Compromise-Resilient Anti-Jamming for Wireless Sensor Networks

Xuan Jiang, Wenhui Hu, Sencun Zhu, and Guohong Cao

Department of Computer Science and Engineering
Pennsylvania State University
University Park, PA 16802
Email: {xjiang, wxh180, szhu, gcao}@cse.psu.edu

Abstract. Jamming is a kind of Denial-of-Service (DoS) attack in which an adversary purposefully emits radio frequency signals to corrupt wireless transmissions. Thus, the communications among normal sensor nodes become difficult or even impossible. Although some research has been conducted on countering jamming attacks, few works considered jamming by insiders. Here, an attacker first compromises some legitimate sensor nodes to acquire the common cryptographic information of the sensor network and then jams the network through those compromised sensors. In this paper, as our initial effort, we propose a compromise-resilient anti-jamming scheme called *split-pairing* scheme to deal with single insider jamming problem in a one-hop network setting. In our solution, the physical communication channel of a sensor network is determined by the group key shared by all the sensor nodes. When insider jamming happens, the network will generate a new group key to be shared only by all non-compromised nodes. After that, the insider jammer is revoked and will be unable to predict the future communication channels used by non-compromised nodes. We implement and evaluate our solution using the Mica2 Mote platform and show it has low recovery latency and communication overhead, and it is a practical solution for resource constrained sensor networks under the single insider jamming attack.

1 Introduction

Wireless communication is vulnerable to jamming-based Denial-of-Service (DoS) attacks in which an attacker purposefully launches signals to corrupt wireless transmissions. Wireless Sensor Networks (WSNs) are especially susceptible to jamming attacks due to the limited resources in computation, communication, storage and energy.

Jamming cannot be adequately addressed by regular security mechanisms such as confidentiality, authentication, and integrity, because jamming targets at the basic transmission and reception capabilities of the physical devices. None of the cryptographic constructions such as encryption/decryption could be directly adopted to solve the problem. Thus, we have to seek new solutions to deal with this severe attack.

Many existing countermeasures against jamming focus on spread spectrum [1, 2] in which the sender and receiver hop among channels or use different spreading sequence to evade the jamming attack. However, to successfully communicate under jamming

attack, both sender and receiver need to know the same hopping or spreading sequence beforehand and keep it secret. Although uncoordinated frequency hopping (UFHSS) and direct spread spectrum (UDSSS) [3–5] proposed to enable key establishment between one pair of nodes without a pre-shared secret under a jammer, these approaches are typically not applicable to WSNs due to the high storage and power cost.

For WSNs, Xu et al. proposed to use channel surfing [6] to deal with a narrow-band and intermittent jammer. Their basic idea is to let sensor nodes switch channels in a way that the jammer cannot predict them. For example, all nodes switch to a different channel $C(n+1) = F_K(C(n))$ to evade jamming after jamming is detected, where K is a group key shared by all nodes, F is a pseudorandom function and $C(n)$ is the original channel used before jamming. However, this technique is limited to outsider attacks and it does not work under node compromises since an inside attacker can acquire both the group key K and the function F.

In this paper, as our initial effort, we consider the insider jamming problem in a one-hop network. In our proposed solution, the physical communication channel is determined by the group key shared by all the sensor nodes. When insider jamming happens, the network will generate a new group key to be shared only by all non-compromised nodes. After that, the inside jammer is revoked and will be unable to predict the future communication channels used by non-compromised nodes. To realize the above idea, we address the following research challenges: *First, how can the non-compromised nodes agree on a new group key in a fully distributed way? Second, how do they distribute the new group key under the presence of the inside jammer.* Specifically, we propose a compromise-resilient anti-jamming scheme called *split-pairing* scheme. Our idea is based on the fact that for any given time the jammer can either jam one channel or none of them when it is switching channel. Thus, by actively splitting non-compromised nodes into two or multiple channels, nodes communicating in jamming-free channels can first reestablish a new common group key. A pairing process is then used to ensure all the nodes can receive the new group key.

We implemented and evaluated our scheme on a Mica2 Mote platform. We show our solution has low recovery latency and communication overhead, and it is a practical solution for resource constrained sensor networks.

The rest of the paper is organized as follows. The related works are presented in Section 2. The models and design goal are described in Section 3. The details of our recovery scheme is addressed in Section 4. We introduce our testbed and metrics in Section 5 and evaluate the performance of our scheme in Section 6. Finally, we discuss some related issues of our solution in Section 7 and conclude the paper in Section 8.

2 Related Works

Jamming models have been widely studied, classified and evaluated. For example, jammers can be classified in terms of capabilities (broadband or narrowband) or behaviors (constant, deceptive, random, reactive) [8]. Jammers discussed and used in prior works [6–11] can be also categorized based on their working layers in the network stack. Physical layer jammers directly emit energy on communication channels to interfere the reception of legitimate transmissions. MAC layer jammers can insert dummy packets or

preambles to deceive the receivers. Cross-layer jammers can attack some specific higher layer network protocols such as TCP or UDP to generate infinite retransmissions.

