The Pennsylvania State University
The Graduate School

SELF-HEALING WIRELESS NETWORKS UNDER INSIDER JAMMING ATTACKS

A Thesis in
Computer Science and Engineering
by
Longquan Li

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

December 2013

The thesis of Longquan Li was reviewed and approved* by the following:

Sencun Zhu
Associate Professor of Computer Science and Engineering
Thesis Advisor

Wang-Chien Lee
Associate Professor of Computer Science and Engineering

Raj Acharya
Professor of Computer Science and Engineering
Head of the Department of Computer Science and Engineering

*Signatures are on file in the Graduate School.

Abstract

As jamming is a very serious threat to the normal operation of wireless networks, recently much research work has been done to deal with it. However, most of them only provide point solutions. 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, by identifying and revoking the jammer and restoring normal data communications among benign nodes, in the presence of jamming by an initially hidden insider node. We implemented and evaluated the protocol with USRP devices and GNURadio. Our evaluation study from experiments shows that our solution can identify and isolate the insider jammer with high performance.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my sincerest gratitude to my advisor Dr. Sencun Zhu for his consistent support and invaluable guidance. During the past two years, he indeed spent considerable time and effort in order to train me as a matured researcher in computer security. It is my great honor to work with him.

I would also like to thank Dr. Wang-Chien Lee for being on my advisory committee and for all his insightful comments.

I am also very thankful for Dr. Raj Acharya. He let me become his TA during my last semester at Penn State and we had a wonderful time together. Now, I become more responsible and more confident.

Finally, I would like to thank my friends and my family for all their love, support and sacrifice. I would never accomplish my work successfully without them. Thank you all.

Dedication

To my family.

Chapter 1

Introduction

1.1 Challenges of Jamming Attacks

These days wireless networks are deployed in all kinds of 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 or even completely paralyze a wireless network.

Jamming attacks mainly fall into two categories: outsider jamming and insider jamming. An outsider jammer randomly picks a channel to jam at one time. If there are M channels available in the network, 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 outsider 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 aspect 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, revoke it and restore its normal communication. We call this the **self-healing** capability.

1.2 Thesis Overview

In this work, we take a first step towards designing a systematic solution for self-healing wireless networks under insider jamming attacks. Unlike some existing work [3, 4, 5], which only 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, revokes 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. Our protocol seamlessly integrates the processes of jammer identification, jammer revocation and key management into one protocol. Our protocol is fully distributed without relying on any trusted central entity, and it is jamming resilient in the sense that the protocol tolerates jamming at any time and at any channel. We implemented our system with USRP devices and GNURadio and evaluate it through experiments. The results show that on average the actual jammer can be identified and revoked by running our system slightly over one time.

1.3 Related Work

Currently there are several directions to cope with jamming attacks.

Jamming Detection Jamming detection is very crucial as the first step to defend against jamming attacks. Various methods [6, 7, 8] have been proposed, based on packet-send ratio, packet-delivery ratio, signal strength, carrier-sensing time, message collisions, signal-to-noise ratio, and so on.

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 func-

tion to generate a hopping or spreading sequence. However, they do not work under insider jamming attacks because the secret key is no longer secure. Liu and Ning [9] proposed BitTrickle to allow communication even in the presence of a broadband and high power reactive jammer. Liu et al. [10] proposed a time-delayed broadcast scheme(TDBS) against inside jammers by regarding broadcast as a series of unicast transmissions distributed in frequency and time. While these schemes are useful, 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. Also, many existing pieces of work [3, 4, 5] only attempt to tolerate the jamming attack and they end up transmitting multiple copies of the same data packet in different channels. Our scheme is a more active solution compared to these existing work, by identifying and revoking the jammer and restoring the normal communication.

Jammer Localization Wireless node localization algorithms normally rely on three types of measurements: received signal strength indicator(RSSI), time of arrival(ToA) or time difference of arrival(TDoA), and angle of arrival (AoA). Liu et al. [11] developed two methods to tolerate malicious attacks against beacon-based location discovery. One is to filter out malicious beacon signals and the other one is to tolerate malicious beacon signals. A common limitation of these schemes is that they cannot be applied directly to localize the jammer, which is a hidden malicious node trying to disrupt the localization process by providing false input and jamming others' communication. Cheng et al. [12, 13] proposed several algorithms to localize one or multiple jammers by geometric methods (e.g., based on the shape of the jammed area) by assuming the jammer(s) are known. Our protocol is not for jammer localization, and it does not assume the knowledge of jammer's identity.

Jammer Identification and Isolation For jammer identification in multi-channel wireless network, Nguyen et al. [14] presented an "alibi" technique. Jiang et al. [15] proposed a compromise-resilient anti-jamming scheme called split-pairing to deal with insider jamming in a one-hop network setting. In their scheme, the group key for determining the network communication channel will be updated and shared only by all non-compromised nodes by assuming the identifier of the jammer is known. Dong and Liu [3] designed a jamming-resistant one-to-many broadcast

system with partial channel sharing to isolate malicious receivers and achieve the significant reduction of the communication overhead. Differently, our work does not assume jammer identification, and it applies to general communication models (instead of one-to-many model).

1.4 Thesis Outline

The rest of the paper is organized as follows. Chapter 2 introduces our wireless network and jamming models, as well as our design goal. Chapter 3 presents the details of our system design, and its security analysis is provided in Chapter 4. Chapter 5 simulates the network situation and provides a case study of how to produce a ranked suspect list to identify the jammer. Chapter 6 reports our implementation and the experimental results. Chapter 7 concludes the paper and discusses our future work.

Chapter 2

Preliminaries

In this chapter, we discuss our wireless network model and jamming model, as well as our design goal.

2.1 Wireless Network Model

First of all, like in most existing work [3, 14, 9, 10], we assume a one-hop network where nodes can hear each other. The nodes exchange data messages through a common channel, which is the most efficient way to share information among them. 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, for example, by receiving a threshold number of junk packets.

For security purpose, each pair of nodes shares a pairwise key (e.g., K_{xy} for Node 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 Node 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 [16, 17], which allow two nodes to establish a pairwise key on the fly as long as they know each other's id.

Although our protocol is presented in the context of a wireless network that communicates in a single channel unless switching is needed to avoid jamming, it can also be adapted to work in networks with either FHSS or DSSS in the

physical layer. Although in both cases insider jamming is easy to perform against the common channels, in FHSS or DSSS, outsider jamming (random jamming) has little chance to succeed against the communication between benign nodes which use private hopping/spreading sequences. As a result, our protocol design can be simplified by removing the redundant transmissions for reliable key distribution or observation sharing.

2.2 Jamming Model

Jamming attacks could be caused by an outsider jammer which does not know the network secret. The outsider jammer randomly scans all the channels and might be able to find the one currently used by the network nodes. 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 outsider jamming attack in this work but focus on insider jamming attacks.

