Thwarting Control-Channel Jamming Attacks from Inside Jammers

Sisi Liu, *Student Member, IEEE,* Loukas Lazos, *Member, IEEE,* and Marwan Krunz, *Fellow, IEEE*
Department of Electrical and Computer Engineering University of Arizona, Tucson, AZ, 85721
E-mail:{sisimm, llazos, krunz}@ece.arizona.edu

Abstract—Coordination of network functions in wireless networks requires frequent exchange of control messages among participating nodes. Typically, such messages are transmitted over a universally known communication channel referred to as the *control channel*. Due to its critical role, this channel can become a prime target of Denial-of-Service (DoS) attacks. In this article, we address the problem of preventing control-channel DoS attacks manifested in the form of jamming. We consider a sophisticated adversary who has knowledge of the protocol specifics and of the cryptographic quantities used to secure network operations. This type of adversary cannot be prevented by anti-jamming techniques that rely on shared secrets, such as spread spectrum. We propose new security metrics to quantify the ability of the adversary to deny access to the control channel, and introduce a randomized distributed scheme that allows nodes to establish and maintain the control channel in the presence of the jammer. Our method is applicable to networks with static or dynamically allocated spectrum. Furthermore, we propose two algorithms for unique identification of the set of compromised nodes, one for independently acting nodes and one for colluding nodes. Detailed theoretical evaluation of the security metrics and extensive simulation results are provided to demonstrate the efficiency of our methods in mitigating jamming and identifying compromised nodes.

Index Terms—Jamming, Denial-of-Service, Control channel, Ad hoc networks, Cognitive radio networks.

1 INTRODUCTION

Organizing a collection of nodes into a wireless network requires cooperative implementation of critical network functions such as neighbor discovery, channel access and assignment, routing, and time synchronization. These functions are coordinated by exchanging messages on a broadcast channel, known as the *control channel*. In most network architectures, including mobile ad hoc, vehicular, sensor, cellular, mesh, and cognitive radio networks (CRNs), the location[1] of the control channel, determined by its frequency band, time slot, or spreading code, is known a priori to all nodes participating in the network [2], [22].

From a security standpoint, operating over a globally known control channel constitutes a single point of failure. Networks deployed in hostile environments are susceptible to Denial-of-Service (DoS) attacks by adversaries targeting the functionality of the control channel [6], [9], [25]. If the adversary is successful, network service can be denied even if other available frequency bands remain operational. One of the most effective ways for denying access to the control channel is by jamming it. In this attack, the adversary interferes with the frequency band(s) used for control by transmitting a continuous jamming signal [19], or several short jamming pulses [16].

Typically, jamming attacks have been analyzed and addressed as a physical-layer vulnerability. Conventional anti-jamming techniques rely extensively on spread spectrum (SS) [19]. These techniques provide bit-level protection by spreading bits according to a secret PN code, known only to the communicating parties. An adversary unaware of this code has to transmit with a power which is several orders of magnitude higher compared to the SS transmission, in order to corrupt a SS signal. However, in packet-radio networks, corrupting a few more bits than the correction capability of the error correcting code (ECC) (about 13% of the packet length for WLANs [16]) is sufficient to force the dropping of a data packet. Hence, the adversary need only stay active for a fraction of the time required for a packet transmission. Moreover, targeting the control channel, which typically operates at a low transmission rate, significantly reduces the adversary's effort. In fact, it was shown that the power required to perform a DoS attack in GSM networks is reduced by several orders of magnitude when the attack targets the control channel [6], [25]. Moreover, potential disclosure of cryptographic secrets (e.g., PN codes) by compromised nodes further reduces the adversary's effort. Note that because control information is broadcasted, PN codes must be shared by all intended receivers. The compromise of a single receiver leaves the network vulnerable to low-effort jamming attacks [6], [16], [25], [26]. In this article, we address the problem of *resisting control channel jamming in the presence of compromised nodes*.

Our Contributions–We consider a sophisticated adversary who exploits knowledge of protocol specifications along with cryptographic secrets to efficiently jam the control channel. This channel can be used by any layer in the protocol stack to broadcast control traffic, which could include coordination information needed for protocol operation in upper layers. To quantify the adversary's ability to deny access to the control channel, new security metrics are defined. A randomized distributed channel establishment and maintenance scheme is developed to allow nodes to establish a new control channel using frequency hopping. Under our scheme, network nodes are able to temporarily access a control channel until the jammer is removed from the network. Our method differs from classic frequency hopping in that no two nodes share the same hopping sequence. This allows for unique identification of compromised nodes by nearby ones. Our scheme is suitable for networks with static or dynamic spectrum assignment (e.g., CRNs). For the latter, we propose a modification of the original scheme to take into account the dynamic nature of channel availability in time and space. Assuming perfect random number generators, we analytically evaluate the proposed anti-

A preliminary version of this paper was presented at the ACM WiSec 2009 Conference (full paper)*, March 2009.*

1. In this work, we define the location of the control channel as a frequency band used to broadcast messages for coordinating network functions.

jamming metrics. We verify our analytic results via extensive simulations. Both static spectrum and dynamic spectrum networks are considered and simulated.

The remainder of this article is organized as follows. In Section 2, we state the network and adversarial models, and propose new security metrics for evaluating control channel jamming. Section 3 describes our proposed control channel architecture. In Section 4, we present our randomized distributed scheme for maintaining control communications when the network is under attack. Section 5 describes the process of identifying compromised nodes. Analytical performance evaluation of our scheme is presented in Section 6. In Section 7, we present related work and in Section 8, we summarize our contributions.

2 Model Assumptions and Metrics

Network Model–We consider a wireless ad hoc network. In the case of a static spectrum assignment, the network is assumed to operate over K orthogonal frequency bands. We use the terms frequency, frequency channel, or simply channel interchangeably to denote a separate frequency band. In the case of dynamic spectrum networks, the number of idle channels at time t, denoted by $K(t)$, varies according to primary radio (PR) activity. The maximum number of idle channels is equal to K. Cognitive radio (CR) nodes are capable of sensing the wireless medium and determining the set of idle channels at any given time. Various sensing methods can be used for this purpose [2].

Each node is equipped with a half-duplex transceiver. This is typical for most wireless devices equipped with a single radio[2]. We further consider a time-slotted system. Network nodes are assumed to be capable of slowly hopping between available frequencies bands. For simplicity, we assume that one frequency hop can occur per time slot. Several messages may be exchanged during each slot. We assume that prior trust has been established between network nodes. Neighboring nodes share pairwise symmetric keys that can be used for secure communication and joint secret generation.

Adversarial Model–The goal of the adversary is to drop packets that are transmitted over the control channel. To do so, the adversary deliberately interferes with transmissions on selected frequency bands within a communication range R_{max}. Messages received by any node that is within the jamming range and at the jammed frequency band are assumed to be irrecoverably corrupted. Network nodes are assumed to be capable of detecting jamming attacks if they are within distance R_{max} from the jammer and are tuned to the jammed frequency band. Several methods are available for jamming detection [29], and any of them can be used for our purposes. We further assume that the adversary can physically compromise network devices and recover the content of their memory, including cryptographic secrets such as PN codes. He is also capable of hopping at the same rate as normal network nodes, thus jamming one channel per time slot (slow hopping jammer). This model is suitable when considering that the jammer is aware of the PN codes used for broadcasting. Therefore, he does not need to hop at a faster rate to jam the control channel. Note that with dedicated hardware, the jammer may be able to hop at a much higher rate than that of regular nodes. However, the jammer's hopping rate is limited by the time that he has to remain on a particular band in order to corrupt a sufficient number of bits from the targeted packet(s). Taking into account the interleaving function at the physical layer, this time can represent a significant portion of the slot duration [16].

Anti-Jamming Metrics–Numerous metrics have been proposed in the literature for evaluating jamming resilience. Traditional anti-jamming metrics such as the jamming-to-noise ratio and the jamming gain are mostly relevant under an external threat model. These metrics capture the amount of power needed by the adversary in order to interfere with legitimate transmissions at the physical layer [1], [16]. In our context, an adversary who is aware of a compromised PN code can follow that code in order to jam the control channel without significantly increasing his transmission power relative to the transmitted signal.

MAC layer metrics, such as the packet send ratio (PSR) and packet delivery ratio (PDR), were introduced by Xu et al. [29]. These metrics are useful for detecting a jamming attack, but are not reflective of the ability of our scheme in resuming the control channel operation. Our scheme aims at identifying the set of compromised nodes. This identification is critical for the re-establishment of the control channel. For this purpose, we define the following security metrics.

Definition 1: **Evasion Entropy $\boldsymbol{E}_{\mathbf{i}}$**–Let I_i be a random variable that denotes the frequency of the control channel during slot i. We define the evasion entropy as:

$$E_i = \mathrm{H}(I_i | I_{i-1}, I_{i-2}, \dots I_0)$$

where $\mathrm{H}(X|Y)$ is the conditional entropy of the random variable X given the random variable Y:

$$\mathrm{H}(X|Y) \overset{\triangle}{=} -\sum_y \sum_x \Pr[y] \Pr[x|y] \log_2 \Pr[x|y].$$

Here, $\Pr[y] = \Pr[Y = y]$ and $\Pr[x|y] = \Pr[X = x | Y = y]$.

The evasion entropy measures the uncertainty in the control channel location, given all previously observed locations and any internal knowledge due to node compromise.

Definition 2: **Evasion Delay *D***–The evasion delay is defined as the time between the successful jamming of the control channel and the re-establishment of a new one.

Definition 3: **Evasion Ratio *ER***–The evasion ratio is defined as the fraction of time that the control channel is available for communication, in the presence of the jammer.

3 Design Motivation

In this section, we motivate our approach for establishing and maintaining the control channel. Our method is based on the observation that the scope of control messages is typically confined to the range of the broadcaster (e.g., RTS/CTS messages). For multi-hop networks, broadcasted control messages can be

2. Our schemes can benefit from a full-duplex transceiver design by exploiting concurrent transmissions/receptions of control information in multiple frequency bands, at the expense of increased hardware complexity. We leave the investigation of the properties and performance of our methods under a full-duplex communication model as future work.

