International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Selective Jamming Attacks Prevented By Packet-Hiding Methods

Sontam Sunil Kumar Reddy[1], K. Raghavendra Rao[2]

[1]M. Tech student, Department of CSE, Anurag Group of Institutions, Hyderabad, India

[2]Assistant Professor, Department of CSE, Anurag Group of Institutions, Hyderabad, India

Abstract: ***The open nature of the wireless medium leaves it vulnerable to intentional interference attacks, typically referred to as jamming. This intended interference with wireless transmissions can be used as a launchpad for increasing Denial-of-Service attacks on wireless networks. in general, jamming has been addressed under an external threat model. Though, 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, we address the problem of selective jamming attacks in wireless networks and in these attacks, the adversary is active only for a less period of time, selectively targeting messages which are highly 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. In this we show that selective jamming attacks can be launched by performing real-time packet classification at the physical layer. To moderate these attacks, we develop three schemes that prevent real-time packet classification by combining cryptographic primitives with physical-layer attributes. Analyzing the security of our methods and evaluate their computational and communication overhead.***

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. Still, the open nature of this medium leaves it helpless to multiple security threats and anyone with a transceiver can listen in on wireless transmissions, inject false messages, or jam genuine ones. While eavesdropping and message injection can be prevented using cryptographic methods, where jamming attacks are much harder to counter. They have been shown to make reality of severe Denial-of-Service. Typically, jamming attacks have been considered under an external threat model in which jammer is not part of the network. In this model, jamming strategies include the continuous or random transmission of high-power interference signals [9], [16]. However, adopting an "always-on" strategy has several disadvantages. First, the opponent has to expend a significant amount of energy to jam frequency bands of interest. Second, the nonstop presence of unusually high interference levels makes this type of attacks easy to detect.

Conventional anti-jamming techniques rely on spread-spectrum communications.[9], 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 defend wireless transmissions under the external threat model. Possible disclosure of secrets due to node compromise neutralizes the gains of SS. The 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.

To launch selective jamming attacks, the opponent must be capable of implementing a "classify-then-jam" strategy before the completion of a wireless transmission. Such plan can be actualized either by classifying transmitted packets using protocol semantics [1], [13], or by decoding packets on the fly In the latter method, the jammer may translate the first few bits of a packet for recovering useful packet identifiers such as packet type, source and the destination address. After classification, the opponent 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.

2. Literature Survey

Set M. Strasser, C. Popper, and S. Capkun [12] proposed a Uncoordinated Frequency Hopping (UFH), a new spread-spectrum anti-jamming technique that does not rely on secret keys A major limitation of common anti-jamming techniques (such as FHSS and DSSS) is their dependency on a secret shared by the sender and the receiver; the secret is required to coordinate the used frequency channels or code sequences and must not be known to an attacker. This dependency precludes the application of common anti-jamming techniques in scenarios where the parties cannot resort to shared secrets; these include anti-jamming key establishment and anti-jamming broadcast to a (partially) unknown or un-trusted group of receivers. Uncoordinated Frequency Hopping (UFH), a new spread-spectrum antijamming technique that does not rely on shared secret keys. With UFH, two communicating nodes hop among a set of known frequency channels in an uncoordinated and random manner.

M. Cagalj, S. Capkun, and J.P. Hubaux [2] proposed how the sensor nodes can exploit channel diversity in order to create wormholes that lead out of the jammed region, during which an alarm can be transmitted to the network operator.

Licensed Under Creative Commons Attribution CC BY

we investigate an attack where the attacker masks the event (event masking) that the sensor network should detect by stealthily jamming an appropriate subset of the nodes. In this way, the attacker prevents the nodes from reporting what they are sensing to the network operator. Timely detection of such stealth attacks is particularly important in scenarios in which sensors use reactive schemes to communicate events to the network sink. Event masking attacks result in a coverage paradox: Even if an event is sensed by one or several nodes (and the sensor network is otherwise fully connected), the network operator cannot be up to date on time about the event.

