Open access • Proceedings Article • DOI:10.1109/ICC.2010.5502322

Selective Jamming Attacks in Wireless Networks — Source link

Alejandro Proano, Loukas Lazos

Institutions: University of Arizona

Published on: 23 May 2010 - International Conference on Communications

Topics: Network packet, Cryptographic protocol, Transmission Control Protocol, Wireless network and Jamming

Selective Jamming Attacks in Wireless Networks

Alejandro Proaño
Dept. of Electrical and Computer Engineering
University of Arizona, Tucson, Arizona
Email: aaproano@email.arizona.edu

Loukas Lazos
Dept. of Electrical and Computer Engineering
University of Arizona, Tucson, Arizona
Email: llazos@ece.arizona.edu

***Abstract*—We address the problem of selective jamming attacks in wireless networks. In these attacks, the adversary selectively targets specific packets of "high" importance by exploiting his knowledge on the implementation details of network protocols at various layers of the protocol stack. We illustrate the impact of selective jamming on the network performance by illustrating various selective attacks against the TCP protocol. We show that such attacks can be launched by performing real-time packet classification at the physical layer. We examine the combination of cryptographic primitives with physical layer attributes for preventing real-time packet classification and neutralizing the inside knowledge of the attacker.**

I. Introduction

Wireless networks are susceptible to numerous security threats due to the open nature of the wireless medium. Anyone with a transceiver can eavesdrop on ongoing transmissions, inject spurious messages, or block the transmission of legitimate ones. One of the fundamental ways for degrading the network performance is by jamming wireless transmissions [9], [11], [19], [20]. In the simplest form of jamming, the adversary corrupts transmitted messages by causing electromagnetic interference in the network's operational frequencies, and in proximity to the targeted receivers [15].

For an adversary agnostic to the implementation details of the network, a typical jamming strategy is the continuous emission of high-power interference signals such as continuous wave tones, or FM modulated noise [15]. However, adopting an "always-on" jamming 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 high interference levels make this type of jamming easy to detect [11], [19], [20]. Third, these attacks are easy to mitigate either by spread spectrum communications [15], spatial retreats [20], or localization and removal of the jamming nodes.

In this paper, we consider a sophisticated adversary model in which the adversary is aware of the implementation details of the network protocols. By exploiting this knowledge, the adversary launches *selective jamming attacks* in which it targets specific packets of "high" importance. For example, jamming of TCP acknowledgments (ACKs) can severely degrade the throughput of a TCP connection due to the congestion control mechanism of the TCP protocol [3]. Compared to continuous jamming, the adversary is active for a short period of time, thus expending orders of magnitude less energy.

To perform selective jamming, the adversary must be capable of classifying transmitted packets in real time, and corrupting them before the end of their transmission. Packet classification can be performed by receiving just a few bytes of a packet, for example, by decoding the frame control field of a MAC-layer frame. We are interested in developing *resource-efficient* methods for preventing real-time packet classification and hence, mitigating selective jamming. Our contributions are summarized below.

A. Our Contributions

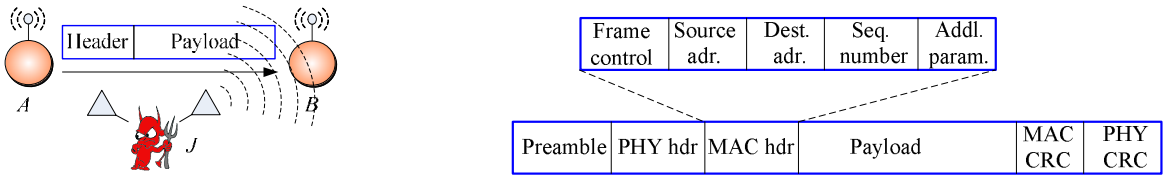
We investigate the feasibility of real-time packet classification for launching selective jamming attacks. We consider a sophisticated adversary who exploits his knowledge on network protocols along with secrets extracted from compromised nodes to maximize the impact of his attack. To mitigate selective jamming, we combine cryptographic mechanisms such as commitment schemes [6], cryptographic puzzles [7], and all-in-one transformations [13], with physical-layer parameters. We further study the impact of various selective jamming strategies on the performance of the TCP protocol.

