COMBATING DOS ATTACKS IN WIRELESS NETWORKS USING LIGHTWEIGHT KEY MANAGEMENT SCHEMES

by

QI DONG

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2010

To my family and friends.

ACKNOWLEDGEMENTS

I will always remember the last four years I spent at our Information Security (iSec) Lab. I appreciate this precious experience, not only because of the challenging but interesting research work, but also because I have met some great people here.

I am very grateful to my supervising professor, Dr. Donggang Liu. I am impressed by his wisdom and his passion for students and research. He guided me to do research and solve the challenging problems. I very much appreciate being his student. I would like to thank the other professor of our iSec Lab, Dr. Matthew Wright. His knowledge and wit makes everyone in our lab, including me, enjoy working with each other and solving the security puzzles. I would also like to thank Dr. Gautam Das and Dr. Yonghe Liu for helpful discussions and insightful comments. All of those four professors have given me great inspiration and excellent guidance during both my course and research work.

I would like to thank all my friends at iSec Lab for their help in my PhD study (sorted by name): Titus Abraham, Apurv Dhadphale, Bikas Gurung, Jun-Won Ho, Safwan M. Khan, Kush Kothari, Pranav Krishnamoorthy, Brent Lagesse, Rongfang Li, Tara Mallesh, Kiran Mehta, Jaideep Padhye, Nabila Rahman, Kartik Siddhabathula, Gauri Vakde, Jin Xin and Dazhi Zhang.

Finally, I would like to give my special thanks to my family for their greatest love and caring. I am deeply indebted to my wife Lulu, our parents and my sister Ying. This dissertation is dedicated to them.

June 15, 2010

ABSTRACT

COMBATING DOS ATTACKS IN WIRELESS NETWORKS USING LIGHTWEIGHT KEY MANAGEMENT SCHEMES

Qi Dong, Ph.D.

The University of Texas at Arlington, 2010

Supervising Professor: Donggang Liu

The unique features of wireless networks lead to many attractive applications in both military and civilian operations. Wireless networking devices usually use batteries as power supply, which requires all the operations to be as efficient as possible. Since the resources such as the battery energy and wireless communication channels are critical in wireless networks, the attacker my try to disable the system operation by launching denial of service (DoS) attacks. Specifically, the attacker prevents the system from working by using up or blocking the limited resources.

This dissertation includes three studies on security mechanisms to combat DoS attacks using lightweight key management schemes. The first study introduces a novel pairwise key establishment technique. It can achieve both high resilience to node compromises and high efficiency by using a small number of additional sensor nodes. This work provides the basic cryptographic building blocks for other techniques proposed in this dissertation.

The second study presents two filtering techniques, a *group-based filter* and a *key chain-based filter*, to handle DoS attacks when digital signatures are used for

broadcast authentication in sensor networks. Both methods are efficient for resource-constrained sensor networks and can significantly reduce the number of unnecessary signature verifications that a sensor node has to perform.

The third study introduces two techniques for jamming-resistant broadcast systems, *partial channel sharing* and *unpredictable channel assignment*. Both schemes can significantly reduce the extra communication cost. The analytic and simulation results show that the proposed approaches greatly push the limit of jamming-resistant broadcast towards optimal.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In wireless networks, the nodes communicate with each other via wireless channels instead of the wired links. This significantly improves the system flexibility, because the nodes are free from the electrical wires or cables. The system deployment becomes easy and fast. Moreover, the nodes are not restricted to the fixed locations and can be mobile. Therefore, wireless networks have been widely deployed and continue to attract a great deal of commercial and research interest.

For instance, wireless sensor networks (WSN) have recently emerged as a premier research topic. A wireless sensor network typically consists of a large number of resource-constrained sensor nodes and possibly a few powerful control nodes (called *base stations*) [1]. These nodes usually communicate and collaborate with their neighbor nodes through low-power wireless links, and provide fine-grained sensing of physical conditions (e.g., temperature) from their immediate surroundings. The unique features of sensor networks lead to many attractive applications, such as target tracking and battlefield surveillance.

However, if the wireless networks are deployed in hostile environments, the attacker can launch malicious attacks to disable the network operation. This research focuses on the effective and efficient mechanisms to mitigate such attacks.

1.1 Motivation

Security has been recognized as a critical requirement for many wireless networking applications, especially in military operations. However, batteries are usually

used as the power supply for wireless networking devices, thus all the network operations should be as efficient as possible. Therefore, when we design the security schemes for wireless networks, we have to minimize the communication overhead, storage overhead, and the computing overhead.

Since resources are critical to wireless networks, a very serious attack is the denial of service (DoS) attack. Specifically, the attacker prevents the system from working by using up or blocking the limited valuable resources (e.g., the battery energy and wireless communication channels). A system is vulnerable to such DoS attacks, if the attacker can (1) consume many resources by cheating the system into performing very expensive operations or (2) easily access and block the system resources. Our ideas to protect wireless networks from such DoS attacks are (1) to avoid the unnecessary expensive operations as early as possible and (2) to make the critical resource accessible by legitimate users but unaccessible by the attacker.

To achieve those goals, we propose to use *efficient key management schemes* to manage the resources. For example, if each wireless communication channel is determined by a unique secret key, a user can easily access one particular channel only if he knows the correct key associated with that channel. On the other hand, key management itself is the cornerstone of most security protocols in wireless networks and needs to be well investigated.

In this dissertation, I study pairwise key establishment scheme, which enables secure one-to-one communication using cryptographic methods such as encryption and authentication. Based on the secure and reliable one-to-one communication, I also investigate how to combat DoS attacks in one-to-many (broadcast) scenarios. Specifically, this research provides solutions to handle two types of attacks, DoS attacks against broadcast authentication and jamming attacks against wireless broadcast systems.

1.1.1 Pairwise Key Establishment

Pairwise key establishment is one of the most fundamental security services. Many techniques have been developed recently to setup pairwise keys in sensor networks [2–11]. Perrig et al. developed the SNEP protocol to provide pairwise key establishment using a key distribution center (KDC) [2]. This approach, however, introduces huge communication overhead and is vulnerable to single point of failure. A number of key pre-distribution schemes were proposed to establish keys without any online KDC [3–7]. These approaches pre-load a small set of secrets into each sensor node before deployment to make sure that after deployment, every two sensor nodes can setup a shared key using their preloaded secrets. However, these approaches either require expensive protocols (e.g., path key establishment) to setup keys or are vulnerable to only a small number of compromised sensor nodes. In addition, some methods assume static sensor networks and the knowledge of sensors' location information [8–12]. These assumptions may not be true in practice.

Techniques to address these problems are provided in Chapter 3.

1.1.2 Providing DoS Resistance for Signature-Based Broadcast Authentication

Due to the sheer number of sensor nodes and the broadcast nature of wireless links, it is often desirable for a base station to broadcast commands and data to the network. The authenticity of such commands and data is critical for the correct operation of sensor networks. If convinced to accept forged or modified commands and data, sensor nodes may perform unnecessary and incorrect operations, and cannot fulfill the intended purposes of the network.

Recent studies have demonstrated that it is possible to perform public key cryptographic operations on resource-constrained sensor platforms [13–16]. However, the significant resource consumption imposed by public key cryptographic operations

makes such mechanisms easy targets of denial of service attacks. For example, an attacker can simply injects many forged broadcast messages and cheat the receiving nodes into performing a large number of unnecessary signature verifications, eventually exhausting their battery power.

Techniques to address these problems are provided in Chapter 4.

1.1.3 Jamming-Resistant Broadcast Systems

Signal jamming is a well-known threat against wireless communication systems. The attacker (or *jammer*) injects sufficient interfering radio signal to block the channels used by legitimate communication participants [17]. As a defense, *spread-spectrum* techniques such as *Frequency Hopping* (FH) and *Direct Sequence Spread Spectrum* (DSSS) spread the signal over a very large bandwidth, which virtually creates a huge number of *logical channels* determined by either the frequency-hopping pattern (FH) or the spreading code (DSSS) [17]. At any given time, only one or a small set of these channels are used for communication, and the selection of these channels is kept hidden from the attacker. This makes the attacker's job much more difficult since he has to jam a great number of channels to hit the channels in use.

Spread spectrum works well for pairwise communication in which only two communication participants are involved. However, in broadcast systems, the sender needs to send the same messages to many receivers. A common practice is to use a wireless channel that is known by all receivers. Unfortunately, if the attacker compromises a single receiver, he becomes an *insider* who knows which channel is currently in use and can block this channel easily. Thus, designing a jamming-resistant broadcast system is particularly challenging when there are insiders.

To combat *insider jammers* in broadcast systems, researchers have recently proposed several group-based approaches [18, 19]. The proactive approach in [18]

organizes receivers into multiple groups and assigns a secret channel to each group. The sender broadcasts a copy of the message on each channel. Each receiver belongs to several groups. Therefore, a receiver can receive broadcast messages as long as one of the groups is free of jamming. However, the number of groups increases significantly with the number of malicious receivers in order to tolerate jamming attacks. In other words, a small number of malicious nodes will cause great communication cost in the system.

Schemes in [19] use the "divide and conquer" strategy to isolate malicious receivers. However, these schemes require the sender to send a separate copy of each broadcast message to every group, causing a lot of communication overhead. The sender needs to send at least $2t$ extra copies of messages for each broadcast to deal with t compromised receivers.

Techniques to address these problems are provided in Chapter 5 and 6.

1.2 Summary of Contributions

The contributions of this dissertation are summarized below:

- *Pairwise key establishment using auxiliary sensors:* The first contribution is a novel technique for pairwise key establishment in sensor networks. The main idea is to deploy a small number of dedicated sensor nodes, called *assisting nodes*, to help key establishment between sensor nodes. Different from the regular nodes in traditional networks where they are used for many kinds of tasks such as sensing the physical environments and forwarding the messages, the assisting nodes are only responsible for key management in the network, representing a novel dimension of using sensor nodes.

  The proposed approach has many advantages over existing approaches. First, it can achieve a very high probability of establishing shared keys between sensor

nodes in the network. Second, a sensor node only needs to make a few local communications and perform a few hash operations to setup a key with any other sensor node. Third, the majority of sensor nodes only need to store a single key in their memory space. Fourth, the proposed approach does not depend on the sensors' location information and can be used for sensor networks with highly-mobile sensor nodes. Finally, our approach still provides high resilience to node compromise attacks. In addition, we also conducted analysis and simulation studies on different sensor deployment models and implemented this approach on TelosB motes [20]. The results demonstrate the benefits of our scheme.

- *Providing DoS resistance for signature-based broadcast authentication in sensor networks:* The second contribution is to apply *pre-authentication filters* to remove bogus messages before the actual signature verification is performed. Specifically, we develop two filtering techniques, a *group-based filter* and a *key chain-based filter*, to help sensor nodes avoid performing many unnecessary signature verifications. Both methods take advantage of the fact that broadcast in sensor networks is usually done through a network-wide flooding protocol, and a broadcast message from a sensor node usually has a small number of immediate receivers due to the low-power, short-range radio used in wireless sensor networks.

  The proposed pre-authentication filters provide complementary capabilities in dealing with DoS attacks against signature-based broadcast authentication. The group-based filter organizes the neighbor nodes of a (local) sender into multiple groups, which are protected by different keys organized in a tree structure. Using these group keys, this mechanism not only facilitates the neighbor nodes

to filter out forged messages, but also helps the sender adaptively isolate compromised nodes that launch DoS attacks.

The key chain-based filter employs a *two-layer* method, completely preventing compromised neighbor nodes from affecting benign ones. The first layer uses one-way key chains to mitigate the DoS attacks against signature verification, and the second layer uses pairwise keys to mitigate the DoS attacks on the verification of the chained keys in the first layer.

Our analytical results show that both group-based and key chain-based filters can efficiently and effectively thwart the DoS attacks on signature verification.

- *Adaptive jamming-resistant broadcast systems with partial channel sharing:* The contribution is two-fold. First, we propose a novel jamming-resistant broadcast system, which organizes receivers into multiple *channel-sharing broadcast groups* and isolates malicious receivers using adaptive re-grouping. Compared to existing approaches, this scheme reduces the communication cost from $2t$ to $(2-\rho)t$ additional messages, where t is the number of compromised receivers and ρ is the channel sharing factor ($0<\rho<1$). As a result, the proposed scheme is much closer to optimal considering the previously proven lower bound of t additional messages (in [19]) under realistic scenarios. Second, a *sequential test based scheme* is also proposed to further improve the performance of our jamming-resistant broadcast system so that the sharing factor ρ can be set larger to save more communication cost without reducing security. In addition, the analytic and simulation results show that the proposed approaches greatly push the performance limit of jamming-resistant broadcast systems towards optimal.
- *Mitigating jamming attacks with unpredictable channel assignment:* The fourth contribution is a further improvement of the solution in Chapter 5. It dynamically forms broadcasting groups in an *unpredictable* way such that insiders

cannot achieve maximal jamming impact. Basically, each receiver has a certain probability to share a common channel with other receivers. If a given channel is jammed and this channel is only assigned to one receiver, this receiver will be considered as one of the insiders. Thus, no matter when the insider chooses to jam the channel, there is a chance that he will be detected. Clearly, the more channels that the insider jams, the higher the probability that he will be detected. Compared with my previous solution, this technique only requires each receiver to listen to one channel, instead of multiple channels at the same time, thereby reducing the hardware cost greatly.

1.3 Organization of the Dissertation

The organization of this dissertation is as follows. The next chapter gives background information on pairwise key establishment, broadcast authentication, and jamming-resistant broadcast systems. Chapter 3 presents pairwise key establishment scheme using auxiliary sensor nodes. Chapter 4 discusses the pre-authentication filter techniques, which provides DoS resistance for signature-based broadcast authentication in sensor networks. Chapter 5 and 6 present the adaptive jamming-resistant broadcast systems with partial channel sharing technique and unpredictable channel assignment technique respectively. Chapter 7 discusses some future research directions on wireless network security.

CHAPTER 2

BACKGROUND

2.1 Pairwise Key Establishment in WSN

Pairwise key establishment has been studied extensively in the area of sensor network security. The goal is to *securely* setup a key for communication between two sensor nodes. A KDC-based scheme is proposed in [2]. In this scheme, every sensor node shares a unique key with the base station, which is then used as a trusted third party for establishing pairwise keys between sensor nodes. However, this method introduces significant communication cost since each key establishment requires the involvement of the base station. In addition, this method also suffers from single point of failure.

To overcome the problem of using a KDC, several key pre-distribution techniques have been developed such as [3–7]. In [3], a trusted third party generates a pool of random keys. Every sensor node then gets a certain number of keys from this pool before deployment. After deployment, each pair of sensor nodes has a certain probability of picking the same random key from the pool, which can then be used for their secure communication. In [4], a similar idea is used. The difference is that every pair of sensor nodes now has to find q common keys, which are picked from the same key pool before deployment, between them to establish a pairwise key. In addition, a *cooperative* protocol was developed to enhance the security of pairwise key establishment [21]. The giant component theory was used to further improve the performance and provide trade-offs between connectivity, storage cost, and security [22]. However,

in these schemes, a few compromised nodes will leak a large fraction of keys shared between sensor nodes.

In [5], a trusted party such as the base station generates a pool of *bivariate polynomials*, instead of random keys. Each polynomial $f(x, y)$ has the *symmetric property*: $f(x, y) = f(y, x)$. Before deployment, for each node i, we pick a subset of polynomials from the pool. For each polynomial $f(x, y)$ picked, we compute a *polynomial share* $f(i, y)$ and assign it to i. After deployment, if two sensor nodes i and j have the polynomial shares from the same bivariate polynomial $f(x, y)$, they can establish a pairwise key directly. Specifically, since node i knows $f(i, y)$, it can compute $f(i, j)$, which is the same as $f(j, i)$ due to the symmetric property. Since node j knows $f(j, y)$, it can compute $f(j, i)$ as well. Based on how polynomials are picked for each node, a *random subset assignment* scheme and a *grid-based* scheme were proposed. The former has each sensor node pick a fixed number of polynomials from the pool. In [6], a solution that is equivalent to this random subset assignment scheme was proposed independently. The later has every sensor node pick polynomials based on a logic *grid* so that every node can easily compute what nodes to contact to establish a pairwise key. In [7], this grid-based idea was also used for key establishment. These schemes can achieve much better security and performance than previous key pre-distribution schemes. However, it is still very expensive to establish keys through intermediate nodes.

In addition to the above schemes, many key pre-distribution techniques proposed to use prior deployment knowledge to improve the performance [8, 9, 11, 12]. The basic idea is to assign keying materials in such a way that two sensor nodes will have a very high probability of establishing a key if their expected locations are close to each other. In addition to the prior deployment knowledge, post deployment knowledge has also been used to improve key pre-distribution in sensor networks [10].

The idea is to assign each sensor node a large number of keying materials before deployment. After deployment, every node ranks keying materials based on its deployment location and only keeps those most useful ones in memory. However, the assumption of knowing the expected location or discovering the deployment location of each sensor node may not be practical.

2.2 Broadcast Authentication in WSN

Broadcast communication is an important service in wireless networks. For example, the base station can disseminate the commands efficiently and rapidly via broadcast communication. Obviously, the authenticity of broadcast message is critical in hostile environment. There are in general two types of solutions for broadcast authentication in sensor networks, *μTESLA* [2] and *digital signature* [13]. μTESLA and its variations achieve broadcast authentication through delayed disclosure of authentication keys. Its efficiency is based on the fact that only symmetric cryptographic operations are needed to authenticate a broadcast message. Despite the efficiency, μTESLA has some undesirable features such as the need for (loose) time synchronization and the authentication delay.

It has been demonstrated that the resource-constrained sensor is able to perform public key cryptographic operations [13]. In addition, there have been continuous efforts to optimize Elliptic Curve Cryptography (ECC) for sensor platforms [14–16]. We thus believe that ECC-based signature schemes will also be an attractive option for broadcast authentication in many applications. However, the significant resource consumption imposed by public key cryptographic operations makes such mechanisms easy targets of Denial-of-Service (DoS) attacks. For example, if ECDSA is used directly for broadcast authentication without further protection, an attacker can simply

broadcast forged packets and force the receiving nodes to perform a large number of unnecessary signature verifications, eventually exhausting their battery power.

Note that receiving packets also consumes the energy on sensor nodes. Hence, an adversary can also launch DoS attacks by sending many bogus packets to jam the channel and exhaust the energy on the victim nodes. However, this is significantly less efficient than the DoS attacks against signature verification. Suppose every packet has 102 bytes payload [23]. A MICAz mote will transmit 133 bytes in the physical layer [24], which will cost the receiving node about $133 \times 8/250,000 = 4.256$ms to receive. On the other hand, verifying a 40-byte ECDSA signature takes about 1.96 seconds [15]. As indicated in [25], an active MICAz CPU will cost about two fifth of the energy of receiving packets for the same period of time. This indicates that the DoS attacks against signature verification is about 184 times more efficient than the simple jamming attack.

The DoS attacks against signature-based broadcast authentication have also been studied in [26, 27]. A weak authentication mechanism using cryptographic puzzles is proposed in [26] to reduce the number of false signature verifications. However, it requires a powerful sender and introduces the sender-side delay. The dynamic window scheme proposed in [27] can control the propagation of fake messages by making smart choices between verifying a message before forwarding it and forwarding a message before verifying it. However, a fake message will be propagated in the network until the victim node verifies the signature.

2.3 Jamming-Resistant Broadcast Systems

In conventional wireless communication, the information bearing baseband signal is modulated onto a proper high-frequency carrier for transmission. The modulated signal occupies a region of radio spectrum centered at the carrier frequency. If

the attacker (or *jammer*) injects sufficient interfering signals into the same spectral region, he can significantly reduce the signal-to-noise ratio (SNR) at the receiver and thus interrupt the wireless communication [17].

Spread spectrum has long been an effective technique to mitigate jamming attacks. Examples include *Frequency Hopping* (FH) and *Direct Sequence Spread Spectrum* (DSSS) [17]. Their idea is to spread the signal over a much larger bandwidth to make it extremely expensive for an adversary to block the communication. In general, the more bandwidth, the better the resistance against jamming attacks. *Orthogonal Frequency Division Multiplexing* (OFDM) and *Software Defined Radio* were proposed for efficient spectrum management in cognitive radio networks [28]. Researchers have studied the combination of spread spectrum and OFDM to improve not only the efficiency and flexibility in spectrum usage [29] but also the resistance against the jamming attacks [30–33].

These jamming-resistant techniques can be modeled as a virtual multi-channel communication system, where the *virtual (or logical) channel* denotes the signal coordinates determined by either the spreading code (DSSS), the frequency-hopping pattern (FH), the sub-carriers (OFDM), or their combinations. In such system, the available spectrum contains a large number of orthogonal channels. Only a small subset of them will be used for communication. If the jammer does not know which subset is in use, he will be forced to either jam a large number of channels with negligible interference in each or only a few of them and leave many others, very likely including those actually used for transmission, interference free [17,34]. Forward error correction (FEC) schemes are also adopted to reduce the interference from jamming attacks and random noises [35–37]. They enhance the robustness of communication through redundancy.

These mechanisms work well for one-to-one communication. However, in broadcast communication, there are many receivers. Once the attacker compromises a receiver and finds out which channels are in use, he can directly block those channels without wasting any effort. The sender can certainly use jamming-resistant one-to-one communication to send a separate copy of each message to each receiver to combat the jamming. However, this introduces significant cost and delay, especially when a huge amount of data (e.g., multimedia data) needs to be disseminated or the data is time-sensitive (e.g., in battlefields or other emergency situations).

Group-based schemes have been proposed to combat *insider jammers* in broadcast systems [18,19]. The idea is to organize receivers into multiple broadcast groups and use different channels for different groups. This ensures that a compromised receiver can only affect the members in the same group. A "divide and conquer" strategy is then used to isolate malicious receivers. However, these schemes require the sender to send a separate copy of each broadcast message to every group, causing a lot of communication overhead. The sender needs to send at least $2t$ extra copies of messages for each broadcast to deal with t compromised receivers.

In addition to jamming-resistant broadcast, researchers have studied jamming-resistant schemes to establish a secret key between two nodes or control channels among the nodes [38–44]. These schemes usually involve a lot of communication overhead, long delay, or special hardware.

2.4 Other Attacks Against Wireless Sensor Networks

An attacker can launch a wide range of attacks against the network. For example, he can eavesdrop, modify, forge, replay, or block any network traffic. Especially, the attacker can *compromise a few nodes and learn all the secrets*, including the keying materials, on the compromised nodes [45]. Examples of other attacks that can

be launched against sensor networks include *sybil attacks* [46], *wormhole attacks* [47], and *node replication attacks* [48]. In sybil attacks, the adversary clones sensor nodes with different IDs to participate in network operations to mislead the application. In wormhole attacks, the attacker creates a wormhole between two sensor nodes that are far away from each other such that he can fool them into establishing an *incorrect* neighbor relation. In node replication attacks, the adversary compromises a few sensor nodes and then create many replicas of these compromised nodes in the field to impact the network at a large scale.

CHAPTER 3
USING AUXILIARY SENSORS FOR PAIRWISE KEY ESTABLISHMENT IN WIRELESS SENSOR NETWORKS

This chapter introduces a novel pairwise key establishment scheme that can achieve both high resilience to node compromises and high efficiency. The main idea is to deploy a small number of additional sensor nodes, called *assisting nodes*, to help key establishment between sensor nodes.

Each assisting node will hold a piece of secret knowledge for each sensor node. The special design endows the secret knowledge with several nice properties: (1) an assisting node have different secret knowledge for different sensor nodes; (2) different assisting nodes have different secret knowledge for the same sensor node; (3) a sensor node does not need to store its secret knowledge for any assisting node, but can easily compute its secret knowledge on any assisting node. Therefore, a sensor node can communicate with any assisting node easily and securely, protected by the corresponding secret knowledge.

Those assisting nodes will work as *local* key distribution centers and can randomly generate a unique *partial* secret key for any sensor node pair on demand. A sensor node pair may receive several different partial keys from different assisting nodes around them. The sensor node pair can compute their final pairwise key easily by XORing all those partial keys. As a result, the final key will be unknown to the attacker unless all the assisting nodes used in key establishment are compromised.

We further discuss the enhancements to handle the potential DoS attacks and the deployment failure of nodes. The analysis shows the efficiency of the proposed scheme. A sensor node only needs to make a few local communications and perform a

few efficient hash operations to setup a key with any other sensor node in the network at a very high probability. The majority of sensor nodes only need to store a single key. Besides, it also provides high resilience to node compromises. We also evaluate the performances of the proposed scheme under different sensor node deployment situations. The theoretical analysis, simulation studies and experiments on TelosB sensor motes demonstrate the advantages of this key establishment protocol in sensor networks.

3.1 System Model

In this dissertation, we use a *homogeneous network model*, in which the network includes a large number of *resource-poor sensor nodes* that cannot transmit messages over long distances or handle long periods of intensive communication and computation activity, and possibly a few resourceful *base stations* that have longer transmission ranges, more energy, and more storage. The resource-poor sensors can be either static or highly mobile; they are mainly used for sensing physical environments, conducting in-network processing, and forwarding messages. The base stations are mainly responsible for collecting/analyzing data from sensor nodes or connecting the sensor network to a traditional wired/wireless network. Due to the low-cost resource-poor sensor nodes, we assume that it is possible to deploy *extra* sensor nodes for some special purposes, e.g., assisting key management.

3.2 Pairwise Key Establishment

This section provides the technical detail of using auxiliary sensors for pairwise key establishment.

3.2.1 Baseline Approach

Sensor nodes are typically deployed to sense the physical conditions in their local surroundings and report observations for various uses. However, in this chapter, we explore a new dimension of using sensor nodes. Specifically, we believe that it is important to deploy sensor nodes that are dedicated to facilitate certain network protocols such as key management. Hence, the main idea of our approach is to deploy additional sensor nodes, *assisting nodes*, to help pairwise key establishment between sensor nodes. For convenience, we call the sensor nodes that are not assisting nodes as the *regular sensor nodes*. The detail of our protocol is presented below.

- *Initialization:* Before deployment, the base station generates a master key K_u for every sensor node u. The master key K_u is only known by the sensor node u and the base station. Every assisting node i will get preloaded with a hash $H(K_u||i)$ for every regular sensor node u, where $H(\cdot)$ is a one-way hash function [49], and "||" denotes the concatenation operation. Hence, an assisting node will need to store n hash images. This clearly introduces considerable storage overhead at assisting sensor nodes. However, we note that the only job of the assisting nodes is to help pairwise key establishment. As a result, they can use all their memory, including the flash memory, to store these hash values. Therefore, we believe that it will be feasible for an assisting node to store a reasonably large number of hash images. For instance, the TelosB mote has 1MB flash memory and can store hash images for a network of 128,000 sensor nodes if every hash image is 8 bytes long. Furthermore, research focusing on high-capacity, energy-efficient storage subsystems on sensor platforms is quite active in recent years. This will soon make it possible to equip sensor nodes with a few gigabytes of flash memory [50]. Therefore, more or longer hash images can

be stored in each assisting sensor node to either support larger sensor networks or achieve higher security.

- *Pairwise Key Establishment:* After deployment, every regular sensor node will first discover all neighboring assisting nodes in its radio range. When a sensor node u needs to establish a pairwise key with another node v, it will send a request to its neighboring assisting node i. The request message includes a random nonce, the ID of the other node v, and will be protected by $H(K_u||i)$, which can be directly computed by u. When i receives the message, it can immediately verify it since the key $H(K_u||i)$ was preloaded to it during the initialization. The assisting node i will serve as the KDC and generate a reply to u, which includes the key $H(H(K_v||i)||u)$. Note that this key can be directly computed by node v given the IDs i and u. In the end, both nodes know $H(H(K_v||i)||u)$ and can use this key to secure their communication. The protocol for key establishment is:

$$
\begin{aligned}
&(1) \quad u \rightarrow i \quad : v, r_1, H(v||r_1||H(K_u||i)) \\
&(2) \quad i \rightarrow u \quad : E_{H(K_u||i)}(v, r_1, H(H(K_v||i)||u)) \\
&(3) \quad u \rightarrow v \quad : i, E_{H(H(K_v||i)||u)}(r_2) \\
&(4) \quad v \rightarrow u \quad : E_{H(H(K_v||i)||u)}(r_2 - 1)
\end{aligned}
$$

  In the above protocol, r_1 and r_2 are two random nonces, and $E_K(X)$ denotes the encryption of message X using key K.

3.2.2 Using Multiple Assisting Nodes

In our baseline approach, only one assisting node is used to establish a secret key between two sensor nodes. Clearly, if this assisting node is compromised, the communication between these two nodes will be insecure. To enhance the security of shared keys between sensor nodes, we propose to use multiple assisting nodes in

establishing a pairwise key. Specifically, two sensor nodes will apply the key establishment of the baseline approach to establish a key using each of the neighboring assisting nodes and use these keys in computing the final key.

Key establishment with multiple assisting nodes: Suppose nodes u and v try to establish a key. The protocol involves the computation of two partial keys, one by node u and the other by node v, and the combination of these two partial keys. The details are given below:

1. *Partial key at node u:* Let $\{u_1, u_2, ..., u_n\}$ be the set of u's neighboring assisting nodes. u contacts each of them (using steps 1 and 2 of the baseline protocol) and receives n keys $\{H(H(K_v||u_1)||u), H(H(K_v||u_2)||u), ..., H(H(K_v||u_n)||u)\}$. The partial key k'_u at u is computed by XORing all these n keys. This partial key can be directly computed by node v given the ID u and the set $\{u_1, u_2, ..., u_n\}$.
2. *Partial key at node v:* Let $\{v_1, v_2, ..., v_m\}$ be the set of v's neighboring assisting nodes. Similarly, node v also contacts each of them (using steps 1 and 2 of the baseline protocol) and receives m keys $\{H(H(K_u||v_1)||v), H(H(K_u||v_2)||v)$, ..., $H(H(K_u||v_m)||v)\}$. The partial key k'_v at v is computed by XORing all these m keys. This partial key can be directly computed by node u given the ID v and the set $\{v_1, v_2, ..., v_m\}$.
3. *Key combination:* In this step, u sends the following message to node v: $\{r_3, S_u, H(r_3||S_u||k'_u)\}$, where r_3 is a random nonce and $S_u = \{u_1, u_2, ..., u_n\}$. When v receives this message, it can directly compute all keys used to compute k'_u given S_u. It can then authenticate the message using k'_u. If authentication succeeds, the final key $k_{u,v}$ will be computed by XORing k'_u and k'_v. Finally, node v will send the following message to node u: $\{r_4, S_v, H(r_4||S_v||k_{u,v})\}$, where r_4 is a random nonce and $S_v = \{v_1, v_2, ..., v_m\}$. When u receives this message, it can derive all keys used to compute k'_v given S_v. It then computes the final

key $k_{u,v}$ and authenticates the message using $k_{u,v}$. If authentication succeeds, it knows that the final key $k_{u,v}$ is now shared with v.

From the above protocol, we can see that the final key is computed by using all keys provided by neighboring assisting nodes. As a result, the final key will be unknown to the adversary unless all neighboring assisting nodes are compromised. In other words, the adversary cannot compromise the confidentiality and integrity of the communication between two nodes unless all assisting nodes used in key establishment are compromised. To defeat our scheme, one possible attack is to jam the communication around the benign assisting nodes such that only malicious assisting nodes are used in computing common keys. Since there are no effective ways to stop an attacker from jamming the communication, we simply ask every sensor node to re-discover new assisting nodes from time to time. This actually forces an adversary to continuously launch jamming attacks since otherwise benign assisting nodes will eventually be used to establish *secure* keys (unknown to the adversary) between sensor nodes. As a result, in this chapter, we are more interested in the performance of our protocol when the adversary stops blocking the communication to and from benign assisting nodes.

Mitigating DoS attacks: Although the confidentiality and integrity of pairwise keys can be ensured as long as at least one benign assisting node is used in our key establishment, the adversary can launch DoS attacks against the protocol, aiming at preventing two nodes from establishing a common key. In particular, the adversary can compromise a few assisting nodes and then have them give incorrect keys to any requesting node. In this way, two sensor nodes that try to establish a pairwise key will have *inconsistent views* of the keys used to compute one or both of their partial keys. Thus, the final key cannot be established. This DoS attack is cost effective for the attacker since one incorrect key can stop the key establishment in the future.

To address this problem, we need to identify consistent keys used in computing a partial key. This is done by simply using a *challenge and response* protocol to test the knowledge of keys between two sensor nodes. Each key $H(H(K_v||i)||u)$ provided by an assisting node i to a node u has two states: *verified* or *unverified*. If the other node v proves its knowledge about this key, u will mark it as *verified*; otherwise, the key will be marked as *unverified*.

Specifically, if a node u discovered that its partial key k'_u is computed differently from the other node v, it marks all keys from assisting nodes as *unverified*. Then for each key provided by its neighboring assisting nodes, it challenges v about the knowledge of this key. This protocol will be the same as the steps (3) and (4) of the baseline protocol. A key that passed the test will be marked as *verified*. We have to mention that an incorrect key provided by a malicious node may be marked as *verified* since the adversary can forge a correct response to u regarding this key. If the adversary keeps doing this, node u will not be able to establish a common key with node v. The result as well as the cost is equivalent to the case where the messages in the key establishment protocol keeps being corrupted. We do not have a perfect solution to address this DoS attack. However, our objective is to ensure that we can establish a common key when the adversary stops manipulating messages.

Once all keys are tested, u will use all verified keys to derive the partial key k'_u. To reduce the cost, the messages for testing different keys can be piggybacked together. Note that when the adversary is still actively manipulating messages, it is possible that k'_u is still different from what node v knows. In this case, we simply restart the challenge and response protocol after a certain period of time. Once the attacker stops manipulating the messages between u and v, the partial key can be correctly computed.

3.2.3 Supplemental Key Establishment

Our later analysis in section 3.3 shows that even a small number of assisting nodes can guarantee a very high probability of establishing pairwise keys between sensor nodes. However, in practice, it is still possible that a regular sensor node is not able to find any assisting sensor node in its neighborhood since the deployment of assisting nodes may not always guarantee full coverage. To deal with this issue, we propose *supplemental key establishment*, in which a regular sensor node will look for those assisting sensor nodes that are multiple hops away from itself. This will certainly increase the chance of finding assisting nodes to use. Since more assisting nodes will be found, more random keys will be used in generating the final pairwise key. As a result, we are also able to achieve better security performance.

- *Supplemental Key Establishment:* In this step, a sensor node u will look for the assisting sensor nodes that are no more than h hops away from itself. This can be easily done by flooding a "discovery" message to all sensor nodes that are no more than $h-1$ hops away from u. The nodes receiving such discovery message will forward their lists of neighboring assisting nodes to node u. The lists could be aggregated during the transmission to reduce the communication cost. Once such set is discovered, the remaining step will be similar to the pairwise key establishment discussed before.