There are two types of insider jammers. One is the active jammer and the other is the reactive jammer. An active jammer continuously emits trashy signals in the channel. A reactive jammer starts jamming the channel only after it senses signals (e.g., preambles of data frames), when it wants to purposely jam certain types of communication messages among other nodes, but keeps silent to hide its identity at most other times. Note that for a reactive jammer, it does not jam once and never jams again. Otherwise, its damage to the network is no more than the occasional signal collision or interference during the normal operation of a wireless network.

For ease of presentation, we assume that during one time slot 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 [15] and $7.6ms$ for Atheros WiFi chipset [18]. 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

$$T_l + T_c \leq T_s \leq 2T_l + T_{jc} \tag{2.1}$$

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. More specifically, the jammer can jam k channels in one time slot if $T_s \geq k(T_l + T_{jc})$.

2.3 Design Goal

Under the above network and jamming models, **our ultimate design goal is to empower a wireless network with self-heal capability under insider jamming attacks**. By self-healing, we mean the insider jammer should be identified and revoked so that the benign nodes can restore their communications. Unlike some existing work [3, 4, 5], which only attempt to tolerate the jamming and ends up transmitting multiple copies of the same data packet in different channels, our goal is to actively identify the insider jammer, revoke 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, our scheme should be efficient by incurring as little performance overhead as possible. Finally, 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 [19],

it assumes an array of antennas are available in the wireless nodes. 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 does 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 may 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. It can jam at any time and any channel of its choice, or disrupt the protocol by injecting false information as a legitimate participant of the network system. 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.

Chapter 3

Proposed Scheme

3.1 System Overview

Fig. 3.1 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 hopping to a new channel $C_i = H(K_0, i)$, where i is the i'th jamming detection. If it is an outsider jamming attack which happens by chance or through random scanning all channels, this type of frequency hopping will greatly thwart continuous jamming (especially in the case of FHSS or DSSS schemes). However, if more than a threshold number of jamming 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}. Through this trial-and-error process we can finally restore the network communication.

Fig. 3.2 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.

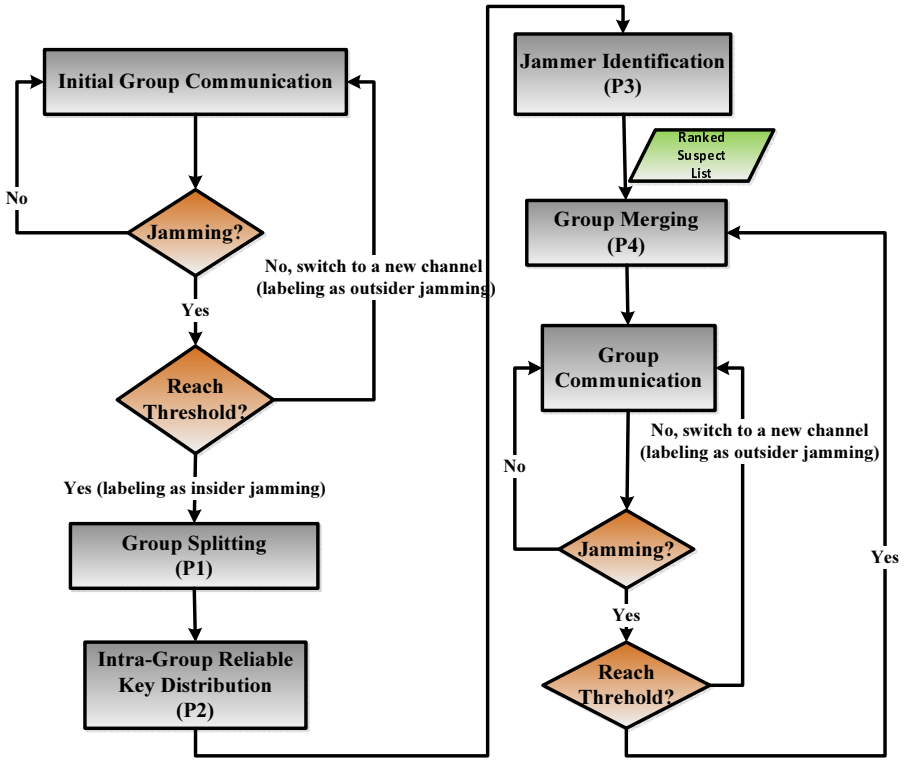


Figure 3.1. Flowchart of our system design

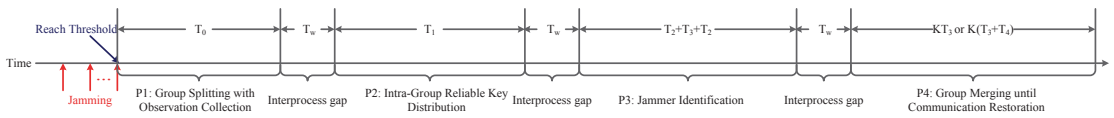


Figure 3.2. Timeline of our protocol

3.2 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 *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. 3.3. 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.

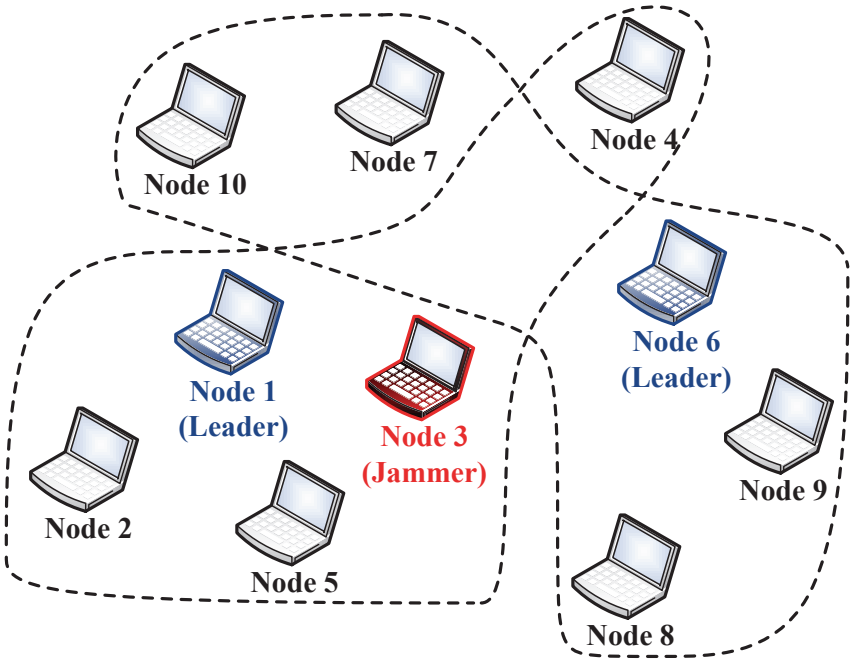


Figure 3.3. The group splitting process

3.3 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, is distributed through pairwise channels to avoid jamming. The process starts with the leader sending the encrypted 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. Normally it takes $\lceil \log(N/2) \rceil$ 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. Take group G2 as an example. In Fig. 3.4, a "X" marked rectangle indicates a certain channel has been jammed in a certain time slot, and a "\" marked rectangle 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, being jammed in its own scheduled receiving time slot earlier. 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).