The remainder of the paper is organized as follows. Section II, presents related work. In Section III, we describe the problem addressed, and state the system and adversarial model assumptions. In Section IV, we illustrate the feasibility of selective jamming attacks. In Section V, we develop methods for preventing selective jamming. Section VI, illustrates the impact of selective jamming on the performance of TCP. In Section VII, we conclude.

II. Related Work

Continuous jamming has been used as a denial-of-service (DoS) attack against voice communication since the 1940s [15]. Recently, several alternative jamming strategies have been demonstrated [11], [12], [19], [20]. Xu et. al. categorized jammers into four models, (a) a constant jammer that continuously emits noise, (b) a deceptive jammer that continuously broadcasts fabricated messages or replays old ones, (c) a random jammer that alternates between periods of continuous jamming and inactivity, and (d) a reactive jammer who jams only when transmission activity is detected.

Intelligent attacks which target the transmission of specific packets were presented in [8], [18]. Thuente considered an attacker who infers eminent packet transmissions based on timing information at the MAC layer. Law et. al. considered



(a) (b)

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

selective jamming attacks in multi-hop wireless networks, where future transmissions at one hop were inferred from prior transmissions in other hops. However, in both [8], [18], real-time packet classification was considered beyond the capabilities of the adversary. Selectivity was achieved via inference from the control messages already transmitted.

Channel-selective jamming attacks were considered in [4], [17]. It was shown that targeting the control channel reduces the required power for performing a DoS attack by several orders of magnitude. To protect control channel traffic, control information was replicated in multiple channels. The "locations" of the channels where control traffic was broadcasted at any given time, was cryptographically protected. In [9], we proposed a randomized frequency hopping algorithm, to protect the control channel inside jammers. Finally, Pöpper et. al. proposed a frequency hopping anti-jamming technique that does not require the sharing of a secret hopping sequence, between the communicating parties [12].

III. Problem Statement and Model Assumptions

A. Problem Statement

Consider the scenario depicted in Figure 1(a). Nodes A and B communicate over the wireless medium and a jamming node J is within communication range of both A and B. Node A transmits a packet m to B which is eavesdropped by node J. Node J is able to classify m by receiving only its first few bytes. J then corrupts m by interfering with its reception at B. We address the problems of (a) evaluating the ability of the adversary in classifying transmitted messages in real-time, and (b) developing resource-efficient mechanisms for preventing real-time packet classification.

B. System and Attacker Model

Network model–Our network consists of a collection of nodes connected via wireless links. Nodes may communicate directly, or over multiple hops. The nodes of the network can establish globally shared keys, either by manual preload, or via an online key distribution center.

Communication Model–Communication can be either broadcast or unicast. Packets are transmitted at a rate of $\mathcal{R}$ bauds. Each symbol corresponds to q bits according to the underlying digital modulation scheme. Here the transmission bit rate is equal to $q\mathcal{R}$ bps. To generalize our analysis, we do not consider any spreading of the data. However, our results hold even if data is spread to a wider spectrum according to any technique such as DSSS or FHSS [15].

Transmitted packets have the generic frame format depicted in Figure 1(b). The preamble is used for synchronizing the sampling process at the receiver. The PHY header contains information regarding the length of the frame and the transmission rate. The MAC header contains information relevant to the MAC layer. In particular, the MAC header determines the MAC protocol version, the type of packet (management, control, or data) and its subtype (e.g. association request/response, RTS, CTS, ACK, etc.), the source and destination addresses plus some additional fields regarding power management, security parameters, and information for future transmissions. The MAC header is followed by the frame body that contains higher layer information. Finally, the MAC frame is protected by a CRC code attached in the CRC field.