The discovery and usage of assisting nodes multiple hops away will certainly introduce additional communication overhead. However, we can see that it only involves the communication with the nodes no more than h hops away. Furthermore, since the probability of finding at least one assisting node in a node's radio range can be made very high, the supplemental key establishment will only happen at a very low probability. In other words, the additional communication will rarely occur. As

Table 3.1. Notations

n	number of the regular nodes
m	number of the assisting nodes
L	length of deployment field
r	communication radius of sensor node
f_c	fraction of compromised sensor nodes
P_{en}	probability of establishing a key between two one-hop neighbors
P_{ea}	probability of establishing a key between any two nodes
P_{cn}	probability of a key between two one-hop neighbors being compromised
P_{ca}	probability of a key between any two nodes being compromised

a result, we believe that such additional overhead will not be a big problem for the current generation of sensor networks.

3.3 Evaluation

This section presents the security and performance analysis for the proposed scheme. We will focus on metrics such as the probability of establishing pairwise keys, the resilience against node compromises, and the overhead. We will also compare our approach with previous approaches in terms of these metrics. For convenience, we list the notations frequently used in analysis in Table 6.1. Note that different sensor deployment models will likely impact the security and performance of key management protocols. As a result, we will also consider the following sensor deployment models during the analysis: (1) *uniform distribution model*, (2) *Gaussian distribution model*, (3) *group deployment model with fixed group centers*, and (4) *group deployment model with random group centers*. We will first study the security and performance of our approach when the uniform distribution model is used in sensor deployment. We will then extend such analysis to other deployment models in Section 3.3.8.

3.3.1 Sensor Deployment Models

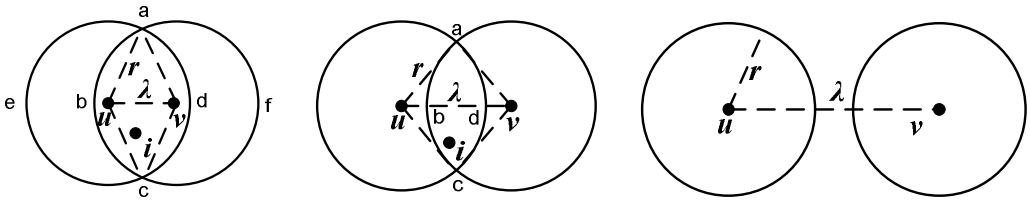
We assume that the network consists of n regular sensor nodes and m assisting nodes. These nodes are deployed in a square-shape field of size $L \times L$. Each sensor node i will be deployed towards an expected location (x_i, y_i), which may be *known or unknown a prior* depending on the actual deployment. It is also possible that the expected locations of sensor nodes follow certain probability distribution functions. For example, if every sensor node is randomly deployed in the field, the expected location is a random distribution over the field.

After deployment, the actual location of a sensor node may be deviate from its expected location due to the deployment errors that can be hardly removed. In other words, after deployment, the sensor node may locate at any point in the field with a certain probability. We hence use the probability density function (pdf) $f_i(x, y)$ to model the distribution of the actual deployment location (X, Y) of every sensor node i. Thus, the probability that this sensor node i appears in a particular area $\mathcal{D}$ can be simply calculated by integrating $f_i(x, y)$ over the area, i.e., $\int\int_{\mathcal{D}} f_i(x, y)dxdy$.

In our analysis, we will first consider the following simple sensor deployment model, in which the actual location of a sensor node i follows a uniform distribution:

$$f_i(x, y) = \begin{cases} \frac{1}{L^2}, & \text{if } 0 \leq x \leq L \text{ and } 0 \leq y \leq L; \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

This deployment model has been used for analyzing many existing key pre-distribution techniques [3–7]. As a result, our analysis will start from this simple model. However, we have also seen some other deployment strategies used in several studies, e.g., the group-based deployment model. Therefore, we will also evaluate the



(a) (b) (c)

Figure 3.1. Three types relationships between two sensor nodes u and v. (a) $0 \leq \lambda \leq r$; (b) $r < \lambda \leq 2r$; (c) $\lambda > 2r$.

security and performance of our approach under other deployment models in Section 3.3.8.

We assume that all sensor nodes have the same signal transmitting and receiving range, which can be modeled as a circle of radius r. Let λ denote the distance between two sensor nodes u and v. As illustrated in Figure 3.1, these two sensor nodes may exhibit one of the following three types of relationship based on r and λ:

(1) In Figure 3.1(a), we have $\lambda \leq r$. This means that the two sensor nodes are located within the signal range of each other. They are able to communicate with each other directly. We thus call them *one-hop neighbors*.

(2) In Figure 3.1(b), we have $r < \lambda \leq 2r$. This means that the two sensor nodes are out of the signal range of each other and cannot talk to each other directly. However, it is possible that they share some common neighbors, e.g., node i in the figure. In this case, they are able to talk to each other *indirectly* via their common one-hop neighbor i. We call u and v *two-hop neighbors*.

(3) In Figure 3.1(c), we have $\lambda > 2r$. This means that the two sensor nodes have to use more than one intermediate sensor nodes to communicate with each other. We simply call u and v *remote node pair*.

3.3.2 Probability of Establishing Keys

In our key establishment protocol, two regular sensor nodes u and v can easily establish a pairwise key as long as one of them can communicate with at least one assisting node. They won't be able to setup a pairwise key if neither of them can talk to any assisting node. Hence, the probability P_e of establishing a pairwise key is equivalent to the probability that either u or v can find at least one assisting node in the neighborhood.

Let (X_u, Y_u) and (X_v, Y_v) denote the actual locations of nodes u and v after deployment. Let $P_{i,u,v}$ be the probability that the assisting sensor node i can directly communicate with either of these two sensor nodes. According to the sensor deployment model, $P_{i,u,v}$ can be estimated by

$$P_{i,u,v} = \int\int_{\mathcal{D}_{u,v}} f_i(x,y)dxdy, \tag{3.2}$$

where $\mathcal{D}_{u,v}$ denotes the area covering the communication range of either u or v, i.e.,

$$\begin{aligned}\mathcal{D}_{u,v} &= \mathcal{D}_u \cup \mathcal{D}_v \\ &= \{(x,y)|((x-X_u)^2 + (x-Y_u)^2 \leq r^2) \vee ((x-X_v)^2 + (x-Y_v)^2 \leq r^2)\}.\end{aligned}$$

Thus, the probability P_e that two regular sensor nodes are able to establish a pairwise key can be estimated by

$$\begin{aligned}P_e &= 1 - \prod_{i=1}^{m}(1 - P_{i,u,v}) \\ &= 1 - \prod_{i=1}^{m}\left(1 - \int\int_{\mathcal{D}_{u,v}} f_i(x,y)dxdy\right).\end{aligned} \tag{3.3}$$

If the deployment follows a uniform distribution, P_e can be further simplified according to Equation 3.1:

$$P_e = 1 - (1 - \frac{\int\int_{\mathcal{D}_{u,v}} dxdy}{L^2})^m = 1 - (1 - \frac{A_{u,v}}{L^2})^m, \tag{3.4}$$

where $A_{u,v}$ denotes the size of $\mathcal{D}_{u,v}$. Clearly, $A_{u,v}$ is simply a function of r and the distance λ between u and v. Since the sensors' actual deployment locations are independent from each other, the expected value of the probability P_e can be calculated by

$$E(P_e) = 1 - \left(1 - \frac{E(A_{u,v})}{L^2}\right)^m.$$

Note that $A_{u,v} \ll L^2$ is usually true in practice, $E(P_e)$ can be further approximated by

$$E(P_e) \approx 1 - e^{-E(A_{u,v}) \times \frac{m}{L^2}}. \tag{3.5}$$

In this chapter, we are interested in two types of pairwise keys: (1) the key established between two one-hop neighboring sensor nodes ($\lambda \leq r$), called the *one-hop key*, and (2) the key shared between any pair of sensor nodes in the network. The former is the most useful type of pairwise keys since it directly impacts the secure communication between neighboring sensor nodes; the latter is also important in case of end-to-end secure communication. Next we will investigate the probability of establishing these two types of keys.

Probability of establishing one-hop keys: As discussed, the probability that two one-hop neighbors u and v can establish a pairwise key is equivalent to the probability that at least one of them can find an assisting node in the neighborhood. Since the assisting sensor nodes are randomly and uniformly deployed, the probability of finding an assisting node to use is determined by the size $A_{u,v}$ of the area $D_{u,v}$ covered by the radio ranges of these two sensor nodes. Let P_{en} denote the probability that two one-hop neighboring nodes can establish a pairwise key.

The case (1) in Figure 3.1 shows the area $D_{u,v}$ covered by the radio range of at least one of the two one-hop neighbors u and v. Let A_{abcd} denote the overlap of their

radio ranges, which is represented by the shadowed area in the case (1) of Figure 3.1. We can then calculate $A_{u,v}$ by

$$\begin{aligned} A_{u,v} &= 2\pi r^2 - A_{abcd} \\ &= 2\pi r^2 - \left(2r^2 \cos^{-1}\left(\frac{\lambda}{2r}\right) - \lambda\sqrt{r^2 - \frac{\lambda^2}{4}} \right). \end{aligned} \tag{3.6}$$

To further estimate P_{en}, we will calculate the expected value of $A_{u,v}$. In this case, the probability distribution function of the distance between u and v being less than the communication radius r is given by

$$F(\lambda) = P(\text{distance} < \lambda) = \frac{\lambda^2}{r^2}.$$

Thus, the probability density function $f(\lambda)$ is given by

$$f(\lambda) = F'(\lambda) = \frac{2\lambda}{r^2},$$

and the expected value of A_{abcd} is given by

$$E(A_{abcd}) = \int_0^r A_{abcd} f(\lambda) d\lambda = \left(\pi - \frac{3\sqrt{3}}{4}\right) \times r^2 = 0.5865\pi r^2. \tag{3.7}$$

Therefore, the expected value of $A_{u,v}$ can be estimated by

$$E(A_{u,v}) = 2\pi r^2 - 0.5865\pi r^2 = 1.4135\pi r^2. \tag{3.8}$$

According to Equation 3.5, the expected value of P_{en} can be calculated by

$$E(P_{en}) \approx 1 - e^{-1.4135\pi r^2 \times \frac{m}{L^2}}. \tag{3.9}$$

Equation 3.9 shows that P_{en} is determined by the number of assisting nodes (m), the communication radius (r), and the size of the field (L). We note that the item $\frac{m}{L^2}$ in the equation implies the average number of assisting nodes in unit area. We thus call it as the *assisting node density*. Obviously, P_{en} increases exponentially with

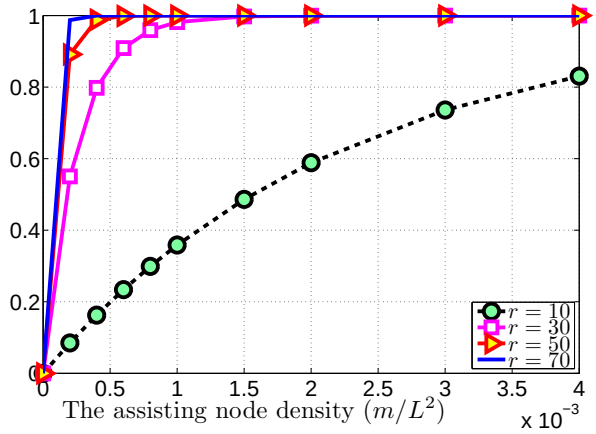


Figure 3.2. The expected value of probability P_{en} v.s. $\frac{m}{L^2}$.

the assisting node density. We can thus easily choose the number of assisting nodes to deploy to meet a certain level of performance. We also note that the number (n) of regular sensor nodes does not impact P_{en}. Indeed, as long as the sensor deployment ensure that a given area is covered by at least one assisting sensor node, any regular node located in this area can setup pairwise keys with its one-hop neighbors. Thus, in our later analysis, instead of using the percentage of assisting nodes ($\frac{m}{n}$), we will use the assisting node density ($\frac{m}{L^2}$) as one parameter to evaluate the performance of our protocol.

Figure 3.2 shows the relationship between the expected value of probability P_{en} and the assisting node density $\frac{m}{L^2}$ given different values of sensor signal radius r. From the figure, we can clearly see that P_{en} does increase with the density of assisting nodes, although the increasing speed varies with different values of r. Table 3.2 lists the signal ranges of several current sensor platforms [20]. Given a reasonable signal radius, e.g., $r = 50$, we can see that a small assisting node density (e.g., 0.002) can guarantee a high probability (greater than 0.89) of establishing keys between one-hop neighboring sensor nodes. In addition, we note that P_{en} is close to 1 when $r = 30$

Table 3.2. Examples of sensor platform signal ranges

Platform	Outdoor Range	Indoor Range
TelosB	75m to 100m	20m to 30m
MICAz	75m to 100m	20m to 30m
MICA2	150m	—
IRIS	> 300m	> 50m
Imote2	> 30m	

and the assisting node density is 0.001. This is because when $r = 30$ and the density is 0.001, a sensor node can find at least one assisting node within its radio range with a very high probability.

Probability of establishing a key between any two nodes: Let P_{ea} denote the probability that any two sensor nodes in the network can establish a pairwise key. Obviously, P_{ea} can be estimated by computing the probability of finding an assisting node in $D_{u,v}$, i.e., the area covered by their radio ranges. To estimate the size $A_{u,v}$ of this area, we have to consider the following two cases differently: (1) there is an overlap between the radio ranges of u and v ($\lambda \leq 2r$), and (2) there is no overlap between their radio ranges ($\lambda > 2r$).

- *Case 1:* This case includes both type (1) and type (2) relationships in Figure 3.1. The probability $P_{a,1}$ that two given sensor nodes are within two-hop range is given by

$$P_{a,1} = \frac{\pi(2r)^2}{L^2}. \tag{3.10}$$

  Let $E(A_{u,v,1})$ denote the expected value of $A_{u,v}$ in case 1. The probability distribution function of the distance between u and v being less than $2r$ is given by

$$F(\lambda) = P(\text{distance} < \lambda) = \frac{\lambda^2}{(2r)^2}.$$

Thus, the probability density function $f(\lambda)$ is given by

$$f(\lambda) = F'(\lambda) = \frac{\lambda}{2r^2},$$

and the expected value of A_{abcd} is given by

$$\begin{aligned} E(A_{abcd}) &= \int_0^{2r} A_{abcd} f(\lambda) d\lambda \\ &= \int_0^{2r} \left(2r^2 \cos^{-1}\left(\frac{\lambda}{2r}\right) - \lambda\sqrt{r^2 - \frac{\lambda^2}{4}} \right) \times \left(\frac{\lambda}{2r^2}\right) d\lambda \\ &= \frac{\pi r^2}{4}. \end{aligned}$$

Therefore, the expected value of $A_{u,v}$ can be estimated by

$$E(A_{u,v,1}) = 2\pi r^2 - E(A_{abcd}) = \frac{3\pi r^2}{4}. \tag{3.11}$$

- *Case 2:* This case is shown as type (3) relationship in Figure 3.1. Let $E(A_{u,v,2})$ denote the expected value of $A_{u,v}$ in case 2, and $P_{a,2}$ denote the probability that two nodes are more than two hops away. Clearly, $A_{u,v,2}$ and $P_{a,2}$ can be estimated by

$$A_{u,v,2} = 2 \times \pi r^2, \tag{3.12}$$

$$P_{a,2} = 1 - \frac{\pi(2r)^2}{L^2}. \tag{3.13}$$

According to Equations 3.10, 3.11, 3.12, and 3.13, we have

$$E(A_{u,v}) = A_{u,v,1} \times P_{a,1} + E(A_{u,v,2}) \times P_{a,2} = 2\pi r^2 - \frac{5\pi^2 r^4}{L^2}. \tag{3.14}$$

Therefore, the expected value of P_{ea} can be calculated by

$$E(P_{ea}) \approx 1 - e^{-E(A_{u,v}) \times \frac{m}{L^2}} = 1 - \left(e^{-2\pi r^2 \times \frac{m}{L^2}}\right) \times \left(e^{\frac{5\pi^2 r^2}{L^2} \times \frac{m}{L^2}}\right). \tag{3.15}$$

Figure 3.3 shows the relationship between the expected value of probability P_{ea} (denoted by $E(P_{ea})$) and the density of the assisting nodes $\frac{m}{L^2}$. We can see that the

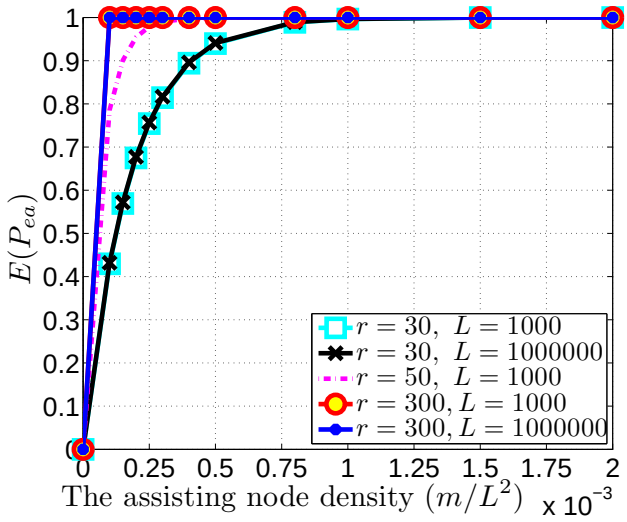


Figure 3.3. The expected value of probability P_{ea} v.s. $\frac{m}{L^2}$.

increasing speed of P_{ea} varies significantly with the signal radius r instead of the field size (L). That is because the item $\left(e^{\frac{5\pi^2 r^2}{L^2} \times \frac{m}{L^2}}\right)$ in Equation 3.15 is very close to 1 given a reasonable setting ($r < L$ and $\frac{m}{L^2} < 0.005$). Therefore, P_{ea} can be further approximated by

$$E(P_{ea}) \approx 1 - e^{-2\pi r^2 \times \frac{m}{L^2}}.$$

Probability of establishing keys through supplemental key establishment: In supplemental key establishment, a sensor node u can use the assisting nodes within its h-hop range to establish keys with other nodes. Let r' denote node u's *searching radius*, within which all the assisting nodes will be used by u to establish pairwise keys with other nodes. Obviously, without supplemental key establishment, $r' = r$. During the supplemental key establishment, u will extend its searching radius r' to h hops away from itself. This can increase the chance of successfully communicating with at least one assistant node. We simply estimate r' in this case as $r' \approx h \times r$. Thus, according to Equations 3.9 and 3.15, we can estimate $E(P_{en})$ (the expected

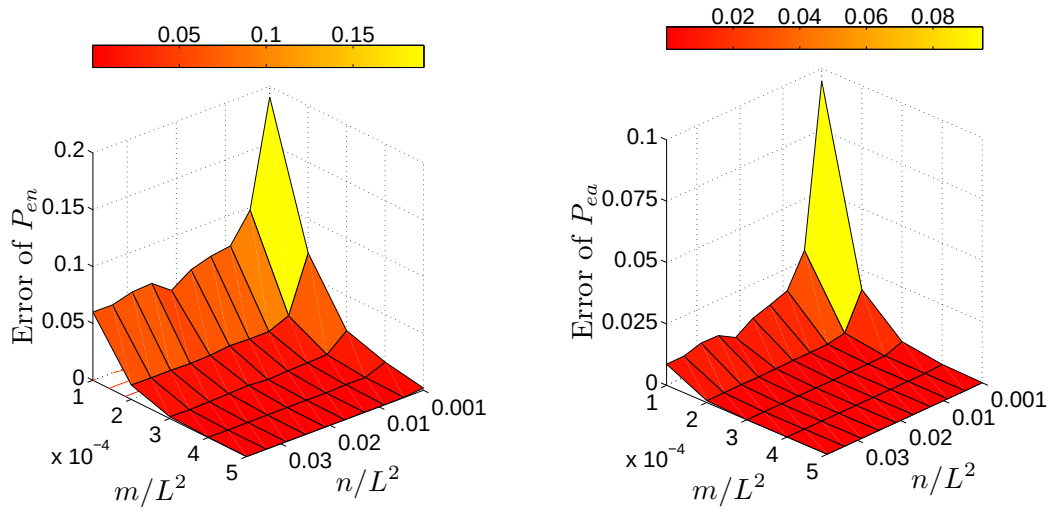


Figure 3.4. The error in estimating the probability of establishing a pairwise key. (a) error of P_{en} and (b) error of P_{ea}. The searching radius r' is approximated by $h \times r$. Assume $h = 2$, $r = 40$, and $L = 1000$.

value of the probability of establishing a key between two one-hop neighbors) and $E(P_{ea})$ (the expected value of the probability of establishing a key between any two sensor nodes) as follows

$$E(P_{en}) \approx 1 - e^{-1.4135\pi(hr)^2 \times \frac{m}{L^2}}, \quad \text{and}$$

$$E(P_{ea}) \approx 1 - \left(e^{-2\pi(hr)^2 \times \frac{m}{L^2}}\right) \times \left(e^{\frac{5\pi^2(hr)^2}{L^2} \times \frac{m}{L^2}}\right).$$

Obviously, such estimation is not precise, because r' is usually less than $h \times r$ in reality, and the *extended search scope* may not be exactly a circle. However, the sensor nodes are usually densely deployed in the target field to guarantee that the network is fully connected and the entire field is fully covered by the sensing ranges of regular sensor nodes. The larger the density of the sensor deployment, the smaller the error of such estimation. We have also conducted a simulation study to evaluate

the error of such theoretical analysis when $h = 2$ (i.e., when two-hop neighboring assisting nodes are used). In the simulation, we set the radio radius $r = 40$ and the deployment field length $L = 1000$, and vary the regular node density $\frac{n}{L^2}$ and the assisting node density $\frac{m}{L^2}$. The evaluation result is shown in Figure 3.4. We can see that given a reasonable configuration, the theoretical result is very close to the simulation result. For example, if $\frac{n}{L^2} \geq 0.005$ and $\frac{m}{L^2} \geq 0.0002$, the error is smaller than 0.0025.

We have also used simulation studies to evaluate the overall benefit of supplemental key establishment. In the simulation, we assume that 10,000 regular sensor nodes are randomly deployed in a 1000×1000 area. The sensor node deployment location follows the uniform distribution that is determined by Equation 3.1. We set the wireless signal radius r to 40. The assisting node density $\frac{m}{L^2}$ varies from 0.0001 to 0.0006. For each value of $\frac{m}{L^2}$, we run the simulation for 100 times and compute the mean of the simulation results.

Figure 3.5 shows both the simulation result and the analytical result. We can see that they match very well. It also indicates that by using two-hop assisting nodes, the probability of establishing a pairwise key between two sensor nodes increases dramatically. For example, even when the assisting node density is only 0.0001, the probability P_{en} of establishing a pairwise key between one-hop neighbors is 0.84, and the probability P_{ea} of establishing a pairwise key between any two sensor nodes is 0.97.

3.3.3 Resilience against Node Captures

We assume that once the attacker captures a sensor node, it can learn all the secrets in this node, including the master key, all pairwise keys established with other nodes, or all preloaded hash images if it is an assisting node. This section will focus on

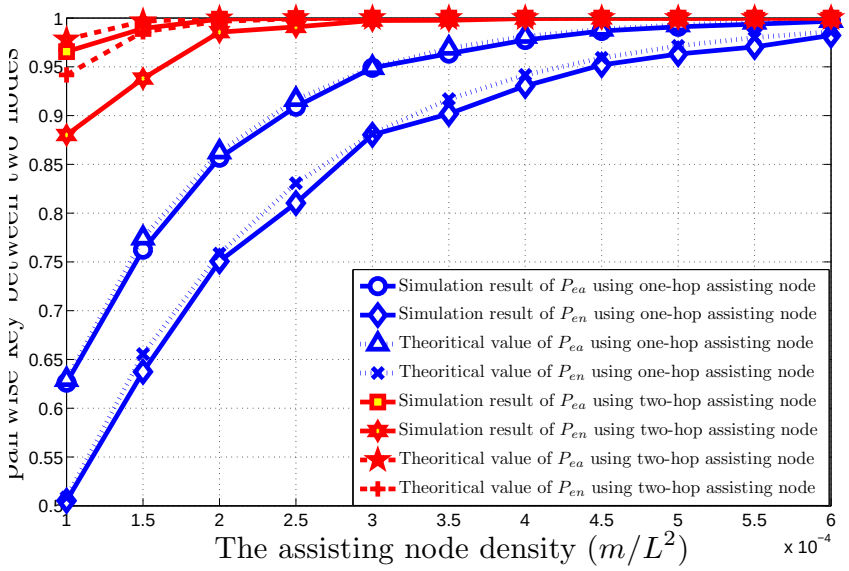


Figure 3.5. The probability that two sensor nodes can establish a pairwise key ($r = 40$ and $L = 1000$).

the security of the master key in any non-compromised sensor node and the pairwise keys shared between non-compromised sensor nodes when there are captured (either regular or assisting) sensor nodes. However, we note that the regular sensor nodes involved in relaying messages to/from assisting nodes cannot understand the content of the messages used in our key establishment protocol. Hence, compromising regular sensor nodes will not help attackers to infer the master key in any non-compromised node or any pairwise key between two non-compromised sensor nodes. Thus, we will focus on the security of our approach when there are compromised assisting nodes.

The master key of a non-compromised sensor node is immune from the node capture attack. The reason is that every sensor node u's master key is only known by the base station and u itself. The assisting nodes are only preloaded with the hash of node u's master key. Due to the one-way property of the hash function, it is computationally infeasible to compute the original key from a hash image. Therefore,

even if some assisting nodes are compromised, the master key of any non-compromised node will be always safe.

Next we will study the security of pairwise keys shared between those non-compromised sensor nodes. In our approach, every sensor node computes the final key by applying the XOR operation on all received keys. As a result, a pairwise key will be unknown to an adversary unless the adversary compromises all the assisting nodes that provide the corresponding keys. This indicates an attractive property of our scheme: *every benign assisting node can guarantee the security of pairwise keys established by any sensor node in its neighborhood as long as it can communicate with this sensor node.*

Certainly, the adversary may try to capture all benign assisting nodes in a small region to compromise the communication in this area. However, assisting nodes are randomly deployed; precisely locating and then capturing specific assisting nodes require substantial effort. In particular, the adversary will need to monitor the communication for a certain period of time and use traffic analysis to identify assisting nodes. By the time the adversary locates all assisting node in a given area, it is very likely that all or most of the pairwise keys in this area have been established. With our security reinforcement idea in Section 3.3.7, the adversary cannot recover any of the previously established pairwise keys. In this chapter, we thus assume that the adversary randomly compromises the node in his monitoring range. We use the number of nodes to compromise to measure the attacker's effort. Under this assumption, it is still possible that an adversary compromises all benign assisting nodes in a region. However, our later analysis in this subsection will show that the resilience of our scheme is significantly better than existing schemes.

We then analyze the overall probability P_c that the pairwise key $K_{u,v}$ between two non-compromised nodes u and v is compromised when a fraction f_c of sensor

nodes, including those assisting nodes, are captured. Let t denote the number of assisting nodes involved in the establishment of this pairwise key. From our earlier analysis, P_c is equal to the probability that all these assisting sensor nodes are captured. Assume that the attacker randomly captures sensor nodes. We know that P_c can be estimated by

$$P_c = f_c^t. \tag{3.16}$$

Let $E(t)$ denote the expected value of the random variable t, i.e., the number of assisting nodes in a sensor's neighborhood. P_c can thus be estimated by computing the probability of key $K_{u,v}$ being compromised when u and v have a total number of $E(t)$ assisting nodes in their neighborhood. If sensor node i's deployment location follows the pdf $f_i(x, y)$, we have

$$E(t) = \sum_{i=1}^{m} \left(\int\int_{\mathcal{D}_{u,v}} f_i(x, y) dx dy \right). \tag{3.17}$$

According to Equation 3.1, we have $E(t) = E(A_{u,v}) \times \frac{m}{L^2}$. Similar to the discussion in Section 3.3.2, we will estimate $E(t)$ and P_c for two types of pairwise keys: (1) the *one-hop* key shared between two one-hop neighboring sensor nodes; and (2) the key shared between any pair of sensor nodes in the network.

- For the key shared between two one-hop neighbors, we estimate the probability P_{cn} that it is compromised according to Equation (3.8):

$$\begin{aligned} E(t) &= 1.4135\pi r^2 \times \frac{m}{L^2}, \\ P_{cn} &= f_c^{E(t)} = f_c^{1.4135\pi r^2 \times \frac{m}{L^2}}. \end{aligned} \tag{3.18}$$

- For the key shared between a random pair of sensor nodes in the field, we estimate P_{ca} that it is compromised according to Equation (3.14), assuming a reasonable setting ($r < L$ and $\frac{m}{L^2} < 0.005$):

$$E(t) = \left(2\pi r^2 - \frac{5\pi^2 r^4}{L^2}\right) \times \frac{m}{L^2} \approx 2\pi r^2 \times \frac{m}{L^2},$$
$$P_{ca} = f_c^{E(t)} \approx f_c^{2\pi r^2 \times \frac{m}{L^2}}. \tag{3.19}$$

Figure 3.6 shows the probability of a pairwise key being compromised when $r = 40$ and $\frac{m}{L^2} = 0.0005$. It includes the results from both theoretical analysis and simulation studies. In the simulation experiments, after establishing the pairwise keys using our scheme, we randomly compromise fraction f_c of assisting nodes, and then we count the number of pairwise keys which are compromised. We calculate the fraction of compromised keys by

$$P_{cn} = \frac{\text{the number of compromised keys between one-hop neighbors}}{\text{the number of established keys between one-hop neighbors}}, \tag{3.20}$$

and

$$P_{ca} = \frac{\text{the number of compromised keys between any pair of nodes}}{\text{the number of established keys between any pair of nodes}}. \tag{3.21}$$

We run simulation 100 times and present the results using the box-plot. From the figure, We can see that even if 50% of assisting sensor nodes are compromised, the fraction of compromised keys shared between non-compromised nodes is still less than 0.2. This shows that our approach is highly resilient to the node compromise attack. We can also see that the theoretical result is very close to the simulation result. We can thus use our theoretical result to predict the trend as well as other characteristics of P_{cn} and P_{ca}.

Figure 3.7 shows the fraction of compromised keys between non-compromised sensor nodes given different assisting node density $\frac{m}{L^2}$. It clearly indicates that we

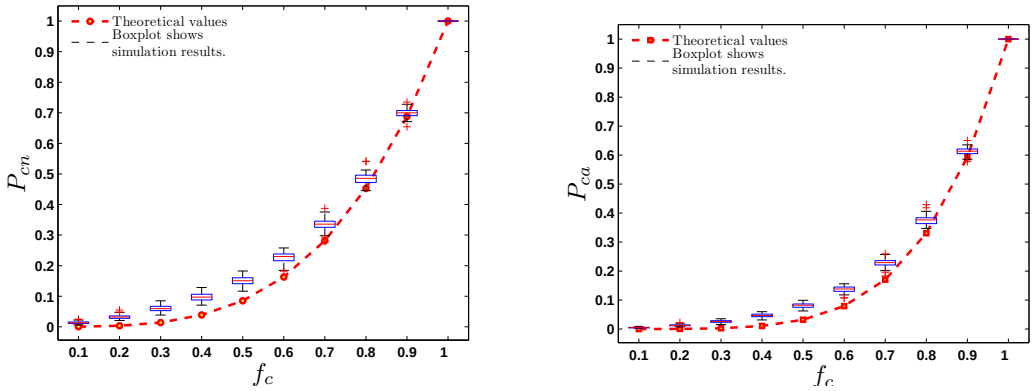


Figure 3.6. The fraction of compromised keys between non-compromised nodes. (a) the keys between two one-hop neighbors and (b) the keys between any pair of nodes ($r = 40$ and $\frac{m}{L^2} = 0.0005$).

can enhance the security of pairwise keys by deploying more assisting sensor nodes. Figure 3.7 also shows the fraction of compromised keys established using the assisting sensor nodes that are no more than two hops away in supplemental key establishment. The result is consistent with our analysis in the previous section, i.e., supplemental key establishment can greatly improve the probability of establishing keys. Figure 3.7 also shows another benefit of the supplemental key establishment: it can enhance the resilience against node capture attack. Indeed, the fraction of compromised keys can be reduced by using the two-hop neighboring assisting nodes for pairwise key establishment. This is because the attacker has to compromise more assisting nodes in order to compromise the pairwise key.

3.3.4 Overhead

This section discusses the overhead of the proposed scheme, including the storage overhead, the communication overhead, and the computation overhead. In the proposed scheme, most of the sensor nodes (i.e., the n regular sensors) only need to

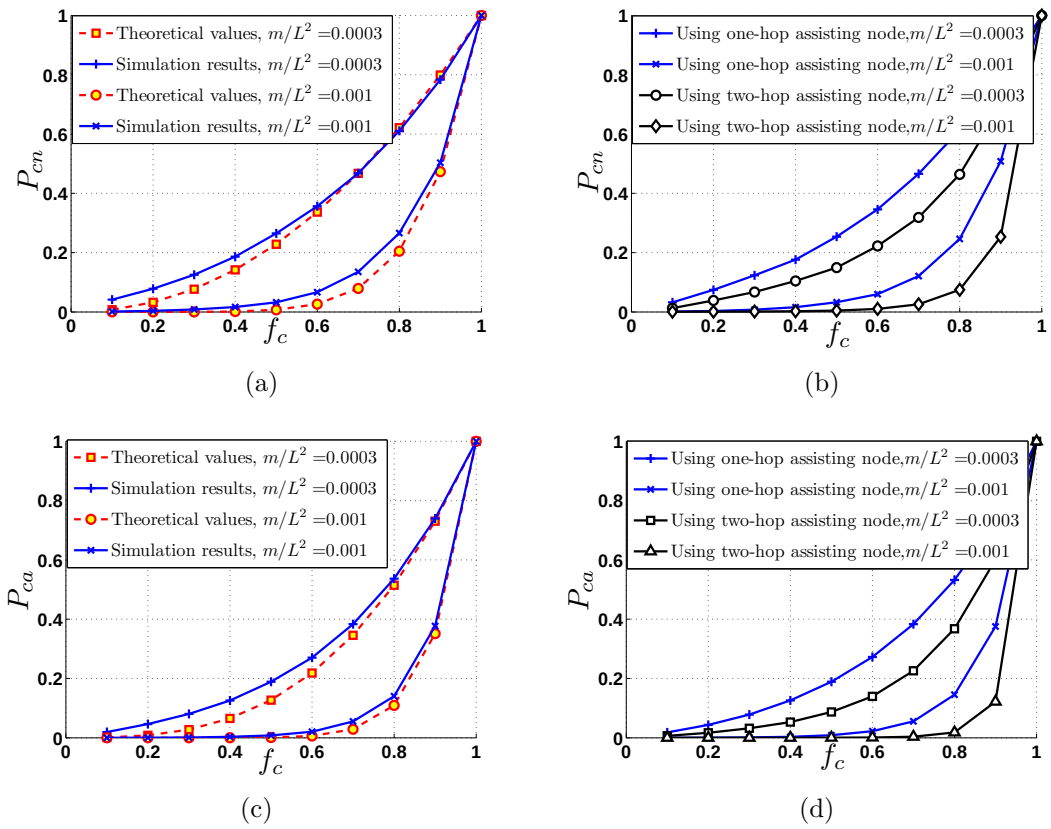


Figure 3.7. The fraction of the compromised keys. (a) the keys between two one-hop neighbors; (b) the keys between two one-hop neighbors; (c) the keys between any pair of benign nodes; (d) the keys between any pair of benign nodes ($r = 40$).

store a single master key, while only a small portion of sensor nodes (i.e., the m assisting nodes) need to store n hash values of the master keys in regular sensor nodes. As we discussed before, the assisting sensor nodes are only responsible for pairwise key establishment and can use all the memory, including the flash memory, to store these hash values.