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

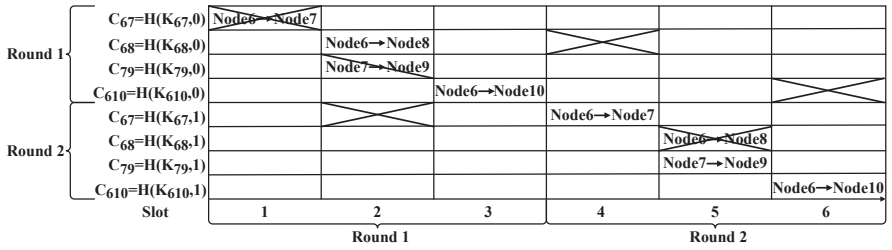


Figure 3.4. Intra-group reliable key distribution for group G2

a result, our protocol must survive under such attacks where the group with the jammer (referred to as the **j-group**) has totally messed up 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 unreliability 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 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 $N' = N/2$ be 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 (details in Appendix A).

$$\pi_0 = \frac{(\prod_{k=1}^{\lfloor \log N' \rfloor} (M - 1 - 2^{k-1}))(M - 1 - (N' - 2^{\lfloor \log N' \rfloor}))}{(M-1)^{\lfloor \log N' \rfloor + 1}} \tag{3.1}$$

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.

$$P_d(\lambda_1) = \sum_{k=1}^{\lambda_1} (1 - \pi_0)^{k-1} \pi_0 \tag{3.2}$$

$P_d(\lambda_1)$ 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. 3.4, 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. 3.2 such that $P_d(\lambda_1) \geq \Phi_1$.

Come back to our example shown in Fig. 3.4. 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. For the nj-group, it 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 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, when the jammer 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.

3.4 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 phase 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 received valid observations to produce a ranked suspect list.

3.4.1 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 (details in Appendix B), which is the minimal number of time slots required to achieve Φ_2:

$$\sum_{N'}^{\lambda_2} \binom{\lambda_2 - 1}{\lambda_2 - N'} (\frac{1}{M-1})^{\lambda_2 - N'} (\frac{M-2}{M-1})^{N'} \geq \Phi_2 \tag{3.3}$$

Take group 2 as an example, and let $M = 40$ and $N' = 5$. As shown in Fig. 3.5, 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. 3.3, $\lambda_2 = 6$. 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).

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

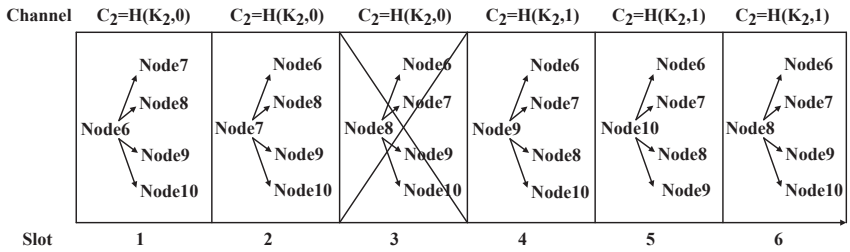


Figure 3.5. Intra-group observation sharing for group G2

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.

3.4.2 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 as 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 (details see Appendix C):

$$\frac{M-N'}{M-1}+\sum_{k=2}^{\lambda_3}\frac{N'-1}{M-1}(\frac{1}{M-1})^{\lambda_3-2}\frac{M-2}{M-1}\geq\Phi_3 \tag{3.4}$$

Take group 2 as an example. As shown in Fig. 3.6, 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 $\Phi_3 = 99\%$.

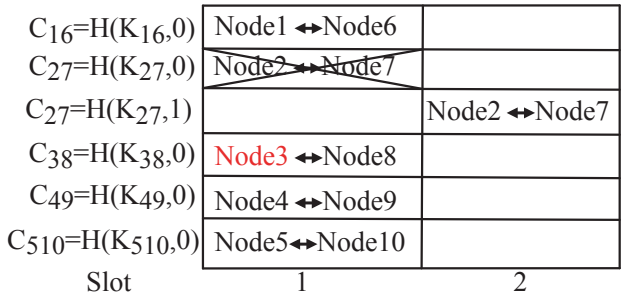


Figure 3.6. Inter-group observation sharing for group G2

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

3.4.3 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 in 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, the group members know they are in the j-group and they do not share observations obtained from their counterparts in the other group in phase II. By now the nodes in the other group already know they belong to the nj-group.

Second, the jammer fully follows the protocol except providing false observations. If the jammer provides different observations to its group members in phase I (by broadcast, its own group member will have received the same observations)

and to its pairing node in phase II, 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, the report of Node 8 is ignored and every node in group G2 is required to take the majority view on group G1. 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 like in the example, one cannot tell which of Node 8 and Node 3 is lying.

The third case is when the jammer provides the same false observation 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 covers 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 $T_2 + T_3 + T_2 = 2T_2 + T_3$.

3.5 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.

3.5.1 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 the 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 revoked.

3.5.2 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 ones. 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. 3.7 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 1 step.

$C_{16}=H(K_{16},0)$	Node1→Node6		Node6→Node1	
$C_{27}=H(K_{27},0)$			Node7→Node2	
$C_{38}=H(K_{38},0)$	Node3→Node8			
$C_{49}=H(K_{49},0)$	Node4→Node9		Node9→Node4	
$C_{510}=H(K_{510},0)$	Node5→Node10		Node10→Node5	
Slot	1	...	n	...

Figure 3.7. Group Merging

As soon as a new group is formed among all nodes but the revoked 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 revoked 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 to 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 Eqn 3.3 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.

3.6 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 total $(N'-1)\lambda_1$ 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.

Chapter 4

Security Analysis

In this chapter, we revisit several specific attacks and show how our protocol handles them.

4.1 Jamming Constraints

In the whole process of our protocol execution, we assume that the jammer can jam at any time at any channel of its choice under two constraints: (1) it takes some time to switch from one channel to another one; (2) it does not know the pairwise channels used by benign nodes; and (3) it does not know the broadcast channels used by benign nodes in another subgroup, so it performs random jamming. As shown in Chapter 2, 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, 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 complete with a very high probability (e.g., over 99%).

4.2 Dishonest Relay

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. In process P2, if serving as a group leader or a relay, the jammer may interfere with the key distribution by relaying false or no messages. This could cause the key distribution process to fail in the j-group. Our protocol tolerates this type of failure. As long as the nj-group can finish P2 successfully, our protocol works the same way.

