Self-Healing Wireless Networks under Insider Jamming Attacks

Longquan Li*, Sencun Zhu*, Don Torrieri†, Sushil Jajodia‡
*The Pennsylvania State University, University Park, PA
†US Army Research Laboratory, Adelphi, MD
‡George Mason University, Fairfax, VA

***Abstract*—As jamming is a very serious threat to the normal operation of wireless networks, recently much research has been done to deal with it. Different from most existing research, In this work, we tackle the jamming attack problem in a systematic way. Specifically, we design a protocol that is capable of self-healing wireless networks under jamming attacks. The protocol identifies and excludes an insider jammer and then restores normal data communications among benign nodes despite the presence of jamming by an initially unknown compromised node. Our scheme integrates key management, jammer identification and jammer isolation in one system. We evaluated the protocol with USRP devices and GNURadio in the context of jammer localization. The experiments show that our protocol can identify and isolate the insider jammer with high accuracy.**

Keywords*—*Self-healing; Insider Jamming Attacks; Jammer Identification; Wireless Networks; Node Revocation.

I. Introduction

Wireless networks are currently deployed in many applications, e.g., military surveillance, environmental monitoring, home automation. However, the wireless nature of such networks exposes them to an easy while powerful attack called *jamming*. By emitting noisy radio signals to decrease signal-to-noise-ratio, jamming attack can very severely interfere the normal communication of a wireless network or even completely paralyze it.

Jamming attacks mainly fall into two categories: external jamming and insider jamming. Suppose there are totally M carrier frequencies and each of them specifies a distinct network channel. An external jammer randomly picks a channel to jam at one time, and the hit ratio (the possibility of jamming the channel being used) is only $1/M$, which is not efficient when M is large. Moreover, it is relatively easy for all the benign nodes in the network to switch to a new channel to evade jamming. Some conventional anti-jamming measures, such as Frequency Hopping Spread Spectrum(FHSS) and Direct Sequence Spread Spectrum(DSSS) [1], [2], can effectively thwart external jamming attacks; however they are unable to handle insider jamming attacks.

An insider jammer knows all the network-wide secrets as other benign nodes do, so it is very hard for the benign nodes to reestablish a new communication channel securely. In recent years, even though many methods have been proposed to deal with jamming attacks, most of them only focus on one or two aspects of the problem, e.g., jamming detection, jammer localization, jammer identification or jammer isolation. For a real network system, what is desirable is a complete solution that enables the network to heal itself under insider jamming attacks. That is, the network should be able to identify the jammer, exclude it by updating the network secrets, and restore the normal network communications. We call this process the *self-healing* capability.

In this work, we design a systematic solution for self-healing wireless networks under insider jamming attacks. Different from some existing work [3], [4], [5], which attempt to tolerate the jamming and ends up transmitting multiple copies of the same data packet in different channels, our protocol actively identifies the insider jammer, excludes it, and *restores the network into the initial state* where all benign nodes communicate data in a single channel. As a result, only one copy of each data packet is needed, minimizing the regular communication overhead. Moreover, our protocol integrates the processes of jammer identification, jammer revocation and key management into one protocol. It is *fully distributed* without relying on any trusted central entity, and each node can be a transmitter and receiver, whereas many existing work [3], [4], [5] assume a single transmitter (acting as the trusted party) and multiple receivers (one (or some) receiver is the jammer). It is jamming resilient in the sense that the protocol tolerates jamming *at any time and at any channel*. We evaluate our protocol with USRP devices and GNURadio in the context of jammer localization. The results show that on average the actual jammer can be identified and excluded by running our system slightly over one time.

II. Preliminaries

A. Wireless Network Model

First of all, as in most existing work [3], [6], [7], [8], we assume a one-hop network where nodes can hear each other. The nodes exchange data messages through broadcasting in a common channel, which is the most efficient way for group communication. Wireless sensor networks and mobile ad hoc networks mostly adopt this group communication model. They can either transmit or receive signals at any time. The nodes have similar capability with respect to signal sensitivity and jamming perception. Hence, they are able to recognize the state whether the communication channel is jammed or not, e.g., by receiving a threshold number of junk packets [9], [10]. Note that in our work we rely on existing mechanisms [9], [10] for jamming detection, and our focus is on how to recover from jamming by an unknown insider jammer.

For security purpose, each pair of nodes share a pairwise key (e.g., K_{xy} for nodes x and y), which can be used to derive a secret channel C_{xy} for their individual communication. For example, $C_{xy} = H(K_{xy}, i)$ for nodes x and y, where H is a secure hash function that maps its input to a random channel

number, and i is a sequence number. There are a number of efficient pairwise key establishment schemes [11], which allow two nodes to establish a pairwise key on the fly as long as they know each other's id.

For concreteness, our protocol is presented in the context of a wireless network that communicates in a single data channel unless switching is needed to avoid jamming. However, it can also be adapted to work in networks with either FHSS or DSSS deployed in the physical layer. Although in both contexts insider jamming is easy to perform against the common channels, in FHSS or DSSS, external jamming (random jamming) against the communication between benign nodes that use private hopping/spreading sequences has little chance to succeed. As a result, our protocol would work more easily without having to deal with packet losses caused by jamming, and accordingly its design could be much simplified by removing redundant transmissions.

B. Jamming Model

Jamming attacks may be caused by an external jammer that does not know the network secret. The external jammer randomly scans all the channels and might be able to find the one currently used by the network. However, such an attack is relatively easy to address. For example, let the original channel used before jamming be $C = H(K, 0)$, where K is the secret key shared by all benign nodes. After jamming is detected, all nodes switch to a different channel $C = H(K, 1)$ to evade jamming. This process can be repeatedly executed. As such, we do not study external jamming attacks in this work but focus on insider jamming attacks.

For insider jammer who knows the initial network-wide secret and understands the protocol, we assume it can jam *at any time and any channel of its choice*, or disrupt our protocol *by injecting false information at any time* as a legitimate participant before it is exposed. The insider jammer is only limited by two constraints: (1) it takes time to switch between physical channels (elaborated further below), and (2) it cannot predict the channels used by benign nodes when the channels are determined by cryptographic keys unknown to the jammer.