Next we evaluate the communication cost for sensor nodes u and v to establish their pairwise key. From our earlier analysis, we know that on average, u and v can find $E(t)$ assisting nodes in the neighborhood. During key establishment, u and v will need to send a request to each of these neighboring assisting nodes to get a key. All

these messages are exchanged in one-hop range. The number of messages exchanged with assisting nodes is $2E(t)$. Due to the limited number of neighboring assisting nodes for each sensor node (small $E(t)$), we believe that the communication cost to establish a pairwise key using one-hop neighboring assisting nodes will not be a big problem for sensor networks.

As we discussed before, even without supplemental key establishment, we can establish pairwise keys between sensor nodes at a very high probability. The supplemental key establishment is usually used when a sensor node cannot find any neighboring assisting node in its radio range. As a result, it will be rarely used. In addition, even if the supplemental key establishment is needed, it will only introduce some additional communication within the h-hop area around the node that needs help for key establishment. The communication cost needed during supplemental key establishment mainly depends on the average number of assisting nodes found within h hops.

The proposed scheme involves small computation overhead. For every neighboring assisting node i, sensor node u will first need to apply one hash operation to compute the key $H(K_u||i)$. To establish a pairwise key with sensor node v, node u will then apply symmetric key operations for $3 \times E(t)$ times on average to generate $E(t)$ key establishment requests, recover $E(t)$ keys from its neighboring assisting nodes, and compute $E(t)$ keys for the message from v. Finally, both u and v will apply one XOR operation on the two partial keys to compute their final pairwise key.

Note that Elliptic Curve Cryptography (ECC) has been demonstrated to be feasible on sensor platforms. Another choice for key establishment would be to directly use ECC-based approaches. Next we will compare our scheme with the ECC-based approach in TinyECC [14] in terms of overhead. We also noticed that the main cost of ECC-based approaches is the computation cost, and the main cost of our approach

is the communication cost. As a result, we focus on the comparison between the cost of public key cryptographic operations involved in ECC-based approaches and the cost of communication involved in using additional assisting nodes. Note that no matter how many assisting nodes are used, the key combination only needs to be performed once. Thus, we focus on the communication cost involved in deriving the partial keys.

TinyECC include ECDSA, which can be used to sign and verify messages, and ECDH, which can be used to establish keys between sensor nodes. To establish a key, a sensor node will need to receive the certified public key of the other node and then verify the signature included in the message. On TelosB platforms, the energy needed for one verification is about 49.37 mJ [14] ($init + verify$). Once the public key is verified, the ECDH key exchange protocol can be used to establish a pairwise key. One round of ECDH consumes about 33.19 mJ of energy ($init$+$key\ establish$). Hence, the energy consumed in total will be about 82.56 mJ.

In TelosB, transmitting one bit consumes about 3.0V×18.8mA/(250kbps) ≈ 0.2088 μJ of energy, and receiving one bit consumes about 3.0V×17.4mA/(250kbps) ≈0.2256 μJ. Assume that each symmetric key is 8 bytes long, each node ID is 2 bytes long, and each random nonce is 4 bytes long. Each request to or reply from an assisting node is 14 bytes long. In TelosB, the MAC layer will add 25 more bytes, and the physical layer will add 6 more bytes. Each request or reply will thus incur the transmission of 45 bytes. Thus, overall, our protocol incurs the transmission and receiving of $2 \times 45 = 90$ bytes for each assisting node used in establishing a pairwise key between two nodes. This will consume $90 \times 8 \times (0.2088 + 0.2256)\mu$J ≈ 0.313 mJ. Therefore, as long as we do not use more than $\frac{82.56}{0.313} \approx 264$ assisting nodes, the cost will be less than the cost for using ECDSA-ECDH.

Another advantage of our approach when compared with ECC-based approaches is that ECC-based approaches allow an adversary to launch DoS attacks to easily and quickly exhaust the energy at sensor nodes [26, 51]. The adversary can simply spam fake signatures to force sensor nodes to perform a significant large number of expensive but unnecessary signature verifications. Current counter measures either require shared pairwise keys between sensor nodes [51] or only control the spreading of fake signatures in the context of broadcast authentication [26]. None of them can be used to address pairwise key establishment effectively.

3.3.5 Security Performance under Other Typical Attacks

In this section, we will analyze the security of the proposed scheme under some other attacks in sensor networks. We focus on sybil attacks [46], wormhole attacks [47], and node replication attacks [48].

Clearly, the sybil attack will not impact the security of pairwise keys shared between sensor nodes since the adversary cannot generate legitimate master keys for those fake nodes. Similarly, the wormhole attack will not allow the adversary to infer the pairwise key shared between sensor nodes. Indeed, wormhole attacks allow sensor nodes to find more assisting sensor nodes, which actually increases the chance of establishing pairwise keys and enhances the resilience against node compromise attacks.

However, the node replication attacks do impact the security of our scheme. By launching node replication attacks, the adversary tries to fool sensor nodes into believing that some compromised assisting nodes are in their neighborhood. Thus, a benign sensor node may communicate with compromised assisting nodes for key establishment. Fortunately, the final pairwise key is generated from all random keys provided by neighboring assisting nodes. As long as one of the assisting nodes is

benign, the final key will be safe. Therefore, as long as a regular sensor node can communicate with at least one benign assisting node, its pairwise keys will be always safe no matter how many malicious assisting nodes are created and placed in its neighborhood.

On the other hand, if two sensor nodes have no benign assisting node in their neighborhood to use due to the deployment error, the pairwise key established between them will be insecure. Specifically, these two victim nodes will accept compromised assisting nodes as their only neighboring assisting nodes and use them to establish their pairwise key. A possible countermeasure is to have sensor nodes periodically use some new assisting node from multi-hop away, which they have not used before. This will increase the chance of using non-compromised assisting nodes in generating the final key.

3.3.6 Comparison with Previous Schemes

In this section, we will compare our scheme with previous techniques for pairwise key establishment in terms of security and overhead. In particular, we will compare it with the basic probabilistic scheme [3], the q-composite scheme [4], the random subset assignment scheme [5], and the grid-based scheme [5]. We do not include those location-based techniques [8, 9, 11, 12] in comparison since our technique does not require any location knowledge.

Security: We use simulation to compare the performance of different schemes. Assume that $n = 20,000$ regular sensor nodes are evenly deployed in the field, and each of them have 50 neighbors. Since the average number of neighbors for every sensor node can be estimated by $\frac{\pi r^2}{L^2} \times n$, we have $\frac{\pi r^2}{L^2} = 2.5 \times 10^{-3}$. For previous schemes, we assume that each sensor node can store 200 keys or polynomial coefficients. Hence, for the grid-based scheme [5], the probability of two nodes sharing a

direct key is 0.014. For all other previous schemes, we set $P = 0.33$ to make sure the network is well connected. From our previous analysis, we know that our approach can guarantee the establishment of pairwise keys at a very high probability (e.g., 0.9) using a small assisting node density (e.g., $\frac{m}{L^2} = 0.002$).

Figure 3.8 shows the fraction of compromised links in the presence of compromised nodes for different schemes ("Basic-1" represents the basic probabilistic scheme with $P_e = 0.014$; "Basic-2" represents the basic probabilistic scheme with $P_e = 0.33$; "q-Comp-1" represents the q-composite scheme with $q = 2, P_e = 0.33$; "q-Comp-1" represents the q-composite scheme with $q = 3, P_e = 0.33$; "RS-1" represents the random subset assignment scheme with $P_e = 0.014$; "RS-1" represents the random subset assignment scheme with $P_e = 0.33$; "Grid" represents the grid-based scheme with $P_e = 0.014$; "Ours" represents the proposed scheme with $\frac{m}{L^2} = 0.0005$, and $P_e = 0.99$). The figure tells us that in terms of protecting the direct keys, our scheme can provide high resilience to node compromises, which is similar to the random subset assignment scheme and the grid-based scheme [5]. In addition, we must remember that our scheme can guarantee a much higher probability of establishing a pairwise key between two sensor nodes in a densely deployed sensor network.

Note that the previous schemes need to employ expensive protocols for path key establishment when two sensor nodes cannot directly setup a pairwise key. As a result, the attacker might discover not only the direct keys but also the indirect (path) keys by compromising the intermediate nodes used in the establishment of the indirect (path) keys. However, our approach does not need to setup path keys. Even if the attacker has captured the nodes that relay the keying information, the key will not be disclosed. Figure 3.9 compares the fraction of compromised (direct or indirect) keys between non-compromised nodes in the presence of compromised nodes ("Basic-1" represents the basic probabilistic scheme with $P_e = 0.014$; "Basic-2" represents the

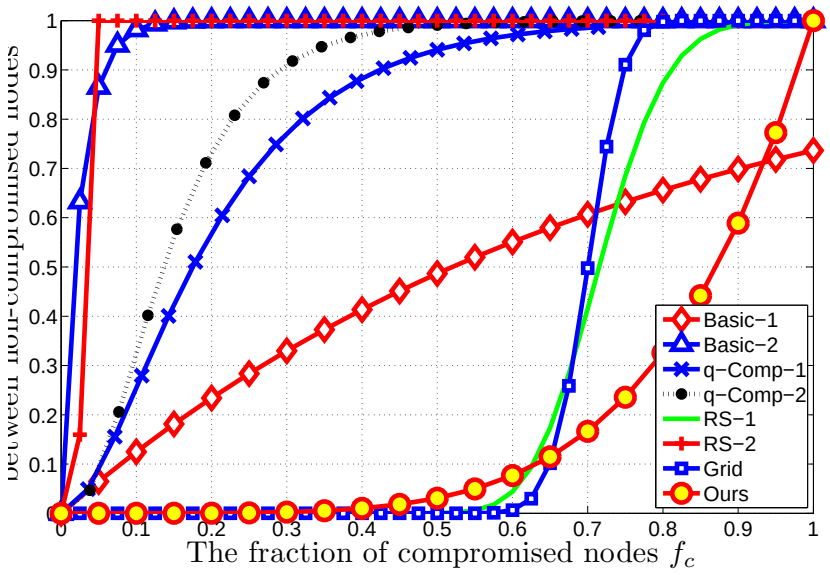


Figure 3.8. The fraction of compromised links between non-compromised nodes in the different schemes.

basic probabilistic scheme with $P_e = 0.33$; "q-Comp-1" represents the q-composite scheme with $q = 2, P_e = 0.33$; "q-Comp-1" represents the q-composite scheme with $q = 3, P_e = 0.33$; "RS-1" represents the random subset assignment scheme with $P_e = 0.014$; "RS-1" represents the random subset assignment scheme with $P_e = 0.33$; "Grid" represents the grid-based scheme with $P_e = 0.014$; "Ours" represents the proposed scheme with $\frac{m}{L^2} = 0.0005$, and $P_e = 0.99$). The figure clearly shows that our scheme performs much better than other schemes. For example, when 70% sensor nodes are compromised, the fraction of compromised pairwise keys between non-compromised sensor nodes is only around 0.18. For other schemes, the adversary will be able to infer at least 88% of the pairwise keys in the network.

Additionally, our proposed scheme can guarantee that a single benign assisting node can protect the keys established by the nodes within its radio range as long as the communication is reliable. Since only local communication is required, reliable

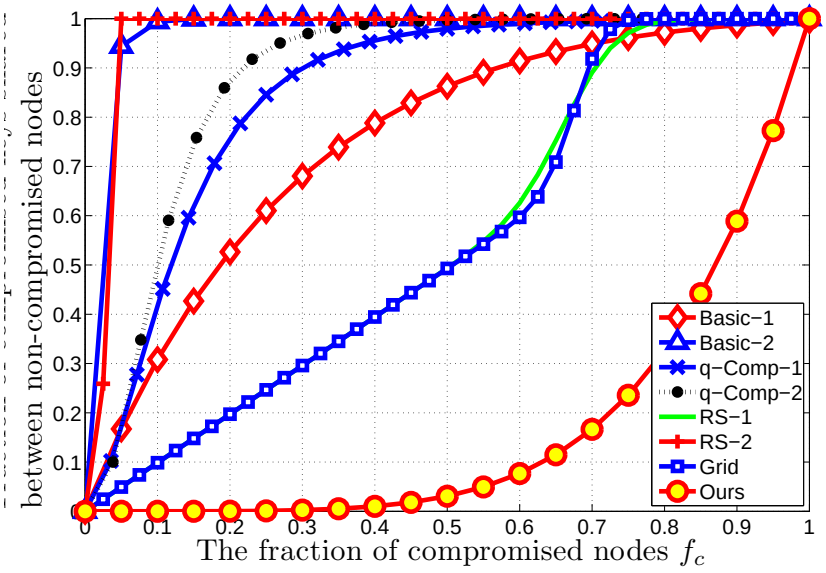


Figure 3.9. The fraction of compromised (direct or indirect) keys between non-compromised nodes in the different schemes.

communication is possible. Certainly, if the attacker completely jams the radio channel, there is no way for sensor nodes to get secure random keys from this assisting node. Fortunately, whenever the attacker stops the jamming attacks, the pairwise key establishment around this assisting sensor node will be secure again.

Overheads: In the proposed scheme, only a single master key is stored in every regular sensor node, while the previous schemes have considerable storage requirements for achieving high performance. For instance, in Figure 3.8 and Figure 3.9, the previous schemes require every node to store 200 entries to achieve the desire performance. In terms of the computation overhead, our approach involves only a few number of symmetric key operations and hash operations. Hence, it will not incur much additional overhead.

From our previous discussion, the communication overhead of our scheme mainly comes from the direct communication in one-hop range. The multi-hop communica-

tion in the supplemental key establishment rarely occurs due to the high probability of establishing pairwise keys in the previous step. On the other hand, for many previous schemes such as the grid-based scheme [5] and PIKE [7], sensor nodes often need to go through the path key establishment to setup keys with other sensor nodes. Such path key establishment could be very expensive in practice since the intermediate sensor nodes that can help establish the pairwise key may be located far away from the two sensor nodes that want to establish a shared key. According to the above discussion, we can clearly see that our proposed approach has significant advantage over existing schemes in terms of the storage and communication overhead.

3.3.7 Security Reinforcement

From our previous analysis, we note that once an assisting node is compromised, the attacker is able to discover all those random keys generated by this node. Although the actual pairwise key is combined from multiple random keys generated by different assisting nodes, it is still not desirable to let the attacker figure out the old random keys. In the following, we give a simple extension to fix this problem by updating the keys at the assisting nodes.

The basic idea is to update the key at every assisting sensor node after the pairwise key establishment. In other words, the hash key at any assisting node will be changed immediately after it is used. As a result, the attacker won't be able to learn any random key generated before even if this assisting sensor node is compromised later at certain point. To achieve this goal, we will take advantage of the one-way hash function $H(\cdot)$. We will also maintain a sequence number for every hash key shared between a regular sensor node and an assisting node. For example, initially, the hash $H_{u,i} = H(K_u||i)$ and the sequence number $S_u = 0$ will be stored in the assisting node i for the regular sensor node u before deployment.

Once the assisting node i receives the request from u to setup a pairwise key with another node v, it will send two encrypted copies of a random key R to u along with the current sequence numbers S_u and S_v via the secure link established based on the hash key $H_{u,i}$ shared between i and u. These two copies of the random key R are encrypted and authenticated by the hash keys $H_{u,i}$ and $H_{v,i}$, respectively. Finally, the assisting node i will replace $H_{u,i}$ with $H(H_{u,i})$ and $H_{v,i}$ with $H(H_{v,i})$, and also increase S_u and S_v by 1.

When the regular sensor node u receives the message from i, it can easily verify the authenticity and confidentiality of the message using the same hash key $H_{u,i}$, which can be computed based on K_u, i and S_u. Node u can then derive the random key R for pairwise key establishment. Such sequence number can certainly be used to deal with the replay attacks as well. Similarly, node v can also derive the random key R from the other encrypted copy of the random key.

By using the one-way hash function, the improved approach can greatly enhance the resilience against node captures. Indeed, a compromised assisting sensor node will not reveal any secret about the pairwise keys established before between non-compromised sensor nodes. However, compromised assisting sensor nodes can still participate in the future pairwise key establishment when new nodes are added into the network. These malicious assisting nodes may disclose valuable information to adversaries. Fortunately, our scheme guarantees that as long as there are at least one benign assisting node in a given area, the final key will be safe no matter how many sensor nodes are compromised. Based on this property, another security reinforcement idea is to *deploy new assisting sensor nodes from time to time to replace the old and untrustworthy assisting nodes.*

3.3.8 Performance under Other Deployment Models

In our previous analysis, we assume that the deployment location of a sensor node follows the uniform distribution, i.e., all the nodes are evenly deployed in the field. However, this may not always be the case in practical situations. If the sensors are deployed through different deployment methods, their actual locations may follow different probability distributions. Figure 3.10 describes four types of sensor node deployment models. Model (1) is the uniform distribution deployment model, which is used in analyzing our scheme and many previous schemes. Model (2) is Gaussian distribution deployment model, in which the sensor node location follows a Gaussian distribution. In addition to these two models, the sensor nodes may also be deployed in groups. In model (3), the deployment point of each group center is predetermined [8], while in model (4), we have no prior knowledge of the group center deployment point [52].

This section focuses on the performance of the proposed scheme under model (2), (3), and (4). Note that the theoretical analysis conducted in the previous sections can also be extended to study the performance under other deployment distribution models. For example, Equations 3.2 and 3.3 can be directly applied under these models.

Performance under Gaussian distribution model: In practice, all the sensor nodes may be deployed into the target field together. For example, they can be dropped from a helicopter or be launched by a cannon, aimed at the center of the field. The actual locations of sensor nodes are affected by many factors such as the carrier speed, the wind speed, the geographic situation, and etc. By this means, the sensor nodes are scattered to cover the target field. In this case, we use a two-dimensional Gaussian distribution to model the actual deployment locations of the

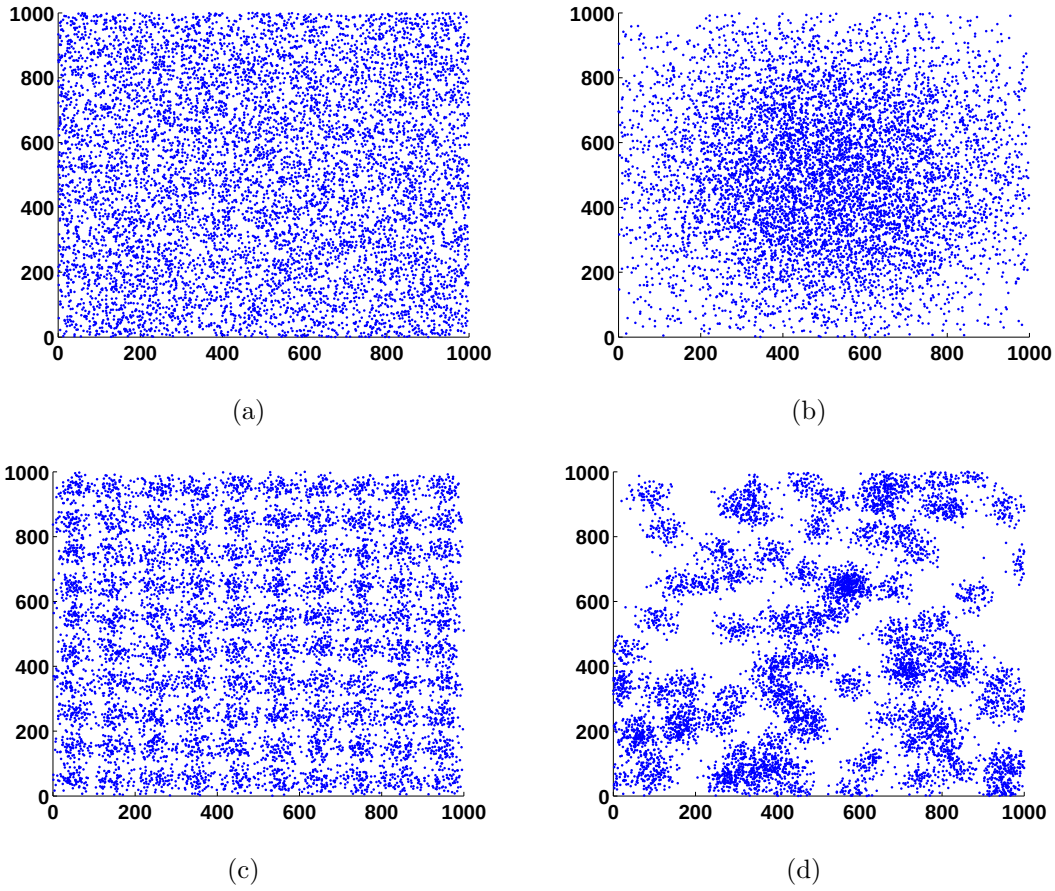
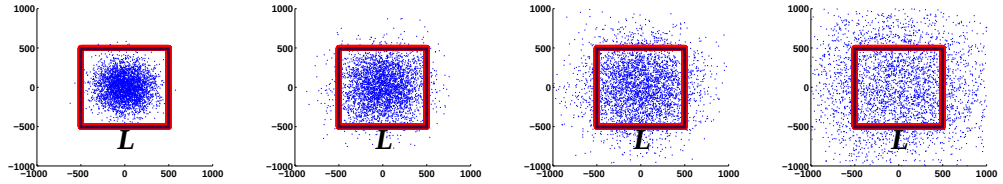


Figure 3.10. Illustration of different deployment models for sensor networks. (a) uniform distribution model; (b) Gaussian distribution model; (c) group-based model with fixed group centers (100 groups); (d) group-based model with random group centers (100 groups).

sensor nodes. Specifically, the probability density function of sensor node i's actual location (X, Y) is the following:

$$f_i(x, y) = \frac{1}{2\pi\sigma^2} e^{-[x^2+y^2]/2\sigma^2}, \tag{3.22}$$

where σ is the standard deviation. This bivariate Gaussian distribution has the mean $(0, 0)$, which denotes i's expected deployment location (i.e., the central point of deployment field). We also note that the abscissa X and ordinate Y are independent



(a) (b) (c) (d)

Figure 3.11. Gaussian deployment modes with different standard deviation σ ($L = 1000$). (a) $\sigma = \frac{L}{6}$; (b) $\sigma = \frac{L}{4}$; (c) $\sigma = \frac{L}{3}$; (d)$\sigma = \frac{L}{2}$.

and Gaussian distributed random variables with mean 0 and variance σ^2 (i.e., $X \sim N(0, \sigma^2)$, and $Y \sim N(0, \sigma^2)$).

When we deploy the sensors into a target field, we certainly want to limit the number of the waisted nodes (i.e., the nodes out of the target field) as small as possible. We also want the sensors to be scattered in the field as evenly as possible, so that every place in the field may be covered with enough nodes. In this Gaussian distribution model, σ is critical to both the field coverage and the network density distribution.

On the one hand, σ is related to the coverage area of the sensor nodes. The Gaussian distribution deployment models with different values of σ are illustrated in Figure 3.11, where the $L \times L$ rectangle denotes the target deployment field. From the figure, the area of sensor network coverage increases with the increasing σ. When $\sigma = \frac{L}{6}$ in Figure 3.11(a), almost every node falls in the region of target field. When $\sigma = \frac{L}{2}$ in Figure 3.11(d), many nodes fall out of the target region. On the other hand, σ is also related to distribution of the sensor node density in the field. Intuitively, in each case in Figure 3.11, the node density in the region close to the center is larger than the node density in the region far from the center.

To understand the impact of σ, we study the random variable $Z = \sqrt{X^2 + Y^2}$, which denotes the distance between the sensor node location and the expected lo-

cation (i.e., the center of target field). Since X and Y are independent and Gaussian distributed ($X \sim N(0, \sigma^2)$, and $Y \sim N(0, \sigma^2)$), Z is Rayleigh distributed (i.e., $Z \sim \text{Rayleigh}(\sigma)$). Specifically, its probability density function (pdf) $f_Z(z)$ and cumulative distribution function (cdf) $F_Z(z)$ are given by

$$f_Z(z) = \frac{z}{\sigma^2} e^{-z^2/2\sigma^2}, \qquad z \geq 0, \tag{3.23}$$

$$F_Z(z) = 1 - e^{-z^2/2\sigma^2}, \qquad z \geq 0. \tag{3.24}$$

Equations 3.23 and 3.24 can be derived from Equation 3.22 and the transformation from rectangular to polar coordinates. The cdf of Z is plotted in Figure 3.12(a). To evaluate the field coverage, we consider the probability that the sensor nodes fall in the circumcircle region of the $L \times L$ target field. Apparently, this circumcircle is centered at the expected location (0,0) with radius of $\frac{L}{\sqrt{2}}$. If the distance between sensor node i and the center is larger than $\frac{L}{\sqrt{2}}$, i falls out of the field and is considered as a waisted node. From Figure 3.12(a), given a certain number of sensor nodes, the larger σ is, the larger area the sensor nodes may cover. When $\sigma = \frac{L}{6}$, $\frac{L}{4}$, $\frac{L}{3}$, and $\frac{L}{2}$, the fraction of nodes which fall in that circumcircle region is about 99%, 98%, 89%, and 63%, respectively.

To evaluate the network density in the field, we consider the probability P_A that a sensor node falls into the infinitesimal area dA. Intuitively, P_A is a univariate function of z, because of the symmetry of sensor deployment result. From either Equation 3.22 or Equation 3.23, P_A can be estimated by

$$P_A = \frac{1}{2\pi\sigma^2} e^{-z^2/2\sigma^2}, \qquad z \geq 0. \tag{3.25}$$

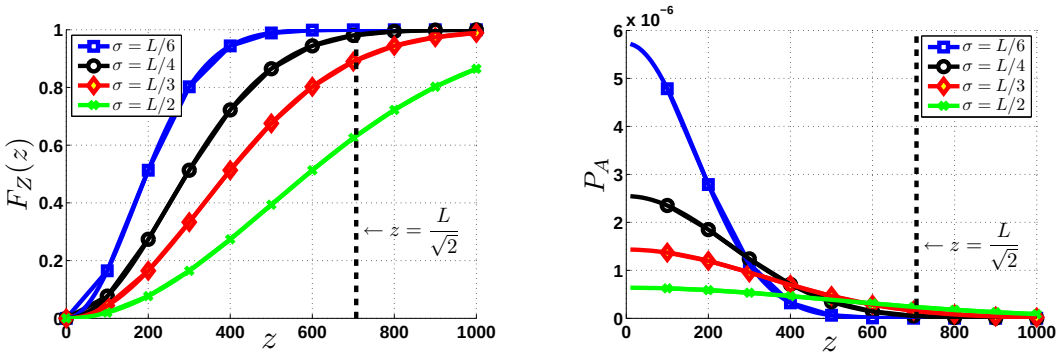


Figure 3.12. Gaussian deployment modes. (a) network coverage and (b) node density distribution ($L = 1000$).

Figure 3.12(b) shows P_A with different values of σ. From these figures, given a certain value of σ, P_A decreases with increasing z. The smaller σ is, the faster P_A decreases. In other words, larger σ implies that the sensor nodes are deployed in the field more evenly.

We conduct the simulation experiments to evaluate the performance of our scheme under this deployment model. Figure 3.13 shows the simulation results, which include the probability of establishing keys and the resilience against node capture attacks for the two type of pairwise keys (i.e., the keys between one-hop neighbors and the keys between any pair of nodes). We note that σ is also a critical parameter related to these performances. Specifically, given a fixed number of sensor nodes deployed in the field, a larger σ leads to a lower probability of establishing pairwise key and a lower level of resilience against the node capture attacks.

Since we have the trade-off between security performance, sensor network coverage, and node density evenness, we need to choose the value of σ carefully. In this chapter, we set $\sigma = \frac{L}{3}$ (shown in Figure 3.11(c)), So that 89% of the actual sensor deployment locations will fall within the circumcircle region of the $L \times L$ target field,

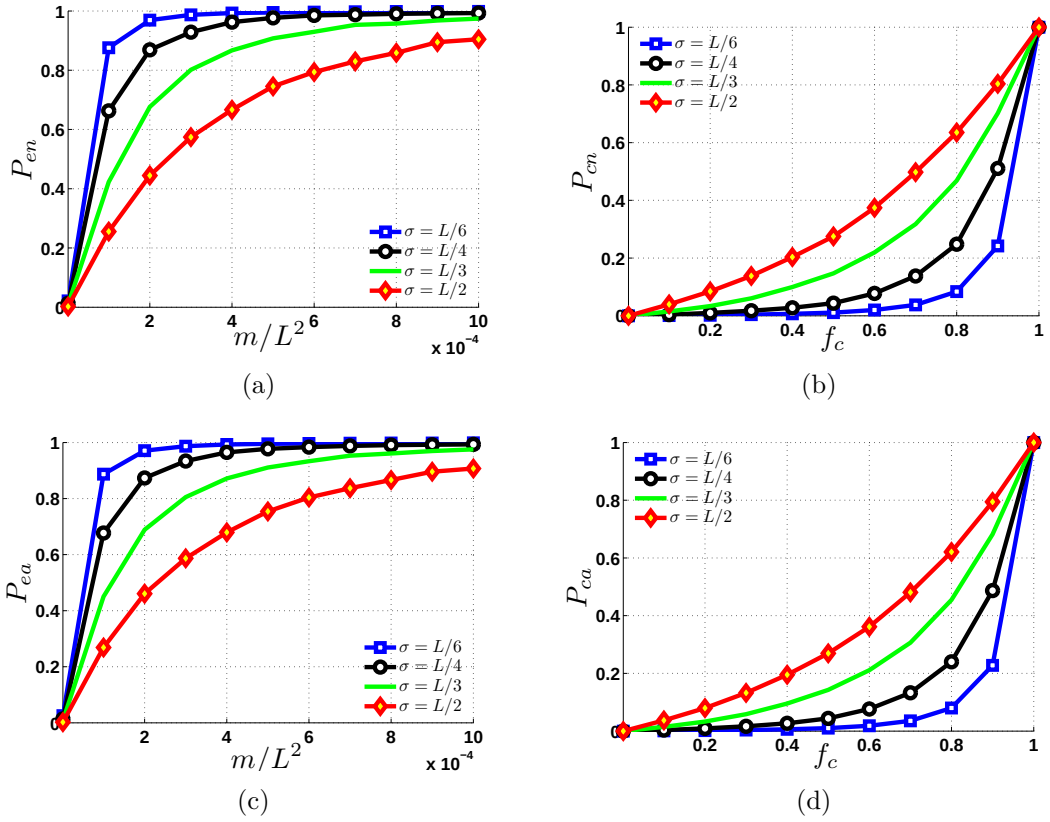


Figure 3.13. Performance under deployment model (2). Assume $r = 40$, and $\frac{m}{L^2} = 0.0005$ in (b) and (d).

which is shown in Figure 3.12(a). Meanwhile, according to Figure 3.12(b), it is also a proper choice to even the node density in the network. Figure 3.13 shows that our scheme can achieve satisfying performance when $\sigma = \frac{L}{3}$.

Performance under group-based deployment model: As discussed before, one limitation of Gaussian deployment model is the trade-off between the evenness of the sensor node densities in difference regions and the proper sensor network coverage in the target field. The group-based deployments can alleviate this problem by deploying the sensor nodes in groups. Each of the groups is intended to cover only a relatively small part of target field. Different groups have the similar sensor node densities, but have different expected deployment locations (i.e., the group centers)

scattered in the field. We hence can cover the field with many groups and narrow the gap of sensor node densities in different regions.

Since each group only needs to cover a relatively small area, the sensor nodes can cover the deployment field with a relatively small value of σ in each group. From the previous discussion, we know that the probability of establishing pairwise key increases with decreasing σ. Thus, the group-based distribution model has the potential of increasing the probability of establishing pairwise keys between sensor nodes.

In the following, we evaluate the performance of the proposed scheme when the group-based deployments are applied. These models are illustrated in Figure 3.10(c) and 3.10(d), in which the sensor nodes are deployed in groups, and each group contains a certain number of regular sensor nodes and assisting sensor nodes.

During the deployment, sensor nodes may not be placed exactly at their expected locations due to the deployment errors that can hardly be controlled. However, if the sensor nodes are deployed in groups, the nodes in the same group will likely be affected by similar factors. In this case, *the sensor nodes in the same group will be closed to each other with a high probability after deployment.* This observation has also been used in previous studies [8, 52]. However, different from their approaches, our approach needs neither the knowledge of deployment locations nor special key assignment for different groups. The only requirement is that each group includes both regular nodes and a few assisting nodes. Intuitively, under the group-based deployment model, it is easy to achieve a satisfying probability that a regular node can find at lease one assisting node in its nearby area.

Specifically, we evenly group the n regular nodes and m assisting nodes into g groups $\{G_j\}_{j=1,\ldots,g}$. Thus, each group will contain $\frac{n}{g}$ regular nodes and $\frac{m}{g}$ assisting nodes. These groups will be deployed in the $L \times L$ field independently. All the

sensor nodes in the same group follow the same deployment distribution. We use the two-dimensional Gaussian distribution to model such sensor deployment. Similar to Equation 3.22, the probability density function for the sensor nodes in group G_j is given by

$$f(x - x_j, y - y_j) = \frac{1}{2\pi\sigma^2} e^{-[(x-x_j)^2+(y-y_j)^2]/2\sigma^2}, \tag{3.26}$$

where (x_j, y_j) is the expected deployment location of the sensor nodes in group G_j (i.e. the group center). In group-based deployment, different groups have different expected deployment locations to cover the deployment field. The expected deployment locations may be fixed and evenly distributed in the field [8], as illustrated by Figure 3.10(c), or randomly distributed in the field [52], as illustrated by Figure 3.10(d).

