Computer Science and Artificial Intelligence Laboratory
Technical Report

MIT-CSAIL-TR-2010-028 June 7, 2010

iJam: Jamming Oneself for Secure Wireless Communication

Shyamnath Gollakota and Dina Katabi

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

CSAIL

iJam: Jamming Oneself for Secure Wireless Communication

Shyamnath Gollakota and Dina Katabi
MIT CSAIL

ABSTRACT

Wireless is inherently less secure than wired networks because of its broadcast nature. Attacks that simply snoop on the wireless medium successfully defeat the security of even 802.11 networks using the most recent security standards (WPA2-PSK). In this paper we ask the following question: Can we prevent this kind of eavesdropping from happening? If so, we can potentially defeat the entire class of attacks that rely on snooping.

This paper presents iJam, a PHY-layer protocol for OFDM-based wireless systems. iJam ensures that an eavesdropper cannot successfully demodulate a wireless signal not intended for it. To achieve this iJam strategically introduces interference that prevents an eavesdropper from decoding the data, while allowing the intended receiver to decode it. iJam exploits the properties of 802.11's OFDM signals to ensure that an eavesdropper cannot even tell which parts of the signal are jammed. We implement iJam and evaluate it in a testbed of GNURadios with an 802.11-like physical layer. We show that iJam makes the data bits at the adversary look random, i.e., the BER becomes close to 50%, whereas the receiver can perfectly decode the data.

1 INTRODUCTION

Wireless security is increasingly important. However, the broadcast nature of the wireless medium makes it inherently less secure than wired networks. Any untrusted node can simply eavesdrop on the medium and listen to transmissions within the wireless network. This weakness has been repeatedly exploited to defeat the security of wireless networks [3, 10]. Attacks that simply snoop on communications between legitimate nodes in a wireless network can successfully defeat the security of even 802.11 networks using the most recent security standards (WPA2-PSK) [2, 29, 3]. In fact, eavesdroppers today can mount such attacks simply by installing tools [1] that crack the passwords of these 802.11 networks by exploiting their broadcast nature. In this paper, we ask whether we can prevent this kind of eavesdropping, and thus fundamentally defeat the entire class of attacks that rely on snooping.

We introduce iJam, a practical PHY-layer technique that ensures that an eavesdropper cannot even demodulate a wireless signal not intended for it. iJam is designed for wireless networks that employ OFDM, which is the modulation scheme of choice for most modern wireless systems such as 802.11 a/g/n, WiMax, LTE *etc.* iJam can be used by a wireless sender-receiver pair to protect sensitive packets that are intended to be heard only by the receiver. In particular, it can be used in conjunction with existing wireless security protocols (such as 802.11 WPA/WPA2) to secure the session key establishment phase by preventing an eavesdropper from overhearing the critical handshake packets, which have been exploited in prior attacks [2, 29, 3].

iJam operates on the physical signal. The basic idea underlying iJam is simple: The sender repeats its transmission, as shown in Fig. 1. For each sample in these repeated transmissions, the receiver randomly jams *either* the sample in the original transmission, or the corresponding sample in the repetition. Since the eavesdropper does not know which signal sample is jammed and which one is clean, it cannot correctly decode data. In contrast, the receiver knows which samples it jammed. Thus, the receiver can pick the correct samples from the signal and its repetition and rearrange them to get a clean signal, which it can decode using standard methods.

While the above idea is simple, transforming it to a practical and robust protocol requires addressing important challenges.

(a) How do we ensure the jammed samples are indistinguishable from the clean samples? Jamming may change the characteristics of the signal which allows the eavesdropper to identify the jammed samples. In particular, recent work shows the feasibility of detecting jammed bits in sensor networks by examining the received signal power [23].

iJam addresses this issue by exploiting the basic properties of the OFDM PHY layer. In §4, we show that in contrast to the low rate transmission schemes using in sensor hardware, where the transmitted signal is highly structured, the OFDM time samples approximate random Gaussian complex variables. Thus, by deriving the jamming signal from a Gaussian distribution, we can ensure that the overall distribution after jamming still resembles the distribution of an OFDM signal. In §5.1, we demonstrate that even if the eavesdropper uses an optimal hypothesis testing strategy, it still experiences a significantly high bit-error rate that prevents decoding the OFDM symbols to the transmitted bits.

(b) How do we ensure that we can jam an eavesdropper independent of its location? It might seem difficult for the receiver to jam an eavesdropper located very close to the sender since the power of the jamming signal at the eavesdropper will be far lower than the power of the sender's transmitted signal. iJam addresses this problem by using a two-way

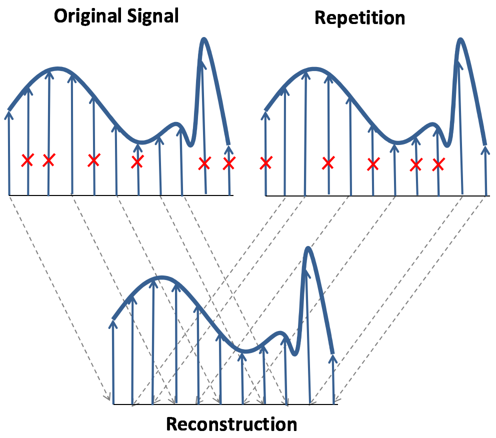


Figure 1—iJam at work. The sender repeats its transmission. The receiver-cum-jammer randomly jams complimentary samples in the original signal and its repetition. To decode, the receiver-cum-jammer, stitches together un-jammed samples to create a clean symbol.

exchange for sensitive packets. Say Bob wants to receive a sensitive packet from Alice. Bob first sends a packet with a nonce to Alice, which Alice jams using our technique. Alice then XORs its sensitive packet with the received nonce and transmits the result to Bob, who jams this transmission. Bob can then XOR the received packet with the nonce he originally sent to obtain Alice's sensitive data. Note that, since Alice and Bob are using iJam, they can decode the packets intended for them even while they jam them. However, the two-way exchange ensures that eavesdroppers cannot obtain the sensitive data. Specifically, if an eavesdropper is closer to Bob than he is to Alice, he might be able to decode Bob's transmission, but cannot decode Alice's transmission, and vice versa for an eavesdropper that is closer to Alice than he is to Bob. Since an eavesdropper needs to decode both packets to decipher Alice's sensitive data, the two-way exchange ensures that Alice and Bob's communication is protected from an eavesdropper independent of its location.

(c) How does the receiver decode despite the channel changing between the first signal and its repetition. Simply stitching the clean samples from the first transmission and its repetition does not produce a decodable signal. The channel typically changes across transmissions. Thus, before replacing a jammed sample with its repetition, iJam estimates the channel observed by both signal samples, inverts the channel impact on the clean sample, and applies to it the channel observed by the jammed sample.

iJam builds on past work on cooperative jamming [19, 12]. In particular, the closest to our work is dialog codes [4], where the receiver jams the transmitted signal to flip specific bits in the packet as received by the eavesdropper. Such a technique is however not applicable to the large class of wireless networks that use OFDM since OFDM does not send individual bits separately on the channel; rather, each OFDM signal sample is a linear combinations of many modulated bits [13]. Further, dialog codes assume that 1) jammed bits are undetectable, and 2) the sender and jammer are synchronized at the eavesdropper to the bit level [4]. Past work however shows that jammed/corrupted bits are easy to identify [16, 23]. Additionally, flipping specific bits requires time and phase synchronization of the transmitter and the jammer at the eavesdropper. Such synchronization needs knowledge of the channels from each of them to the eavesdropper, but these channels cannot be obtained without the cooperation of the eavesdropper in the first place! In contrast, iJam ensures that jammed bits are undetectable by exploiting the Gaussian statistics of OFDM signals.[1] Further, since it does not try to flip specific bits at the eavesdropper, it does not require the jammer and transmitter to be synchronized.

We implement iJam in GNURadio and evaluate it in a 20-node testbed of USRP radios [15]. Nodes in our testbed use a 802.11-like physical layer. Our evaluation reveals the following.

