Combining Cryptographic Primitive to Prevent Selective Jamming Attacks

Omana Bangaru R

University of MVJCE Department of Computer Science. Bangalore, INDIA

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 launch pad 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, the problem of selective jamming attacks in wireless networks is addressed. In these attacks, the adversary is active only for a short period of time, selectively targeting messages of high importance. 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 are illustrated. The selective jamming attacks can be launched by performing real-time packet classification at the physical layer is shown. To mitigate these attacks, the three schemes that prevent real-time packet classification by combining cryptographic primitives with physical-layer attributes are developed. The security of the methods and evaluating the computational and communication overhead is analyzed.

Keywords—*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. In the simplest form of jamming, the adversary interferes with the reception of messages by transmitting a continuous jamming signal, or several short jamming pulses. 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. 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. Conventional anti-jamming techniques rely extensively on spread-spectrum (SS) communications, or some form of jamming evasion (e.g., slow frequency hopping, or spatial retreats). 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 un- der 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, or by decoding packets on the fly. 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. 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

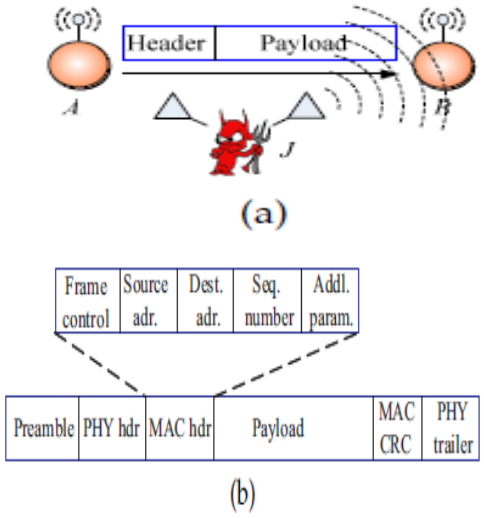


Fig.1. (a) Realization of a selective jamming attack, (b) a generic frame format for a wireless network.

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 crypto- graphic 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.

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 pair wise keys or asymmetric cryptography. Communication Model–Packets are transmitted at a rate of 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 α β q data bits, where α/β is the rate of the PHY-layer encoder. Here, the transmission bit rate is equal to qR bps and the information bit rate is α β qR 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 ad- dresses, 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, the adversary can pro-actively jam a number of bits just below the ECC capability early in the transmission are assumed. 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. 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 crypto- graphic problems is assumed to be time-consuming. For the purposes of analysis, given a cipher text, 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 Classifications

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 rst errors. For simplicity, we consider a block interleave that is denned by a matrix Ad×β1. The de-interleave 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·β bits for de-interleaving. The d·β 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 ⌈dβ q ⌉ 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 β of the code words.

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 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

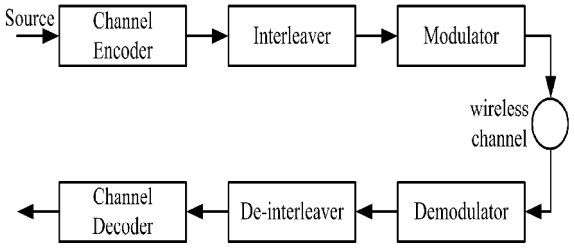


Fig. 2. A generic communication system diagram.

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, a α/β-block code may protect m from up to e errors per block. Alternatively, a α/β-rate convolutional encoder with a constraint length of Lmax, and a free distance of e bits provides similar protection. For our purposes, we assume that the rate of the encoder is α/β. At the next block, interleaving is applied to protect m from bu

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 cipher text 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 m1,m2,...mx, and each cipher text block ci, is generated as:

$c1 = IV, ci+1 = Ek(ci \oplus mi), i = 1,2,...,x, (1)$

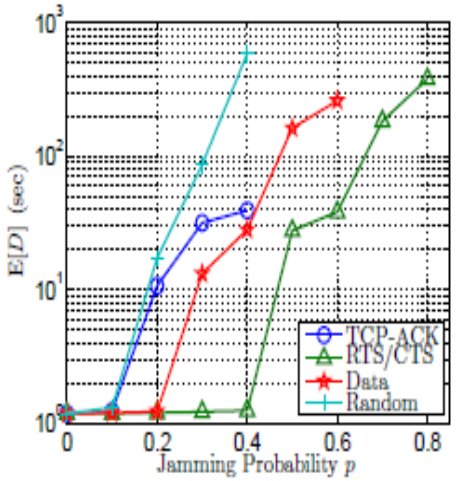
where Ek(m) denotes the encryption of m with key k. The plaintext mi is recovered by:
$mi = ci \oplus Dk(ci+1), i = 1,2,...,x. (2)$

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 ci+1 is sufficient to recover mi if k is known (c1 = 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.

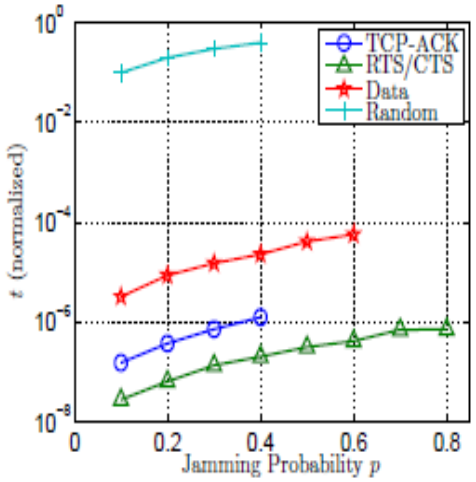
However, such a solution is not useful if the compromised node obtains the new key. This can only be avoided if there is a mechanism by which the set of compromised nodes can be identified. Such a task is non-trivial when the leaked key is shared by multiple nodes. Any node that possesses the shared 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 cipher text pre- fix. Hence, an adversary who is aware of the underlying protocol specifics (structure of the frame) can use the static cipher text portions of a transmitted packet to classify it.

4. Impact of Selective Jamming

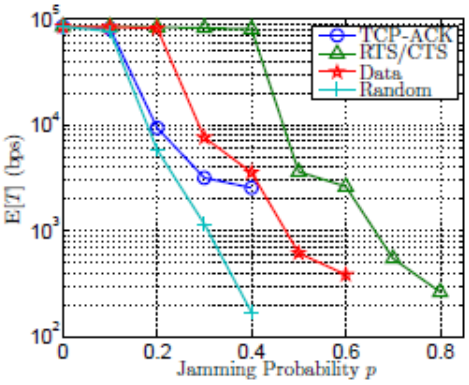
In this section; we illustrate the impact of selective jamming attacks on the network performance. We used OPNETTM Modeller 14.5 [18] to implement selective jamming attacks in two multi-hop wireless network scenarios. In the first scenario, the attacker targeted a TCP connection established over a multi-hop wireless route. In the second scenario, the jammer targeted network-layer control messages transmit- ted 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



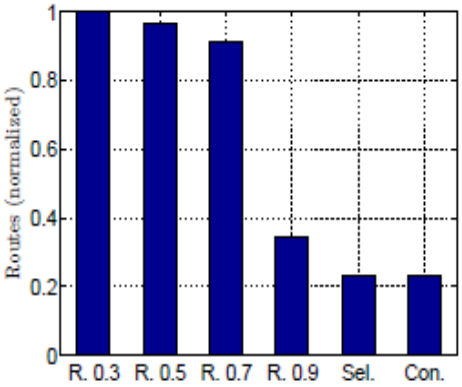
(a)



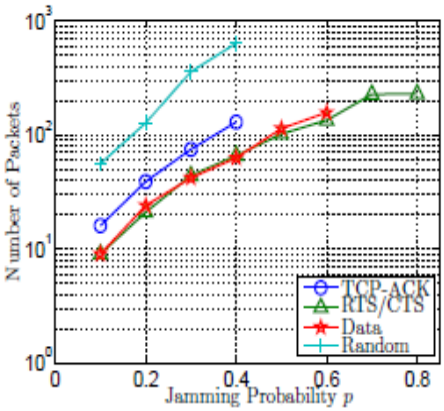
(d)