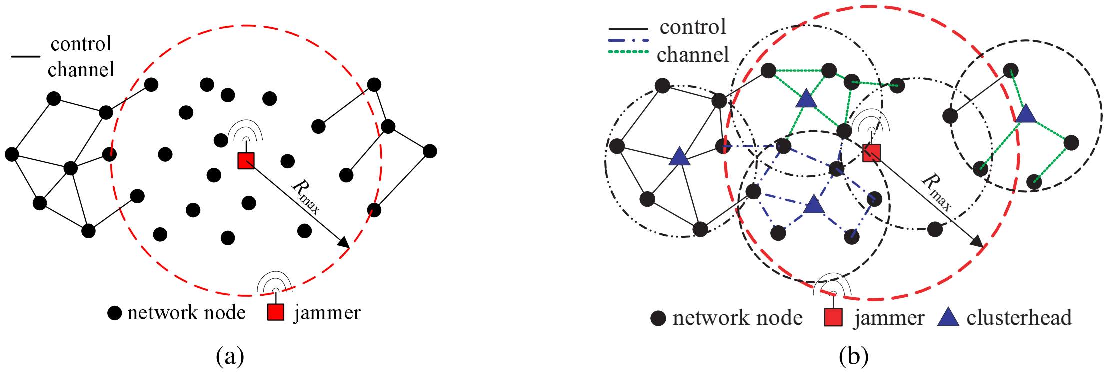


Fig. 1: (a) The adversary blocks all control messages within range R_{max} by jamming a single frequency band, (b) the control channel is located at different channels within each cluster. The impact of the jammer is now confined to clusters within R_{max} that use the jammed channel.

relayed on the same or on a separate frequency band. Allocating different control channels to different neighborhoods within the same collision domain can potentially increase the control-channel throughput due to the reduction in interference between such neighborhoods. Moreover, allocating one unique channel for control has the following significant disadvantages: (a) a long-range transmission can jam the control channel for multiple neighborhoods, (b) the control channel re-establishment process has to be coordinated network-wide, and (c) the compromise of a single node reveals any shared PN codes used for broadcasting.

The impact of long-range jamming attacks can be significantly reduced by varying the control channel in space and time. Such a design also reduces the delay and communication overhead of the control channel re-establishment process, because it requires only local coordination. To mitigate the impact of jamming, we adopt a cluster-based architecture, where the network is partitioned into a set of clusters. Each cluster establishes and dynamically maintains its own control channel. In this design, it is sufficient to ensure that nodes can receive broadcast control messages from members of their own cluster, and that nodes at the boundaries of multiple clusters are aware of the control channels associated with these clusters. The control-channel establishment and maintenance process is facilitated by a clusterhead (CH) node within each cluster. CHs are regular nodes that are temporarily assigned with the responsibility of mitigating jamming, and can be periodically rotated. Several methods are readily available for organizing a wireless network into clusters and electing CHs [31].

In Fig. 1(a), we show an implementation of the control channel using one frequency. All nodes within the jammer's range are denied access to the control channel. In Fig. 1(b), we show a clustered approach where each CH is responsible for the establishment and maintenance of a separate control channel within its cluster. The impact of the jammer is now confined to clusters within R_{max} that use the jammed frequency.

4 Control Channel Implementation

Consider a given cluster, where each node is within the range of the CH. Suppose the current control channel is jammed by an adversary. The main idea behind our scheme is to have each node in the cluster hop between channels in a pseudo-random fashion, following a unique hopping sequence not known to other nodes. If the jammer captures the hopping sequence of a compromised node, then by design this node can be uniquely identified. After identification, the CH updates the hopping sequences of all nodes in the cluster except the compromised one. After this update, the effectiveness of a jammer who exploits knowledge from a compromised node becomes equivalent to the effectiveness of a jammer who hops randomly between channels. Note that our method is not a permanent solution for the control channel allocation, nor can it be used permanently for data communications due to its high communication overhead and delay. Rather, our scheme temporarily maintains control communication until the jammer and any compromised nodes are identified.

The hopping sequences assigned to various nodes are designed to overlap at certain time slots, which represent the control channel. These slots are kept secret. Given the uncertainty in the control channel location, control transmissions must be repeated in several slots to (probabilistically) ensure reception by the intended parties. Our scheme consists of five phases: (a) hopping sequence generation, (b) hopping sequence assignment, (c) control channel access, (d) compromised node identification, and (e) hopping sequence update. For dynamic spectrum networks, an intermediate step is applied to adjust the hopping sequences according to the current channel availability. We now describe each of the above phases.

4.1 Hopping Sequence Generation

By design, the hopping sequences assigned to different cluster members overlap only in a pre-defined number of slots, which are used to implement a broadcast control channel. In order to protect the secrecy of the control channel, the hopping sequences must satisfy the following properties: (a) *high evasion entropy*; knowledge of previous hops does not reveal any information about future ones, and (b) *high minimum Hamming distance*; when interpreted as codewords, any two sequences should have a high Hamming distance so that a compromised node can be identified.

Suppose that the cluster consists of n nodes plus the CH, and let the set of available channels be $\{1, \ldots, K\}$. To construct n hopping sequences of length $L+M$, where M denotes the number of slots implementing the control channel, the CH executes the following steps:

Step 1: Generate n random sequences s_j, $1 \leq j \leq n$, each of

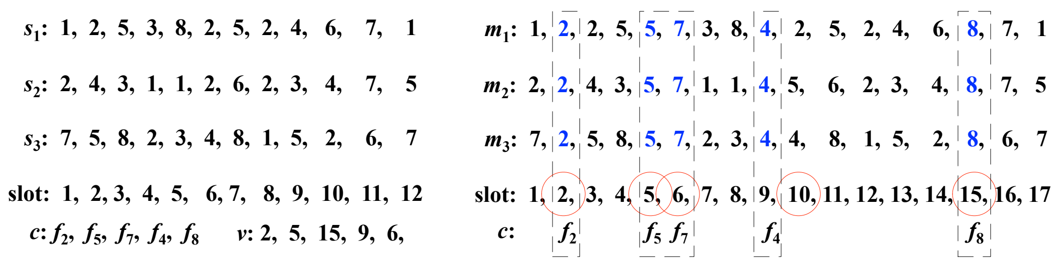

s_1: 1, 2, 5, 3, 8, 2, 5, 2, 4, 6, 7, 1
s_2: 2, 4, 3, 1, 1, 2, 6, 2, 3, 4, 7, 5
s_3: 7, 5, 8, 2, 3, 4, 8, 1, 5, 2, 6, 7
slot: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
c: f_2, f_5, f_7, f_4, f_8
v: 2, 5, 15, 9, 6,
m_1: 1, 2, 2, 5, 5, 7, 3, 8, 4, 2, 5, 2, 4, 6, 8, 7, 1
m_2: 2, 2, 4, 3, 5, 7, 1, 1, 4, 5, 6, 2, 3, 4, 8, 7, 5
m_3: 7, 2, 5, 8, 5, 7, 2, 3, 4, 4, 8, 1, 5, 2, 8, 6, 7
slot: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17
c: f_2 f_5 f_7 f_4 f_8

Fig. 2: Hopping sequence generation for $L = 12, M = 5$ and $K = 8$. The control-channel location vector c is interleaved with the random sequences s_1, s_2, and s_3 at the slot positions indicated by the M-long vector v.

length L. For each sequence $s_j = \{s_j(1), \dots, s_j(L)\}$, we have $\Pr[s_j(i) = k] = \frac{1}{K}$, where $k = 1, 2, \dots, K$.

Step 2: Generate a *random channel location vector* $c = \{c(1), \dots, c(M)\}$, where $\Pr[c(i)= k] = \frac{1}{K}$ for $i = 1, \dots, M$, and $k = 1, 2, \dots, K$.

Step 3: Generate a *random slot position vector* $v = \{v(1), \dots, v(M)\}$, where $v(i) \in \{1, \dots, L + M\}$, with $v(i) \neq v(j),\ \forall i \neq j$.

Step 4: In every sequence s_j, insert element $c(i)$ before element $s_j(v(i))$ to generate a new sequence m_j.

In Fig. 2, we show an example of the hopping sequence generation phase for three nodes, with $L = 12, M = 5$, and $K = 8$. Here, the indexes $\{1, \dots, 8\}$ correspond to eight frequency bands $\{f_1, \dots, f_8\}$. In Step 1, three random sequences s_1, s_2, and s_3 of length $L = 12$ are generated, with $s_j(i) \in \{1, \dots, 8\}$. In Step 2, a random channel location vector c of length $M = 5$ is generated with $c(i) \in \{1, \dots, 8\}$. This vector indicates the frequency bands of the control channel. In Step 3, the random slot position vector v is generated. This vector indicates the five slots where the control channel is implemented. In Step 4, the sequences m_1, m_2, and m_3 are obtained based on s_1, s_2, s_3, c and v. Note that because the m_j's are a result of random interleaving of random sequences, *they are also random.* However, the sequences $m_j,\ 1 \leq j \leq n$, are not mutually independent, because c is interleaved in specific slots of all sequences. Despite this fact, it still holds that knowledge of one sequence does not reveal any information regarding the other. This is due to the fact that the vector v, which indicates the slot locations where two hopping sequences overlap by design, is not known to non-CH nodes. Therefore, by knowing one hopping sequence m_j, one cannot predict the other sequence.

4.2 Generation for Dynamic Spectrum Networks

In dynamic spectrum networks, the set of channels available for use varies temporally and spatially. Consider a CRN. Suppose that the nodes are assigned hopping sequences m_j's, generated as in Section 4.1. Denote channel availability during time slot i by $\mathcal{F}_i = \{ch_i(1), ch_i(2), \dots, ch_i(K(i))\}$, where $K(i)$ is the number of idle channels during slot i, $K(i) \leq K$. Here, $ch_i(j)$ denotes the index of the jth idle channel, $ch_i(j) \in \{f_1, \dots, f_K\}$. The set of idle channels in each time slot can be determined by the underlying channel sensing process [2], with all nodes in a particular cluster agreeing on the same set [10]. However, two different clusters may have two different sets of idle channels. To adjust m_j to a hopping sequence m'_j for dynamic spectrum networks, each cluster node executes the following steps.

Step 1: Determine the channel availability set for time slot i : $\mathcal{F}_i = \{ch_i(1),\ ch_i(2), \dots, ch_i(K(i))\}$.

Step 2: Map index $m_j(i)$ to index $m'_j(i) = m_j(i) \pmod{K(i)} + 1$.

Step 3: Access frequency band $\mathcal{F}_i(m'_j(i))$.