- The bit error rate at an eavesdropper ranges from 40-60%, which means that an eavesdropper cannot do much better than randomly guessing the contents of the packet. This is true even in extreme scenarios such as the eavesdropper being very close to the transmitter or the jammer, as well as at various positions between them.
- Jamming has no impact on packet decodability at the intended receiver. Specifically, for the range of SNRs in $[7, 25]$ dB, the bit error rate with and without jamming is the same.
- iJam is efficient. For an average receiver capable of using 16-QAM, iJam can securely deliver a 512 bit key (nonce) from a sender to a receiver in about 14ms. Thus, it can protect the two nonces typically used to crack 802.11 WPA2-PSK passwords in about 28 ms.

2 Related Work

Jamming has traditionally been used in adversarial manner to prevent others from communicating over the wireless medium [28]. Recently however, there has been interest in cooperative or friendly jamming methods. In [19, 12, 7], a trusted third party jams the communication from sender to receiver. The jamming signal is known to the receiver, which decodes using interference cancellation. In contrast, the eavesdropper does not know the jamming signal and hence cannot decode the transmission. The work in [22] presents a variation on the above model where the sender itself transmits the data combined with a jamming signal. The third party node transmits an anti-jamming signal that cancels out the jamming signal at the receiver but not at the adversary. In contrast to the above work, iJam achieves security without a trusted third party, and is further implemented and evaluated in a testbed.

iJam is related to dialog codes [4], where the receiver jams the transmitted signal to flip specific bits in the packet as

[1]Note that this does not contradict the work on SoftPHY and partial packets [16] because these schemes assume the error is caused by noise while here the jamming is designed to ensure undetectability. See §4 for the details.

received by the eavesdropper. This approach however is not applicable to OFDM networks since in OFDM every signal sample refers to multiple bits [13]. Further, it assumes tight synchronization and that jammed bits are undetectable.

iJam builds on past work in the area of physical layer security. Specifically, work in information theory[27, 14] has shown that if the channel to the receiver is better than the channel to eavesdropper, the sender-receiver pair can securely communicate. Special codes have been proposed to secure the communication in this setting [25]. iJam provides physical layer security but it does not rely on the channel to the adversary being worse than the channel to the receiver.

Recent works on physical security extract secret keys from the properties of the wireless channel [17, 21, 5, 20]. Some of the proposed techniques can be used with existing 802.11 hardware [17]. These techniques however require the channel between the sender and receiver to change quickly and be reciprocal [17]. In comparison, iJam modifies the PHY layer, but it imposes no constraints on the wireless channel, and hence it works for any environment, whether it is mobile or static.

Finally, successful attacks on 802.11's most recent security protocol have exploited the broadcast nature of wireless [2, 10]. Specifically, 802.11 WPA/WPA2-PSK uses a 2-way handshake for authentication and establishing a session key. An AP sends to the client a random nonce as a challenge, and expects to receive a password-encrypted version of that nonce as a response. A correctly encrypted response indicates that the client possesses the correct password. However, an eavesdropper can simply snoop on the medium, observe both the unencrypted nonce in the AP's challenge (plaintext), and its encrypted version in the client's response (ciphertext), then mounts standard dictionary attacks on the combination. Since in practice users tend to optimize for convenience and go for short and human-understandable passwords, such attacks have been successful, even in the presence of a strong cryptographic protocol [3, 29].

3 Adversary and Jammer Models

We start by describing the jammer and adversary.

3.1 Jammer

Wireless hardware typically has separate transmit and receive chains. A switch connects the antenna to the transmit chain during packet transmission and switches the connection to the receive chain during reception. This switching process incurs a short delay, which is negligible for traditional systems. In iJam however, since we jam individual signal samples, we need to switch between the transmit and receive chains with no delay. Thus, when jamming a particular packet, iJam turns off the switch between the transmit and receive chains, i.e., it keeps both chains running and connected to the antenna for the duration of the packet. This allows us to jam while receiving. The jamming signal is set to zero whenever the hardware wants to receive a clean signal sample, and is non-zero otherwise. We note the following:

- Such a setup is already supported by various wireless hardware. For example, the USRP radio boards allow the user to turn the switching between the receive chain and transmit chain off using a software command.
- The above does not contradict the fact that wireless radios cannot transmit and receive simultaneously. Whenever the receiver sends a non-zero jamming signal, the corresponding received sample is corrupted because the transmit power overwhelms the receive chain at that moment.
- Jamming does not hamper Automatic Gain Control (AGC). The AGC is set based on the packet's preamble and is fixed for the duration of the packet. Since iJam does not jam the preamble, it does not affect AGC settings.

3.2 Adversary

We focus on a passive eavesdropper. The eavesdropper may know the iJam protocol, the location of the communicators, and other public information such as the details of OFDM and 802.11.

The eavesdropper can be anywhere in the network and is free to move or stay static. The location and the characteristics of the eavesdropper's channel are unknown to the jammer or the sender. The eavesdropper, however, cannot be in two locations at the same time. Multiple eavesdroppers may exist in the network but we assume they do not collude.

The eavesdropper can operate at the packet level, the bit level, or the signal level. For example, the eavesdropper can consider the jamming signal as noise and try to decode in the presence of jamming. Or it can take a stronger approach and examine the received signal samples in an attempt to distinguish jammed samples from clean samples. In §5.1, we discuss these approaches and show that iJam is robust to both.

Also, instead of considering the jamming signal as noise, the adversary can implement interference cancellation or joint decoding in an attempt to simultaneously decode the jamming signal and the original transmission and separate them. This approach however does not work because basic results in multiuser information theory say that decoding multiple signals is impossible if the total information rate is outside the capacity region [26]. In iJam, we ensure that the information rate at the eavesdropper exceeds the capacity region by making the jammer transmit at an excessively high information rate. This can be easily done by making the jamming signal samples i.i.d.s and sending them at a very dense modulation. Specifically, we use a modulation of 65,536 QAM. (This corresponds to having a resolution of 8 bits for both the I and Q components of the signal.) In comparison with existing 802.11 bit rates which use a maximum of 64 QAM, this is an excessively high bit rate.

Finally, we assume that the eavesdropper's hardware is as powerful as that of the sender and receiver, i.e., it has the same number of antennas and does not use a directional antenna, unless the receiver uses a directional antenna. iJam can

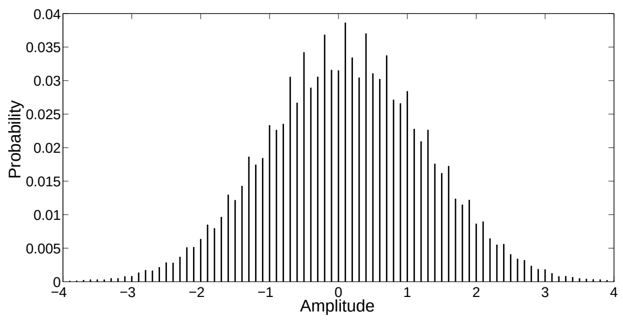


Figure 2—Amplitude Distribution of OFDM signal samples. The OFDM uses a 64-point FFT and modulates bits using 4QAM. The distribution follows a zero-mean Gaussian.

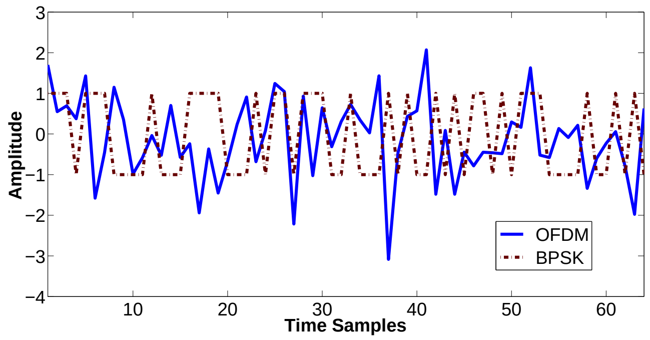


Figure 3—Time domain signal samples for BPSK and OFDM. In contrast to BPSK, the OFDM signal spans a wide range of values.

be extended to work even when these two constraints do not hold, as we explain in §9. However, the prototype described in this paper does not implement these extensions.