For ease of presentation, we assume that during one time slot of duration T_s, a jammer can jam at most one channel. This is because when a jammer switches channels to jam, there is a channel switch latency T_l. To clarify, T_l is also a channel switch latency for benign nodes. Measurements show that the typical latency is 34ms for Mica2 wireless sensor motes [12] and $7.6ms$ for Atheros WiFi chipset [13]. For example, for a Mica2 mote with transmission rate of 19.2Kbps, it can transmit 81 bytes during the time when a jammer switches to another channel. For WiFi that has the transmission rate of WiFi 54Mb/s, a device can transmit 53KB of data within one switching latency (7.6ms). As we will see later, all the messages involved in our protocol are relatively small for a one-hop network, so the jammer will not be able to jam the communications of nodes in multiple channels within a single time slot of length T_s as long as we set T_s appropriately. Specifically, we let T_s satisfy

$$T_l + T_c \leq T_s \leq 2T_l + T_{jc}, \tag{1}$$

where T_c denotes the maximal time for two benign nodes to exchange message once (i.e., one message followed by an immediate acknowledgement), and T_{jc} denotes the minimal time for the jammer to send a jamming signal (in the extreme case just one byte). Because $T_l + T_c \leq T_s$, we can guarantee that two benign nodes will have time to switch into a channel and communicate once within T_s. Because $T_s \leq 2T_l + T_{jc}$, we can guarantee that the jammer can only jam one channel in one time slot.

C. Design Goal

Under the above network and jamming models, *our ultimate design goal* is to empower a wireless network with *self-healing capability* under insider jamming attacks. By self-healing, we mean the insider jammer should be identified and excluded so that the benign nodes can restore their communications. Unlike some existing work [3], [4], [5], which only attempt to tolerate jamming and ends up transmitting multiple copies of the same data packet in different channels, our goal is to actively identify the insider jammer, exclude it, and restore the network into the initial state where all benign nodes communicate data in a single channel. If this goal is realized, only one copy of each data packet is needed, minimizing the regular communication overhead. Besides, we aim at making our scheme very general in two aspects. First, no special hardware is needed; otherwise, the scheme may not work for low-end sensor nodes. For example, while highly accurate localization algorithms exist, e.g., based on directional finding [14], it is assumed that wireless nodes have an array of antennas. Second, the scheme should adopt a general design principle that applies when some component of our system is replaced by techniques of the same type. For example, our protocol will not rely on a particular jammer identification algorithm. Instead, it can be any algorithm that takes any type of observations as its inputs, and the algorithm is allowed to have detection errors.

There are a number of technical challenges in achieving our design goal. *The biggest challenge comes from the fact that an insider jammer can have arbitrary behavior while fully understanding all operations of our protocol*. This is the most sophisticated attack model. Note that although in the literature there are a few well-known attack models, including constant jammer, deceptive jammer, reactive jammer, and random jammer [9], they cannot be directly applied here because we assume jamming can happen at any time and any channel during the operation of our protocol. Another challenge is due to the *distributed* nature of our protocol. There is no central trusted node in the system, and a node cannot trust any other node until the jammer is exposed or identified. Also, despite all attacks, our protocol should execute correctly in a distributed way. Given the complexity of the problem we are facing, at this stage of our research, we concentrate on a single insider jammer case. The understanding obtained from this study will shed light on solving the multiple insider jammer case in the future.

III. Proposed Scheme

A. System Overview

Fig. 1 depicts the flowchart of our system design. In the very beginning, there are totally N nodes, and they communicate in channel C_0 determined by the group key K_0 as $C_0 = H(K_0, 0)$. For convenience, we mainly elaborate the situation when N is an even number. The total number of

channels for the network is M with constraint of $M \geq N/2$. Now when there is a jamming attack detected, the nodes will react by changing to a new channel $C_i = H(K_0, i)$, where i is the i'th jamming detection. If it is an external jamming attack which happens by chance or through random scanning all channels, this type of frequency change will greatly thwart continuous jamming (especially in the case of FHSS or DSSS schemes). However, if more than a threshold number of jamming packets are detected within a fixed period of time, the attack is labeled as an insider jamming attack.

Then our protocol proceeds to the group splitting process (P1), which divides all the nodes (including the jammer, because we do not know who is the jammer yet) into two subgroups. Then inside each subgroup, a reliable group key distribution protocol (process P2) is executed to generate its own group key and then distribute the key to all the subgroup members, in the presence of possible jamming attack. Then in the process P3, nodes exchange collected jamming information to identify the jammer. Since most jammer identification algorithms are probabilistic, the output of P3 is a ranked suspect list. After that, in process P4, the two subgroups merge into a single group without the most suspicious node N_{j1}. That is, the group establishes a new group key that is not known to N_{j1} and starts to communicate in new channels derived from this secret key. Now the network is restored to the normal communication mode. However, if N_{j1} is not the real jammer, insider jamming would happen again. This indicates that the previous revocation decision was wrong. So the process P4 is repeated by taking back N_{j1} but revoking N_{j2}, the second most suspicious node. Through this trial-and-error process we can finally restore the network communication.

Fig. 2 shows the timeline for our protocol, where we can see the time for each process as well as the inter-process time gap T_w. The meaning of each time variable will be elaborated in later sections.

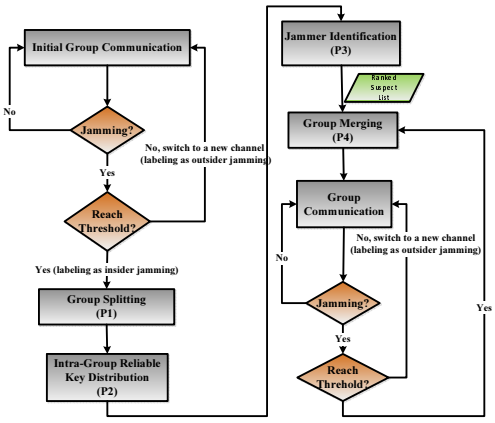


Fig. 1. Flowchart of our system design

B. Process P1: Group Splitting

After an insider jamming attack is determined, all nodes in the network will split into two groups in a distributed way. Without loss of generality, we assign these nodes with identifiers (ids) 1, $\cdots$, N. Nodes with smaller ids, i.e., $\{1, \cdots, \lfloor \frac{N}{2} \rfloor\}$, form a new group $G1$, while nodes with bigger ids, $\{\lfloor \frac{N}{2} \rfloor + 1, \cdots, N\}$, form the second group $G2$. In each (sub)group, the node with the smallest id acts as the leader of its group. For example, let us assume a group has nodes from 1 to 10, and node 3 is the insider jammer, shown in Fig. 3. After group splitting, group G1 has nodes 1 to 5 and its leader is node 1; and group G2 has nodes 6 to 10 and its leader is node 6. The time for group splitting is T_0 that is determined by specific network platforms. Note that this process is deterministic, so there is no communication involved.