Guolong Lin Guevara Noubir [6] investigated and analyzed the performance of combining a cryptographic interleaver with various coding schemes to improve the robustness of wireless LANs for IP packets transmission [1]. A concatenated code that is simple to decode and can maintain a low Frame Error Rate (FER) under a jamming effort ratio of 15%. We argue that LDPC codes will be very suitable to prevent this type of jamming.

Ronald L. Rivest [7] proposed a new mode of encryption for block ciphers, we this call all-or-nothing encryption. This mode has the interesting major property that one must decrypt the entire ciphertext before one can determine even one message block. This way that Brute-force searches against all-or-nothing encryption are slowed down by a factor equal to the number of blocks in the ciphertext. We give a exact way of implementing all-or-nothing encryption using a" package transform" as a pre-processing step to a ordinary encryption mode.

Wenyuan Xu, Wade Trappe, Yanyong Zhang and Timothy Wood [15] examined radio interference attacks from both sides of the issue: first, they studied the problem of conducting radio interference attacks on wireless networks, and second they examine the critical issue of diagnosing the presence of jamming attacks. Specifically, they proposed four different jamming attack models that can be used by an adversary to disable the operation of a wireless network, and estimate their effectiveness in terms of how each method affects the ability of a wireless node to send and receive packets. Then they discuss different measurements that serve as the basis for detecting a jamming attack, and discover scenarios where each measurement by itself is not enough to reliably classify the presence of a jamming attack.

Wireless communication is susceptible to radio interference, which prevents the response of communications. While evasion strategies have been proposed, such strategies are costly or unsuccessful against broadband jammers.

Wenyuan Xu, Wade Trappe and Yanyong Zhang proposed [16] an alternative to evasion strategies that involves the establishment of a timing channel that exists in spite of the existence of jamming. The timing channel is built using failed packet reception times. We show that it is possible to detect failed packet events in spite of jamming. We then explore single sender and multi-sender timing channel constructions that may be used to build a low-rate overlay link-layer. We talk about implementation issues that we have overcome in constructing such jamming resistant timing channel, and current the results of validation efforts using the MICA2 platform. To finish, we examine additional error correction and authentication mechanisms that may be used to cope with adversaries that both jam and seek to corrupt our timing channel.

3. Real-Time Packet Classification

In this section, we describe how the adversary can classify packets in real time, prior to the packet transmission is completed. Once a packet is classified, the opponent 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 changes before it is transmitted over the wireless channel. At receiver, the signal is demodulated, de-interleaved, and decoded, to get better the original packet m.

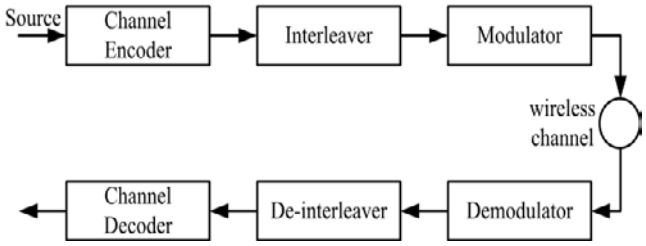

Figure 2: A generic communication system diagram

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 required redundancy for protecting m against channel errors. For ex, an α/β-block code may protect m from up to e errors per block. Instead, an α/β-rate convolutional encoder with a constraint the length of Lmax, and a free distance of ebits provides similar protection. For our purposes, we think that the rate of the encoder is α/β. At next block, interleaving is applied to protect m from burst errors. For ease, 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. At last, the digital modulator maps the received bit stream to symbols of length q, and change them into suitable waveforms for transmission over the wireless channel. Typical modulation techniques include BPSK,OFDM,16(64)-QAM, and CCK.

In order to recover any bit of m, the beneficiary must collect d.β bits for de-interleaving. Then the d.β de-interleaved bits are then passed during the decoder. Ignoring any propagation and decoding delays, the delay in anticipation of decoding the first block of data is $\lceil \frac{d\beta}{q} \rceil$ symbol durations. As an ex, in the 802.11a standard, an operating at the lowest rate of 6 Mbps, and the 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 highest data rate of 54 Mbps, 216 bits of data are improved per symbol.

