

# Packet-Hiding Methods for Preventing Selective Jamming Attacks

Alejandro Proaño and Loukas Lazos

Dept. of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, USA

E-mail: {aproano, llazos}@ece.arizona.edu

**Abstract**—The open nature of the wireless medium leaves it vulnerable to intentional interference attacks, typically referred to as jamming. This intentional interference with wireless transmissions can be used as a launchpad for mounting Denial-of-Service attacks on wireless networks. Typically, jamming has been addressed under an external threat model. However, adversaries with internal knowledge of protocol specifications and network secrets can launch low-effort jamming attacks that are difficult to detect and counter. In this work, we address the problem of selective jamming attacks in wireless networks. In these attacks, the adversary is active only for a short period of time, selectively targeting messages of high importance. We illustrate the advantages of selective jamming in terms of network performance degradation and adversary effort by presenting two case studies; a selective attack on TCP and one on routing. We show that selective jamming attacks can be launched by performing real-time packet classification at the physical layer. To mitigate these attacks, we develop three schemes that prevent real-time packet classification by combining cryptographic primitives with physical-layer attributes. We analyze the security of our methods and evaluate their computational and communication overhead.

**Index Terms**—Selective Jamming, Denial-of-Service, Wireless Networks, Packet Classification.

## 1 INTRODUCTION

Wireless networks rely on the uninterrupted availability of the wireless medium to interconnect participating nodes. However, the open nature of this medium leaves it vulnerable to multiple security threats. Anyone with a transceiver can eavesdrop on wireless transmissions, inject spurious messages, or jam legitimate ones. While eavesdropping and message injection can be prevented using cryptographic methods, jamming attacks are much harder to counter. They have been shown to actualize severe Denial-of-Service (DoS) attacks against wireless networks [12], [17], [36], [37]. In the simplest form of jamming, the adversary interferes with the reception of messages by transmitting a continuous jamming signal [25], or several short jamming pulses [17].

Typically, jamming attacks have been considered under an external threat model, in which the jammer is not part of the network. Under this model, jamming strategies include the continuous or random transmission of high-power interference signals [25], [36]. However, adopting an “always-on” strategy has several disadvantages. First, the adversary has to expend a significant amount of energy to jam frequency bands of interest. Second, the continuous presence of unusually high interference levels makes this type of attacks easy to detect [17], [36], [37].

Conventional anti-jamming techniques rely extensively on spread-spectrum (SS) communications [25], or some form of jamming evasion (e.g., slow frequency hopping, or spatial retreats [37]). SS techniques provide bit-level protection by spreading bits according to a secret pseudo-noise (PN) code, known only to the communicating parties. These methods can only protect wireless transmissions under the external threat model. Potential disclosure of secrets due

to node compromise, neutralizes the gains of SS. Broadcast communications are particularly vulnerable under an internal threat model because all intended receivers must be aware of the secrets used to protect transmissions. Hence, the compromise of a single receiver is sufficient to reveal relevant cryptographic information.

In this paper, we address the problem of jamming under an internal threat model. We consider a sophisticated adversary who is aware of network secrets and the implementation details of network protocols at any layer in the network stack. The adversary exploits his internal knowledge for launching *selective jamming attacks* in which specific messages of “high importance” are targeted. For example, a jammer can target route-request/route-reply messages at the routing layer to prevent route discovery, or target TCP acknowledgments in a TCP session to severely degrade the throughput of an end-to-end flow.

To launch selective jamming attacks, the adversary must be capable of implementing a “classify-then-jam” strategy before the completion of a wireless transmission. Such strategy can be actualized either by classifying transmitted packets using protocol semantics [1], [33], or by decoding packets on the fly [34]. In the latter method, the jammer may decode the first few bits of a packet for recovering useful packet identifiers such as packet type, source and destination address. After classification, the adversary must induce a sufficient number of bit errors so that the packet cannot be recovered at the receiver [34]. Selective jamming requires an intimate knowledge of the physical (PHY) layer, as well as of the specifics of upper layers.

**Our Contributions**—We investigate the feasibility of real-time packet classification for launching selective jamming attacks, under an internal threat model. We show that such attacks are relatively easy to actualize by exploiting knowledge of network protocols and cryptographic primitives extracted from compromised nodes. We investigate the

*A preliminary version of this paper was presented at IEEE ICC 2010 Conference. This research was supported in part by NSF (CNS-0844111, CNS-1016943). Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the author(s) and do not necessarily reflect the views of NSF.*

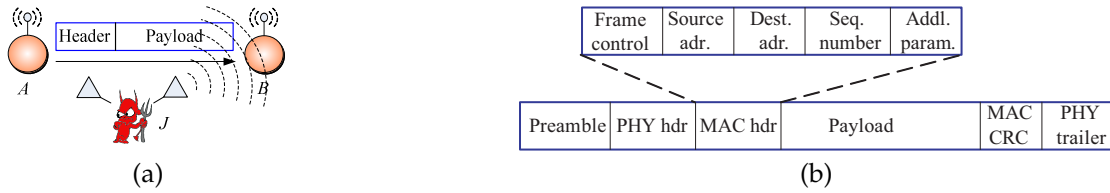


Fig. 1. (a) Realization of a selective jamming attack, (b) a generic frame format for a wireless network.

impact of selective jamming on critical network functions. Our findings indicate that selective jamming attacks lead to a DoS with very low effort on behalf of the jammer. To mitigate such attacks, we develop three schemes that prevent classification of transmitted packets in real time. Our schemes rely on the joint consideration of cryptographic mechanisms with PHY-layer attributes. We analyze the security of our schemes and show that they achieve strong security properties, with minimal impact on the network performance.

The remainder of the paper is organized as follows. In Section 2, we describe the problem addressed, and state the system and adversarial model. In Section 3, we show the feasibility of selective jamming attacks. Section 4 illustrates the impact of selective jamming. In Sections 5, 6, and 7, we develop methods for preventing selective jamming. In Section 8, we evaluate the impact of our attack mitigation methods on the network performance. Section 9, presents related work. In Section 10, we conclude.

## 2 PROBLEM STATEMENT AND ASSUMPTIONS

### 2.1 Problem Statement

Consider the scenario depicted in Fig. 1(a). Nodes  $A$  and  $B$  communicate via a wireless link. Within the communication range of both  $A$  and  $B$  there is a jamming node  $J$ . When  $A$  transmits a packet  $m$  to  $B$ , node  $J$  classifies  $m$  by receiving only the first few bytes of  $m$ .  $J$  then corrupts  $m$  beyond recovery by interfering with its reception at  $B$ . We address the problem of *preventing the jamming node from classifying  $m$  in real time, thus mitigating  $J$ 's ability to perform selective jamming*. Our goal is to transform a selective jammer to a random one. Note that in the present work, we do not address packet classification methods based on protocol semantics, as described in [1], [4], [11], [33].

### 2.2 System and Adversary Model

**Network model**—The network consists of a collection of nodes connected via wireless links. Nodes may communicate directly if they are within communication range, or indirectly via multiple hops. Nodes communicate both in unicast mode and broadcast mode. Communications can be either unencrypted or encrypted. For encrypted broadcast communications, symmetric keys are shared among all intended receivers. These keys are established using pre-shared pairwise keys or asymmetric cryptography.

**Communication Model**—Packets are transmitted at a rate of  $\mathcal{R}$  bauds. Each PHY-layer symbol corresponds to  $q$  bits, where the value of  $q$  is defined by the underlying digital

modulation scheme. Every symbol carries  $\frac{\alpha}{\beta}q$  data bits, where  $\alpha/\beta$  is the rate of the PHY-layer encoder. Here, the transmission bit rate is equal to  $q\mathcal{R}$  bps and the information bit rate is  $\frac{\alpha}{\beta}q\mathcal{R}$  bps. Spread spectrum techniques such as frequency hopping spread spectrum (FHSS), or direct sequence spread spectrum (DSSS) may be used at the PHY layer to protect wireless transmissions from jamming. SS provides immunity to interference to some extent (typically 20 to 30 dB gain), but a powerful jammer is still capable of jamming data packets of his choosing.

Transmitted packets have the generic format depicted in Fig. 1(b). The preamble is used for synchronizing the sampling process at the receiver. The PHY layer header contains information regarding the length of the frame, and the transmission rate. The MAC header determines the MAC protocol version, the source and destination addresses, sequence numbers plus some additional fields. The MAC header is followed by the frame body that typically contains an ARP packet or an IP datagram. Finally, the MAC frame is protected by a cyclic redundancy check (CRC) code. At the PHY layer, a trailer may be appended for synchronizing the sender and receiver.

**Adversary Model**—We assume the adversary is in control of the communication medium and can jam messages at any part of the network of his choosing (similar to the Dolev-Yao model). The adversary can operate in full-duplex mode, thus being able to receive and transmit simultaneously. This can be achieved, for example, with the use of multi-radio transceivers. In addition, the adversary is equipped with directional antennas that enable the reception of a signal from one node and jamming of the same signal at another. For analysis purposes, we assume that the adversary can pro-actively jam a number of bits just below the ECC capability early in the transmission. He can then decide to irrecoverably corrupt a transmitted packet by jamming the *last symbol*. In reality, it has been demonstrated that selective jamming can be achieved with far less resources [32], [34]. A jammer equipped with a single half-duplex transceiver is sufficient to classify and jam transmitted packets. However, our model captures a more potent adversary that can be effective even at high transmission speeds.

The adversary is assumed to be computationally and storage bounded, although he can be far superior to normal nodes. In particular, he can be equipped with special purpose hardware for performing cryptanalysis or any other required computation. Solving well-known hard cryptographic problems is assumed to be time-consuming. For the purposes of analysis, given a ciphertext, the most efficient method for deriving the corresponding plaintext is assumed

to be an exhaustive search on the key space.

The implementation details of every layer of the network stack are assumed to be public. Furthermore, the adversary is capable of physically compromising network devices and recovering stored information including cryptographic keys, PN codes, etc. This internal adversary model is realistic for network architectures such as mobile ad-hoc, mesh, cognitive radio, and wireless sensor networks, where network devices may operate unattended, thus being susceptible to physical compromise.

### 3 REAL-TIME PACKET CLASSIFICATION

In this section, we describe how the adversary can classify packets in real time, before the packet transmission is completed. Once a packet is classified, the adversary may choose to jam it depending on his strategy.