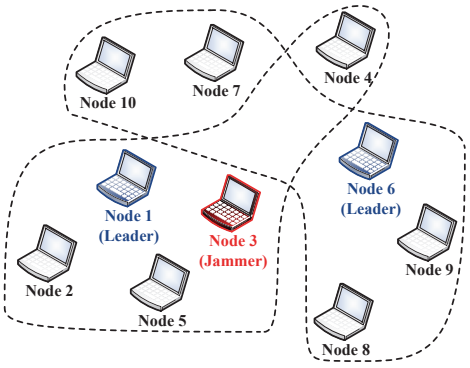


Fig. 3. The group splitting process

C. Process P2: Reliable Intra-group Key Distribution

Inside each subgroup, the group leader is responsible for generating and distributing a new subgroup key to its members. Because the jammer identifier is unknown at this stage, the jammer might be a group leader or a group member. In either case, it will get a copy of its subgroup key. The subgroup key, encrypted by pairwise keys known only to pairs of communicating nodes, is distributed through pairwise channels to avoid jamming. The process starts with the leader sending the encrypted subgroup key to another node of larger id, referred to as a *relay* node, in their pairwise channel in the first time slot. If the transmission is successful, in the next time slot, the relay node will also serve as a sender to relay the key to another node of larger id (a relay node could also be the jammer). Repeatedly more relays will be generated. The whole process is like tree growing and it is a deterministic process. That is, knowing its group member ids, every node can determine when and which channel to switch into and whom it is going to talk to in that channel. As the number of relay nodes doubles in each step, normally it will take $\lceil \log(N/2) \rceil$ time slots to reach all the subgroup members.

Because the jammer does not know the pairwise channels used by other nodes, over time it may randomly choose some channels to jam. As a result, jamming might occur in some pairwise channels in some time slots. In this case, some member nodes might not obtain the key in the end of their designated time slots. Fig. 4 shows an example scenario for group G2, where a "X" marked cell indicates a channel has been jammed, and a "\" marked cell indicates that even though there is no jamming in this channel, a relay cannot forward the key to another node because it did not obtain the key earlier, being jammed in its own scheduled receiving time slot. After 3 time slots (one round), nodes 7 and 9 still have not received their key. To address this jamming problem, execution of the entire key distribution process takes more than one round; for the next round, the pairwise channel used for each given pair of nodes x and y is changed (e.g., from $H(K_{xy}, 1)$ in the previous round to $H(K_{xy}, 2)$ in the next round).

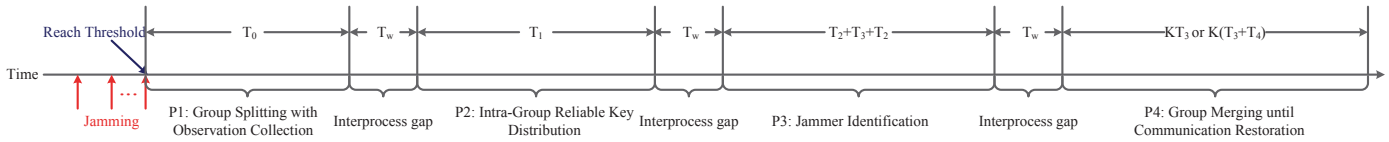


Fig. 2. Timeline of our protocol

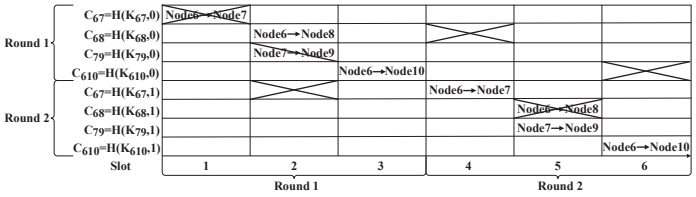


Fig. 4. An example scenario of intra-group reliable key distribution for group G2

Note that if the jammer becomes a leader or relay, it probably chooses to relay no key or relay random keys to confuse the other members in its own group. As a result, our protocol must survive under such attacks where the group with the jammer (referred to as the *j-group*) has malfunctioned while the group without the jammer (referred to as *nj-group*) has been randomly jammed. Considering such attacks, our protocol design aims to ensure the nj-group will establish its group key, even though we do not know which group is the nj-group. In theory, because of the unreliable nature of wireless communication and existence of persistent jamming attacks, there will be no guarantee for a group to reach any agreement within a time threshold. In practice, however, we can always make trade-off between reliability and transmission overhead. Specifically, we propose a system design parameter $\Phi_1 \in (0,1)$, which is the expected probability for all the members in the nj-group to acquire the same group key after the P2 process. Next we aim to determine λ_1, the minimal number of rounds to be executed in P2 so that Φ_1 is achieved.

Let $M(\geq N/2)$ be the number of available channels in the network and $N' = N/2$ the number of nodes in a subgroup. Denote π_0 as the probability that no channels used by the nj-group have been jammed in one round (each round has totally $\lceil \log(N') \rceil$ time slots). We can derive π_0 as the following.

$$\pi_0 = \frac{(\prod_{k=1}^{\lfloor \log N' \rfloor}(M-1-2^{k-1}))(M-1-(N'-2^{\lfloor \log N' \rfloor}))}{(M-1)^{\lfloor \log N' \rfloor+1}} \tag{2}$$

Accordingly, $1-\pi_0$ means the probability at least one pairwise channels are jammed in one round. Then we can derive $P_d(\lambda_1)$, a lower bound on the overall probability of successful key distribution after λ_1 rounds of key distributions.

$$P_d(\lambda_1) = \sum_{k=1}^{\lambda_1}(1-\pi_0)^{k-1}\pi_0 \tag{3}$$

$P_d(\lambda_1)$ is a lower bound because in some cases even if a pairwise channel is jammed, it would not affect the relay process. For example, in Fig. 4, Node 6 has relayed the key to Node 8 in round 1 . So even if the jammer is able to jam the pairwise channel in round 2 (time slot #5), it will not affect the protocol. Given the system parameter Φ_1, our task is to derive λ_1 according to Eqn. 3 such that $P_d(\lambda_1) \geq \Phi_1$.

Return to our example shown in Fig. 4. Let us assume $\Phi_1 = 99\%$. In this particular case, given $M = 40$ and $N' = 5$, then $\pi_0 = 0.900689$. By setting $\lambda_1 = 2$, Φ_1 is achieved. That is, after 2 rounds, all nodes in group G2 obtain the same group key with probability $\geq 99\%$.