Most physical layer countermeasures rely on the spread spectrum technique. These solutions require both the sender and receiver share the same key and pseudorandom function to generate a hopping or spreading sequence. [3–5] studied the problem of key establishment without pre-shared secret under jamming. In [3, 4], a node pair establishes a new key by randomly hopping on a large number of channels until meet. Following that, UDSSS [5] was proposed for broadcast communication. Basically, the sender sends a message repeatedly and the receivers synchronize the transmission by a sliding-window approach and despread the received message by searching through a set of codes. However, most of the physical layer approaches require sophisticated processing unit and storage device which are not applicable to sensors.

Researchers studied jamming attacks on a broadcast control channel in [12, 13]. [12] considered an inside attacker who could compromise nodes to obtain the cryptographic information such as hopping sequences. A cluster head generates hopping sequences for each member in which some positions of the sequences share the same frequency bands for the control channel. The compromised node is identified by computing metrics such as the expected hamming distance. The cluster head then updates the hopping sequences and redistributes them. Dealing with the similar problem, [13] proposed a framework for the control channel access scheme, using the random assignment of cryptographic keys to hide the location of control channels. Both the schemes, however, run in a centralized manner with the help of either cluster heads or trusted authorities.

For WSNs, [6] discussed evasion strategies called channel surfing under a narrowband and intermittent jammer. The basic idea is that the jammed nodes change channels which cannot be predicted by the jammer. However, if the attacker could jam two channels at any time, the channel surfing scheme will not work. To solve the problem, [14] considered a way of communication even under the interference based on the timing covert channel. This scheme is effective against a broadband and constant/persistent attacker. In their solution, the detection of failed packets is proved by the experiment. A timing-based overlay and a coding/decoding scheme are established to convey information. Though interesting, besides being low-rate, it is unclear how this technique could be extended to a network with multiple users.

3 System Model and Design Goal

3.1 Network Model and Security Assumptions

We assume each node in the network has multiple channels and can switch to different channels. For example, the Mica2 mote, which is equipped with Chipcon CC1000 radio, has 32 effective channels for radio transmission from 902MHz to 928MHz with a separation of 800KHz in different channels [15]. As our first step towards addressing the insider jamming problem, in this paper, we focus on a one-hop network in which each node can directly communicate with all other nodes within one-hop range. This model is widely used and studied in recent works [3–5, 12–14] and to our knowledge, no work has studied jamming in a multihop network yet.

For security purpose, we assume every pair of nodes share a pairwise key. The issue of establishing pairwise keys for sensor nodes was the most well studied one in sensor

network security research. Many pairwise key establishment schemes [16–18] allow two nodes to establish a pairwise key on the fly as long as they know each other's id. In our work, we choose the Blundo scheme [20] for our solution since the Blundo scheme provides clear security guarantee and eases our presentation. In the Blundo scheme, a bivariate symmetric polynomial $f(x, y)$ with degree of t is chosen in advance and $f(i, y)$ is preloaded on sensor i. The pairwise key with node j on i can be generated by evaluating the function $f(i, j)$. The scheme provides unconditional secrecy if no more than t nodes collude. For the storage cost, a node needs to store a univariate polynomial represented by $t + 1$ coefficients. The size of a coefficient is the same as that of a symmetric key. For example, if a sensor network wants to tolerate the compromises of tens of nodes, it needs to store tens of coefficients. The size of a typical key on sensor is typically 8 or 16 bytes [19]; hence, each node needs to store hundreds of bytes of keying material. This storage overhead is affordable for low-end sensor motes with 4KB RAM.

To ease our presentation, we assume that legitimate nodes have detected and identified the jammer. Jammer localization and identification for WSNs are still open issues, although recently some efforts have been made towards addressing them, for example, RF fingerprinting for sensor nodes [26], jammer localization [27] and software-based attestation [28–31]. Nevertheless, the focus of this paper is recovery from insider jamming.

3.2 Attacker Model

As our initial effort, we assume that the attacker may compromise a single node to obtain such confidential information as group key. The group key is used to derive the channel used by all the sensor nodes. We discuss how to extend our scheme for the case of multi-jammers in Section 7. We also assume that the attacker launches jamming through the compromised sensor. That is, the jammer has the same physical capabilities in terms of power and frequency band as the normal sensor do. Two reasons for this assumption are that it is obvious that if the jammer is a high-powered, broadband aggressive device, there is no hope to construct a jamming-resilient sensor network with the current low-end sensors. A powerful jammer, on the other hand, can be easily noticed by defenders since it violates normal communication pattern while jamming by the insider sensor node is more stealthy. Nevertheless, we assume the compromised sensor launches signals as strong as possible to maximize the attack's damage. In the rest of the paper, informally, when we use "the attacker, it actually refers to the compromised sensor.

In our attack model, the attacker has following parameters:

- *Jamming Probability.* The attacker can jam up to n channels with probability $p_i (1 \leq i \leq n)$ for channel i.
- *Channel Switch Latency.* (t_l) The attacker needs time $t_l (t_l > 0)$ to change from one channel to another. From our experiment in Section 6, the typical latency is 34ms for Mica2 mote. For MicaZ mote [24], t_l is 132us. For 802.11 WiFi chipset, similar results can be found in recent research works. For example, in [21], the measurement result of t_l for Atheros chipset was reported as 7.6ms.
- *Sensing and Jamming Duration.* We consider two types of jammers: active and reactive. For active jammers, attackers launch jamming signal immediately without

sensing. We denote the jamming duration as t_j. For reactive attackers, attackers sense the traffic before jamming. Active attackers do not sense, so they may jam some channels that have no traffic. As such, active attacks have shorter response time but are not energy efficient; on the contrary, reactive attacks have longer response times but are more energy efficient.