From our analysis, it is evident that intercepting the first few symbols of a packet is sufficient for obtaining relevant

Licensed Under Creative Commons Attribution CC BY

header information. For ex, consider the transmission of a TCP-SYN packet used for establishing a TCP connection at the transport layer. Suppose an 802.11a PHY layer with a transmission rate of 6 Mbps. At 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 order, the output of the encoder is split into 25 blocks of 48 bits each and interleaved on a per-symbol basis. At last, each of the blocks is modulated as an OFDM symbol for transmission. The information enclosed in each of the 25 OFDM symbols is as follows:

- Symbols 1-2 contain the PHY-layer header and the first byte of MAC header. And the PHY header reveals the length of the packet, transmission rate, and synchronization information. The 1st 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 have the source and destination MAC addresses, and the length of the IP packet header.
- Symbols 11-17 contain the source and destination IP addresses and the size of the TCP datagram carried by the IP packet, and the other IP layer information. The first 2 bytes of the TCP datagram reveal the source port.
- Symbols 18-23 hold 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 different ways. MAC layer classification is achieved by receiving the first 10 symbols. IP layer categorization is achieved by receiving symbols 10 and 11, as TCP layer classification is achieved by symbols 12-19. Perceptive solution to selective jamming would be the encryption of transmitted packets (including headers) with a static key. Though, for broadcast communications, this standing decryption key must be known to all intended receivers and therefore, 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 ex, consider the cipher-block chaining (CBC) mode of encryption. 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 ciphertext block c_i, is generated as:

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

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

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

Note from (2) that reception of ci+1 is sufficient to recover mi if k is known (c1 = IV is also known). So, 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. Also, such a solution is not useful if the compromised node obtains the new key. This is avoided if there is a mechanism by which the set of compromised nodes can be identified. Such a job is non-trivial when the leaked key is shared by multiple nodes. Any node that have the shared key is a candidate malicious node. Also, even if the encryption key of a hiding scheme were to remain secret, the fixed portions of a transmitted packet could potentially lead to packet classification. This is because for computationally-capable encryption methods such as block encryption and the encryption of a prefix plaintext with the same key yields a static ciphertext prefix. Therefore, 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 Modeler 14.5 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.

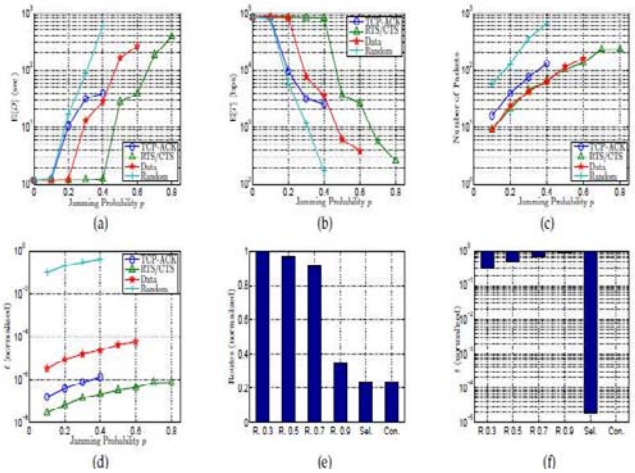

Figure 3: (a) Average application delay E[D], (b) average effective throughput E[T], (c) number of packets jammed, (d) fraction of time the jammer is active, (e) number of connections established in the network, (f) fraction of time the jammer is active. R p: random jammer with probability p; Con.: constant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

5. Hiding Based on Commitments

In this section, we explain 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 assign to a value m, to an entity V while keeping m hidden. assurance schemes are formally defined as follows.