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. The adversary can operate in full-duplex mode, thus being able to receive and transmit concurrently. This can be achieved, for example, with the use of multiple radios. 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. The adversary is assumed to be computationally bounded, although he can be significantly more powerful than the network devices. Solving well-known hard cryptographic problems is assumed to be time-consuming.

The implementation details of the network functions at every layer of the protocol stack are assumed to be public. For example, the adversary is aware of the digital modulation scheme, the error correction and detection schemes, the MAC, and routing protocol specifications, etc. Furthermore, the adversary is capable of physically compromising network devices and recovering stored information including cryptographic keys, pseudo-random (PRN) sequences, certificates, etc. Hence, the adversary can decrypt any information encrypted with globally known keys, or jam communications protected by globally known PRN sequences.

IV. Real-time Packet Classification

In this section, we describe methods for real-time packet classification. Once a packet is classified, the adversary may choose to jam it depending on his strategy.

Consider the communication system depicted in Figure 2. At the transmitter, a message m of length ℓ passes through a channel encoder and an interleaver before it is digitally modulated for transmission in the wireless channel. At the

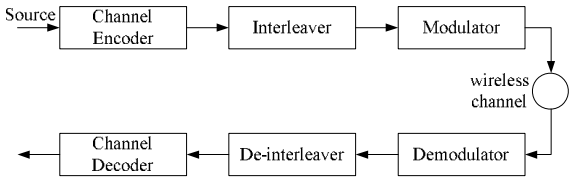


Fig. 2. Communication system diagram.

receiver, the received signal is demodulated, de-interleaved, and decoded before the original message m is recovered.

Several methods may be used for channel encoding. For example, a (n, k) block code can protect m from up to e errors per block. Alternatively, an a/b convolutional encoder can be considered with a constraint length of L_{max}, and a free distance of e bits. Interleaving is used to protect m from burst errors. For simplicity, consider a block interleaver of depth d that processes blocks of length n (in the case of convolutional encoding, blocks of n bits do not correspond to codewords). Finally, the digital modulator maps the bit stream into symbols to be transmitted over the wireless channel. Assume here that q bits are mapped to a single symbol.

To decode any data, the receiver must first collect $d \cdot n$ bits before it is able to de-interleave them. Once the received data is de-interleaved, it can be decoded using either block or convolutional decoding. Ignoring any propagation and decoding delays, the delay until the decoding of the first block of data is equal to $\Delta = \lceil \frac{d \cdot n}{q} \rceil$, measured in symbol durations. Thus, a receiver obtains $d \cdot k$ data every $\lceil \frac{d \cdot n}{q} \rceil$ symbols, in the case of a block encoder and $\frac{a}{b} dn$ bits in the case of a convolutional encoder.

As an example, the 802.11 standard uses a convolutional encoder of various rates to achieve different transmission speeds, with interleaving occurring per OFDM symbol [2]. At the lowest rate of 6 Mbps, data is passed by a $\frac{1}{2}$-rate encoder before it is mapped to an OFDM symbol of 48 bits. In this case, decoding of one OFDM symbol provides 24 bits of data. At the highest data rate of 54 Mbps, 216 bits of data are recovered per OFDM symbol.

Using the first few symbols, the adversary can obtain the header information for the transmitted packet and classify it accordingly. As an example, a MAC layer packet in the 802.11 standard can have a size up to $\ell = 2344$ bytes with a header of 30 bytes. At a rate of 6 Mbps, 98 OFDM symbols are needed to complete the transmission of the entire packet (24 bits per symbol), while the header is contained in only 10 symbols. Note that the frame type and sub-type are contained in the first two bytes of the MAC frame. Hence, a MAC frame can be classified after the reception of the first OFDM symbol, following the physical layer preamble and header. The adversary has the opportunity to corrupt the remaining symbols to successfully jam a transmitted packet.

A typical method of combating jamming is by using spread spectrum (SS) communications [15]. However, not all wireless systems are allocated sufficient bandwidth for spreading. Moreover, SS can prevent jamming only if the PRN sequence used to spread the signal is kept secret. For broadcast communications such as the transmission of control information, any PRN sequence must be known to all intended receivers. Our adversary model assumes that nodes can be physically compromised and secrets such as global PRN sequences are revealed to the adversary. In this case, SS alone cannot prevent the real-time packet classification and jamming.