Consider the generic communication system depicted in Fig. 2. At the PHY layer, a packet  $m$  is encoded, interleaved, and modulated before it is transmitted over the wireless channel. At the receiver, the signal is demodulated, de-interleaved, and decoded, to recover the original packet  $m$ .

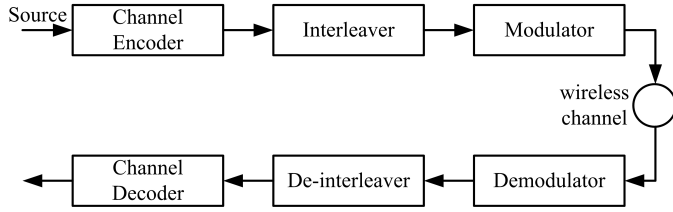


Fig. 2. A generic communication system diagram.

The adversary's ability in classifying a packet  $m$  depends on the implementation of the blocks in Fig. 2. The channel encoding block expands the original bit sequence  $m$ , adding necessary redundancy for protecting  $m$  against channel errors. For example, an  $\alpha/\beta$ -block code may protect  $m$  from up to  $e$  errors per block. Alternatively, an  $\alpha/\beta$ -rate convolutional encoder with a constraint length of  $L_{max}$ , and a free distance of  $e$  bits provides similar protection. For our purposes, we assume that the rate of the encoder is  $\alpha/\beta$ . At the next block, interleaving is applied to protect  $m$  from burst errors. For simplicity, we consider a block interleaver that is defined by a matrix  $A_{d \times \beta}^1$ . The de-interleaver is simply the transpose of  $A$ . Finally, the digital modulator maps the received bit stream to symbols of length  $q$ , and modulates them into suitable waveforms for transmission over the wireless channel. Typical modulation techniques include OFDM, BPSK, 16(64)-QAM, and CCK.

In order to recover any bit of  $m$ , the receiver must collect  $d \cdot \beta$  bits for de-interleaving. The  $d \cdot \beta$  de-interleaved bits are then passed through the decoder. Ignoring any propagation and decoding delays, the delay until decoding the first block of data is  $\lceil \frac{d\beta}{q} \rceil$  symbol durations. As an example, in the 802.11a standard, operating at the lowest rate of 6 Mbps,

1. Without loss of generality we assume that the number of columns of the interleaving matrix is equal to the length  $\beta$  of the codewords.

data is passed via a 1/2-rate encoder before it is mapped to an OFDM symbol of  $q = 48$  bits. In this case, decoding of one symbol provides 24 bits of data. At the highest data rate of 54 Mbps, 216 bits of data are recovered per symbol.

From our analysis, it is evident that intercepting the first few symbols of a packet is sufficient for obtaining relevant header information. For example, consider the transmission of a TCP-SYN packet used for establishing a TCP connection at the transport layer. Assume an 802.11a PHY layer with a transmission rate of 6 Mbps. At the PHY layer, a 40-bit header and a 6-bit tail are appended to the MAC packet carrying the TCP-SYN packet. At the next stage, the 1/2-rate convolutional encoder maps the packet to a sequence of 1,180 bits. In turn, the output of the encoder is split into 25 blocks of 48 bits each and interleaved on a per-symbol basis. Finally, each of the blocks is modulated as an OFDM symbol for transmission. The information contained in each of the 25 OFDM symbols is as follows:

- Symbols 1-2 contain the PHY-layer header and the first byte of the MAC header. The PHY header reveals the length of the packet, the transmission rate, and synchronization information. The first byte of the MAC header reveals the protocol version and the type and subtype of the MAC frame (e.g., DATA, ACK).
- Symbols 3-10 contain the source and destination MAC addresses, and the length of the IP packet header.
- Symbols 11-17 contain the source and destination IP addresses, the size of the TCP datagram carried by the IP packet, and other IP layer information. The first two bytes of the TCP datagram reveal the source port.
- Symbols 18-23 contain the TCP destination port, sequence number, acknowledgment number, TCP flags, window size, and the header checksum.
- Symbols 24-25 contain the MAC CRC code.

Our example illustrates that a packet can be classified at different layers and in various ways. MAC layer classification is achieved by receiving the first 10 symbols. IP layer classification is achieved by receiving symbols 10 and 11, while TCP layer classification is achieved by symbols 12-19.

An intuitive solution to selective jamming would be the encryption of transmitted packets (including headers) with a static key. However, for broadcast communications, this static decryption key must be known to all intended receivers and hence, is susceptible to compromise. An adversary in possession of the decryption key can start decrypting as early as the reception of the first ciphertext block. For example, consider the cipher-block chaining (CBC) mode of encryption [27]. To encrypt a message  $m$  with a key  $k$  and an initialization vector  $IV$ , message  $m$  is split into  $x$  blocks  $m_1, m_2, \dots, m_x$ , and each ciphertext block  $c_i$ , is generated as:

$$c_1 = IV, \quad c_{i+1} = E_k(c_i \oplus m_i), \quad i = 1, 2, \dots, x, \quad (1)$$

where  $E_k(m)$  denotes the encryption of  $m$  with key  $k$ . The plaintext  $m_i$  is recovered by:

$$m_i = c_i \oplus D_k(c_{i+1}), \quad i = 1, 2, \dots, x. \quad (2)$$

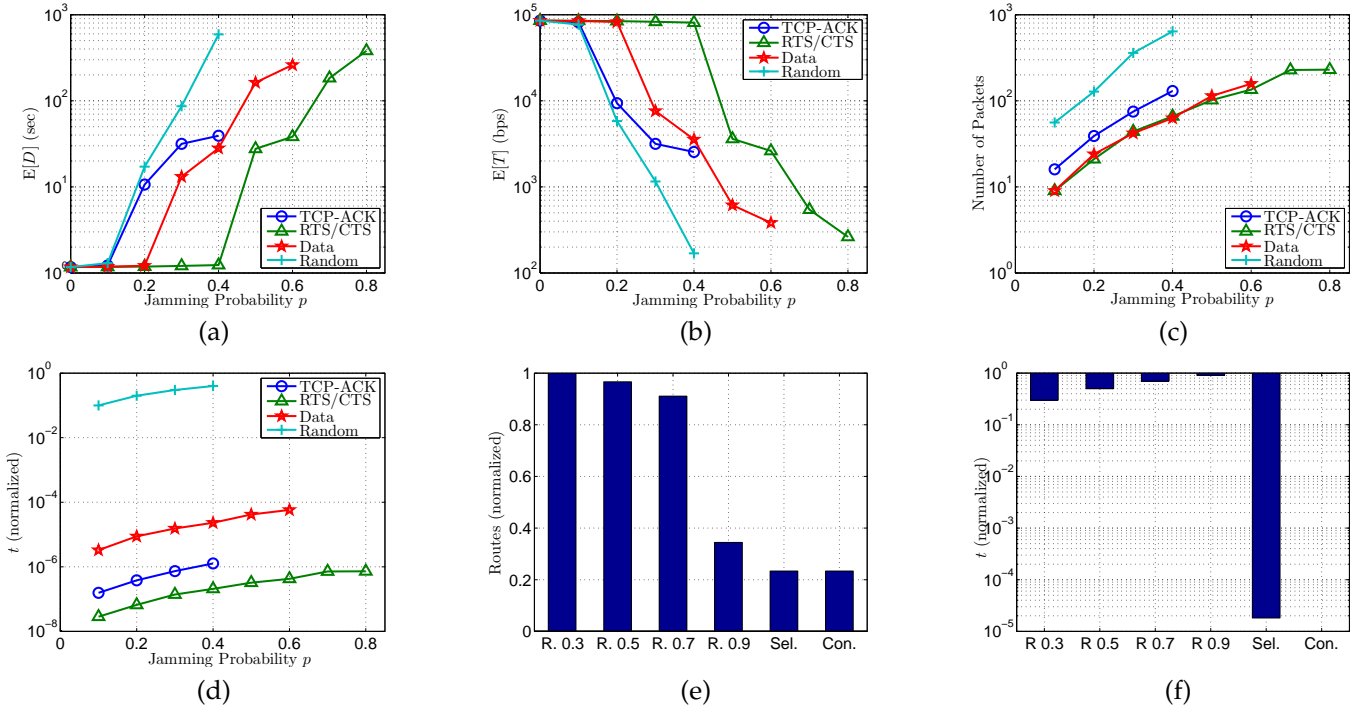


Fig. 3. (a) Average application delay  $E[D]$ , (b) average effective throughput  $E[T]$ , (c) number of packets jammed, (d) fraction of time the jammer is active, (e) number of connections established in the network, (f) fraction of time the jammer is active. R  $p$ : random jammer with probability  $p$ ; Con.: constant jammer; Sel.: selective jammer.

Note from (2) that reception of  $c_{i+1}$  is sufficient to recover  $m_i$  if  $k$  is known ( $c_1 = IV$  is also known). Therefore, real-time packet classification is still possible.

One solution to the key compromise problem would be to update the static key whenever it is compromised. However, such a solution is not useful if the compromised node obtains the new key. This can only be avoided if there is a mechanism by which the set of compromised nodes can be identified. Such a task is non-trivial when the leaked key is shared by multiple nodes. Any node that possesses the shared key is a candidate malicious node.

Moreover, even if the encryption key of a hiding scheme were to remain secret, the static portions of a transmitted packet could potentially lead to packet classification. This is because for computationally-efficient encryption methods such as block encryption, the encryption of a prefix plaintext with the same key yields a static ciphertext prefix. Hence, an adversary who is aware of the underlying protocol specifics (structure of the frame) can use the static ciphertext portions of a transmitted packet to classify it.

#### 4 IMPACT OF SELECTIVE JAMMING

In this section, we illustrate the impact of selective jamming attacks on the network performance. We used OPNET<sup>TM</sup> Modeler 14.5 [18] to implement selective jamming attacks in two multi-hop wireless network scenarios. In the first scenario, the attacker targeted a TCP connection established over a multi-hop wireless route. In the second scenario, the jammer targeted network-layer control messages transmitted during the route establishment process.

**Selective Jamming at the Transport Layer**—In the first set of experiments, we setup a file transfer of a 3 MB file between two users  $A$  and  $B$  connected via a multi-hop route. The TCP protocol was used to reliably transport the requested file. At the MAC layer, the RTS/CTS mechanism was enabled. The transmission rate was set to 11 Mbps at each link. The jammer was placed within the proximity of one of the intermediate hops of the TCP connection. Four jamming strategies were considered: (a) selective jamming of cumulative TCP-ACKs, (b) selective jamming of RTS/CTS messages, (c) selective jamming of data packets, and (d) random jamming of any packet. In each of the strategies, a fraction  $p$  of the targeted packets is jammed.