3.3 Design Goal

Our goal is to design security mechanisms to minimize the damages caused by the inside jammer. More specifically, we consider a scenario where a normal node could be compromised and deceived as a malicious inside jammer. The attacker could use any cryptographic information known by the normal node to facilitate the jamming attack. For example, the jammer could always predict the next channel used for communication and launch jamming signals to block the eligible network traffic. The goal of our proposed security mechanism is to construct and propagate a new group key to all non-compromised nodes under the presence of the inside jammer so that the new key can be used to establish a keyed secret channel which cannot be predicted by the inside attacker, thus excluding it from the network.

4 The Split-Pairing Scheme

Our basic idea is to split the jammed nodes into two groups, each of which works on a different channel. Since at any given time the attacker can jam only one channel, or neither of them when it is switching channel, the group free of jamming may conduct key propagation. The scheme consists of three phases. Phase I deals with how to split the network into two groups and assign communication channels to them. Then, we design the protocol for intra-group key propagation in phase II to ensure that all nodes in one of the two groups will share the new key at the end of this phase. In phase III, nodes in two groups are paired to propagate the new key from one group to the other.

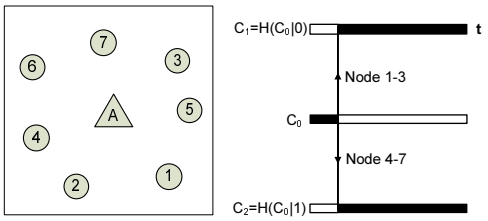


Fig. 1. (a) Network topology (b) The illustration of channel switch for key reestablishment.

4.1 Phase I: Channel Splitting

Suppose all nodes work on channel C_0 originally and r channels available to switch. Starting from time t_0, one node is compromised and begins to jam channel C_0. After the

jammer has been detected and identified, all N non-compromised nodes will be aware of it. They will switch to new channels in a distributed way. Without loss of generality, let us denote their ids as $1, \cdots, N$. In this phase, nodes with lower ids $\{1, \cdots, \lfloor \frac{N}{2} \rfloor\}$ switch to channel $C_1 = H(C_0|0)$, and nodes with higher ids $\{\lfloor \frac{N}{2} \rfloor + 1, \cdots, N\}$ switch to channel $C_2 = H(C_0|1)$, where H is a secure hash function preloaded in the sensor nodes and maps to one of r channels.

The channel switching and splitting process is illustrated in Figure 1(a) and Figure 1(b). When node A is identified as a compromised node, nodes with lower ids, i.e., nodes 1, 2 and 3, switch to channel $C_1 = H(C_0|0)$ and nodes in higher ids, i.e., nodes 4, 5, 6 and 7, switch to channel $C_2 = H(C_0|1)$.

4.2 Phase II: Jamming and Key Propagation within A Group

Once channel splitting finishes, the node with the smallest id in each group acts as the group leader to generate a new group key, which is then propagated within each group. That is, node 1 is the group leader of the first group, and node $\lfloor \frac{N}{2} \rfloor + 1$ is the group leader of the second group. Then, the new group key K is generated based on the pairwise key $K_{1,\lfloor \frac{N}{2} \rfloor + 1}$ shared between two leaders by applying $K = F_{(K_{1,\lfloor \frac{N}{2} \rfloor + 1})}(0)$, where F is a pseudorandom function. The desirable advantage is that the new group key is generated without any communication and thus the jammer cannot interfere it. Since the key $K_{1,\lfloor \frac{N}{2} \rfloor + 1}$ is unknown to the attacker, it cannot predict the new group key although the pseudorandom function F is publicly known.

Once the group leaders have generated the same new key, they will only need to propagate the new key to all their group members. Clearly, the new key has to be encrypted to preclude the compromised attacker from eavesdropping. To propagate K, the simple solution is to let the group leader unicast the key to each group member. To save communication cost, we use reliable broadcast. Specifically, the group leader broadcasts the key to all group members and gets the acknowledgements (acks) from each of them. The group leader will retry if any acks are missing.

Specifically, in the broadcast message M_1, the new key K is encrypted by different pairwise keys shared between the leader and each member. For group 1, node 1 broadcasts M_1 and starts a timer

$$M_1 = Mapping||E_{K_{1,2}}(T|2|K)||...||E_{K_{1,\lfloor \frac{N}{2} \rfloor}}(T|\lfloor \frac{N}{2} \rfloor|K).$$

where T is the $timestamp$ to prevent replay attacks. After successfully receiving and decrypting M_1, node i sends back a confirmation message to the group leader 1 or $\lfloor \frac{N}{2} \rfloor + 1$. For group 1, node i sends back

$$M_2 = E_{K_{1,i}}(T|i|K)||i.$$

If any confirmations are missing due to jamming or collision, a new key propagation message M_1 is reconstructed and sent out after timeout. Only unconfirmed nodes are required to send back confirmations to reduce the traffic and collision. This procedure continues until all confirmations are received by the leader.