4.3 Falsehood Disseminator

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 it happens that the node to be revoked 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.

4.4 Summary

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 revoke the jammer mainly depends on the accuracy of the underlying jammer identification algorithm given the observations.

Chapter 5

Simulation

In this part, we will simulate the network situation and provide a case study of how to produce a ranked suspect list to identify the jammer. Specifically, to acquire this list, we apply a classic triangulation-based localization algorithm with shared observations that are distances between the jammer and the receivers, assuming the locations of wireless nodes are known in advance and fixed. Note that our proposed scheme is very general, so other types of observations can also be used to identify suspicious behaviors. Without loss of generality, these distances are not set very accurately in order to simulate computation errors and external interference, and may also include a false distance which the jammer contributes in the simulation. Thus, we also apply the Grubb's test to remove outliers to improve the measurement accuracy. Last we evaluate its performance.

5.1 The Grubb's Test

The Grubb's test [20] is also called the extreme studentized deviate (ESD) test.

Definition 1 (ESD Test)**.** *Given a data set* $\Gamma = \{x_1, x_2, \cdots, x_n\}$*, The mean of* Γ *is denoted as* $\overline{x}$ *and the standard deviation of* Γ *is denoted as* s*. Let*

$$T_i = |x_i - \overline{x}|/s, where i = 1, \cdots, n.$$

T_i *is also called the corresponding T-value of* x_i*. Let* x_j *be the observation that leads to the largest* $|x_i - \overline{x}|/s$*, where* $i = 1, \cdots, n$*. Then* x_j *is an outlier when* T_j

exceeds a tabled critical value λ. In our work, we always regard x_j as the outlier no matter what λ is.

The real jammer may contribute false distance information to other nodes, aiming at hiding itself and misleading the calculation. To deal with this issue, we will apply the Grubb's test to detect the outlier and discard it in order to improve the measurement accuracy. Certainly, if the outlier is not what the jammer provides, this method still works well because ESD test always gets rid of the location with the highest deviation and in this case the location provided by the jammer is much closer to others than the outlier.

5.2 Simulation Setup and Results

We assume that all nodes in the network are deployed randomly at some certain (x, y) in the coordinate system, where both x and y are between $(-50, 50)$. We consider two cases. In the first case, only the nodes in the nj-group have the consistent view and they know the other group is the j-group. The number of nodes N in the whole network is the only one variable that we need to deal with and analyze. However, in the second case, all nodes in the network have a consistent view which means j-group is not exposed yet and our protocol can only rely on the observations to identify the suspicious ones. In addition to N, in this case we still need to consider another crucial variable d_j that is the distance information the jammer provides to other nodes. Without loss of generality, We will set up several different tests based on different cases with corresponding variables. The simulation results for the first case are shown in Fig. 5.1, where we presents the changing test accuracy with the change of N. $Rank = 1$ indicates the percentage that the most suspicious node we compute from our test is the real jammer; $Rank = 2$ indicates the percentage that the real jammer ranks second in the suspect rank list, produced from the test; $Rank > 2$ indicates the percentage that the rank of the jammer in the suspect rank list is beyond $Rank = 2$. The simulation results for the second case are shown in Fig. 5.2 and Fig. 5.3. In Fig. 5.2, we elaborate the changing test accuracy in three different particular cases($d_j \in [0, 20]$, $d_j \in [60, 80]$ and $d_j \in [120, 140]$) with the change of N. Moreover, in Fig. 5.3, we specify the changing test accuracy in another three different particular cases($N = 6$, $N = 10$

and $N = 15$) with the change of d_j. Note that the value v in "X" coordinate in Fig. 5.3 represent the range of d_j from $(v-10, v+10)$. For example, $v = 30$ means the distance the jammer provides varies from 20 to 40.

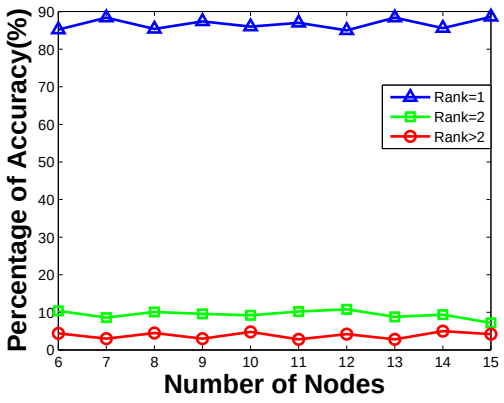
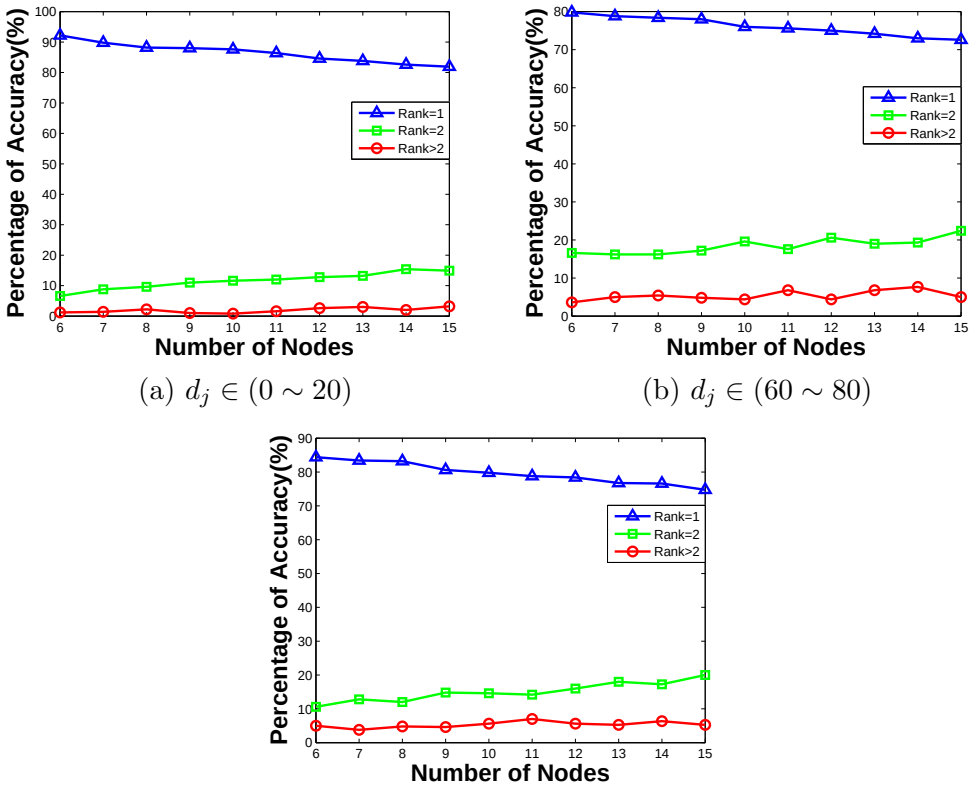


Figure 5.1. Jammer localization evaluation(accuracy vs number of nodes)

5.3 Simulation Evaluation

In the first case, we can see from Fig. 5.1 that the percentage of $Rank = 1$ varies with N. This situation arises from a trade-off. When N increases, the number of nodes in nj-group also increases; that is, benign nodes could get more distance information from each other, which can mitigate computation errors and external interference when they localize the jammer and produce the suspect rank list. Thus, the accuracy will increase correspondingly. However, when N increases, it is more likely that some node are located very close to the jammer because all positions of nodes are randomly deployed. In this case, because of computation errors and external interference, the most suspicious location calculated by nodes in nj-group is not always the position of the jammer and thus a benign node might be regarded as the most suspicious node, which leads to decline of accuracy.