The following example illustrates the above procedure. Consider three CRs that have been assigned the hopping sequences in Fig. 2. Suppose that for slot 4, the set of idle channels is $\mathcal{F}_4 = \{f_2, f_3, f_5, f_7, f_8\}$ $(K(4) = 5)$. CR$_1$ executes Step 2 above and computes $m'_1(4) = [m_1(4) \pmod{K_4})] + 1 = [5 \pmod 5] + 1 = 1$. In Step 3, CR$_1$ determines the next hop to be $\mathcal{F}_4(1) = f_2$. Similarly, CR$_2$ determines $m'_2(4) = [m_2(4) \pmod{K_4}] + 1 = [3 \pmod 5] + 1 = 4$, which denotes the 4th channel in the idle channel list, i.e., f_7. CR$_3$ hops to the same channel though its original index $m_3(4) \neq m_2(4)$. Suppose now that the set of idle channels for slot 7 has changed to $\mathcal{F}_7 = \{f_2, f_5, f_6, f_7\}$ (in reality, PR activity varies at a much slower rate compared to the scale of time slots). The CRs adjust their sequences to the current set of idle channels. The resulting sequences are shown in Fig. 3.

4.3 Hopping Sequence Assignment

The hopping sequences generated by the CH are assigned to individual cluster nodes via secure pairwise communication. Using pre-shared pairwise keys, the CH can establish pairwise shared PN codes with the members of its cluster. Note that the compromise of a cluster node only reveals the PN code shared between that node and the CH, while the rest of the pairwise PN codes remain secret. Thus, these codes can be used for jamming-resistant pairwise communication (but not for broadcasting of control information). The steps of the hopping sequence assignment for a node n_j are as follows:

Step 1: The CH and node n_j establish a pairwise PN code (this code can be either preloaded or generated based on a pairwise key K_{CH,n_j}).

Step 2: The CH provides n_j with the hopping sequence m_j, encrypted using the pairwise key K_{CH,n_j}. Message integrity is achieved through a message authentication code (MAC).

Step 3: Node n_j erases from its memory any information regarding the identity of the CH.

Step 3 ensures that after PN code assignment, the identity of the CH becomes a secret. Hence, an adversary who may later

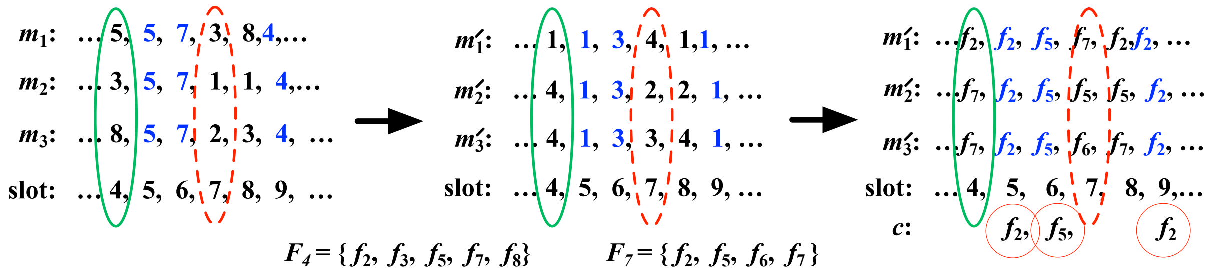

m_1: ... 5, 5, 7, 3, 8, 4, ...
m_2: ... 3, 5, 7, 1, 1, 4, ...
m_3: ... 8, 5, 7, 2, 3, 4, ...
slot: ... 4, 5, 6, 7, 8, 9, ...
m'_1: ... 1, 1, 3, 4, 1, 1, ...
m'_2: ... 4, 1, 3, 2, 2, 1, ...
m'_3: ... 4, 1, 3, 3, 4, 1, ...
slot: ... 4, 5, 6, 7, 8, 9, ...
m'_1: ... f_2, f_2, f_5, f_7, f_2, f_2, ...
m'_2: ... f_7, f_2, f_5, f_5, f_5, f_2, ...
m'_3: ... f_7, f_2, f_5, f_6, f_7, f_2, ...
slot: ... 4, 5, 6, 7, 8, 9, ...
c: f_2, f_5, f_2
$F_4 = \{f_2, f_3, f_5, f_7, f_8\}$
$F_7 = \{f_2, f_5, f_6, f_7\}$

Fig. 3: Adjusting the hopping sequences to account for dynamic channel availability.

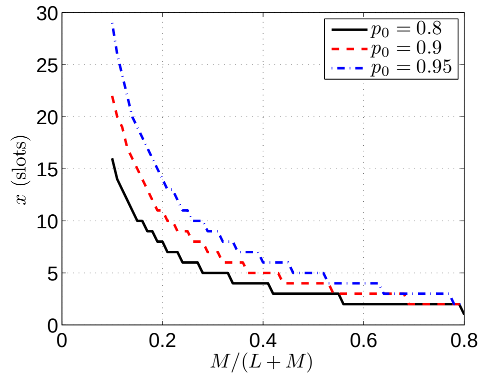


Fig. 4: Number of slots required for accessing at least one control channel slot with probability p_0 as a function of the ratio $\frac{M}{L+M}$.

on compromise n_j cannot selectively target the compromise of the CH. Note that once hopping sequences are assigned, cluster nodes need not know the CH identity in order to access the control channel. In any case, the CH can prove his role to various nodes by using his knowledge of the PN codes that were assigned to individual nodes during the assignment phase. Any cluster node attempting to impersonate the CH would fail to “authenticate” itself, because it is not aware of the PN codes originally assigned.

4.4 Control Channel Access

The hopping sequences assigned to cluster members are designed to implement the control channel only during the slots indicated by the random slot position vector v. To prevent an adversary who compromises one cluster node from jamming the control slots, we require by design that v is not known to cluster nodes. Hence, nodes are not aware of which time slots solely belong to the control channel. To broadcast a control message, a node must repeat its transmission over consecutive slots. The goal here is to ensure that a control-channel slot is accessed at least once during this repetitive transmission. Let x denote the number of retransmissions of a control message. An appropriate value for x can be probabilistically computed based on the design parameters of the hopping sequences. That is, we can tune x such that a control-channel slot occurs with a desired probability p_0. The probability that the number of occurrences z of a control slot is larger than one in a total of x slots is

$$\Pr[z \geq 1] = 1 - \Pr[z = 0] = 1 - \left(\frac{L}{L+M}\right)^x. \tag{1}$$

Setting $\Pr[z \geq 1] \geq p_0$ and solving for x yields

$$x \geq \lceil \frac{\log(1-p_0)}{\log L - \log(L+M)} \rceil. \tag{2}$$

Fig. 4 depicts x versus $\frac{M}{L+M}$ for various values of p_0. We observe that for $\frac{M}{L+M} \geq 0.5$, fewer than 5 repetitions are required for all three values of p_0. For smaller values of $\frac{M}{L+M}$, the required number of slots x increases up to 28 slots when $\frac{M}{L+M} = 0.1$ and $p_0 = 0.95$. The ratio $\frac{M}{L+M}$ controls the tradeoff between the efficiency of the broadcast communication and resilience to jamming under node compromise. A higher ratio decreases the necessary number of retransmissions for a successful broadcast, but also increases the time needed for the identification of a compromised PN sequence.

Note that several cluster nodes may want to broadcast a control message during the same control slot. Although we do not specify the MAC mechanism for coordinating access to this common slot, well-known multiple access techniques, ranging form pure random access to p-persistent CSMA protocols to CSMA with virtual carrier sensing, can be employed. Broadcast control messages are not acknowledged so as to avoid an ACK implosion situation [27]. This is in line with typical wireless protocols such as the 802.11 family. Therefore, a transmitting node does not know if its transmission was performed over a control-channel slot or whether the transmission attempt was successful. For this reason, the node must repeat such a transmission x times.

Upon the successful transmission of a broadcast control packet on a slot that belongs to the control channel, all n-odes are able to correctly receive that packet. Since packets are transmitted/received in the context of a particular proto-col/application/network function, they are accordingly passed on to upper layers of the network stack. Note that cluster nodes may receive multiple copies of the same control packet, because trans-missions are repeated on multiple slots, and multiple sequences may coincide on slots other than the control channel slots. This replication of information is indicated by the inclusion of the same sequence number on the copies of the same packet (e.g., at the MAC layer header). That is, a node repeating the broadcast of the same control packet on multiple slots, keeps all fields of the packet identical. Hence, using the sequence number field, cluster nodes can reject multiple copies of the same control packet.

4.5 Hopping Sequence Update

Hopping sequences need to be updated when the CH detects that a node has been compromised. In this case, the CH is responsible for assigning new sequences to all uncompromised nodes. To do so, the CH synchronizes with the PN code of each individual node, prove his role as a CH, and assigns a new PN code. In detail, the following steps are executed:

Step 1: The CH synchronizes with PN code m_j (m_j is only known to the CH and n_j).

Step 2: The CH communicates to n_j a portion of m_j, which is meant to prove the CH's knowledge of m_j. This communication is secured by the pairwise key K_{CH,m_j}.

Step 3: The CH assigns a new sequence m_j^* to n_j. This communication is secured by the pairwise key K_{CH,m_j}.

Step 4: Node n_j erases all information regarding the identity of the CH.

The hopping sequence update phase differs from the initial hopping sequence assignment phase in the pairwise PN code used for communication. After the initial assignment, cluster nodes hop according to their m_j's. Hence, the CH has to follow each m_j to individually communicate with each node. Note that the compromise of node n_j does not reveal sequence $m_\ell, \ell \neq j$. Hence, the jammer cannot prevent the update of non-compromised nodes. The case of a CH compromise, which reveals all hopping sequences to the adversary, is addressed through CH rotation, as detailed in Section 5.3. Once a CH rotation has occurred, the new CH updates the hopping sequences of all cluster nodes by following the initial hopping sequence assignment process.

In Step 2, the CH proves his role to every cluster member that is assigned a new sequence. This step is necessary because information regarding the CH identity is erased after the initial hoping sequence assignment. Note that the pairwise key shared between the CH and a cluster node n_j is not sufficient to authenticate the role of CH. Other cluster nodes may share pairwise keys with n_j. To avoid CH impersonation, the CH exploits his knowledge of the unique hopping sequences assigned to each node. When updating node n_j, the CH securely communicates part of the current sequence m_j (future hops) to n_j. Upon reception of a correct partial sequence, n_j will accept the sequence update performed in Step 3. After the successful assignment of m_j^*, node n_j erases the identity of CH from its memory (n_j is an uncompromised node and hence, will conform to Step 4). Steps 1-4 have to be repeated by every legitimate node in the cluster, leading to the isolation of the compromised node(s).