In TinyOS 2.0.1, the MAC layer frame structure has a data payload of 28bytes. Given a typical key size of 8 bytes [19], one frame can include at most 3 encryptions of a group key. Also, node ID is 1 byte and encryption id is 1byte. For Mica2 with transmission rate of 19.2Kbps, the transmission time for M_1 with three encryptions (i.e., the subgroup size is 4 counting the leader) is $\tau_1 \approx \frac{(8Bytes*3)+1Byte)}{19.2Kbps} = 10.42ms$ and for M_2 is $\tau_2 \approx \frac{(8+1)Bytes}{19.2Kbps} = 3.75ms$, the one-round communication time will be $\tau_0 = \tau_1 + 3 * \tau_2 = 21.67ms$. It is worth noting that the one-round time $\tau_0 < t_l$, where t_l=34ms for Mica2. That is, for a group of size 4, a keying message can be transmitted successfully during the course of the jammer switching to another channel, jamming a minimal packet, and returning. If the group size is larger than 4, we need to embed the multiple encryptions into two or more broadcast messages. Suppose that the key propagation time in one group without jamming is T_{kr}. Given the number of nodes in each group and the packet loss rate, we can compute the expected message transmission round $E[Y]$ based on [25]. Hence, $T_{kr} = \tau_0 E[Y]$.

Unfortunately, in practice the key propagation messages M_1 and M_2 can be corrupted by jamming and the actual key propagation needs more time. In order to estimate this time, we consider the optimal jamming strategies in which the attacker can maximize the total key propagation time for this phase. Since the hash function H and the original channel C_0 are publicly known, the attacker knows channels C_1 and C_2 by computing the same hash values. However, the attacker has only one air interface and thus at any given time it can jam only one channel or neither of them when it is switching channel. This means that at least one of two groups are free of jamming at any time, and this group can execute the above key propagation protocol. In other words, the attacker cannot simultaneously prevent the key reestablishment for both groups and the best it can do is to prolong the key propagation time of phase II.

Theorem 1. *The optimal jamming strategy for a single jammer is to actively jam two channels with an equal probability.*

Proof. We denote T_j as the overall jamming duration in phase II. The total key propagation time for Phase II is T. In our system model, p_i is the probability for the attacker to launch jamming on channel i. For group i, the time it is free of jamming is $T_i = T - T_j p_i$. In order to maximize the key propagation time, an optimal attacker would minimize the maximum free-of-jamming time for all groups. Here we consider the case of two groups $i = 1, 2$. We formalize the optimization problem as follows:

$$\begin{aligned} &\min_{p_1,p_2} \max_{p_1,p_2} (T - T_j p_1, T - T_j p_2) \\ &s.t. p_1 + p_2 = 1 \\ &p_{1,2} \geq 0 \end{aligned} \tag{1}$$

If $T - T_j p_1 \geq T - T_j p_2$, we have $p_1 \leq p_2$. Then, the problem is simplified to:

$$\min_{p_1 \geq 0, p_1 \leq p_2, p_1 + p_2 = 1} (T - T_j p_1) \tag{2}$$

The solution is $p_1 = p_2 = 0.5$. Similarly, we have the same result when $T - T_j p_2 \geq T - T_j p_1$.

To estimate the key propagation time T in one group, we consider a typical optimal case for the attacker where the attacker alternates between two channels and jams each channel for a period of t_j. If it starts with group 1, group 2 will be able to complete key propagation ahead of group one or at the same time as group one. We consider the worst case in which each jam leads to a retransmission. The number of retransmissions for one group due to jamming is $\frac{T}{2(t_j+t_l)}$ and the time for retransmission is $T_{jr} \approx \frac{T}{2(t_j+t_l)}\tau_0$. The finish time T is

$$T \approx T_{kr} + T_{jr} = \frac{2(t_j + t_l)}{2(t_j + t_l) - \tau_0} * T_{kr}. \tag{3}$$

4.3 Phase III: Key Propagation between Groups

After one group finishes the key propagation, this group excludes the attacker by the keyed secret channel. It is possible that the attacker chooses to jam group 2 all the way so that few nodes in group 2 can obtain the new group key. If so, nodes in group 1 can propagate the group key to nodes in group 2 by pairing one node in group 1 with another node in group 2. For simplicity, we pair the two nodes with the lowest ids in two groups, the second lowest and so on. If N is odd, group 2 will have one more node left. We pair it to node 1 since the two lowest id nodes, 1 and $\lfloor \frac{N}{2} \rfloor + 1$, are group leaders they do not need to communicate in this phase. Therefore, node 1 is actually only responsible for that extra node. That is better than pairing this extra node to any other node in group 1, which is already paired. In Figure 1(a), we pair node 1, 4; 2, 5; 3, 6 and 1, 7.

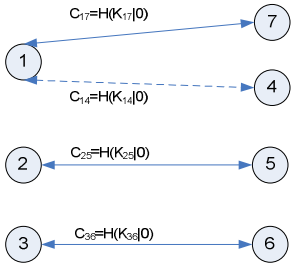


Fig. 2. Network pairing for the illustrated network in Fig.1.