The P2 process stops after λ_1 rounds. Because each round takes $\lceil \log N' \rceil$ time slots, the total time for P2 is $T_1 = \lambda_1 \lceil \log N' \rceil$ time slots. The nj-group has a probability of over Φ_1 to succeed. For the j-group, there are two possible cases. In case C1, the member nodes have failed to establish a subgroup key by the end of P2 because the jammer forwarded a false key or did not forward any key (as a relay or group leader); in case C2, the group has also successfully established a subgroup key which is known to the jammer because it followed the protocol for whatever reason. Regardless of its status in the end of P2, the j-group will terminate too and proceed to the next process P3 due to the deterministic nature of our protocol.

D. Process P3: Jammer Identification

Process P3 has three phases. First, nodes in each group try to share their earlier observations on the jamming attacks and such observations will be merged to identify the jammer. This is called intra-group observation sharing. The second is an inter-group observation sharing phase, where individual nodes from two groups pair up to share their observations in a channel determined by their pairwise key. Finally, each node uses valid observations to produce a ranked suspect list.

1) Phase I: Intra-Group Observation Sharing: The sharing of observations in this phase will be through broadcast in the channel determined by the newly established subgroup key. For the nj-group, this is easy to perform because its new key is unknown to the jammer. Although random jamming might affect its operation, it can be overcome by redundant transmission. Denote the group key in the nj-group as K_{nj}. Then nodes in the nj-group can share their observations in a new channel $C_{nj} = H(K_{nj}, 0)$. Without jamming, intra-group observation sharing can be finished within $N' = N/2$ time slots because each node broadcasts its observation once. However, random jamming may interrupt broadcast by chance. So, if jamming occurs in the nj-group, nodes will switch into a new channel $H(K_{nj}, i)$ after the i^{th} jamming.

For reliability concerns, we again propose a system design parameter $\Phi_2 \in (0,1)$, which is the expected probability of nodes in the nj-group obtaining observations from all other group members after phase I. Then, we derive a threshold λ_2 , which is the minimal number of time slots required to achieve Φ_2:

$$\sum_{N'}^{\lambda_2}\binom{\lambda_2-1}{\lambda_2-N'}(\frac{1}{M-1})^{\lambda_2-N'}(\frac{M-2}{M-1})^{N'} \geq \Phi_2 \tag{4}$$

Take group 2 as an example, and let $M = 40$ and $N' = 5$. As shown in Fig. 5, the broadcast channel is jammed at time

slot 3 (relative to the beginning of P3, when it is Node 8's turn to broadcast its observation). So the group switches into a new channel to continue the sharing process. In the end, the jammed transmission is repeated by Node 8. If one would like to achieve $\Phi_2 = 99\%$, then based on Eqn. 4, $\lambda_2 = 6$. That is, we only need one additional time slot to retransmit the previously jammed broadcast (no actual transmission is needed in this additional time slot if none of the earlier transmissions was jammed).

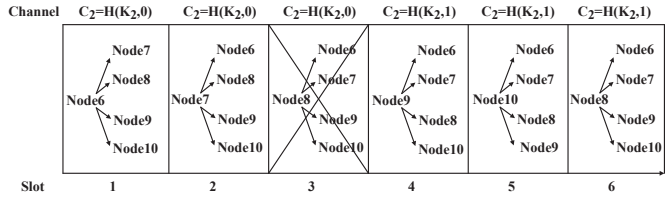


Fig. 5. Intra-group observation sharing for group G2

For the j-group, if it is in case C1, without a common group key, the nodes will not have the common channel to share observations. As a result, benign nodes in the j-group will not have received the observations from all its group members, so they will know they belong to a j-group. In the next phase, they will communicate such information to the nj-group. If it is in case C2, the nodes in the j-group may share observations as in the nj-group. However, the jammer may report false observations to avoid being identified.

For both j-group and nj-group, the overall time for this phase is $T_2 = \lambda_2$ time slots because our protocol proceeds in a deterministic way. By the end, the nodes in the nj-group have shared their observations. The nodes in the j-group may or may not have shared their observations. If not, the benign nodes know they are in a j-group.

2) Phase II: Inter-Group Observation Sharing: This phase is designed to allow all nodes to know whether they belong to the nj-group or the j-group if the jammer has prevented the j-group from sharing observations. If the jammer follows all the steps in the protocol, by the end of this phase, nodes will not know which group they belong to. In this case, the protocol moves on to the next phase.

Specifically, in the inter-group observation sharing phase, nodes between two groups pair up to exchange their observation lists via pairwise channels. Nodes with ids $\{1, \cdots, \lfloor \frac{N}{2} \rfloor\}$, always pair up with nodes with ids $\{1 + \lfloor \frac{N}{2} \rfloor, \cdots, \lfloor \frac{N}{2} \rfloor + \lfloor \frac{N}{2} \rfloor\}$, respectively. A node will share its N' observations (collected from its own group in phase I) with its counterpart in the other group. Again, this process may suffer from jamming and hence redundant transmission is deployed as a countermeasure. We propose a third system parameter $\Phi_3 \in (0, 1)$, which is the expected probability that benign pairwise nodes will finish exchanging all observations after this inter-group sharing phase. Then, we derive a threshold λ_3, which is the minimal number of time slots required to achieve Φ_3:

$$\frac{M - N'}{M - 1} + \sum_{k=2}^{\lambda_3} \frac{N' - 1}{M - 1} (\frac{1}{M - 1})^{\lambda_3 - 2} \frac{M - 2}{M - 1} \geq \Phi_3 \qquad (5)$$

Take group 2 as an example. As shown in Fig. 6, given $\Phi_3 = 99\%$, $M = 40$ and $N' = 5$, then $\lambda_3 = 2$. That is, for pairwise inter-group sharing, 2 time slots are sufficient to achieve the overall success rate of $\Phi_3 = 99\%$.

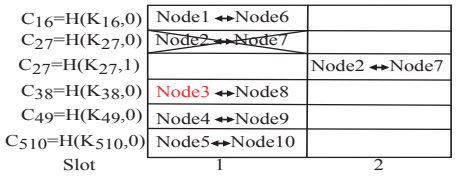


Fig. 6. Inter-group observation sharing for group G2

The inter-group sharing phase takes $T_3 = \lambda_3$ time slots.

3) Phase III: Producing A Ranked Suspect List: The purpose of this phase is to allow the nj-group to reach a consistent view on both groups. Specifically, it involves an intra-group broadcast sharing process in which nodes in each group share the list of observations previously obtained from the other group in phase II. This sharing is via a common channel determined by their group key.