In the second case, from these two figures, we can know that if d_j is fixed, the percentage of $Rank = 1$ will keep decreasing with the increasing N, but both the percentage of $Rank = 2$ and $Rank > 2$ will keep increasing. Also, if N is fixed, the percentage of $Rank = 1$ will decrease at first and then increase with the increasing d_j, but both the percentage of $Rank = 2$ and $Rank > 2$ will perform on the opposite. These two conclusions properly match the theory and the practice. For



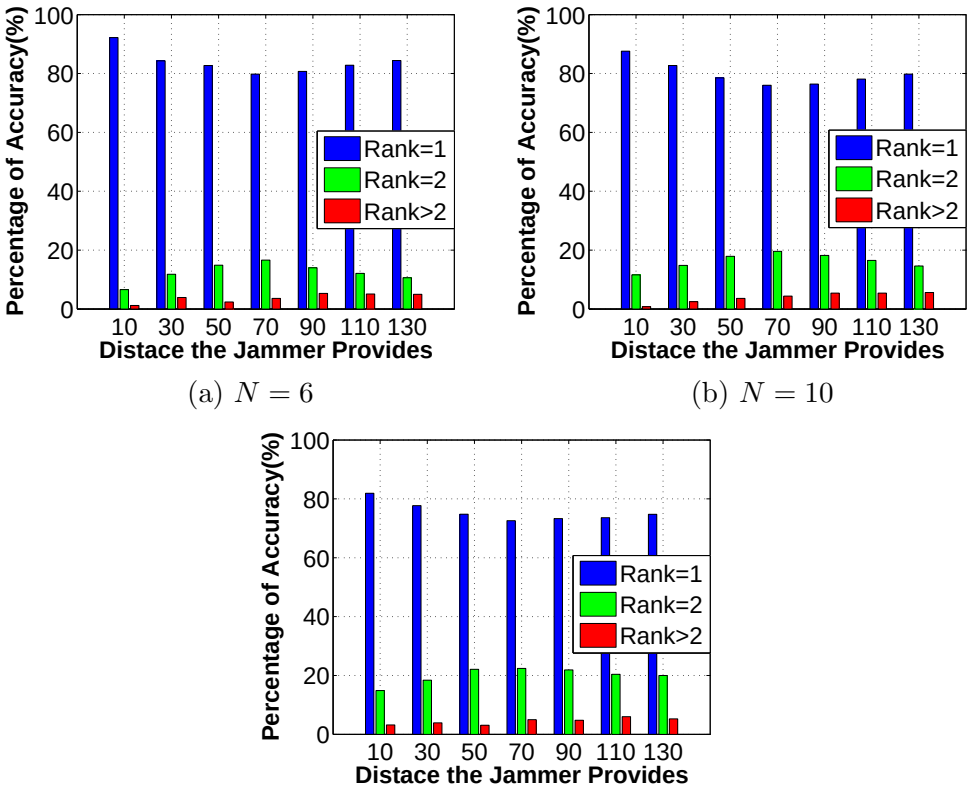
(a) $d_j \in (0 \sim 20)$ (b) $d_j \in (60 \sim 80)$

(c) $d_j \in (120 \sim 140)$

Figure 5.2. Jammer localization evaluation(accuracy vs number of nodes)

example, assuming N is fixed, if d_j is very small, it is easy to identify the jammer since the fake information can be only derived from the jammer and it is not sufficient to hide the jammer. If d_j is very large, it is more likely to be regarded as the outlier and then removed by the Grubb's test so it also cannot hide the jammer effectively. Therefore, if the jammer wants to hide itself more effectively, it has to provide a proper d_j, which is not too small or too large. However, even though in this extreme case, the percentage of $Rank = 1$ is still high(over 70% cases). On the other hand, assuming d_j is fixed, if N is larger, even though benign nodes can obtain more observations to identify the jammer, the jammer is more likely close enough to some benign nodes so that it could scapegoat these benign nodes.

If the jammer falls into $Rank = 1$, we can remove it and recover the regular network just through one step by using our protocol. Otherwise, we have to achieve our goal through some extra steps. No matter which case occurs, the ranked suspect list always provides a theoretical basis for the next steps of jammer



(a) $N = 6$

(b) $N = 10$

(c) $N = 15$

Figure 5.3. Jammer localization evaluation(accuracy vs distance the jammer provides) identification.

Chapter 6

Implementation and Performance Evaluation

In this chapter, we will explain our protocol implementation, experiment setup, and experimental results.

6.1 Implementation and Experimental Environment

We implemented a prototype of our protocol with the USRP platform running GNURadio [21]. Each USRP is a wireless node and it uses RFX2400 daughter board and runs on Ubuntu 11.10. The detailed parameters of USRP set in our experiments are shown in Table 6.1. For the concreteness of our evaluation, we also 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.

To estimate distances, we apply a widely used signal propagation model—the

Table 6.1. Technical Details of USRP

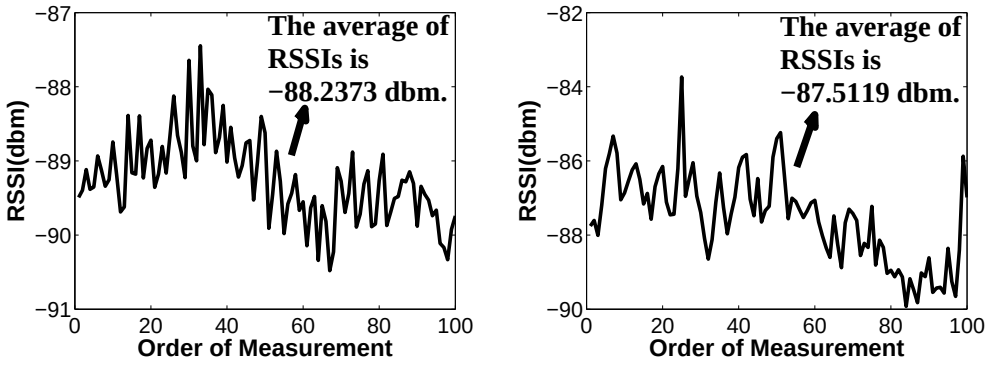
Parameter	Value
Frequency Range	2.3∼2.9 GHz
Transmit Power	50mW(17dBm)
Modulation	DBPSK
Samples per Symbol	6
Bits per Symbol	1
Bit rate	100kb/s

Log-normal shadowing model [22].

$$P_r[dBm] = P_0 - 10\alpha \log(\frac{d}{d_0}) + X_\sigma \tag{6.1}$$

In Eqn 6.1, 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 [23]. X_σ is a Gaussian noise (random variable) with zero mean 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.

