Jam*. For all experiments, we set the maximum number of retransmissions at the link layer to 4, which means a sender may send the same packet up to 5 times. All experiments are performed in an indoor office environment.

For performance evaluation metrics, we use the packet reception ratio (PRR) and the average number of required transmissions per packet (ATX) defined as,

$$PRR = \frac{\text{number of successfully received unique packets}}{\text{number of sent unique packets}}, \quad (2)$$

$$ATX = \frac{\text{total number of transmissions}}{\text{number of successfully received unique packets}}. \quad (3)$$

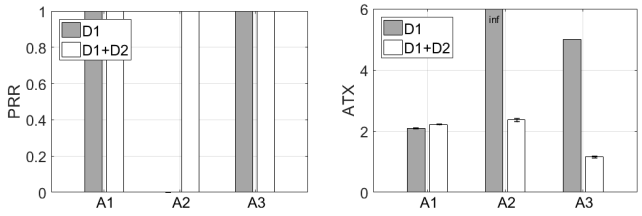
ATX is used to measure the overhead due to packet retransmissions, and it should ideally converge to the well-known ETX (expected number of transmissions) metric [20] (or vice versa) in the link layer unless the link is completely broken (in which case, PRR is zero and ETX/ATX becomes infinity). The minimum value of ATX is 1 when all transmissions are successful without any retransmission, and a smaller ATX value indicates better transmission performance.

A. Single-hop Scenario

First, we study the packet delivery performance for a single-hop unicast transmission scenario with a sender, a receiver and a jammer. The sender and the receiver are placed 2 m away from each other and at a 1 m height from the floor. The jammer is placed 1.5 m away from them at the same height as shown in Fig. 5. The attacker starts jamming as soon as possible (as explained in Fig. 2), and in the default setting, sends a 9-bytes jamming signal for both data jamming and ACK jamming. We use 3 blocks when performing multi-block data shift which results in a block size of 17 bytes for our 45 bytes payload and 6 bytes network-layer header. For each experiment, the sender sends 100 data packets at an inter-packet interval of 3 seconds. Tests are repeated five times and we average the results. Error bars represents 95% confidence intervals.

Table II presents the effect of each of the three attackers without any defense mechanism in a single-hop scenario. When there is no attacker (N0, the baseline case), both the PRR and ATX are almost 1, which means that there are very few packet losses and retransmissions. When there is a reactive jammer (A1) or a fake ACK attacker (A2), PRR and ATX become zero and infinite, respectively, which means that both the attackers cause all the packet transmissions to fail. In case of the jamming ACK attack (A3), the sender is unaware of the packet reception at the receiver. Thus, the receiver correctly receives most, if not all, of the packets because the jammer only jams the ACK while the sender retransmits the same packet 4 times due to no ACK reception. After sending the fifth transmission including 4 retransmissions, the sender gives up and sends the next packet. For this reason, although PRR of this case is 1, the sender exhausts the maximum retransmission count, which makes ATX 5. Fig. 6 shows the performance of *Dodge-Jam*. The white and gray bars represent the results with and without *Dodge-Jam*, respectively. If *Dodge-Jam* is in action, PRR results are almost 1 in all cases (Fig. 6(a)), which proves that all the attacks have been successfully defeated. When the link is under the reactive jamming (A1) or fake ACK attack (A2), *Dodge-Jam* uses retransmission-based multi-block data shift to recover the jammed packet, which makes ATX approximately 2.

To investigate deeper into the contribution of individual components of *Dodge-Jam*, we plot Fig. 7. In the case of reactive jamming (A1) attack, if we apply multi-block data



(a) PRR (b) ATX

Fig. 7. Effects of *Dodge-Jam* components under various attacks.

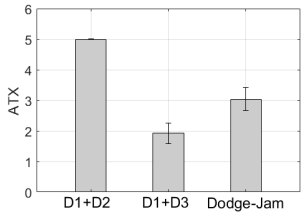
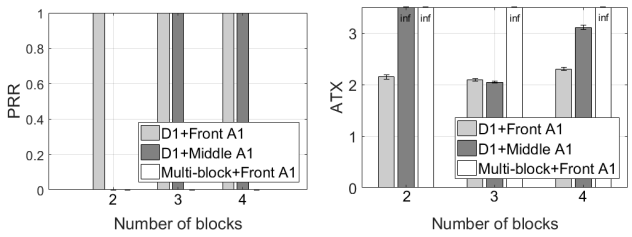
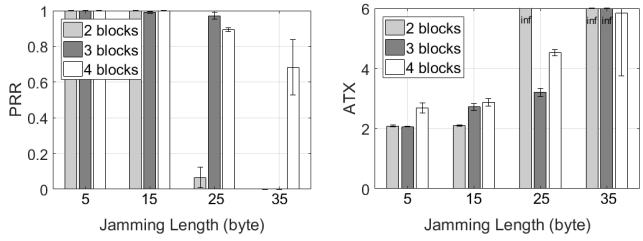