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, anywhere A is known as the committer and V as the verifier; set M denotes the message space, and

Licensed Under Creative Commons Attribution CC BY

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 (C, d) = commit(m), where (C, d) is called the commitment/de-commitment pair. At the end of stage t1, A releases the assurance C to V. In the open stage t2, A releases the opening value d. Upon reception of d, V opens the assurance C, by applying function f2 = open(), thus obtaining a value of m′ = open(C, d). This stage culminates in either acceptance (m′ = m) or rejection $(m' \neq m)$of the commitment by V. Commitment schemes assure 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 de-commitment 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 function of the verifier is assumed by any receiver R, including the jammer J. The dedicated value m is the packet that S wants to correspond to R. To transmit m, the sender computes the corresponding commitment/de-commitment pair (C, d), and broadcasts C. The hiding property ensures that m is not revealed during the broadcast of C. To reveal m, the sender releases the de-commitment value d, in which case m is obtained by all receivers, as well as J. Note that the hiding property, as defined in assurance 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 [3].

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 expose m by exposing d. Though, 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. Therefore, the jammer has the opportunity to jam d instead of C once m has been classified. To avoid this scenario, we introduce the strong hiding property:

Strong Hiding: For each polynomial-time party V interacting with A and possessing pairs (C, dpart) 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-negligble 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 major part of d becomes known to the verifier, the trivial attacks, such as brute forcing the unknown bits of d, turn into 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 inspiration 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) = commit(m), where,

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

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 $k \in \{0, 1\}^s$ 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 reaction of d, any receiver R computes

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

where π_1^{-1} denotes the inverse permutation of π_1. To satisfy the strong hiding property, the packet transport d is formatted so that all bits of d are modulated in the last few PHY layer symbols of the packet. To get better d, any receiver must receive and decode the last symbols of the transmitted packet, thus preventing early exposé 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 de-commitment value d (i.e., the decryption key k) is carried in the same packet as the dedicated value C. This saves the extra packet header needed for transmitting d individually. To get 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 defeat 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 payload, followed by the clip containing the CRC code. At first, m is permuted by applying a publicly known permutation π1. The reason of π1 is to randomize the

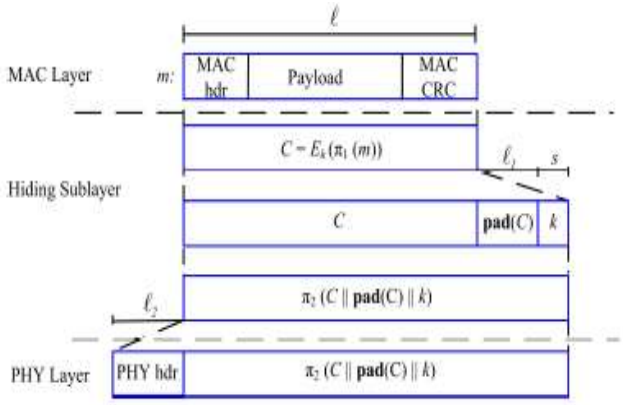

Figure 4: Processing at the hiding layer

input to the encryption algorithm and delay the reception of critical packet identifiers such as headers. Once the

Licensed Under Creative Commons Attribution CC BY

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 ciphertext), in Section 5.4 we show that both are essential to achieve packet hiding.

In the next step, a padding function pad() appends pad(C) bits to C, production of a multiple of the symbol size. Finally, C||pad(C)||k is permuted by applying a publicly known permutation π2. The reason 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 there 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 essential for minimizing the time for which parts of k become exposed. Let ℓ1 denote the number of bits padded to C. For ease, 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 (C, d) 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, $\frac{\alpha}{\beta}(\ell + \ell 1 + \ell 2 + s)$. For the last symbol of transmission to include $\frac{\alpha}{\beta}$q bits of the key k, it must embrace that,

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

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 significant frame fields which can be used for packet classification across multiple plaintext blocks. So, to reconstruct these fields, all equivalent ciphertext blocks must be received and decrypted. Also, header information is pushed at the end of π1(m). This prevents early reception of the corresponding ciphertext blocks.