5 Identification of Compromised Nodes

In this section, we develop algorithms for the identification of compromised nodes. We first address the case of one compromised node, and then extend the treatment to multiple ones.

5.1 Compromise of a Single Node

Suppose that one cluster member n_j has been compromised. The adversary acquires the unique hopping sequence m_j assigned to this node. Because the slots implementing the control channel are secret, the adversary must follow m_j to efficiently jam the control channel. However, following m_j reveals the identity of the compromised node n_j. This identification is based on the Hamming distance between the sequences assigned to nodes and the jamming hopping sequence. In the following two propositions, we analytically evaluate the expected Hamming distance.

Proposition 1: For two random and independently generated sequences m_j and m_ℓ, defined over an alphabet $\mathcal{A} = \{1, \ldots, K\}$, the expected Hamming distance $\mathrm{E}[d(m_j \ , m_\ell)]$ as a function of the sequence length X is given by

$$\mathrm{E}[d(m_j, m_\ell)] = \frac{K-1}{K} X. \tag{3}$$

Proof: The proof is provided in Appendix 1. □

If the adversary has not compromised any node and is hopping according to a random sequence m_{jam}, the average Hamming distance between m_{jam} and any of the assigned sequences $m_j, \ 1 \leq j \leq n$, must increase at a rate of $\frac{K-1}{K}$. On the other hand, if m_{jam} is a subset of the sequence m_j of a compromised node n_j, the Hamming distance between m_{jam} and m_j is expected to be significantly lower (note that although the adversary may be aware of m_j, he may choose to follow only a subset of it to avoid being identified). The CH can exploit this observation to identify the compromised node.

In dynamic spectrum networks, the hopping sequences are not necessarily random, because of their adjustment to account for spectrum availability. Randomness is preserved only when the number of idle channels $K(i)$ is a factor of the alphabet size that was used to generate the original sequences. In the general case, the expected Hamming distance is expressed by Proposition 2.

Proposition 2: Consider two random and independently generated sequences m_j and m_ℓ that are defined over an alphabet $\mathcal{A} = \{1, \ldots, K\}$. Suppose that the sequences are adjusted to m'_j and m'_ℓ, respectively, according to the process outlined in Section 4.2. The expected Hamming distance $E[d(m'_j, m'_\ell)]$ as a function of the length X of the sequences is

$$\begin{aligned}\mathrm{E}[d(m'_j, m'_\ell)] \quad &= \quad \Bigg(1 - (K(i) - y_K) \cdot \left(\frac{x_K}{K}\right)^2 \\ &\qquad - y_K \cdot \left(\frac{x_K + 1}{K}\right)^2 \Bigg) \cdot X \end{aligned} \tag{4}$$

where $x_K \triangleq \lfloor \frac{K}{K(i)} \rfloor$ and $y_K \triangleq [K \pmod{K(i)}]$.

Proof: The proof is provided in Appendix 2. □

Identification process: For identification purposes, the CH exploits his knowledge of the subsequences s_j, which are unique to individual nodes. Let s_{jam} denote the subsequence followed by the jammer, excluding the slot positions in vector v. To identify a compromised node, the CH measures the Hamming distance between s_{jam} and every assigned sequence s_j. Note that the half-duplex transceiver assumption limits the monitoring capabilities of the CH to a single channel per slot. Because the hopping sequence s_{jam} is not known in advance, the CH periodically tunes to s_j's of different nodes to compute the Hamming distance. To do so, the CH needs only to know if channel $s_j(i)$ was jammed at slot i. We now describe the steps for the identification process for a single compromised node. The pseudo-code is shown in Algorithm 1.

Algorithm 1 Identification of a Single Compromised Node

```
1: $Initialize: d(s_j, s_{jam}) = 0,\ \forall j;\ j = 1;\ i = 0;\ CN \leftarrow \emptyset$
2: while $J$ ==FALSE do
3:    for $x = 1,\ x \leq X,\ x++$ do
4:       if $m_j(i)$ NOT JAMMED & $m_j(i) \notin v$ then
5:          $d(s_j, s_{jam}) = d(s_j, s_{jam}) + 1$
6:       end if
7:       if $d(s_j, s_{jam}) < \mathrm{E}[d(s_j, s_{jam})] - \delta_x$ && $x > \gamma_0$ then
8:          $J$ =TRUE, $CN \rightarrow n$, break
9:       else
10:         $i++$
11:      end if
12:   end for
13:   if $J$ ==TRUE then
14:      break
15:   else
16:      $j++$
17:   end if
18: end while
19: return $CN$
```

Step 1: Initialize $d(s_j, s_{jam}) = 0,\ \forall j$.

Step 2: Synchronize with the hopping sequence m_j of a randomly selected node n_j.

Step 3: For each slot i, $i \notin v$, if $m_j(i)$ is not jammed, set $d(s_j, s_{jam}) = d(s_j, s_{jam}) + 1$.

Step 4: After some number of slots $X \geq \gamma_0$, if $d(s_j, s_{jam}) < \mathrm{E}[d(s_j, s_{jam})] - \delta_X$, then node n_j is considered to be compromised.

Step 5: Randomly pick another node and repeat Steps 2-4 for X slots.

In Algorithm 1, each node is monitored for at least γ_0 slots to obtain an accurate estimate of $d(s_j, s_{jam})$.[3]. The tolerance margin δ_X is computed based on the standard deviation σ_X of the Hamming distance. For example, considering $\delta_X = 3\sigma_X$ yields a 99.7% chance for the Hamming distance of two random sequences to be within that margin. The value of δ_X provides a tradeoff between the speed of identification (smaller δ_X yields tighter bounds on $\mathrm{E}[d(s_j, s_{jam})]$) and the false-positive identification rate In the case of static spectrum assignment, the standard deviation of $d(s_j, s_{jam})$ is equal to $\sigma_X = \sqrt{\frac{K-1}{K^2}X}$. The value of σ_X for dynamic spectrum networks can be easily computed based on eq. (4), in Appendix 2.

We emphasize that the value X takes into account only slots that belong to subsequences s_j. In reality, the delay in the identification of compromised nodes is larger due to the interleaving of common control slots that belong to c. The latter slots do not contribute to the identification process and hence, are excluded from the computation. As an example, consider the hopping sequences shown in Fig. 2. Let $s_{jam} = s_2$, and $\gamma_0 = 10$. For $X = 10$, $\mathrm{E}[d(s_j, s_\ell)] = 8.75$, $\delta_X = 3$, and $\sigma_X = 3.1$. Initially, the CH follows s_1 for $X = 10$ slots. After the first ten slots, $d(s_1, s_{jam}) = 8$. The CH switches to sequence s_2. Because s_2 is the jamming sequence, $d(s_2, s_{jam}) = 0$. Thus, node n_2 is declared compromised. For this set of parameters, the jammer can avoid detection, only if he partially follows s_2 for a fraction $\alpha(X) \leq 40\%$ of the monitoring interval X. As X increases, the fraction $\alpha(X)$ converges to its expected value in the case of random jamming. This can be easily shown from the detection condition of Step 4 of the identification algorithm. Assuming an adversary which is active only for a fraction $\alpha(X)$ of the X slots corresponding to a compromised s_j and setting $\delta_X = \tau\sigma_X$, where τ denotes some desirable constant, the detection condition yields

$$\begin{aligned} d(s_j, s_{jam}) &= \mathrm{E}[d(s_j, s_{jam})] - \delta_X \Rightarrow \\ (1-\alpha(X))X &= \frac{K-1}{K}X - \tau\sqrt{\frac{K-1}{K^2}X} \Rightarrow \\ \alpha(X) &= \frac{1}{K} + \tau\sqrt{\frac{K-1}{K^2 X}}, \end{aligned} \tag{5}$$

where we have used the fact that $d(s_j, s_{jam}) = (1-\alpha(X))X$, when the jammer is following only a fraction $\alpha(X)$ of a compromised sequence s_j. From (5), it is evident that $\alpha(X) \rightarrow \frac{1}{K}$ when $X \rightarrow \infty$. This is a fairly intuitive result that indicates that with the progression of the monitoring period X, a jammer that partially follows a compromised PN code, cannot deviate from the behavior of a random jammer without being detected.

An implicit assumption of our identification process is that the CH is able to detect when a jamming signal interferes with the reception of a control message within his cluster. It is possible for the jammer to tune his transmission power so as to interfere with the reception at a cluster node, but not at the CH. Such a low-power jammer has a limited impact on the network due to his small jamming range. We are primarily concerned with the scenario presented in Fig. 1, in which a high power jammer attempts to deny the control channel within a large network area.

5.2 Compromise of Multiple Nodes

When several nodes are simultaneously compromised, the jammer can combine the acquired hopping sequences to reduce the number of jammed slots. Without loss of generality, assume that nodes $n_1, \ldots, n_q$, $q < n$, are compromised. Suppose that the jamming sequence s_{jam} consists of the (time, frequency) pairs that are common to compromised sequences $\{m_1, \ldots, m_q\}$, excluding the slots in v. In the case of static spectrum networks, the expected length of s_{jam} is given by Proposition 3.

Proposition 3: The expected length $\mathrm{E}[X]$ of a sequence s_{jam} consisting of the channel locations common to q random hopping sequences $\{s_1, \ldots, s_q\}$ of length X is

$$\mathrm{E}[X] = \left(\frac{1}{K}\right)^q X. \tag{6}$$

Proof: The probability that q random sequences overlap at slot i is $\left(\frac{1}{K}\right)^q$. For the random sequences $\{s_1, \ldots, s_q\}$, the expected number of overlapping channel locations is expressed by the expected number of successes in repeating X Bernoulli trials with parameter $\left(\frac{1}{K}\right)^q$. □

3. Faster identification of the compromised nodes can be achieved if the CH evaluates the Hamming distance of all sequences s_k for which $s_k(i) = s_j(i)$, on slot i, when the CH follows the sequence s_j. This method is expected to speed up the identification process by a factor of $\frac{1}{K}$.

Note that the adversary cannot differentiate between the slot positions assigned to the control channel and the $\left(\frac{1}{K}\right)^q L$ slot positions that match due to the s_j's. Hence, the adversary must jam all slots common to the compromised sequences to efficiently deny access to the control channel. For the case of dynamic spectrum networks, the expected length of the sequence s_{jam} is given by Proposition 4.