In our experiment, a wireless network system is simulated in a 10.6×10.6 classroom. We first fix a random position for the jammer and then select other $N-1$ random positions for benign nodes. 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 assume that we can 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). Fig. 6.1 shows the actual RSSIs measured by two nodes that are 6 meters and 4 meters away from the jammer, respectively. 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



(a) Distance from the jammer is 6m (b) Distance from the jammer is 4m

Figure 6.1. Acquiring RSSIs

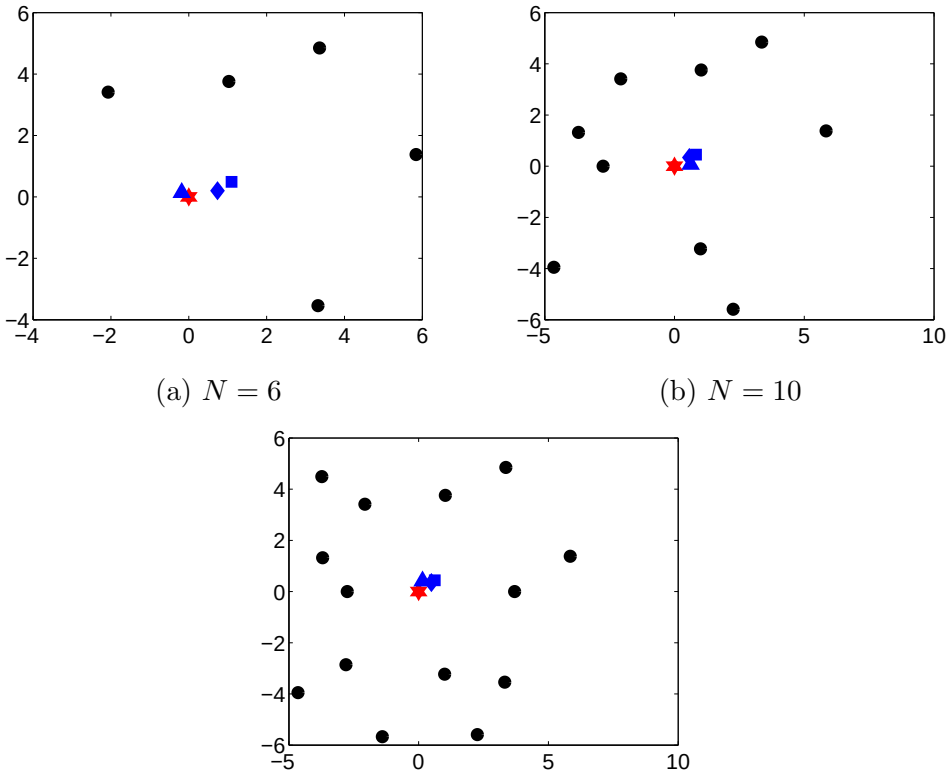
degree of variation.

The jammer localization algorithm which we deploy in experiments 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 shown in Chapter 5 to detect the outlier and discard it in order to improve the measurement accuracy.

6.2 Experimental Results

Our experiments include two cases within $M = 40$ available channels, corresponding to the two cases in process P4. In the first case, we let 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. 6.2. 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



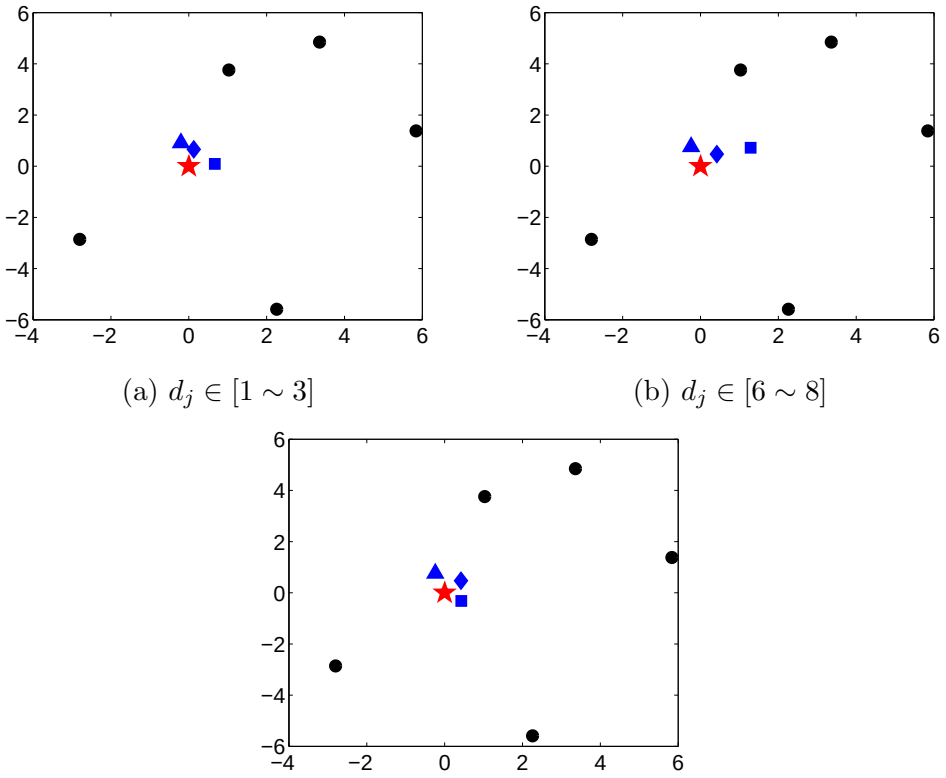
(a) $N = 6$ (b) $N = 10$

(c) $N = 15$

Figure 6.2. Jammer identification accuracy in wireless network of different sizes. Circles represent the locations of benign nodes, and (red) hexagons 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 ($\alpha = 2.2$).

is shown in Fig. 6.3, 6.4 and 6.5. Here we set $N = 6, 10$ and 15 respectively, simulating a small-size wireless network. From these figures, we can see that the localization error in $d_j \in [1 \sim 3]$ and $d_j \in [12 \sim 14]$ is about the same, both smaller than 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 would impact the identification accuracy, because the Gaussian noise cannot be eliminated for the lack of sufficient number of RSSIs. So, in the next experiment, we set $N = 15$ and $d_j \in [6 \sim 8]$, and let each node only collect 1 or 10 RSSIs from the jamming signals. The results are shown in Fig. 6.6. Comparing Fig. 6.6 and Fig. 6.5(b), we can conclude that the number of RSSIs does not influence the accuracy much. Thus, the algorithm can