4 Impact of Jamming on OFDM Signals

At a high level, OFDM works as follows: the transmitter takes a sequence of bits and converts them into complex numbers by applying quadrature amplitude modulation (QAM). Next, the transmitter takes blocks of N such complex numbers ($N = 64$ in 802.11), and apply the inverse fast fourier transform (IFFT) to them, i.e.:

$$\mathbf{x}_k = \sum_{n=0}^{N} \mathbf{X}_n e^{i2\pi kn/N},$$

where $\mathbf{X}_n$ is a modulated complex number. The output of the IFFT, i.e., $\mathbf{x}_k$, is then transmitted on the channel as the time samples of the signal.[2] Thus, each time sample in the signal is a linear combination of multiple modulated bits.

The design of OFDM has two implications for jamming:

(a) In OFDM, it is hard to distinguish jammed samples from clean ones. Since each sample is a linear combination of N random modulated bits, by the central limit theorem, each of the OFDM time samples approximately takes values from a random Gaussian distribution [26]. Fig. 2 shows the distribution of OFDM signal samples at the output of GNURadio, for an OFDM system that uses 4-QAM modulation and $N = 64$ (which is the value used in 802.11). The figure confirms the above argument, showing that the signal distribution follows a zero-mean Gaussian. Thus, the amplitude of an OFDM sample can take a wide range of values.

Compare such an OFDM signal with BPSK, a transmission system commonly used in sensor hardware [16]. BPSK transmits a "0" bit as -1 and a "1" bit as +1. Thus, in the time domain, the BPSK signal takes only two values (-1,+1). Fig. 3 shows an OFDM signal against a BPSK signal. Since the BPSK signal takes only two values, it is relatively easy to identify jammed bits as those far away from the two expected values. In contrast, since the OFDM signal spans a whole range of values, it is hard to simply look at the amplitude of a sample and identify whether it is jammed or not.

Further, if one picks the jamming signal also from a zero-mean Gaussian distribution, then the combination of the jamming and original signals will also have Gaussian statistics (this is because a linear combination of two independent Gaussians is a Gaussian). This makes it even harder to tell which sample is jammed. In §5.1, we analyze an eavesdropper that uses hypothesis testing to identify the jammed samples and show that iJam is resilient to such attack.

(b) In OFDM, there is no one-to-one map between a bit and a time sample. In OFDM, each time sample is a linear combination of many modulated bits. Thus, jamming a particular OFDM sample, affects multiple bits at the same time. Compare that to BPSK, where each bit is modulated into one signal sample, and thus, jamming a particular sample of a BPSK signal corrupts only the bit that is encoded into that sample. This difference between OFDM and BPSK means that jamming proposals for sensor networks which flip specific bits do not apply to OFDM.[3]

5 iJam

iJam is a PHY-layer technique that enables two wireless nodes to exchange unencrypted confidential packets, in the presence of an eavesdropper. Without loss of generality, we focus on exchanging a random key of B bit length. Once the two nodes have exchanged a random key, they can use it either as a onetime pad [27], or as an encryption key in a shared-key cryptographic protocol. Our default value of B is 512 bits. Larger keys can be obtained by repeating the process multiple times.

For the rest of this paper, we focus on one sender and its receiver, and describe how iJam enables the sender to deliver a random key to the receiver in the presence of an eavesdropper. Also, while iJam applies to any OFDM based system, we focus our description on 802.11.

[2]The transmitter also appends a cyclic prefi x [13].

[3]One might think that with OFDM, individual bits can be flipped by jamming specifi c frequency bins and then applying the IFFT. This, however, does not work because fi rst, the frequency domain signal can use 4QAM or higher modulations, and second, the IFFT operation spreads frequency domain jamming across all the time samples, which overwhelms the AGC at all time samples, precluding decoding by the intended receiver.

The basic idea in iJam is as follows. The 802.11 driver at the sender generates a random sequence of B bits, which we refer to as a *salt*. The driver delivers the salt to the PHY for transmission, along with the standard packet header. The PHY generates the OFDM signal corresponding to the packet. The OFDM symbols corresponding to the header are generated as usual without repetition. However, for each OFDM symbol corresponding to the salt, the PHY sends 2 copies back-to-back.

The PHY layer at the receiver starts by decoding the packet's header. If the header is marked to indicate a iJam packet and the MAC address matches the receiver's MAC address, the PHY waits until the end of the header, then starts jamming the transmission.[4] For each received signal sample from the salt, the PHY either jams the original sample or its repetition. Since an OFDM symbol and its repetition are back-to-back, the PHY knows how to match a sample and its repetition. To jam a sample, the PHY transmits a signal sample that is drawn randomly from a zero-mean Gaussian distribution whose variance is set to the variance of an OFDM signal with the same modulation.[5]

To decode the salt, the PHY stitches the unjammed samples together to create a clean version of the OFDM signal corresponding to the salt. It then decodes this clean signal to obtain the bits in the salt. After decoding, the receiver checks the packet checksum and if the packet is correctly received, it sends an acknowledgment to the sender. If the sender does not receive an ack, it repeats the process with a different random salt. Once the sender and receiver have successfully exchanged a salt, they can use it to generate a random key.

In the following few sections we expand this basic idea to make it robust to various adversarial scenarios.

5.1 Optimal Strategy to Detect Jammed Samples

To make iJam robust, we need to ensure that an eavesdropper cannot distinguish jammed samples from uncorrupted samples. We had earlier argued that it is difficult for an eavesdropper to simply look at an OFDM sample and identify whether it is jammed or not. However, since iJam repeats each sample, an eavesdropper has additional information, and can compare an OFDM sample with its repetition to guess which one is jammed.

In particular, a jammed sample is the sum of two zero-mean Gaussian variables: the data sample received from the sender, and the jamming sample received from the receiver. Recall that the sum of two independent zero-mean Gaussian variables is also a zero-mean Gaussian variable, whose variance is the sum of the two variances [26]. Therefore, the jammed samples have higher variance than clean samples.

[4]In practice, the hardware pipeline at the receiver has a decoding delay of 2-4 OFDM symbols. Thus, to ensure that the receiver can jam all data samples, the transmitter inserts a pad of 4 OFDM symbols at the end of the header. Since the receiver knows the delay in its pipeline and the size of the pad, it knows when the first data sample starts and hence can jam it.

[5]The receiver knows the modulation of the packet from the header. Given a particular modulation, the variance of the OFDM signal is known and can be computed using Matlab.

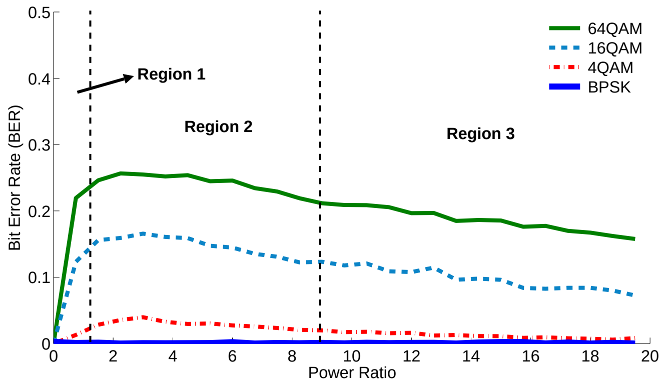


Figure 4—Performance of an optimal hypothesis-testing adversary. The figure shows the Bit Error Rate (BER) for different modulations as a function of the ratio of the jamming power to the transmitter power at the eavesdropper. The graph can be divided into three regions: Region 1 where the power from the jammer is lower than the transmitter, Region 2 where the power ratio is such that it maximizes the BER, and Region 3 where the power from the jammer is significantly higher than the transmitter.