Proposition 4: The expected length $\mathrm{E}[X]$ of a sequence s_{jam} consisting of the channel locations common to q hopping sequences $\{s'_1, \ldots, s'_q\}$ of length X is

$$\mathrm{E}[X] = \sum_z \left(y_K \left(\frac{x_K + 1}{K} \right)^q + (K(z) - y_K) \left(\frac{x_K}{K} \right)^q \right) X_z \tag{7}$$

where z denotes the number of channel availability changes over the course of $X = \sum_z X_z$ slots, and X_z denotes the number of slots of the sequences s_j for which the number of idle channels is equal to $K(z)$.

Proof: The proof follows similar steps to the proof of Proposition 3 and is omitted. □

To identify compromised nodes, the CH correlates the random sequences s_j, $1 \leq j \leq n$, with the jammed channel locations. The CH follows a monitoring sequence s_{CH}, which is a concatenation of subsequences from the s_j's.

$$s_{CH} = s_1(1:Y)||\ldots||s_n((n-1)Y+1:nY)$$

where Y denotes the number of slots that belong to the subsequences s_j and are devoted to monitoring a node. Note here that our computations ignore slots that belong to vector v. In reality, to monitor a subsequence s_j for Y slots, the CH must monitor n_j for $(1 + \frac{M}{L+M})Y$ slots, on average. The CH maintains a matrix

$$A = \{a_{ji} | a_{ji} \in \{0,1\}\}_{n \times nY} \tag{8}$$

where each row j corresponds to node n_j and each column i corresponds to the ith slot. Note that only non-control slots are taken into account in the construction of matrix A. For A, an element $a_{ji} = 1$ if while residing on channel f during the ith slot, the CH detects f as jammed and $s_j(i) = f$. Otherwise, $a_{ji} = 0$. Considering each row $A(j)$ as a codeword, the CH computes the codeword weight $W(A(j))$ $\forall j$, and ranks the weights in a descending order. The weight $W(C)$ of a binary codeword C is defined as the number of ones in the codeword. Compromised nodes are expected to have a significantly larger weight than other nodes and their weights will be of the same order. Assuming q compromised nodes, the expected weight for a codeword $A(j)$, $j = 1, \cdots, q$ is

$$\mathrm{E}[W(A(j))] = \left(\frac{1}{K}\right)^q qY \tag{9}$$

where $\mathrm{E}[W(A(j))]$ is computed over the qY slots devoted to the monitoring of the q compromised nodes. The CH identifies the set of nodes with high weights and compares those weights to their expected value, expressed in (9). If the weight of a codeword exceeds the expected value by some margin δ_q, i.e., $W(A(j)) \geq \left(\frac{1}{K}\right)^q qY + \delta_q$, the corresponding node n_j is identified as compromised. The parameter δ_q is a tolerance margin related to the standard deviation of $W(A(j))$. The pseudo-code for the identification of multiple compromised nodes is shown in Algorithm 2.

Algorithm 2 Identification of Multiple Compromised Nodes

```
1: $Initialize: A = 0,\ W = 0,\ j = 1,\ j =$FALSE$, CN \leftarrow \emptyset$
2: $v, n;$
3: $m_{CH} = s_1(1:Y)||\ldots||s_n((n-1)Y+1:nY)$
4: while $J ==$FALSE do
5:     if $m_{CH}(i)$ JAMMED & $i \notin v$ then
6:         $a_{ji} = 1,\ W(j)++,\ \forall i,\ \exists\ s_{CH}(i) = s_j(i),\ i++$
7:     end if
8:     if $i > \gamma_1$ // sufficient sampling then
9:         sort($W$) // sort weights in a descending order
10:        find $j,\ \exists\ W(A(j)) - \mathrm{E}[W(A(j)) > \delta_q,\ CN \leftarrow j$
11:        $J =$TRUE
12:    end if
13: end while
14: return $CN$
```

During the execution of Algorithm 2, the CH is not aware of the number of compromised nodes q. Without knowing q, the CH compares the computed weight of each node with multiple threshold values δ_q, for different q's. If any node violates any threshold value, it is declared compromised and its revocation is initiated via a hopping sequence update.

5.3 Compromise of the Clusterhead

By compromising the CH, the adversary can obtain all sequences s_j, $1 \leq j \leq n$, the corresponding sequences m_j, as well as c and v. Using his knowledge of c and v, the adversary can deny control-channel access to all cluster nodes by jamming only the control channel locations. To address such a strong attack, the role of the CH has to be periodically rotated among cluster members. The steps of the hopping sequence assignment in the case of a CH rotation are as follows:

Step 1 : The new CH randomly hops between channels.

Step 2 : In each slot, the CH attempts to communicate with a cluster node n_j to establish a pairwise shared PN code. Random hopping continues until the establishment of the PN code is confirmed by both parties (via an ACK message).

Step 3 : The CH assigns a new hopping sequence m_j^* to n_j, using the pairwise shared PN code. The sequence m_j^* conforms to the design outlined in Section 4.1.

Step 4 : Node n_j erases all information regarding the identity of the new CH.

Step 5 : Steps 1-4 are repeated until all legitimate nodes are assigned new hopping sequences, except for the previous CH.

When a CH rotation occurs, the new CH n_i has to update all cluster nodes except the previous CH with new PN codes. Because the new CH is not aware of the current PN sequences followed by each cluster node n_j, it randomly hops to different channels in order to meet each node and assign it a new hopping sequence. If n_i meets a node n_j, it first establishes a pairwise PN code with n_j (via a commonly derived seed) and then updates that node with m_j^*. For simplicity, we have made the assumption that one slot is sufficient for communicating m_j^* to n_j. In reality, several packets may be needed to do that. The communication between the new CH and any cluster node is still susceptible to jamming activity. However, because the PN code used by the two parties is not known to the jammer, the transmission will eventually be successful. The reception of m_j^* is acknowledged by n_j via an acknowledgement message. The new CH repeats this process until all cluster nodes are assigned new PN codes and have acknowledged their reception.

Note that initiation of a CH rotation has to be invoked by the individual cluster nodes, since the current CH is compromised. Nodes can declare the CH to be compromised if they cannot access the control channel for a prolonged period of time (computed in Section 6.4). Any cluster node other than the current CH may decide to become the CH and initiate the CH rotation process. If more than one nodes decide to become CHs, a cluster may be partitioned to smaller clusters.

6 Performance Analysis

In this section, we analytically study the anti-jamming metrics introduced in Section 2, and validate our analysis via simulations. We evaluate both static and dynamic spectrum networks.

Simulation Setup: For static spectrum networks, we construct the hopping sequences m_j according to the process described in Section 4.1. Under an external jammer model, the adversary jams channels in a random fashion. Under an internal jammer model, the adversary jams only those slots in which the compromised sequences overlap, and remains silent in all other slots. The simulations are run for 5,000 slots.

For dynamic spectrum networks, we simulate PR activity to obtain temporally varying spectrum availability. We consider a cellular network as the primary network (PRN), operating over $K = 10$ frequency bands. The call arrival process at the PRN follows a Poisson distribution with an arrival rate of $\lambda = 2$ calls/min. The call duration is assumed to be exponentially distributed with parameter μ. For each value of μ, we run the simulation until 5,000 calls are completed. The set of idle channels is updated each time a new call arrives, or when a call is terminated. The jammer is assumed to be aware of the set of idle channels at every slot. The slot duration is set to 100 msec. Each secondary node (e.g., CR node) dynamically adjusts its hopping sequence according to the process described in Section 4.2.

6.1 External Jammer

We first consider the case of an external jammer. In this scenario, the hopping sequences followed by each cluster node remain secret. Before we compute the metrics of interest, we derive the optimal jamming strategy for an external jammer. Without any inside information, the jammer must guess the location of the control channel. The optimal jamming strategy is obtained from the following proposition.

Proposition 5: The optimal strategy of an external jammer is to continuously jam the most frequently visited channel.

Proof: The proof is provided in Appendix 3. □

When the channel location vector c is random, the jammer is expected to have the same likelihood of success, regardless of how he constructs m_{jam}. This can be easily seen fromm eq. (6) of Appendix 3, when $p_1 = \ldots = p_K = p$. Note that for the subsequence c_{jam} which is of interest, it holds that $\Pr[c(i) = c_{jam}(i)] = p$, irrespective of the values of q_i's. If c is not random (this is true for dynamic spectrum networks where the modulo operation reduces the randomness in c), the optimal jamming strategy is to continuously jam the most probable channel. Based on equation (4) of Appendix 2, in the case of dynamic networks channels $\{1, \ldots, y_K\}$ ($y_K = [K \pmod{K(i)}] > 0$) occur in the hopping sequences m'_j with the highest probability. Therefore, the optimal jamming strategy is to continuously jam any of the channels in $\{ch_i(1), \ldots, ch_i(y_K)\}$, yielding a success probability of $\frac{\lfloor \frac{K}{K(i)} \rfloor + 1}{K}$ per slot. In fact, choosing any probability distribution which distributes the probability mass on the set $\{ch_i(1), \ldots, ch_i(y_K)\}$ yields the same probability of success. Given the optimal jamming strategy, we now evaluate the proposed anti-jamming metrics for an external jammer.

6.1.1 Evasion Entropy

The elements $m_j(i)$ of a sequence m_j are generated independently for each slot. Hence, knowledge of previous control channel locations does not reveal any information about future ones. In this case, $E_i = H(I_i)$. For static spectrum networks, $m_j(i)$ is drawn from a uniform distribution, yielding the maximum value for the evasion entropy, i.e., $E_i = \log_2 K$ bits. For dynamic spectrum networks, E_i depends on the number of idle channels $K(i)$. Using the probability distribution of eq. (4) in Appendix 2, it can be shown that

$$E_i = \frac{1}{K}\left(\log_2\left(\frac{K^{y_K - K(i)x_K}}{(x_K+1)^{(x_K+1)y_K} x_K^{(K(i)-y_K)x_K}}\right)\right) \quad (10)$$

where $x_K \triangleq \lfloor \frac{K}{K(i)} \rfloor$ and $y_K \triangleq [K \pmod{K(i)}] > 0$.