There are three possible cases to consider. First, the jammer has prevented its group members from sharing their observations in phase I. With fewer than N' observations, its group members know they are in the j-group, so in phase II they do not share observations obtained from their counterparts in the other group. Consequently, by now the nodes in the other group already know they belong to the nj-group.

Second, the jammer fully follows the protocol, but provides different observations to its group members in phase I (by broadcast, its own group members will have received the same observations) and to its pairing node in phase II. In this case, nodes in either group will detect the inconsistency and take the majority result as the output. For example, node 3 in group G1, the jammer in our running example, may have shared its own observation O_3 with all the other members in phase I, but later shares a list of N' falsified observations with its pairing node 8 in phase II. Then, the nodes in group G2 will notice there are inconsistent views on group G1 by comparing the list reported by node 8 and those lists reported by nodes 6, 7, 9, and 10. In this case, every node in group G2 is required to take the majority view on group G1 and the report of node 8 is ignored. Hence, consistency is reached in the end of this phase. Note that in this case nodes cannot tell whether they belong to a j-group or not, because as in the example, one cannot tell which of node 8 and node 3 is lying.

Third, the jammer provides the same observation (true or false) to all its group members and also to its pairing node in the nj-group. In this case, neither group will know whether it is a j-group or nj-group at this moment.

In all these three cases, based on the shared observations, nodes will produce a ranked suspect list. For the first case, the nj-group will only use their N' local observations to produce the suspect list that only includes the nodes in the j-group. In the second and third cases, with the same view, both groups will use the merged N observations to produce the suspect list that includes all nodes. Because this phase is like the intra-group observation sharing phase, so it can also be finished within T_2. If we consider the time for calculating the suspect list negligible, the total time for process P3, including three phases, is $T_2 + T_3 + T_2 = 2T_2 + T_3$.

E. Process P4: Group Merging

In this process, we adopt the trial-and-error strategy to isolate the jammer and restore the normal communications among all benign nodes. Here we consider two cases based on the output of P3.

1) Case 1: The first case is that only the nodes in the nj-group have the consistent view, and they know the other group is the j-group. Let us assume the group G2 is the nj-group. If Node N_{j1} has the highest ranking in the suspect list, then nodes in G2 will derive a temporal group key $K_{T1} = F_{K_2}(N_{j1})$, where K_2 is their group key established in P2, F is a cryptographically strong pseudo-random function, and it is infeasible to invert F to compute K_2. The nodes in $G2$ will then pair up with the nodes in $G1$ excluding N_{j1} to distribute K_{T1} via their secret pairwise channels. In the end, all nodes except N_{j1} will obtain the same group key K_{T1}. After that, they will switch to a new channel $C_3 = H(K_{T1}, 0)$ to form a new group $G3$.

After the above merging process, if the new group G_3 is never jammed again by an insider, we will consider N_{j1} as the real jammer. Otherwise, if insider jamming happens again, the jammer is still in G_3 and hence node N_{j1} is indeed a benign node. Therefore, this time we will restore N_{j1} and consider the Node N_{j2}, the one ranked the second in the list as the jammer for removal. A process similar to the above will be executed. However, in the above case of false revocation, K_{T1} is known to the hidden jammer, so a new group key is needed. Instead of restarting the protocol from the process P2, in our design rekeying is easy. Basically, the nodes in $G2$ will derive a new key as $K_{T2} = F_{K_2}(N_{j2})$ and again share it with the nodes in $G1$ except N_{j2}. All the nodes except N_{j2} will form a new group $G4$ and communicate at a new channel $C_4 = H(K_{T2}, 0)$. This process can be repeated until the jammer is excluded.

2) Case 2: In the second case, all nodes in the network have a consistent view. This is a relatively more complex case because the j-group is not exposed yet, and our protocol can only rely on the observations to identify the suspicious nodes. Assume that the most suspicious node is N_{j1} and it belongs to $G1$. Then, similar to that in case 1, a temporal group key $K_{T1} = F_{K_2}(N_{j1})$ is distributed via pairwise channels and the group merges without N_{j1}. If N_{j1} is not the jammer and N_{j2} belonging to $G2$ has the second highest ranking, another temporal group key $K_{T2} = F_{K_1}(N_{j2})$ is distributed via pairwise channels and a similar process follows. This process stops when we isolate the real jammer from the network. The actual number of iterations needed in our protocol is determined by the accuracy of the underlying algorithm for identifying the jammer given the observations.

Fig. 7 shows an example for the second case, where node 7 has the highest ranking. In this example, nodes 1 to 5 separately transmit the key K_T to nodes 6, 8, 9, 10 at time slot 1 via pairwise channels. However, node 7 is not the real jammer and the newly informed group will experience insider jamming again in the future. If this happens, e.g., at time slot $n-1$, the group at time slot n will take node 7 back into the group but remove node 3, which has the second highest ranking. After that, the real jammer Node 3 is successfully removed, and the whole network could avoid insider jamming attacks. When the outcome of the suspect list is relatively accurate, the total time will be close to one step.

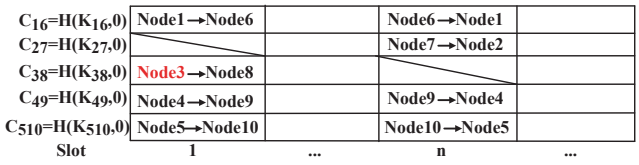


Fig. 7. Group Merging

As soon as a new group is formed among all nodes but the excluded one, all the member nodes are required to broadcast their presence. This will help address a potential attack in case 2. It happens when the jammer has not been identified yet, and the suspicious node to be excluded is a good node from the nj-group. The jammer may forward a forged temporal group key K'_T or forward no key to its pairing node. Consequently, its pairing node will fail to join the re-merged group. In this case, our protocol will simply consider either the missing node or its pairing node (i.e., the jammer in this case) as the actual jammer. Hence, the suspect list is now narrowed down to two nodes only. This indicates that not following the key distribution process could be a more disadvantageous strategy for the jammer.

The time complexity of process P4 is as follows. During temporal group key distribution via pairwise channels, random jamming may also succeed with some chance. Our countermeasure here is the same as that in inter-group observation sharing (phase II of process P3), so the time for reliable key distribution is within T_3. The above broadcast process is also similar to the intra-group observation sharing phase in process P3 except the number of broadcast nodes is changed from N' to $N-1$. We can obtain $T_4 = \lambda_4$ by replacing N' and λ_2 in Equation 4 with $N-1$ and λ_4, respectively. Depending on which case it is, the overall time in P4 is kT_3(Case 1)/$k(T_3+T_4)$ (Case 2) where k is the number of iterations in P4.