We conduct simulation to evaluate the performance of our scheme under the group-based deployment model (3) and (4). In the experiments, we set $r = 40$, $L = 1000$, and divide the sensor nodes into g groups, where g is set to be $1, 4, 25$, or 100 separately. (When g is equal to 1, it becomes the Gaussian distribution model.) Thus, each group is deployed to cover the area with the side length $l = \frac{L}{\sqrt{g}}$. Based on the previous discussion, we set the standard deviation $\sigma = \frac{l}{3} = \frac{L}{3\times\sqrt{g}}$ to achieve both satisfying field coverage and node density evenness.

The simulation results for model (3) and (4) are presented in Figures 3.14 and **??**, respectively. Figures 3.14(a), 3.14(c), 3.15(a), and 3.15(c) show that we can improve the probability (P_{en} and P_{ea}) of establishing pairwise keys by deploying sensor nodes in groups when the assisting node number is small. Given the limited assisting node density, the more groups we use, the higher the probability of establishing keys we get.

We also evaluate the resilience against the node capture attacks by calculating the fraction of compromised pairwise keys according to Equation 3.20 and 3.21. Figure

3.14(c) and 3.14(d) show that model (3) cannot provide better resilience against the node capture attacks than model (2) the Gaussian deployment model in the experiments. In model (2), many sensor nodes (including both the regular nodes and the assisting nodes) fall in the center region of the field, which is illustrated in Figure 3.10(b). Thus, those regular nodes can find several assisting nodes for the pairwise key establishment. In model (3), the assisting nodes are evenly assigned to each group, which is illustrated in Figure 3.10(c). The more groups we have, the more evenly the sensor nodes are scattered in the field. Although this makes it easy for the regular nodes in any place of the field in model (3) to find the assisting node, the number of the assisting nodes each regular node in model (3) find is usually smaller than the number of assisting nodes that each group contains, and is also smaller than the number of assisting nodes the regular node in model (2) find. For the pairwise keys between the one-hop neighbors, the neighboring nodes often share the same assisting nodes. Once their shared assisting nodes are compromised, their pairwise key is no longer safe. For the pairwise keys between any two nodes in the network, the situation may be better. Because that pair of nodes may belong to different group, and have their own neighboring assisting nodes, the total number of the assisting nodes helping them to set up their pairwise key is usually larger than the number of each node's assisting node neighbors.

Different from mode (3), model (4) can provide similar resistance against node capture attacks to mode (2), which is shown in Figure 3.15(b) and 3.15(d). Figure 3.10(d) shows that the coverage of the groups in model (4) may overlap with each other. Thus, a regular node may not only find the assisting node neighbors in its groups, but also find the assisting node neighbors in other groups. Hence, the sensor nodes in this model usually have more neighboring assisting nodes than the nodes in model (3).

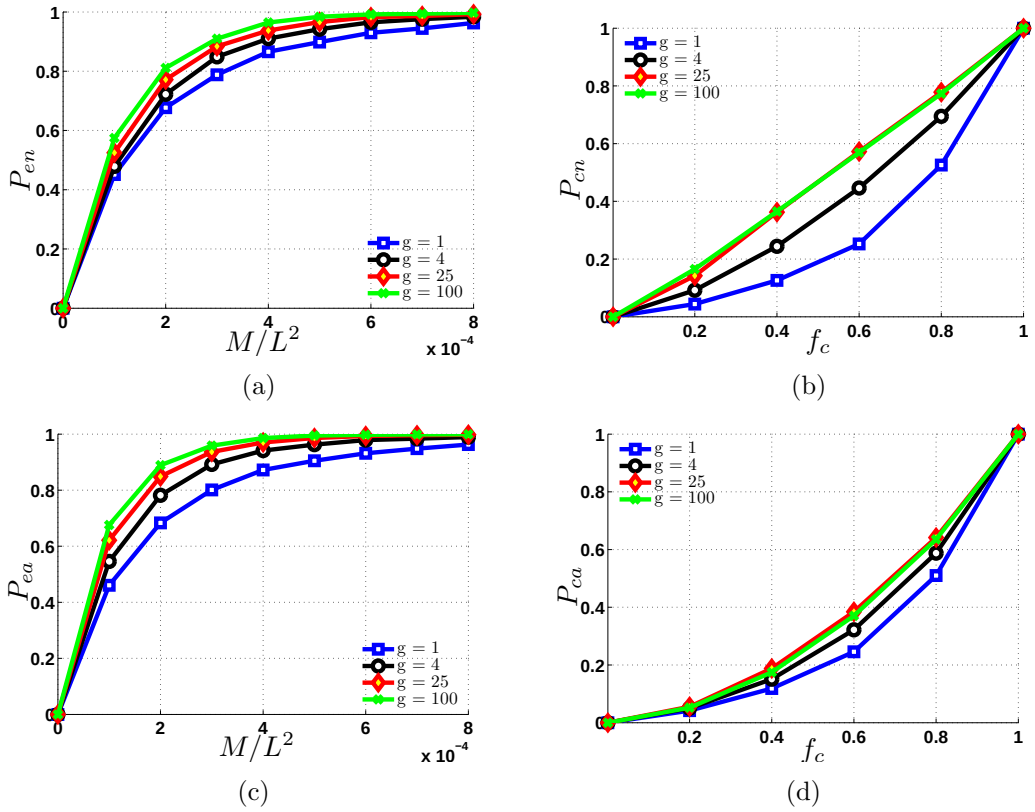


Figure 3.14. The performance under group-based deployment model (3), in which the expected group center locations are fixed and evenly distributed in the field. Assume $r = 40$, $L = 1000$ and $\frac{m}{L^2} = 0.0005$ in (c) and (d).

Finally, Figure 3.16 shows the performance of our proposed pairwise key establishment scheme under the four different sensor deployment models. Under the current parameter setting, the probability of pairwise key establishment of the uniform deployment model (1) is in between the performances of the Gaussian deployment model and group-based models. On the other hand, model (1) can provide the best resistance against the compromised nodes in the four models. We also notice that the security performance under model (4) is better than model (3). However, model (4) also has its own disadvantage. That is, it cannot guarantee the adequacy of sensor network coverage in the field, which can be observed from Figure 3.10(d).

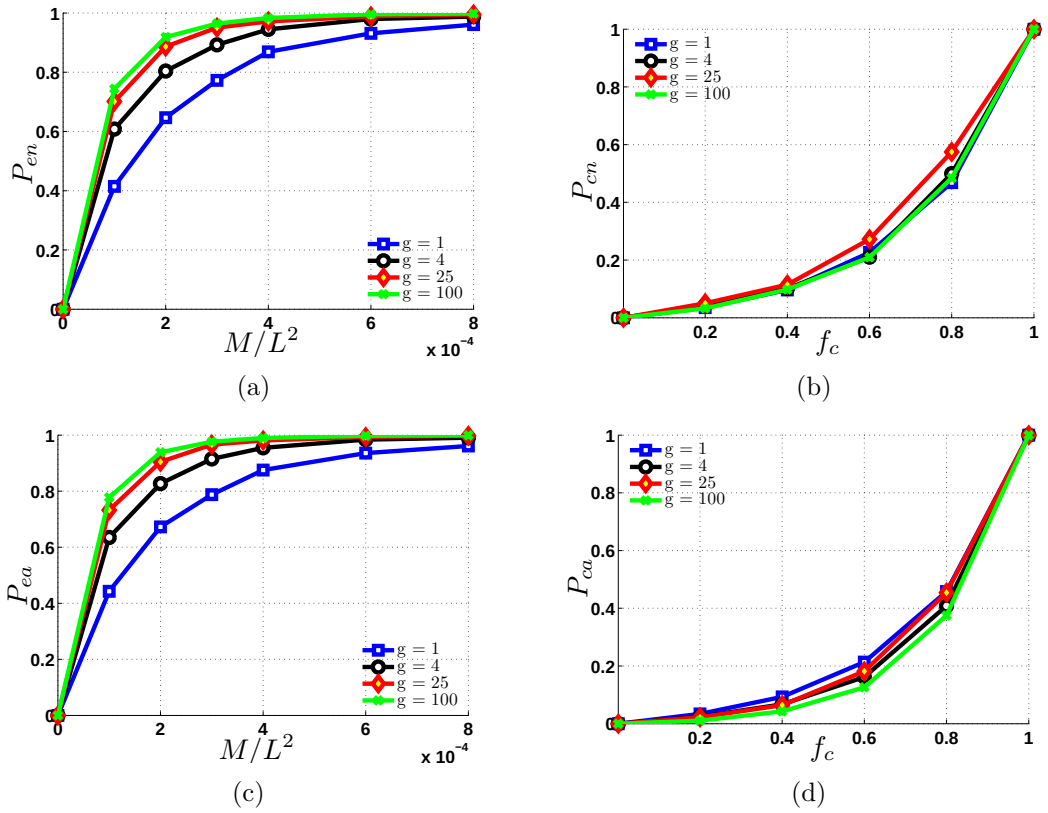


Figure 3.15. The performance under group-based deployment model (4), in which the expected group center locations are randomly distributed in the field. Assume $r = 40$, and $L = 1000$, and $\frac{m}{L^2} = 0.0005$ in (c) and (d).

3.4 Implementation

Based on our overhead analysis, we can see that our proposed pairwise key establishment approach is efficient for resource-constrained sensor nodes. In this section, we will describe some implementation issues.

We have implemented the prototype protocol for the proposed scheme using TinyOS platform [53]. We use RC5 module [54] to implement the security primitives such as hash and MAC operations, assuming 8-byte long hash values and keys. The protocol is designed to be as transparent as possible to the applications. In the protocol, the regular sensor node will first send request messages to its neighboring

assisting nodes and wait for their responses. An assisting node generates the random key and send the reply message to every requesting sensor node. After the regular sensor node collects the random keys, it will combine these keys to derive the final pairwise key.

Our scheme has been tested on the TelosB [20] motes. For the assisting nodes, the additional code space is 2598 bytes in the ROM, and the extra usage of data space in the RAM is 892 bytes. We also make use of the 1M flash memory on the chip to store the hash values of the regular nodes' master keys. For the regular nodes that need to setup secure communication links with 50 neighbors, the additional code space in the ROM is 2102 bytes, and the extra data space in the RAM is 682 bytes. Clearly, the proposed scheme is practical for sensor networks in terms of the code size.

3.5 Summary

In this chapter, we developed a novel scheme to establish pairwise keys in sensor networks. This scheme takes advantage of special nodes (the assisting nodes) in the network for key management, exploring a new dimension of using sensor nodes. The analysis indicates that our scheme has significant advantage over existing approaches. By making use of these cheap assisting nodes, we reduce the burden of regular sensor nodes and further extend the lifetime of the whole network.

In this work, we didn't test our scheme in a real sensor network that consists of thousands or millions of sensor nodes. It is interesting to study how our technique performs in such a large sensor network. In addition, we are also interested in addressing questions like how to tolerate the communication error and delay in bad channel conditions and how to withstand the large deployment error that causes a lot of multi-hop communication. We also note that the additional assisting nodes are

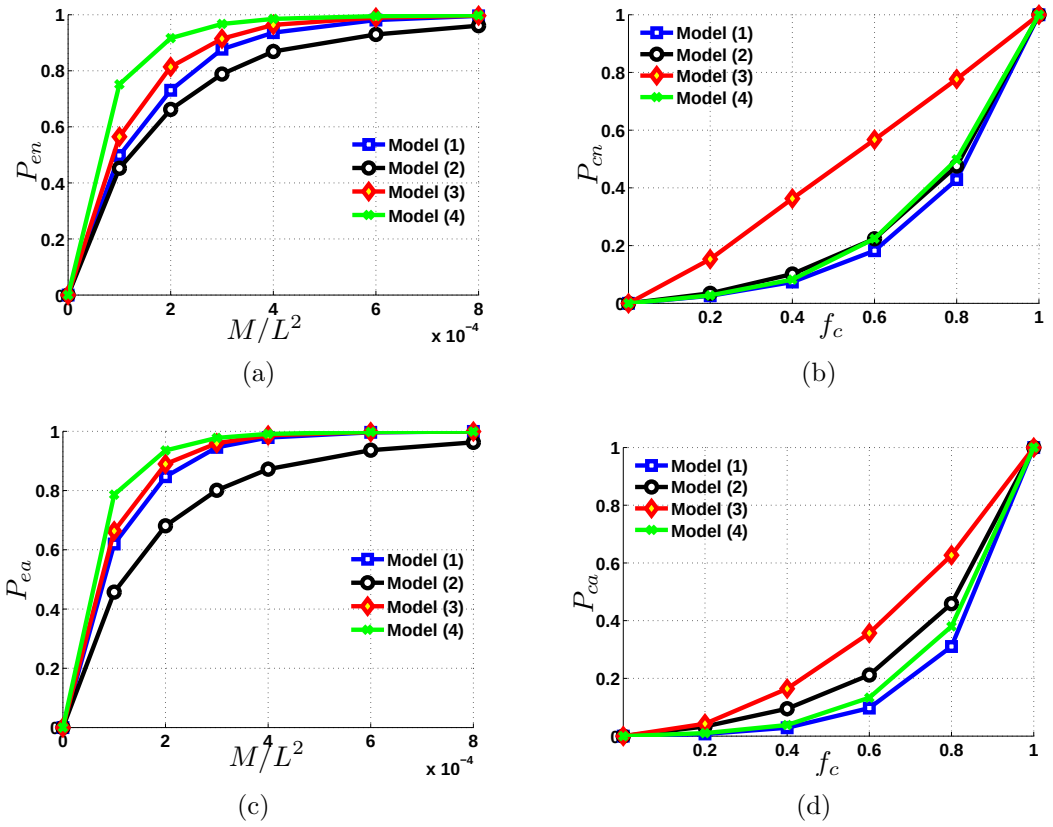


Figure 3.16. The performance under different deployment models illustrated Fig. 3.10. Assume $r = 40$, $L = 1000$, and $\frac{m}{L^2} = 0.0005$ in (c) and (d). For model (2), $\sigma = 333.3$; for model (3) and (4), $g = 100$ and $\sigma = 33.3$.

only deployed to help establish the pairwise keys. It is also interesting to make further use of these nodes to defend the network against attacks. Finally, in our scheme, each assisting node has to store a hash key for each sensor node deployed in the field. For a super large-scale sensor network that consists of millions or even billions of sensor nodes, the storage overhead at assisting nodes will be a big problem for our protocol. We are interesting in ideas to address this problem in the future.

CHAPTER 4

PRE-AUTHENTICATION FILTERS: PROVIDING DOS RESISTANCE FOR SIGNATURE-BASED BROADCAST AUTHENTICATION IN SENSOR NETWORKS

The significant resource consumption imposed by public key cryptographic operations makes such mechanisms easy targets of denial of service (DoS) attacks. If digital signatures such as ECDSA [13] are used directly for broadcast authentication without further protection, an attacker can simply inject forged packets and force the receiving nodes to perform a large number of unnecessary signature verifications, eventually exhausting their battery power. This chapter studies how to effectively mitigate such DoS attacks when signatures are used for broadcast authentication in sensor networks.

We propose to use *pre-authentication filters* to remove bogus messages before verifying the actual digital signatures. Our methods take advantage of the fact that broadcast in sensor networks is usually done by a network-wide flooding protocol [55,56], and a flooding message from a sensor node only has a small number of receivers due to the low-power, short-range radio. Therefore, we focus on the situation where *a sensor node (local sender) needs to re-broadcast a digitally signed message to its neighbors (local receivers)*, and filter out forged messages in a "hop-by-hop" manner.

The proposed filtering techniques is based on the fact that a symmetric key cryptographic operation is much faster than a public key cryptographic operation. Our schemes let the local sender share some keys with its neighboring receivers. Before re-broadcasting a message, the sender uses those shared keys to generate some

commitment values (based symmetric key cryptography), and adds those commitment values to this message. After receiving the re-broadcasted messages, the receivers will quickly verify the corresponding commitment values first. If the commitment values are invalid, the receivers will identify the message is forged, and will simply ignore it without performing the heavy digital signature verification.

The design of pre-authentication filters essentially focuses on how to manage the keys between the sender and its receivers and how to generate the commitment values based on the keys. In this chapter, we develop two filtering techniques, a *group-based filter* and a *key chain-based filter*; the former manages the receivers into groups and assigns the same key to the same group members, the latter explores the one-way property of the hash key chain. Both methods can significantly reduce the number of unnecessary signature verifications that a sensor node has to perform. The analytical results also show that these two techniques are efficient and effective for resource-constrained sensor networks.

4.1 Pre-Authentication Filters

This section presents the details of two filtering techniques, a *group-based filter* and a *key chain-based filter*, to handle DoS attacks against signature verification. Since both filters are independent from the broadcast protocol, we simply assume a flooding scheme for broadcast and will not discuss how it is achieved and focus on the situation where a sensor node needs to re-broadcast a digitally signed message to its neighbors.

4.1.1 Group-Based Filter

A simple method to filter out forged messages is to authenticate the broadcast message, in a "hop-by-hop" manner, with a (local) group key shared among a (local)

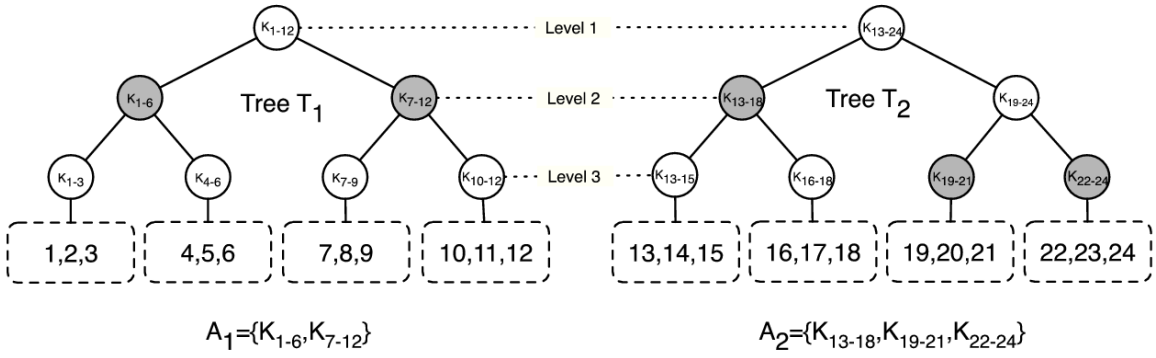


Figure 4.1. Example of key trees ($m = 2$, $L = 2$, and $b = 24$).

sender and its neighbor nodes. As a result, an adversary will not be able to forge messages without compromising the group key. However, sensors could be captured [45], which allows an adversary to forge as many messages as he wants using the group key on the compromised nodes. Alternatively, a sensor node can add a message authentication code (MAC) to a broadcast message for each of its neighbor nodes. However, this incurs large communication overhead even for a moderate neighbor size.

The above two simple ideas represent two extreme cases. One achieves high efficiency (only one MAC for every message), but is vulnerable to a single compromised neighbor; the other achieves high security, but introduces high communication overhead. In this subsection, we present a *group-based* method to trade-off communication efficiency with security. Specifically, every sender divides its neighbor nodes into multiple groups. Every group uses a group key for pre-authentication filtering. When a group key becomes suspicious, we divide the corresponding group into smaller groups to isolate suspicious nodes.

4.1.1.1 Protocol Description

The group-based filter consists of three steps: *initialization*, *broadcasting*, and *re-keying*. In the first step, a (local) sender pre-loads keying materials to its neighbor nodes. In the second step, the sender re-broadcasts authenticated messages. In the last step, the sender re-selects keys dynamically to deal with compromised neighbor nodes.

In this chapter, we assume the existence of a pairwise key establishment protocol in the network. This protocol can help every pair of sensor nodes to setup a secret key to protect their communication. A number of key pre-distribution protocols can be used for our purpose [3–5]. We let $K_{u,v}$ denote the shared pairwise key between nodes u and v. We assume that each node can add m MACs into a broadcast message, where each MAC is q-bit long.

Initialization: For each sender node u, let $N(u)$ be the set of its neighbors. u first divides $N(u)$ into m equal-sized groups, $\{g_1, ..., g_m\}$. We assume each group g_i has s nodes. For each g_i, we further divide it into 2^L equal-sized sub-groups, called *unit groups*, where L is a system parameter that determines the number of bits allocated for a unit group for filtering purposes. (Indeed, it is $\frac{q}{2^L}$ bits). For example, in Figure 4.1, we have two groups $g_1 = \{1, 2, ..., 12\}$ and $g_2 = \{13, 14, ..., 24\}$. Each of them is further split into 4 unit groups with 3 nodes in each unit group.

For each g_i, we construct a full binary tree T_i from the 2^L unit groups with each leaf representing a unit group. Each node in T_i is assigned a unique and random key. The resulting tree is called the *key tree*. (In total, we have m key trees for every sender.) The keying materials for each neighbor in group g_i includes the keys on the path from the corresponding unit group to the root of key tree T_i. For example, in Figure 4.1, node 16 belongs to g_2 and has three keys K_{16-18}, K_{13-18} and K_{13-24}.

Each key k in the j-th level of T_i defines a *level-j group*, which includes all nodes that know this key. Hence, every g_i is a level-1 group, and every unit group is a level $L+1$ group. Let $l(k)$ be the level number of the group defined by key k in its key tree. The key tree T_i is used for filtering the messages sent to the nodes in g_i. We only use a subset A_i of the keys in T_i for filtering. This set, called *active key set*, is initialized as only including the root of T_i, and adjusted (in the third step) based on the suspiciousness of the nodes in g_i. In general, every path from a leaf to the root covers exactly one key in this set. The purpose is to make sure that every node in g_i will be able to find a piece of information in A_i to filter out forged messages. For example, in Figure 4.1, $A_2 = \{K_{13-18}, K_{19-21}, K_{22-24}\}$. This key set is picked because K_{13-24} and K_{19-24} were reported to be suspicious. We can see that every path from a leaf to the root of T_2 includes exactly one key in the A_2.

Broadcasting: Assume node u has received an authenticated, *digitally signed message* M, and needs to re-broadcast it according to the flooding protocol. u will add m *commitment values* to this message, one for each group g_i. The commitment values are generated as follows.

Consider a given group g_i. For every $k \in A_i$, node u computes a *commitment fragment*, which is the first $\frac{q}{2^{l(k)}}$ bits of $H(k||M)$, where H is a one-way hash function and "||" is the concatenation operation. The fragments generated from all the keys in A_i are then concatenated together, producing a complete commitment value for the nodes in g_i. The overall length of this commitment value will be the same as the length of a MAC. For example, in Figure 4.1, $A_2 = \{K_{13-18}, K_{19-21}, K_{22-24}\}$. If $q = 64$, the corresponding commitment value includes the first 32 bits of $H(K_{13-18}||M)$, the first 16 bits of $H(K_{19-21}||M)$, and the first 16 bits of $H(K_{22-24}||M)$. Let W be the set of all the commitment values (there are m of them) generated from all key trees. Node u will simply broadcast $\{M, W\}$.

Suppose a neighbor v in group g_i receives $\{M, W\}$. It will first identify the fragment in W that is generated based on a key k it knows. The information needed for identification can be easily obtained from u, as we will show in the third step. If the fragment is the same as the first $\frac{q}{2^{l(k)}}$ bits of $H(k||M)$, node v will do the actual signature verification; otherwise, it will ignore the message.

When the signature verification succeeds, node v extracts M and returns M to the sensor application for various uses. For example, it may also decide to relay M based on the flooding protocol. When the signature verification fails, node v checks if the number of failed signature verifications exceeds a threshold τ during the last $w = 2^{\frac{q}{2^L}}$ forged messages, i.e., the messages that failed either pre-authentication filtering or signature verification. (We will discuss why we set w in this way in our later analysis.) If so, node v believes that k is suspicious. In this case, if $l(k) < L+1$, node v will stop accepting any message from u and also notify u to adjust the corresponding active key set. If $l(k) = L + 1$, v will stop processing the next w messages from u before returning to normal.

Re-Keying: When the sender u receives a report that the key k of T_i is suspicious, if the level number $l(k) < L + 1$, it will use the two keys corresponding to key k's two child nodes to replace key k in the active key set A_i. In other words, u splits the level $l(k)$ group defined by k into two smaller groups to isolate suspicious nodes. For example, in Figure 4.1, if K_{13-18} is found to be suspicious, we will use K_{13-15} and K_{16-18} to replace K_{13-18} in A_2. Sender u will also notify the affected neighbors about the splitting so that they are able to identify the correct fragments in a broadcast message for pre-authentication filtering.

4.1.1.2 Security Analysis

In our approach, a sensor node will not verify the signature unless the corresponding fragment is valid. When none of the sender u and the nodes in $N(u)$ is compromised, an adversary has to guess the keys on the roots of key trees. Given the hardness of inverting a one-way hash function, we know that it is computationally infeasible for the attacker to bypass pre-authentication filtering. This prevents the adversary from mounting DoS attacks against any of the sensor nodes in $N(u)$.

We now focus on the security when there are compromised sensor nodes. We will study the security of our approach in the following two cases: (1) the sender is benign, and (2) the sender is compromised. After the analysis, we will summarize some important conclusions about the proposed approach.

The group-based approach has many system parameters that need to be configured properly. The configuration of these parameters will be discussed during the analysis.

Security under Benign Sender: Consider a benign node u that receives an authenticated message and needs to re-broadcast it according to the flooding protocol. Assume that the adversary has compromised N_c nodes in $N(u)$. We note that if a level $L+1$ key is compromised, the adversary can forge messages with correct fragments and disable the broadcast authentication at the sensor nodes who share such key. We are thus interested in *how many benign neighbors will be affected by compromised neighbors*. That is, the number of benign neighbors that from time to time stop processing messages relayed by sender u.

Note that the most effective way of attacking our protocol is to make sure that the compromised nodes belong to different unit (level $L+1$) groups since a single

malicious node in a unit group can disable the broadcast authentication of the nodes in this group. Hence, the number of benign nodes affected is no more than

$$N_c \times (\frac{s}{2^L} - 1) \tag{4.1}$$

For example, when $L = 2$ and $s = 8$, the attacker can only affect no more than N_c benign neighbors. In addition, the attacker has to continuously launch such attack since otherwise these affected nodes will return to normal situation after w messages from u. We also note that the number of affected benign nodes can be reduced by having a smaller s or larger L. However, having a small s leads to a larger m, and therefore more communication overhead. Having a larger L will reduce the security of our approach since the adversary is able to force nodes to do more unnecessary verifications, as we will see next.

When there are compromised neighbor nodes, it is possible that some non-compromised level $L+1$ keys are used for pre-authentication filtering. When a benign node v is using a non-compromised level $L+1$ key for pre-authentication filtering, we are interested in *the probability of an adversary bypassing pre-authentication filtering.* Since the length of a commitment fragment for a level $L + 1$ key is $\frac{q}{2^L}$-bit long, this probability can be estimated by

$$p_f = \frac{1}{2^{\frac{q}{2^L}}} \tag{4.2}$$

Equation 4.2 shows that a larger L may result in more unnecessary signature verifications at those benign nodes. For example, suppose $q = 64$, when $L = 3$, we have $p_f = \frac{1}{256} \approx 0.0039$; when $L = 4$, we have $p_f = 0.0625$. Together with Equation 4.1, we may want to find the largest value L that meets the required p_f.

When v uses a non-compromised level $L + 1$ key for pre-authentication filtering, we are also interested in *how likely v will stop processing messages from u when*

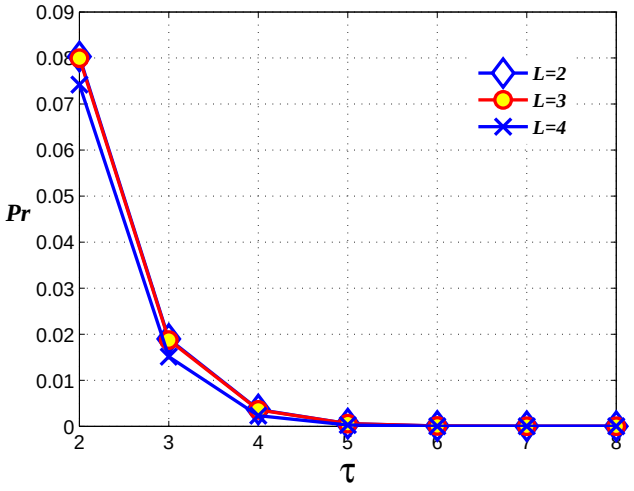


Figure 4.2. Probability p_r v.s. τ. Assume $q = 64$.

the adversary forges messages. This is equivalent to the probability p_r of the adversary bypassing pre-authentication filtering τ times in a window of $w = 2^{\frac{q}{2L}}$ forged messages. This probability can be estimated by

$$p_r = 1 - \sum_{i=0}^{\tau} \binom{w}{i} (\frac{1}{w})^i (\frac{w-1}{w})^{w-i} \tag{4.3}$$

Figure 4.2 shows that p_r decreases quickly with τ. Indeed, a small τ will make it very difficult for an attacker to disable the broadcast authentication at any benign node that has a non-compromised level $L + 1$ key. We also note that the value of L does not affect such probability much, making it much easier for us to configure threshold τ. This also explains our configuration of w in the protocol description.

There is a small probability that an adversary can disable the pre-authentication filter at a node with a non-compromised level $L + 1$ key. However, this node will continue the protocol after w messages from the sender, which makes the attacker's job much harder. We can thus conclude that the adversary cannot gain much benefit by attacking those having non-compromised level $L + 1$ keys.

Security under Compromised Sender: When the sender u is compromised, it can certainly forge broadcast messages with correct commitment values. We consider the worst case scenario where the sender u keeps forging broadcast messages but the number of failed verifications in any window of $2^{\frac{q}{2^L}}$ forged messages is always no more than τ. In this case, obviously, all the neighbor nodes will keep processing the messages from node u and verifying the corresponding signatures. Fortunately, due to threshold τ, we can clearly see that the fraction p_f of forged messages that will lead to unnecessary signature verifications is always bounded by

$$p_f = \frac{\tau}{2^{\frac{q}{2^L}}} \tag{4.4}$$

For example, when $q = 64$, $L = 3$ and $\tau = 3$, we have $p_f = 0.012$. We can see that our group-based approach can effectively mitigate the DoS attacks launched by a compromised sender given a reasonable setting of parameters.

Summary: First of all, whenever a benign node receives a true broadcast message from a benign sender, it will never miss this true message as long as the pre-authentication filter is not disabled due to a compromised neighbor node in the same level $L + 1$ group. The impact of compromised nodes is given by Equation 4.1. For the forged messages comes from an adversary who may compromised the sender and some neighbor nodes, we have the following theorem.

Theorem 1. *The fraction of forged messages that can lead to unnecessary signature verifications at a benign node is no more than*

$$\frac{(1 - f_c)^{\frac{s}{2^L}-1} + \tau \times [1 - (1 - f_c)^{\frac{s}{2^L}-1}]}{2^{\frac{q}{2^L}}},$$

where f_c is the fraction of compromised sensor nodes.

Proof. Note that the size of a level $L + 1$ group is $\frac{s}{2^L}$. Hence, for a given neighbor node, the probability that the sender and all other nodes in the same level $L + 1$

group are benign can be estimated by $(1-f_c)^{\frac{s}{2^L}-1}$. In this case, the fraction of forged messages that will lead to unnecessary signature verifications is given by Equation 4.2. When either the sender or one of the nodes in the same level $L+1$ group is compromised, the fraction of forged messages that will lead to unnecessary signature verifications is given by Equation 4.4. Overall, the fraction of forged messages that will lead to unnecessary signature verifications at a benign node is no more than

$$\frac{(1-f_c)^{\frac{s}{2^L}-1}+\tau\times[1-(1-f_c)^{\frac{s}{2^L}-1}]}{2^{\frac{q}{2^L}}}.$$

□

Theorem 1 indicates the security of our group-based approach in dealing with the DoS attacks against signature verification. Since L, s and τ are usually quite small, we can see that our group-based filter can effectively defeat the DoS attacks against signature verification. For example, when $L=3, s=16, \tau=3$, $q=64$ and $f_c=0.1$, the fraction of forged messages leading to unnecessary verifications is only about 0.0047.

4.1.1.3 Overheads

According to the protocol description, every sensor node needs to store a set of key trees (as a sender), and $L+1$ keys for every neighbor (as a receiver). Since the keys on each key tree can be derived from a random seed, they only need one master key. Hence, the storage overhead can be estimated by $b\times(L+1)\times q$ bits, where b is the average neighbor size in the network.

The communication overhead introduced by our protocol comes from three parts: (1) the distribution of keying materials in the initialization, (2) the reports from every neighbor node indicating unsafe keys and the notifications to neighbor nodes indicating the change of active key sets, and (3) the space needed for the com-

mitment values in broadcast messages. The distribution of keying materials only happens during the initialization; it only involves the delivery of $L+1$ keys for every neighbor nodes and only needs to be done once for each neighbor. For the reports and notifications, we note that they are done between neighbors and the number of messages is usually limited. Hence, the main communication overhead introduced is the additional m MACs on every broadcast message.

We use MICAz motes to show the energy consumption for the additional MACs. With the 250kbps data rate and 3.0V power level, MICAz will keep the current draw at no more than 17.4mA in TX mode and 19.7mA in RX mode [25]. Assume $m = 8$ and $q = 64$. The energy cost of sending the additional MACs can be estimated as $3.0 \times 17.4 \times 64 \times 8/250,000 = 0.107$mJ, and the energy cost of receiving can be estimated as $3.0 \times 19.7 \times 64 \times 8/250,000 = 0.121$mJ. On the other hand, the current draw of an active MICAz CPU is 8mA [25]. As discussed before, verifying an ECDSA signature takes about 1.96s on MICAz motes. Thus, the energy cost of a signature verification can be estimated as $3.0 \times 8 \times 1.96 = 47.04$mJ, which is about 400 times more than the energy cost of sending or receiving the additional MACs. This clearly explains the benefit of our filtering method.

According to the protocol description, each sender needs to do up to $m \times 2^L$ additional hash operations on average. For each receiver, it needs to perform one additional hash operation to verify the corresponding fragment.

4.1.1.4 Extension: Adaptive Re-Grouping

Our previous analysis indicates that we usually prefer a small L to make sure that an adversary cannot fool sensor nodes to perform many unnecessary signature verifications. However, we also mentioned that a smaller L will make it possible for

a single compromised sensor node to disable the broadcast authentication at more benign nodes.

In the following, we present an *adaptive re-grouping* extension to deal with compromised neighbor nodes and provide more flexibility in configuring parameter L. The basic observation is that benign nodes will always report suspicious keys in the key trees to the sender. This provides evidence for the sender to reason about the suspiciousness of neighbor nodes. The sender can then rank and group neighbor nodes according to the suspiciousness, making sure that *a benign neighbor is likely to be in the same group with other benign neighbors and a compromised neighbor is likely to be in the same group with other compromised neighbors.* Achieving this will makes it much more difficult for an adversary to impact benign sensor nodes.