6.1.2 Evasion Delay

Proposition 6: In static spectrum networks, the expected evasion delay $\mathrm{E}[D]$ for re-establishing the control channel when no node has been compromised is

$$\mathrm{E}[D] = \frac{K}{K-1} \cdot \frac{L+M}{M}. \quad (11)$$

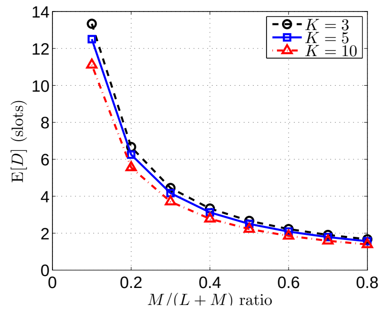
Proof: The proof is provided in Appendix 4. □

For the case of dynamic spectrum networks, the probability of evading jamming in slot i is equal to $(1 - \Pr[\mathcal{M}])$, where $\Pr[\mathcal{M}]$ is given in eq. (5d) of Appendix 2. Therefore, $\mathrm{E}[\mathcal{R}] = \frac{1}{1-\Pr[\mathcal{M}]}$, whereas $\mathrm{E}[\mathcal{N}]$ remains the same as in static networks. Substituting $\mathrm{E}[\mathcal{R}]$ and $\mathrm{E}[\mathcal{N}]$ yields

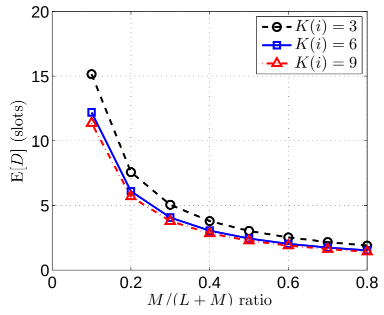
$$\mathrm{E}[D] = \frac{1}{1 - \Pr[\mathcal{M}]} \cdot \frac{L+M}{M}. \quad (12)$$

6.1.3 Evasion Ratio

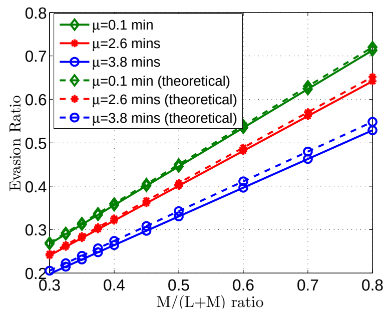
The evasion ratio reflects the communication efficiency of the control channel. It measures the fraction of slots used for control



(a)

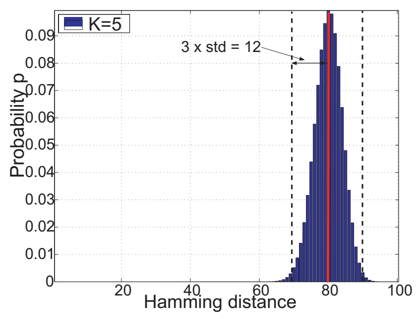


(b)

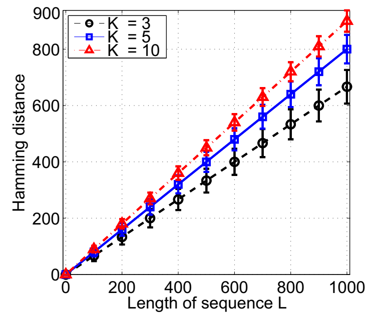


(c)

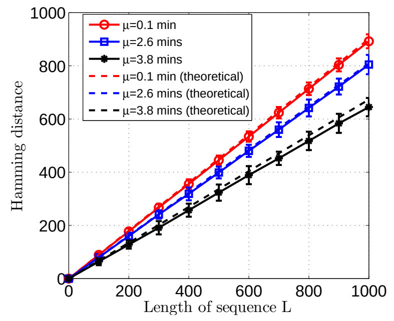
Fig. 5: (a) $\mathrm{E}[D]$ as a function of the ratio $\frac{M}{L+M}$ for static spectrum networks, (b) $\mathrm{E}[D]$ as a function of the ratio $\frac{M}{L+M}$ for dynamic spectrum networks, (c) $\mathrm{E}[ER]$ as a function of $\frac{M}{L+M}$ for dynamic spectrum networks.



(a)



(b)



(c)

Fig. 6: (a) pmf of the Hamming distance between two random sequences of length 100, (b) expected Hamming distance as a function of a sequence of length L for static spectrum networks (error margins denote 99.7% confidence intervals), (c) expected Hamming distance as a function of L for dynamic spectrum networks.

communication in the presence of a jammer. The expected value of the evasion ratio $\mathrm{E}[ER]$ can be directly derived by taking the inverse of the evasion delay.

6.1.4 Simulation and Numerical Examples

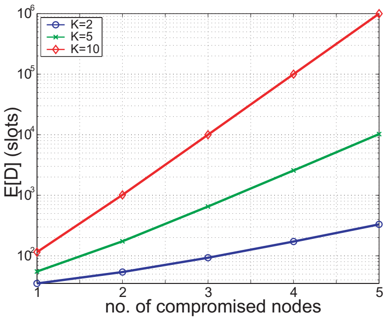
In Fig. 5(a), we show the expected evasion delay as a function of the ratio $\frac{M}{M+L}$ for static spectrum networks. The ratio $\frac{M}{M+L}$ denotes the fraction of time devoted to the control channel. It can be observed that the evasion delay drops with the increase in $\frac{M}{M+L}$. This is due to the fact that the control channel occurs more frequently and hence, the jammer will be unsuccessful in guessing the location of the control channel in fewer slots. However, in the event of a node compromise, fewer slots are available to identify compromised sequences when $\frac{M}{M+L}$ increases. In Fig. 5(b), we show the expected evasion delay as a function of $\frac{M}{M+L}$ for various values of $K(i)$ and for dynamic spectrum networks. This graph corresponds to equation (12). For a fixed value of $K(i)$, a behavior similar to the case of static spectrum networks is observed.

The evasion ratio can be obtained by inverting the values of the evasion delay. $\mathrm{E}[ER]$ increases linearly with $\frac{M}{M+L}$. To take into account temporal variations in spectrum availability in the case of CRNs, we compute the evasion ratio under simulated PRN activity. In Fig. 5(c), we show the evasion ratio as a function of $\frac{M}{L+M}$ for dynamic spectrum networks. Solid lines correspond to the simulation values, while dashed lines correspond to the theoretical ones. To obtain the theoretical values, we used Equation (12) to calculate the evasion delay and then computed its inverse value. For the calculation of $\Pr[\mathcal{M}]$, the mean value of the number of idle channels $E[K]$, obtained via simulation, was used. For the simulation results, we assumed the adversary is aware of the set of idle channels $\mathcal{F}_i$ in each slot i. Based on the optimal jamming strategy, the adversary jams the most probable channel in each slot. If the adversary succeeds in jamming the control channel in slot i, we measure the delay until the control channel is re-established. The evasion ratio is computed as the inverse value of the average evasion delay. From Fig. 5(c), we observe that the simulated values closely match the theoretical ones. As expected from the theoretical analysis, the evasion ratio is a linear function of the fraction of time that the control channel is available.

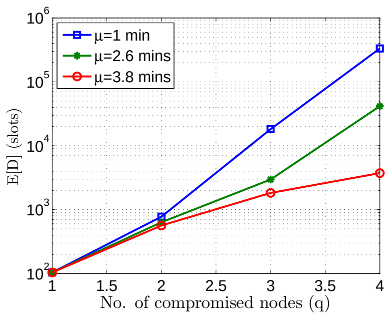
6.2 Compromise of a Single Node

When a single node n_j is compromised, its hopping sequence m_j is revealed to the adversary. By following m_j, the adversary can jam all slots implementing the control channel. In this case, $E_i = 0$ and $\mathrm{E}[ER] = 0$, for as long as the compromised node is undetected. The evasion delay is equal to the time required to identify the compromised node. Under a single compromised node scenario, we evaluate the properties of the Hamming distance between randomly hopping sequences and correlated ones, that lead to the identification of the compromised node.

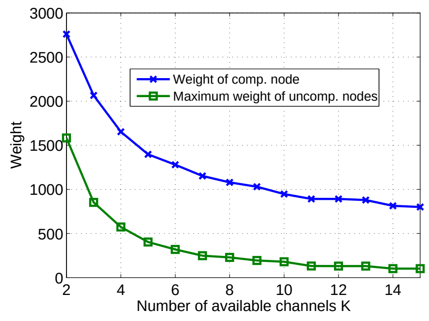
In Fig. 6(a), we show the pmf of the Hamming distance between two random sequences when an alphabet $\mathcal{A} = \{1, \dots, 5\}$ is used for random sequence generation. The pmf is concentrated in a small region around the mean. For a sequence of length



(a)



(b)



(c)

Fig. 7: (a) $\mathrm{E}[D]$ as a function of the number of compromised nodes for static spectrum networks, (b) $\mathrm{E}[D]$ as a function of the number of compromised nodes for dynamic spectrum networks, (c) weight of the compromised node compared to the maximum weight of uncompromised ones.

100, 99.7% of possible random sequences are expected to have a Hamming distance of at least 68. If a jammer overlaps with m_j in more than 32 slots per 100, the CH will declare m_j to be compromised.

In Fig. 6(b), we show the expected Hamming distance as a function of the sequence length L for static spectrum networks. Error margins indicate the 99.7% confidence interval (three standard deviations). We observe that the event of node compromise can be easily identified by comparing the hopping sequence of the jammer to those assigned to cluster nodes. The Hamming distance must fall within well-confined margins, allowing fast identification of the compromised node.

In Fig. 6(c), we show the expected Hamming distance as a function of L for dynamic spectrum networks. Both theoretical and simulation values are shown. Note that the temporal variations in channel availability do not significantly affect the expected Hamming distance, which increases linearly with the length of the hopping sequences. The allowable deviation from the expected value remains small, leading to fast identification if a jammer follows a compromised sequence.

6.3 Compromise of Multiple Nodes

When multiple nodes are compromised, the jammer can combine their hopping sequences to obtain the channel locations of the slots in which these sequences overlap. This reduces the adversary's effort to jam the control channel (fewer slots need to be jammed) and makes the identification process more difficult. As in the case of a single compromised node, $E_i = 0$ and $\mathrm{E}[ER] = 0$. To evaluate the evasion delay, we compute the time required to identify the set of compromised nodes, assign new sequences to uncompromised ones, and re-establish the control channel.