Fig. 8. ATX results under A3* with various combination of defense mechanisms. Since this attack only jams the ACK frames and does not jam the data packet, PRR is 1 for all cases from the receiver's point of view.



(a) PRR (b) ATX

Fig. 9. Performance of multi-block data shift and multi-block without data shift with the number of blocks and the jamming position.



(a) PRR (b) ATX

Fig. 10. Performance of multi-block data shift with the number of blocks and the jamming length

shift (D1), ATX is approximately 2 while PRR is 1, which means that the receiver recovers the jammed packet after a retransmission in most cases. If we apply ACK channel hopping (D2) in addition to multi-block data shift (D1), this combination, denoted as D1+D2, yields almost the same results with D1 only case, from which it is clear that D1 is powerful enough to defend against reactive jammer (A1). However, in the case of fake ACK attack (A2), the receiver cannot recover a jammed packet with multi-block data shift (D1) because the sender does not retransmit the packet once it receives the fake ACK from the attacker. When we apply the combination of D1+D2, it can avoid fake ACK attack by changing the channel to send the ACK. ATX in this case is about 2.36, which is slightly larger than the ATX results under reactive jamming (A1). This is because, since the fake ACK attacker must listen to the sequence number, its jamming position is slightly delayed compared to that of the reactive jammer. This sometimes results in the fake ACK attacker corrupting more than one block and increases ATX. In the case of jamming ACK attack (A3), it is challenging to avoid the attack only with multi-block data shift (D1) because the attacker jams ACK frames to make the sender keep retransmitting the packets. In order to avoid A3, the combination of D1+D2 is required to hide the ACK channel from the attacker. This has been successful as can be seen from the fact that ATX is approximately 1.

We also consider another kind of jamming ACK attack which changes the ACK channel in the same way as the multi-ACK channel hopping. We call this attack 'channel hopping jamming ACK attack' (A3*). Fig. 8 shows the experiment results under A3*. ATX of D1+D2 is 5 which implies that ACK channel hopping (D2) is insufficient to avoid A3*. The expected ATX of multi-ACK channel hopping (D3), $E[ATX_{D3}]$, is calculated as,

$$E[ATX_{D3}] = \sum_{k=1}^{5} k \cdot (n-1) \cdot \left(\frac{1}{n}\right)^{k} + 5 \cdot \left(\frac{1}{n}\right)^{5} \quad (4)$$

where n is the number of ACK channels used in D3. If D3 uses two channels to send ACK ($n = 2$), $E[ATX_{D3}]$ is approximately 1.94.

In the case of *Dodge-Jam*, we obtain the expected ATX, $E[ATX_{\text{Dodge-Jam}}]$, as

$$E[ATX_{\text{Dodge-Jam}}] = 1 + \sum_{k=3}^{5} k \cdot \left(\frac{1}{3}\right)^{k-2} + 5 \cdot \frac{1}{2} \cdot \left(\frac{1}{3}\right)^{3}, \quad (5)$$

which becomes approximately 2.72. The experimental results in Fig. 8 match the results of the theoretical calculation, which validates both our analysis and the correctness of our experiments.

To investigate the impact of data shifting and the number of blocks B of multi-block data shift (D1) on *Dodge-Jam*'s performance, we plot Fig. 9. The light gray bars represent the results of D1 when the link is under reactive jamming (A1) that transmits a jamming signal as soon as possible. We call this attack 'front A1' in this experiment and use it as A1 in all other experiments. The dark gray bars show the results of D1 when there is an A1 attacker who transmits a jamming signal with some delay which causes the attacker to jam the packets in the midst of transmission. We call this attack 'middle A1'. The white bars show the results of the multi-block without data shift. Only with multi-block, ATX is infinite and PRR is 0 for all the cases because the receiver cannot recover the jammed packet even after maximum retransmissions. When we apply D1 under 'front A1' attacker, ATX is about 2 regardless of the number of blocks because, in most of the cases, the attacker