Adaptive Re-Grouping: We focus on level $L+1$ keys. When a level $L+1$ key is found to be unsafe, all nodes having this key are suspicious. We consider them as equally suspicious. A sensor node will send a report to the sender when its level $L+1$ key is found to be unsafe. The adaptive re-grouping protocol works as follows.

The sender u maintains a boolean variable $c(i)$ for every neighbor node $i \in N(u)$, indicating if this neighbor node is suspicious. Initially, we have $c(i) = FALSE$ for every $i \in N(u)$. Whenever the sender u receives a report from its neighbor node i saying that its level $L+1$ key is suspicious, node u will set $c(j) = TRUE$ for every neighbor node j in the same level $L+1$ group as node i.

The sender u periodically re-groups neighbor nodes. During each round of re-grouping, node u first classifies all current level $L+1$ groups into two categories: the first category includes those with non-suspicious level $L+1$ keys, and the second category includes those with suspicious level $L+1$ keys. This classification can be done using the boolean variables $\{c(i)\}_{i \in N(u)}$. The sensor nodes in the second category are then re-grouped randomly, and we expect that some of these randomly organized

level $L+1$ groups only include benign nodes so that they will be identified as benign with a very high probability in the next round. In this way, we will build more and more benign level $L+1$ groups.

After re-grouping, sender u then re-generates random keys for every key tree. After the assignment of new keys, we will have new key trees for the new group construction. The sender will then distribute new keying materials to neighbor nodes and reset all the boolean values $\{c(i)\}_{i\in N(u)}$ to FALSE.

Advantages: The total number of level $L+1$ groups can be estimated by $G = m \times 2^L$. Consider the moment right before the $i+1$-th re-grouping at sender u. Let m_i be the number of level $L+1$ groups that include only benign neighbors. For each of the groups, the probability of being marked as suspicious after the $i+1$-th re-grouping can be estimated by Equation 4.3. Thus, on average, these m_i groups in the first category will contribute $m_i(1-p_r)$ to m_{i+1}.

During the re-grouping, the sender will believe that there are up to $(G-m_i(1-p_r))$ suspicious level $L+1$ groups. Assume there are N_c compromised neighbors. After the $i+1$-th re-grouping, the average number of level $L+1$ groups in the second category that include only benign nodes can be estimated by

$$(G-m_i(1-p_r))(\frac{G-m_i(1-p_r)-1}{G-m_i(1-p_r)})^{N_c}.$$

Overall, after the $i+1$-th re-grouping, the average number of level $L+1$ groups that include only benign sensor nodes (m_{i+1}) can be estimated by

$$m_i(1-p_r)+(G-m_i(1-p_r))(\frac{G-m_i(1-p_r)-1}{G-m_i(1-p_r)})^{N_c}.$$

Figure 4.3 shows the performance of our re-grouping approach. We consider the worst scenario where the compromised nodes belong to different level $L+1$ groups. Hence, $m_0 = G - N_c = m \times 2^L - N_c$. Clearly, without re-grouping, the number of

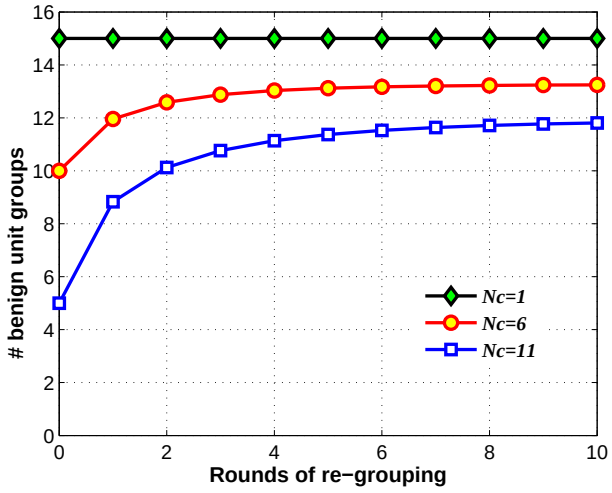


Figure 4.3. Performance of adaptive re-grouping ($L = 3$, $\tau = 3$ and $m = 2$).

affected groups is N_c. With re-grouping, we can clearly see from the figure that we significantly increase the average number of unaffected level $L + 1$ groups.

Overheads: The proposed re-grouping idea does not add much overhead to sensor nodes. Indeed, for every neighbor node, it only needs to report at most one extra message and get re-initialized after a certain period of time. There are no other additional overheads for them. For the sender, it needs to collect at most one report from each neighbor node and re-initialize the keying materials after a certain period of time. In addition, it needs to maintain a boolean value for each level $L + 1$ group, re-organize category-2 groups, and re-generate all keying materials. These additional steps will not add too much cost for the sender.

4.1.2 Key Chain-Based Filter

Due to the usage of group keys in the previous method, compromising a neighbor allows an attacker to bypass the pre-authentication filter at many benign nodes and fool them to do unnecessary verifications. This is one major issue for the previous

method. In the following, we give another pre-authentication filter that doesn't offer an adversary much benefit by compromising neighbors.

Our main idea is to introduce an *asymmetry* property so that a receiver (neighbor) can filter but not forge messages. One-way key chains meet such requirement. A one-way key chain $\{K_1, ..., K_n\}$ has the following property: from any K_j, all former keys can be derived by iteratively hashing this key ($K_i = H^{j-i}(K_j), 0 \leq i < j$), but none of the later keys can be computed. The hash image of the first key K_1 is called the *commitment* of this key chain. With the commitment, anybody can verify the authenticity of any later key by only performing hash operations.

With one-way key chains, a simple pre-authentication filter (used by LHAP [57]) works as follows: the sender always adds the next unreleased key in the key chain into a new broadcast message. When a neighbor receives any message, it will perform the actual signature verification only when the chained key included in the message is found to be new to this neighbor. A long key chain is often used to support network operations continuously. However, an adversary may claim a key close to the end of key chain and fool a node to perform a large number of unnecessary hash operations, causing a DoS attack.

We apply a *two-layer filter* to deal with the DoS attacks on the verification of signatures and chained keys. The first layer employs a one-way key chain to filter out fake signatures, and the second layer uses existing pairwise keys to prevent a node from doing a significant number of unnecessary hash operations. Our approach is different from LHAP in that we have an additional layer of protection to stop attacks on the verification of chained keys.

The main intuition of the second layer is to *control the number of hash operations used for verifying a chained key.* A receiver (neighbor) will verify a chained key only when either the verification costs only a few hash operations or there is additional

evidence (i.e., a *commitment value* generated from the shared key with the sender) about the authenticity of the message. Our design guarantees that a receiver can always find additional evidence in a small number of consecutive messages. Hence, when a receiver realizes that it has to do many hash operations to verify the key, it chooses to wait for additional evidence before doing the verification. This significantly enhances the security against the DoS attacks on the verification of chained keys since the attacker has to guess the pairwise key and then forge the evidence. After a successful verification, this receiver can immediately catch up with the sender since it knows the most recently released key in the key chain.

4.1.2.1 Protocol Description

The key chain-based filter consists of three steps, *initialization*, *broadcasting*, and *re-keying*. We assume that every node can add m MAC values and a sequence number into every broadcast message, where each MAC value is q-bit long.

Initialization: During the initialization, each node u first discovers a set $N(u)$ of neighbors and generates a one-way key chain of length n: $\{K_1, \cdots, K_n\}$. It then distributes the key chain commitment $K_0 = H(K_1)$ to every v in $N(u)$ via a secure channel established by the pairwise key $K_{u,v}$.

Sender u also maintains a variable idx to record the index of the next key in the key chain. Initially, we have $idx = 1$. This variable decides which authentication key in the key chain to use. In addition, u also organizes its neighbors into disjoint groups and picks one group for use in every round of broadcast authentication in *a round-robin manner*. These groups are used in the second layer of pre-authentication filter. For convenience, we call them *layer-2 authentication groups*. The value of idx directly determines which group to use for each broadcast message.

When a layer-2 authentication group is picked for a broadcast message, every node in this group can find additional evidence (a commitment value) in the message to do the filtering. Let l be the length of a commitment value. Since the first layer filtering needs space for one key, we have $q(m-1)$ bits left for additional values. Hence, we can add $\frac{q(m-1)}{l}$ commitment values in a broadcast message. The size of an authentication group is $\frac{q(m-1)}{l}$ (the last group may have fewer nodes).

Broadcasting: Assume node u has received an authenticated, digitally signed message M and needs to re-broadcast it according to the flooding protocol. Node u will first control the broadcast rate. Specifically, it will ensure that there will be no more than one true broadcast message from itself per T seconds on average, where T is a system parameter that determines the maximum frequency of relaying digitally-signed broadcast messages.

After rate-controlling, node u will get the next key K_{idx} from the key chain and compute a set W of commitment values. To generate W, node u will first select the next layer-2 authentication group based on index idx. For every node v in this group, node u uses the first l bits of $H(M||idx||K_{idx}||K_{u,v})$ as the commitment value. The set W includes all commitment values generated in this way. Node u will then broadcast $\{M, idx, K_{idx}, W\}$ and increment the variable idx by one.

Figure 4.4 shows an example of re-broadcasting an authenticated, digitally-signed message M. In the example, node u has seven neighbors $\{1, ..., 7\}$, which are organized into three layer-2 authentication groups, $\{1, 2, 3\}$, $\{4, 5, 6\}$ and $\{7\}$. Node u first adds i and K_i to the message, and then computes and adds the commitment values for the second layer-2 authentication group $\{4, 5, 6\}$ to the message. The final broadcast packet includes M, i, K_i, and three commitment values $\{X_4, X_5, X_6\}$.

Suppose a neighbor node v receives $\{M, idx, K_{idx}, W\}$ from sender u. It first checks if W includes the commitment value generated from the key $K_{v,u}$ shared with

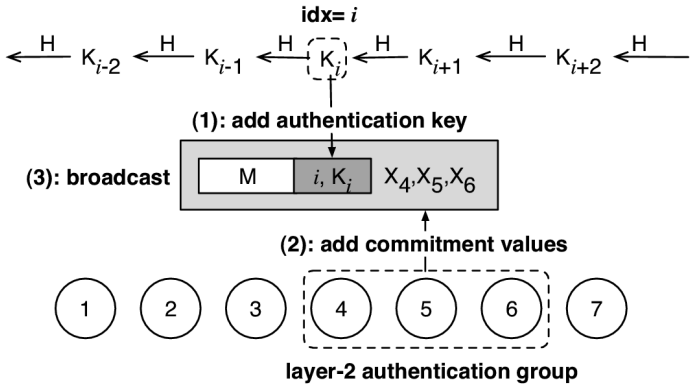


Figure 4.4. Example of the key chain-based approach. X_j is the first l bits of $H(M||i||K_i||K_{u,j})$.

sender u. This checking can be easily done with the value idx included in the message, since the sender u always picks a layer-2 authentication group for use in a round-robin manner. Let K_i be the chained authentication key that node v stores in its memory. The result of this checking will lead to the following two cases.

- If W does not include the commitment value generated from the key $K_{v,u}$, node v will check whether $0 < idx - i \leq B$, where B is a threshold value that determines how many hash operations a node is willing to pay for verifying a chained key. If not, node v will ignore the message.
- If W does include the commitment value generated from the key $K_{v,u}$, node v will first check whether the included commitment value equals the first l bits of $H(M||idx||K_{idx}||K_{v,u})$. If not, node v will ignore the message; otherwise, it will further check whether $idx - i > 0$ (for *freshness*). If not, node v will ignore the message.

Once the message has passed the above checking, node v will verify the key K_{idx} using K_i. If such verification fails, node v will check whether the verification of authentication keys has failed more than τ times during the last 2^l forged messages

that include the commitment value for node v. If yes, node v will stop processing the next 2^l messages from node u before returning to normal.

When the verification of the chained key K_{idx} succeeds, node v also checks whether the sender discloses chained keys at a rate of no more than one key per T seconds. If not, node v considers sender u as compromised and then removes u from its neighbor set $N(v)$. Otherwise, node v will perform the actual signature verification and use the new key K_{idx} to replace K_i. When the signature verification succeeds, node v extracts M and returns M to the sensor application for various uses. For example, it may also decide to relay M based on the flooding protocol.

Re-Keying: After every round of re-broadcast, node u will check whether $idx >= n$. If so, node u will generate a new key chain and distribute the initial commitment of the new key chain to every neighbor node. In addition, it will also reset idx to 1. When a neighbor node receives the updated key chain commitment, it will set its copy of chained keys to this commitment for future authentication.

4.1.2.2 Security Analysis

In the key chain-based method, sensor nodes only verify signatures in those messages that have passed our two-layer filtering. Similar to the security analysis for the group-based scheme, we will evaluate the security of the key chain-based scheme in two cases: (1) the sender is benign, and (2) the sender has been compromised. After security analysis, we will also summarize some important conclusions.

The key chain-based approach has many system parameters that need to be configured properly. The configuration of these parameters will be discussed during the analysis.

Security under Benign Sender: When the sender is benign, we first consider the first layer of pre-authentication filtering, which is achieved by an one-way key chain. This layer of pre-authentication filtering guarantees the following property.

Lemma 1. *Given a benign sender, the number of unnecessary signature verifications a benign receiver performs will not exceed the number of true broadcast messages.*

Proof. The freshness requirement of the chained key guarantees that nobody can forge broadcast messages using undisclosed keys in the key chain. Thus, no matter how many neighbor nodes are compromised, the total number of failed signature verifications that a benign neighbor performs will never exceed the total number of authentication keys in the key chain. Since the sender is benign, the total number of unnecessary signature verifications a benign neighbor node needs to do will never exceed the total number of true broadcast messages from the sender. □

Lemma 1 clearly indicates that an adversary is not able to convince any benign sensor node to do a significant number of unnecessary signature verifications. However, as mentioned before, an adversary can fool a sensor node to do a large number of unnecessary hash operations, causing a DoS attack. In the following, we will study how our second layer of pre-authentication filtering addresses this problem.

The second layer of pre-authentication filtering at sender u appends commitment values for $\frac{q(m-1)}{l}$ neighbors in each broadcast message. When a neighbor v receives a forged broadcast message that does not include the commitment for node v, it will perform no more than B hash operations in verifying the chained key included in the message.

When the broadcast message does include the commitment value for node v, the probability that the adversary can successfully generate the correct commitment value is $\frac{1}{2^l}$. Node v will need to do one hash operation for an incorrect commitment

value and up to $n+1$ hash operations for a correct commitment value (one for verifying the commitment value and n for verifying the chained key included in the message). Thus, for those forged broadcast messages that include the commitment values for node v, the average number of additional hash operations that node v has to do will not exceed $1 + \frac{n}{2^l}$. This tells us that we can simply set $n = (B-1) \times 2^l$ to make sure that a benign neighbor node will not perform more than B additional hash functions for each forged message on average. For example, when $l = 8$ and $B = 6$, we can set $n = 1,280$. As a result, in this chapter, we always set

$$n = (B-1) \times 2^l \tag{4.5}$$

Equation 4.5 also shows how to configure parameters B, l and n. Parameter B usually needs to be large enough to make sure that the non-malicious message loss will not impact the protocol. In other words, a sensor node should be able to receive at least one of any B consecutive messages from a sender with a very high probability. We then set parameter n based on the storage and computation costs that a sender is willing to pay. For example, when $n = 900$, a sender can allocate space for 30 keys to save one key per every 30 keys in the key chain, $\{K_{30}, K_{60}..., K_{900}\}$, and allocate space for 30 keys to save the keys for immediate use, $\{K_{30j+1}, K_{30j+2}, ..., K_{30j+10}\}$. As a result, the sender will need to first generate the whole key chain (900 keys) and then generate the other 870 keys again during the pre-authentication filtering. Overall, the sender will allocate space for 60 keys for the key chain, and perform no more than two hash operations on average to find a key to use. In general, the storage overhead to save a key chain of length n is $2\sqrt{n}$. After configuring B and n, we can easily set l.

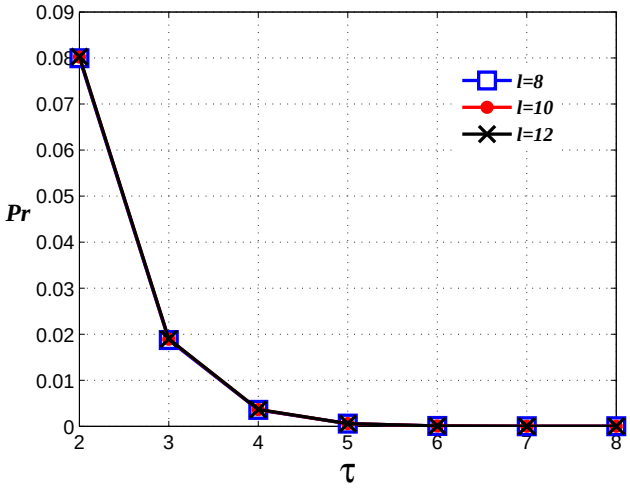


Figure 4.5. Probability p_r v.s. threshold τ.

We also note that when the sender is benign, the adversary still has a small chance to bypass the second layer of pre-authentication filtering by randomly guessing the commitment value. If the number of messages that passed the layer-2 filtering but failed the layer-1 filtering is larger than τ during the last 2^l forged messages that contain the commitment value for a given sensor node, this node will stop accepting messages from the sender. We are thus interested in the probability p_r that *a benign sensor node stops accepting any broadcast message from a benign sender*. Since the probability of the adversary making a correct guess of the commitment value is about 2^{-l}, the probability p_r can be estimated by

$$p_r = 1 - \sum_{i=0}^{\tau} \binom{2^l}{i} (\frac{1}{2^l})^i (1 - \frac{1}{2^l})^{2^l - i} \tag{4.6}$$

Figure 4.5 shows that a small τ is sufficient to make p_r very low. Also note that l does not affect p_r much. We can thus easily configure τ to tolerate forged messages.

Security under Compromised Sender: If the sender u is compromised, it can always forge broadcast messages with correct chained keys and commitment

values. However, our approach can still effectively prevent a compromised sender from launching DoS attacks against signature verification by controlling the broadcast rate.

Lemma 2. *In case of a compromised sender, the average number of unnecessary signature verifications a benign node needs to do will never exceed one per T seconds.*

Proof. The proof of Lemma 1 indicates that the number of unnecessary signature verifications a benign neighbor needs to do will never exceed the number of authentication keys in the key chain. We also note that our approach controls the rate of disclosing chained keys to one per T seconds. Hence, the average number of unnecessary signature verifications a benign node needs to do will never exceed one per T seconds even if the sender is compromised. □

Lemma 2 shows that a compromised sender is not able to convince any benign sensor node to do a significant number of unnecessary signature verifications. However, a compromised sender can fool a sensor node to perform a significant number of unnecessary hash functions for a forged message. Similarly, we will also study how our second layer of filtering addresses this problem.

Similar to the case of a benign sender, when a sensor node receives a forged broadcast message (from a compromised sender) that does not include the commitment value for itself, it will perform no more than B hash operations in verifying the chained key included in the message.

However, when the broadcast message does include the commitment value for the receiver, the adversary can always include a correct commitment since the compromised sender knows all the keys. Fortunately, due to the threshold τ, the compromised sender cannot force a benign node to do more than $n \times \tau$ additional hash operations in a window of 2^l forged messages that include the commitment of this node. The reason is that once the number of messages that failed the first layer fil-

tering exceeds the threshold τ, the neighbor node will stop accepting messages from the compromised sender. Since n is set to $(B-1) \times 2^l$ (Equation 4.5), the average number of additional hash operations a benign neighbor needs to do will not exceed

$$\frac{n \times \tau}{2^l} + 1 = (B-1) \times \tau + 1 \tag{4.7}$$

Summary: First of all, whenever a benign sensor node receives a true broadcast message from a benign sender, it has a very high chance of being able to authenticate this true message for a reasonable B. For the forged messages, we have the following two theorems.

Theorem 2. *On average, a benign sensor node will not do more than one unnecessary signature verifications in every T seconds for a given sender.*

The above theorem can be easily derived by generalizing Lemma 1 and Lemma 2. The detail of the proof will be thus skipped for simplicity. This theorem indicates the security performance of our key chain-based approach in dealing with the DoS attacks on signature verification.

Theorem 3. *The average number of additional hash operations a benign sensor node needs to do is no more than $B(1-f_c) + [(B-1)\tau + 1]f_c$, where f_c is the fraction of compromised sensor nodes.*

Proof. Note that when the sender is benign, the average number of additional hash operations will not exceed B. When the sender is compromised, the average number of additional hash operations will not exceed $(B-1)\tau + 1$. Since the probability of any given sensor node being compromised is f_c, the average number of additional hash operations a benign sensor node needs to do will not exceed $B(1-f_c) + [(B-1)\tau + 1]f_c$. □

Theorem 3 indicates the security performance of our key chain-based filter in dealing with the DoS attacks on the verification of chained keys. Since B and τ are usually quite small, we can see that our key chain-based filter can effectively defeat this kind of DoS attacks.

4.1.2.3 Overheads

From the protocol description, any sensor node u needs to store a one-way key chain (as a sender) and a key chain commitment for every neighbor sender (as a receiver). Hence, the overall storage overhead can be estimated by $(b+2\sqrt{n}) \times q$ bits, where n is length of the key chain and b is the average neighbor size.

The communication overhead includes two parts: (1) the distribution of the key chain commitment, and (2) the space for the chained key and the commitment values in every broadcast message. Note that each key chain can be used for n broadcast messages. We thus believe that the communication overhead needed for distributing the initial key chain commitment will not be a problem given a reasonably long key chain. In addition, each broadcast message will include $m \times q$ bits additional information (i.e., the chained key and the commitment values).

For every broadcast message, a sender u will need to perform $|W|$ hash operations for generating the commitment values and an average of 2 hash operations for generating a key in the key chain to use. On the receiver side, Theorem 3 shows the average number of additional hash operations a benign sensor node needs to do. Overall, we can see that the key chain-based method is also effective and efficient.

4.1.3 Discussion

The detailed analysis on the security against DoS attacks and the overheads of the proposed two methods is given in the previous two subsections. In the following,

we will discuss the security of these two schemes under some other common attacks and compare the proposed two methods in terms of security and overheads.

4.1.3.1 Security under Other Attacks

In the following, we will discuss the security of our two methods in the presence of *sybil* attacks [46], *wormhole* attacks [47], and *node replication* attacks [48].

Sybil Attacks: In sybil attacks, an adversary tries to clone sensor nodes with different IDs. However, some key pre-distribution techniques (e.g., [5]) can effectively remove the impact of sybil attacks since the keying materials for pairwise key establishment is bound with the node ID. When any of these key pre-distribution techniques is employed, we will be free from sybil attacks.

Wormhole Attacks: Wormhole attacks do impact the security of our proposed approaches since they can increase the size of neighbor nodes for a sensor node. However, such impact is very limited since a significant increase in the number of neighbor nodes usually indicates a wormhole attack. In this case, we can simply pick a subset of them for use in our protocol since as long as a sensor node can deliver its message to a sufficient number of neighbors, the broadcast protocol will work correctly.

Node Replication Attacks: By launching node replication attacks, the adversary can increase the fraction of compromised nodes in a local area. According to Theorem 1, we can see that this attack does impact our group-based approach. However, as we discussed, we can always configure parameters such as τ properly or re-grouping sensor nodes to mitigate the impact of increasing the fraction of compromised nodes. On the other hand, according to Theorem 2 and Theorem 3, increasing the fraction of compromised nodes does not generate much impact on the security of the key chain-based approach. Moreover, though the probability of finding a ma-

licious sender is high, a node can always switch to other nodes if it notices that a particular sender is suspicious. Therefore, as long as the benign nodes in the network are well-connected, our protocols will work correctly.

4.1.3.2 Comparison

Both pre-authentication filters have advantages and disadvantages when compared with each other. In terms of security, the main difference between them is that the group-based approach allows a compromised neighbor node to disable the broadcast authentication at a number of other benign neighbor nodes even if the sender is benign. In contrast, the key chain-based approach does not have this problem. Thus, the key chain-based method can often perform better than the group-based approach when there is a large fraction of compromised sensor nodes. In addition, according to Theorem 2, we know that the key chain-based approach can have better security when there is a low rate of true broadcast messages from the base station. However, when there is only a small fraction of compromised sensor nodes, the group-based approach can be more secure in the sense that the fraction of fake signatures leading to unnecessary signature verifications can be made very small. In contrast, according to Theorem 2, we know that even without any compromised node, the key chain-based approach allows the adversary to fool sensor nodes to perform many unnecessary signature verifications if there is a high rate of true broadcast messages from the base station.

In terms of overheads, the group-based method performs slightly better than the key chain-based approach. First, the storage overhead of the group-based method ($b \times (L+1)$ keys) is comparable to that of the key chain-based method ($b+2\sqrt{n}$ keys). Second, the communication cost of the group-based method only involves the initial distribution of keying materials and up to $L+1$ reports from every neighbor node

to the sender. In contrast, the key chain-based method has to update the keys at neighbor nodes once for a while. Finally, the group-based approach requires up to $m \times 2^L$ hash operations for a sender and one hash operation for a receiver. In contrast, the key chain-based approach requires $\frac{(m-1)q}{l} + 2$ hash operations for a sender, which is comparable to the group-based method, and $B(1 - f_c) + [(B - 1)\tau + 1]f_c$ hash operations for a receiver, which can be many more than the group-based approach since B has to be configured to accommodate the lossy channel.

4.2 Summary

ECC-based signature schemes have attracted a lot of attention recently for broadcast authentication in sensor networks. However, such approaches are vulnerable to DoS attacks against signature verifications. This chapter shows how the proposed group-based and key chain-based methods effectively mitigate such DoS attacks.

It may be desirable to evaluate the performance of our approaches through field experiments to obtain more useful results such as the energy savings under attacks. In addition, we will also seek efficient solutions in scenarios where a sender's signal range can reach all or most of the sensor nodes in the network.

CHAPTER 5

ADAPTIVE JAMMING-RESISTANT BROADCAST SYSTEMS WITH PARTIAL CHANNELS SHARING

Wireless broadcast systems are particularly vulnerable to jamming attacks launched by *insiders* who know which channel is used for broadcast. The existing solutions [18, 19] involve a lot of communication overhead. In this chapter, we propose a novel jamming-resistant broadcast system, which organizes receivers into multiple *channel-sharing broadcast groups* and isolates malicious receivers using adaptive re-grouping. Instead of sending messages using one channel [18, 19], we divide such channel into multiple smaller ones and let different receivers partially share their channels. In this way, the data sent over the shared channels can reach more than one receivers, saving substantial communication cost.

In addition, we also propose an *sequential test based scheme* to achieve high decision accuracy and decision speed even if the resource is limited or the attacker is very powerful. The basic idea to get more observations about the channel condition before making any decision, so that we can reduce the decision error rate. We adopt Lai's Bayes sequential test scheme [58] to balance the decision accuracy and decision speed, thus we can make accurate decisions as soon as possible.

The analytic and simulation results show that the proposed approaches greatly push the performance limit of jamming-resistant broadcast systems towards optimal.

5.1 Network and Adversary Model

Table 6.1 lists some frequently used notations. In this chapter, we consider that a benign sender needs to broadcast messages to a set of r receivers, $\{R_1, R_2, \cdots, R_r\}$. We assume that the system contains totally n orthogonal *virtual channels*, which are determined by spreading code, frequency-hopping pattern, sub-carriers, or their combinations. The proposed techniques in this chapter can be considered as group-based approaches. In a group-based approach, we organize the receivers into multiple groups and assigns m ($m \ll n$) channels to each group for broadcast communication. The receivers in the same group listen to and receive messages from the m channels assigned to the group. We assume that the sender and the receivers are well time synchronized and every receiver knows when to start and stop the transmission at any given channel.

Let C_G be the set of *active* channels assigned to group G ($|C_G| = m$). Every broadcast message M will be processed according to certain forward error correction (FEC) codes at packet-level [35–37] and divided into m packets such that M can be recovered from any set of $(1 - \eta) \times m + 1$ correct packets. These m packets will be transmitted on the active channels of every group, one for each channel. Thus, a receiver in group G can always recover the original message M unless $\eta \times m$ or more channels in C_G are jammed.

We assume that each receiver shares a secret key with the sender; this key is used to select the secret channels for the jamming-resistant one-to-one communication with the sender. Hence, this *private and jamming-resistant channel* allows a receiver to send feedback to the sender for making more informed decisions. In addition, we also assume that the sender can detect packet loss at every active channel. This can be achieved by (1) having the sender monitor the active channels, (2) adding extra

Table 5.1. Frequently Used Notations

R	set of receivers, i.e., $R = \{R_1, R_2, \cdots, R_r\}$
R'	set of compromised receivers
C_G	set of broadcast channels assigned to group G
C'_G	set of jammed channels in C_G
t	number of the compromised receivers, i.e., $t = \|R'\|$
m	number of broadcast channels assigned to each receiver
n	total number of available channels in the system
j	number of channels can be jammed at one time
η	fraction of corrupted channels we can tolerate
ρ	channel sharing factor, i.e., fraction of shared channels

monitoring nodes in the network, or (3) asking those existing receivers to report the channel condition.

The attacker's goal is to prevent as many receivers from receiving broadcast messages as possible. We assume that the attacker knows all technique specifics, including configuration parameters. We consider both *outsider attackers* and *insider attackers*. An outsider attacker does not know which channels are used. Thus he only randomly selects channels to jam. We assume that outsider attackers have power constraints that only allow them to block no more than j channels at one time.

An insider attacker can compromise some receivers and learn all their secrets, including their assigned channels. In this chapter, we focus on detecting the *active* malicious node, called *traitor*, whose channel information is currently being used by the attacker to launch jamming attacks. For those stealthy or selective attackers that behave normally most of the time, our goal is to catch them after they jam the communication for more than a small number of times. We let t denote the total number of compromised receivers.

5.2 Scheme I: Adaptive Re-Grouping with Partial Channel Sharing

For each broadcast message, the goals of jamming-resistant broadcast are: (i) the sender sends as few copies of this message as possible and (ii) the message can be correctly delivered to as many receivers as possible in the presence of insider jammers. The former indicates the need for a small number of groups. However, the latter requires small group size since a traitor can block the messages to all other receivers in the same group. This usually leads to a large number of groups. In Scheme I, we address such dilemma by proposing an *adaptive re-grouping* idea to isolate traitors without increasing the number of groups and a *partial channel sharing* idea to reduce the number of active channels needed to deliver broadcast messages.

Specifically, we classify the groups into *trusted groups* (TG) and *suspicious groups* (SG). There is only one trusted group but could be multiple suspicious groups. The idea is to adaptively re-group the receivers if the attacker launches the jamming attack so that the benign nodes are more likely to be merged into the trusted group, and the traitors are more likely to be included in a number of small suspicious groups.

The high-level description of our protocol is given below. The trusted group includes all receivers *currently* believed (by our protocol) to be trustworthy. It is possible that the trusted group becomes untrusted later when more observations about the channel condition are available. Once this happens, we split the group into two suspicious groups (or a *suspicious group pair*) and set the trusted group to be empty. We let this pair of suspicious groups *partially* share their channels for less communication cost, instead of using completely different set of channels. A group is said to be a suspicious group if we *currently* cannot determine whether it contains traitors; we need more observations about the channel condition to make the decision. Once we determine one of the group pair contains traitors, we split it into two smaller suspicious groups for further processing. Meanwhile, if we can not determine the other

Table 5.2. Decision Making Criteria in TG examination

Observation		Decision
E_0	$\|C'_{TG}\| < \eta m$	Accept H_0: $R' \cap TG = \emptyset$
E_1	$\|C'_{TG}\| \geq \eta m$	Accept H_1: $R' \cap TG \neq \emptyset$

group contains traitors at that time, we will believe it to be trustworthy and merge it into the trusted group. In other words, a suspicious group is merged into the trusted group only when its peer is determined to be untrustworthy. The above procedure continues until all traitors are isolated or the jamming attack stops.

Given the high-level protocol description, the remaining issues are: (1) how to detect the existence of traitors in the trusted group, (2) how to partially share channels between a suspicious group pair, (3) how to detect the untrustworthy group in a suspicious group pair, and (4) how to identify and remove traitors. We answer these problems in the following.

Detecting the existence of traitors in the trust group: Initially, all receivers belongs to the same trusted group TG; they share the same m channels. Note that if the trusted group contains no traitors, it is very difficult for the attacker to jam enough (at least ηm) channels to block the communication, as we will show in Section 5.3. Thus, if the number of jammed channels exceeds ηm, i.e., the receivers cannot recover the broadcast message, then it is very likely that the trusted group has traitors at this moment. Hence, we simply monitor whether $|C'_{TG}|$, i.e., the number of jammed channels in TG, exceeds ηm. If not, the group is still a trusted group; otherwise, we consider it as untrusted and immediately split it into two suspicious groups. (The trusted group becomes empty in this case.) Table 5.2 lists the criteria for making decisions.

Table 5.3. Decision Making Criteria in SGP examination

Observation			Decision
E_2	$\lvert C'_{SG_1}\rvert < \eta m$ and $\lvert C'_{SG_2}\rvert < \eta m$		Accept H_2: No traitors
E_3	$\lvert C'_{SG_1}\rvert \geq \eta m$ or $\lvert C'_{SG_2}\rvert \geq \eta m$	$\lvert EC'_1\rvert < \lvert EC'_2\rvert$	Accept H_3: SG_2 has traitors
E_4		$\lvert EC'_1\rvert = \lvert EC'_2\rvert$	Accept H_4Both have traitors
E_5		$\lvert EC'_1\rvert > \lvert EC'_2\rvert$	Accept H_5: SG_1 has traitors

Group splitting with partial channel sharing: When a group is determined to be untrusted, we randomly split it into two equal-sized suspicious groups, i.e., a suspicious group pair. We focus on a given suspicious group pair SGP during the discussion. We let SG_1 and SG_2 denote the two groups of SGP. We assign m randomly selected channels to each of these two groups. Let C_{SG_1} and C_{SG_2} be the channels assigned, respectively, to SG_1 and SG_2. In this chapter, SG_1 and SG_2 share a random fraction ρ of channels. Let SC be the shared common channels and EC_1 and EC_2 be the private (exclusive) channels for groups SG_1 and SG_2, respectively.

In each round of re-grouping, a receiver only receive its channel set. The sender will never leak any information about which channels belong to SC, EC_1, or EC_2. Apparently, if both groups in SGP have traitors, the attacker can find out the channel assignment and figure out all channel sharing information; this impact will be evaluated in Section 5.3.