An eavesdropper can exploit this fact to improve its ability to identify jammed samples.

Specifically, the eavesdropper can apply an optimal hypothesis testing strategy as follows. Given two transmissions of the same sample, one of which is jammed by the receiver, let S_1 denote the first OFDM sample received by the eavesdropper, and let S_2 denote the second. Let H_1 denote the hypothesis that S_1 is jammed, H_2 the hypothesis that S_2 is jammed, and C the condition that one of S_1 and S_2 is jammed. The eavesdropper can now apply a maximum likelihood test as follows:

$$Pr(H_1|S_1,S_2,C) \quad {}^{H_1}\gtrless_{H_2} \quad Pr(H_2|S_1,S_2,C)$$

Substituting the probabilities for H_1 and H_2, we get:

$$P(S_1 \textit{ is jammed}|S_1,S_2,C) \quad {}^{H_1}\gtrless_{H_2} \quad P(S_2 \textit{ is jammed}|S_1,S_2,C)$$

Thus, the optimal hypothesis testing reduces to the following:

$$P(S_1,S_2|S_1 \textit{ is jammed},C)^{H_1}\gtrless_{H_2}P(S_1,S_2|S_2 \textit{ is jammed},C)$$

After substituting the Gaussian probabilities and rearranging the terms, the maximum likelihood test reduces to:

$$|S_1|^2 \quad {}^{H_1}\gtrless_{H_2} \quad |S_2|^2$$

Thus, when comparing a sample to its repetition, the eavesdropper's best guess is to assume the one with the smaller magnitude to be clean. The eavesdropper can then apply this test to all samples and their repetitions to obtain its best guess of the transmitted salt.

Given that the eavesdropper applies the above optimal strategy, how well does she perform? We calculate an upper bound on the performance of the eavesdropper by simulating in Matlab the worst scenario: the eavesdropper receives the transmitted signal perfectly (with infinite SNR) in the absence of the jammer. For each modulation scheme (BPSK, 4-QAM, 16-QAM, 64-QAM over OFDM), we vary the power of the jammer, use the optimal hypothesis test to collect a clean signal and decode using standard methods. Fig. 4 plots the bit error rate as a function of the ratio of the jamming power to the transmitted power at the eavesdropper. The figure shows 4 graphs one for each modulation scheme. Three points are worth noting:

- Jamming can significantly increase the BER at the eavesdropper for 4-QAM, 16-QAM, and 64-QAM. However, iJam still needs an additional mechanism to amplify this BER to 50%, which would ensure that it receives no information from its snooped receptions.
- There is a significant range of relative powers of the sender and the jammer at the eavesdropper for which the BER is maximized. In particular, the BER is close to the maximum when the sender's power at the eavesdropper, $P_{s\rightarrow e}$, and the jammer's power at the eavesdropper, $P_{j\rightarrow e}$ satisfy the relationship $1 < \frac{P_{j\rightarrow e}}{P_{s\rightarrow e}} < 9$. Since the power ratio at the eavesdropper depends on its location, iJam needs an additional mechanism that works at all power ratios to allow it to be location independent.
- Finally, the BER for BPSK over OFDM is very low and hence we cannot use iJam's scheme directly. Thus, iJam needs to find an alternative approach to transmit over channels that have low SNR and for which 802.11 would typically use BPSK over OFDM.

The next three sections address the above three challenges. We start with making iJam location independent.

5.2 Making iJam Location Independent

As we saw in Fig. 4, the simple jamming idea works only in region 2, i.e., when $1 < \frac{P_{j\rightarrow e}}{P_{s\rightarrow e}} < 9$. So, how do we deal with scenarios in which the eavesdropper is in a location that does not satisfy the above constraint?

(a) Dealing with Region 1 (i.e., $\frac{P_{j\rightarrow e}}{P_{s\rightarrow e}} \leq 1$.) Region 1 occurs in locations where the jamming power is too low. This means that the eavesdropper is not really affected by the jamming and therefore has a low BER. iJam addresses this problem by using a 2-way exchange of salts. Say Alice and Bob want to exchange a random key. Alice first sends a random salt to Bob, which Bob jams using our technique. Bob then sends a new random salt to Alice, which Alice jams using our technique. Both Alice and Bob know the two salts, the one they received and the one they sent. They XOR the two salts to obtain the random key.

Given this choice of key, we can completely ignore eavesdroppers in region 1, that is eavesdroppers for which $\frac{P_{j\rightarrow e}}{P_{s\rightarrow e}} \leq 1$. Specifically, the eavesdropper cannot obtain the key unless she correctly decodes both salts. Yet, for any eavesdropper either the power received from Alice is larger than the power received from Bob or the opposite. Since Bob acts as the jammer for the first salt and Alice acts as the jammer for the second salt exactly one salt will fall in region 1. Thus, as long as we can deal with region 2 and 3, we can ensure that the eavesdropper will not get both salts, and the communication is secure. Since region 2 naturally has the highest BER, we only need to worry about region 3.

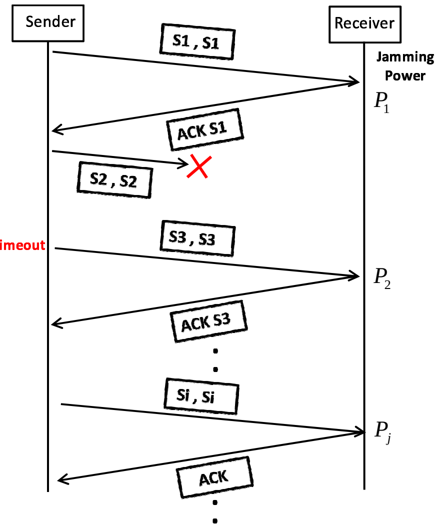


Figure 5—Making iJam location independent. The sender transmits salt and its repetition and waits for acknowledgments. In the absence of an acknowledgments, the sender timeouts, discards the unacked salt and transmits a new random salt.

(b) Dealing with Region 3 (i.e., $\frac{P_{j\rightarrow e}}{P_{s\rightarrow e}} \geq 9$.) Region 3 occurs in locations where the jamming power is too high. This is problematic because the eavesdropper can identify the jammed samples with high probability and hence obtains a low BER. So the solution to this problem is to reduce the jamming power so that the ratio $\frac{P_{j\rightarrow e}}{P_{s\rightarrow e}}$ stays relatively small. The problem however is that for any power value that the jammer picks, there exist eavesdropping locations for which the ratio $\frac{P_{j\rightarrow e}}{P_{s\rightarrow e}}$ is too high and others for which the ratio is too low. Thus, the jammer cannot cover all eavesdropping locations with one setting for the jamming power.

To address the above problem, iJam uses L different power levels to jam. Specifically, instead of transmitting one random salt in each direction, iJam transmits L salts in each direction (L salts from Alice to Bob and L salts from Bob to Alice). As before, the OFDM symbols corresponding to each of these salts are repeated twice.

As shown in Fig. 5, the jammer jams each salt and its repetition using a different power level. In particular, the jammer jams the first salt (and its repetition) using the maximum power supported by the hardware P_1. It jams the second salt (and its repetition) using a power $P_2 = \frac{P_1}{9}$, and the third salt (and its repetition) using a power $P_3 = \frac{P_2}{9}$, and so on until it jams the L^{th} salt (and its repetition) with a power level $P_L = \frac{P_{L-1}}{9}$. After exchanging L random salts in each direction, the two nodes generate the key by XOR-ing all $2L$ salts together. Note that the adversary can not correctly decode the key. This is because, for every adversary location, there exists at least one salt for which the power ratio satisfies the condition, $1 \leq \frac{P_{j\rightarrow e}}{P_{s\rightarrow e}} \leq 9$.

We note the following two points:

- First, since the receiver may fail to decode a salt, we need the receiver to acknowledges every salt and the transmitter to continue sending salts until the receiver acknowledges L such salts. The key is then generated by xor-ing only the acked salts. Also the jammer moves on to the next power level only after it correctly decodes a salt.