In order to safely propagate the new group key from one group to the other, paired parties in different groups communicate in a keyed secret channel based on their pairwise key. Suppose node $i(1 \leq i \leq \lfloor \frac{N}{2} \rfloor)$ and $j(\lfloor \frac{N}{2} \rfloor + 1 \leq j \leq N)$ are paired and they share a pairwise key K_{ij}. Then, they switch to channel $C_{ij} = H(K_{ij}|0)$. In some rare cases, two or more pairs are hashed to the same channel due to the limited channel resource. We use random back-off mechanism to avoid collision. In Figure 2, we show the pairing and channel switching of the network in Figure 1(a).

After channel switching, all nodes that have received the new group key switch to the reception mode and wait for a request from their paired parties. For the key propagation, since phase II can guarantee that nodes in one group have correctly received the new group key, two cases may occur for the pair i and j. One is that both i and j have correctly received the new group key. In this case, i and j do not communicate to save

energy and avoid unnecessary traffic and collision. The other is that either i or j has received the new group key. Without loss of generality, we assume that i has received the new key but j has not. In this case, j initiates key reestablishment by sending a message M_1 to node i:

$$M_1 = T||j||MAC_{K_{ij}}(T|j).$$

where T is a timestamp and MAC is a message authentication algorithm. Node i replies to j message M_2:

$$M_2 = E_{K_{ij}}(T|i|K).$$

node j decrypts message M_2 to obtain K. Note that here M_2 does not include a separate MAC because the knowledge of T and i serves as a way of (weak) authentication. Last, node j returns a confirmation message M_3 to i:

$$M_3 = E_{K_{ij}}(T|j|K).$$

Note that given a typical size of 4-byte MAC [19] all three messages are short and the time for this exchange for the Mica2 mote is $\tau_3 < \frac{8Bytes*3}{19.2Kbps} = 10ms$, which can be completed within t_l. In other words, as long as attacker is jamming a channel other than C_{ij} at the beginning of this phase, inter-group communication of pair ij can complete without jamming. To deal with some rare case that the attacker has chance to jam the communication on pair ij, paired nodes maintain a timer and the timeout could be set to τ_3 or a bit more to tolerate lost time synchronization. Since nodes can detect failed packet [14], if any exchange message is detected to be failed, paired parties stop the exchange protocol and wait for a timeout. When a timeout occurs, they switch to another channel $C'_{ij} = H(K_{ij}|1)$, set timer and retry until one party can successfully propagate the new group key to the other.

5 Sensor Testbed and Metrics

5.1 Testbed Configurations and Implementation of the Jammer

The testbed consists of 17 Mica2 motes [15] deployed at fixed locations in an indoor laboratory. Each sensor mote has a 902-928MHz Chipcon CC1000 radio, which has 32 800KHz channels. Each mote is within the communication range of other motes and the transmission rate is 19.2Kbps. All motes run TinyOS version 2.0.1 [23].

In TinyOS 2.0.1, the module *CC1000ControlP* provides interface *CC1000Control* and command *tuneManual()* to control channel switching. Since Chipcon CC1000 uses a digital frequency synthesizer, a programmable register can be used to change the frequency and then achieve channel switching.

In TinyOS 2.0.1, the implementation of the mote-to-mote communication depends on the radio chip. For Chipcon CC1000, the communication is implemented in two modules: *CC1000CsmaP* and *CC1000SendReceiveP* under directory *tinyos-2.x/tos/chips/cc1000*. *CC1000CsmaP* provides CSMA and low-power sensing logic, whereas *CC1000SendReceiveP* provides the send-and-receive logic for CC1000 radio. The send-and-receive logic includes Request-to-Send (RTS) and Clear-to-Send (CTS) commands. A node starts data transmission after receiving CTS. CSMA provides two mechanisms for media access control: random backoff and carrier sensing. The random backoff mechanism is used

to reduce further collisions where the backoff delay is randomly set to [1,32] bytes initially. The sensing mechanism is used to determine if there is any ongoing communication on the channel. It requires the air interface to read received signal strength indication (RSSI) every 80 microseconds up to 5 readings. If all 5 readings are above a threshold, the backoff mechanism is activated. After each RSSI reading, the threshold is updated and thus it is is adaptively changed with the current channel condition.

We modify the TinyOS source code to implement the jammer. We disable the random backoff and the sensing mechanisms so that the jammer can send out packets arbitrarily to jam the channel. Specifically, we use command *disableCca()* provided by the *CsmaControl* interface in module *CC1000CsmaP* to bypass the media access control. We let the jammer's air interface stay in the transmission mode by using *enterTXState()*. We change the send-and-receive logic so that the jammer always receives CTS after sending a RTS.

In order to explore the impact of jamming duration, we bypass the MAC layer and directly use the command *writeByte()* provided by the interface *HplCC1000Spi*. In this way, the shortest jamming time can be as low as one byte ($t_j \approx 0.42ms$). For longer jamming duration, we have to increase the maximum message size defined in *message.h*, so that the jamming signal can last as long as 100ms.

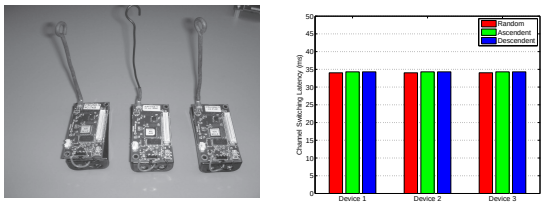
5.2 Performance Metrics

In our recovery scheme, we assume that the physical device has channel switching latency; thus, we first measure the switching latency for Mica2 motes. For the evaluation, we focus on measuring the recovery latency as the number of jammers and the jamming duration change. We also consider the size of the network in these measurements.