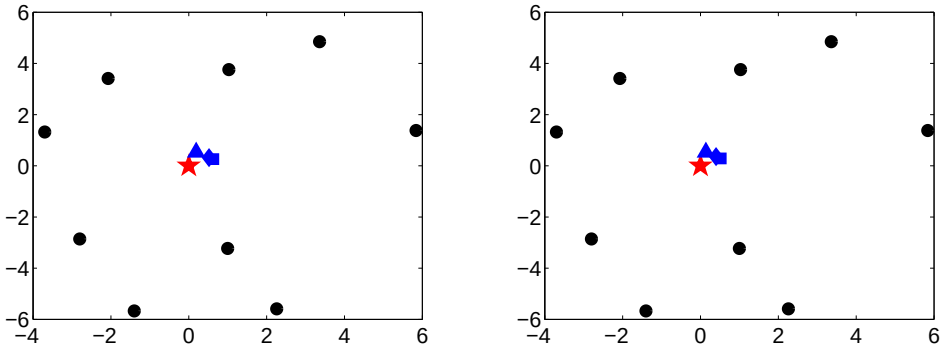
(a) $d_j \in [1 \sim 3]$ (b) $d_j \in [6 \sim 8]$

(c) $d_j \in [12 \sim 14]$

Figure 6.3. Jammer identification accuracy when the jammer inputs false distance information. Circles represent the locations of benign nodes, and (red) hexagons 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 $N = 6$.

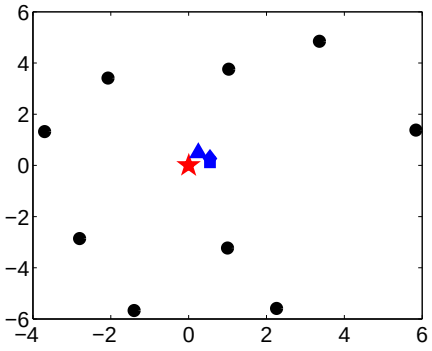
work effectively even though the jammer only transmits very few trashy signals.

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.



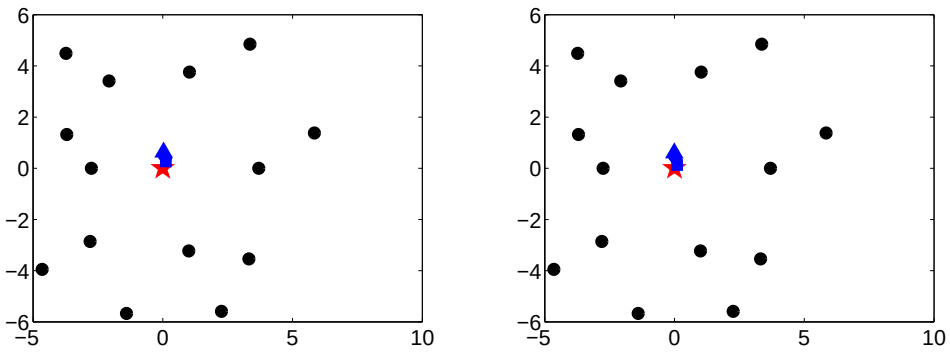
(a) $d_j \in [1 \sim 3]$

(b) $d_j \in [6 \sim 8]$

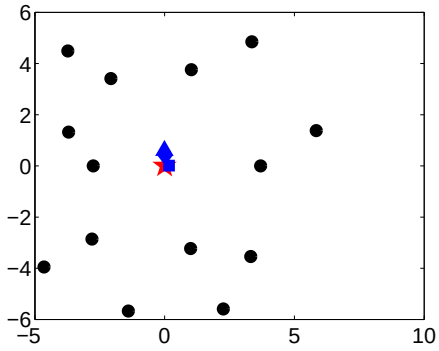


(c) $d_j \in [12 \sim 14]$

Figure 6.4. Jammer identification accuracy when the jammer inputs false distance information. Circles represent the locations of benign nodes, and (red) hexagons 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 $N = 10$.

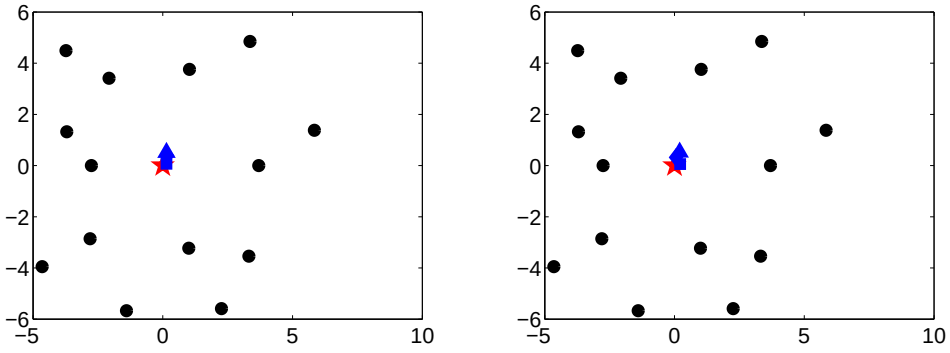


(a) $d_j \in [1 \sim 3]$ (b) $d_j \in [6 \sim 8]$



(c) $d_j \in [12 \sim 14]$

Figure 6.5. Jammer identification accuracy when the jammer inputs false distance information. Circles represent the locations of benign nodes, and (red) hexagons 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 $N = 15$.



(a) Each node only gets 1 RSSI (b) Each node only gets 10 RSSI

Figure 6.6. Jammer identification accuracy versus the number of jamming packets observed. Circles represent the locations of benign nodes, and (red) hexagons 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 $N = 15$ and the false distance provided by the jammer is 6$\sim$8.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

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. The implementation and experiments with USRP devices show that our scheme works with high performance.

7.2 Future Work

There are two directions for our future work. First, we will come up a more formal way of modeling the attack behavior so that we can provide more theoretical analysis of our scheme. Second, we will extend the one-jammer model to handle the multi-jammer case.

