This paper was presented as part of the main technical program at IEEE INFOCOM 2011

Physical Layer Wireless Security Made Fast and Channel Independent

Shyamnath Gollakota and Dina Katabi
Massachusetts Institute of Technology

Abstract – **There is a growing interest in physical layer security. Recent work has demonstrated that wireless devices can generate a shared secret key by exploiting variations in their channel. The rate at which the secret bits are generated, however, depends heavily on how fast the channel changes. As a result, existing schemes have a low secrecy rate and are mainly applicable to mobile environments. In contrast, this paper presents a new physical-layer approach to secret key generation that is both fast and independent of channel variations. Our approach makes a receiver jam the signal in a manner that still allows it to decode the data, yet prevents other nodes from decoding. Results from a testbed implementation show that our method is significantly faster and more accurate than state of the art physical-layer secret key generation protocols. Specifically, while past work generates up to 44 secret bits/s with a 4% bit disagreement between the two devices, our design has a secrecy rate of 3–18 Kb/s with 0% bit disagreement.**

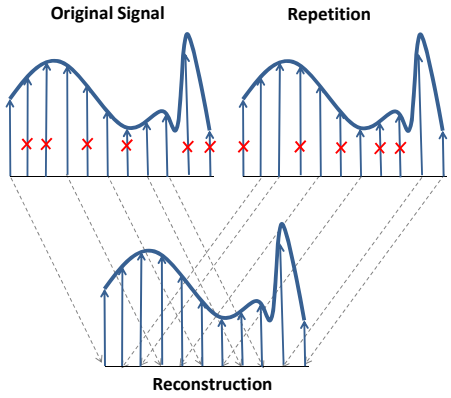


Figure 1—iJam at work. The sender repeats its transmission. The receiver-cum-jammer randomly jams complimentary samples in the original signal and its repetition. To decode, the receiver-cum-jammer, stitches together unjammed samples to create a clean symbol.

1 Introduction

Physical layer security enables two wireless nodes to exchange secret data in the presence of an eavesdropper, without encryption [24]. It is an information-theoretic construct that exploits randomness at the wireless physical layer and does not require computational hardness [12]. It may be used to replace encryption when the communicating devices lack the computational resources for prime number generation (e.g., sensors [21, 23]), or to generate a continuous stream of fresh secret keys that strengthen existing cryptographic protocols [28, 12]. Physical layer security is rooted in Shannon's work on perfect secrecy [24]. It has experienced a renaissance in recent years with a plethora of new theoretical results that characterize secrecy capacity [5, 9, 30], develop codes for secure communications [18, 26], and exploit channel variations across time, space, and users for higher information rates. These theoretical advances have culminated with the emergence of practical systems, where wireless devices have been empirically shown to use the characteristics of their wireless channel to generate a secret key in the presence of an eavesdropper [21, 17, 6].

Existing practical systems however are highly limited in the rate at which they generate secret bits. Today, the highest empirically achieved secrecy rate is only 44 bits/s [6]. Further, achieving this rate requires mobility and incurs 4% average bit disagreement between communicating nodes. The low secrecy rate is because existing schemes extract secret bits from the channel variations, and hence cannot generate new secret bits unless the channel changes. In fact, experimental results show that in static scenarios, the extracted bits have very low entropy making them less suitable for a secret key [17].

In this paper, we investigate a new approach to physical layer security that is independent of channel variations, and thus works even when the channel is static. We introduce iJam, a channel-independent PHY technique that ensures that an eavesdropper cannot even demodulate a wireless signal not intended for it. We show that iJam achieves orders of magnitude higher secrecy rates than existing schemes with no bit disagreement.

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 the 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.

iJam builds on past theoretical work on cooperative jamming [22, 7]. Past work however typically separates the jammer from the receiver and hence requires an out-of-band channel to inform the receiver of the jamming signal [19, 13, 7]. In contrast, iJam presents a practical implementation of a receiver-cum-jammer, eliminating the need for out-of-band

channels and third party intervention. iJam also addresses the following practical challenges in using jamming for secret key extraction:

(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 [25]. iJam addresses this issue by exploiting the basic properties of OFDM transmissions. In §4, we show that in contrast to alternative transmission schemes (e.g., BPSK or QAM), 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 bit-error rate that is almost as high as a random guess.