In Fig. 3(a), we show the average delay  $E[D]$  for completing the file transfer, as a function of the jamming probability  $p$  (averaged over repeated experiments). In Fig. 3(b), we show the average throughput  $E[T]$  as a function of  $p$ . It can be observed that all jamming attacks have significant impact on  $E[D]$  which grows several orders of magnitude larger compared to the delay in the absence of a jammer. Similarly, the effective throughput drops drastically under both random and selective jamming attacks. TCP performance under jamming of TCP-ACKs can be interpreted by the congestion control mechanism of the TCP protocol. When cumulative ACKs are lost (in our case jammed), the sender has to retransmit all unacknowledged data packets, thus increasing the incurred delay while reducing the effective throughput. At the same time, the sender interprets the loss of ACKs as congestion and throttles its packet transmission rate by reducing the size of the

transmission window. This leads to a further slow down of the application. Note that, for values of  $p > 0.4$ , the TCP connection is aborted for the case of random and TCP-ACK jamming, due to the repeated timeouts at the sender.

Fig. 3(c) depicts the number of packets that were jammed by the adversary for each value of  $p$ . Finally, Fig. 3(d) shows the fraction of time that the jammer remained active. Here, for selective jamming attacks, we assumed that 13% of the packet has to be corrupted in order to be dropped [17]. In the case of random jamming, the adversary is not aware of the type of packets transmitted (by means of processing the header of these packets). Hence, he is assumed to jam the entire packet in order to drop it. We observe that selective jamming requires the jamming of approximately one order of magnitude less packets than random jamming. This is because, as the packet transmission rate of the sender drops fewer packets need to be selectively targeted. Moreover, in selective jamming, the fraction of time the adversary remains active is *several orders of magnitude* less compared to random jamming. From Fig. 3(d), we observe that targeting control packets such as RTS/CTS messages and TCP-ACKs yields the lowest jamming activity, because control packets are significantly smaller compared to data packets. Such low-effort jamming attacks are not only efficient in terms of energy expenditure, but also challenging in localizing and physically removing the jamming devices. Typical methods of transmitter localization such as received signal strength and angle of arrival measurements require that the jamming device remains active for extended periods of time.

**Selective Jamming at the Network Layer**—In this scenario, we simulated a multi-hop wireless network of 35 nodes, randomly placed within a square area. The AODV routing protocol was used to discover and establish routing paths [19]. Connection requests were initiated between random source/destination pairs. Three jammers were strategically placed to selectively jam non-overlapping areas of the network. Three types of jamming strategies were considered: (a) a continuous jammer, (b) a random jammer blocking only a fraction  $p$  of the transmitted packets, and (c) a selective jammer targeting route request (RREQ) packets.

In Fig. 3(e), we show the number of connections established, normalized over the number of connections in the absence of the jammers. Fig. 3(f) shows the fraction of time that the jammer was active during our simulation, for each jamming strategy. We observe that a selective jamming attack against RREQ messages is equally effective to a constant jamming attack. However, selective jamming is several orders of magnitude more efficient as it is illustrated in Fig. 3(f). On the other hand, random jamming fails to disrupt the route discovery process due to the flooding mechanism of AODV.

## 5 HIDING BASED ON COMMITMENTS

In this section, we show that the problem of real-time packet classification can be mapped to the hiding property of commitment schemes, and propose a packet-hiding scheme based on commitments.

### 5.1 Mapping to Commitment Schemes

Commitment schemes are cryptographic primitives that allow an entity  $A$ , to commit to a value  $m$ , to an entity  $V$  while keeping  $m$  hidden. Commitment schemes are formally defined as follows [7].

*Commitment Scheme:* A commitment scheme is a two-phase interactive protocol defined as a triple  $\{\mathcal{X}, \mathcal{M}, \mathcal{E}\}$ . Set  $\mathcal{X} = \{A, V\}$  denotes two probabilistic polynomial-time interactive parties, where  $A$  is known as the committer and  $V$  as the verifier; set  $\mathcal{M}$  denotes the message space, and set  $\mathcal{E} = \{(t_i, f_i)\}$  denotes the events occurring at protocol stages  $t_i$  ( $i = 1, 2$ ), as per functions  $f_i$  ( $i = 1, 2$ ). During commitment stage  $t_1$ ,  $A$  uses a commitment function  $f_1 = \text{commit}()$  to generate a pair  $(C, d) = \text{commit}(m)$ , where  $(C, d)$  is called the commitment/decommitment pair. At the end of stage  $t_1$ ,  $A$  releases the commitment  $C$  to  $V$ . In the open stage  $t_2$ ,  $A$  releases the opening value  $d$ . Upon reception of  $d$ ,  $V$  opens the commitment  $C$ , by applying function  $f_2 = \text{open}()$ , thus obtaining a value of  $m' = \text{open}(C, d)$ . This stage culminates in either acceptance ( $m' = m$ ) or rejection ( $m' \neq m$ ) of the commitment by  $V$ . Commitment schemes satisfy the following two fundamental properties:

- *Hiding:* For every polynomial-time party  $V$  interacting with  $A$ , there is no (probabilistic) polynomially-efficient algorithm that would allow  $V$  to associate  $C$  with  $m$  and  $C'$  with  $m'$ , without access to the decommitment values  $d$  or  $d'$  respectively, and with non-negligible probability.
- *Binding:* For every polynomial-time party  $A$  interacting with  $V$ , there is no (probabilistic) polynomially-efficient algorithm that would allow  $A$  to generate a triple  $(C, d, d')$ , such that  $V$  accepts the commitments  $(C, d)$  and  $(C, d')$ , with non-negligible probability.

In our context, the role of the committer is assumed by the transmitting node  $S$ . The role of the verifier is assumed by any receiver  $R$ , including the jammer  $J$ . The committed value  $m$  is the packet that  $S$  wants to communicate to  $R$ . To transmit  $m$ , the sender computes the corresponding commitment/decommitment pair  $(C, d)$ , and broadcasts  $C$ . The hiding property ensures that  $m$  is not revealed during the transmission of  $C$ . To reveal  $m$ , the sender releases the decommitment value  $d$ , in which case  $m$  is obtained by all receivers, including  $J$ . Note that the hiding property, as defined in commitment schemes, does not consider the partial release of  $d$  and its implications on the partial reveal of  $m$ . In fact, a common way of opening commitments is by releasing the committed value itself [7].

For most applications, partial reveal of  $m$  with the partial release of  $d$  does not constitute a security risk. After all, the committer intends to reveal  $m$  by exposing  $d$ . However, in our context, a partial reveal of  $m$  while  $d$  is being transmitted can lead to the classification of  $m$  before the transmission of  $d$  is completed. Thus, the jammer has the opportunity to jam  $d$  instead of  $C$  once  $m$  has been classified. To prevent this scenario, we introduce the *strong hiding* property:



- *Strong Hiding*: For every polynomial-time party  $V$  interacting with  $A$  and possessing pairs  $(C, d_{part})$  and  $(C', d'_{part})$ , there is no (probabilistic) polynomially-efficient algorithm that would allow  $V$  associate  $C$  with  $m$  and  $C'$  with  $m'$ , with non-negligible probability. Here,  $d_{part}$  and  $d'_{part}$  are partial releases of  $d$  and  $d'$ , respectively, and the remaining parts of  $d$  and  $d'$  are assumed to be secret.

In the above definition, it is easily seen that the release of  $d_{part}$  must be limited to a fraction of  $d$ , in order for  $m$  to remain hidden. If a significant part of  $d$  becomes known to the verifier, trivial attacks, such as brute forcing the unknown bits of  $d$ , become possible.

## 5.2 A Strong Hiding Commitment Scheme (SHCS)

We propose a strong hiding commitment scheme (SHCS), which is based on symmetric cryptography. Our main motivation is to satisfy the strong hiding property while keeping the computation and communication overhead to a minimum. Assume that the sender  $S$  has a packet  $m$  for  $R$ . First,  $S$  constructs  $(C, d) = \text{commit}(m)$ , where,

$$C = E_k(\pi_1(m)), \quad d = k.$$

Here, the commitment function  $E_k()$  is an off-the-shelf symmetric encryption algorithm (e.g., DES or AES [27]),  $\pi_1$  is a publicly known permutation, and  $k \in \{0, 1\}^s$  is a randomly selected key of some desired key length  $s$  (the length of  $k$  is a security parameter). The sender broadcasts  $(C||d)$ , where “||” denotes the concatenation operation. Upon reception of  $d$ , any receiver  $R$  computes

$$m = \pi_1^{-1}(D_k(C)),$$

where  $\pi_1^{-1}$  denotes the inverse permutation of  $\pi_1$ . To satisfy the strong hiding property, the packet carrying  $d$  is formatted so that *all bits of  $d$  are modulated in the last few PHY layer symbols of the packet*. To recover  $d$ , any receiver must receive and decode the last symbols of the transmitted packet, thus preventing early disclosure of  $d$ . We now present the implementation details of SHCS.

## 5.3 Implementation Details of SHCS

The proposed SHCS requires the joint consideration of the MAC and PHY layers. To reduce the overhead of SHCS, the decommitment value  $d$  (i.e., the decryption key  $k$ ) is carried in the same packet as the committed value  $C$ . This saves the extra packet header needed for transmitting  $d$  individually. To achieve the strong hiding property, a sublayer called the “hiding sublayer” is inserted between the MAC and the PHY layer. This sublayer is responsible for formatting  $m$  before it is processed by the PHY layer. The functions of the hiding sublayer are outlined in Fig. 4.

Consider a frame  $m$  at the MAC layer delivered to the hiding sublayer. Frame  $m$  consists of a MAC header and the payload, followed by the trailer containing the CRC code. Initially,  $m$  is permuted by applying a publicly known permutation  $\pi_1$ . The purpose of  $\pi_1$  is to randomize the

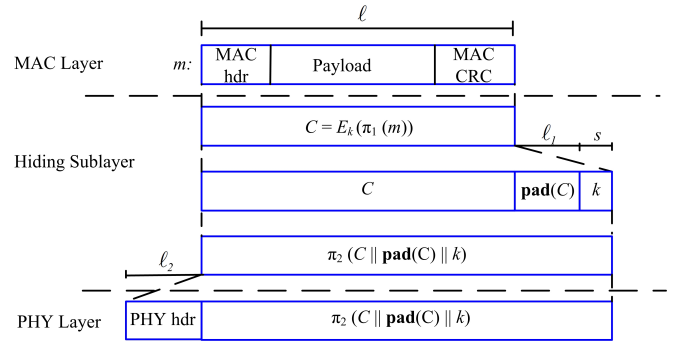


Fig. 4. Processing at the hiding sublayer.

input to the encryption algorithm and delay the reception of critical packet identifiers such as headers. After the permutation,  $\pi_1(m)$  is encrypted using a *random* key  $k$  to produce the commitment value  $C = E_k(\pi_1(m))$ . Although the random permutation of  $m$  and its encryption with a random key  $k$  seemingly achieve the same goal (i.e., the randomization of the ciphertext), in Section 5.4 we show that both are necessary to achieve packet hiding.