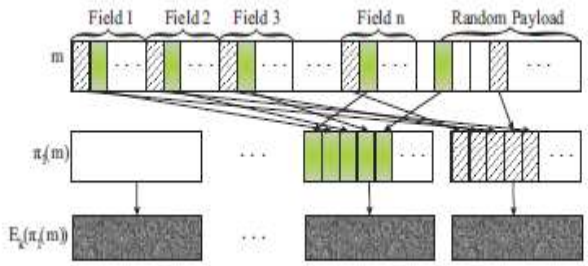

Figure 5: Application of permutation π1 on m1

For example, consider the transmission of a MAC frame of length 2,336 bytes which take 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. Suppose 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 ciphertext 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 procedure 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 ℓi bits long, the last ℓi ciphertext blocks must be received and decrypted. If ℓi>ℓb, where ℓb denotes the ciphertext block length, the bit position process continues in a round-robin fashion. The second goal of the permutation π1 is to randomize the plaintext blocks. presumptuous a chance payload, the permutation distributes the payload bits to all plaintext blocks processed by the encryption function, therefore randomizing each ciphertext block.

Permutation π2 is applied to reverse the effects of interleaving on the bits of k, so that k is contain at the packet trailer. Interleaving can be applied transversely multiple frequencies on the same symbol (e.g., in OFDM), or it may span multiple symbols. For example, consider a d×β block interleaver. with no loss of generality, assume that β = 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 π2 rearranges the bits of k at the interleaver matrix A d×β in such a way that all bits of k appear in the last n columns. So, the bits of k will be modulated as the last n symbols of the transmitted packet. Note that this action affects only the interleaver block(s) that carries k. For rest of the packet, the interleaving function is performed normally, accordingly preserving the benefits of interleaving. For the 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 Resource Overhead of SHCS

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

1) **Communication Overhead**–For each 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. therefore, the communication overhead of SHCS is equal to s + (ℓb − (ℓ mod ℓb)), per packet. now, we do not account for the padding string pad(C), since the addition of pad(C) does not increase the number of transmitted symbols.
2) **Computation Overhead**–The computation overhead of SHCS is one symmetric encryption at the sender and one symmetric decryption at the receiver. Since the header information is permuted as a trailer and encrypted, all receivers in the district of a sender must receive the entire packet and decrypt it, by the packet type and destination can be determined. Though, in wireless protocols such as 802.11, the whole packet is received at the MAC layer before it is decided if the packet must be discarded or be further processed [4]. If some parts of the MAC header are deemed not to be useful information to the jammer, they can stay unencrypted in the header of the packet, hence avoiding the decryption operation at the receiver.

Licensed Under Creative Commons Attribution CC BY

6. Hiding Based On Cryptographic Puzzles

In this section, we present a packet hiding scheme based on cryptographic puzzles. The main idea after 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 attention. The time required for obtaining the solution of a puzzle depends on its hardness and the computational ability of the solver [5]. The advantage of the puzzlebased scheme is that its security does not rely on the PHY layer parameters. Though, it has higher computation and communication overhead.

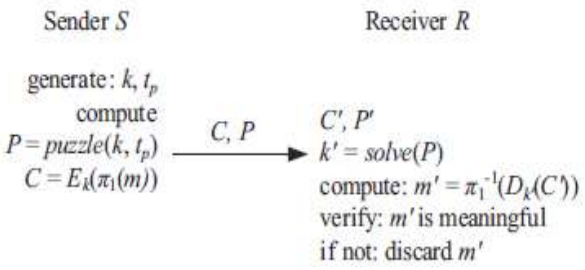

Figure 6: The cryptographic puzzle-based hiding scheme.

In our context, we use cryptographic puzzles to temporary hide transmitted packets. A packet m is encrypted with a by chance selected symmetric key k of a attractive length s. The key k is blinded using a cryptographic puzzle and sent to the receiver. For a computationally bounded opponent, 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 opponent 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 = puzzle(k,tp), where puzzle() denotes the puzzle generator function, and tp indicate the time required for the solution of the puzzle. Parameter tp is considered in units of time, and it is directly reliant on the assumed computational capability of the opponent, denoted by N and measured in computational operations per second. After creating 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 situation 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 execution details which impact security and concert. Cryptographic puzzles are primitives originally suggested by Merkle as a method for establishing a secret over an insecure channel. They find a wide range of request 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 request of a precisely controlled number of modulo operations. Time-lock puzzles have several smart features such as the fine granularity in controlling tp and the sequential nature of the computation. Moreover, the puzzle creation 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. now, it is assumed that the solver can perform N squarings modulo g per second. Note that Kh can be computed proficiently if φ(g) = (u − 1)(v − 1) or the factorization of g are known, or else 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 expose the rest of the puzzle information in the order (Kh, t, a). Note that if any of t, a are unidentified, any value of k is possible [8].