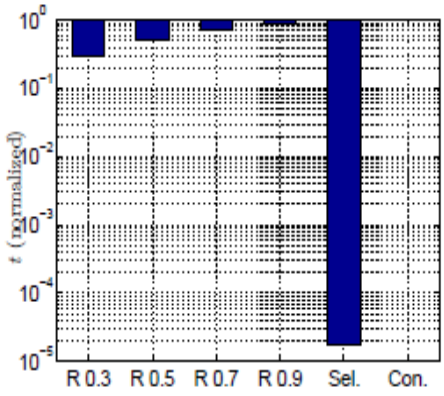
(b)



(e)



(c)



(f)

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 ale 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 slowdown 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. 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 fewer packets than random jamming. This is because, as the packet transmission rate of the sender drops fewer packets needed 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 {X,M,E}. Set X = {A,V } denotes two probabilistic polynomial-time interactive parties, where A is known as the committer and V as the verifier; set M denotes the message space, and set E = {(ti,fi)} denotes the events occurring at protocol stages ti (i = 1,2), as per functions fi (i = 1,2). During commitment stage t1, A uses a commitment function f1 = commit () to generate a pair (Cod) = commit (m), where (Cod) is called the commitment/recommitment pair. At the end of stage t1, A releases the commitment C to V. In the open stage t2, A releases the opening value d. Upon reception of d, V opens the commitment C, by applying function f2 = open (), thus obtaining a value of m′ = open (Cod). This stage culminates in either acceptance (m′ = m) or rejection (m′ 6= 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) polynomials- efficient algorithm that would allow A to generate a triple (C,d,d′), such that V accepts the commitments (Cod) and (Cod′), 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/recommitment pair (Cod), and broadcasts C. The hiding property ensures that m is not revealed during the transmission of C. To reveal m, the sender releases the recommitment 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, dpart) and (C′, d′part), there is no (probabilistic) polynomials-efficient algorithm that would allow V associate C with m and C′ with m′, with non-negligible probability. Here, dpart 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 dpart 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 (Cod) = commit (m), where,
C = Ek (π1 (m)), d = k. Here, the commitment function Ek () is an off-the-shelf symmetric encryption algorithm (e.g., DES or AES [27]), π1 is a publicly known permutation, and k ∈ {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 = π−1 1 (Dk(C)), where π−1 1 denotes the inverse permutation of π1. To satisfy the strong hiding property, the packet carrying d is format- ted 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 sub layer called the "hiding sub layer" is inserted between the MAC and the PHY layer. This sub layer is responsible for formatting m before it is processed by the PHY layer. The functions of the hiding sub layer are outlined in Fig. 4. Consider a frame m at the MAC layer delivered to the hiding sub layer. 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 π1. The purpose of π1 is to randomize the

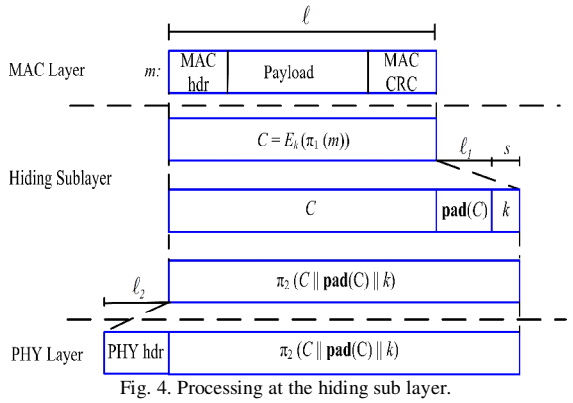


Fig. 4. Processing at the hiding sub layer.