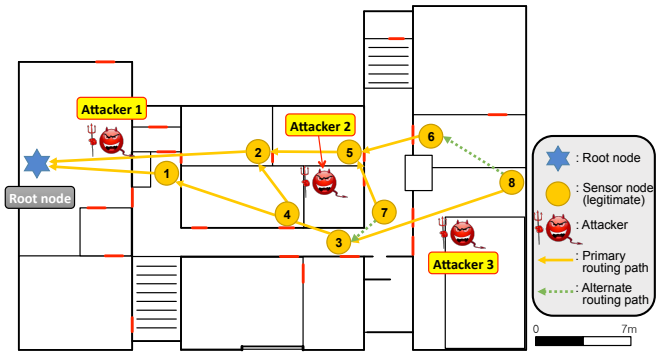


Fig. 11. Testbed topology map for the multihop scenario.

jams the first block of a packet and D1 recovers the packet after about one retransmission. In the case of 'middle A1', the attacker jams two blocks when the number of blocks is 2 or 4. It decrease the PRR of the D1 with 2 blocks to 0. If we separate a packet into more blocks, the probability that more blocks will be broken increases because the size of each block becomes smaller. As a result, separating a packet into too few blocks and too many blocks are both detrimental.

From the results in Fig. 9, a good empirical number of blocks is 3. However, the optimal number of blocks (or block sizes) also varies with the length of a jamming signal. Fig. 10 depicts the performance of D1 depending on the number of blocks and the jamming length. If the length of a jamming signal is under 15 bytes, all the PRR results show near 1 and ATX results are below 3. The reason that ATXs show slightly larger value than 2 is that, in some cases, 2 blocks are jammed and 2 retransmissions are required to recover them. As the number of blocks increases, the block size becomes smaller, and this increases the probability that 2 blocks are broken. If the jamming length exceeds 25 bytes, PRRs drops drastically when two blocks are used because the jamming signal is long enough to jam both blocks. When the length of the jamming signal is 35 bytes, the attacker even jams 3 blocks and decreases the PRR results of 2 and 3 block cases to 0. Thus, it is possible to recover from a long-length jamming by dividing a block into many smaller blocks, but when the jamming length is short, the probability that several blocks will be broken increases and performance decreases.

B. Multihop Scenario

Studying jamming attacks and defense techniques under multihop scenarios is important because the LLN routing protocol (e.g. RPL [21] and its variants [22],[23],[24]) may (or may not) act independently to avoid links that are problematic depending on the attack type. For some attacks, upon detection of insufficient packet delivery performance (e.g. ETX), the routing protocol may naturally select an alternate link for its routing path without any need for jamming defense. Of course, this assumes that an alternate link exists. On the other hand, for some other attacks, the routing protocol may not be able to detect link failure and use the same path for an extended amount of time unless some higher-layer action is performed. Our evaluation in this subsection studies these cases.

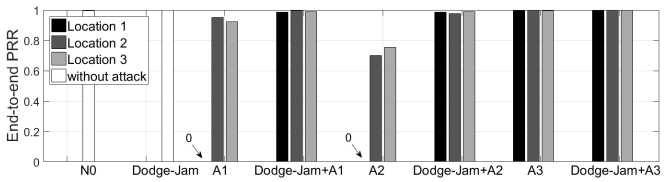
To study the impact of jamming attacks on the performance of *Dodge-Jam* in multihop network scenarios, we deployed an LLN testbed as depicted in Fig. 11. There are 8 legitimate LLN sender nodes (depicted in circles) and one sink node (marked with a star) in an indoor environment. Solid arrows depict a snapshot of the routing topology during one of our baseline experiments, which shows a 3-hop network. This routing topology is the outcome of RPL, the IETF standard IPv6 Routing Protocol for Low-power and lossy networks [21], with the default MRHOF objective function. At the beginning of each experiment, we let RPL construct the routing topology for 5 minutes. Once the routing paths have been established, each legitimate LLN node sends 100 end-to-end data packets to the root with an inter-packet interval of 30 seconds.

Using this testbed setup, we experiment with one attacker at a time at three different locations for the attacker. The first location (attacker 1) is close to the root, disrupting communications between the root and two first-hop nodes which forward all the traffic from their children nodes. For this reason, we expect this attacker location to have the most significant impact on network performance. The second location (attacker 2) is located in the middle of the routing tree and the network. It is expected to affect several links near it. The last location (attacker 3) is farther away from the root, closer to a leaf node which has the highest hop-distance to the root, and interferes with links that are far away from the root. This location is expected to have only limited impact on the performance. The dotted arrows in Fig. 11 represent alternate links that RPL nodes may select as their routing path towards the root if they detect link failure.