Suppose that q nodes are compromised in a cluster of n nodes. According to Algorithm 2, each of the q nodes is expected to have a weight of $\left(\frac{1}{K}\right)^q qY$, where Y is the time in slots that the CH uses to monitor each of the q compromised nodes. Let γ_0 be the number of *jammed* slots that are required for identification of the compromised nodes. To observe γ_0 jammed slots, the CH needs to monitor channels according to m_{CH} for an average time of $qX = K^q\gamma_0$ slots. Upon identification of the compromised nodes, the CH must assign new hopping sequences to the remaining $(n-q)$ uncompromised nodes, yielding an additional delay of $(n-q)X_c$ slots, where X_c is the number of slots needed to assign a new sequence. Once sequences are assigned, a delay equal to the first occurrence of the control channel under a random jammer is incurred. Thus, the total expected evasion delay is:

$$E[D] = K^q\gamma_0 + (n-q)X_c + \frac{K}{K-1}\cdot\frac{L+M}{M}. \qquad (13)$$

In Fig. 7(a), we show $\mathrm{E}[D]$ as a function of q for static spectrum networks. For simplicity, we take $X_c = 1$ and $\gamma_0 = 100$ slots. We observe that for large values of K, $\mathrm{E}[D]$ is very large when $q \geq 3$. This is due to the fact that the probability of overlapping among the q sequences at random becomes very small for large K. Thus, a much longer observation period is required to identify compromised nodes. For faster identification, the CH may limit the assigned hopping sequences to a subset of $\mathcal{M}$.

Fig. 7(b) shows $\mathrm{E}[D]$ as a function of q for dynamic spectrum networks. We observe a low value of $\mathrm{E}[D]$ when the PR activity is high. This behavior can be explained as follows. High values of μ translate into a smaller number of idle channels $K(i)$. Therefore, CRs hop between a smaller set of channels. The probability of compromised sequence overlap in a slot i, $i \notin v$, a necessary condition for their identification, increases with the reduction in $K(i)$. This is also evident from (13). Hence, the CH is able to identify compromised nodes faster using Algorithm 2.

To verify the effectiveness of Algorithm 2, we perform the following simulation experiment. We consider a cluster of 10 nodes and generate 10 hopping sequences, each of length 5,000. q of those sequences are assumed to be compromised. The jammer computes the jamming sequence m_{jam} as the intersection of the compromised sequences, and jams only the slots in which the q sequences overlap. Algorithm 2 is executed to compute the weight of each hopping sequence. The CH monitors the K channels according to sequence m_{CH}, following each sequence in a round-robin manner for 100 slots. In Fig. 7(c), we show the average weight $\mathrm{E}[W]$ of a compromised node compared with the maximum weight obtained from the set of uncompromised sequences, as a function of K. We observe that compromised nodes have a consistently higher weight than uncompromised nodes, leading to identification of the former ones. Fig. 8 shows a comparison between the weight of compromised nodes and the maximum weight of uncompromised nodes, as a function of q.

When the number of compromised nodes is small (less than 4),

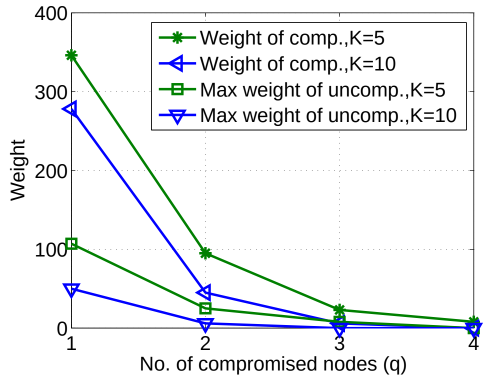


Fig. 8: Average weight of compromised nodes and maximum weight of uncompromised ones versus q.

the weight of a compromised node is sufficiently distinct from the weight of an uncompromised node. However, for higher values of q, compromised sequences have less probability to coincide in each slot except the control channel slots. In this scenario, the CH must monitor each node for a large number of slots, in order to measure disparities in the weights of different sequences.

6.4 Compromise of the Clusterhead

If the CH is compromised, the adversary knows the hopping schedules of all nodes in the cluster as well as the slots of the control channel. Hence, the evasion entropy and the evasion ratio are equal to zero. The evasion delay $\mathrm{E}[D]$ is equal to the sum of three components: (a) the delay $\mathrm{E}[D_1]$, until the compromise of the CH is detected, (b) the delay $\mathrm{E}[D_2]$, of assigning new hopping sequences to cluster members, and (c) the delay $\mathrm{E}[D_3]$. for re-establishing the control channel.

Cluster nodes consider the CH compromised when $E[ER]$ falls below a threshold value ρ_0 for an extended period of time. The parameter ρ_0 is fixed and depends on the expected delay under a fixed number of compromised nodes. Let q_0 be the maximum tolerable number of compromised nodes within a cluster, before the CH is assumed to be compromised. The evasion delay when q_0 nodes are compromised ($E[D_1]$ in the calculation of $E[D]$) is given by equation (13), with $q = q_0$. The computation of $E[D_1]$ is based on fixed system parameters such as γ_0, K, L, M, and X_c. Taking the inverse of $E[D_1]$ when $q = q_0$, yields the threshold value ρ_0 that triggers a CH rotation. To detect the compromise of the CH, individual nodes compare $E[ER]$ with ρ_0.

Proposition 7: The expected delay until the new CH assigns new hopping sequences to $n-1$ cluster nodes (excluding the compromised CH) is

$$\mathrm{E}[D_2] = \frac{K^2}{K-1}(n-1)X_c. \tag{14}$$

Proof: The proof is provided in Appendix 5. □

With the assignment of new sequences, the adversary's success becomes equivalent to that of an external jammer. An additional delay $\mathrm{E}[D_3]$ is incurred until a slot implementing the control channel occurs in the new hopping sequences. This delay is equal to the evasion delay in the case of the external jammer. The values of $\mathrm{E}[D_2]$ and $\mathrm{E}[D_3]$ are negligible compared with $\mathrm{E}[D_1]$, given that $\mathrm{E}[D_1]$ grows exponentially with the number of compromised nodes, whereas $\mathrm{E}[D_2]$ and $\mathrm{E}[D_3]$ are constant. Hence, the expected value for the evasion delay under CH compromise approximates the expected evasion delay as calculated in (13) for the maximum acceptable value of q.

7 Related Work

Jamming in wireless networks has been extensively studied. Most prior research assumes that the jammer is an external entity, oblivious to the protocol specifics and cryptographic secrets [19]. Recently, several works have considered the problem of jamming by an internal adversary, who exploits knowledge of network protocols and secrets to launch DoS attacks on layers above the physical layer [6], [14], [17], [18], [24]–[26]. In this section, we classify related work based on the adversarial model.

Jamming Under an Internal Threat Model– Chan et al. considered the problem of control-channel jamming in the context of GMS networks [6]. They proposed the replication of control information over multiple channels according to a binary encoding based key (BBK) assignment. Assuming an adversary who is capable of jamming only one channel per time slot, the authors derived necessary conditions to guarantee control channel access to all users within several slots. They also showed that the BBK assignment leads to the identification of a certain number of compromised nodes.

Tague et al. proposed a cryptographic key-based mechanism for hiding the control-channel slots [26]. Nodes can only discover a subset of these locations with some probability. Their method allows for graceful degradation in the control-channel secrecy as a function of the number of compromised nodes, as opposed to the threshold approach in [6]. Further, they proposed an algorithm called GUIDE for identifying compromised nodes based on the set of jammed control channels. They formulated the identification problem as a maximum likelihood estimation problem [26]. All methods in [6], [25], [26] consider a server-client model, where base stations are assumed to be secure.

Chiang et al. proposed an anti-jamming scheme for broadcast communications in DS- and FH-CDMA systems [7]. Their method organizes broadcast PN codes into a binary key tree. Each node on the tree corresponds to a unique PN code, known only to a subset of users. Every message is spread by multiple PN codes such that all users can decode using exactly one code. Identification of compromised nodes is achieved by relating the PN code adopted by the jammer to those known to each user.

Several schemes eliminate the need for secret PN codes [3], [14], [17], [24]. Baird et al. proposed the BBC algorithm, which can recover jammed messages under some special conditions. can insert arbitrary messages into the broadcast channel but cannot erase any of the original messages. Pöpper et al. proposed a solution called Uncoordinated DSSS (UDSSS) [17]. In their scheme, broadcast transmissions are spread according to a PN code that is randomly selected from a public set of codes. At the receiving end, nodes have to record transmitted messages and attempt to decode them by exhaustively applying every PN code in the public codebook. Because the selected PN code is not known a priori to any receiver, the jammer has to guess the PN code, thus significantly complicating the jamming task. However, message transmissions have to be repeated several times to allow receivers to synchronize with the transmitter. Strasser et al. proposed an uncoordinated frequency hopping (UFH) scheme for establishing shared secret keys between devices that do not share any prior secrets, in the presence of a jammer [24]. In UHF, the transmitter and receiver hop between channels at random. After some number of hops, they are able to exchange a common pairwise key and independently derive a pairwise shared PN code. An improvement in communication latency and jamming resistance of the original UHF scheme was presented in [23], by combining coding techniques with hashing. Slater et al. improved the communication efficiency of UFH by using Merkle trees, distillation codes, and erasure coding [21].

Liu et al. proposed RD-DSSS, a randomized differential DSSS scheme that enables jamming-resistant broadcast using only publicly known PN codes [14]. In RD-DSSS, a “0” bit is encoded using two randomly selected PN codes with low correlation, while a “1” bit is encoded using two PN codes with high correlation. The selected PN codes are appended at the end of each message, thus slightly decreasing the communication efficiency compared with the original DSSS. Recovery of the PN codes that were selected by the sender is achieved only after the transmitted message is received.

Jamming Under an External Threat Model– Under an external threat model, jamming is often mitigated by employing SS techniques [19], [20]. In these techniques, the transmitted narrowband signal is spread over a larger bandwidth according to a secret PN code. Anti-jamming properties are achieved because more energy is required to cause interference in a larger bandwidth. The typical processing gain in SS communications is in the range of 20 to 30 dB [19], [20].

Xu et al. studied the problem of jamming in systems where spreading is not possible (or effective) [28]–[30]. They studied the problem of detecting physical-layer and MAC-layer DoS attacks based on jamming [30]. They proposed a slow frequency hopping method to avoid jamming, but assumed that hopping sequences remain secret. For mobile networks, they proposed the use of spatial retreats to avoid communication within the jammed area. Formal measures for detecting jamming attacks were introduced in [29]. Xu et al. also proposed the establishment of a timing-based low bitrate covert channel to notify nodes outside the jamming area about the presence of a jammer [28]. This channel maps the inter-arrival times of corrupted packets into bits. Cagalj et al. proposed wormhole-based anti-jamming techniques for sensor networks [5]. Using a wormhole link, sensors within a jammed region establish communications outside this region, and notify them regarding ongoing jamming attacks.