- Second, the number of jamming levels, L, can be computed given an upper and lower bounds on the jamming power. The upper bound is set to the maximum power supported by the hardware and the lower bound is set to the noise power. Given typical values for the maximum 802.11 power and the noise power we estimate L to be about 10 different power levels.[6]

5.3 BER Magnification

Now that we have made iJam's performance location independent, we address the second challenge. In particular, we would like to magnify the BER at the eavesdropper to be close to 50%, so that the eavesdropper gets no more information than she would get from a random guess.

To do so, we again use the XOR technique from the previous section. Specifically, instead of transmitting just one salt (and its repetition) at every power level, the transmitter transmits a train of M salts. The final salt for each power level is then constructed by XOR-ing all these M salts.

iJam can achieve a BER around 50% in the final salt by picking an appropriate value of M depending on the modulation used. In particular, since the jamming signal is random, the bits which are in error in each of the M salts are independent. If the BER in each of the individual salt is x, the probability that the i^{th} bit is uncorrupted in all M salts decreases exponentially with M as $(1-x)^M$. Thus, by picking an appropriate M, iJam magnifies the BER at the eavesdropper to close to 50%.

The final issue is that iJam needs to ensure that the receiver receives M correct salts, even in the presence of a magnified BER to the eavesdropper. First, the bit error rate at the receiver and eavesdropper are independent of each other. This property is fundamental to the wireless medium [26]. Second, the bit error rate at the adversary does not change depending on whether the receiver successfully receives the salt or not. Thus, all we need to ensure is that the receiver successfully receives some M salts from the transmitter. To do this, as in the previous section, if the receiver fails to decode a particular salt, she does not acknowledge the salt. The transmitter discards unacked salts. At each power level, however, the transmitter continues to send salts until the receiver acknowledges M of them. The final salt for the power level is then generated by XOR-ing only the acked salts.

5.4 Making iJam work at BPSK SNRs

Finally, as shown in Fig. 4, we cannot transmit salts using BPSK over OFDM because of the resulting low BER. Thus, we need an alternate mechanism to transmit salts to a receiver that has low SNR and for which 802.11 would typically use BPSK over OFDM.

To deliver packets to such a receiver, while maintaining a high BER at the eavesdropper, the transmitter sends using 4-QAM over OFDM. However, since 4QAM has a much higher bit error rate at the low SNRs at which BPSK operates, the receiver is likely to see many bit errors in the whole packet. To counter this effect, a iJam transmitter splits a salt into several sub-salts and sends a CRC checksum for each sub-salt. Since sub-salts are smaller, they are much more likely to be correctly received than a complete salt despite the higher BER of 4-QAM. The receiver only acknowledges and uses correct sub-salts for constructing the final salt. An unacked sub-salt is discarded by the transmitter. The transmitter however transmits as many sub-salts as are necessary to create a full length salt. For example, depending on the SNR, the transmitter might choose the size of the sub-salt to be 128 bits and send a CRC check for every 128 bits. In §*8.2.2*, we show that using this mechanism, iJam can successful establish keys with receivers whose SNR is as low as 6 dB, which is at the lower end of the operational regime for BPSK over OFDM [9].

5.5 Summary

To summarize, say Alice and Bob want to exchange a random key. Alice transmits to Bob L sequences of M salts, and Bob jams each of these L sequences with a different power level. Similarly, Bob transmits to Alice L sequences of M salts, and Alice jams each of these L sequences with a different power level. The final key is generated by XOR-ing all $2ML$ salts together.

To reach low SNR receivers that typically require BPSK over OFDM, a iJam sender transmits its salts using 4QAM. It however divides each salt into several sub-salts and sends a CRC checksum for each sub-salt. The salt is constructed by concatenating successful received sub-salts. The rest of the protocol stays the same as above.

6 PRACTICAL ISSUES

Finally, iJam needs to address a few practical issues to ensure decodability at the receiver, and successful jamming at the eavesdropper.

6.1 Decoding at the Receiver

An iJam receiver takes the clean samples from the original OFDM symbol and its repetition and combines them to create a single clean OFDM symbol. However, naively combining the samples across the two symbols does not work. This is because, in practice, the received signal typically has a frequency offset. This results from the fact that it is virtually impossible to manufacture two radios centered at the same exact frequency. Hence, there is always a small frequency difference, Δf, between the sender and the receiver [13]. This frequency offset causes a linear displacement in the phase of the received signal that increases over time. In particular, the

[6] 802.11 transmits around 15dBm and have a noise fbor around -95dBm [6]. This translates to about 10-11 different power levels.

phase in the OFDM signal increases by $2\pi\Delta f$ every sample. As a result, instead of receiving the samples $y_1, y_2, \cdots, y_N$, the receiver receives,

$$y_1 e^{2\pi\Delta f}, y_2 e^{4\pi\Delta f}, \cdots, y_N e^{2N\pi\Delta f}$$

The standard OFDM receiver has algorithms to account for this phase shift in a standard OFDM signal. However, since iJam combines samples from an OFDM symbol and its repetition, and gives the resulting signal to the standard OFDM receiver, iJam needs a mechanism to deal with this phase shift.

Specifically, if the repetition signal is D samples away from the original signal, then it is received at the receiver as,

$$y_1 e^{2(D+1)\pi\Delta f}, y_2 e^{2(D+2)\pi\Delta f}, \cdots y_N e^{2(D+N)\pi\Delta f}$$

Therefore, the samples in the original and the repetition signals do not have the same phase. In order to combine samples across the two, the iJam receiver should account for this phase. We note that the phase of every sample differs by exactly $2D\pi\Delta f$ between the original samples and their repetition. So, an iJam receiver multiplies all the samples in the repetition signal by $e^{-2D\pi\Delta f}$ before combining. It can easily do this because it knows D and also can estimate the frequency offset, Δf, from the preamble using standard correlation techniques [11]. Now the original samples and their repetition have the same phase and hence, the receiver can combine clean samples across them to create a clean symbol. The receiver then gives these resulted samples to the standard OFDM receiver, which then applies its algorithms to deal with frequency offset and to perform channel tracking.

Note that frequency offset is the only channel effect that iJam needs to address to ensure that stitching clean samples from an OFDM produces a correct representation of the original OFDM symbol. This is due to the fact that iJam makes the transmitter send an OFDM symbol and its repetition back-to-back. Since the duration of an OFDM symbol is much shorter than the channel coherence time [26], channel attenuation and delays do not change over such interval.

6.2 Jamming at the Eavesdropper

We note the following points about iJam's jamming.

- First, for iJam to work, the jammer needs to corrupt complementary samples in the original OFDM symbol and its repetition. The reader might think that, to achieve this, the jammer needs to be synchronized with the transmitter. Synchronizing different transmitters is a hard problem in wireless communication [8, 18, 11]. Fortunately, iJam does not require this synchronization. In particular, it is important to note that iJam only needs to identify complementary samples (i.e., a sample and its repetition) *at the intended receiver*. Thus, all iJam needs to know is the start of the OFDM symbol as seen by the receiver. Since the jammer is the same as the receiver, this is naturally achieved with current OFDM synchronization algorithms. In particular, in order to decode, standard OFDM receivers have a packet detection algorithm to detect the beginning of the first OFDM symbol. Once the receiver locks on the beginning of the first OFDM symbol, it simply advances by the length of an OFDM symbol, which is a known parameter of the system. So all the jammer has to do is to let the standard OFDM receiver algorithm locate the OFDM symbols. Once this is done, the jammer can match a sample with its repetition because they are separated by exactly the length of an OFDM symbol. Therefore, iJam can perform the jamming operation without any need for additional synchronization.