6 Experimental Results

6.1 Channel Switching Latency

In order to jam a communication channel, the attacker has to switch to that communication channel and send out at least a packet of 1 byte for the CC1000 chip. There is a minimum channel switching latency due to the limitations of the physical device. Three Mica2 motes as shown in Figure 3(a) are selected to measure this channel switching latency. We consider two switching modes: sequential switching and random switching. In the sequential switching mode, motes switch to one channel and send one minimum packet, then they switch to the next adjacent channel until all 32 channels are used. We consider two cases, ascendant and descendent. In the ascendant case, motes start from the lowest frequency channel to the highest, while the descendent case uses the reverse order. We run the test 1000 times for both cases. We get the average and divide it by 32 to get the switching latency between two adjacent channels. In the random switching mode, motes randomly select the next channel. Similar to the sequential mode, we run the test 1000 times to get the switching latency between two arbitrary channels. As shown in figure 3(b), the switching latency is independent of the channel switching mode, and it is around 34ms for all three motes.



(a) (b)

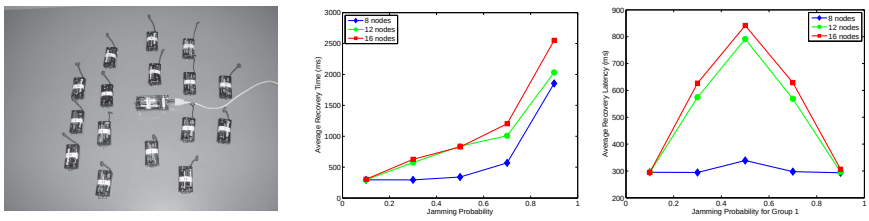
Fig. 3. (a) Three Mica2 motes are used for measuring the channel switching latency. (b) The channel switching latency for the three Mica2 motes

6.2 The Performance of the Split-Pairing Scheme

In this subsection, we conduct experiments to study the effectiveness of the split-pairing scheme described in Section 4, in which we consider a single jammer and the network is split into two groups. We measure the impact of the following two parameters: jamming probability and jamming duration.

The Impact of Jamming Probability Since the jammer can only jam one channel at a time, it selects one of the two channels used for intra-group communication with some probability (the jamming probability) and sends a minimum size packet, then it repeats this process. We will measure how the jamming probability affects the recovery latency.

We deploy 8, 12 and 16 nodes in the network and manually put a jammer in the center of the network to ensure that it can jam all the nodes. Legitimate nodes are split into two groups of 4, 6 and 8 nodes respectively. The network with 16 nodes and one jammer is shown in Figure 4(a). We set the retransmission timeout to be 250ms since one round of communication should be finished within 250ms. For different network size, we measure the recovery latency of the splitting phase for both groups by running our scheme 20 times and compute their average.



(a) (b) (c)

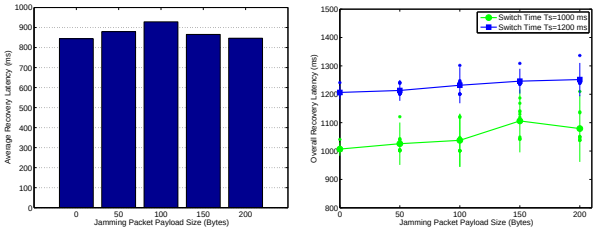
Fig. 4. (a) A network with 16 legitimate nodes and one jammer. (b) The recovery latency of the splitting phase (Phase I and II) for a single group. (c) the recovery latency of the splitting phase (Phase I and II) which is the minimum latency of both groups

Figure 4(b) shows the average recovery latency of one group. As can be seen, the average latency increases with the jamming probability since nodes have to retransmit after the data is jammed. For a group of 4 nodes (the 8-node line in the figure considering there are two groups), the recovery latency does not change too much as the jamming probability increases from 0.1 to 0.3. This is because all versions of the group

key can be embedded into one message which makes the key propagation message (M_1) less vulnerable of being jammed. However, when the group size increases to 6 or 8 nodes, different versions of the group key have to be split into two messages, and either one being jammed will lead to a retransmission, thus increasing the recovery latency. Moreover, as the network size increases, more confirmation messages (M_2) are required for key propagation and are more likely to be jammed, thus further increasing the recovery latency.

Figure 4(c) shows the recovery latency of the splitting phase in our scheme, which is the minimum of both groups. Since the jammer cannot jam two groups simultaneously, jamming one group always means free of jamming in the other group. After the jamming probability of group 1 is larger than 0.5, the minimum recovery latency should be the latency of group 2. This explains why the recovery latency starts to decrease after the jamming probability is larger than 0.5. When the jamming jamming probability is 0.5, the recovery latency reaches the highest point, which is consistent with our results on optimal jammer.

The Impact of Jamming Duration In this subsection, we evaluate the impact of the jamming duration. We deploy a network of 16 nodes and fix the jamming probability to be 0.5. The retransmission time is set to be 250ms in Phase II and 70ms in Phase III. We add 0, 50, 100, 150 and 200 bytes to the jamming packet to construct different jamming durations.