In the next step, a padding function  $\text{pad}()$  appends  $\text{pad}(C)$  bits to  $C$ , making it a multiple of the symbol size. Finally,  $C||\text{pad}(C)||k$  is permuted by applying a publicly known permutation  $\pi_2$ . The purpose of  $\pi_2$  is to ensure that the interleaving function applied at the PHY layer does not disperse the bits of  $k$  to other symbols. We now present the padding and permutation functions in detail.

**Padding**—The purpose of padding is to ensure that  $k$  is modulated in the minimum number of symbols needed for its transmission. This is necessary for minimizing the time for which parts of  $k$  become exposed. Let  $\ell_1$  denote the number of bits padded to  $C$ . For simplicity, assume that the length of  $C$  is a multiple of the block length of the symmetric encryption algorithm and hence, has the same length  $\ell$  as the original message  $m$ . Let also  $\ell_2$  denote the length of the header added at the PHY layer. The frame carrying  $(C, d)$  before the encoder has a length of  $(\ell + \ell_1 + \ell_2 + s)$  bits. Assuming that the rate of the encoder is  $\alpha/\beta$  the output of the encoder will be of length,  $\frac{\alpha}{\beta}(\ell + \ell_1 + \ell_2 + s)$ . For the last symbol of transmission to include  $\frac{\alpha}{\beta}q$  bits of the key  $k$ , it must hold that,

$$\ell_1 = \frac{\alpha}{\beta} \left( q - \left( (\ell + \ell_2) \frac{\beta}{\alpha} \right) \bmod q \right). \quad (3)$$

**Permutation**—The hiding layer applies two publicly known permutations  $\pi_1$  and  $\pi_2$  at different processing stages. Permutation  $\pi_1$  is applied to  $m$  before it is encrypted. The purpose of  $\pi_1$  is twofold. First, it distributes critical frame fields which can be used for packet classification across multiple plaintext blocks. Hence, to reconstruct these fields, all corresponding ciphertext blocks must be received and decrypted. Moreover, header information is pushed at the end of  $\pi_1(m)$ . This prevents early reception of the corresponding ciphertext blocks.

For example, consider the transmission of a MAC frame of length 2,336 bytes which carries a TCP data packet. The

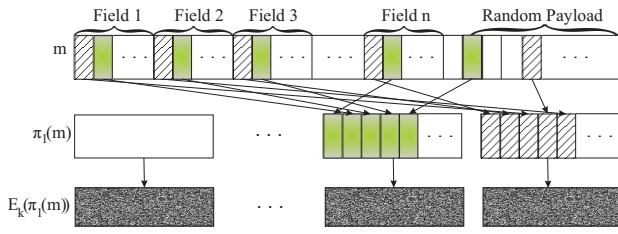


Fig. 5. Application of permutation  $\pi_1$  on packet  $m$ .

MAC header is 28 bytes long and has a total of 18 distinct fields. TCP header is 20 bytes long (assuming no optional fields) and has 17 distinct fields. Assume the encryption of a fixed block of 128 bits. Packet  $\pi_1(m)$  is partitioned to 146 plaintext blocks  $\{p_1, p_2, \dots, p_{146}\}$ , and is encrypted to produce 146 ciphertext blocks  $C = c_1 || c_2 || \dots || c_{146}$ . Each field of the TCP and MAC headers is distributed bit-by-bit from the most significant bit (MSB) to the least significant bit (LSB) to each of the plaintext blocks in the reverse block order. This process is depicted in Fig. 5.

For fields longer than one bit, bits are numbered from the LSB to the MSB and are placed in reverse order to each plaintext block. To recover any field  $i$  that is  $\ell_i$  bits long, the last  $\ell_i$  ciphertext blocks must be received and decrypted. If  $\ell_i > \ell_b$ , where  $\ell_b$  denotes the ciphertext block length, the bit placement process continues in a round-robin fashion. The second goal of the permutation  $\pi_1$  is to randomize the plaintext blocks. Assuming a random payload, the permutation distributes the payload bits to all plaintext blocks processed by the encryption function, thus randomizing each ciphertext block.

Permutation  $\pi_2$  is applied to reverse the effects of interleaving on the bits of  $k$ , so that  $k$  is contained at the packet trailer. Interleaving can be applied across multiple frequencies on the same symbol (e.g., in OFDM), or it may span multiple symbols [9]. For example, consider a  $d \times \beta$  block interleaver. Without loss of generality, assume that  $\beta = q$ , and let the last  $n$  rows of the last block passed via the interleaver correspond to the encoded version of the random key  $k$ . Permutation  $\pi_2$  rearranges the bits of  $k$  at the interleaver matrix  $A_{d \times \beta}$  in such a way that all bits of  $k$  appear in the last  $n$  columns. Therefore, the bits of  $k$  will be modulated as the last  $n$  symbols of the transmitted packet. Note that this operation affects only the interleaver block(s) that carries  $k$ . For the rest of the packet, the interleaving function is performed normally, thus preserving the benefits of interleaving. For PHY layer implementations in which interleaving is applied on a per-symbol basis (e.g., 802.11a and 802.11g), the application of permutation  $\pi_2$  is not necessary.

#### 5.4 Security Analysis

In this section, we analyze the security of SHCS by evaluating the ability of  $J$  in classifying a transmitted packet at different stages of the packet transmission.

**Release of  $C$** —We first examine if  $J$  can classify  $m$  by observing the commitment value  $C$ . Though  $C$  and  $k$  are part of the same packet, symbols corresponding to  $C$  are

received first. The jammer can attempt to classify  $m$  by launching a ciphertext-only attack on  $C$  as early as the reception of the first ciphertext block. Because the encryption key is refreshed at every transmission, a very small number of ciphertext blocks are available for cryptanalysis. Appropriate selection of the key length  $s$  can prevent this type of attack. Note that  $s$  can be well below the cryptographic standards, due to the limited time available to the adversary (until the transmission is completed). For instance, a 56-bit long DES key is more than adequate for our purposes, since the fastest known brute force attack on DES takes almost a day [24]. Other types of known attacks such as differential and linear cryptanalysis are not applicable, because they require the collection of a large number of chosen or known plaintext/ciphertext pairs [27].

Even if the key for a particular packet is revealed to the adversary, packet classification is delayed until the end of  $C$ 's transmission. The application of the permutation function  $\pi_1$  distributes frame fields to ciphertext blocks in the reverse order of transmission, with the MSBs from each field appearing on the last ciphertext block. Hence, reception of all blocks of  $C$  is required for the complete recovery of headers. To minimize the communication overhead,  $k$  must be selected to be of the smallest length adequate for the protection of  $C$ , for the time required to transmit one packet. However, special care must be taken to withstand codebooks attacks on  $k$ . In such attacks, the adversary can encrypt a particular message of interest with all possible keys and construct a look-up table (codebook) of all possible ciphertexts. If the encryption of all possible messages with all possible keys results in unique ciphertexts, there is a 1-1 correspondence between a ciphertext and the generating plaintext/key pair. This property is guaranteed with high probability when the plaintext space  $\mathcal{M}$  and the key space  $\mathcal{K}$  are much smaller than the ciphertext space  $\mathcal{C}$ . Assuming the encryption of a plaintext block  $m_i$  with a key  $k_i$  randomly maps to a ciphertext  $c_i = E_{k_i}(m_i)$ , every ciphertext  $c_i \in \mathcal{C}$  occurs with probability  $p_c = \frac{1}{|\mathcal{C}|}$ . The problem of finding the probability that all  $|\mathcal{M}||\mathcal{K}|$  ciphertexts produced by the encryption of all plaintexts with all keys are unique, can be formulated as a "birthday problem" [27]:

$$\Pr[\text{ciphertexts unique}] \approx e^{\frac{-|\mathcal{M}||\mathcal{K}|(|\mathcal{M}||\mathcal{K}|-1)}{2|\mathcal{C}|}}.$$

As an example, consider the encryption of a message  $m = \{m_1, m_2, \dots, m_x\}$  with a key  $k$  of length 56 bits, using blocks of 128 bits. For a fairly small plaintext space (e.g.,  $|\mathcal{M}| = 16$ ), the probability of ciphertext uniqueness is equal to 99.8%. Thus, the adversary can recover  $k$ , by launching a codebook attack on  $m_1$ . The remaining  $c_i$ 's are decrypted in real-time, using the known value of  $k$ . Here, the plaintext space for  $m_1$  is considered to be small because of the structure imposed by the static header of a packet (all fields of the header are known to the adversary). Randomization of the plaintext, ensures that all plaintexts are possible, thus equating the plaintext space with the ciphertext space.

**Partial release of  $d$** —Depending on the PHY layer implementation,  $d = k$  requires  $n \geq 1$  symbols for its transmission. Hence, part of  $k$  may become known before the completion of the transmission of the packet at hand. This release reduces the search space for a brute force attack on  $k$ . Assume that the adversary pro-actively jams a few symbols below the ECC capability of the receiver during the transmission of  $C$ . In the best case, he can postpone his decision to jam a transmitted packet until the transmission of the last symbol (jam one more symbol to drop the packet). He must therefore complete the classification process before the last symbol is transmitted. Assuming that the adversary waits until the maximum number of bits of  $k$  are released, the key search space before the transmission of the last two symbols is equal to  $2^{\frac{\beta}{2}q}$  keys. The adversary must be capable of performing on average  $\mathcal{N} = 2^{(2\frac{\beta}{2}q-1)}\mathcal{R}$  decryptions per second in order to find  $k$  before the last symbol is transmitted<sup>2</sup>. Here, we have assumed that, on average, half the key space must be searched.