An obvious solution to packet classification is the encryption of transmitted packets with a resource-efficient cryptographic mechanism such as symmetric key encryption [16]. In this case, the entire packet has to be encrypted including headers (it is a standard practice to leave headers unencrypted, so that receivers can abort early the reception of packets not destined to them). However, as in the case of SS, the decryption key must be known to all broadcast receivers and hence, this key is susceptible to compromise.

For typical symmetric encryption schemes such as block encryption, reception of one block of ciphertext is sufficient to obtain the corresponding block of plaintext, if the decryption key is known. For example, consider a block encryption mechanism operating in cipher-block chaining (CBC) mode [16]. To encrypt a message m of length $\ell \geq n$, with a key k and an initialization vector IV, message m is split into x blocks $m_1, m_2, \ldots m_x$ with $x = \lceil \frac{\ell}{n} \rceil$. Each block is processed to produce a ciphertext $c_i,\ i = 1, 2, \ldots, x+1$ of length n:

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

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

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

where $D_k(c)$ denotes the decryption of ciphertext c with key k. It is easy to see from (1) that reception of c_{i+1} is sufficient to recover m_i when k is known (the $c_1 = IV$ is also known). Therefore, the adversary can still perform real-time classification of packets encrypted with a compromised key.

V. Mitigation of Selective Jamming

In this section, we propose three schemes for countering selective jamming. Our goal is to transform a selective jammer to a random one. This can be achieved by overwhelming the adversary's computational ability to perform real-time packet classification. We first show that our problem can be mapped to the hiding property of commitment schemes [6].

A. Mapping to Commitment Schemes

Commitment schemes are fundamental cryptographic primitives that allow a committer P, commit to a value m to a verifier V while keeping m hidden. Initially, P provides V with a commitment $C = commit(m, r)$, where $commit$ is some commitment operation, and r is a random number. At a later stage, P can release additional information that reveals m. A scheme that does not allow the computation of m from C without additional information from P is called *perfect* or *hiding*, while a scheme that does not allow P to change m to a value m' once C is released, is called *binding* [6].

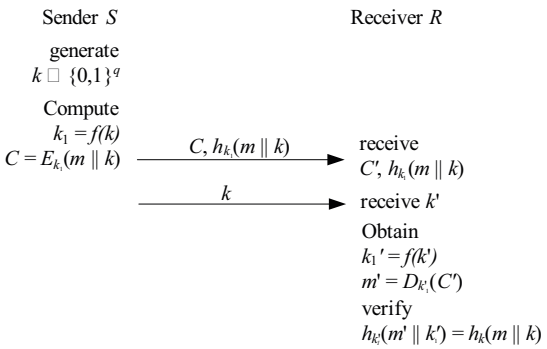


Fig. 3. A commitment scheme for preventing packet classification.

In our context, the role of the committer is assumed by the transmitting node S. The role of the verifier V is assumed by any receiver R within the communication range of S, including the jammer J. Note that S has no particular interest in modifying m after he has committed to it, since its primary goal is to communicate m. However, satisfying the binding property ensures that, (a) only S can release information that reveals m, and (b) the only value that R can accept is m. To prevent selective jamming, S first transmits C that hides m from any receiver, including J. Once the transmission of C is completed, S reveals additional information that "opens" C. Intended receivers are able to read m. We now provide a scheme that prevents packet classification based on the idea of commitments.