F. Performance Evaluation

From the above analysis, we know that P1, P2, P3 and P4 will take T_0, T_1, $2T_2+T_3$ and kT_3(Case 1)/$k(T_3+T_4)$(Case 2), respectively, where k is a constant denoting the number of iterations in P4. For message complexity, in P2, there are altogether $(N'-1)\lambda_1$ messages transmitted in each subgroup. In P3, $2\lambda_2$ messages at most are transmitted during intra-group sharing for each subgroup, and $2(N'+\lambda_3-1)$ messages at most are transmitted during inter-group communication. Thus, the total number of messages transmitted in P3 is $2(2\lambda_2 + N'+\lambda_3-1)$. In P4, the number of messages transmitted is $k(N'+\lambda_3-1)$(Case1)/$k[2(N'+\lambda_3-1)+\lambda_4]$(Case 2). We can add up these numbers to get the overall message complexity of our scheme in different cases.

IV. Security Analysis

In this section, we revisit several specific attacks and show how our protocol handles them. In the whole process of our protocol execution, we assume that the jammer can jam *at any time* and *at any channel of its choice* under two general constraints: (1) it takes some time to switch from one channel to another one; (2) it does not know the pairwise channels or broadcast channels used by benign nodes, so it performs

random jamming. As shown in Section III.B, in our design, by setting the time slot duration T_s appropriately, the jammer will only be able to switch into one channel to jam during T_s. In each process of our protocol, communication channels between benign nodes are determined by their pairwise key that is unknown to the jammer. As long as there are communications involved, the jamming probability has been taken into account in deciding the number of rounds or time slots for each process (e.g., P2, P3 and P4) so that it can be completed with a very high probability.

Besides jamming, the jammer may aim at disrupting the protocol by not following the operation at any process of its choice, or injecting any kind of false information (keys, observations) to prevent other nodes from communication or reaching a consistent view. Next we describe all possible attacks. Process P1 is deterministic and no communication is involved, so the attacker cannot interfere it. In process P2, if serving as a group leader or a relay, the jammer may interfere with the key distribution process by relaying false or no messages. This could cause key distribution to fail in the j-group. Our protocol tolerates this type of failure, because the nj-group can finish P2 successfully, and our protocol works the same way deterministically.

To attack P3, the jammer may fabricate false information (e.g., observations) and forward them to other nodes during intra-group sharing or to a benign node during inter-group sharing. Such false information is detected through broadcast sharing, and corrected by taking the majority view (in phase III of P3). To attack P4, the jammer may fabricate false temporal group key and distribute it to its pairing node when by mistake the node to be excluded is from the nj-group. As we discussed in case 2 of P4, this is a more disadvantageous situation for the jammer as it exposes itself quickly. Also, by using a cryptographically strong pseudo-random function F, our scheme can easily regenerate a new secret group key when false exclusions have realized.

As a summary, despite all these types of misbehavior by the jammer, our protocol executes in a deterministic way in each process. The time needed to accurately identify and exclude the jammer mainly depends on the accuracy of the underlying jammer identification algorithm given the observations.

V. Performance Evaluation

A. Experimental Environment

We evaluate our protocol with the USRP platform running GNURadio [15]. Each USRP is a wireless node and it uses RFX2400 daughter board and runs on Ubuntu 11.10. We implement a simplified version of our protocol for evaluation purpose with about 600 lines of Python code. The detailed parameters of USRP set in our experiments are shown in Table I. Note that our algorithm applies to any jammer identification algorithm that enables each node to output a ranked list of suspects. For the concreteness of our evaluation, we apply a jammer localization algorithm for jammer identification, assuming the locations of wireless nodes are known in advance and fixed. The algorithm takes received signal strength indicators (RSSIs) as the input, derives the distances between the jammer and the receivers (such distances are the observations to be shared in the process P3 of our protocol), and finally estimates the jammer's location based on distance information. The node whose location is closest to the estimated location is the most suspicious and hence considered as the jammer.

TABLE I. Technical Details of USRP

Parameter	Value
Frequency Range	2.3∼2.9 GHz
Transmit Power	50mW(17dBm)
Modulation	DBPSK
Samples per Symbol	6
Bits per Symbol	1
Bit rate	100kb/s

To estimate distances, our implementation applies a widely used signal propagation model—the Log-normal shadowing model [16]:

$$P_r[dBm] = P_0 - 10\alpha \log(\frac{d}{d_0}) + X_\sigma \quad (6)$$

In Equation 6, P_r is the receiving power at the receiver; P_0 is the receiving power at unit distance d_0; d is the distance from the transmitter to the receiver, which is to be estimated. α is the path loss exponent depending on the specific propagation environment. It varies from 1.2 to 8, and in free space α is 2 [17]. X_σ is a Gaussian noise with mean zero and standard deviation σ. Since all nodes have the same transmission power, we can measure P_0 at fixed d_0 by any one node, and regard P_0 and d_0 as known variables. X_σ and α can be also set as fixed values in the beginning stage of the network formation. Thus, a node can calculate its distance d to the transmitter as soon as it obtains P_r from the received signals.

Our experiment assumes a wireless network system is deployed in a 10.6×10.6 square feet classroom. We first select a random position for the jammer and then select other $N-1$ random positions for benign nodes. For our testing we use 3 USRPs. To scale up to a relatively large network (e.g., over 10 nodes), we assign every usrp a different role: a leader, a jammer, or a benign member. We reuse them for different roles in different phases. In the case of jammer localization, multiple benign nodes need to hear the jamming signals. We hence move the benign USRP to the location of each benign node one at a time while letting the jammer USRP keep jamming. Note that although this setting is not identical to multiple nodes receiving signals simultaneously, as long as the jamming signals are at the same strength, the statistical feature of RSSIs of jamming signals do not vary over time. We assume that the jamming occurs at time t_j and after that, all benign nodes start to collect RSSIs from the jamming signals. RSSI values are always maintained in the range of (-95,-10) by the wireless driver. In our experiment, we set $d_0 = 2$ meters and obtain $P_0 = -80.4889dbm$. Moreover, we read 100 consecutive RSSIs from each node in order to eliminate the Gaussian noise, and regard the average of these 100 RSSIs as the value of P_r (we will also test our scheme by only reading 1 or 10 RSSIs later). Then, nodes can compute their distances to the jammer d based on Eqn. 6. In our experiment, the classroom is a free space environment, so we set $\alpha = 2.0, 2.2$ and 1.8, in turn, to simulate some degree of variation. The jammer localization algorithm we deploy here is similar to a classic triangulation-based localization algorithm except it is able to remove an outlier. This is because the jammer may

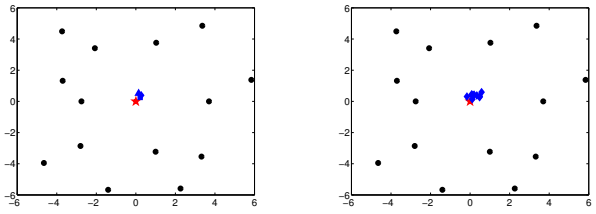
contribute false distance information to other nodes to avoid being localized. To deal with this issue, we apply the Grubb's test [18] to detect the outlier and discard it in order to improve the measurement accuracy.