Determining the untrusted group in SGP: The goal here is to determine which group (SG_1, SG_2, or both) contains traitors. Let us consider an example when the sender notices that $|C'_{SG_1}| \geq \eta m$. There are three cases. First, when only SG_1 contains traitors, the attacker can easily select and jam ηm or more channels in C_{SG_1}. Second, when only SG_2 contains traitors, the attacker can jam some channels in C_{SG_2} expecting to jam some channels in SC and then randomly jam the channels in all

other available channels expecting to jam the channels in EC_1. It is possible that $|C'_{SG_1}| = |SC'| + |EC'_1| \geq \eta m$. Third, both SG_1 and SG_2 contains traitors. In this case, the attacker knows the channel assignment and can arbitrarily jam the channels in C_{SG_1}. An effective scheme is thus needed to distinguish these three cases given the observations about the channel condition.

Our scheme takes advantage of the fact that the channel assignment of a traitor-free group is always kept secret. In other words, the sender only sends C_{SG_1} to the receivers in SG_1, but does not leak anything about which channels belong to EC_1 or SC. Hence, we have the following observation: *if only one suspicious group in SGP has traitors, the probability that its private channels are jammed is usually higher than the probability that the private channels of the peer (traitor-free) group in SGP are jammed.* The reason is that the adversary does not know which channels are private to the peer group and can only randomly select channels from all available channels to jam. Therefore, we can determine which group has traitors by studying the jammed channels.

More specifically, if any receiver in the suspicious group pair SGP is blocked from receiving broadcast messages, we mark the group which has more jammed private channels as the one that contains traitors. Note that it is possible that both SG_1 and SG_2 contain traitors. In this case, the attacker can identify EC_1, EC_2 and SC. Thus, he can jam exactly the same number of channels in both EC_1 and EC_2. This is, however, unlikely to happen in the case where at most one group contains the traitor. Therefore, if $|EC_1| = |EC_2|$, we mark both SG_1 and SG_2 as untrusted. Table 5.3 lists the decision criteria on determining untrusted groups. Detailed analysis will be given in Section 5.3.

Detecting and revoking the traitor: If a traitor keeps attacking the broadcast system, the size of its group will be reduced continuously. Once we detect a group

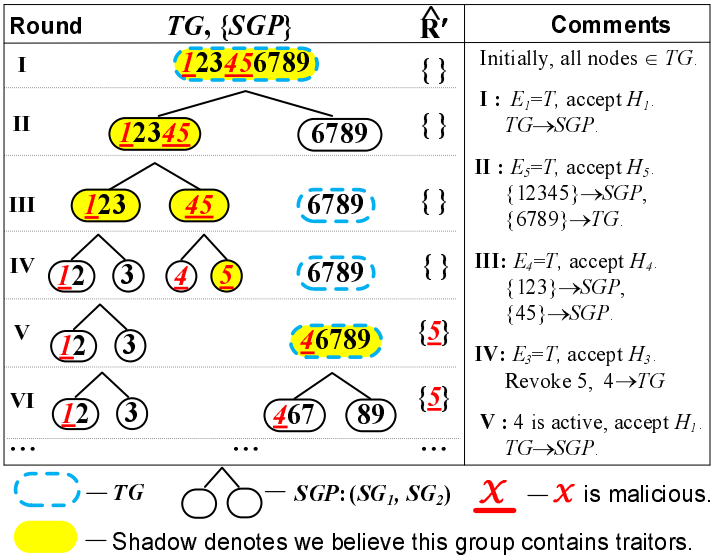
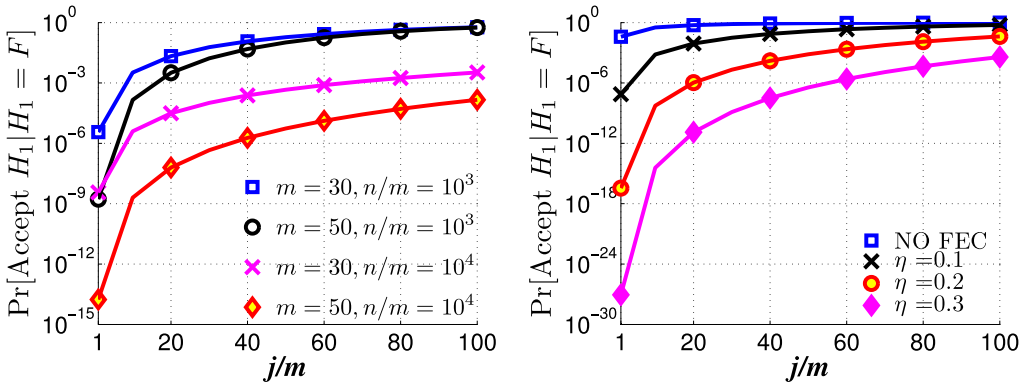


Figure 5.1. Illustration of procedure 1.

with only one member has the traitor, we can directly revoke this member from the system. Procedure 1 shows the pseudocode of our jamming-resistant broadcast scheme. Figure 5.1 shows an example of our jamming-resistant broadcast scheme.

5.3 Analysis of Scheme I

There are two traitor detection modules in Scheme I, one for detecting the existence of traitors in TG and the other for detecting which group in SGP cannot be untrusted. A traitor detection is used to decide one of the hypotheses in Tables 5.2 and 5.3 based on the channel condition. In our analysis, we will focus on the *decision error rate* $\Pr[\text{Accept } H | H = F]$, where F represents FALSE. We will also discuss the impact of system parameters (m, N, η, and ρ) and the attacker related parameter (j) on our protocol. We assume $\rho > \eta$.



(a) (b)

Figure 5.2. The decision error rate of traitor detection on a traitor-free TG. $\eta = 0.1$ in (a); $m = 40$ and $n = 10^3 \times m$ in (b).

5.3.1 Performance of Traitor Detection for the Trusted Group

Pr[Accept $H_0|H_0$=F]: If $|C'_{TG}| < \eta m$, the nodes in TG can recover the broadcast message, the attacker does not block the communication. Although some receivers may have been compromised, there are no traitors actively involved in jamming the communication. It is therefore reasonable to believe that $\Pr[\text{Accept } H_0|H_0 = F] = 0$. Nevertheless, we can always determine the existence of malicious receivers once the attacker starts the jamming attacks.

Pr[Accept $H_1|H_1$=F]: If TG contains no traitors, the attacker does not know which m channels belong to C_{TG}, he can only randomly select j channels from all n channels to jam. Thus, $|C'_{TG}|$, the number of TG's jammed channels, follows the hypergeometric distribution $f(i; n, m, j)$:

$$f(i; n, m, j) = \binom{m}{i}\binom{n-m}{j-i} \Big/ \binom{n}{j}. \tag{5.1}$$

Thus, $\Pr[\text{Accept } H_1|H_1 = F]$ can be calculated by

$$\Pr[\text{Accept } H_1|H_1 = F] = \sum_{i=\eta\times m}^{m} f(i; n, m, j) \tag{5.2}$$

Figure 5.2 shows the decision error rate for traitor detection on a traitor-free TG under different settings. Parameter n is determined by factors like the design

of spreading code, the channel hopping patterns, and the available spectrum in a specific channelization scheme. Intuitively, a larger n means more resistance against jamming attacks. This is shown in Figure 5.2(a). In multi-carrier systems, m is often used for resource allocation. If a node requires more bandwidth, we can assign more channels. On the other hand, we can increase m to improve the resistance against the jamming attacks as shown in Figure 5.2(a). η is determined by the forward error correction (FEC) scheme. Figure 5.2(b) shows that we can enhance the resistance against jamming by increasing η. Based on Figure 5.2, we choose $n/m = 10^3$, $j/m = 1 \sim 100$, and $\eta = 0.1, 0.2, 0.3$ in all our later analysis.

5.3.2 Performance of Detecting the Untrusted Group in SGP

Pr[Accept $H_2|H_2=F$]: Similar to the analysis in the last subsection, it is reasonable to believe that $\Pr[\text{Accept } H_2|H_2 = F] = 0$ in this case since the communication is not blocked.

Pr[Accept $H_x|H_x=F]|_{x=3,4, \text{ or } 5}$.: Since our scheme has the same performance when $x = 3$ or $x = 5$, we only discuss the cases when $x = 3$ or $x = 4$. According to the definitions of the hypotheses in Table 5.3, we have

$$\neg H_3 \Leftrightarrow (\neg H_4 \wedge H_5) \vee H_2$$
$$\neg H_4 \Leftrightarrow (\neg H_4 \wedge H_5) \vee (\neg H_4 \wedge H_3) \vee H_2.$$

For simplicity, we define the following four probabilities:

$$\begin{cases} P_1: & \Pr[\text{Accept } H_3|\neg H_4 \wedge H_5 = T], \\ P_2: & \Pr[\text{Accept } H_3|H_2 = T], \end{cases}$$
$$\begin{cases} P_3: & \Pr[\text{Accept } H_4|\neg H_4 \wedge H_5 = T], \\ P_4: & \Pr[\text{Accept } H_4|H_2 = T], \end{cases}$$

Table 5.4. Probabilities that the attacker succeeds in making particular events occur

Pr[Event | Condition]			Parameters in equation 5.1.			
	Events	Conditions	N_t	N_d	N_s	i
P_1	$E_3 = T$	$\neg H_4 \wedge H_5 = T$	$\lvert C_{SG_1}\rvert = m$	$\lvert EC_1\rvert = (1-\rho)m$	$\lvert C'_{SG_1}\rvert \in [0, m-1]$	See Inequality 5.4.
			$\lvert C^c_{SG_1}\rvert = n - m$	$\lvert EC_2\rvert = (1-\rho)m$	$j - \lvert C'_{SG_1}\rvert$	See Inequality 5.5 and 5.6.
P_2		$H_2 = T$	$\lvert C\rvert = n$	$\lvert C_{SG_2}\rvert = m$	j	$\lvert C'_{SG_2}\rvert \in [\eta m, m]$
			$\lvert C_{SG_2}\rvert = m$	$\lvert EC_2\rvert = (1-\rho)m$	$\lvert C'_{SG_2}\rvert$	$\lvert EC'_2\rvert \in [1, \text{MIN}(\lvert C'_{SG_2}\rvert, (1-\rho)m)]$
			$\lvert {C'_{SG_2}}^c\rvert = n - \lvert C'_{SG_2}\rvert$	$\lvert EC_1\rvert = (1-\rho)m$	$j - \lvert C'_{SG_2}\rvert$	$\lvert EC'_1\rvert \in [0, \lvert EC'_2\rvert - 1]$
P_3	$E_4 = T$	$\neg H_4 \wedge H_5 = T$	$\lvert C_{SG_1}\rvert = m$	$\lvert EC_1\rvert = (1-\rho)m$	$\lvert C'_{SG_1}\rvert \in [\eta m, m]$	$\lvert EC'_1\rvert \in [0, \text{MIN}(\lvert C'_{SG_1}\rvert, (1-\rho)m)]$
			$\lvert C^c_{SG_1}\rvert = n - m$	$\lvert EC_2\rvert = (1-\rho)m$	$N_{s,2} \in [0, j - \lvert C'_{SG_1}\rvert]$	$\lvert EC'_2\rvert = \lvert EC'_1\rvert$
P_4		$H_2 = T$	$\lvert C\rvert = n$	$\lvert C_{SG_2}\rvert = m$	j	$\lvert C'_{SG_2}\rvert \in [\eta m, m]$
			$\lvert C_{SG_2}\rvert = m$	$\lvert EC_2\rvert = (1-\rho)m$	$\lvert C'_{SG_2}\rvert$	$\lvert EC_2\rvert \in [0, \text{MIN}(\lvert C'_{SG_2}\rvert, (1-\rho)m)]$
			$\lvert {C'_{SG_2}}^c\rvert = n - \lvert C'_{SG_2}\rvert$	$\lvert EC_1\rvert = (1-\rho)m$	$j - \lvert C'_{SG_2}\rvert$	$\lvert EC'_1\rvert = \lvert EC'_2\rvert$

where T represents TRUE. Thus, we have

$$\Pr[\text{Accept } H_3 | H_3 = F] \leq \text{MAX}(P_1, P_2)$$

$$\Pr[\text{Accept } H_4 | H_4 = F] \leq \text{MAX}(P_3, P_4)$$

P_1 is the probability that event E_3 occurs, given the condition that only SG_1 contains traitors. In this case, the attacker only knows C_{SG_1} and cannot distinguish EC_1 from SC. To interrupt the communication at SG_2, the attacker may first jam a number of channels in C_{SG_1} expecting to hit some channels in SC, and then jams $(j - |C'_{SG_1}|)$ channels randomly selected from the other available $(n - |C'_{SG_1}|)$ channels expecting to hit some channels in EC_2. Event E_3 describes the case where (i) at least one of $|C'_{SG_1}|$ and $|C'_{SG_2}|$ exceed $\eta \times m$ and (ii) $|EC_1| < |EC_2|$. These limit the ranges of $|C'_{SG_1}|$, $|EC'_1|$, and $|EC'_2|$. Apparently, we have $|C'_{SG_1}| \neq m$ since otherwise $|EC'_1| = (1 - \rho) \times m$ and $|EC'_1| \geq |EC'_2|$. Thus, we have

$$0 \leq |C'_{SG_1}| \leq m - 1. \tag{5.3}$$

Consequently, we also have

$$0 \leq |EC'_1| \leq \text{MAX}(|C'_{SG_1}|, (1 - \rho)m - 1), \tag{5.4}$$

If $|C'_{SG_1}| < \eta m$, the attacker has to make $|EC'_2|$ greater than $\eta m - |SC'_1|$ to ensure that $|C'_{SG_2}| \geq \eta m$ in order to interrupt the communication at SG_2. Since $|SC'_1| = |C'_{SG_1}| - |EC'_1|$, we have

$$|EC'_2| \geq \begin{cases} \text{MAX}(|EC'_1| + 1, \eta m - |C'_{SG_1}| + |EC'_1|) & \\ & \text{if } |C'_{SG_1}| < \eta m, \\ |EC'_1| + 1 & \text{if } |C'_{SG_1}| \geq \eta m. \end{cases} \tag{5.5}$$

Obviously, the upper bound of $|EC'_2|$ is given by:

$$|EC'_2| \leq \text{MIN}((j - |C'_{SG_1}|), (1 - \rho)m), \tag{5.6}$$

Therefore, P_1 can be estimated by:

$$P_1 = \sum_{|EC_2'|} \sum_{|EC_1'|} f(|EC_1'|; m, (1-\rho)m, |C'_{SG_1}|) \times f(|EC_2'|; n-m, (1-\rho)m, j-|C'_{SG_1}|),$$

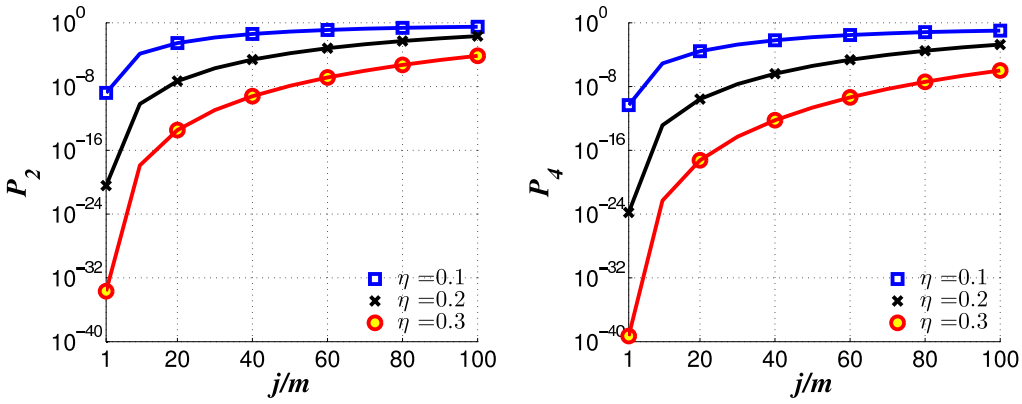
where $f(i : N_t, N_d, N_s)$ is the hypergeometric distribution (i.e. Equation 5.1). The ranges of $|C'_{SG_1}|$, $|EC_1'|$, and $|EC_2'|$ are given by Inequalities 5.3,5.4, and 5.5, respectively. Similarly, we can calculate P_2, P_3, and P_4. The results are summarized in Table 5.4.

5.3.3 Performance under Worst-Case Scenarios

In this subsection, we will analyze the worst-case performance of our protocol when the attacker takes the best possible strategy (i.e., he launches various types of jamming attacks to introduce the maximum number of wrong decisions). The analysis is conducted in the following there cases: (1) no group contains traitors; (2) only one suspicious group contains traitors; (3) both of the suspicious group pair contain traitors. We skip the case where the trusted group contains traitors, because once the attacker blocks the communication, we will notice the existence of traitors.

No group in SGP contains traitors: In this case, the attacker is not aware of any channel assignment, and can only jam randomly selected channels. The best strategy is to jam as many channels as he could. Apparently, the energy limit determines the attacker's jamming ability, i.e., the more channels he can block at the same time (or the greater the value of j), the higher probability he can interrupt the communication. This is shown in Figures 5.2 and 5.3.

Only one group in SGP contains traitors: Suppose only SG_1 has traitors and the attacker intends to fool us into believing that SG_2 contains traitors. Let us consider P_1 and P_3. In addition to j, there is another parameter that can affect the



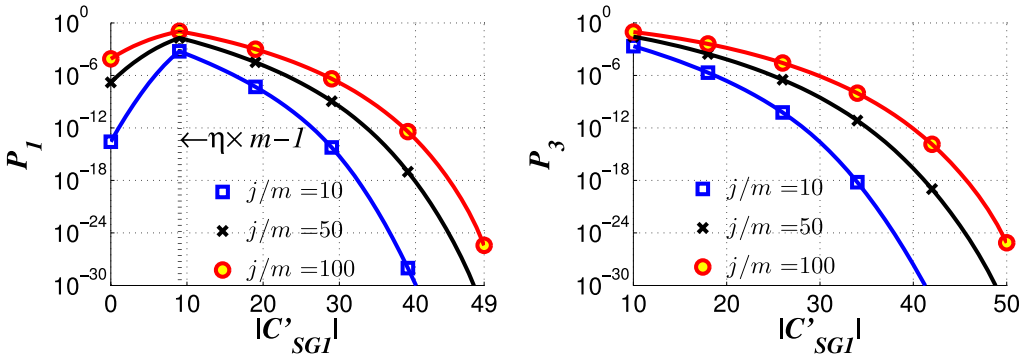
(a) (b)

Figure 5.3. The decision error rate in traitor detection on a traitor-free SGP. (a) Pr[Accept $H_3|H_2 = T$] and (b) Pr[Accept $H_4|H_2 = T$]. Assume $m = 50$, $n/m = 10^3$, and $\rho = 0.5$.

attacker's jamming performance, i.e., $|C'_{SG_1}|$. Different from j, which is limited by the attacker's energy, $|C'_{SG_1}|$ can be controlled by the attacker.

Figure 5.4 shows the impact of $|C'_{SG_1}|$. From Figure 5.4(a), we know that P_1 reaches its maximum value $P_{max,1}$ when $|C'_{SG_1}| = \eta m - 1$. This means that the best jamming strategy is to jam $\eta m - 1$ channels in C_{SG_1} and spend the rest of energy on jamming the channels randomly selected from $C^c_{SG_1}$. Similarly, from Figure 5.4(b), $P_3 \leq P_{max,3} = P_3|_{|C'_{SG_1}|=\eta m}$. Intuitively, the insider jammer has more impact than the outsider attacker. Given the same configuration, we have $P_{max,1} > P_2$ and $P_{max,3} > P_4$, which can be seen from comparing Figure 5.3(a) with Figure 5.4(a), and comparing Figure 5.3(b) with Figure 5.4(b). As a result, Pr[Accept $H_3|H_3{=}F] \leq P_{max,1}$ and Pr[Accept $H_4|H_4{=}F] \leq P_{max,3}$. The worst-case performance is illustrated in Figure 5.5. From Figures 5.2, 5.3, and 5.5, we can see that the performance degrades with a larger j, but can be improved dramatically by increasing η.

Both groups of SGP contain traitors: In this case, the attacker knows the channel assignment and sharing information. However, this does not mean that the attacker can evade our protocol and block the broadcast messages arbitrarily. To



(a) (b)

Figure 5.4. The decision error rate in traitor detection on SGP. (a) $\Pr[\text{Accept } H_3 | \neg H_4 \wedge H_5 = T]$ and (b) $\Pr[\text{Accept } H_4 | \neg H_4 \wedge H_5 = T]$. Assume $m = 50$, $n/m = 10^3$, $\rho = 0.5$ and $\eta = 0.2$.

prevent a victim R_i from receiving the broadcast message, the attacker has to jam at least ηm channels assigned to R_i, which means one of the events E_3, E_4, or E_5 must occur. Correspondingly, we will accept either H_3, H_4, or H_5, and split the groups to isolate traitors. Thus, the attacker's best strategy here is to "sacrifice" the traitors in one group to hide the traitors in the other group. Note that the hidden traitors will be assigned into the trusted group, and have the chance to launch another attack. However, we note that the hidden traitor can simply be handled at the moment when its knowledge is reused for launching the jamming attacks in the future.

Overall impact of compromised receivers: In the following, we analyze how well this protocol performs under various types of jamming attacks including the selective attacks, where the insider jammers do not launch attacks continuously but choose to only attack at selected times. We consider the following questions: given t compromised receivers, how many benign receivers will lose the broadcast message in each round of jamming attacks, and how many rounds of jamming attacks can the attacker launch?

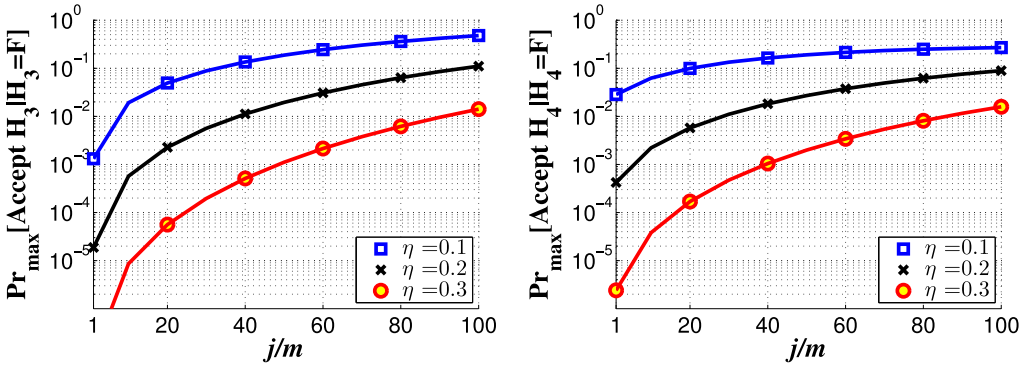
From Figure 5.1, we notice that the adaptive re-grouping leads to a tree-like structure. Each group that is believed to contain traitors is split into two suspicious

groups, which can be considered as its children. If a traitor R'_i keeps active, i.e., jamming, its group will be split continuously until R'_i becomes the leaf of the tree, i.e., it becomes the only member of its group. In this case, if R'_i keeps acting maliciously, it will be removed from the system. Thus the attacker will keep R'_i inactive at certain point so that the sender has to keep sending messages to its group, wasting the sender's energy.

When we split an untrusted group into two suspicious groups, it is possible that both contain traitors but later we only detect one of them being untrustworthy. In other words, the attacker can hide the traitors in one of the groups by simply not using their secrets to jam the communication, and thus this group will be merged into the trusted group. The attacker may take advantage of this and make use of the secrets of its traitor one by one to maximize the interference.

Let R'_x denote the x-th active traitor whose secret is used to launch jamming attacks. In the beginning, the attacker keeps using R'_1's secret for jamming until R'_1 becomes a leaf, producing two suspicious groups and one trusted group. This introduces $\lceil \lg(|R|) \rceil$ rounds of attack. After that, the attacker will make use of the secret at another traitor R'_2 for jamming until R'_2 becomes a leaf, producing 4 suspicious groups and one trusted group. This introduces $\lceil \lg(|R| - 2) \rceil$ rounds of attacks. This procedure continues until all traitors become leaves, producing $2t$ suspicious groups and one trusted group. Hence, R'_x can be used for $\lceil \lg(|R| - 2(x-1)) \rceil$ times.

Note that the number of receivers affected by the active traitor R'_x varies in each round of attack. This is because the size of R'_x's group is reduced by half after each round of attack. Therefore, the maximum numbers of receivers affected by R'_x in each round of attack are $\{|R| - 2(x-1) - 1, \lceil \frac{|R|-2(x-1)}{2} \rceil - 1, \lceil \frac{|R|-2(x-1)}{4} \rceil -$



(a) (b)

Figure 5.5. The decision error rate in traitor detection on SGP when only one group contains traitors ($m = 50$, $n/m = 10^3$, and $\rho = 0.5$). (a) MAX(P_1, P_2) and (b) MAX(P_3, P_4).

$1, \cdots 1\}$. Overall, the maximum rounds of jamming attacks can be estimated by $\sum_{x=1}^{t} \lceil \lg(|R| - 2(x-1)) \rceil$.

Communication overhead and impact of ρ: Obviously, it is not possible to merge all the trusted receivers into a single group unless all the traitor nodes can be exactly pinpointed. However, if a malicious receiver becomes the single member in its own group, it may choose to hide himself and never attack again. There is no effective mechanism to detect such non-active attacker, and the sender has to send extra copies to guarantee that no legitimate receiver will lose the message. The existing solutions require $2t$ extra copies [19]. According to the previous analysis, our approach reduces such extra overhead to $(2-\rho)t$ copies. If $\rho = 0.5$, the communication overhead is $1.5t$ extra messages. Apparently, a larger ρ means fewer copies we need to send for each broadcast and thus less communication overhead. By properly configuring ρ, we can have much less communication cost than the previous methods.

However, we note that the value of ρ impacts P_1, P_2, P_3, and P_4. Thus, it will also impacts the decision error rate. Figure 5.6 plots the maximal decision error rates on testing hypotheses H_3 and H_4 with different parameters.

We first look at $\Pr_{max}$[Accept $H_4|H_4$=F]. Based on the previous analysis, we know that

$$\Pr_{max}[\text{Accept } H_4|H_4 = F] = f(0; m, (1-\rho)m, \eta m),$$

which apparently increases with ρ. Figure 5.6(b) also confirms that the decision error rate on H_4 increases with ρ.

We then look at $\Pr_{max}$[Accept $H_3|H_3$=F]. Suppose only group SG_1 has traitors, the attacker's best strategy is to jam $\eta m - 1$ channels in C_{SG_1} and as many channels in ${C_{SG_1}}^c$ as possible. If the attacker hits no channel in EC_1 and one channel in EC_2, he can fool us into believing that SG_1 is traitor-free but SG_2 is not. The probability of this happening can be estimated by

$$\Pr_{max}[\text{Accept } H_3|H_3 = F] = \Pr[EC_1' = 0] \times \Pr[EC_2' > 1].$$

We note that this is not a monotonic function. The reason is that

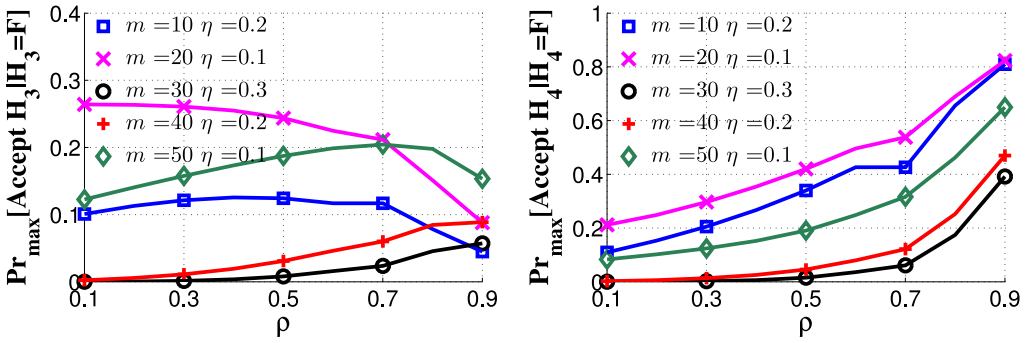
$$\Pr[EC_1' = 0] = f(0; m, (1-\rho)m, \eta m - 1).$$

increases with ρ, but

$$\Pr[EC_2' \geq 1] = 1 - f(0; n-m, (1-\rho)m, j - \eta m + 1),$$

decreases with ρ. In addition, we also need to consider other parameters such as m, n, j, and η. Moreover, since $\Pr[EC_2' \geq 1]$ is usually small, the decision error rate on testing H_3 is still relatively low even when the decision error rate on testing H_4 with the same parameter set is very high, which can be seen from Figure 5.6(a) and Figure 5.6(b).

From the above discussion, we believe that given a reasonable system configuration and attacker's jamming ability, our system can resist jamming attacks with very low decision error rates. However, we also note that if we have very limited resources



(a) (b)

Figure 5.6. The impact of ρ in resource-constraint systems. (a) the impact of ρ on testing H_3 and (b) the impact of ρ on testing H_4. Assume $n/m = 10^3$, and $j/m = 50$.

(i.e., very small m, n and η) and strong attackers (i.e., a very large j), our system will generate a high decision error rate in order to save more cost in terms of communication. This is shown in Fig. 5.6. In the next section, we will present a scheme to cope with the resource limitation and powerful attackers and also to improve the performance so that we can set a larger ρ to save more communication cost.

5.4 Scheme II: Sequential Test Based Detection

To reduce the decision error rate, we propose to get more observations about the channel condition before making any decision. Generally, the more observations, the lower the decision error rate (and the larger ρ we can set). However, waiting for more observations increases the cost and the decision delay. For example, the sender will need to assign new channels to replace those jammed ones.

We propose to use a risk function to capture the impact of detection errors and delays. Without loss of generality, we consider a loss of 1 if the decision is wrong, and a loss of 0 if the decision is correct. We then introduce c to represent the ratio of the

cost of waiting for one more observation over the cost of making a wrong decision. Thus, the problem becomes to find a solution to minimize the following risk function:

$$z = c \times E[S] + \Pr[\text{The decision is wrong}], \tag{5.7}$$

where variable S denotes the total number of channel observations we have collected at the moment when we stop the sequential test and make a decision.

5.4.1 Problem Statement

As discussed before, when we detect jamming attacks in SGP (i.e., $\text{MAX}(|C'_{SG_1}|, |C'_{SG_2}|) \geq \eta m$), we know that at least one of them contains traitors. Note that if both groups contain traitors, we can always identify at least one untrusted group. The missed one that also contains traitors can be simply handled in the future. As a result, in this section, we focus only on the case when only one of these two groups contain traitors.

The main problem now is to tell which group is more likely to contain traitors given the observations about the channel condition. In the following, we will convert such traitor detection problem into an estimation problem of two random variables X and Y that are given by

$$X = \begin{cases} 1, & \text{if } |EC'_1| < |EC'_2|, \\ 0, & \text{if } |EC'_1| \geq |EC'_2|. \end{cases} \quad Y = \begin{cases} 1, & \text{if } |EC'_1| > |EC'_2|, \\ 0, & \text{if } |EC'_1| \leq |EC'_2|. \end{cases}$$

Since only one group contains traitors, we can see that X and Y will both follow a Bernoulli distribution with a mean of P_x and P_y, respectively. Specifically, we have

$$\begin{cases} Pr[X = 1] = P_x = 1 - Pr[X = 0] \\ Pr[Y = 1] = P_y = 1 - Pr[Y = 0] \end{cases}$$

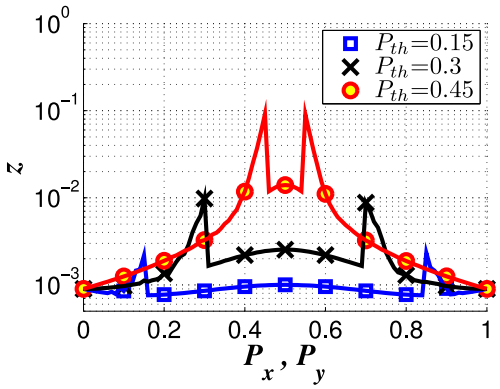


Figure 5.7. The risk z v.s. P_x or P_y ($c = 10^{-4}$).

Let $P_{th} = \Pr_{max}[\text{Accept } H_3 | H_3 = F]$. We also have $P_{th} = \Pr_{max}[X = 1 | SG_2 \text{ has no traitors}]$. In other words, if SG_2 has no traitors, we have $P_x \leq P_{th}$. This implies:

$$P_x > P_{th} \Rightarrow H_3 : SG_2 \text{ has traitors.} \tag{5.8}$$

Similarly, we have

$$P_y < 1 - P_{th} \Rightarrow H_3 : SG_2 \text{ has traitors.} \tag{5.9}$$

$$P_y > P_{th} \Rightarrow H_4 : SG_1 \text{ has traitors.} \tag{5.10}$$

$$P_x < 1 - P_{th} \Rightarrow H_4 : SG_1 \text{ has traitors.} \tag{5.11}$$

From 5.8, 5.9, 5.10, and 5.11, we know that we can use P_x and P_y for detecting the group that contains traitors. We can then apply Lai's Bayes Sequential Test [58] to address the problem.

5.4.2 Applying Lai's Bayes Sequential Test

We do not use those popular sequential test methods such as the fictitious optimal fixed sample size test or Wald's sequential probability ratio test since they require the knowledge of fixed P_x and P_y to achieve the optimal results. In our case, the attacker can arbitrarily change these values.

In [58], Lai gives a Bayes sequential test scheme to test the composite hypothesis: $H : P < P_0$ versus $K : P > P_0$. Lai shows that the risk z (i.e., Equation 5.7) of this sequential test is asymptotically equivalent to that of the fictitious optimal fixed sample size test that assumes the knowledge of P. The stopping rule of Lai's Bayes sequential test is

$$S = \inf\{s \geq 1 : I(\overline{P}_s, P_0) \geq \frac{h_0(c \times s)}{2c \times s^2}\},$$

where $\inf\{s\}$ denotes the infimum of set {s}; $I(\widetilde{P}, P)$ is the Kullback-Leibler information number given by $I(\widetilde{p}, p) = \widetilde{p}\log(\widetilde{p}/p) + (1 - \widetilde{p})\log((1 - \widetilde{p})/(1 - p))$. When $\widetilde{p} = 0$ or 1, we define $I(\widetilde{p}, p) = \log 2$; $h_0(\cdot)$ is a function given in [58], which is also listed below:

$$h_0(x) = \begin{cases} (2/\pi)^{1/2}(x^{-1/2} - 5x^{-5/2}/48\pi)/4, & \text{if } x \geq 0.8, \\ \exp(-0.69x - 1), & \text{if } 0.1 \leq x < 0.8, \\ 0.39 - 0.015x^{-1/2}, & \text{if } 0.01 \leq x < 0.1, \\ (t(2\log(1/x) + \log\log(1/x) & \\ -\log 4\pi - 3\exp(-0.016x^{-1/2})))^{1/2} & \text{if } x < 0.01. \end{cases}$$

The terminal decision rule (the final decision) is to accept H or K according to $\overline{P}_S > P_0$ or $\overline{P}_S < P_0$.