- The jammer can accurately jam individual samples without worrying about the wireless multi-path affecting the ability of the receiver to decode. In particular, since the receiver and the jammer use the same antenna, the receiver receives the jamming signal effectively on the wire, instead of the wireless medium. Thus, at the intended receiver, the jamming signal does not suffer any multi-path effect and hence the jammer can accurately jam individual samples. We, however, note that since the jamming signal traverses the wireless medium to the adversary, the wireless multi-path further corrupts the signal at the adversary. Specifically, if the signal traverses multiple paths from the jammer to the eavesdropper, the eavesdropper receives multiple copies of every jamming sample, each with a different delay. Thus, each jamming sample ends up corrupting more than a single sample in the sender's signal, increasing the effect of jamming.

- Finally, OFDM transmits a cyclic prefix (CP) between every two OFDM symbols to prevent inter symbol interference. Since the CP just repeats some samples from the OFDM symbol, the jammer needs to jam these repetitions as well to prevent the eavesdropper from exploiting them. An iJam jammer therefore corrupts the cyclic prefix by jamming all samples between every two OFDM symbols, as detected by the receiver. Note that since the jamming signal corrupts only the samples between the OFDM symbols at the receiver, which are anyway discarded, the ability of the receiver to decode does not change.

7 IMPLEMENTATION

We implement iJam using the Universal Software Radio Peripherals (USRPs). We use the RFX2400 daughterboards which operate in the 2.4 GHz range. We build on top of the GNU Radio software base and a 802.11-like physical layer. Specifically, the transmitter maps the bits to the OFDM subcarriers. We use 64 subcarriers in our OFDM implementation, which is the same number of subcarriers used by 802.11. The mapping from bits to subcarriers is done by using different modulations. In particular, we use 4-QAM, 16-QAM and 64-QAM. Finally, each packet consists of the OFDM preamble, a header and a 1500-byte payload.

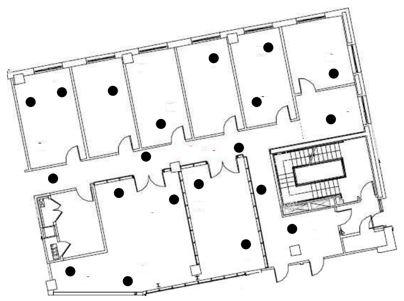


Figure 6—Testbed Topology

7.1 Timing Requirements

iJam requires the OFDM receiver to know the start of the OFDM symbol in real-time so that it can jam complementary samples in an OFDM symbol and its repetition. While this can be done in hardware or FPGA implementation, where the receive can make decisions online, it is hard to do in USRPs. It is well known that USRP/GNURadio can not achieve strict timing required for 802.11 because GNURadio manipulates the signal on the PC in user mode [15]. Thus, our evaluation does not pick the jamming pattern on the fly. Instead it uses a fixed pattern which is determined offline. Additionally, our evaluation picks patterns that do not require the jammer to know symbol boundaries because this information is available only after decoding, which is done offline. Specifically, the jammer uses the jamming patten 1010101.., i.e., it continuously jams every alternate sample. If the cyclic prefix is of odd length, the jammer effectively jams complimentary samples in the symbol pair. So, we use 17 samples for the cyclic prefix, which is one more sample than that used in 802.11. Note that using this pattern to evaluate iJam is justifiable because it is one of the plausible instantiations of a random jamming pattern. We also tried other patterns and cyclic prefix combinations that have the same property, such as 11001100.. and a cyclic prefix length of 18 (2 *modulo* 4). We found no difference in performance across patterns. Further, the results are similar in nature to the simulation results in MATLAB which use completely random patterns.

8 RESULTS

We evaluate iJam in a representative indoor testbed shown in Fig. 6. The testbed has 20 nodes in both line-of-sight and non-line-of-sight configurations. Each node is a commodity PC connected to a USRP.

8.1 Effectiveness of Jamming

8.1.1 *Can iJam achieve high BER at an eavesdropper?*

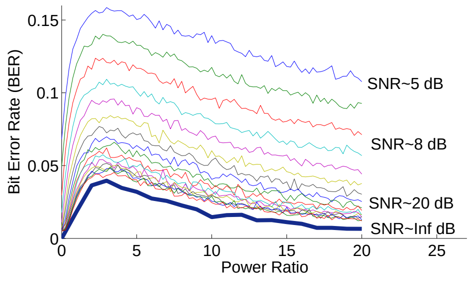
We would like to confirm that, in practice, the impact of jamming at the eavesdropper follows the theoretical predictions from §5.1. In particular, we want to check how an optimal hypothesis testing eavesdropper performs as a function of the ratio of the power it receives from the jammer and sender, $\frac{P_{j\to e}}{P_{s\to e}}$.

To perform this experiment, we randomly pick two nodes from the testbed to be the sender and the receiver/jammer. For each choice of sender and receiver/jammer nodes, we place the eavesdropper node at various random locations and also control the jamming power, in order to span the whole range of power ratios.

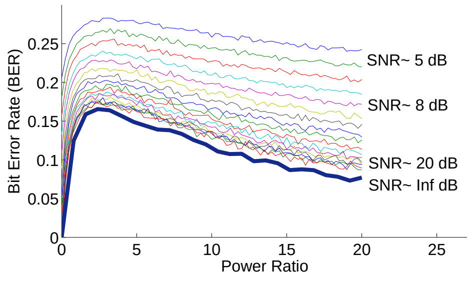
The ability of the eavesdropper to decode a salt from the sender depends on the SNR of the sender's signal in the absence of the jammer. Thus, we consider eavesdropper locations that cover the range of 802.11 sender SNRs. We plot the results of this experiment for 4-QAM, 16-QAM and 64-QAM over OFDM in Fig. 7. The x-axis shows the ratio of the signal power of the jammer to the signal power of the sender at the eavesdropper. The y-axis shows the BER. Each of the lines represents the sender's SNR at the eavesdropper, in the absence of jamming. The bold lines show the theoretical BER for a hypothetical eavesdropper who gets a noiseless signal (infinite SNR) from the sender.

The experiment reveals the following:

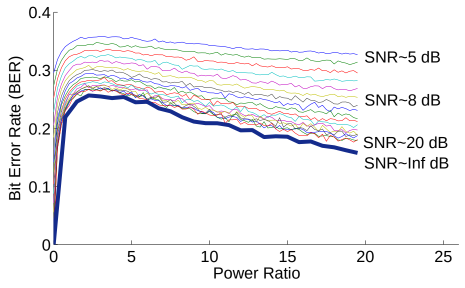
- First, the BER at the eavesdropper follows the theoretical predictions from §5.1. The BER is low when the ratio is either too high or too low. Further, the BER is at or close to its maximum when the ratio, $\frac{P_{j\to e}}{P_{s\to e}}$, is between 1 and 9,



(a) 4QAM



(b) 16QAM



(c) 64QAM

Figure 7—Effectiveness of Jamming at eavesdropper. For all modulation schemes over OFDM, the Bit Error Rate (BER) is maximized when the power ratio is between 1 and 9.

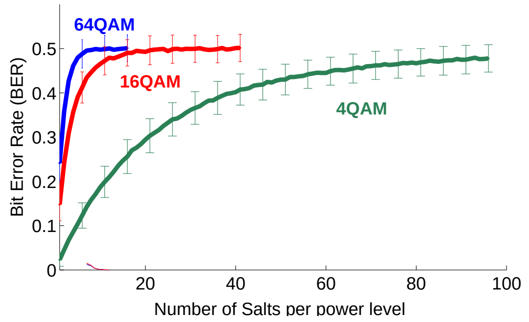


Figure 8—Error Magnification. For every modulation, iJam can magnify the Bit Error Rate (BER) to 50% by picking an appropriate value of M.

and this works independent of the modulation used.[7]

- Second, eavesdroppers that receive the sender signal at low SNR have a much higher BER than the theoretical BER of an eavesdropper with an infinite sender SNR. This behavior is expected since channel noise decreases the eavesdropper's overall ability to decode and hence aligns with the jammer's goal.