Input to the encryption algorithm and delay the reception of critical packet identifiers such as headers. After the permutation, π1 (m) is encrypted using a random key k to produce the commitment value C = Ek (π1 (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 cipher text), in Section 5.4 we show that both are necessary to achieve packet hiding. In the next step, a padding function pad () appends pad(C) bits to C, making it a multiple of the symbol size. Finally, C‖pad(C) ‖k is permuted by applying a publicly known permutation π2. The purpose of π2 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 ℓ1 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 ℓ as the original message m. Let also ℓ2 denote the length of the header added at the PHY layer the frame carrying (Cod) before the encoder has a length of (ℓ + ℓ1 + ℓ2+s) bits.

Assuming that the rate of the encoder is α/β the output of the encoder will be of length, β α (ℓ + ℓ1 + ℓ2 + s). For the last symbol of transmission to include α β q bit of the key k, it must hold that,
ℓ1 =α β q − (ℓ + ℓ2) β αmod q). (3) Permutation–The hiding layer applies two publicly known permutations π1 and π2 at different processing stages. Permutation π1 is applied to m before it is encrypted. The purpose of π1 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 cipher text blocks must be received and decrypted. Moreover, header information is pushed at the end of π1 (m). This prevents early reception of the corresponding cipher text blocks. For example, consider the transmission of a MAC frame of length 2,336 bytes which carries a TCP data packet.

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 π1 (m) is partitioned to 146 plaintext blocks {p1, p2,..., p146}, and is encrypted to produce 146 cipher text blocks C = c1‖c2‖...‖c146. 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.

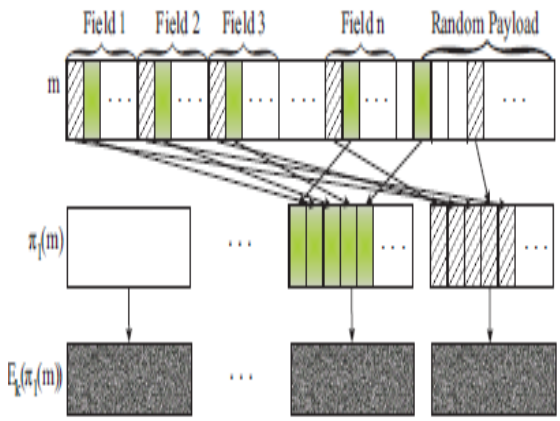


Fig 5 application of permutation

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 ℓi bits long, the last ℓi cipher text blocks must be received and decrypted. If ℓi > ℓb, where ℓb denotes the cipher text block length, the bit placement process continues in a round- robin fashion. The second goal of the permutation π1 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 cipher text block. Permutation π2 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 × β block interleave. Without loss of generality, assume that β = q, and let the last n rows of the last block passed via the interleave correspond to the encoded version of the random key k. Permutation π2 rearranges the bits of k at the interleave matrix Ad×β 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 interleave 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 π2 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 cipher text-only attack on C as early as the reception of the rest cipher text block. Because the encryption key is refreshed at every transmission, a very small number of cipher text 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/cipher text 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 π1 distributes frame fields to cipher text blocks in the reverse order of transmission, with the MSBs from each field appearing on the last cipher text block. Hence, reception of all blocks of C is required for the complete recovery of headers. To minimize the communication over- head, 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 cipher texts. If the encryption of all possible messages with all possible keys results in unique cipher texts, there is a 1-1 correspondence between a cipher text and the generating plaintext/key pair. This property is guaranteed with high probability when the plaintext space M and the key space K are much smaller than the cipher text space C. Assuming the encryption of a plaintext block mi with a key ki randomly maps to a cipher text ci = Eki(mi), every cipher text ci ∈ C occurs with probability pc = 1 |C|. The problem of finding the probability that all |M||K| cipher texts produced by the encryption of all plaintexts with all keys are unique can be formulated as a "birthday problem":
Pr [cipher texts unique] ≈ e
−|M|·|K| (|M|·|K|−1) 2|C|.