5.4.3 Bayes Sequential Test Based Detection

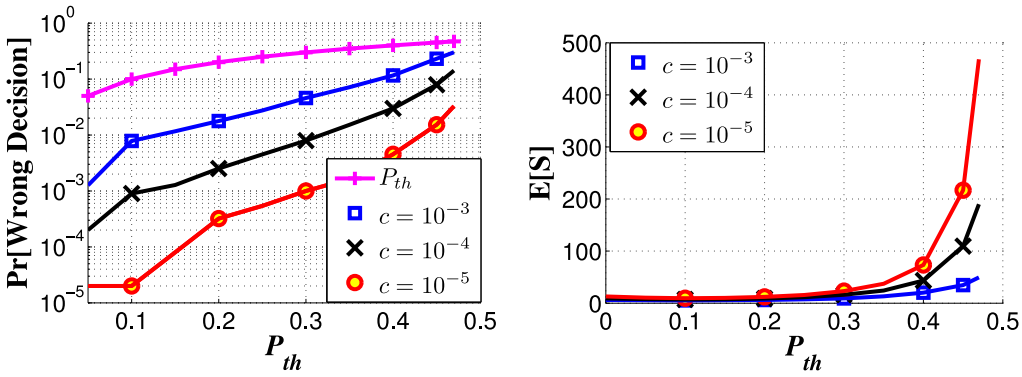
Based on Lai's stopping rule and terminal decision rule, we propose our Bayes sequential test based detection in Procedure 3, which can be used to replace Line 15 in Procedure 1 to have an improved scheme based on Bayes sequential tests.

5.4.4 Performance Analysis

Procedure 3 requires two system parameters, c and P_{th}. c is introduced to balance the detection error and delay; P_{th} is the function of the system parameters (m, n, ρ, and η) and the attacker's jamming ability (j), as discussed in Section 5.3.3. Figure 5.6(a) shows that $P_{th} < 0.5$ is usually true. Since the cost of making a wrong decision is very high, c is usually very small. We thus set $c = 10^{-4}$ as in [58]. We then use simulation to evaluate the performance under different values of P_x or P_y, which can be controlled by the attacker. The results in Figure 5.7 are based on 50,000 rounds of simulation. It shows that our Bayes sequential test based approach can achieve a low risk no matter what strategy the attacker takes.

Figure 5.7 also shows that the best strategy for the attacker is to let $P_x = P_{th}$ or $P_y = P_{th}$. Figure 5.8 plots both of the expected number of lost messages before making a decision and the decision error rate. It shows that our scheme can achieve very good performance even in the worst case scenario where the attacker always takes the best strategy. It also shows that we can configure c according to the application to balance the decision delay and the error rate.

Figure 5.8(a) also plots P_{th}, the probability of making a wrong decision of accepting H_3 using our first scheme when the attacker takes the best jamming strategy. We can see that our Bayes sequential test based detection dramatically reduces the decision error rate with very small number of samples (less than 10 in most cases). Unlike the first scheme, the second scheme can achieve small decision error rates even if ρ is set large (e.g., 0.9). As a result, our sequential test based detection scheme can further reduce the communication overhead by configuring a larger ρ.



(a) (b)

Figure 5.8. Performance of sequential test based detection in the worst-case scenario. (a) the wrong decision rate and (b) shows the decision making speed. P_{th} is the probability of making a wrong decision of accepting H_3 using our first scheme in the worst-case scenario.

5.5 Summary

In this chapter, we propose an adaptive jamming-resistant broadcast system with partial channel sharing. Compared to existing approaches, the proposed scheme significantly reduces the communication overhead without sacrificing security. Moreover, the sequential test based scheme allows us to further reduce the communication cost, greatly pushing the limit of jamming-resistant broadcast towards optimal.

In the future, we are particularly interested in developing our systems on real wireless communication platforms. It is also highly desirable to evaluate the performance of our approaches through field experiments to obtain more useful results. For example, the analysis on our jamming-resistant broadcast schemes assumes reliable communication if no jammer attacks the network. In other words, during the analysis, we believe that the packet loss is always caused by jammers. We believe that there exists insider jammers if a receiver cannot recover a message. Although this is often true in case of reliable communication, we may draw wrong conclusions in reality since the wireless communication is quite unreliable. The high channel loss

may introduce a high decision error rate into our proposed approaches. It is thus important to conducting a thorough analysis on how channel loss rates impact our approaches.

In our schemes, if a suspicious group only has one compromised receiver, this compromise receiver can figure out the group size by looking at how many times the re-grouping process happened. Thus, if the compromised receiver realizes itself the only member in the group, it can stop jamming to prevent itself from being caught. In the future, we are going to make the group size information unpredictable to the receivers by randomizing the re-grouping process. In this way, we can increase the chance of detecting traitors.

Procedure 1 Jamming-Resistant Broadcast

$TG \leftarrow R$
$C_{TG} \leftarrow$ Randomly select m channels
$\{SGP\} \leftarrow \emptyset$, $\widehat{R'} \leftarrow \emptyset$
repeat
  Broadcast according to C_{TG} and $\{C_{SGP}\}$ and monitor the jammed channels C'.
  if $TG \neq \emptyset$ **then** // begin TG examination:
    if $|C'_{TG}| \geq \eta m$ **then** // $E_1 = T$
      $SplitGroup(TG)$// Accept H_1
    else
      Assign new channels to replace the jammed channels.
    end if
  end if
  if $\{SGP\} \neq \emptyset$ **then** // begin SGP examination:
    for $k = 0$ to $|\{SGP\}|$ **do**
      $ImmediateDetection(SGP_k)$
      if $H_4 = T$ **then**
        $SplitGroup(SG_{k,1})$, $SplitGroup(SG_{k,2})$
      else if $H_3 = T$ **then**
        $SplitGroup(SG_{k,2})$, $TG \leftarrow SG_{k,1} \cup TG$
      else if $H_5 = T$ **then**
        $SplitGroup(SG_{k,1})$, $TG \leftarrow SG_{k,2} \cup TG$
      else
        Assign new channels to replace the jammed ones.
      end if
    end for
  end if
until finish broadcast

Procedure 2 ImmediateDetection

Input: $SPG = \{SG_1, SG_2\}$

Output: H_3, H_4, H_5

1: $H_3 \leftarrow F$, $H_4 \leftarrow F$, $H_5 \leftarrow F$
2: **if** $(|C'_{SG_1}| \geq \eta m) \vee (|C'_{SG_2}| \geq \eta m)$ **then**
3: **if** $|EC'_1| < |EC'_2|$ **then** // $E_3 = T$
4: $H_3 \leftarrow H$
5: **else if** $|EC'_1| = |EC'_2|$ **then** // $E_4 = T$
6: $H_4 \leftarrow H$
7: **else** // $E_5 = T$
8: $H_5 \leftarrow H$
9: **end if**
10: **end if**

Procedure 3 SequentialDetection

Input: $SPG = \{SG_1, SG_2\}$, c, s, X, Y, P_{th}

Output: H_3, H_4, H_5

$H_3 \leftarrow F$, $H_4 \leftarrow F$, $H_5 \leftarrow F$
if $(|C'_{SG_1}| \geq \eta m) \vee (|C'_{SG_2}| \geq \eta m)$ **then**
  $s \leftarrow s + 1$
  if $|EC'_1| < |EC'_2|$ **then** // $E_3 = T$
    $X \leftarrow X + 1$
  else if $|EC'_1| > |EC'_2|$ **then** // $E_5 = T$
    $Y \leftarrow Y + 1$
  end if
  if $(I(\frac{X}{s}, P_{th}) \geq \frac{h_0(c \times s)}{2c \times s^2}) \wedge (\frac{X}{s} > P_{th})$ **then**
    $H_3 \leftarrow T$
  else if $(I(\frac{X}{s}, 1 - P_{th}) \geq \frac{h_0(c \times s)}{2c \times s^2}) \wedge (\frac{X}{s} < 1 - P_{th})$ **then**
    $H_5 \leftarrow T$
  end if
  if $(I(\frac{Y}{s}, P_{th}) \geq \frac{h_0(c \times s)}{2c \times s^2}) \wedge (\frac{Y}{s} > P_{th})$ **then**
    $H_5 \leftarrow T$
  else if $(I(\frac{Y}{s}, 1 - P_{th}) \geq \frac{h_0(c \times s)}{2c \times s^2}) \wedge (\frac{Y}{s} < 1 - P_{th})$ **then**
    $H_3 \leftarrow T$
  end if
  if $(H_3 = H) \wedge (H_5 = H)$ **then**
    $H_4 \leftarrow T$
  end if
end if

CHAPTER 6

MITIGATING JAMMING ATTACKS IN WIRELESS BROADCAST SYSTEMS WITH UNPREDICTABLE CHANNEL ASSIGNMENT

The technique in Chapter 5 reduces the extra communication cost significantly. However, this approach requires receivers to listen to multiple channels simultaneously, increasing the hardware cost significantly. Another common problem for the dynamic-grouping approaches is that the attacker can predict the channel assignment and achieve their maximal jamming impact without being detected.

This chapter proposes a improvement with *unpredictable channel assignment*, which prevents the insider attackers from knowing when to stop jamming. The high-level idea is to reassign the channels periodically, such that two receivers have a certain probability to share the newly assigned channel. However, we keep the channel sharing information secret from the receivers. When they share the same channel, we can reduce the communication overhead by sending only one copy of the message, instead of two. When they use different channels, and the attacker launches a jamming attack, then we can identify the malicious node, because the jammed channel has only one user, i.e., the malicious one. One benefit is that the attacker cannot achieve maximal jamming impact without being detected. In addition, the proposed technique does not require a receiver to operate on multiple channels at the same time.

6.1 Network and Adversary Model

In this chapter, we study a wireless broadcast system in which a benign sender sends messages to a set of receivers. We assume that the system contains n orthogonal

virtual channels, which are determined by either spreading codes, frequency-hopping patterns, sub-carriers, or their combinations. The sender will select a subset of those n channels to broadcast messages. If the sender assigns receivers u and v the same channel, both of them will listen to that channel and be able to receive the message from the sender at the same time. We assume that the sender and receivers are time synchronized and thus every receiver knows when the transmission will start and stop on any given channel.

Note that spread spectrum has been shown to be very resistant to jamming in pairwise communication, i.e., one-to-one communication, as long as the channels are picked securely. Hence, we assume that each receiver shares a secret key with the sender. This key is used to select a *private channel* for jamming-resistant pairwise communication between them. In other words, each receiver can communicate with the sender individually without being jammed. This private channel allows the sender to update the channel assignment for this receiver.

The attacker's goal is to prevent as many receivers from receiving broadcast messages as possible. We assume that the attacker knows all technique specifics, including the configuration parameters, and is able to jam up to j channels during each message transmission. We consider not only *outsider attackers* but also *insider attackers*. The only difference between these two type of attackers is the knowledge of channel assignment information. The outsider attacker does not know which channel is used and only selects a random set of channels to jam. The insider attacker can compromise receivers and learn all their secrets, including the channels assigned to them, which are often shared with some other benign receivers and thus can be used to launch jamming attacks effectively. We assume that the adversary can compromise up to t receivers.

Our goal in this chapter is to detect *active* malicious receivers, which we call *traitors*, whose assigned channel information is currently used by the attacker to launch jamming attacks. For example, the attacker could compromise a receiver and figure out what channel is currently used for receiving messages. This channel is very likely to be used by many other receivers for receiving broadcast messages. As a result, the attacker can jam this channel to block other receivers from getting the broadcast messages. Certainly, the attacker may choose to jam the channel in an *stealthy* way to reduce the chance of being detected. For example, he may only try to jam the channel when important messages are being sent. In this chapter, we consider stealthy jammers as well. Specifically, our goal is to ensure that the probability of detecting a jammer increases with the number of times he jams the channel. In other words, the stealthy jammer may behave normally and stay in the broadcast system for a long time. However, if he behaves maliciously more than a small number of times, our approach will identify him with very high probability.

In this chapter, we assume that the sender can detect whether a channel is being jammed with jamming detection techniques [59, 60]. This can be achieved by measuring the packet loss on active channels, i.e., the channels used for broadcasting messages. For example, we can ask the sender to monitor the active channels, add extra monitoring nodes in the network, or ask the existing receivers to report the channel condition.

Table 6.1 lists some frequently used notations in this chapter.

6.2 Unpredictable Channel Assignment

Dynamic-grouping techniques [19] address the jamming problem by trying to isolate the traitors in the system. The basic idea is to organize receivers into groups and use different channels for different groups. If the channel assigned to a given

Table 6.1. Frequently Used Notations

t	Number of compromised receivers
n	Total number of available channels in the system
j	Number of channels jammed during each message transmission
p	Probability that a pair of nodes use two different channels
m	Number of messages transmitted through one channel

group is jammed, the sender splits this group into two equal-sized, smaller groups and assigns a new channel to each of them. In this way, malicious receivers can be isolated step by step. For simplicity, we assume a network size of 2^h for some integer h. Therefore, at some point, we will have up to t groups with only two members in each group, where t is the number of compromised receivers.

If the channel assigned to a two-member group is jammed, then one of the group members must be malicious. However, we cannot determine which one is malicious. In existing approaches, the jammed two-member group will be split again and each member will get a new channel to receive messages. After splitting, the attacker will notice that his channel has been changed h times, which indicates that he is the only user of the channel currently assigned to him. In this case, he will simply stop jamming to evade detection. (If a channel is jammed and has only one user, then this user must be the traitor.) Hence, the attacker can achieve his maximal impact by forcing the sender to send messages via $2t + 1$ channels.

From the above discussion, we can see that insiders know when to stop jamming to achieve their maximal impact on the system. In this paper, we propose *unpredictable channel assignment* to prevent insiders from knowing when to stop jamming. The main improvement of this mechanism relates to how we handle two-member groups. In other words, we apply one of the existing dynamic-grouping techniques at the beginning, and switch to our approach when the group being jammed is a

two-member group. We call the two nodes in a two-member groups as a *suspicious node pair*.

The high-level idea of our approach is as follows. For each suspicious node pair, we reassign the channels periodically, such that the suspicious node pair has a certain probability to share the newly assigned channel. In our scheme, each receiver only knows of which channel he should listen for broadcast messages but has no idea if that channel is only used by him or shared with the other peer receiver in the pair. In other words, the channel sharing information is unpredictable to the attacker. When the suspicious node pair share the same channel, we can reduce the communication overhead by sending only one copy of each broadcast message, instead of two copies as required in previous approaches. When the two nodes use different channels, and the attacker launches a jamming attack, then the traitor can be detected since the jammed channel has only one user, i.e., the traitor.

In the following, we focus on one suspicious node pair to present and analyze our *unpredictable channel assignment* approach.

6.2.1 Basic Approach

Figure 6.1 shows a flowchart of our proposed scheme. We now follow the flowchart and discuss our solution in detail.

Random Channel Assignment: The sender assigns the node pair new channels every m broadcast messages. For each channel assignment, the sender will assign these two nodes two different channels with probability p. In other words, they will share the same channel with probability $1-p$ as shown in Figure 6.2. The channels are always randomly selected from n available broadcast channels. The sender will update the new channel information to each receiver through their pairwise private channel. Every receiver only knows of which channel he should listen for the broadcast

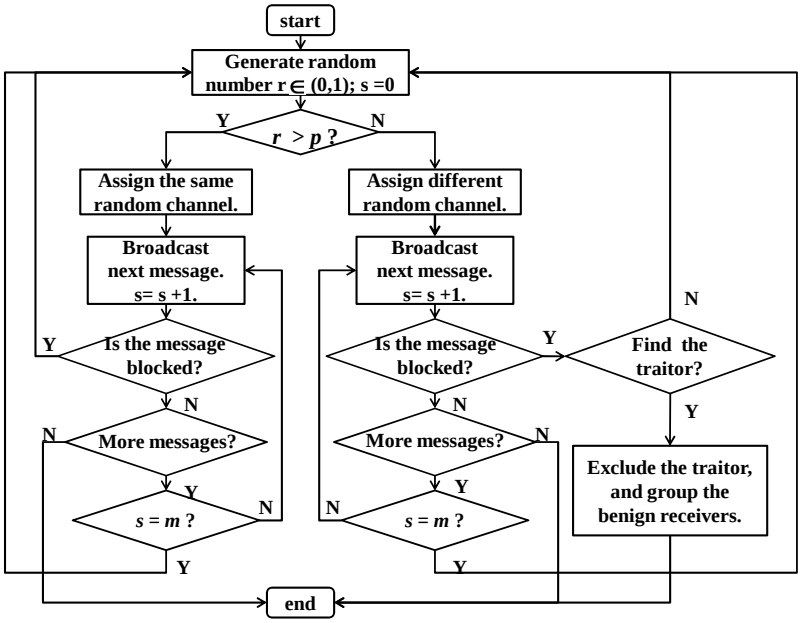


Figure 6.1. The wireless broadcast system with insider jammer detection in Section 6.2 and 6.3.1.

messages; he does not know which channel is used by the other receiver or whether they are sharing the same channel.

Making the channel sharing information unknown to the receiver does not prevent the two nodes in a suspicious node pair from receiving any broadcast message. However, this unpredictability makes the attacker's job much harder. Note that before jamming a channel, a smart insider would want to ensure that there are other users listening to the same channel for broadcast messages. If he is the only user of the jammed channel, he will be immediately detected for jamming. In all existing schemes, the attacker can easily determine whether he is the only user of a given channel. However, in our scheme, it becomes very hard for the attacker to determine this. When two receivers are assigned different channels, jamming the channel only used by the traitor will expose that traitor. We can easily detect and block the traitor from the system. On the other hand, if the attacker does not jam the traitor's channel,

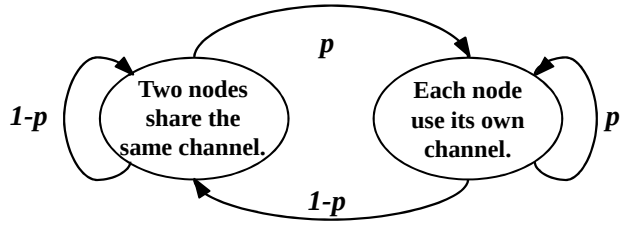


Figure 6.2. Random channel assignment. The probability that each receiver is assigned different channel from each other is p. The probability that the suspicious node pair is assigned the same channel is $1 - p$.

the sender can broadcast messages to both of the receivers through the jamming-free channel and save communication cost. This creates a dilemma for the attacker: if he decides to jam the channel, he will be detected with some probability. If he decides not to jam the channel, the communication cost will be reduced in the system.

Message Broadcast and Channel Reassignment: After sending the new channel assignment information to each receiver in a suspicious node pair, the sender will maintain a counter s to track the number of broadcast messages that have been sent through that channel. If either of the following two conditions is met, the sender will reassign new channels to the node pair: (1) the sender has sent m messages through that channel, i.e., $s = m$; or (2) the message on the assigned channel is blocked due to jamming attack. Otherwise, the sender will continue to use the current channel to broadcast messages until no more broadcast messages exist in the queue, i.e., all broadcast messages have been delivered successfully.

Simple Traitor Detection: As shown in Figure 6.1, when a channel is blocked, the sender will need to determine which of the two nodes is the traitor. We will describe a simple but effective traitor detection scheme in this section and will discuss further enhancements to this scheme in Section6.3.

One simple traitor detection scheme is based on the following observations. If the attacker does not know which channel is used by a particular benign receiver,

it is very hard for him to select and jam the right channel from a large number of potential channels. To increase his success rate, the attacker may simply jam the channel assigned to malicious receivers, expecting to affect the other benign receivers who are sharing that channels. However, if no one else is assigned to use a given channel, we can easily determine the traitor. Therefore, *if the sender assigns node u an unshared channel, and this unshared channel is blocked, then the sender will believe that u is malicious.*

Once the sender determines that one of the nodes in a suspicious node pair is malicious and blocks that node from the system, it will simply assume that the other node is benign. From then on, the sender will put this node into one of the groups that does not suffer from jamming attacks. Certainly, it is possible that the other node is an inactive malicious node, i.e., a compromised node that always behaves like a benign node. However, our approach can handle this very well. This is because once this node becomes active, it will be eventually isolated by using dynamic regrouping [19] and excluded from the system by using our scheme if the attacker chooses to maximize its jamming impact.

The above approach is also applicable to the other suspicious node pairs. We can directly apply this protocol on each of the suspicious node pairs. If the attacker keeps launching the jamming attack using the malicious nodes, we will eventually detect them. Otherwise, we can reduce the communication cost for each pair from two copies in the previous approaches to $1 + p$ copies, as we show in Section 6.2.2.

6.2.2 Analysis

In the following, we study the performance of the basic approach in terms of communication overhead and security.

Communication Cost: We focus on the additional communication cost to tolerate t malicious receivers. Specifically, we calculate the number of extra copies (denoted by c) to be sent for each broadcast message.

The dynamic regrouping schemes [19] confine the malicious nodes into suspicious node pairs. Therefore, a clever attacker will try to maximize its impact by making the number of suspicious node pairs as large as possible. This can be achieved by having each suspicious pair contain only one malicious node. In this case, the total number of suspicious node pairs is t, as shown in [19]. In previous methods, those suspicious node pairs will eventually lead to $2 \times t$ single-member groups. Therefore, for each broadcast message, the sender needs to send $2 \times t$ extra copies of each broadcast message. On the contrary, our approach can reduce the extra communication cost, as shown in the following theorem.

Theorem 4. *In the proposed approach, the expected number of extra copies to be sent for each broadcast message is no more than* $(1+p) \times t$*, i.e.,* $E[c] \leq (1+p) \times t$*, where* t *is the number of malicious receivers.*

Proof. In our protocol, the sender will reassign new channels to each suspicious node pair every m broadcast messages. Note that two nodes will use different channels with probability p. As a result, for each message, we have

$$\Pr[\text{The sender sends 2 copies to each pair}] = p, \text{ and}$$
$$\Pr[\text{The sender sends 1 copy to each pair}] = 1 - p.$$

Since t malicious receivers can introduce at most t suspicious node pairs, we have

$$E[c] \leq (2 \times p + 1 \times (1-p)) \times t = (1+p) \times t.$$

□

Therefore, the proposed approach can reduce the extra communication cost of the existing solutions by $(1-p)/2$. The lower the probability p, the smaller the number of extra copies need to be sent, and thus the more the energy we can save. In addition, it is shown in [19] that the lower bound of the number of extra copies is t in the worst case. Thus, this approach actually pushes limit of jamming-resistant broadcast towards optimal by setting p to a small value. Certainly, a smaller p may lead to a longer delay in detecting malicious insiders. We will revisit this issue later.

Besides the communication cost of broadcasting messages, the sender also needs to periodically update the channel information for each receiver. More specifically, if the attacker stops launching jamming attacks, the sender sends new channel information to both nodes in a suspicious node pair every m broadcast messages. If the attacker keeps jamming the channels before m messages are broadcast, the sender needs to reassign the channels even more frequently. However, as will be shown later, if only one of the suspicious node pair is malicious, the attacker can not launch many jamming attacks, otherwise the traitor is caught. On the other hand, if both of the nodes are malicious, the attacker is not able to use the traitors' channel to jam any other benign nodes. Certainly, the traitors may just try to waste the system's energy by jamming their own channels and forcing the sender to reassign the channel repeatedly. We will present an enhancement protocol to detect and stop this attack quickly in Section 6.3.2.

Detection Rate and Detection Speed: The attacker will try to maximize its jamming impact on the system by letting each suspicious node pair contain one malicious node. Therefore, in the following, we evaluate the performance of traitor detection in a suspicious node pair consisting of one malicious node and one benign node. We are interested in *the probability P_d that we can identify the active malicious node whose assigned channel information is used by the attacker to launch at most x*

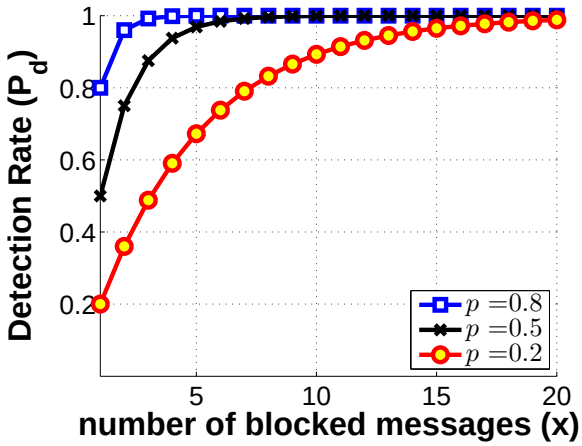


Figure 6.3. The probability P_d that an insider jammer is detected v.s. the total number of messages it can blocked x.

jamming attacks. x is used to evaluate the damage caused by the malicious node. In other words, x implies the speed of detection, and P_d denotes the probability that a traitor u can block x messages at its benign peer v.

Theorem 5. *If a suspicious node pair consist of a benign node and a malicious node, then* $P_d = 1 - (1-p)^x$.

Proof. If the insider attacker jams the channel to block the broadcast messages at the benign node, he will be caught unless the traitor is assigned to the same channel as the benign node. According to the proposed protocol, the probability that the suspicious node pair share the same channel is $(1-p)$. Furthermore, the probability that the attacker succeeds in x rounds of attacks without being caught is $(1-p)^x$. Therefore, $P_d = 1 - (1-p)^x$. □

Figure 6.3 illustrates the relationship between P_d, p, and x. We can see that P_d increases with both p and x. Obviously, the more likely it is the two nodes in a suspicious pair use different channels (larger p), the more likely that the attacker will jam an unshared channel and be exposed as the traitor (larger P_d). In other words, a

larger value of p implies a slower number of jamming attacks the jammer can launch before being caught. On the other hand, according to Theorem 4, a large p implies higher communication cost; thus, p should be small. Nevertheless, our scheme can actually detect traitors quickly even if p is small. According to Figure 6.3, we can see that the active traitor will be caught with very high probability before the 15th round of jamming, even if p is very small.

If both nodes in a suspicious node pair are malicious, the attacker can not jam the benign nodes. The only benefit is that He can keep the sender reassigning the channels frequently. We will make a slight modification of our protocol to address this problem in Section 6.3.2.

Please note we only analyze the performance of detecting the *active* traitor whose assigned channel is used by the attacker to block the broadcast message at the benign node. We do not try to detect the *passive* traitor, whose assigned channel is never used for jamming purpose. Indeed, there is no effective way to identify such malicious node as long as it behaves normally. However, once it becomes active, we can easily catch it.

False Alarm Rate: Theorem 5 shows that the proposed approach can identify the traitor with a very high probability even if the attacker only launches jamming attacks a few times. Thus, to avoid being caught, a cautious attacker may never jam the channels assigned to the malicious insiders. Instead, he will make blind guesses and randomly jam wireless channels, hoping to hit the channels used by some benign nodes and block their broadcast messages. He will succeed when the nodes in the suspicious pair are using different channels. In other words, if he happens to hit the channel used by the benign node in the pair, the sender will make a wrong decision and consider this benign node to be a traitor. Since every node will be assigned a new channel every m broadcast messages, we will study the false alarm rate P_f, i.e.,

the probability that a benign node is identified as a traitor by mistake during the period of transmitting m broadcast messages through its assigned channel.

Theorem 6. *The probability P_f of a benign node being identified as a traitor by mistake during the period of broadcasting m messages is*

$$P_f = p \times \sum_{i=1}^{m} P_{s,i}, \tag{6.1}$$

where $P_{s,i}$ is the probability that the unshared channel of the benign node is jammed for the first time at the i-th broadcast messages. $P_{s,i}$ can be estimated by

$$P_{s,i} = \frac{1 - \sum_{k=1}^{i-1} P_{s,k}}{\frac{n-t}{j} - (i-1)}, \tag{6.2}$$

where $P_{s,0} = 0$.

Proof. Let us calculate $P_{s,i}$ first. A clever attacker will avoid not only the t channels assigned to the t malicious receivers, but also the channels he has already jammed in the previous $i-1$ rounds. Given the attacker's ability of jamming j channels during the period of one message transmission, the probability that node u's unshared channel is jammed can be estimated by $\frac{j}{n-t-(i-1)\times j}$. On the other hand, the probability that u's unshared channel has not been jammed before is $1 - \sum_{k=1}^{i-1} P_{s,k}$. Thus, we have

$$P_{s,i} = \frac{j}{n - t - (i-1) \times j} \times \left(1 - \sum_{k=1}^{i-1} P_{s,k}\right).$$

Obviously, the probability that u's unshared channel is jammed during the period of broadcasting all m messages is $\sum_{i=1}^{m} P_{s,i}$. Moreover, the probability that u's channel is unshared is P. Therefore,

$$P_f = p \times \sum_{i=1}^{m} P_{s,i},$$

□

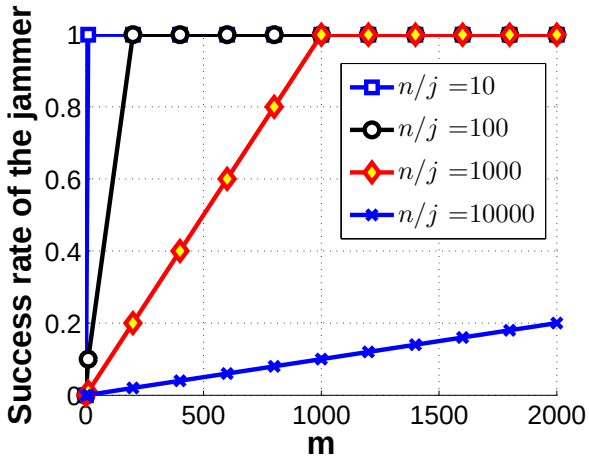


Figure 6.4. The probability that the outsider attacker can successfully block the one-to-one communication system within the period of transmitting m messages.

According to Equations 6.1 and 6.2, P_f is affected by n/j, t/j, p and m. In the following, we will discuss how these parameters affect P_f.

- Impact of n/j: This is the ratio between the number of total available channels in the system and the number of the channels the jammer can block in each round of attack. As a special case in Equations 6.1 and 6.2, if $t = 0$, the attacker is an outsider attacker and does not compromise any receiver. In this case, receivers are all benign and share the same channel. This is equivalent to a *one-to-one* communication system. If p is set to be 1, P_f becomes the probability that the attacker can successfully jam a one-to-one communication channel within the period of transmitting m messages. Figure 6.4 shows the relationship between the probability that the jammer succeeds in blocking the one-to-one communication system and the possible jamming duration (m) with different values of n/j. Obviously, n should be large enough to combat the jammer. In practice, n is usually very large in any multi-channel based jamming-resistant communication system. Thus, in the following analysis, we will n/j to be 10^4.

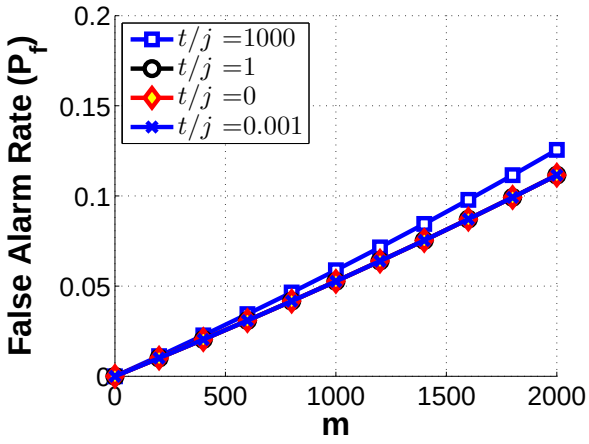


Figure 6.5. Impact of t/j. The false alarm rate P_f is the probability that a benign node is identified as a traitor by mistake. Assume $n/j = 10^4$ and $p = 0.5$.

- Impact of t/j: t is the number of compromised receivers in the system, and we usually have $n \gg t$. Thus, according to Equation 6.2, t/j will have much less impact on P_f than n/j. This is shown in Figure 6.5, which plots the relationship between the false alarm rate P_f and t/j. In the following analysis, we thus set t/j to be 1.
- Impact of p: p denotes the probability that the sender assigns each node in the node pair to a different channel. A smaller p implies a higher probability that the node pair uses the same channel. Since a cautious attacker does not want to expose the malicious node, he will not jam a channel assigned to any of the malicious nodes. For a given suspicious pair, the attacker needs to jam the channel assigned only to the benign node to fool the sender into believing that this node is malicious. Clearly, the larger the value of p is, the higher the chance that the attacker can succeed is. According to Equation 6.1, the false alarm rate P_f increases along with p. This is also shown in Figure 6.6, which plots the relationship between P_f and p. Therefore, we should choose smaller

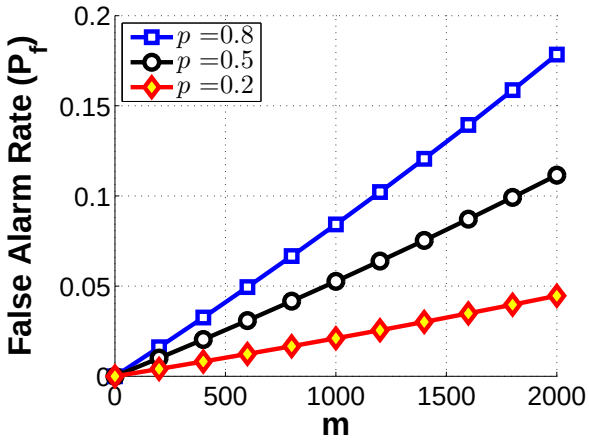


Figure 6.6. Impact of p ($j/t = 10^4$ and $j/t = 1$).

p to get a lower false alarm rate P_f. As we showed earlier, we also choose a smaller p for lower communication cost c.

- Impact of m: m is the number of broadcast messages to be sent before the sender assigns a new channel. From Equations 6.1 and 6.2, and as shown in Figures 6.5 and 6.6, P_f increases with m. Thus, the sender needs to update the channels as frequently as possible. On the other hand, whenever the sender changes broadcast channels, it needs to send the new channel information to each receiver, which also increases the communication cost. Therefore, we need to make m as large as possible while still meeting the security requirement of a small P_f. In Section 6.3.1, we will present a modification to the scheme to guarantee a small P_f even if m is very large.

6.3 Enhancements

In this section, we will discuss some techniques to enhance the accuracy, reliability, scalability, and efficiency of the basic scheme presented in Section 6.2.

6.3.1 Reducing The False Alarm Rate

In the basic scheme, we monitor whether receiver u's unshared channel is jammed. Once this happens, u will be identified as a traitor. However, this decision could be wrong since the attacker may randomly jam wireless channels and happen to hit u's unshared channel. From our previous analysis, the false alarm rate will not be negligible if (1) the system has limited resources (i.e., a small number of n), (2) the attacker is very powerful (i.e., a large number of jammed channels j), or (3) the sender uses the same channel too many times (i.e, a large m). To reduce the false alarm rate, we propose to collect more observations before making a final decision. Thus, instead of making a decision based on a single jamming event, we monitor the frequency that the jamming event occurs.