(a) (b)

Fig. 5. (a) The recovery latency of the splitting phase (Phase I and II) under different jamming duration (Jamming probability=0.5, Network size=16 nodes) (b) Recovery latency for the split-pairing scheme (including all 3 phases) under different jamming duration (Jamming probability=0.5, Network size=16 nodes)

Figure 5(a) shows the average recovery latency of the splitting phase by running our scheme 20 times. As can be seen, the recovery latency increases when the packet size increases from 0 to 100 bytes, and then decreases when the packet size increases from 100 bytes to 200 bytes. When the packet size increases from 0 to 100 bytes, the recovery latency is longer since the channel is jammed longer, and ongoing messages are more likely to be jammed and retransmitted. However, when the jammer stays in one group longer (100-200 bytes), the other group has larger chance to finish its intra-group communication. Since the recovery latency is the minimum key propagation time of both groups, the splitting phase completes as long as one group finishes the key propagation. Thus, jamming in one group longer gives the opportunity for the other group to finish earlier without any interruption, thus reducing the recovery latency.

Figure 5(b) shown the recovery latency of the split-pairing scheme including all three phases. We set the phase II to III Let T_s denote the switch time from phase II to phase III. We consider two cases $T_s = 1000ms$ and $T_s = 1200ms$. This is because the splitting phase can be finished between 4 to 5 broadcast rounds. Since the retransmission timeout is 250ms, the splitting phase should be finished between time 250*4=1000ms to 1250ms. If we set T_s smaller than 1000ms, the splitting phase may not complete. If we set T_s larger than 1250ms, both groups may have finished the key propagation and the pairing phase (Phase III) is not required any more. By setting time to be 1000ms and 1200ms, we can investigate the impact of the jamming duration for both splitting phase (Phase II) and pairing phase (Phase III). For each jamming duration and switch time, we record the overall latency, and we repeat the experiment 20 times. We also compute the mean and the 95% confidence interval shown as vertical bar in Figure 5(b).

For $T_s = 1200ms$, the latency does not change too much compared with the case of $T_s = 1000ms$. Given Figure 5(a), both groups have adequate time to finish the key propagation and therefore less communication is needed in the pairing phase. However, when the jamming duration increases, the latency slightly increases and the variability becomes larger. This is because the recovery difference between two groups in the splitting phase becomes more significant with longer jamming duration, thus more communication is needed in the pairing phase. This trend becomes more obvious with $t_s = 1000ms$. With $T_s = 1000ms$, the latency increases significantly between 100-150Bytes and declines between 150-200Bytes. Since pairing in phase III needs more communication when the jamming duration increases, the random scan of the jammer in the pairing phase may have more chances to corrupt the communication and more messages are needed to be retransmitted. Therefore, the latency becomes larger. However, when the jamming duration increases, the jammer can scan less number of channels for a given period of time which reduces the chance of packets being jammed, thus the overall recovery latency becomes smaller.

7 Discussion

Our scheme can be extended to the multiple-jammer case. In this case, we split the network (excluding the jammer) into N_1 groups where $N_1 >$ number of jammers and each group has a leader whose id is publicly determined, like in our basic scheme. Then, a multi-variate version of the Blundo scheme, such as that in [32], can be used to derive a global group key K with these N_1 leader ids as input to the polynomial, similar to the process in Phase II of the basic scheme. By far no communication is involved. Next, each group leader tries to distribute the key K to its group members. Because $N_1 >$number of jammers, at least one group (called recovered group) will be able to share K to all its group members. In the pairing phase, one-to-one pairing is replaced by one-to-many pairing where we pair one node from the recovered group with nodes each from each subgroup to be recovered. Again, a multivariate version of the Blundo scheme can be used to calculate a pairing key for each set of nodes only given the node ids as the input. Finally, a pairing key is used to select a new communication channel and securely deliver K to the other nodes in each paired group. Through these three similar phases, all the nodes will know the same global key K.

By far our presentation is based on the Mica2 mote platform, but our scheme can be applied to some other platforms too. For example, Atheros 802.11 WiFi chipset has the channel switching latency of 7.6ms. Given the transmission rate of WiFi 54Mb/s and a key size of 256bits, more than 1500 keys can be transmitted within one switching latency. Thus, our scheme works much more effectively for the WiFi platform.

For the MicaZ sensor, the channel switching latency is 132us; however, the minimum time for key propagation communication is 424us [24]. It consists of the time for the jammer to leave the key propagation channel, send a minimum packet and then return. Given the transmission rate of 250Kbps, only about 13 bytes could be transmitted. Considering the MAC frame header and key size, 13 bytes are not enough to transmit one key. To deal with this difficulty, we can apply the chained hash fragmentation [3]. The basic idea is to divide a large frame into small fragments. By hashing fragment cyclically, fragments can be linked to reconstruct the original frame after receiving all fragments.

8 Conclusions and Future Work

In this paper, we consider the insider jamming problem in a one-hop network and propose a compromise-resilient jamming recovery scheme. We exploit the fact that the jammer can only work on one channel for any given time and nodes in the other channels will be free of jamming which can execute recovery. In the evaluation, we implement our scheme on the Mica2 mote platform and show that the solution is efficient and has low recovery latency.

To the best of our knowledge, this is the first paper to address the inside jamming issue in WSNs. As our initial work, we do not expect to solve all the problems. In the future, we will further investigate the efficient solutions for multiple colluding inside jammers. Also, we will study how to recover a multi-hop network under the presence of jamming.