(b) How do we ensure that we can jam an eavesdropper independent of its location? The effectiveness of jamming depends on the eavesdropper's location with respect to the sender and the jammer. If the eavesdropper is too close to the sender, the jamming power at the eavesdropper will be far lower than the power of the sender's transmitted signal, which may allow the eavesdropper to decode the sender's signal despite jamming. iJam addresses this problem using a two-way protocol with multiple jamming powers. Specifically, to generate a secret key shared between Alice and Bob, both nodes take turns in sending and jamming. Hence, no eavesdropper can be always closer to the sender than the jammer. The secret key is then constructed by XOR-ing the bits sent in the two directions. Further, the protocol runs multiple such iterations at different jamming powers that are strategically chosen to ensure robustness to eavesdropper location. We show in §5.2 that iJam ensures the eavesdropper, regardless of its location, gets no information about the key.

We implement iJam in GNURadio and evaluate it in a 20-node testbed of USRP2 radios [15] with 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 sender 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 fast and accurate. A typical 802.11 receiver can generate secret keys with zero disagreements, at a rate of 3–18Kb/s, depending on the modulation.

2 Related Work

iJam builds on prior schemes that demonstrate the practicality of secret key extraction from wireless transmissions [17, 21, 2, 20]. Say Alice and Bob want to establish a secret key. These schemes work by using the time-varying wireless channel from Alice to Bob. Assuming reciprocity, Alice and Bob can both derive this channel information using wireless transmissions. These schemes then extract secret bits from this time-varying channel information. However, in order to derive a number of uncorrelated bits, the channel has to change quickly. As a result, these schemes have a low secrecy rate and are mainly applicable to mobile environments. In contrast, iJam provides a channel-independent approach for secret key extraction, and hence is fast and can operate even in scenarios where the channel stays static for long intervals.

Jamming has traditionally been used in adversarial manner to prevent others from communicating over the wireless medium [29]. Recently, however, there has been interest in cooperative or friendly jamming. In [19, 13, 7], a trusted third party jams the secret key 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 secret key. The work in [22, 10] presents a variation on the above model where the sender itself transmits the key combined with a jamming signal. A third party node transmits an anti-jamming signal that cancels out the jamming signal at the receiver but not at the adversary. In contrast, iJam achieves security without a trusted third party, and is further implemented and evaluated in a testbed.

iJam is related to dialog codes [1], where the receiver jams the transmitted signal to flip specific bits in the packet as received by the eavesdropper. This approach assumes a modulation scheme where each bit is sent separately on the channel, e.g., BPSK. Past work however shows that jammed bits are easy to identify in such modulation schemes [25]. Additionally, flipping specific bits requires phase synchronization of the sender 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.

3 Adversary Model

We assume the adversary can listen to all communications in the network. It can also measure the channels between itself and the communicating nodes. Further, it can be anywhere in the network and is free to move or stay static.

The adversary can operate at the packet, bit, or signal levels. For example, it 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 them.

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. This approach however does not work because basic results in multiuser information

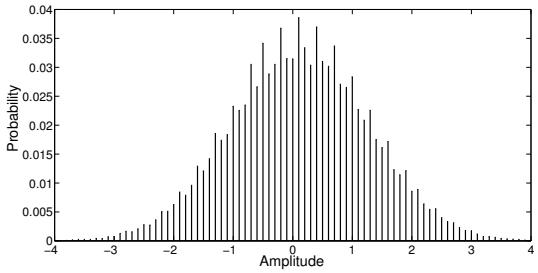


Figure 2—Amplitude Distribution of OFDM signal samples. The OFDM uses a 64-point FFT and modulates bits using 4QAM. The distribution follows a zero-mean Gaussian.