Our intuition is that even if the attacker happens to jam benign node u's unshared channel once, he can hardly block u's unshared channel repeatedly in a short period of time. Note that we randomly reassign node u a new channel when u's channel is jammed. Thus, we have the following theorem.

Theorem 7. *The probability that the outsider attacker blocks u's channel for at least τ times during w consecutive messages will not exceed $P_{\tau,w}$, which is given by*

$$P_{\tau,w} = 1 - \sum_{i=0}^{\tau-1} \binom{w}{i} (P_J)^i (1-P_J)^{w-i}, \quad where$$
$$P_J = \frac{j}{n-t-(m-1)\times j}.$$

Proof. According to Equation 6.2,

$$P_{s,i} < \frac{1}{\frac{n-t}{j} - (i-1)} < \frac{j}{n-t-(m-1)\times j}.$$

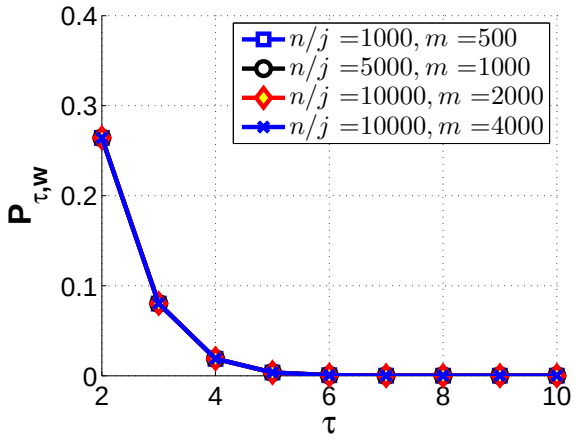


Figure 6.7. The probability of getting at least τ successes in w trials ($\frac{j}{t} = 1$).

Thus, if node u is benign, the probability that its assigned channel is jammed during every broadcast message transmission should not exceed a threshold P_J, where

$$P_J = \frac{j}{n - t - (m-1) \times j}.$$

On the other hand, let us consider the random variable X, which follows the binomial distribution with parameters w and P_J, i.e., $X \sim B(w, P_J)$. The probability of getting at least τ successes in w trials is given by

$$P_{\tau,w} = 1 - \sum_{i=0}^{\tau-1} \binom{w}{i} (P_J)^i (1 - P_J)^{w-i}.$$

Therefore, the probability that the outsider attacker blocks u's channel for at least τ times during w consecutive messages will not exceed $P_{\tau,w}$. □

If we set $w = 1/P_J$, the expected number of times that an outsider attacker can block u's channel during w consecutive messages should be less than one. Therefore,

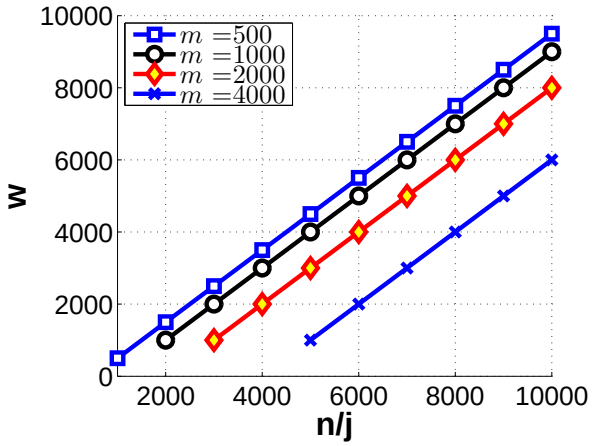


Figure 6.8. The observation window w ($j/t = 1$).

the chance that the attacker blocks u's channel multiple times should be very small. In this case, $P_{\tau,w}$ can be estimated by

$$P_{\tau,w} = 1 - \sum_{i=0}^{\tau-1} \binom{w}{i} \left(\frac{1}{w}\right)^i \left(\frac{w-1}{w}\right)^{w-i}, \quad \text{where} \tag{6.3}$$

$$w = \frac{1}{P_J} = \frac{n-t}{j} - m + 1. \tag{6.4}$$

Let $w = 1/P_J$. Figure 6.7 and Figure 6.8 show $P_{\tau,w}$ under different system parameters. Figure 6.7 shows that $P_{\tau,w}$ decreases quickly with τ. For example, when $\tau = 5$, $P_{\tau,w}$ is as low as 0.0037, and it is very difficult for the outsider attacker to block u's channel for five times during w messages. Thus, if u's channel is jammed for five or more times during w messages, we can identify that u is malicious with a very high accuracy. Based on this observation, we modify the basic scheme as follows.

Protocol Modification: In the modified protocol, whenever node u's message is blocked, we reassign new random channels to u and its peer v and wait for more evidence. We set w according to Equation 6.4, and set τ to be a small number (e.g, five). We then keep track of how many times that u's unshared channel is jammed during the last w broadcast messages transmitted through u's unshared channel. If

this number exceeds τ, u is considered to be malicious and will be removed from the system. As it is still possible though unlikely that node u could be a benign node, another reasonable option is to stop sending messages to u only for the next w messages. In other words, u will be temporarily removed from the system, and will be in the system again after w messages.

Performance Analysis: From previous analysis, the false alarm rate should not exceed $P_{\tau,w}$ given by Equation 6.3 and 6.4. Figure 6.7 also illustrates that $P_{\tau,w}$ is very small with a small value of τ, even if the available channels are limited or the attacker can jam many channels in one message transmission (i.e., a small value of n/j). This enhancement also allows us to reuse the same channel to transmit more messages than the basic scheme (i.e., a large m).

In addition, the proposed enhancement also limits the damage caused by malicious insiders. A compromised node u can only slightly effect its benign peer v, as shown in the following theorem.

Theorem 8. *Malicious node u can only cause its benign peer v to lose $\tau \times (1-p)/p$ messages on average during every w broadcast messages.*

Proof. If u is malicious, the attacker is always aware of u's channel. To block the communication at its benign peer v efficiently, the attacker needs to jam the channel assigned to u so that he can succeed when u and v share the same channel. Let P_{attack} denote the probability that the attacker jams the channel assigned to u, $P_{unshared}$ denote the probability that u's unshared channel is jammed, and P_{shared} denote the probability that u's shared channel is jammed. We have $P_{unshared} = P_{attack} \times p$ and $P_{shared} = P_{attack} \times (1-p)$. Thus, $P_{shared} = P_{unshared} \times (1-p)/p$.

On the other hand, a careful attacker will not jam u's unshared channel τ times. Otherwise, the malicious node u will be identified. Therefore, the expected number of lost messages at v will not exceed $\tau \times (1-p)/p$. □

6.3.2 Dealing with Other Attacks

This section will discuss how to deal with the attacker whose goal is not to block the legitimate communications but to disrupt our protocol.

Attack Description: In our basic protocol, if a suspicious node pair's shared channel is jammed, the sender will simply reassign the new channels and continue to transmit the messages through the new channels, without any punishment. If the attacker puts two malicious nodes in the same pair, he will know whether those two malicious nodes are sharing their channel or not at any time. He may keep on jamming their shared channels and force the sender to reassign the channels frequently. Such an attack only blocks the malicious nodes' own channels and is not very interesting for the attacker who seeks to jam. However, the attacker may be interested in wasting the sender's energy by making him send new channel information continuously. We thus need to *monitor not only the jammed unshared channels but also the jammed shared channels.*

Intuition: In such an attack, the attacker will jam the shared channels with high frequency so that he can force the sender to send new channel information frequently. The attacker also needs to avoid jamming the unshared channels since otherwise the traitor will be detected. On the other hand, if only one of the suspicious node pair is malicious, the jammed channels will be very likely to include both unshared ones and shared ones. Also, among the channels assigned to a suspicious node pair, the ratio between the number of unshared channels and the number of shared channels is about $\frac{p}{1-p}$. Therefore, if the ratio between the number of jammed shared

channels and the number of jammed unshared channels is much larger than $\frac{p}{1-p}$, it is very likely that both of the nodes in the suspicious node pair are compromised.

Protocol Modification: Our protocol should be robust and efficient in all of the following four situations: (i) M_0: neither of the node pair is malicious (i.e., the attacker is an outsider attacker); (ii) M_u: u is malicious; (iii) M_v: v is malicious; (iv) M_{uv}: both nodes in the node pair are malicious. (Please note that those four situations are not mutually exclusive. In fact, $\neg M_0 = M_u \cup M_v$ and $M_u \cap M_v = M_{uv}$.) Specifically, the outsider attacker (M_0) can not fool us into believing M_u, M_v, or M_{uv}. Only one compromised node (either $\neg M_{uv} \cap M_u$ or $\neg M_{uv} \cap M_v$) should not fool us into believing M_{uv}. To guarantee a low decision error rate, we propose to identify M_u, M_v or M_{uv} in the following two steps.

The fist step removes the possibility of M_0. Similar to our first enhancement technique, we make use of Theorem 7. Specifically, we keep tracking of the number $J_{s,w}$ of jammed shared channels of a suspicious node pair u and v in the most recent w (Equation 6.4) messages transmitted through their shared channels. If $J_{s,w} \geq \tau$ (τ is a small integer, e.g., five), at least one of the node pair is malicious. As shown before, it is very difficult for the outsider attack to fool us with this decision rule.

After we know at least one of u and v is malicious, the second step will be to determine M_u or M_v. Moreover, if both M_u and M_v turn out to be true, we know both u and v are malicious (M_{uv}). To identify M_u, let us analyze the situation where only node v is malicious (i.e., $\neg M_{u,v} \cap M_v$). Since we know at least one of u and v is malicious, if we observe an anomalous event that is impossible to occur when only v is malicious ($\neg M_{u,v} \cap M_v$), we can then conclude that u is malicious (M_u). To assist the decision making, we set a very small threshold ε. If the probability of a particular event happening is less than ε, we simply consider it as unlikely to occur. Therefore, ε serves as the maximum decision error rate that the system can tolerate.

Suppose only node v is malicious (i.e., $\neg M_{u,v} \cap M_v$). Let J_v denote the number of v's jammed unshared channels and J_s denote the number of jammed channels shared by u and v. J_s should follow the binomial distribution with parameters $J_v + J_s$ and $1 - P$, i.e., $J_s \sim B(J_v + J_s, 1 - P)$. The probability of getting exactly J_s shared channels is given by the probability mass function:

$$f(J_s; J_s + J_v, 1 - p) = \binom{J_s + J_v}{J_s} (1 - p)^{J_s} p^{J_v}.$$

Since our previous protocol will catch the single malicious node according to its jammed unshared channels, the traitor v will be careful and keep the number of its jammed unshared channels less than the threshold. However, the single malicious node does not know whether its channel is shared or not. Thus, it is quite difficult for it to jam many shared channels (i.e., large J_s) and only a few unshared channels (i.e., small J_v). In fact, when $\frac{J_s}{J_v} > \frac{1-p}{p}$, $f(J_s; J_s + J_v, 1 - p)$ decreases with increasing J_s. Based on this observation, we calculate $f(J_s; J_s + J_v, 1 - p)$ whenever the shared channel is jammed. *If the probability $f(J_s; J_s + J_v, 1 - p) < \varepsilon$, we believe it is unlikely that only v is malicious ($\neg M_{u,v} \cap M_v$).* Since $J_{s,w} \geq \tau$ already shows that it impossible that both u and v are benign, we believe *u is malicious (M_u).* Therefore, based on the above discussion, we have another criterion to identify malicious node u:

Theorem 9. *If (i) $J_{s,w} \geq \tau$, (ii) $\frac{J_s}{J_v} > \frac{1-p}{p}$, and (iii) $f(J_s; J_s + J_v, 1 - p) < \varepsilon$, u is malicious. The false alarm rate of this decision rule is MIN$(P_{\tau,w}, \varepsilon)$.*

Similarly, we also keep tracking of J_u, the number of u's jammed unshared channels. If $J_{s,w} \geq \tau$, $\frac{J_s}{J_u} > \frac{1-p}{p}$, and $f(J_s; J_s + J_u, 1 - p) < \varepsilon$, we know that v is malicious (M_v). Furthermore, if both M_u and M_v are true, M_{uv} is true, i.e., both u and v are malicious.

In summary, the modification is simple. We just need to keep tracking of $J_{s,w}$, J_s, J_u, and J_v, and make decisions according to Theorem 9.

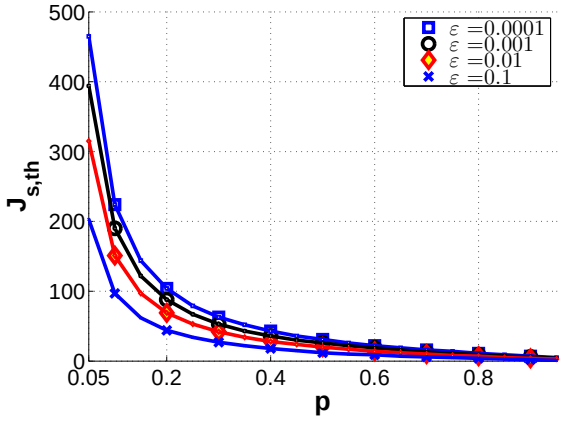


Figure 6.9. The maximum number of shared channels that a malicious node pair can jam before being caught during the period, in which no more than 8 messages are jammed for w consecutive messages transmitted through the unshared channel.

Performance Analysis: Since the attacker aims not to block the communication but to waste the sender's energy, we are interested in finding how many channels the insider attacker can jam before being caught, i.e., the number of channel assignment messages that a pair of malicious nodes can request from the sender. The jammed channels are either shared or unshared between the suspicious node pair. Let $J_{s,th}$ and $J_{un,th}$ denote the maximum number of shared and unshared channels that a malicious node pair can jam before being caught, respectively. Furthermore, $J_{un,th}$ includes $J_{u,th}$ and $J_{v,th}$, the maximum number of u's and v's jammed unshared channels before the traitor is caught, respectively. In other words, $J_{u,th}+1$, $J_{v,th}+1$, and $J_{s,th}+1$ should be the minimum values that will trigger the traitor detection thresholds.

We seek to bound the maximum number of jammed unshared channels. Obviously, when u uses its unshared channel, v also uses its unshared channel; when u uses a shared channel, v also uses the exact same channel. If their unshared channels are jammed at the same time, the sender will reassign the new channel only once. To

reach the maximal possible number of jammed unshared channel, a clever attacker should not jam both u's and v's unshared channel at the same time. Therefore, $J_{uv,th} = J_{u,th} + J_{v,th}$.

According to the traitor detection criterion in Section 6.3.1, neither J_u nor J_v should reach τ during every w consecutive messages transmitted through the unshared channels. Otherwise, either u or v will be identified as the traitor. Thus $J_{uv,th}$ should not exceed the threshold of $2 \times \tau - 2$ during every w consecutive messages transmitted through the unshared channels. In other words, $J_{uv,th}$ may keep increasing along with the total number of transmitted broadcast messages, but its maximum rate of growth is limited to $2 \times \tau - 2$ during every w consecutive messages transmitted through the unshared channels.

The detection criterion in this section mainly limits the number of jammed shared channels J_s. If J_s does not reach τ during w consecutive messages transmitted through the shared channels, it will not trigger the traitor detection mechanism. Such damage is very limited and an aggressive attacker may want J_s to exceed that threshold. However, he must be careful and keep $f(J_s; J_s + J_u, 1 - p) \geq \varepsilon$ and $f(J_s; J_s + J_v, 1 - p) \geq \varepsilon$. Otherwise, either malicious u or v will be identified. When $\frac{J_s}{J_u} > \frac{1-p}{p}$, $f(J_s; J_s + J_u, 1 - p)$ decreases with J_s but increases with J_u. This implies that the more jammed unshared channels there are (either J_u or J_v), the more J_s the attacker can achieve. Thus, the attacker needs to increase the number of jammed unshared channel (J_{un}) to maximize J_s. We are looking for a value of $J_{s,th}$ that satisfies the following criterions:

$$J_{s,th} = \sup\{J_s : f(J_s; J_s + J_{uv,th}, 1 - p) < \varepsilon\},$$

where $\sup\{J_s\}$ denotes the supremum of set $\{J_s\}$.

Similar to $J_{uv,th}$, $J_{s,th}$ may also increase along with the total number of transmitted broadcast messages. In other words, *if the attacker wants to jam more channels, he has to stop blocking the legitimate messages from time to time.* Obviously, the growth of $J_{s,th}$ is limited by the growth of $J_{uv,th}$.

Therefore, this upper bound is given by the following Theorem 10 and illustrated in Figure 6.9.

Theorem 10. *The rate of growth of $J_{s,th}$ is limited by the rate of growth of $J_{uv,th}$. According to $J_{s,th} = \sup\{J_s : f(J_s; J_s + J_{uv,th}, 1-p) \geq \varepsilon\}$, where the growth of $J_{uv,th}$ is limited to $2 \times \tau - 2$ for w consecutive messages transmitted through the unshared channel.*

6.3.3 Summary of Enhancement

In summary, the integrated traitor detection algorithm is given in Procedure 4, and its performance against different types of attackers is listed as follows:

- Outsider Attacker: In this case, neither u nor v is malicious. The probability that the attacker fools us into making a wrong decision is given by Theorem 7.
- Only u or v is malicious: Theorem 8 provides the expected number of jammed channels. The probability that the attacker fools us into believing both nodes are malicious is $\text{MIN}(P_{\tau,w}, \varepsilon)$, as shown by Theorem 9.
- Both u and v are malicious: Theorem 10 provides the maximum number of channels that a malicious node pair can jam before being caught.

6.4 Summary

In this chapter, we propose an adaptive jamming-resistant broadcast system with unpredictable channel assignment. Compared to existing approaches, the proposed approach reduces the communication overhead significantly without increasing

the hardware complexity and scarifying resilience against malicious receivers. Analytic and simulation results show that the proposed approach is both efficient and effective.

We are considering the following two directions to further improve the proposed approach. One is to regroup the suspicious nodes according to the jamming pattern. In this chapter, we do not break a particular suspicious node pair until we identify the insider malicious node. However, even before we have enough evidence to identify the traitor, the existing jamming pattern may already give some clue about each node's likelihood of being malicious. Based on this likelihood, we can regroup all node pairs and rearrange them repeatedly, until the malicious nodes are isolated. Another potential improvement is to adaptively change the value of p according to the jamming pattern. This will make it even more difficult for the attacker to predict the channel assignment. Moreover, we plan to implement the proposed approach on real wireless communication platforms and evaluate it through field experiments.

Procedure 4 TraitorDetection

Input: suspicious node pair (u and v), p, ε, τ, w, m

Output: Identify the malicious nodes.

$J_s \leftarrow 0$, $J_u \leftarrow 0$, $J_v \leftarrow 0$, $NoTraitor \leftarrow$ TRUE
$J_{s,w} \leftarrow 0$, $J_{u,w} \leftarrow 0$, $J_{v,w} \leftarrow 0$
repeat
    $s \leftarrow 0$. Randomly assign new channel C_u to u and C_v to v
    ($\Pr[C_u = C_v] = p$).
    repeat
        For each broadcast message:
        $s \leftarrow s + 1$,
        if $C_u \neq C_v$ **then**
            Update $J_{u,w}$, J_u, $J_{v,w}$, and J_v
            if $J_{u,w} \geq \tau$ **then**
                u is malicious! Stop procedure!
            end if
            if $J_{v,w} \geq \tau$ **then**
                v is malicious! Stop procedure!
            end if
        else // $C_u = C_v$
            Update $J_{s,w}$ and J_s
            if $J_{s,w} \geq \tau$ **then**
                $NoTraitor \leftarrow$ FALSE
            end if
            if $NoTraitor =$ FALSE **then**
                if $(f(J_s; J_s + J_u, 1-p) < \varepsilon) \wedge (\frac{J_s}{J_u} > \frac{1-p}{p})$ **then**
                v is malicious! Stop procedure!
                end if
                if $(f(J_s; J_s + J_v, 1-p) < \varepsilon) \wedge (\frac{J_s}{J_v} > \frac{1-p}{p})$ **then**
                u is malicious! Stop procedure!
                end if
            end if
        end if
    until $(s = m) \vee$(either u or v loses message)
until finish broadcast

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Contributions

In this dissertation, I study the problems in the following two areas: (1) fundamental cryptographic mechanisms in wireless networks, and (2) security mechanisms to combat denial of service (DoS) attacks.

- *Fundamental cryptographic mechanisms in wireless networks:* The communication between two nodes are usually protected with their pairwise key. As one of the most fundamental security services, pairwise key establishment has received a lot of attention recently. However, the existing solutions are either vulnerable to a few compromised sensor nodes or involve expensive protocols for establishing keys.

  We develop novel scheme to establish pairwise keys with the help of a small number of cheap nodes (assisting nodes). This explores a new dimension of using sensor nodes. The theoretical analysis and simulation results demonstrate its significant advantage over existing approaches: it is efficient in terms of storage overhead, communication overhead, and computation overhead; it is resilient to the compromised nodes; it can achieve a very high probability of establishing the keys; it does not require any prior or post deployment knowledge, and is therefore desirable for scenarios where it is difficult to deploy sensor nodes at their expected locations or correctly estimate the sensors' locations after deployment. The implementation also demonstrates the proposed scheme is practical for wireless sensor networks.

- *security mechanisms to combat denial of service (DoS) attacks:* Wireless networks usually run with the limited resources, which makes them easy targets of denial of service attacks. Based on the key management solutions, we develop the efficient and effective mechanisms to mitigate DoS attacks against two valuable resources, battery-supplied energy and wireless communication channels.

  One DoS attack strategy is to keep the victim nodes performing expensive operations and eventually exhaust their limited battery-supplied energy. For example, wireless networks are vulnerable to such attacks, if broadcast authentication is purely based on the expensive digital signature mechanism. The attacker can inject a large number of bogus broadcast messages with invalid signature to waste the victim receivers' energy. We design two pre-authentication filter techniques, a *group-based filter* and a *key chain-based filter*, based on key management schemes. The proposed techniques only require fast and cheap operations to remove bogus messages before verifying the actual digital signatures.

  Wireless communication is particularly vulnerable to signal jamming attacks, especially when the insider attacker exists in the broadcast system. Knowing which channels are used, the attacker can easily inject enough interference signals and block the legitimate communication. We borrow the key management idea and propose adaptive jamming-resistant broadcast systems with partial channel sharing techniques. The proposed schemes not only manage the wireless channels efficiently but also isolate the malicious receivers effectively. We also adopt a Bayes sequential testing technique and push limit of jamming-resistant broadcast towards optimal. Our further improvement avoids increasing the hardware complexity and is able to identify the active malicious receivers.

7.2 Future Work

The research work of this dissertation can lead to other future work as follows:

- *Fundamental cryptographic mechanisms in wireless networks:* In our study, the additional assisting nodes are only deployed to help establish the pairwise keys. It is also interesting to make further use of these nodes to defend the network against attacks. Moreover, assisting nodes need to store hash keys for every other nodes. For a super large-scale network containing millions or even billions of sensor nodes, the storage overhead at assisting nodes will not be ignorable any more. This problem needs further investigation.
- *security mechanisms to combat denial of service (DoS) attacks:* Despite the advantages of our two pre-authentication filter techniques, both of them have limitations. Specifically, the group-based filter allows compromised nodes to send forged messages before they are isolated. Although the key chain-based filter performs better to tolerate compromised nodes, it defers to the group-based filter in the ability to tolerate packet losses. It is interesting to integrate these two techniques together eliminate the limitations.

  In our study of jamming-resistant systems, we believe that the packet loss is always caused by jammers. This is often true when the Media Access Control (MAC) protocol works well. However, we may draw wrong conclusions in reality when the wireless communication is quite unreliable. It is thus highly desirable to conduct a thorough analysis on how channel loss rates impact our approaches.

REFERENCES

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

[2] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and D. Tygar, "SPINS: Security protocols for sensor networks," in *Proceedings of Seventh Annual International Conference on Mobile Computing and Networks (MobiCom)*, July 2001.

[3] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS)*, November 2002, pp. 41–47.

[4] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *IEEE Symposium on Security and Privacy (S&P)*, May 2003, pp. 197–213.

[5] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS)*, October 2003, pp. 52–61.

[6] W. Du, J. Deng, Y. S. Han, and P. Varshney, "A pairwise key pre-distribution scheme for wireless sensor networks," in *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS)*, October 2003, pp. 42–51.

[7] H. Chan and A. Perrig, "PIKE: Peer intermediaries for key establishment in sensor networks," in *Proceedings of IEEE Infocom*, Mar. 2005.

[8] W. Du, J. Deng, Y. S. Han, S. Chen, and P. Varshney, "A key management scheme for wireless sensor networks using deployment knowledge," in *Proceedings of IEEE INFOCOM*, March 2004.

[9] D. Liu and P. Ning, "Location-based pairwise key establishments for static sensor networks," in *2003 ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN)*, October 2003, pp. 72–82.

[10] ——, "Improving key pre-distribution with deployment knowledge in static sensor networks," *ACM Transaction on Sensor Networks (TOSN)*, vol. 1, no. 2, 2005.

[11] Z. Yu and Y. Guan, "A key pre-distribution scheme using deployment knowledge for wireless sensor networks," in *Proceedings of ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, Apr. 2005.

[12] D. Huang, M. Mehta, D. Medhi, and L. Harn, "Location-aware key management scheme for wireless sensor networks," in *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks (SASN)*, October 2004, pp. 29 – 42.

[13] N. Gura, A. Patel, and A. Wander, "Comparing elliptic curve cryptography and RSA on 8-bit CPUs," in *Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, August 2004.

[14] A. Liu and P. Ning, "TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks," in *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*, April 2008.

[15] H. Wang, B. Sheng, C. C. Tan, and Q. Li, "WM-ECC: an Elliptic Curve Cryptography Suite on Sensor Motes," College of William and Mary, Computer Science, Williamsburg, VA, Tech. Rep. WM-CS-2007-11, 2007.

[16] D. J. Malan, M. Welsh, and M. D. Smith, "A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography," in *Proceedings of First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (IEEE SECON 2004)*, october 2004, pp. 71–80.

[17] R. Poisel, *Modern Communications Jamming Principles and Techniques.* Artech House Publishers, 2003.

[18] Y. Desmedt, R. Safavi-Naini, H. Wang, L. Batten, C. Charnes, and J. Pieprzyk, "Broadcast anti-jamming systems," *Comput. Netw.*, vol. 35, no. 2-3, pp. 223–236, 2001.

[19] J. Chiang and Y. Hu, "Dynamic jamming mitigation for wireless broadcast networks," *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pp. 1211–1219, April 2008.

[20] Crossbow Technology Inc., "Wireless sensor networks," http://www.xbow.com/Home/HomePage.aspx, accessed in July 2009.

[21] R. D. Pietro, L. V. Mancini, and A. Mei, "Random key assignment for secure wireless sensor networks," in *2003 ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN)*, October 2003.

[22] J. Hwang and Y. Kim, "Revisiting random key pre-distribution schemes for wireless sensor networks," in *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks (SASN)*, October 2004, pp. 43 – 52.

[23] IEEE Computer Society, "IEEE standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks specific requirements part 15.4: wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs)," *IEEE Std 802.15.4-2003*, 2003.

[24] Texas Instruments Inc., "2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver," http://focus.ti.com/lit/ds/symlink/cc2420.pdf, accessed in January 2008.

[25] Crossbow Technology Inc., "MICAz 2.4GHz Wireless Module," http://www.xbow.com/Products/productdetails.aspx?sid=164, accessed in January 2008.

[26] P. Ning, A. Liu, and W. Du, "Mitigating dos attacks against broadcast authentication in wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 4, no. 1, 2008, to appear.

[27] R. Wang, W. Du, and P. Ning, "Containing denial-of-service attacks in broadcast authentication in sensor networks," in *MobiHoc '07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing.* New York, NY, USA: ACM, 2007, pp. 71–79.

[28] I. F. Akyildiz, W. Lee, M. C. Vuran, and S. Mohanty, "Next generation/dynamic spectrum access/cognitive radio wireless networks: a survey," *Comput. Netw.*, vol. 50, no. 13, pp. 2127–2159, 2006.

[29] K. Fazel and S. Kaiser, *Multi-Carrier and Spread Spectrum Systems: From OFDM and MC-CDMA to LTE and WiMAX.* Wiley, 2008.

[30] K. Cheun, K. Choi, H. Lim, and K. Lee, "Antijamming performance of a multicarrier direct-sequence spread-spectrum system," *Communications, IEEE Transactions on*, vol. 47, no. 12, pp. 1781–1784, Dec 1999.

[31] E. Lance and G. Kaleh, "A diversity scheme for a phase-coherent frequency-hopping spread-spectrum system," *Communications, IEEE Transactions on*, vol. 45, no. 9, pp. 1123–1129, Sep 1997.

[32] S. Zhou, G. Giannakis, and A. Swami, "Digital multi-carrier spread spectrum versus direct sequence spread spectrum for resistance to jamming and multipath," *Communications, IEEE Transactions on*, vol. 50, no. 4, pp. 643–655, Apr 2002.

[33] H. Zhang and Y. Li, "Anti-jamming property of clustered ofdm for dispersive channels," in *Military Communications Conference, 2003. MILCOM 2003. IEEE*, vol. 1, Oct. 2003, pp. 336–340 Vol.1.

[34] M. Simon, J. Omura, R. Scholtz, and B. Levitt, *Spread spectrum communications handbook*. McGraw-Hill, Inc., 2001.

[35] M. Luby, "Lt codes," in *FOCS '02: Proceedings of the 43rd Symposium on Foundations of Computer Science*. IEEE Computer Society, 2002, p. 271.

[36] A. Shokrollahi, "Raptor codes," *IEEE/ACM Trans. Netw.*, vol. 14, no. SI, pp. 2551–2567, 2006.

[37] C. Karlof, N. Sastry, Y. Li, A. Perrig, and J. Tygar, "Distillation codes and applications to dos resistant multicast authentication," in *Proc. 11th Network and Distributed Systems Security Symposium (NDSS)*, 2004.

[38] M. Strasser, C. Pöpper, S. Capkun, and M. Cagalj, "Jamming-resistant key establishment using uncoordinated frequency hopping," in *SP '08: Proceedings of the 2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE Computer Society, 2008, pp. 64–78.

[39] M. Strasser, C. Pöpper, and S. Čapkun, "Efficient uncoordinated fhss anti-jamming communication," in *MobiHoc '09: Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2009, pp. 207–218.

[40] D. Slater, P. Tague, R. Poovendran, and B. Matt, "A coding-theoretic approach for efficient message verification over insecure channels," in *WiSec '09: Proceedings of the second ACM conference on Wireless network security*. ACM, 2009, pp. 151–160.

[41] Y. Liu, P. Ning, H. Dai, and A. Liu, "Randomized differential dsss: Jamming-resistant wireless broadcast communication," in *Proceedings of the 29th IEEE International Conference on Computer Communications (INFOCOM '10)*, 2010.

[42] L. Lazos, S. Liu, and M. Krunz, "Mitigating control-channel jamming attacks in multi-channel ad hoc networks," *WiSec '09: Proceedings of the second ACM conference on Wireless network security*, 2009.

[43] A. Chan, X. Liu, G. Noubir, and B. Thapa, "Control channel jamming: Resilience and identification of traitors," 2007.

[44] P. Tague, M. Li, , and R. Poovendran, "Mitigation of control channel jamming under node capture attacks," *IEEE Transactions on Mobile Computing*, 2009.

[45] C. Hartung, J. Balasalle, and R. Han, "Node compromise in sensor networks: The need for secure systems," U. Colorado at Boulder, Tech. Rep. CU-CS-990-05, Jan. 2005.

[46] J. Newsome, R. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: Analysis and defenses," in *Proceedings of IEEE International Conference on Information Processing in Sensor Networks (IPSN 2004)*, Apr 2004.

[47] Y. Hu, A. Perrig, and D. Johnson, "Packet leashes: A defense against wormhole attacks in wireless ad hoc networks," in *Proceedings of INFOCOM*, April 2003.

[48] B. Parno, A. Perrig, and V. Gligor, "Distributed detection of node replication attacks in sensor networks," in *IEEE Symposium on Security and Privacy*, May 2005.

[49] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," *Journal of the ACM*, vol. 33, no. 4, pp. 792–807, October 1986.

[50] G. Mathur, P. Desnoyers, D. Ganesan, and P. Shenoy, "Ultra-low power data storage for sensor networks," in *Information Processing in Sensor Networks, 2006(IPSN 2006)*, April 2006.

[51] Q. Dong, D. Liu, and P. Ning, "Pre-authentication filters: Providing dos resistance for signature-based broadcast authentication in wireless sensor networks," in *Proceedings of ACM Conference on Wireless Network Security (WiSec)*, 2008.

[52] D. Liu, P. Ning, and W. Du, "Group-based key pre-distribution in wireless sensor networks," *ACM Transactions on Sensor Networks*, 2008, to appear.

[53] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. S. J. Pister, "System architecture directions for networked sensors," in *Architectural Support for Programming Languages and Operating Systems*, 2000, pp. 93–104.

[54] R. Rivest, "The RC5 encryption algorithm," in *Proceedings of the 1st International Workshop on Fast Software Encryption*, vol. 809, 1994, pp. 86–96.

[55] H. Lim and C. Kim, "Multicast tree construction and flooding in wireless ad hoc networks," in *Proceedings of ACM Modeling, Analysis, and Simulation of Wireless and Mobile Systems*, 2000.

[56] W. Peng and X. Lu, "On the reduction of broadcast redundancy in mobile ad hoc networks," in *Proceedings of ACM International Symposium on Mobile and Ad Hoc Networking and Computing*, 2000.

[57] S. Zhu, S. Xu, S. Setia, and S. Jajodia, "LHAP: A lightweight hop-by-hop authentication protocol for ad-hoc networks," in *Proceedings of the Workshop on Mobile and Wireless Network (MWN)*, 2003.

[58] T. Lai, "Nearly optimal sequential tests of composite hypotheses," *The Ananals of Statistics*, vol. 16, no. 2, pp. 856–886, 1988.

[59] M. Li, I. Koutsopoulos, and R. Poovendran, "Optimal jamming attacks and network defense policies in wireless sensor networks," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, May 2007, pp. 1307–1315.

[60] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The feasibility of launching and detecting jamming attacks in wireless networks," in *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2005.

BIOGRAPHICAL STATEMENT

Qi Dong received his B.S. degree in Electrical Engineering from Jilin University, China, in 2002, his M.S. degree in Electrical Engineering from Beijing University of Posts and Telecommunications, China, in 2005, and Ph.D. degree in Computer Engineering from The University of Texas at Arlington in 2010, respectively. His current research interest is security and reliability in distributed systems.