For example, assume an 802.11a PHY layer operating at 6 Mbps, with every symbol carrying 24 bits of information. Consider  $k$  to be a 56-bit DES key, fitting in three symbols. The computational capability of the adversary must be equal to  $\mathcal{N} \approx 3.52 \times 10^{19}$  decryptions/sec in order to recover  $k$  before the completion of the packet transmission. The fastest known hardware implementation of a DES cracker achieves a throughput of  $2.92 \times 10^{11}$  keys per second [24]. For an operating rate of 54 Mbps, all 56 bits of the key  $k$  fit in one symbol (symbol size is 216 bits), thus preventing the partial release of the decommitment value  $d$ .

A brute force attack on  $k$  may be successful if  $q \leq \frac{\beta}{2\alpha} \log_2 \mathcal{N} + 1$ . For instance, when  $\mathcal{N} = 2.92 \times 10^{11}$ , the adversary can find  $k$  if  $q \leq 19$  bits. In fact, for small values of  $q$  (e.g., 4), the adversary can launch the brute force attack on  $k$ , several symbols before the end of  $k$ 's transmission. Therefore, SHCS is suitable for PHY layer implementations where the number of bits per symbol  $q$  is sufficiently large. Note that our security analysis has excluded all processing delays from the time that symbols are received to the time that they become available for cryptanalysis.

**Binding property**—The binding property is not a security requirement of SHCS under our adversary model. Since the primary goal of any sender  $S$  in the network is to communicate  $m$ ,  $S$  has no interest in modifying  $m$  after he has committed to it. However, under a more general adversary model, the jammer may launch denial of service attacks by making the receiver  $R$  to accept a  $k' \neq k$  such that  $m' = D_{k'}(C)$  is a meaningful message. Even though SHCS is not designed to ensure the binding property of commitment schemes, generating a  $k' \neq k$  that opens a valid value of  $m' \neq m$  is a computationally hard task. In order to find such a  $k'$ , the jammer has to launch a brute force attack on  $C$ . Here, not only the attack must be performed in a timely manner, but  $m'$  has to be in the right

format (source/destination address must be in the context of the communications, CRC code must be valid, etc). Given that  $k$  is transmitted right after  $C$ , the jammer has no time to find an appropriate  $k'$  that would lead to the decryption of an acceptable  $m'$ , assuming that such  $m'$  exists. If  $m'$  is not meaningful, substituting  $k$  with  $k'$  is equivalent to a jamming attack on  $m$  without classification (no selectivity).

The binding property can be theoretically achieved if a random string  $r$  is appended to  $m$  [23]. In this case, the commitment/decommitment pair  $(C, d)$  is,

$$C = (\gamma, \delta) = (E_k(m||r), r), \quad d = k.$$

Provided that  $r$  is sufficiently long, a computationally bounded jammer cannot find a  $k'$  such that  $D_{k'}(C) = m' || r$ . In this case,  $r$  preserves the integrity of message  $m$ . Since the addition of  $r$  is not necessary for preventing real-time classification of  $m$ , we leave the implementation of the binding property to the discretion of the system designer.

## 5.5 Resource Overhead of SHCS

In this section we analyze the per-packet communication and computational overhead of SHCS.

**Communication Overhead**—For every packet  $m$ , a random key  $k$  of length  $s$  is appended. Also,  $(\ell_b - (\ell \bmod \ell_b))$  bits of overhead are added by the encryption algorithm, to convert a plaintext of length  $\ell$  to a multiple of the encryption block. Thus, the communication overhead of SHCS is equal to  $s + (\ell_b - (\ell \bmod \ell_b))$ , per packet. Here, we do not account for the padding string  $\text{pad}(C)$ , because the addition of  $\text{pad}(C)$  does not increase the number of transmitted symbols.

**Computation Overhead**—The computation overhead of SHCS is one symmetric encryption at the sender and one symmetric decryption at the receiver. Because the header information is permuted as a trailer and encrypted, all receivers in the vicinity of a sender must receive the entire packet and decrypt it, before the packet type and destination can be determined. However, in wireless protocols such as 802.11, the complete packet is received at the MAC layer before it is decided if the packet must be discarded or be further processed [9]. If some parts of the MAC header are deemed not to be useful information to the jammer, they can remain unencrypted in the header of the packet, thus avoiding the decryption operation at the receiver.

## 6 HIDING BASED ON CRYPTOGRAPHIC PUZZLES

In this section, we present a packet hiding scheme based on cryptographic puzzles. The main idea behind such puzzles is to force the recipient of a puzzle execute a pre-defined set of computations before he is able to extract a secret of interest. The time required for obtaining the solution of a puzzle depends on its hardness and the computational ability of the solver [10]. The advantage of the puzzle-based scheme is that its security does not rely on the PHY layer parameters. However, it has higher computation and communication overhead.

2. A more accurate calculation of  $\mathcal{N}$  would assume an adversary trying a brute force attack on  $k$ , with the reception of the first ciphertext block, and adjusting the searching space according to the partial release of  $k$ .



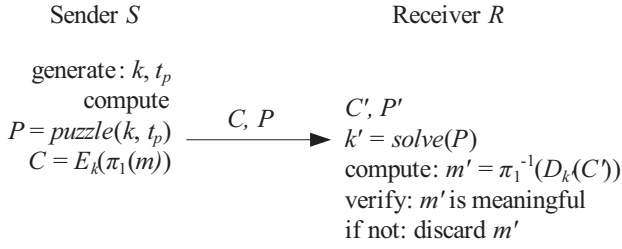


Fig. 6. The cryptographic puzzle-based hiding scheme.

In our context, we use cryptographic puzzles to temporarily hide transmitted packets. A packet  $m$  is encrypted with a randomly selected symmetric key  $k$  of a desirable length  $s$ . The key  $k$  is blinded using a cryptographic puzzle and sent to the receiver. For a computationally bounded adversary, the puzzle carrying  $k$  cannot be solved before the transmission of the encrypted version of  $m$  is completed and the puzzle is received. Hence, the adversary cannot classify  $m$  for the purpose of selective jamming.

### 6.1 Cryptographic Puzzle Hiding Scheme (CPHS)

Let a sender  $S$  have a packet  $m$  for transmission. The sender selects a random key  $k \in \{0,1\}^s$ , of a desired length.  $S$  generates a puzzle  $P = \text{puzzle}(k, t_p)$ , where  $\text{puzzle}()$  denotes the puzzle generator function, and  $t_p$  denotes the time required for the solution of the puzzle. Parameter  $t_p$  is measured in units of time, and it is directly dependent on the assumed computational capability of the adversary, denoted by  $\mathcal{N}$  and measured in computational operations per second. After generating the puzzle  $P$ , the sender broadcasts  $(C, P)$ , where  $C = E_k(\pi_1(m))$ . At the receiver side, any receiver  $R$  solves the received puzzle  $P'$  to recover key  $k'$  and then computes  $m' = \pi_1^{-1}(D_{k'}(C))$ . If the decrypted packet  $m'$  is meaningful (i.e., is in the proper format, has a valid CRC code, and is within the context of the receiver's communication), the receiver accepts that  $m' = m$ . Else, the receiver discards  $m'$ . Fig. 6 shows the details of CPHS.

### 6.2 Implementation Details of CPHS

In this section, we consider several puzzle schemes as the basis for CPHS. For each scheme, we analyze the implementation details which impact security and performance. Cryptographic puzzles are primitives originally suggested by Merkle as a method for establishing a secret over an insecure channel [16]. They find a wide range of applications from preventing DoS attacks to providing broadcast authentication and key escrow schemes.

**Time-lock Puzzles**—Rivest et al. proposed a construction called *time-lock puzzles*, which is based on the iterative application of a precisely controlled number of modulo operations [22]. Time-lock puzzles have several attractive features such as the fine granularity in controlling  $t_p$  and the sequential nature of the computation. Moreover, the puzzle generation requires significantly less computation compared to puzzle solving.

In a time-lock puzzle, the puzzle constructor generates a composite modulus  $g = u \cdot v$ , where  $u$  and  $v$  are two large random prime numbers. Then, he picks a random  $a$ ,  $1 < a < g$  and hides the encryption key in  $K_h = k + a^{2^t} \bmod g$ , where  $t = t_p \cdot \mathcal{N}$ , is the amount of time required to solve for  $k$ . Here, it is assumed that the solver can perform  $\mathcal{N}$  squarings modulo  $g$  per second. Note that  $K_h$  can be computed efficiently if  $\phi(g) = (u-1)(v-1)$  or the factorization of  $g$  are known, otherwise a solver would have to perform all  $t$  squarings to recover  $k$ . The puzzle consists of the values  $P = (g, K_h, t, a)$ .

In our setup, the value of the modulus  $g$  is known a priori and need not be communicated (may change periodically). The sender reveals the rest of the puzzle information in the order  $(K_h, t, a)$ . Note that if any of  $t, a$  are unknown, any value of  $k$  is possible [22].

**Puzzles based on hashing**—Computationally limited receivers can incur significant delay and energy consumption when dealing with modulo arithmetic. In this case, CPHS can be implemented from cryptographic puzzles which employ computationally efficient cryptographic primitives. Client puzzles proposed in [10], use one-way hash functions with partially disclosed inputs to force puzzle solvers search through a space of a precisely controlled size. In our context, the sender picks a random key  $k$  with  $k = k_1 || k_2$ . The lengths of  $k_1$  and  $k_2$  are  $s_1$ , and  $s_2$ , respectively. He then computes  $C = E_k(\pi_1(m))$  and transmits  $(C, k_1, h(k))$  in this particular order. To obtain  $k$ , any receiver has to perform on average  $2^{s_2-1}$  hash operations (assuming perfect hash functions). Because the puzzle cannot be solved before  $h(k)$  has been received, the adversary cannot classify  $m$  before the completion of  $m$ 's transmission.

### 6.3 Security Analysis of CPHS

With the completion of the transmission of  $P$ , any receiver can recover  $m$ . Therefore, a selective jammer must attempt to classify  $m$  before the transmission of  $P$  has been completed. We analyze the security of CPHS at different stages of its execution.

**Reception of  $C$** —The jammer can attempt to classify  $m$  by cryptanalyzing ciphertext  $C = E_k(\pi_1(m))$ . This attack is identical to the effort of classifying  $m$  with the transmission of  $C$  at the SHCS. The same analysis presented in Section 5.4 holds for the case of CPHS. The selection of a key of adequate length (e.g., 56-bit DES key) is sufficient to prevent both ciphertext-only and codebook attacks.