9 Acknowledgment

This work was supported in part by Army Research Office under MURI grant W911NF-07-1-0318 and NSF CAREER 0643906.

References

[1] J. G. Proakis: Digital Communications, 4th edition. McGraw-Hill, (2000)
[2] C. Schleher: Electronic Warfare in the Information Age. MArtech House, (1999)
[3] M. Strasser, C. Popper, S. Capkun, M. Cagalj: Jamming-resistant Key Establishment using Uncoordinated Frequency Hopping. IEEE Symposium on Security and Privacy, (2008)
[4] M. Strasser, C. Popper, S. Capkun: Efficient Uncoordinated FHSS Anti-Jamming Communication. ACM Mobihoc, (2009)
[5] C. Popper, M. Strasser, S. Capkun: Jamming-resistant Broadcast Communication without Shared Keys. USENIX Security Symposium, (2009)
[6] W. Xu, W. Trappe, Y. Zhang: Channel Surfing: Defending Wireless Sensor Networks from Jamming and Interference. ACM IPSN (2007)

[7] M. Li, I. Koutsopoulos, R. Poovendran: Optimal Jamming Attacks and Network Defense Policies in Wireless Sensor Networks. IEEE Infocom, (2007)
[8] W. Xu, W. Trappe, Y. Zhang, T. Wood: The Feasibility of Launching and Detecting Jamming Attacks in Wireless Networks. ACM Mobihoc (2005)
[9] A. D. Wood, J. A. Stankovic, S. H. Son: JAM: A Jammed-Area Mapping Service for Sensor Networks. Proc. of the 24th IEEE International Real-Time Systems Symposium, (2003)
[10] R. Poisel: Modern Communications Jamming Principles and Techniques. Artech House Publisher, (2006)
[11] T. Brown, J. James, A. Sethi: Jamming and sensing of encrypted wireless ad hoc networks. ACM Mobihoc, (2006)
[12] L. Lazos, S. Liu, M. Krunz: Mitigating Control-Channel Jamming Attacks in Multi-Channel Ad Hoc Networks. ACM WiSec, (2009)
[13] P. Tague, M. Li, R. Poovendran: Mitigation of Control Channel Jamming under Node Capture Attacks. IEEE Transactions on Mobile Computing, (2009)
[14] W. Xu, W. Trappe, Y. Zhang: AntiJamming Timing Channels for Wireless Networks. ACM WiSec, (2008)
[15] Chipcon CC1000 Radio's Datasheet: http://www.chipcon.com
[16] H. Chan, A. Perrig, D. Song: Random Key Predistribution Schemes for Sensor Networks. IEEE Security and Privacy Symposim, (2003)
[17] D. Liu, P. Ning, R. Li: Establishing Pairwise Keys in Distributed Sensor Networks. ACM CCS, (2003)
[18] S. Zhu, S. Setia, S. Jajodia: LEAP+: Efficient Security Mechanisms for Large-scale Distributed Sensor Networks. ACM TOSN, (2006)
[19] C. Karlof, N. Sastry, D. Wagner: TinySec: A Link Layer Security Architecture for Wireless Sensor Networks. ACM SenSys, (2004)
[20] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, M. Yung: Perfectly-secure key distribution for dynamic conferences. Advances in Cryptology, Proc. of CRYPTO, (1993)
[21] V. Navda, A. Bohra, S. Ganguly, D. Rubenstein: Using Channel Hopping to Increase 802.11 Resilience to Jamming Attacks. IEEE Infocom, (2007)
[22] O. Ureten, N. Serinken: Wireless security through RF fingerprinting. Canadian Journal of Electrical and Computer Engineering, (2007)
[23] Tinyos homepage: http://webs.cs.berkeley.edu/tos/
[24] A. Wood, J. Stankovic, G. Zhou: DEEJAM: Defeating Energy-Efficient Jamming in IEEE 802.15.4-based Wireless Networks. IEEE SECON, (2007)
[25] D. Towsley, J. Kurose, S. Pingali: A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols. IEEE Journal on Selected Areas in Communications, (1997)
[26] B. Danev, S. Capkun: Transient-based identification of wireless sensor nodes. ACM/IEEE IPSN, (2009)
[27] Y. Sun, X. Wang: Jammer localization in wireless sensor networks. Proc. of the 5th International Conference on Wireless communications, networking and mobile computing, (2009)
[28] A. Seshadri, A. Perrig, L. van Doorn, P. Khosla: Swatt: Software-based attestation for embedded devices. IEEE Symposium on Security and Privacy, (2004)
[29] M. Shaneck, K. Mahadevan, V. Kher, Y. Kim: Remote software-based attestation for wireless sensors. ESAS, (2005)
[30] T. Park, K. G. Shin: Soft tamper-proofing via program integrity verification in wireless sensor networks. IEEE Transactions on Mobile Computing, (2005)
[31] Y. Yang, X. Wang, S. Zhu, G. Cao: Distributed softwarebased attestation for node compromise detection in sensor networks. IEEE SRDS, (2007)
[32] Y. Zhou and Y. Fang: A two-layer key establishment scheme for wireless sensor networks. IEEE Transactions on Mobile Computing, (2007)