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 implementing from cryptographic puzzles which employ computationally efficient cryptographic primitives. Client puzzles proposed in [5], 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 select a random key k with k = k1||k2. The lengths of k1 and k2 are s1, and s2, correspondingly. He then computes C = Ek(π1(m)) and transmits (C, k1, h(k)) in this particular order. To get k, any receiver has to perform on average 2s2−1 hash operations (assuming perfect hash functions). since the puzzle cannot be solved before h(k) has been received, the challenger cannot classify m before the completion of m's transmission.

6.3 Resource Overhead of CPHS

Communication Overhead–The per-packet communication overhead of CPHS is equal to the length of P, in count to the padding added by the encryption function. If the puzzle is recognized using time-locks, the length of P is equivalent to the lengths of Kh, a, and t. The value Kh is calculated 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 lesser than a, g, and Kh, and depends on the computational capability of the opponent. 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. Since messages need to stay hidden for only a short period of time, the modulo can be selected to be of much smaller size and be periodically refreshed. In the container of hash-based puzzles, the communication transparency is equal to the transmission of the key k1 which is of length s1 and the hash value h(k). The distinctive length of hash function is 160 bits.

Licensed Under Creative Commons Attribution CC BY

Computation Overhead–In time-lock puzzles, the sender has to relate one permutation on m, do one symmetric encryption, and one modulo squaring process to hide k. On the receiver side, the receiver has to present t modulo squaring action to recover k, one symmetric decryption to recover π1(m), and relate the inverse permutation. In the container of hash-based puzzles, the modulo squaring operation is alternate by, on average, 2s2−1 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 calculation overhead. Such transformations were originally proposed by Rivest to slow down brute force attacks against block encryption algorithms [7]. 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 = {m1, . . . ,mx} to a sequence of pseudo-messages m′ = {m′ 1, . . . ,m′ x′}, is an AONT if [7]: (a) f is a bijection, (b) it is computationally infeasible to get any part of the original plaintext, if one of the pseudo-messages is indefinite, 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 acknowledged to obtain any part of the plaintext. So, brute force attacks are slowed down by a factor equal to the number of ciphertext blocks, without any alter on the size of the secret key. Note that the inventive AONT proposed in [7] is computationally secure. Several AONT schemes have been anticipated that extend the definition of AONT to undeniable security [10]. 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 do packet classification until all pseudo-messages corresponding to the original packet have been received and the inverse transformation has been applied. Packet m is separation 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 recipient, the inverse transformation f−1 is applied after all x′ pseudo-messages are received, in arrange to recover m.

7.2 Implementation details of the AONT-HS

In this section, we explain two AONTs which can be employed in AONT-HS; a linear transformation [10], and the original package transformation [7].

Linear AONT–In [10], 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. Clarify 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 provides a method for constructing such M which is as follows. Let u = vi, where v is prime and i is a positive integer. Choose λ ∈ Fu such that λ /∈ {n−1 (mod v), n−2 (mod v)} and define the linear AONT LT to be,

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

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:

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

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

Note from (6), (7) that if any of the {m′i} is missing, all values of mi are possible, for each i. Thus, the linear AONT provides undeniable security.