**Solving  $P$** —The transmission of  $k$  in the form of a puzzle  $P$  prevents any receiver from recovering  $k$  for at least time  $t_p$ , after the puzzle has been received. A jammer may try to guess and solve  $P$  before its transmission is completed. In the best case, the adversary must finish the classification of  $m$  before the transmission of the last symbol of  $P$ . The number of possible puzzle values at the beginning of the second to last symbol are  $2^{2^{\frac{a}{g}}}$ . Assuming a brute force attack on the missing bits of the puzzle, the computational load of the adversary increases on average to  $2^{2^{\frac{a}{g}}-1} t_p$ .

The value of  $t_p$  has already been selected to prevent the puzzle solution until its transmission is completed. Hence, early solution of  $P$  before all its bits are received cannot be achieved. Note that the security of CPHS is not dependent on the PHY layer parameter  $q$ , but on the selection of  $t_p$ . Therefore, this method is applicable even to wireless systems where  $q$  obtains relatively small values.

#### 6.4 Resource Overhead of CPHS

**Communication Overhead**—The per-packet communication overhead of CPHS is equal to the length of  $P$ , in addition to the padding added by the encryption function. If the puzzle is realized using time-locks, the length of  $P$  is equal to the lengths of  $K_h, a$ , and  $t$ . The value  $K_h$  is computed modulo  $g$  and has the same length as  $g$ . Similarly,  $a$  has a length equal to the length of  $g$ . The size of  $t$  is potentially smaller than  $a, g$ , and  $K_h$ , and depends on the computational capability of the adversary. The security of time locks depends on the difficulty in factoring  $g$  or finding  $\phi(g)$ , where  $\phi()$  denotes the Euler  $\phi$ -function. Typical values of  $g$  are in the order of 1,024 bits [27]. Since messages need to remain hidden for only a short period of time, the modulo can be chosen to be of much smaller size and be periodically refreshed. In the case of hash-based puzzles, the communication overhead is equal to the transmission of the key  $k_1$  which is of length  $s_1$  and the hash value  $h(k)$ . The typical length of hash function is 160 bits [27].

**Computation Overhead**—In time-lock puzzles, the sender has to apply one permutation on  $m$ , perform one symmetric encryption, and one modulo squaring operation to hide  $k$ . On the receiver side, the receiver has to perform  $t$  modulo squaring operations to recover  $k$ , one symmetric decryption to recover  $\pi_1(m)$ , and apply the inverse permutation. In the case of hash-based puzzles, the modulo squaring operation is substituted by, on average,  $2^{s_2-1}$  hashing operations.

### 7 HIDING BASED ON ALL-OR-NOTHING TRANSFORMATIONS

In this section, we propose a solution based on All-Or-Nothing Transformations (AONT) that introduces a modest communication and computation overhead. Such transformations were originally proposed by Rivest to slow down brute force attacks against block encryption algorithms [21]. An AONT serves as a publicly known and completely invertible pre-processing step to a plaintext before it is passed to an ordinary block encryption algorithm. A transformation  $f$ , mapping message  $m = \{m_1, \dots, m_x\}$  to a sequence of pseudo-messages  $m' = \{m'_1, \dots, m'_x\}$ , is an AONT if [21]: (a)  $f$  is a bijection, (b) it is computationally infeasible to obtain any part of the original plaintext, if one of the pseudo-messages is unknown, and (c)  $f$  and its inverse  $f^{-1}$  are efficiently computable.

When a plaintext is pre-processed by an AONT before encryption, all ciphertext blocks must be received to obtain any part of the plaintext. Therefore, brute force attacks are slowed down by a factor equal to the number of ciphertext

blocks, without any change on the size of the secret key. Note that the original AONT proposed in [21] is computationally secure. Several AONT schemes have been proposed that extend the definition of AONT to undeniable security [26]. Under this model, all plaintexts are equiprobable in the absence of at least one pseudo-message.

#### 7.1 An AONT-based Hiding Scheme (AONT-HS)

In our context, packets are pre-processed by an AONT before transmission but remain unencrypted. The jammer cannot perform packet classification until all pseudo-messages corresponding to the original packet have been received and the inverse transformation has been applied. Packet  $m$  is partitioned to a set of  $x$  input blocks  $m = \{m_1, \dots, m_x\}$ , which serve as an input to an AONT  $f : \{\mathbb{F}_u\}^x \rightarrow \{\mathbb{F}_u\}^{x'}$ . Here,  $\mathbb{F}_u$  denotes the alphabet of blocks  $m_i$  and  $x'$  denotes the number of output pseudo-messages with  $x' \geq x$ . The set of pseudo-messages  $m' = \{m'_1, \dots, m'_{x'}\}$  is transmitted over the wireless medium. At the receiver, the inverse transformation  $f^{-1}$  is applied after all  $x'$  pseudo-messages are received, in order to recover  $m$ .

#### 7.2 Implementation details of the AONT-HS

In this section, we describe two AONTs which can be employed in AONT-HS; a linear transformation [26], and the original package transformation [21].

**Linear AONT**—In [26], Stinson showed how to construct a linear AONT when the alphabet of the input blocks is a finite field  $\mathbb{F}_u$ , with the order  $u$  being a prime power. He showed that if an invertible matrix  $M = \{m_{ij} | m_{ij} \in \mathbb{F}_u, m_{ij} \neq 0\}_{x \times x}$  exists, then the transformation  $f(m) = mM^{-1}$  is a linear AONT. He also provided a method for constructing such  $M$  which is as follows.

Let  $u = v^i$ , where  $v$  is prime and  $i$  is a positive integer. Choose  $\lambda \in \mathbb{F}_u$  such that  $\lambda \notin \{n-1 \pmod{v}, n-2 \pmod{v}\}$  and define the linear AONT  $LT$  to be,

$$LT = \begin{bmatrix} 1 & 0 & \dots & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 1 \\ 1 & 1 & \dots & 1 & \lambda \end{bmatrix} \quad (4)$$

Given  $m = \{m_1, \dots, m_x\}$ ,

$$m'_x = \lambda m_x + \sum_{j=1}^{x-1} m_j, \quad m'_i = m_i + m'_x, \quad 1 \leq i \leq (x-1). \quad (5)$$

Conversely, given  $m' = \{m'_1, \dots, m'_{x'}\}$ , the original input  $m = \{m_1, \dots, m_x\}$  is recovered as follows:

$$m_i = m'_i - m'_x, \quad 1 \leq i \leq (x-1), \quad (6)$$

$$m_x = \gamma(m'_1 + \dots + m'_{x-1} - m'_x), \quad \gamma = \frac{1}{n - \lambda - 1}. \quad (7)$$

Note from (6), (7) that if any of the  $\{m'_i\}$  is missing, all values of  $m_i$  are possible, for every  $i$ . Thus, the linear AONT provides undeniable security.

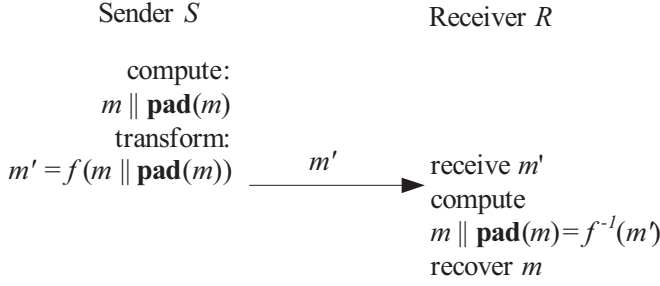


Fig. 7. The AONT-based Hiding Scheme (AONT-HS).

**The Package Transform**—In the package transform [21], given a message  $m$ , and a random key  $k'$ , the output pseudo-messages are computed as follows:

$$m'_i = m_i \oplus E_{k'}(i), \text{ for } i = 1, 2, \dots, x \quad (8)$$

$$m'_{x+1} = k' \oplus e_1 \oplus e_2 \oplus \dots \oplus e_x, \quad (9)$$

where  $e_i = E_{k_0}(m'_i \oplus i)$ , for  $i = 1, 2, \dots, x$ , and  $k_0$  is a fixed publicly-known encryption key. With the reception of all pseudo-messages message  $m$  is recovered as follows:

$$k' = m'_{x+1} \oplus e_1 \oplus e_2 \oplus \dots \oplus e_x, \quad (10)$$

$$m_i = m'_i \oplus E_{k'}(i), \text{ for } i = 1, 2, \dots, x. \quad (11)$$

Note that if any  $m'_i$  is unknown, any value of  $k'$  is possible, because the corresponding  $e_i$  is not known. Hence,  $E_{k'}(i)$  cannot be recovered for any  $i$ , making it infeasible to obtain any of the  $m_i$ .

**Hiding Sublayer Details**—AONT-HS is implemented at the hiding sublayer residing between the MAC and the PHY layers. In the first step,  $m$  is padded by applying function  $\text{pad}()$  to adjust the frame length so that no padding is needed at the PHY layer, and the length of  $m$  becomes a multiple of the length of the pseudo-messages  $m'_i$ . This will ensure that all bits of the transmitted packet are part of the AONT. In the next step,  $m \parallel \text{pad}(m)$  is partitioned to  $x$  blocks, and the AONT  $f$  is applied. Message  $m'$  is delivered to the PHY layer. At the receiver, the inverse transformation  $f^{-1}$  is applied to obtain  $m \parallel \text{pad}(m)$ . The padded bits are removed and the original message  $m$  is recovered. The steps of AONT-HS are shown in Fig. 7.

### 7.3 Security Analysis of the AONT-HS

**Partial reception of  $m'_i$ ,  $i < x'$** —In the AONT-HS, the jammer may attempt to classify  $m$  without receiving all  $m'_i$  ( $1 \leq i \leq x'$ ). By definition, AONTs prevent the computation of any part of  $m$  without the reception of all the pseudo-messages. In fact, for the linear AONT, undeniable security is achieved. The jammer can launch a brute force attack on  $m$  as early as the reception of  $m'_1$ . However, the system of equations formed by  $m'_i$ 's when at least one is missing, has a number of solutions equal to the message space. All these solutions are equiprobable.