8.1.2 *How well can iJam magnify BER?*

In §5.3, we provided a mechanism that allows iJam to magnify the BER to close to 50%. In particular, at each power level, the sender transmits M salts and amplifies the BER by XOR-ing these salts. Here, we evaluate how iJam's BER amplification performs with different modulations. Since, the value of M should be sufficient to magnify the BER to 50% in all cases, we consider the worst-case eavesdropper for this experiment, *i.e.*, an eavesdropper that has the lowest BER in the absence of jamming. From Fig. 7, this corresponds to locations where the eavesdropper has a high SNR from the sender, and where the power ratio is either of the extremes, i.e., 1 or 9. Thus, to find M, we only consider eavesdropper locations that satisfy these constraints. In each experiment, the sender transmits 10000 salts. Fig. 8 shows the value of the BER as a function of M for different modulations. The experiment reveals the following.

- For all modulation schemes, iJam can magnify the BER to 50% by picking an appropriate value for M.
- At 16-QAM and 64-QAM, iJam requires about 15 to 30 salts to achieve a BER of 50%, while 4-QAM needs about 90 salts to approach the same BER. This is because, for a given power ratio, the original BER for 16-QAM and 64-QAM is much higher than 4-QAM, and hence it takes only a few transmissions to achieve a cumulative 50% BER. In contrast, since the initial BER with 4QAM is low, it requires a larger number of salts to get to 50%.
- The total time for iJam to transmit a salt across all power levels is small. Specifically, iJam needs M (15 for 64-QAM, 30 for 16-QAM, 90 for 4-QAM) salts per power level, and 10 power levels in total. Since each salt is small (512 bits), multiple salts can fit in a single 1500 byte 802.11 packet. Thus, iJam requires around 7 packets for 64-QAM, 14 packets for 16-QAM and 42 packets at 4-QAM. Even including additional MAC overheads such as DIFS, contention window *etc.*, each packet takes less than 1ms to deliver. Thus, a 512 bit salt can be delivered within 7-42 ms.

[7]We also run the experiments with soft combining at the adversary. However, this approach produces a higher eavesdropper BER than the optimal hypothesis testing adversary.

- As noted earlier, iJam can be combined with existing 802.11 security protocols to protect the initial packet handshake leading the key exchange, and hence defeat recent successful attacks on 802.11 [2]. Given the above results, iJam can be used to protect both the *ANonce* and *SNonce* exchanged in the 802.11 handshake in 14-84 ms.

8.1.3 *Testbed BER*

Finally, in this section, we use a representative indoor testbed as a case study to investigate the BER achieved by iJam for eavesdroppers at different locations. The testbed has 20 nodes that form a variety of line-of-sight and non-line-of-sight topologies. In this section, we experiment with various locations irrespective of the power ratios. Specifically, we randomly pick two nodes from the testbed to be Alice and Bob. All the other nodes eavesdrop on the channel.

We run the complete protocol from §5.5 to evaluate iJam's BER. Alice and Bob want to establish a key. Alice transmits to Bob L sequences of M salts, and Bob jams each of these L sequences with a different power level. Similarly, Bob transmits to Alice L sequences of M salts, and Alice jams each of these L sequences with a different power level. The final key is generated by XOR-ing all $2ML$ salts together. We calculate the BER in the random key generated.

We repeat the experiment with random Alice-Bob pairs. For each Alice-Bob location pair, we run all three modulation schemes: 4-QAM, 16-QAM and 64-QAM over OFDM. For each modulation, the number of salts at each power level, M, is set to the value from the previous section.

Fig. 9 plots a CDF of BER in the key as decoded by the eavesdropper. The CDF is taken across all the eavesdropper locations and different modulations. The figure shows that in our testbed, iJam provides a median BER of 50% which is as good as randomly guessing the bits in the packet. Further, the CDF is tightly concentrated around the median, *i.e.*, the BER of almost all eavesdroppers in the testbed is between 40-60%. Thus, iJam can ensure that an eavesdropper cannot decode the key.

8.2 Effectiveness of Receiver

8.2.1 *Can an iJam receiver decode while jamming?*

In this result, we show that an iJam receiver can still decode in the presence of the jammer. Specifically, we show that iJam's algorithm to combine OFDM samples from consecutive repeated symbols to produce a single symbol works. The standard performance metric for a receiver is the BER

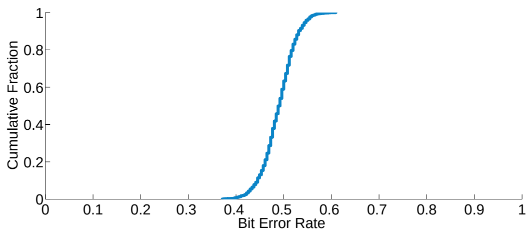


Figure 9—Bit Error Rate (BER) for the Whole Testbed. The figure shows a CDF of the BER in our testbed at different eavesdropper locations. BERs of almost all eavesdroppers is between 40-60%.

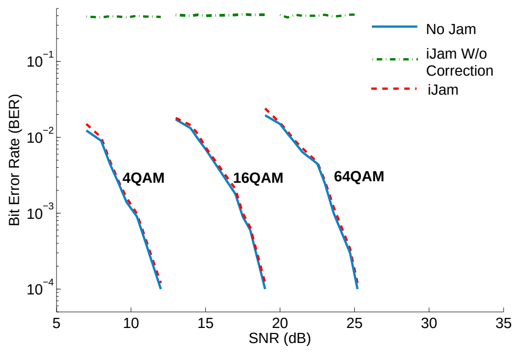


Figure 10—Comparison of Receiver Bit Error Rate (BER). The figure shows that, for all modulations and SNRs, the BER at the intended receiver, in the presence of jamming is similar to that without jamming. The figure also shows that phase correction is critical for iJam to work.

as a function of the SNR [24, 26], and we check whether the receiver in the presence of jamming can match the BER of a jammer-free receiver at every SNR, and for every modulation.

As before, we pick random node pairs in the testbed to act as a sender and a receiver. For each pair of nodes, the sender transmits packets to the receiver using different modulations. The receiver transmits its jamming signal at maximum power as this is the worst case scenario for decoding. We plot the BER as a function of the SNR from the sender to the receiver in the absence of jamming. The results do not change much when the receiver uses lower power levels for jamming.

Fig 10 shows the BER as a function of the SNR for three scheme: 1) jammer-free receptions, 2) iJam without the phase correction algorithm from §6, and finally 3) iJam with phase correction. The figure shows the following:

- Phase correction is crucial for iJam to work. Fig 10 shows that the chunks are completely undecodable when phase correction is not employed. This should not come as a surprise because even a traditional OFDM receiver that uses a single OFDM symbol has to apply phase correction on the received samples. In the absence of phase correction, the phases of most samples are incorrect and therefore the OFDM receiver gets most of the bits wrong.

- For all modulation schemes, the SNR at a iJam receiver is similar both with and without jamming. This shows that iJam can remove the effect of jamming precisely at the receiver, and does not affect the ability of the receiver to decode across the entire range of SNRs, even while the eavesdropper experiences a BER of around 50%.

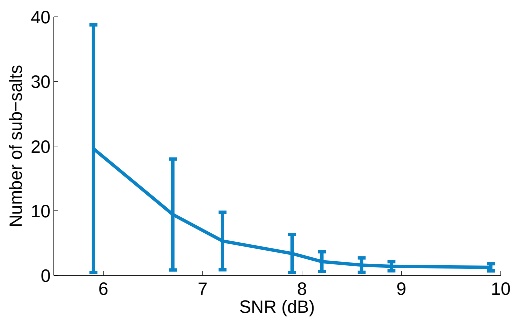


Figure 11—Number of sub-salt transmissions required at BPSK SNRs. The figure shows the number of sub-salts the sender transmits before the receiver at BPSK SNRs can correctly receive one sub-salt.