B. A Scheme Based on Commitments

Assume that S wants to communicate a message $m \in \{0,1\}^l$ for R. First, S selects a random key $k \in \{0,1\}^q$, where q is the number of bits mapped to a symbol at the physical layer. To utilize off-the-shelf encryption algorithms, k is expanded to $k_1 = f(k)$, where $f : \{0,1\}^q \rightarrow \{0,1\}^z$ is a public injective function, and $z = |k_1|$ is the length required by a block encryption mechanism such as DES or AES [16]. After the generation of k_1, S generates the commitment value $C = E_{k_1}(m)$, and broadcasts $\{C, h_{k_1}(m||k)\}$, where $h_{k_1}()$ is a collision-resistant keyed one-way hash function. and $||$ denotes the concatenation operation. S chooses a new k for every transmission, ensuring the randomness of C and $h_{k_1}(m||k)$ when the same m has to be transmitted. To "open" C, S releases the random key k. Upon reception of a k', R computes $k_1' = f(k')$, and obtains $m' = D_{k_1'}(C')$. The integrity of the message (i.e., $m' = m$) is verified by checking that $h_{k_1'}(m'||k') = h_{k_1}(m||k)$. Upon verification, R obtains $m' = m$. The proposed scheme is shown in Figure 3(a).

To classify m, the jammer must be capable of obtaining any part of m before the end of the transmission of k. The classification can be initated as early as the reception of the first ciphertext block of C. For a packet of length l and a ciphertext block of size n, the available decryption time is $t_d = \frac{l-n}{q\mathcal{R}}$ sec, where $\mathcal{R}$ is the transmission rate in bauds. Assuming a key pool of size $|\mathcal{K}|$, the adversary must be capable of performing on average $\mathcal{N} = \frac{|\mathcal{K}|}{2 \times t_d}$ decryptions per second (on average, the decryption key will be found after $\frac{|\mathcal{K}|}{2}$ keys have been tried).

As an example, consider the 802.11a standard which achieves a rate of 6 Mbps rate using BPSK modulation, with 24 bits being mapped to a single OFDM symbol (i.e $q = 24$). There are 2^{24} choices for k, hence $|\mathcal{K}| = 2^{24}$. In the worst case scenario, a packet m of length $l = 2344$ bytes (maximum length at the MAC layer) has to be transmitted, providing the adversary with the longest time for finding k. To send a packet m, S transmits C (2344 bytes), $h_k(m||r)$ (20 bytes), and k (3 bytes). In total, the adversary has 0.393 msec to find k and classify m, before k is transmitted. Thus, the adversary must be capable of performing $\mathcal{N} = 2.13 \times 10^(10)$ decryptions per second. This number only increases for higher values of q.

The communication overhead introduced by the commitment scheme is equal to the lengths of $h_{k_1}(m||k)$ and k (i.e 160 + q bits) per packet. Assuming packets of maximum length, and $q = 24$, the communication efficiency drops to 99%. For a control packet such as a TCP-SYN of length $l = 52$ bytes, the efficiency drops to 69%. To increase the communication efficiency, successive data messages can be encrypted with the same k. However, in this case all messages encrypted with the same key must be received before any one of them can be decrypted.

C. A Scheme Based on Cryptographic Puzzles

Cryptographic puzzles involve the creation of problems that are solvable within a finite time interval t_p which depends on the hardness of the puzzle and the computational ability of the solver. Such puzzles were first suggested by Merkle [10] and have found various applications including the prevention of DoS attacks [7].

In our context, the idea of cryptographic puzzles is employed to overwhelm the computational ability of the adversary in classifying packets. In essence, this is an implementation of a commitment scheme where the committer never reveals the information needed to open the commitment, but such information is obtained after solving a puzzle.

The time required for solving a puzzle can be controlled by using time-lock puzzles [14]. Assume that S wants to broadcast a message $m \in \{0,1\}^l$. S generates a composite modulus $n = p \times q$ where p and q are two large random prime numbers. Then he computes $\phi(n) = (p-1)(q-1)$ and $t = D \times T$, where D is the number of squarings modulo n per second that a device can perform, and T is the time that it takes to solve the puzzle. S encrypts m with a randomly selected key, $k \in \{0,1\}^s$, using a conventional symmetric algorithm such as AES [16], getting $E_k(m)$. Then S chooses a random number a modulo n and computes $C_k = k + a^{2t} \pmod{n}$, that could be done efficiently by defining $e = 2^t \pmod{\phi(n)}$ and computing $C_k = k + a^e \pmod{n}$. Finally, S transmits $\{n, a, t, E_k(m), C_k\}$. Receiver R has to compute $b = a^{2t} \pmod{n}$ to get k, and then m. Note that, without knowing p and q there is no efficient alternative to get b, but to perform the necessary squaring operations. A value of T equal to the transmission delay of C_k is sufficient to prevent the disclosure of any part of m before of the end of the transmission of C_k.