**Partial release of  $m'_x$** —With the partial release of the last pseudo-message  $m'_x$ , the space of the possible original messages  $m$  is reduced. As stated by our adversarial model, the classification of  $m$  must be completed before the last

symbol of  $m'_x$ , is transmitted. The search space for  $m'_x$  is reduced to its smallest value before the transmission of the last two symbols, in which case the possible values of  $m$  are equal to  $2^{2^{\frac{\alpha}{\beta}q}}$ . The adversary must be capable of solving on average  $2^{2^{\frac{\alpha}{\beta}q-1}}$  systems of linear equations in time equal to the length of one symbol ( $\frac{1}{R}$  sec), in the case of the linear AONT, or perform the same number of decryptions for the case of the package transform. For instance when  $q = 48$  and  $\alpha/\beta = 1/2$  (802.11a), the search space is equal to  $1.4 \times 10^{14}$ . As in the case of SHCS, when the value of  $q$  becomes small ( $q \leq \frac{\beta}{2\alpha} \log_2 \mathcal{N} + 1$ ), a brute force attack on  $m$  is possible. Therefore, AONT-HS is suitable for PHY layer implementations where  $q$  is sufficiently large.

### 7.4 Resource Overhead of the AONT-HS

**Communication Overhead**—In AONT-HS, the original set of  $x$  messages is transformed to a set of  $x'$  pseudo-messages, with  $x' \geq x$ . Additionally, the function  $\text{pad}()$  appends  $(\ell_b - (\ell \bmod \ell_b))$  bits in order to make the length of  $m$  a multiple of the length  $\ell_b$  of the pseudo-messages  $m'$ . Hence, the communication overhead introduced is  $(\ell_b(x' - x) + \ell_b - (\ell \bmod \ell_b))$  bits. For the linear AONT,  $x = x'$ , and therefore, only the padding communication overhead is introduced. For the package transform, the overhead is equal to the length of one pseudo-message ( $x' = x + 1$ ).

**Computation Overhead**—The linear AONT requires only elementary arithmetic operations such as string addition and multiplication, making it particularly fast due to its linear nature. The package transform requires  $x'$  symmetric encryptions at the sender and an equal amount of symmetric decryptions at the receiver. Note that the length of the plaintext for the  $x'$  encryptions is relatively small compared to the length of message  $m$  (indexes  $1, \dots, x$  are encrypted). Therefore, only one ciphertext block is produced per pseudo-message. Assuming a pseudo-message block size equal to the ciphertext block size  $\ell_b$ , the computational overhead of the  $x'$  encryptions required by the package transform is equivalent to the overhead of one encryption of a message of length  $\ell + \ell_b$ .

## 8 EVALUATION OF PACKET-HIDING TECHNIQUES

In this section, we evaluate the impact of our packet-hiding techniques on the network performance via extensive simulations. We used the OPNET<sup>TM</sup> Modeler 14.5 [18] to implement the hiding sublayer and measure its impact on the effective throughput of end-to-end connections and on the route discovery process in wireless ad-hoc networks. We chose a set of nodes running 802.11b at the PHY and MAC layers, AODV for route discovery, and TCP at the transport layer. Aside from our methods, we also implemented a simple MAC layer encryption with a static key.

**Impact on real-time systems**—Our packet-hiding methods require the processing of each individual packet by the hiding sublayer. We emphasize that the incurred processing delay is acceptable, even for real-time applications. The SCHS requires the application of two permutations and one

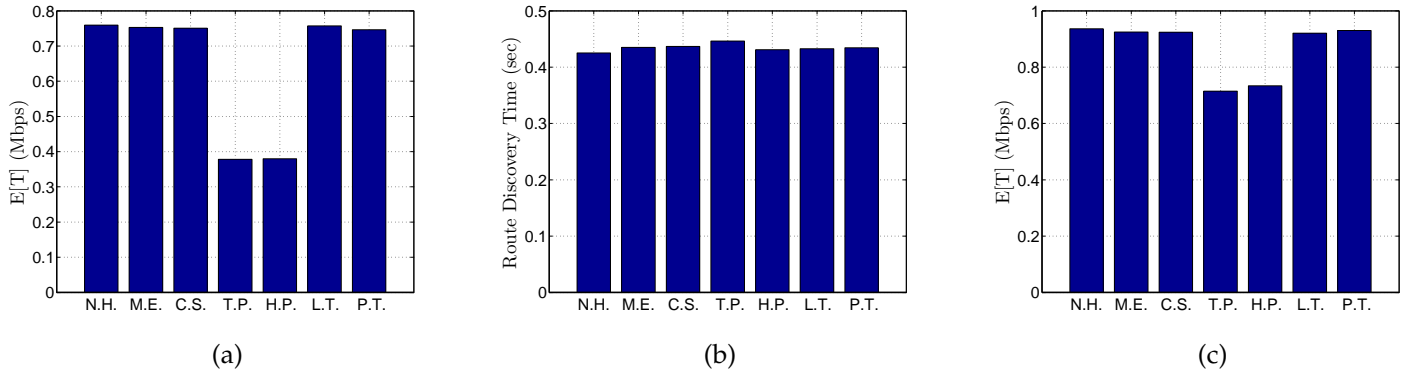


Fig. 8. (a) Average effective throughput (line network topology), (b) average route discovery time (non-congested network), (c) average effective throughput (congested network). N.H.: No packet hiding, M.E.: MAC-layer encryption with a static key, C.S.: SHCS, T.P.: Time-lock CPHS, H.P.: Hash-based CPHS, L.T.: Linear AONT-HS, P.T.: Package transform.

symmetric encryption at the sender, while the inverse operations have to be performed at the receiver. Such operations can be implemented in hardware very efficiently. Symmetric encryption such as AES can be implemented at speeds of tens of Gbps/s when realized with Application Specific Integrated Circuits (ASICs) or Field Programmable Gate Arrays (FPGAs) [6]. These processing speeds are orders of magnitude higher than the transmission speeds of most current wireless technologies, and hence, do not impose a significant delay.

Similarly, the AONT-HS performs linear operations on the packet that can be efficiently implemented in hardware. We note that a non-negligible processing delay is incurred by the CPHS. This is due to the cryptographic puzzle that must be solved at the receiver. As suggested in Section 6, CPHS should only be employed when the symbol size at the PHY layer is too small to support the SHCS and AONT-HS solutions. The processing delays of the various schemes are taken into account in our experimental evaluations.

**Experimental evaluation**—In the first set of experiments, we setup a single file transfer between a client and server, connected via a multi-hop route. The client requested a 1 MB file from the server. We evaluated the effects of packet hiding by measuring the effective throughput of the TCP connection in the following scenarios: (a) No packet hiding (N.H.), (b) MAC-layer encryption with a static key (M.E.), (c) SHCS (C.S.), (d) Time-lock CPHS (T.P.), (e) Hash-based CPHS (H.P.), (f) Linear AONT-HS (L.T.), and (g) AONT-HS based on the package transform (P.T.).

In Fig. 8(a), we show the effective throughput averaged over 100 different traces. We observe that MAC-layer encryption, SHCS and the linear AONT-HS achieve an effective throughput close to the throughput in the absence of packet hiding. This is justified by the relatively small communication overhead of each hiding method and the small queuing delay at intermediate routers due to the absence of any cross traffic. The AONT-HS based on the package transform achieved slightly lower throughput, because it occurs a per-packet overhead of 128 bits as opposed to 56 bits for SHCS. We also observe that hiding

techniques based on cryptographic puzzles decrease the effective throughput of the TCP connection to half, compared to the no hiding case. This performance is anticipated since the time required to solve a puzzle after a packet has been received at the MAC layer is equal to the transmission time of each packet. While this constitutes a significant performance reduction, we emphasize that cryptographic puzzles were suggested as a candidate solution only when the symbol size is so small that more efficient hiding methods do not provide adequate levels of security.

In the second set of experiments, we studied the impact of packet hiding on the route discovery process in an ad-hoc network. We generated a random topology of 54 nodes placed in an area of  $500 \times 400 \text{ m}^2$ . Nodes discovered routes using the AODV routing protocol. A total of twenty client/server pairs exchanged messages of size 1 KB using the TCP protocol, at randomly chosen starting times. The size of the message exchanged between pairs of nodes was kept small in order to avoid skewing of the route discovery performance due to network congestion.

The average route discovery delay is shown in Fig. 8(b). This delay is defined as the time difference between the transmission of the first RREQ from a source and the reception of the corresponding RREP from the destination. We observe that the impact of packet hiding on the route discovery delay is minimal compared to the case where no packet hiding is employed. This similarity in performance is due to the expanding ring search technique of AODV, which is used to prevent unnecessary network-wide dissemination of RREQs [19]. In order to discover a route, the originating node sends a RREQ with a time-to-live (TTL) value equal to one hop, and waits for the corresponding RREP. If the RREP is not received before a timeout value ( $t_o$ ) expires, the originating node increases the TTL and the timeout  $t_o$ , and re-broadcasts the RREQ. This process is repeated until a valid RREP is received, or the TTL value exceeds the maximum diameter of the network. The expanding ring search technique introduces a dominant delay in comparison to the delay introduced by the packet-hiding techniques. For example, in the case of

time-lock CPHS, the per-packet delay overhead is  $t_p = 448$   $\mu$ sec. Using the default values specified by RFC 3561 [19], the value of the first timeout is  $t_o = 240$  msec, which is 535 times higher than  $t_p$ , making the total delay introduced by time-lock CPHS insignificant.

In the third set of experiments, we evaluated the performance of TCP in a congested ad-hoc network. We considered the same network topology used in the second set of experiments. Twenty source/destination pairs simultaneously exchanged 2 MB files using TCP. In Fig. 8(d), we show the effective throughput averaged over all 20 TCP connections. We observe that efficient packet-hiding techniques such as SHCS, and AONT-HS have a relatively small impact on the overall throughput. This is because in a congested network, the performance is primarily dependent on the queueing delays at the relay nodes. The communication overhead introduced by the transmission of the packet-hiding parameters is small and hence, does not significantly impact the throughput. On the other hand, for CPHS, we observe a performance reduction of 25% – 30% compared to the case of no packet-hiding. This reduction is attributed to the delay introduced by CPHS for the reception of each packet. Note that in the congested network scenario, the throughput reduction of CPHS is smaller compared to the non-congested one because nodes can take advantage of the queueing delays to solve puzzles.

## 9 RELATED WORK

Jamming attacks on voice communications have been launched since the 1940s [25]. In the context of digital communications, the jamming problem has been addressed under various threat models. We present a classification based on the selective nature of the adversary.

### 9.1 Prior work on Selective Jamming

In [33], Thunte studied the impact of an external selective jammer who targets various control packets at the MAC layer. To perform packet classification, the adversary exploits inter-packet timing information to infer eminent packet transmissions. In [11], Law et al. proposed the estimation of the probability distribution of inter-packet transmission times for different packet types based on network traffic analysis. Future transmissions at various layers were predicted using estimated timing information. Using their model, the authors proposed selective jamming strategies for well known sensor network MAC protocols.