B. Experimental Results

Our experiments include two cases with $M = 40$ available channels, corresponding to the two cases in process P4. In the first case, we set the network size $N = 6, 10, 15$ and $\alpha = 2.0, 2.2, 1.8$, in turn. According to our protocol, here the jammer localization algorithm only uses the observations (i.e., distances) from the *nj*-group. The results are shown in Fig. 8. It shows that the estimated locations of the jammer under different α values are all very close to its actual location. Also, when N increases, the identification accuracy increases as the estimated location gets closer to jammer's actual location. This is because there are more benign observations as input.

In the second case, jammer's false observation will be taken into calculation. We assume that the jammer reported false distance information $d_j \in [1 \sim 3], [6 \sim 8], [12 \sim 14]$, in turn, to avoid identification. Also, the Grubb's test is used to remove outliers. The impact of false distance input on identification accuracy is shown in Fig. 9. Here we set $N = 6$, simulating a small-size wireless network. When $N = 10$ or 15, our result shows that the identification accuracy improves further, because of increasing benign observations as input. From these figures, we can see that the localization errors in $d_j \in [1 \sim 3]$ and $d_j \in [12 \sim 14]$ are about the same, both smaller than in $d_j \in [6 \sim 8]$. This is because a very large false distance will be removed as an outlier by the localization algorithm.

In a real situation, the jammer may only transmit a few jamming packets, instead of 100 jamming packets. This might impact the identification accuracy, because the Gaussian noise cannot be eliminated for the lack of sufficient number of RSSIs. Hence, in the next experiment, we set $N = 15$ and $d_j \in [6 \sim 8]$, and let each node only collect 1 RSSI from the jamming signals. Fig. 10(a) shows the cases under different α values, and Fig. 10(b) shows 10 individual cases of estimated locations when $\alpha = 2.0$. Comparing Fig. 10 and Fig. 8(c), the algorithm can work effectively even though the jammer only transmits very few trashy signals.



(a) different α values (b) 10 cases at $\alpha = 2.0$

Fig. 10. Jammer identification accuracy at different cases. Circles represent the locations of benign nodes, and (red) star represents the actual jammer location. The estimated jammer locations are represented by blue shapes. Here $N = 15$ and the false distance provided by the jammer is 6∼8.

In the above experiments, the jammer has the highest ranking in the suspect node list. This leads to one-step identification and isolation in our protocol and the network is self-healed after that. If there was an error in ranking the suspicious nodes, the last process, P4, of our protocol would repeat whenever the jammer is detected jamming again.

VI. Related Work

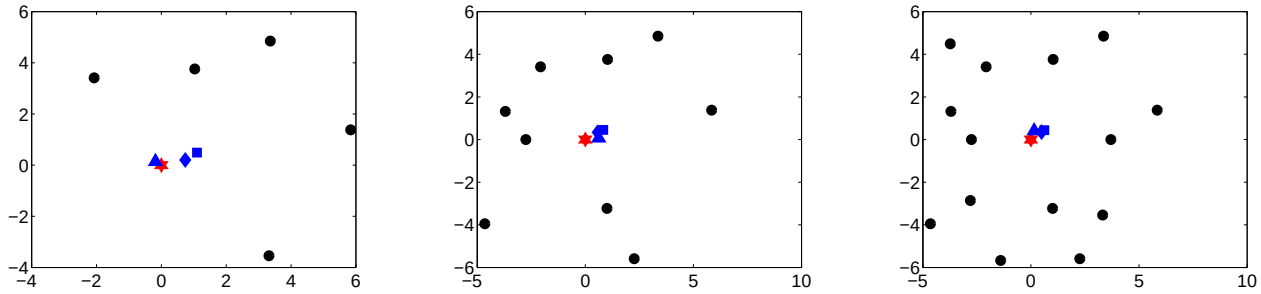
Jamming has been extensively studied in the past years. Due to space limit, we only highlight the most relevant ones.

Jamming Detection Jamming detection is very crucial the first step to defend against jamming attacks. Various methods [9], [10] have been proposed for four different jamming models (e.g., constant, deceptive, random, or reactive), based on packet-send ratio, packet-delivery ratio, signal strength, carrier-sensing time, message collisions, signal-to-noise ratio, and so on. In our work, we do not address the detection problem.

Communication under Jamming Attacks Spread-spectrum (SS) physical-layer techniques using direct-sequence or frequency-hopping spread spectrum are well-known countermeasures against jamming. These solutions require that both the sender and the receiver share the same key and the same pseudorandom function to generate a hopping or spreading sequence. However, they do not work under insider jamming attacks because the secret key is no longer secure. UFHSS [19] deals with the problem of key establishment without pre-shared secret under jamming. Liu and Ning [7] proposed BitTrickle to allow communication even in the presence of a broadband and high power reactive jammer. Liu et al. [8] proposed a time-delayed broadcast scheme(TDBS) against insider jammers by regarding broadcast as a series of unicast transmissions distributed in frequency and time. While these schemes are novel, they are relatively passive towards the jamming attack in the sense that normal network communication is no longer possible and network throughput becomes much lower.