In this scheme, the transmission of $\{n, a, t, C_k\}$ introduces a communication overhead of 40 bytes per packet (n, a, t are double numbers of 8 bytes, and C_k is a key of size 128 bits). As in the case of commitment schemes, to improve the efficiency of this scheme, we can use the same key to encrypt n consecutive data messages. In this case, the receiver only solves the puzzle once for every n data packets transmitted, and the delay is reduced by $(n-1) \times T$ amount of time.

If the computational power of the receiving node is a concern, computationally efficient cryptographic puzzles based on the partial reveal of the input to one-way hash functions can be used [7]. However, such puzzles are pararellizable and can be solved in a shorter time from a computationally advanced adversary. To prevent pararellization, puzzles can be constructed by the iterative application of an encryption function such as AES with partial reveal of the decryption key. In this case, the adversary has to perform a number of decryptions in a sequential manner.

D. A Scheme Based on All-or-nothing Transformations

We now examine a solution based on All-Or-Nothing Transformations (AONT). Such transformations were originally proposed by Rivest to slow down brute force search attacks [13]. 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. The defining property of an AONT is that the *entire* output of the transformation must be known before the input can be computed. When combined with block encryption, all blocks of the ciphertext must be decrypted to obtain any part of the plaintext, thus slowing down a brute force attack by a factor equal to the number of ciphertext blocks. The following AONT transform known as the *package transform* was proposed by Rivest [13]:

1) The input message m is split to blocks m_1,m_2,$\ldots$, m_x.
2) A random key $k' \in \{0,1\}^s$ is selected.
3) The output message m' consisting of m'_1,m'_2, $\ldots$,m'_x,m'_{x+1} is computed as follows:

$$\begin{aligned} m'_i &= m_i \oplus E_{k'}(i), \text{ for } i = 1, 2, \ldots, x, \\ m'_x &= k' \oplus e_1 \oplus e_2 \oplus \cdots \oplus e_x, \end{aligned}$$

where

$$e_i = E_{k_0}(m'_i \oplus i), \text{ for } i = 1, 2, \ldots, x,$$

and k_0 is a fixed publicly-known encryption key.

In this transform, to obtain k', all m'_i must be known:

$$k' = m'_{x+1} \oplus e_1 \oplus e_2 \oplus \cdots \oplus e_x.$$

Once k' is known, m_i's can be computed as:

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

In our context, an AONT prevents the real-time packet classification of a message m, when its AONT m' is transmitted over the wireless medium. The adversary has to collect all m'_i before any part of m can be computed. The security strength of the package transform lies in the length of k', which has to be sufficiently long so that the adversary cannot guess it before of the completion of the transmission of m'. While a long key length is suggested for long term security (e.g., 128 bits), a shorter key can hide m, for a short time.

Compared to a plaintext transmission of m, the AONT transform adds a communication overhead equal to one block length (length of m'_i.) The computational overhead is approximately equal to one symmetric encryption for the sender and one block symmetric decryption of m for the receiver [13].