In [1], Brown et al. illustrated the feasibility of selective jamming based on protocol semantics. They considered several packet identifiers for encrypted packets such as packet size, precise timing information of different protocols, and physical signal sensing. To prevent selectivity, the unification of packet characteristics such as the minimum length and inter-packet timing was proposed. Similar packet classification techniques were investigated in [4].

Liu et al. considered a smart jammer that takes into account protocol specifics to optimize its jamming strategy

[14]. The adversary was assumed to target control messages at different layers of the network stack. To mitigate smart jamming, the authors proposed the SPREAD system, which is based on the idea of stochastic selection between a collection of parallel protocols at each layer. The uncertainty introduced by this stochastic selection, mitigated the selective ability of the jammer. Greenstein et al. presented a 802.11-like wireless protocol called Slyfi that prevents the classification of packets by external observers. This protocol hides all explicit identifiers from the transmitted packets (e.g. MAC layer header and payload), by encrypting them with keys only known to the intended receivers [8].

Selective jamming attacks have been experimentally implemented using software-defined radio engines [32], [34]. Wilhelm et al. implemented a USRP2-based jamming platform called RFReact that enables selective and reactive jamming [34]. RFReact was shown to be agnostic to technology standards and readily adaptable to any desired jamming strategy. The success rate of a selective jamming attack against a 802.15.4 network was measured to be 99.96%. Blapa et al. studied selective jamming attacks against the rate-adaptation mechanism of 802.11 [32]. They showed that a selective jammer targeting specific packets in a point-to-point 802.11 communication was able to reduce the rate of the communication to the minimum value of 1 Mbps, with relatively little effort (jamming of 5-8 packets per second). The results were experimentally verified using the USRP2/GNU radio platform.

Several researchers have suggested channel-selective jamming attacks, in which the jammer targets the broadcast control channel. It was shown that such attacks reduce the required power for performing a DoS attack by several orders of magnitude [3]. To protect control-channel traffic, the replication of control transmission in multiple channels was suggested in [3], [30], [31]. The “locations” of the control channels where cryptographically protected. In [12], Lazos et al. proposed a randomized frequency hopping algorithm to protect the control channel from inside jammers. Strasser et al. proposed a frequency hopping anti-jamming technique that does not require the existence of a secret hopping sequence, shared between the communicating parties [28].

### 9.2 Non-Selective Jamming Attacks

Conventional methods for mitigating jamming employ some form of SS communications [5], [25]. The transmitted signal is spread to a larger bandwidth following a PN sequence. Without the knowledge of this sequence, a large amount of energy (typically 20-30 dB gain) is required to interfere with an ongoing transmission. However, in the case of broadcast communications, compromise of commonly shared PN codes neutralizes the advantages of SS.

Pöpper et al. proposed a jamming-resistant communication model for pairwise communications that does not rely on shared secrets. Communicating nodes use a physical layer modulation method called Uncoordinated Direct-Sequence Spread Spectrum (UDSSS) [20]. They also proposed, a jamming-resistant broadcast method in which



transmissions are spread according to PN codes randomly selected from a public codebook [20]. Several other schemes eliminate overall the need for secret PN codes [15], [29].

Lin et al. showed that jamming 13% of a packet is sufficient to overcome the ECC capabilities of the receiver [13]. Xu et al. categorized jammers into four models: (a) a constant jammer, (b) a deceptive jammer that broadcasts fabricated messages, (c) a random jammer, and (d) a reactive jammer that jams only if activity is sensed [37]. They further studied the problem of detecting the presence of jammers by measuring performance metrics such as packet delivery ratio [35]–[37]. Cagalj et al. proposed wormhole-based anti-jamming techniques for wireless sensor networks (WSNs) [2]. Using a wormhole link, sensors within the jammed region establish communications with outside nodes, and notify them regarding ongoing jamming attacks.

## 10 CONCLUSION

We addressed the problem of selective jamming attacks in wireless networks. We considered an internal adversary model in which the jammer is part of the network under attack, thus being aware of the protocol specifications and shared network secrets. We showed that the jammer can classify transmitted packets in real time by decoding the first few symbols of an ongoing transmission. We evaluated the impact of selective jamming attacks on network protocols such as TCP and routing. Our findings show that a selective jammer can significantly impact performance with very low effort. We developed three schemes that transform a selective jammer to a random one by preventing real-time packet classification. Our schemes combine cryptographic primitives such as commitment schemes, cryptographic puzzles, and all-or-nothing transformations (AONTs) with physical layer characteristics. We analyzed the security of our schemes and quantified their computational and communication overhead.

## REFERENCES

- [1] T. X. Brown, J. E. James, and A. Sethi. Jamming and sensing of encrypted wireless ad hoc networks. In *Proceedings of MobiHoc*, pages 120–130, 2006.
- [2] 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.
- [3] A. Chan, X. Liu, G. Noubir, and B. Thapa. Control channel jamming: Resilience and identification of traitors. In *Proceedings of ISIT*, 2007.
- [4] T. Dempsey, G. Sahin, Y. Morton, and C. Hopper. Intelligent sensing and classification in ad hoc networks: a case study. *Aerospace and Electronic Systems Magazine, IEEE*, 24(8):23–30, August 2009.
- [5] Y. Desmedt. Broadcast anti-jamming systems. *Computer Networks*, 35(2-3):223–236, February 2001.
- [6] K. Gaj and P. Chodowiec. FPGA and ASIC implementations of AES. *Cryptographic Engineering*, pages 235–294, 2009.
- [7] O. Goldreich. *Foundations of cryptography: Basic applications*. Cambridge University Press, 2004.
- [8] B. Greenstein, D. McCoy, J. Pang, T. Kohno, S. Seshan, and D. Wetherall. Improving wireless privacy with an identifier-free link layer protocol. In *Proceedings of MobiSys*, 2008.
- [9] IEEE. IEEE 802.11 standard. <http://standards.ieee.org/getieee802/download/802.11-2007.pdf>, 2007.
- [10] A. Juels and J. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Proceedings of NDSS*, pages 151–165, 1999.
- [11] Y. W. Law, M. Palaniswami, L. V. Hoesel, J. Doumen, P. Hartel, and P. Havinga. Energy-efficient link-layer jamming attacks against WSN MAC protocols. *ACM Transactions on Sensors Networks*, 5(1):1–38, 2009.
- [12] 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*, pages 169–180, 2009.
- [13] G. Lin and G. Noubir. On link layer denial of service in data wireless LANs. *Wireless Communications and Mobile Computing*, 5(3):273–284, May 2004.
- [14] X. Liu, G. Noubir, and R. Sundaram. Spread: Foiling smart jammers using multi-layer agility. In *Proceedings of INFOCOM*, pages 2536–2540, 2007.
- [15] Y. Liu, P. Ning, H. Dai, and A. Liu. Randomized differential DSSS: Jamming-resistant wireless broadcast communication. In *Proceedings of INFOCOM*, San Diego, 2010.
- [16] R. C. Merkle. Secure communications over insecure channels. *Communications of the ACM*, 21(4):294–299, 1978.
- [17] G. Noubir and G. Lin. Low-power DoS attacks in data wireless lans and countermeasures. *Mobile Computing and Communications Review*, 7(3):29–30, 2003.
- [18] OPNET. OPNET<sup>tm</sup> modeler 14.5. <http://www.opnet.com/>.
- [19] C. Perkins, E. Belding-Royer, and S. Das. RFC 3561: Ad hoc on-demand distance vector (AODV) routing. *Internet RFCs*, 2003.
- [20] C. Pöpper, M. Strasser, and S. Čapkun. Jamming-resistant broadcast communication without shared keys. In *Proceedings of the USENIX Security Symposium*, 2009.
- [21] R. Rivest. All-or-nothing encryption and the package transform. *Lecture Notes in Computer Science*, pages 210–218, 1997.
- [22] R. Rivest, A. Shamir, and D. Wagner. Time-lock puzzles and timed-release crypto. *Massachusetts Institute of Technology*, 1996.
- [23] B. Schneier. *Applied cryptography: protocols, algorithms, and source code in C*. John Wiley & Sons, 2007.
- [24] SciEngines. Break DES in less than a single day. <http://www.sciengines.com>, 2010.
- [25] M. K. Simon, J. K. Omura, R. A. Scholtz, and B. K. Levitt. *Spread Spectrum Communications Handbook*. McGraw-Hill, 2001.
- [26] D. Stinson. Something about all or nothing (transforms). *Designs, Codes and Cryptography*, 22(2):133–138, 2001.
- [27] D. Stinson. *Cryptography: theory and practice*. CRC press, 2006.
- [28] M. Strasser, C. Pöpper, and S. Čapkun. Efficient uncoordinated fhss anti-jamming communication. In *Proceedings of MobiHoc*, pages 207–218, 2009.
- [29] M. Strasser, C. Pöpper, S. Čapkun, and M. Cagalj. Jamming-resistant key establishment using uncoordinated frequency hopping. In *Proceedings of IEEE Symposium on Security and Privacy*, 2008.
- [30] P. Tague, M. Li, and R. Poovendran. Probabilistic mitigation of control channel jamming via random key distribution. In *Proceedings of PIMRC*, 2007.
- [31] 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.
- [32] B. Thapa, G. Noubir, R. Rajaramanand, and B. Sheng. On the robustness of IEEE802.11 rate adaptation algorithms against smart jamming. In *Proceedings of WiSec*, 2011.
- [33] D. Thuento and M. Acharya. Intelligent jamming in wireless networks with applications to 802.11 b and other networks. In *Proceedings of the IEEE Military Communications Conference MILCOM*, 2006.
- [34] M. Wilhelm, I. Martinovic, J. Schmitt, and V. Lenders. Reactive jamming in wireless networks: How realistic is the threat? In *Proceedings of WiSec*, 2011.
- [35] W. Xu, W. Trappe, and Y. Zhang. Anti-jamming timing channels for wireless networks. In *Proceedings of WiSec*, pages 203–213, 2008.
- [36] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proceedings of MobiHoc*, pages 46–57, 2005.
- [37] W. Xu, T. Wood, W. Trappe, and Y. Zhang. Channel surfing and spatial retreats: defenses against wireless denial of service. In *Proceedings of the 3rd ACM workshop on Wireless security*, pages 80–89, 2004.