As an example, consider the encryption of a message m = {m1, m2, mx} with a key k of length 56 bits, using blocks of 128 bits. For a fairly small plaintext space (e.g., |M| = 16), the probability of cipher text uniqueness is equal to 99.8%. Thus, the adversary can recover k, by launching a codebook attack on m1. The remaining ci's are decrypted in real-time, using the known value of k. Here, the plaintext space for m1 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 cipher text space.

Partial release of d–Depending on the PHY layer implementation, d = k requires n ≥ 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 trans- mission 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 22 α β q keys. The adversary must be capable of performing on average N = 2(2α β q−1) R decryptions per second in order to find k before the last symbol is transmitted2. 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 N ≈ 3.52×1019 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 × 1011 keys per second.

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 ≤ β 2α log2 N + 1. For instance, when N = 2.92 × 1011, the adversary can find k if q ≤ 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′ 6= 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′ 6= k that opens a valid value of m′ 6= 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

2. A more accurate calculation of N would assume an adversary trying a brute force attack on k, with the reception of the first cipher text block, and adjusting the searching space according to the partial release of k.
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 (Cod) is,
C = (γ, δ) = (Ek(m||r), r), d = k.
Provided that r is sufficiently long, a computationally bounded jammer cannot find a k′ such that Dk′(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, (ℓb − (ℓ mod ℓb)) bits of overhead are added by the encryption algorithm, to convert a plaintext of length ℓ to a multiple of the encryption block. Thus, the communication overhead of SHCS is equal to s + (ℓb − (ℓ mod ℓb)), per packet. Here, we do not account for the padding string pad(C), because the addition of 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 destine- tin 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.

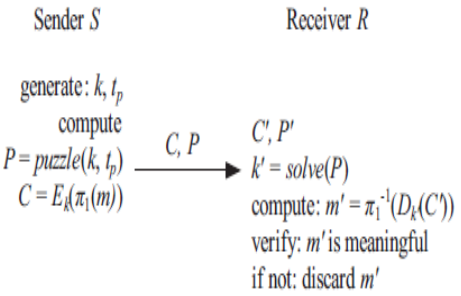


Fig. 6. The cryptographic puzzle-based hiding scheme.

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

In our context, we use cryptographic puzzles to temporary 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 has a packet m for transmission. The sender selects a random key k ∈ {0, 1} s, of a desired length. S generates a puzzle P = puzzle (k, tp), where puzzle () denotes the puzzle generator function, and tp denotes the time required for the solution of the puzzle. Parameter tp is measured in units of time, and it is directly dependent on the assumed computational capability of the adversary, denoted by N and measured in computational operations per second. After generating the puzzle P, the sender broadcasts (C, P), where C = Ek (π1 (m)). At the receiver side, any receiver R solves the received puzzle P′ to recover key k′ and then computes m′ = π−1 (Dk′ (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 and 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. Time-lock puzzles have several attractive features such as the fine granularity in controlling tp 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 · 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 Kh = k + a2t mod g, where t = tp · N, is the amount of time required to solve for k. Here, it is assumed that the solver can perform N squaring modulo g per second. Note that Kh can be computed efficiently if φ (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, Kh, 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 (Kh, t, a). Note that if any of t, an are unknown, any value of k is possible. 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 = k1||k2. The lengths of k1 and K2 are s1, and s2, respectively. He then computes C = Ek (π1 (m)) and transmits (C, k1, h (k)) in this particular order. To obtain k, any receiver has to perform on average 2s2−1 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 crypt analyzing cipher text C = Ek (π1 (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 cipher text-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 tp, 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 is 22 α β q. assuming a brute force attack on the missing bits of the puzzle, the computational load of the adversary increases on average to 22α β q−1tp.

The value of tp 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 tp. 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 Kh, a, and t. The value Kh 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 Kh, and depends on the computational capability of the adversary. The security of time locks depends on the difficulty in factoring g or finding φ (g), where φ () denotes the Euler φ–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 k1 which is of length s1 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 π1 (m), and apply the inverse permutation. In the case of hash-based puzzles, the modulo squaring operation is substituted by, on average, 2s2−1 hashing operations.