VI. Impact of Selective Jamming on TCP

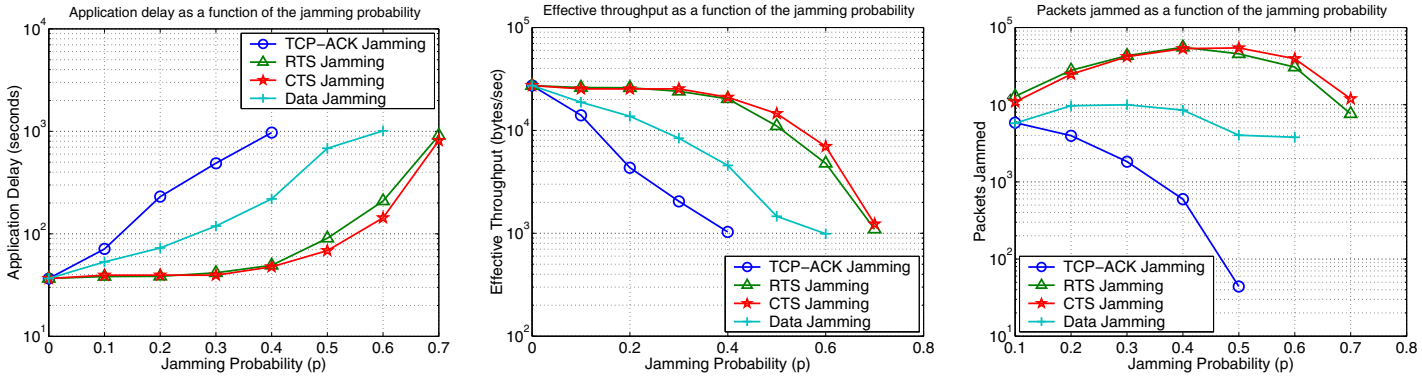
In this section, we illustrate the impact of selective jamming attacks on the network performance. In particular, we implemented a selective jamming attack against a TCP connection established over a multi-hop wireless route. In our experiments, we used the OPNET^{TM} Modeler 14.5 [1]. In the simulated scenario, a user requested a 10^6 bytes file from a server node several hops away, and the TCP protocol was used to transport the requested file. At the MAC layer, the RTS/CTS mechanism for implementing virtual carrier sensing was enabled. The transmission rate was set to 11 Mbps. 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 data packets, (b) selective jamming of RTS messages, (c) selective jamming of CTS messages, and (d) selective jamming of cumulative TCP-ACKs. In all four strategies only a fraction p of the selected messages was jammed.

A. Performance Evaluation

In order to quantify the impact of selective jamming, we measured the application delay until the file transfer was completed. We also measured the average effective throughput of the TCP connection as the fraction of the file size over the time until the file transfer was completed. Finally, we measured the number of packets that the adversary blocked in each of the four jamming strategies. In Figure 4(a), we show the application delay as a function of the jamming probability p. In Figure 4(b), we show the average effective throughput of the TCP connection as a function of p. Finally, Figure 4(c) depicts the number of packets that were jammed by the adversary for each value of p.

We observe that for a TCP connection, a selective jamming attack against TCP ACKs is significantly more harmful and efficient than all other jamming strategies. By jamming 40% of TCP ACKs, the application delay is one order of magnitude larger compared to jamming just data, and two orders of magnitude larger than jamming RTS or CTS messages at the MAC layer. Moreover, for values of p larger than 0.4, the TCP connection was aborded due to the repeated timeout of the sender. Likewise, the average effective throughput drops significantly faster when TCP ACKs are jammed, compared to jamming regular data transmissions or RTS and CTS message exchanges as it is shown in Figure 4(b).

The interpretation of the effectiveness of the selective jamming of TCP ACKs lies in the congestion control mecha-



(a) (b) (c)

Fig. 4. (a) Application Delay, (b) average effective throughput, and (c) number of packets jammed.

nism of the TCP protocol [5]. When cumulative ACKs are lost (in our case jammed), the sender has to retransmit all unacknowledged data packets, thus increasing the incurred delay while reducing the effective throughput. At the same time, the sender interprets the loss of ACKs as congestion and throttles its packet transmission rate by reducing the size of the transmission window. This leads to a further slow down of the application. A critical observation in Figure 4(c), is that as the packet transmission rate of the sender drops, the jammer has to jam fewer TCP ACKs thus reducing the amount of time that the jammer has to remain active.