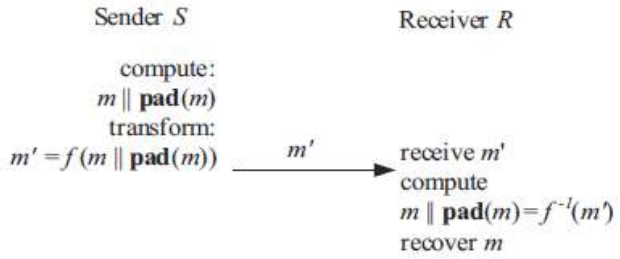

Fig.7. The AONT-Based Hiding Scheme(AONT-HS)

The Package Transform–In the package transform, given a message m, and a unsystematic key k′, the output pseudo-messages are computed as follows:

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

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

where ei = Ek0 (m′i ⊕i), for i = 1, 2, . . . , x, and k0 is a fixed publicly-known encryption key. With the response of all pseudo-messages message m is recovered as follows:

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

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

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 improved for any i, creating 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 1st 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 compound 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

Licensed Under Creative Commons Attribution CC BY

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 recipient, the inverse transformation f−1 is applied to obtain m||pad(m). The padded bits are removed and the original message m is improved. The steps of AONT-HS are shown in Fig. 7.

7.3 Resource Overhead of the AONT-HS

Communication Overhead–In AONT-HS, the inventive 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 statement overhead is introduced. For the package change, 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 mainly 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 extent of the plaintext for the x′ encryptions is relatively small compared to the length of message m (indexes 1, . . . , x are encrypted). So, only one ciphertext block is produced per pseudo-message. pretentious a pseudo-message block size equal to the ciphertext 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. An internal adversary model in which the jammer is part of the network under attack, thus being aware of the protocol stipulation and shared network secrets. The jammer can categorize transmitted packets in real time by decoding the first few symbols of an ongoing transmission. The contact of selective jamming attacks on network protocols such as TCP and routing. Our findings illustrate that a selective jammer can significantly impact performance with very low attempt. We developed three schemes that transform a selective jammer to a random one by preventing real-time packet classification. Our schemes unite cryptographic primitives such as obligation schemes, cryptographic puzzles, and all or-nothing transformations (AONTs) with physical layer characteristics. It 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 antijamming techniques in sensor networks. IEEE Transactions on Mobile Computing, 6(1):100–114, 2007.
[3] O. Goldreich. Foundations of cryptography: Basic applications. Cambridge University Press, 2004.
[4] IEEE. IEEE 802.11 standard. http://standards.ieee.org/getieee802/ download/802.11-2007.pdf, 2007.
[5] A.Juels and J.Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In Proceedings of NDSS, pages 151–165, 1999.
[6] 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.
[7] R. Rivest. All-or-nothing encryption and the package transform. Lecture Notes in Computer Science, pages 210–218, 1997.
[8] R. Rivest, A. Shamir, and D. Wagner. Time-lock puzzles and timed release crypto. Massachusetts Institute of Technology, 1996.
[9] M. K. Simon, J. K. Omura, R. A. Scholtz, and B. K. Levitt. Spread Spectrum Communications Handbook. McGraw-Hill, 2001.
[10] D. Stinson. Something about all or nothing (transforms). Designs, Codes and Cryptography, 22(2):133–138, 2001.
[11] D. Stinson. Cryptography: theory and practice. CRC press, 2006.
[12] M. Strasser, C. Popper, S. Capkun, and M. Cagalj. Jamming resistant key establishment using uncoordinated frequency hopping. In Pro-ceedings of IEEE Symposium on Security and Privacy, 2008.
[13] D. Thuente andM. 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.
[14] M. Wilhelm, I. Martinovic, J. Schmitt, and V. Lenders. Reactive jamming in wireless networks: How realistic is the threat? In Proceedings of WiSec, 2011.
[15] W. Xu, W. Trappe, and Y. Zhang. Anti-jamming timing channels for wireless networks. In Proceedings of WiSec, pages 203–213, 2008.
[16] 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.
[17] 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.

Author Profile

Sontam Sunil Kumar Reddy received the B.Tech degree in computer science and Engineering from JNTU Anantapur in 2012 and pursuing M.tech. degree in Computer science and Engineering from JNTU Hyderabad.

K. Raghavendra Rao working as assistant professor in Computer Science Engineering from CVSR College of Engineering from Anurag Group of Institutions Venkatapur(V), Ghatkesar(M), Ranga Reddy District, Hyderabad-500088, Telangana State.

Licensed Under Creative Commons Attribution CC BY