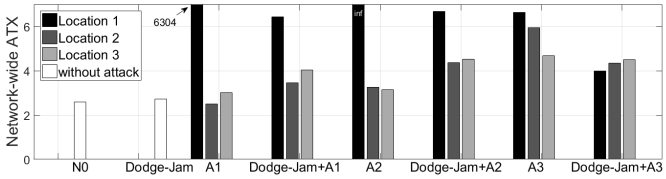
Fig. 12 depicts the end-to-end PRR and the network-wide ATX performance of the multihop RPL network with and without *Dodge-Jam*. When calculating the network-wide ATX, we take into account all transmissions including retransmissions and forwardings, and divide it by the total number of successfully received packets at the root. This network-wide ATX indirectly shows how much extra effort, on average, was needed by the network to deliver one end-to-end data packet to the root.

The leftmost white bars demonstrate the results without any attack, and shows that the PRR and the ATX of both baseline (N0) and *Dodge-Jam* are near 1 and 2.5, respectively without any attack. Attacker location 1 attacks both incoming links at the root. Since all the packets must be delivered through these two links, if the attacker act as reactive jamming (A1) or fake ACK (A2) attacker, it affects the network (without *Dodge-Jam*) significantly and decreases the PRR to almost zero, a complete black-out. The ATX is drastically increased under all kinds of attacks for this attacker location without *Dodge-Jam*.

In the case of attacker locations 2 and 3, there are alternate links on which to send packets. Thus, if the attackers act as reactive jammer (A1) or jamming ACK attacker (A3), nodes will experience some packet losses, and will change their routing path to send packets on as a natural parent



(a) PRR



(b) ATX

Fig. 12. Performance of *Dodge-Jam* in a multihop scenario with one attacker at three different attacker locations.

selection process of the routing protocol. With *Dodge-Jam*, as it recovers jammed packets under A1 or A2 while not changing the routing path, the ATX results increase slightly but *Dodge-Jam* increases the PRRs to near 1, which means that it mostly does not lose packets. However, under fake ACK attack (A2) without *Dodge-Jam*, nodes are unable to detect link failures. Thus, nodes will repeatedly attempt to transmit packets through jammed links. As can be seen in Fig. 12(a), the PRR decreases considerably in the case of A2, which indicates that the attacker adversely affects the nodes near it and all the nodes in their subtree. On the other hand, when *Dodge-Jam* is used, the PRR results are above 98%, even for those cases with 0% PRR without *Dodge-Jam*. When the network is under A3, the ATX results are decreased in all cases compared to those without *Dodge-Jam*.

Our experimental results show that *Dodge-Jam* successfully defeats all three attacks regardless of the attacker location, and allows the network under attack to continue to operate and communicate that would otherwise have been useless without *Dodge-Jam*.

VII. Conclusion

In this paper, we have presented an anti-jamming technique, termed *Dodge-Jam*, for low-power and lossy networks. It is composed of ACK channel hopping, multi-ACK channel hopping and multi-block data shift techniques to avoid stealthy jamming attackers and recover packets from jammed transmissions. We implement *Dodge-Jam* on real low-power embedded devices, and evaluate its performance in both single-hop setup and multihop wireless testbed. Our evaluation results show that *Dodge-Jam* successfully defeats three types of stealthy and energy-efficient jamming attackers, and recovers packet reception ratio from 0% to over 98% in both single-hop and multi-hop scenarios.

As our future work, we plan to study how *Dodge-Jam* can defeat other types of attackers, analyze the impact and performance of attackers and *Dodge-Jam* in terms of energy consumption, and develop techniques to find the location of attackers with the information obtained from the network.

References

[1] M. Erol-Kantarci and H. T. Mouftah, "Wireless sensor networks for cost-efficient residential energy management in the smart grid," *IEEE Transactions on Smart Grid*, vol. 2, no. 2, pp. 314–325, 2011.

[2] H.-S. Kim, H. Cho, M.-S. Lee, J. Paek, J. Ko, and S. Bahk, "MarketNet: An Asymmetric Transmission Power-based Wireless System for Managing e-Price Tags in Markets," in *ACM SenSys*, 2015.