8.2.2 *Does iJam work at BPSK SNRs?*

Finally, because of its very low BER, iJam does not use BPSK over OFDM. iJam instead uses the sub-salts mechanism in §5.4 to deliver salts to locations which traditionally require BPSK because of their SNR. In this section, we evaluate if iJam can indeed do so. To test this, we consider pairs of nodes in the testbed which require BPSK over OFDM to communicate. For each pair, the sender transmits using 4QAM over OFDM. To counter the high BER that results from operating 4QAM at BPSK SNRs, the iJam sender divides salts into sub-salts of length 128 bits and sends a CRC for each sub-salt. The sender transmits sub-salts until the receiver receives one sub-salt that passes the CRC.

Fig. 11 plots the average number of different sub-salts the sender transmits before the receiver correctly receives one of them. The x axis shows the SNR values of the sender signal at the receiver. The SNR values span the typical BPSK operational regime (5-10dB) [9]. The figure shows that as the SNR decreases, the sender needs to send more sub-salts before the receiver can correctly receive one of them. While the number of such sub-salts is higher at lower SNRs, this is an acceptable overhead for the BPSK SNR locations. This is because the communicating nodes can use iJam only for protecting the key exchange which occur once at the beginning of the session. Once a key is established, it can be used to encrypt further packets using shared-key cryptography. The key point is that iJam can use 4-QAM to confidentially deliver salts even to receivers which traditionally require BPSK.

9 Discussion

The adversary model in §3, assumes that the adversary does not have either directional antennas or more antennas than either the client or the AP. In this section, we discuss how iJam can be extended to work without these constraints.

First, an adversary can deploy directional antennas to boast the power it receives from the transmitter. To overcome this, a iJam sender transmits data at a lower power to counteract the directional power gain at the adversary by increasing the ratio of jammer power to sender power. Reducing the power at the transmitter, however, comes with a tradeoff. In particular, the receiver now gets a lower powered signal from the transmitter. If the receiver is also equipped with a directional antenna, reducing the power at the sender does not affect the rate at the receiver because the receiver can counterbalance the power reduction by the directional gain. However, if the receiver does not have a directional antenna (even though the eavesdropper still does), the rate at which packets can be exchanged with the receiver reduces. However, this may still be acceptable because iJam is used only at the beginning of the session to protect the key exchange packets.

Next, say the eavesdropper has more antennas than either the AP or the client. As a result, it gets additional combinations of the transmitted and jamming signals. An adversary who knows the multi-tap channel from the receiver/jammer to itself can then use these additional combinations to separate the transmission signal from the jamming signal. However, since a iJam jammer does not send a preamble (known signal) while jamming, the adversary cannot determine this multitap channel from the jamming signal. However, the adversary will be able to leverage the fact that the receiver/jammer also transmits usual data packets at other times, which it can use to estimate the channel. To prevent this, all iJam nodes apply a random multi-tap filter, unknown to any other party, on its data transmissions, but not on its jamming signal. As a result, the adversary now sees a completely different multi-tap channel on data packets and jamming signals, and hence cannot use the channel estimate from data transmissions to subtract the jamming signal. Note that receivers can still continue to decode data packets even though the sender applies a random multi-tap channel, because this multi-tap channel can just be estimated using standard OFDM algorithms. We defer the evaluation of these extensions for future work.

10 CONCLUSION

This paper presents iJam, a PHY technique that allows a sender-receiver pair to protect sensitive packets that are intended to be heard only by the receiver. It does this by strategically jamming the signal so as to prevent an eavesdropper from decoding the data, while allowing the intended receiver to decode it. The paper also provides an prototype implementation of iJam showing its practical feasibility. While the focus of this paper is to protect sensitive packets such as the 802.11 key exchange packets, we believe that the mechanism introduced in this paper open the door for other techniques that hide every packet sent on the wireless medium, instead of just the sensitive ones. This would allow us to convert wireless into a truly untappable medium.

REFERENCES

[1] Aircrack: www.aircrack-ng.org.

[2] Cracking WEP and WPA wireless networks. docs.alkaloid .net/index.php/ Cracking WEP and WPA Wireless Networks #WPA Flavours.

[3] What no one is saying about WPA2 security. blogs.computer world.com/14638/ whatnooneissayingaboutwpa2security.

[4] A. Arora and L. Sang. Dialog codes for Secure Wireless Communications. In *IPSN*, 2009.

[5] B. Azimi-Sadjadi, A. Kiayias, A. Mercado, and B. Yener. Robust Key Generation from Signal Envelopes in Wireless Networks. In *ACM CCS*, 2007.

[6] M. Brodsky and R. Morris. In defense of Wireless Carrier Sense. In *Sigcomm*, 2009.

[7] L. Dong, Z. Han, A. P. Petropulu, and H. V. Poor. Cooperative jamming for wireless physical layer security. *ARXIV:0907.4996*, 2009.

[8] A. Dutta, D. Saha, D. Grunwald, and D. Sicker. SMACK - A Smart Acknowledgment Scheme for Broadcast Messages in Wireless Networks. In *Sigcomm*, 2009.

[9] F. Edalat. Real-time Sub-carrier Adaptive Modulation and Coding in Wideband OFDM wireless Systems. *MIT*, 2008.

[10] F M Halvorsen and Olav Haugen and Martin Eian and Stig Mjolsnes. An improved Attack on TKIP. In *LNCS,2009*.

[11] S. Gollakota and D. Katabi. ZigZag Decoding: Combating Hidden Terminals in Wireless Networks. In *Sigcomm*, 2008.

[12] X. He and A. Yener. Secure communication with a byzantine relay. In *Proc. of ISIT 2009*.

[13] J. Heiskala and J. Terry. *OFDM Wireless LANs: A Theoretical and Practical Guide*. 2001.

[14] I.Csiszar and J. Korner. Broadcast channels with confi dential messages. *IEEE Trans. on Info. Theory*, 1978.

[15] E. Inc. Universal software radio peripheral. http://ettus.com.

[16] K. Jameison and H. Balakrishnan. PPR: Partial Packet Recovery for Wireless Networks. In *Sigcomm*, 2007.

[17] S. Jana, M. Clark, S. Kasera, N. Patwari, and S. Krishnamurthy. On the Effectiveness of Secret Key Extraction from Wireless Signal Strength in Real Environment. In *ACM MobiCom*, 2009.

[18] S. Katti, S. Gollakota, and D. Katabi. Embracing Wireless Interference: Analog Network Coding. In *Sigcomm*, 2007.

[19] L. Lai and H. E. Gamal. The relay-eavesdropper channel:Cooperation for secrecy. *IEEE Trans. on Info. Theory*, 2008.

[20] Z. Li, W. Xu, R. Miller, and W. Trappe. Securing Wireless Systems via Lower Layer Enforcements. In *WiSe*, 2006.

[21] S. Mathur, W. Trappe, N. Mandayam, C. Ye, and A. Reznik. Radio-telepathy: Extracting a Secret Key from an Unauthenticated Wireless Channel. In *MobiCom*, 2008.

[22] R. Negi and S. Goel. Secret communication using artifi cial noise. *Proc. Vehic. Tech Conf*, 2005.

[23] M. Strasser, B. Danev, and S. Capkun. Detection of reactive jamming in sensor networks. *ACM TOSN*, 2010.

[24] J. K. Tan. An Adaptive Orthogonal Frequency Division Multiplexing Baseband Modem for Wideband Wireless Channels. Master's thesis, MIT, 2006.

[25] A. Thangaraj, S. Dihidar, A. R. Calderband, S. W. McLaughlin, and K. M. Merolla. Capacity achieving codes for the wiretap channel with applications to quantum key distribution. *CoRR*, 2004.

[26] D. Tse and P. Vishwanath. *Fundamentals of Wireless Communications*. Cambridge University Press, 2005.

[27] A. Wyner. The wire-tap channel. *Bell Tech Journal*, 1975.

[28] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless

networks. *MobiHoc*, 2005.
[29] Y Shaked and A Wool. Cracking the Bluetooth PIN. In *Mobisys,2005*.