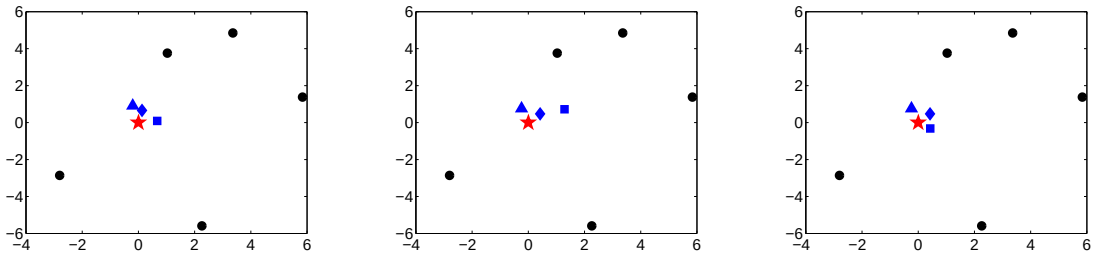
Besides, Chiang and Hu [4] proposed a novel cross-lay jamming mitigating mechanism for wireless broadcast networks based on the concept of code (key) tree. Dong and Liu [3] introduced a jamming-resistant broadcast system that organizes receivers into multiple channel-sharing broadcast groups and isolates malicious receivers using adaptive regrouping. In [5], a transmitter seeks to broadcast a message to multiple receivers such that colluding groups of compromised nodes cannot jam the reception of any other node.

While almost all works (including ours) assume a one-hop network setting, our work distinguishes itself from previous work [7], [8], [3], [5], [4] in a number of ways. First, the network and communication model is different. Previous work mostly assumes a single transmitter and multiple receivers, whereas in our work every node is a transmitter and a receiver. Second, for most efficient group communication, our work aims at ultimately providing a common secure channel or spread spectrum code for all benign nodes so that any *data* packet only needs to be broadcast once, whereas many existing works [4], [5], [8], [3] aim to tolerate the jamming attack or isolate the jammer by transmitting multiple copies of the same data packet in different channels. Third, our scheme is fully distributed, whereas previous schemes use the transmitter as the default network controller. Fourth, previous work does not consider the transmission reliability issue, whereas our protocol design explicitly takes message losses into consideration. Because of the above differences on network, communication and security models, there is no ground to directly compare our work with previous ones. Rather, they address the insider jamming issue in different settings.



(a) $N = 6$ (b) $N = 10$ (c) $N = 15$

Fig. 8. Jammer identification accuracy in wireless network of different sizes. Circles represent the locations of benign nodes, and (red) star represents the actual jammer location. The estimated jammer locations are represented by blue triangle ($\alpha = 1.8$), blue diamond ($\alpha = 2.0$), and blue square ($\alpha = 2.2$).



(a) False distance info. from the jammer (1∼3) (b) False distance info. from the jammer (6∼8) (c) False distance info. from the jammer (12∼14)

Fig. 9. Jammer identification accuracy when the jammer inputs false distance information. Circles represent the locations of benign nodes, and (red) star represents the actual jammer location. The estimated jammer locations are represented by blue triangle ($\alpha = 1.8$), blue diamond ($\alpha = 2.0$), blue square ($\alpha = 2.2$). Here $N = 6$.

VII. Conclusions and Future Work

In this paper, we have proposed a self-healing protocol to recover regular network communications among all benign nodes under insider jamming attacks. The protocol seamlessly integrates jammer identification, jammer revocation, and key management, by which all nodes except the jammer will eventually acquire a new key and establish new secret channels for their group communication.

This work is still a first step to solving a very challenge problem and there are a few directions for our future work. First, A more formal way of modeling the attack behavior is needed so that we can provide more theoretical security analysis. Second, it is desirable to extend the one-jammer one-hop model to handle the multi-jammer multi-hop case. Third, our scheme will also need to handle network dynamics caused by node join/leave. Fourth, implementing and evaluating the full protocol at a more realistic setting is also our future work.

Acknowledgement: The work of Sencun Zhu was supported in part by NSF grant CCF-1320605 and a Google gift. The work of Sushil Jajodia was performed while he was a Visiting Researcher at the US Army Research Laboratory. We also thank the reviewers for helpful comments.

References

[1] P. A. Richard, *Modern Communications Jamming Principles and Techniques (The Artech House Information Warfare Library)*. Artech House Publishers, 2003.

[2] D. Torrieri, *Principles of Spread-Spectrum Communication Systems, 2nd ed.* Springer, 2011.

[3] Q. Dong and D. Liu, "Adaptive jamming-resistant broadcast systems with partial channel sharing." in *ICDCS*, 2010.

[4] J. T. Chiang and Y.-C. Hu, "Cross-layer jamming detection and mitigation in wireless broadcast networks," *ACM Transaction on Networking (ToN)*, 2011.

[5] Y. D. et al., "Broadcast anti-jamming systems," *Computer Networks*, 2001.

[6] H. Nguyen, T. Pongthawornkamol, and K. Nahrstedt, "A novel approach to identify insider-based jamming attacks in multi-channel wireless networks," in *Milcom*, 2009.

[7] Y. Liu and P. Ning, "Bittrickle: Defending against broadband and high-power reactive jamming attacks." in *INFOCOM*, 2012, pp. 909–917.

[8] S. Liu, L. Lazos, and M. Krunz, "Thwarting inside jamming attacks on wireless broadcast communications," in *WiSec*, 2011.

[9] W. Xu, K. Ma, W. Trappe, and Y. Zhang, "Jamming sensor networks: attack and defense strategies," *IEEE Networks Special Issue on Sensor Networks*, vol. 20, 2006.

[10] A. D. Wood, J. A. Stankovic, and S. H. Son, "Jam: A jammed-area mapping service for sensor networks," in *RTSS*, 2003, pp. 286–297.

[11] S. Zhu, S. Setia, and S. Jajodia, "Leap+: Efficient security mechanisms for large-scale distributed sensor networks," *ACM Transaction on Sensor Networks (TOSN)*, 2006.

[12] X. Jiang, W. Hu, S. Zhu, and G. Cao, "Compromise-resilient anti-jamming for wireless sensor networks." in *ICICS*, 2010.

[13] V. Navda, A. Bohra, S. Ganguly, and D. Rubenstein, "Using channel hopping to increase 802.11 resilience to jamming attacks," in *INFOCOM*, 2007, pp. 2526–2530.

[14] D. Torrieri, "Direction finding of a compromised node in a spread-spectrum network," in *IEEE Milcom*, 2012.

[15] "Gnu radio," Tech. Rep. [Online]. Available: gnuradio.org

[16] T. S. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

[17] A. Neskovic, N. Neskovic, and G. Paunovic, "Modern approaches in modeling of mobile radio systems propagation environment," *IEEE Communications Surveys and Tutorials*, vol. 3, no. 3, pp. 2–12, 2000.

[18] "Grub's test for outliers," http://en.wikipedia.org/wiki/Grubb's_test_for_outliers.

[19] M. Strasser, C. Pöpper, and S. Čapkun, "Efficient uncoordinated fhss anti-jamming communication," in *MobiHoc*, 2009, pp. 207–218.