[3] J. Paek, J. Hicks, S. Coe, and R. Govindan, "Image-Based Environmental Monitoring Sensor Application Using an Embedded Wireless Sensor Network," *Sensors*, vol. 14, no. 9, pp. 15 981–16 002, 2014.

[4] J. Paek, B. Greenstein, O. Gnawali, K.-Y. Jang, A. Joki, M. Vieira, J. Hicks, D. Estrin, R. Govindan, and E. Kohler, "The Tenet Architecture for Tiered Sensor Networks," *ACM Transactions on Sensor Networks*, vol. 6, no. 4, pp. 34:1–34:44, Jul. 2010.

[5] A. D. Wood, J. A. Stankovic, and G. Zhou, "DEEJAM: Defeating energy-efficient jamming in IEEE 802.15. 4-based wireless networks," in *IEEE International Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, June 2007, pp. 60–69.

[6] J. Ko, J. Lim, Y. Chen, R. Musaloiu-E., A. Terzis, G. Masson, T. Gao, W. Destler, L. Selavo, and R. Dutton, "MEDiSN: Medical Emergency Detection in Sensor Networks," *ACM Transactions on Embedded Computing Systems (TECS), Special Issue on Wireless Health Systems*, 2010.

[7] H. Huang, S. Xiao, X. Meng, and Y. Xiong, "A remote home security system based on wireless sensor network and gsm technology," in *International Conference on Networks Security Wireless Communications and Trusted Computing (NSWCTC)*, 2010.

[8] S.-S. Chiang, C.-H. Huang, and K.-C. Chang, "A minimum hop routing protocol for home security systems using wireless sensor networks," *IEEE Transactions on Consumer Electronics*, vol. 53, no. 4, 2007.

[9] Z. Lu, W. Wang, and C. Wang, "Hiding traffic with camouflage: Minimizing message delay in the smart grid under jamming," in *INFOCOM*, 2012.

[10] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The feasibility of launching and detecting jamming attacks in wireless networks," in *MobiHoc*, 2005.

[11] Y. Xiao, S. Sethi, H.-H. Chen, and B. Sun, "Security services and enhancements in the IEEE 802.15. 4 wireless sensor networks," in *IEEE Global Telecommunications Conference (GlobeComm)*, 2005.

[12] Z. Zhang, J. Wu, J. Deng, and M. Qiu, "Jamming ACK attack to wireless networks and a mitigation approach," in *IEEE Global Telecommunications Conference (GlobeComm)*, 2008.

[13] A. Dvir, L. Buttyan, and T. V. Thong, "SDTP+: Securing a distributed transport protocol for WSNs using Merkle trees and Hash chains," in *IEEE International Conference on Communications (ICC)*, 2013.

[14] M.-H. Park, "Challenge-response based ACK message authentication," *Electronics letters*, vol. 48, no. 16, pp. 1021–1023, 2012.

[15] D. Rossi, M. Omana, D. Giaffreda, and C. Metra, "Secure communication protocol for wireless sensor networks," in *2010 East-West Design Test Symposium (EWDTS)*, Sept 2010, pp. 17–20.

[16] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling Ultra-Low Power Wireless Research," in *Proceedings of IPSN/SPOTS*, April 2005.

[17] F. Ashraf, Y. C. Hu, and R. H. Kravets, "Bankrupting the jammer in WSN," in *IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (MASS)*, Oct 2012, pp. 317–325.

[18] Texas Instruments, "2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver," 2006.

[19] D. Moss and P. Levis, "BoX-MACs: Exploiting physical and link layer boundaries in low-power networking," *Computer Systems Laboratory Stanford University*, pp. 116–119, 2008.

[20] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A High-Throughput Path Metric for Multi-Hop Wireless Routing," in *MobiCom*, Sep. 2003.

[21] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," *RFC 6550*, Mar. 2012.

[22] H.-S. Kim, H. Kim, J. Paek, and S. Bahk, "Load balancing under heavy traffic in RPL routing protocol for low power and lossy networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 4, pp. 964–979, 2017.

[23] H.-S. Kim, J. Paek, D. E. Culler, and S. Bahk, "Do Not Lose Bandwidth: Adaptive Transmission Power and Multihop Topology Control," in *IEEE DCOSS'17*, June 2017.

[24] H.-S. Kim, H. Im, M.-S. Lee, J. Paek, and S. Bahk, "A measurement study of TCP over RPL in low-power and lossy networks," *Journal of Communications and Networks*, vol. 17, no. 6, pp. 647–655, 2015.