Note that at the MAC layer, the number of transmission attempts for data packets is four. Hence, in order to successfully jam a TCP ACK, a total of four messages need to be jammed. On the other hand RTS/CTS messages are retransmitted a total of seven times before the sender stops its retransmission attempts. It is therefore much harder to drop all RTS/CTS retransmissions relevant to a data packet that would cause its retransmission at the TCP layer. In addition, RTS/CTS packets are significantly smaller (20 bytes long) compared to data packets, making their retransmission fairly fast. Thus, selective jamming against ACKs or data is more effective than targeting control packets at the MAC layer.

VII. Conclusion

We addressed the problem of selective jamming in wireless networks. We illustrated the effectiveness of selective jamming attacks by implementing such attacks against the TCP protocol. We showed that an adversary can exploit its knowledge of the protocol implementation to increase the impact of his attack at a significantly lower energy cost. We illustrated the feasibility of selective jamming attacks by performing real-time packet classification. To mitigate selective jamming, we proposed several methods that combine cryptographic primitives such as commitment schemes, cryptographic puzzles, and all-or-nothing transformations with physical layer attributes.

Acknowledgements

This research was supported in part by the National Science Foundation (under grant CNS 0844111).

References

[1] OPNET^{tm} modeler 14.5. http://www.opnet.com/solutions/network_rd/modeler.html.

[2] IEEE 802.11 standard. http://standards.ieee.org/getieee802/download/802.11-2007.pdf, 2007.

[3] A. Al Hanbali, E. Altman, and P. Nain. A survey of tcp over ad hoc networks. *IEEE Communications Surveys & Tutorials*, 7(3):22–36, 2005.

[4] A. Chan, X. Liu, G. Noubir, and B. Thapa. Control channel jamming: Resilience and identification of traitors. In *Proceedings of the IEEE ISIT*, 2007.

[5] D. Comer. *Internetworking with TCP/IP: principles, protocols, and architecture*. Prentice Hall, 2006.

[6] I. Damgard. Commitment schemes and zero-knowledge protocols. *Lecture notes in computer science*, 1561:63–86, 1999.

[7] A. Juels and J. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Proceedings of the Network and Distributed System Security Symposium*, pages 151–165, 1999.

[8] Y. W. Law, M. Palaniswami, L. V. Hoesel, J. Doumen, P. Hartel, and P. Havinga. Energy-efficient link-layer jamming attacks against WSN MAC protocols. *ACM Transactions on Sensor Networks*, 5(1):1–38, 2009.

[9] L. Lazos, S. Liu, and M. Krunz. Mitigating control-channel jamming attacks in multi-channel ad hoc networks. In *Proceedings of the second ACM conference on wireless network security*, pages 169–180, 2009.

[10] R. C. Merkle. Secure communications over insecure channels. *Communincations of the ACM*, 21(4):294–299, 1978.

[11] G. Noubir and G. Lin. Low-power DoS attacks in data wireless LANs and countermeasures. *ACM SIGMOBILE Mobile Computing and Communications Review*, 7(3):29–30, 2003.

[12] C. Pöpper, M. Strasser, and S. Čapkun. Jamming-resistant broadcast communication without shared keys. In *Proceedings of the USENIX Security Symposium*, 2009.

[13] R. Rivest. All-or-nothing encryption and the package transform. *Lecture Notes in Computer Science*, pages 210–218, 1997.

[14] R. Rivest, A. Shamir, and D. Wagner. Time-lock puzzles and timed-release crypto. 1996.

[15] M. Simon, J. Omura, R. Scholtz, and B. Levitt. *Spread spectrum communications handbook*. McGraw-Hill Companies, 1994.

[16] D. Stinson. *Cryptography: theory and practice*. CRC press, 2006.

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

[18] D. Thuente and M. Acharya. Intelligent jamming in wireless networks with applications to 802.11b and other networks. In *Proceedings of the IEEE MILCOM*, 2006.

[19] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 46–57, 2005.

[20] 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.