Appendix A

Proof of Eqn 3.1

We discuss two cases in the nj-group under the relay mechanism: (1) there exists a positive integer k such that $N' = 2^k$ and (2) such a k does not exist. In the first case, the number of time slots in one round is $\log N'$. Moreover, $1, 2, \ldots, 2^{\log N'-1}$ channels are used in $1st$, $2nd$, ..., $\log N'th$ time slot, respectively, during the relay-based key distribution. Given a time slot in one round, the jammer can choose to jam any of the $M-1$ channels excluding its own one. Thus, each channel in $1st$, $2nd$, ..., $\log N'th$ time slot will not be jammed with the probability of $\frac{M-1-1}{M-1}, \frac{M-1-2}{M-1}, \ldots, \frac{M-1-2^{\log N'-1}}{M-1}$, respectively. Then, we can get

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

In the second case, the number of time slots in one round is $\lfloor \log N' \rfloor + 1$. There are $1, 2, \ldots, 2^{\lfloor \log N' \rfloor - 1}, N' - 2^{\lfloor \log N' \rfloor}$ channels that are used in $1st$, $2nd$, ..., $\lfloor \log N' \rfloor th$, $\lfloor \log N' \rfloor + 1th$ time slot, respectively. Similarly, each channel in $1st$, $2nd$, ..., $\lfloor \log N' \rfloor th$, $\lfloor \log N' \rfloor + 1th$ time slot will not be jammed with the probability of $\frac{M-1-1}{M-1}, \frac{M-1-2}{M-1}, \ldots, \frac{M-1-2^{\lfloor \log N' \rfloor - 1}}{M-1}, \frac{M-1-(N'-2^{\lfloor \log N' \rfloor})}{M-1}$, respectively. Then, we can get

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

Combining the above two cases together, we can get

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

because in the first case, $\lfloor \log N' \rfloor = \log N'$, $2^{\lfloor \log N' \rfloor} = N'$ and

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

Appendix B

Proof of Eqn 3.3

To satisfy Φ_2, we need to derive a threshold λ_2 as the minimal number of time slots for broadcast sharing. λ_2 is at least N', because there are N' nodes that need to broadcast their observations. Thus, if $(\frac{M-2}{M-1})^{N'} \geq \Phi_2$, we can know $\lambda_2 = N'$. Otherwise, $\lambda_2 > N'$. In the latter case, there are $\lambda_2 - N'$ time slots jammed among $1st$, $2nd$, ..., $\lambda_2 - 1th$ time slots. Note that it is impossible that $\lambda_2 th$ time slot is jammed. Otherwise, the minimal number of time slots is not λ_2, but at least $\lambda_2 + 1$, which violates the definition of λ_2. Thus, the probability of the success for broadcast at $k(k > N')th$ time slot is

$$\binom{k-1}{k-N'}(\frac{1}{M-1})^{k-N'}(\frac{M-2}{M-1})^{N'-1}\frac{M-2}{M-1}$$

which can be written as

$$\binom{k-1}{k-N'}(\frac{1}{M-1})^{k-N'}(\frac{M-2}{M-1})^{N'}$$

Note that if $k = N'$,

$$\binom{k-1}{k-N'}(\frac{1}{M-1})^{k-N'}(\frac{M-2}{M-1})^{N'} = (\frac{M-2}{M-1})^{N'}$$

so this equation can be applied when $k \geq N'$. Therefore, to satisfy Φ_2, we can let $\lambda_2 = N', N'+1, \ldots$ sequentially until we find one to satisfy the equation:

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

Appendix C

Proof of Eqn 3.4

The probability that one channel is jammed in $1st$ time slot is $\frac{N'-1}{M-1}$ and thus the probability that each channel is not jammed in $1st$ time slot is $1 - \frac{N'-1}{M-1} = \frac{M-N'}{M-1}$. Because the jammer can at most jam one channel in $1st$ time slot, there is only one channel that will be used after $1st$ time slot. Thus, the probability of jamming the channel being used after $1st$ time slot is simply $\frac{1}{M-1}$. Then, the probability of the success for inter-group communication at $k(k > 1)$'th time slot is

$$\frac{N'-1}{M-1}(\frac{1}{M-1})^{k-2}\frac{M-2}{M-1}$$

Therefore, to satisfy Φ_3, we can let $\lambda_3 = 2, 3, \ldots$ sequentially until we find one to satisfy the equation:

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

Bibliography

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

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

[3] Dong, Q. and D. Liu (2010) "Adaptive Jamming-Resistant Broadcast Systems with Partial Channel Sharing," in *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*.

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

[5] Desmedt, Y., R. Safavi-Naini, H. Wang, L. Batten, C. Charnes, and J. Pieprzyk (2001) "Broadcast anti-jamming systems," *Computer Networks*.

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

[7] Çakiroğlu, M. and A. T. Özcerit (2008) "Jamming detection mechanisms for wireless sensor networks," in *Proceedings of the 3rd international conference on Scalable information systems*, InfoScale '08, pp. 4:1–4:8.

[8] Wood, A. D., J. A. Stankovic, and S. H. Son (2003) "JAM: A Jammed-Area Mapping Service for Sensor Networks," in *RTSS*, pp. 286–297.

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

[10] Liu, S., L. Lazos, and M. Krunz (2011) "Thwarting inside jamming attacks on wireless broadcast communications," in *Proceedings of the fourth ACM conference on Wireless network security*, WiSec'11.

[11] Liu, D., P. Ning, and W. K. Du (2005) "Attack-resistant location estimation in sensor networks," in *Proceedings of the 4th international symposium on Information processing in sensor networks*.

[12] Cheng, T., P. Li, and S. Zhu (2012) "An Algorithm for Jammer Localization in Wireless Sensor Networks." in *Proceedings of The IEEE International Conference on Advanced Information Networking and Applications (AINA)*.

[13] ——— (2011) "Multi-jammer Localization in Wireless Sensor Networks." in *CIS*, pp. 736–740.

[14] Nguyen, H., T. Pongthawornkamol, and K. Nahrstedt (2009) "A novel approach to identify insider-based jamming attacks in multi-channel wireless networks," in *Proceedings of the 28th IEEE conference on Military communications*.

[15] Jiang, X., W. Hu, S. Zhu, and G. Cao (2010) "Compromise-Resilient Anti-jamming for Wireless Sensor Networks." in *ICICS*.

[16] Liu, D., P. Ning, and R. Li (2003) "Establishing Pairwise Keys in Distributed Sensor Networks," in *Proceedings of ACM Conference on Computer and Communications Security (CCS)*.

[17] Zhu, S., S. Setia, and S. Jajodia (2006) "LEAP+: Efficient Security Mechanisms for Large-scale Distributed Sensor Networks," *ACM Transaction on Sensor Networks (TOSN)*.

[18] Navda, V., A. Bohra, S. Ganguly, and D. Rubenstein (2007) "Using Channel Hopping to Increase 802.11 Resilience to Jamming Attacks," in *INFOCOM*, pp. 2526–2530.

[19] Torrieri, D. (2012) "Direction Finding of a Compromised Node in a Spread-Spectrum Network," in *IEEE Milcom*.

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

[21] *GNU Radio*, *Tech. rep.*
URL `gnuradio.org`

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

[23] Neskovic, A., N. Neskovic, and G. Paunovic (2000) "Modern Approaches in Modeling of Mobile Radio Systems Propagation Environment," *IEEE Communications Surveys and Tutorials*, **3**(3), pp. 2–12.