7. Hiding Based On All-Or-Nothing Trans

FORMATIONS 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 trans-formation f, mapping message m = {m1,⋯ ,mx} to a sequence of pseudo-messages m′ = {m′ 1,⋯ ,m′x′}, is an AONT if [21]: (a) f is a bisection, (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 cipher text 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 cipher text 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 be- fore transmission but remain unencrypted. The jammer can- not performs 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 = {m1,..., mx}, which serve as an input to an **AONT f : {Fu} x → {Fu} x′.Here**,
Fu denotes the alphabet of blocks mi and x′ denotes the number of output pseudo-messages with x′ ≥ x. The set of pseudo-messages m′ = {m′ 1, 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, Stinson showed how to construct a linear AONT when the alphabet of the input blocks is a finite field Fu, with the order u being a prime power. He showed that if an invertible matrix M = **{mij|mij ∈Fu, mij 6= 0} x×x** exists, then the transformation f (m) = mM−1 is a linear AONT. He also provided a method for constructing such M This is as follows.

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

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

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

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

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

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

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 = mi ⊕ Ek′(i), for i = 1,2,...,x (8) m′x+1 = k′ ⊕ e1 ⊕ e2 ⊕ ⋯ ⊕ ex, (9) where ei = Ek0(m′ i ⊕i), for i = 1,2,...,x, and k0 is a fixed publicly-known encryption key. With the reception of all pseudo-messages message m is recovered as follows: k′ = m′x+1 ⊕ e1 ⊕ e2 ⊕ ⋯ ⊕ ex, (10) mi = m′ i ⊕ Ek′(i), for i = 1,2,...,x. (11) Note that if any m′ i is unknown, any value of k′ is possible, because the corresponding ei is not known.

Hence, Ek′(i) cannot be recovered for any i, making it infeasible to obtain any of the mi. Hiding Sub layer Details–AONT-HS is implemented at the hiding sub layer residing between the MAC and the PHY layers. In the first step, m is padded by applying function 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||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||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 ≤ i ≤ 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 mix 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 22 α β q. The adversary must be capable of solving on average 22α β q−1 systems of linear equations in time equal to the length of one symbol (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 α/β = 1/2 (802.11a), the search space is equal to 1.4 × 1014. As in the case of SHCS, when the value of q becomes small (q ≤ β 2α log2 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′ ≥ x. Additionally, the function pad() appends (ℓb − (ℓ mod ℓb)) bits in order to make the length of m a multiple of the length ℓb of the pseudo-messages m′. Hence, the communication overhead introduced is (ℓb (x′–x) +ℓb– (ℓ mod ℓb)) 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 sym- metric 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, x are encrypted). Therefore, only one cipher text block is produced per pseudo-message. Assuming a pseudo-message block size equal to the cipher text block size ℓb, 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 ℓ + ℓb.

8. Conclusion

The problem of selective jamming attacks in wireless networks is addressed. An internal adversary model is considered in which the jammer is part of the network under attack, thus being aware of the protocol specifications and shared network secrets. The jammer can classify transmitted packets in real time by decoding the first few symbols of an ongoing transmission is

shown. The impact of selective jamming attacks on network protocols such as TCP and routing is evaluated. Our findings show that selective jammers can significantly impact performance with very low effort. Three schemes that transform a selective jammer to a random one by preventing real-time packet classification are developed. Our schemes combine cryptographic primitives such as commitment schemes, cryptographic puzzles, and all-or-nothing transformations (AONTs) with physical layer characteristics. The security of our schemes and quantified their computational and communication overhead is analyzed.

References

[1] Energy-Efficient Link-Layer Jamming Attacks Against Wireless Sensor Network MAC Protocols march-2006

[2] Wormhole-Based and Anti-Jamming Techniques in Sensor Networks- Sep 2008

[3] Mitigating Control-Channel Jamming Attacks in Multi-channel Ad Hoc Networks march-2009

[4] Jamming and Sensing of Encrypted Wireless Ad Hoc Networks'. 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.

[5] 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.