Jamming Beyond the PHY Layer– The use of jamming as a vehicle for launching DoS attacks against higher-layer functionalities was studied in [4], [5], [8], [11]–[13], [15], [18]. Brown et al. demonstrated that a jammer can exploit implicit packet identifiers such as packet size, timing, and sequence number at the transport or network layer to classify transmitted packets and launch selective jamming attacks [4]. Proaño and Lazos showed the feasibility of selective jamming by performing real-time packet classification. Liu et al. proposed a layered architecture called SPREAD to mitigate the impact of smart jammers that target multiple layers of the network stack [13]. SPREAD randomizes protocols at each layer, thus increasing the adversary’s uncertainty with respect to the protocol execution. Finally, Li. et al. provided a game theoretic approach to optimal jamming and anti-jamming strategies at the MAC layer [11].

8 Conclusions

We addressed the problem of control-channel jamming attacks from insider nodes. We proposed a randomized distributed scheme for maintaining and establishing a broadcast channel using frequency hopping. Our method differs from classical frequency hopping in that the communicating nodes are not synchronized to the same hopping sequence. Instead, each node follows a unique hopping sequence. We further proposed a mechanism for adjusting hopping sequences to dynamic spectrum conditions without incurring any extra overhead. Our scheme can identify compromised nodes through their unique sequences and exclude them from the network. We evaluated the performance of our scheme both in static- and dynamic-spectrum networks, based on the metrics of evasion entropy, evasion delay, and evasion ratio. We further evaluated the Hamming distance between the jamming sequence and those assigned to compromised and uncompromised nodes. Our proposed scheme can be utilized as a temporary solution for re-establishing the control channel until the jammer and the compromised nodes are removed from the network.

Acknowledgments

Part of this work was conducted while M. Krunz was a visiting researcher at the University of Carlos III, Madrid, and Institute IMDEA Networks, Spain. This research was supported in part by NSF (under grants CNS-0844111, CNS-0721935, CNS-0904681, CNS-1016943, IIP-0832238), Raytheon, and the Connection One center. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] D. Adamy. *EW 101: A first course in electronic warfare*. Artech House Publishers, 2001.
[2] I. Akyildiz, W. Lee, M. Vuran, and S. Mohanty. Next generation dynamic spectrum access cognitive radio wireless networks: A survey. *Computer Networks*, 50(13):2127–2159, 2006.
[3] L. C. Baird, W. L. Bahn, M. D. Collins, M. C. Carlisle, and S. C. Butler. Keyless jam resistance. In *Proceedings of the 2007 IEEE Workshop on Information Assurance United States Military Academy*, 2007.
[4] T. X. Brown, J. E. James, and A. Sethi. Jamming and sensing of encrypted wireless ad hoc networks. In *Proceedings of the ACM MobiHoc*, pages 120–130, 2006.
[5] M. Cagalj, S. Capkun, and J.-P. Hubaux. Wormhole-based anti-jamming techniques in sensor networks. *IEEE Transactions on Mobile Computing*, 6(1):100–114, 2007.
[6] A. Chan, X. Liu, G. Noubir, and B. Thapa. Control channel jamming: resilience and identification of traitors. In *Proceedings of ISIT*, 2007.

[7] J. T. Chiang and Y.-C. Hu. Cross-layer jamming detection and mitigation in wireless broadcast networks. In *Proceedings of the MobiCom*, pages 346–349, 2007.

[8] Y. W. Law, L. V. Hoesel, J. Doumen, P. Hartel, and P. Havinga. Energy efficient link-layer jamming attacks against wireless sensor network MAC protocols. In *Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, 2005.

[9] L. Lazos, S. Liu, and M. Krunz. Mitigating control-channel jamming attacks in multi-channel ad hoc networks. In *Proceedings of the 2nd ACM Conference on Wireless Network Security (WiSec)*, pages 169–180, 2009.

[10] L. Lazos, S. Liu, and M. Krunz. Spectrum opportunity-based control channel assignment in cognitive radio networks. In *Proceedings of SECON*, pages 135–143, 2009.

[11] M. Li, I. Koutsopoulos, and R. Poovendran. Optimal jamming attacks and network defense policies in wireless sensor networks. In *Proceedings of the INFOCOM*, 2007.

[12] G. Lin and G. Noubir. On link-layer denial of service in data wireless LANs. *Journal on Wireless Communications and Mobile Computing*, 2004.

[13] X. Liu, G. Noubir, R. Sundaram, and S. Tan. SPREAD: Foiling smart jammers using multi-layer agility. In *Proceedings of the INFOCOM Mini Symposium*, 2007.

[14] Y. Liu, P. Ning, H. Dai, and A. Liu. Randomized differential DSSS: Jamming-resistant wireless broadcast communication. In *Proceedings of the INFOCOM*, 2010.

[15] J. M. McCune, E. Shi, A. Perrig, and M. K. Reiter. Detection of denial of message attacks on sensor network broadcasts. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2005.

[16] G. Noubir and G. Lin. Low power DoS attacks in data wireless LANs and countermeasures. In *Proceedings of the ACM MobiCom*, 2003.

[17] C. Popper, M. Strasser, and S. Čapkun. Anti-jamming broadcast communication using uncoordinated spread spectrum techniques. *IEEE Journal on Selected Areas in Communication*, 28(5):703–715, 2010.

[18] A. Proaño and L. Lazos. Selective jamming attacks in wireless networks. In *Proceedings of ICC*, 2010.

[19] M. K. Simon, J. K. Omura, R. A. Scholtz, and B. K. Levitt. *Spread Spectrum Communications Handbook*. McGraw-Hill, 2001.

[20] B. Sklar. *Digital Communications, Fundamentals and Applications*. Prentice-Hall, 2001.

[21] D. Slater, P. Tague, R. Poovendran, and B. Matt. A coding-theoretic approach for efficient message verification over unsecure channels. In *Proceedings of the ACM Conference on Wireless Security (WiSec)*, 2009.

[22] J. So and N. H. Vaidya. Multi-channel MAC for ad hoc networks: Handling multi-channel hidden terminals using a single transceiver. In *Proceedings of the ACM MobiHoc*, pages 222–233, 2004.

[23] M. Strasser, C. Popper, and S. Capkun. Efficient uncoordinated FHSS anti-jamming communication. In *Proceedings of the ACM MobiHoc*, 2009.

[24] M. Strasser, C. Popper, S. Capkun, and M. Cagalj. Jamming-resistant key establishment using uncoordinated frequency hopping. In *Proceedings of IEEE Symposium on Security and Privacy*, 2008.

[25] P. Tague, M. Li, and R. Poovendran. Probabilistic mitigation of control channel jamming via random key distribution. In *Proceedings of PIRMC*, 2007.

[26] P. Tague, M. Li, and R. Poovendran. Mitigation of control channel jamming under node capture attacks. *IEEE Transactions on Mobile Computing*, 8(9):1221–1234, 2009.

[27] Y. Tseng, S. Ni, Y. Chen, and J. Sheu. The broadcast storm problem in a mobile ad hoc network. *Wireless Networks*, 8(2/3):153–167, 2002.

[28] W. Xu, W. Trappe, and Y. Zhang. Anti-jamming timing channels for wireless networks. In *Proceedings of the 1st ACM Conference on Wireless Security (WiSec)*, 2008.

[29] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proceedings of the ACM MobiHoc*, pages 46–57, 2005.

[30] W. Xu, T. Wood, W. Trappe, and Y. Zhang. Channel surfing and spatial retreats: Defenses against wireless denial of service. In *Proceedings of Wireless Security Workshop (WiSe)*, 2004.

[31] J. Yu and P. Chong. A survey of clustering schemes for mobile ad hoc networks. *IEEE Communications Surveys & Tutorials*, 7(1):32–48, 2005.

Sisi Liu received the B.S. and M.S. degree in electrical engineering from University of Electronic Science and Technology of China,Chengdu, China, in 2004 and 2007, respectively. She is currently a research assistant working toward a Ph.D degree at the Advanced networking lab, Department of Electrical and Computer Engineering, University of Arizona, Tucson. Her research interests lie in the areas of cognitive radio-based, ad hoc, and wireless sensor networks with emphasis on network security, physical layer attacks, and misbehavior detection.

Loukas Lazos is an Assistant Professor of Electrical and Computer Engineering at the University of Arizona. He received his Ph.D. in Electrical Engineering from the University of Washington in 2006. In 2007, he was the co-director of the Network Security Lab at the University of Washington. Dr. Lazos joined the the University of Arizona in August 2007. His main research interests are in the areas of networking, security, and wireless communications, focusing on the identification, modeling, and mitigation of security vulnerabilities, visualization of network threats, and analysis of network performance. He is a recipient of the NSF CAREER Award (2009), for his research in security of multi-channel wireless networks. He has served and continues to serve on technical program committees of many international conferences and on the panels of several NSF directorates.

Marwan Krunz is a professor of ECE at the University of Arizona. He also holds a joint appointment at the same rank in the CS department. He is the UA site director for Connection One, a joint NSF/state/industry IUCRC cooperative center that focuses on wireless communication systems and networks. Dr. Krunz received his Ph.D. degree in electrical engineering from Michigan State University in 1995. He joined the University of Arizona in January 1997, after a brief postdoctoral stint at the University of Maryland, College Park. He previously held visiting research positions at INRIA, HP Labs, University of Paris VI, and US West (now Qwest) Advanced Technologies. In 2010, he was a visiting researcher at Institute IMDEA and a Chair of Excellence ("Cátedra de Excelencia") at the University of Carlos III, Madrid, Spain. Dr. Krunzs research interests lie in the fields of computer networking and wireless communications, with focus on distributed radio resource management in wireless and sensor networks; protocol design; and secure communications. He has published more than 170 journal articles and refereed conference papers, and is a co-inventor on three US patents. He is a recipient of the NSF CAREER Award (1998). He currently serves on the editorial board for the IEEE Transactions on Network and Service Management. Previously, he served on the editorial boards for the IEEE/ACM Transactions on Networking, the IEEE Transactions on Mobile Computing, and the Computer Communications Journal. He served as a TPC chair for various international conferences, including INFOCOM04, SECON05, and WoWMoM06. He is an IEEE Fellow.