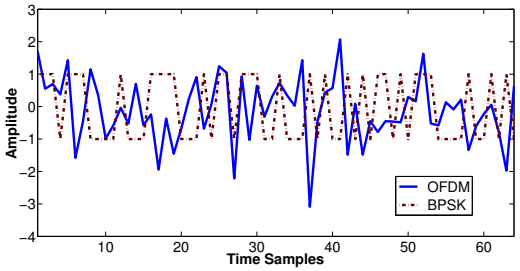


Figure 3—Time domain signal samples for BPSK and OFDM. In contrast to BPSK, the OFDM signal spans a wide range of values.

theory say that decoding multiple signals is impossible if the total information rate is outside the capacity region [27]. In iJam, we ensure that the information rate at the eavesdropper exceeds the capacity region by making the jammer transmit at an excessively high rate. This can be 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 to existing 802.11 bit rates, which use a maximum of 64 QAM, this is an excessively high bit rate.

The secrecy bit rates reported in this paper are for the case where the eavesdropper's hardware is as powerful as that of the sender and receiver. The ideas underlying iJam, however, can be applied in certain scenarios where the eavesdropper's hardware may be more powerful than that of the sender and the receiver, albeit with lower secrecy rates. For example, the eavesdropper may use directional antennas to obtain a power gain which reduces the secrecy rate. But the basic ideas underlying iJam can still help exchange secret keys between the sender and the receiver. Evaluating these scenarios, however, is beyond the scope of this paper.

Finally, similarly to all practical past work on physical layer wireless security we assume the adversary is passive and not interested in mounting a man-in-the middle attack [21, 17, 6]. There is a growing literature on authenticating wireless devices based on their location or their physical properties [3], which can be used to address such attacks.

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.[1] 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 [27]. 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 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.

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 single 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.

5 iJam

iJam is a PHY-layer technique that enables two wireless nodes to exchange an unencrypted secret key, in the presence of an eavesdropper. Without loss of generality, we focus on ex-

[1]The transmitter also appends a cyclic prefix [14].

changing a secret key of B bits (the default is $B = 512$ bits). Larger keys can be obtained by repeating the process multiple times. Also, while iJam is a general secrecy technique, we focus our description on 802.11.

iJam works as follows. The sender generates a random sequence of B bits, which we refer to as a **salt**. It delivers the salt to its PHY for transmission, along with the standard packet header. The PHY generates the OFDM signal corresponding to the packet. However, for each OFDM symbol corresponding to the salt, the PHY sends 2 copies of the symbol back-to-back.

The PHY layer at the receiver starts by decoding the packet's header. If the header is marked to indicate an 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.[2] 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.[3]

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. If the bits pass the checksum, the receiver 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 the secret key. In the following few sections we expand this basic idea to make it robust to various adversarial scenarios.

5.1 The Adversary's Optimal Strategy for Detecting Jammed Samples

To make iJam robust, we need to ensure that an eavesdropper cannot distinguish jammed samples from clean 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. However, since iJam repeats each sample, an eavesdropper has additional information: it can compare an OFDM sample against 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 [27]. Therefore, jammed samples have higher variance than clean samples. An eavesdropper can exploit this fact to improve its ability to identify jammed samples.

[2]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.

[3]The receiver knows the modulation of the packet from the header. Given a particular modulation, the variance of the OFDM signal can be pre-computed.

Specifically, the eavesdropper can apply an optimal hypothesis testing strategy as follows. Consider 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 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 events corresponding to H_1 and H_2, we get:

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

Thus, the optimal hypothesis testing reduces to the following:

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

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 optimal estimate of the salt.

Say that the eavesdropper applies the above optimal strategy, how well does she perform? Let us compute an upper bound on the performance of such eavesdropper. To do so, we simulate in Matlab the case where the eavesdropper receives the transmitted signal 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 and use the optimal hypothesis test to estimate the salt. Fig. 4 plots the bit error rate as a function of the ratio of the jamming power to the sender's power at the eavesdropper. The figure shows 4 lines, one for each modulation scheme. The figure reveals three important points:

- Jamming can produce high BERs at the eavesdropper for 4-QAM, 16-QAM, and 64-QAM. However, to ensure a BER as high as a random guess, i.e., a BER of 50%, iJam needs an additional mechanism that amplifies the BER at the adversary.
- The BER is close to the maximum when the sender's power at the eavesdropper, $P_{s\to e}$, and the jammer's power at the eavesdropper, $P_{j\to e}$ satisfy the relationship $1 < \frac{P_{j\to e}}{P_{s\to e}} < 9$. However, this condition may not be satisfied at the eavesdropper's location. Hence, 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.

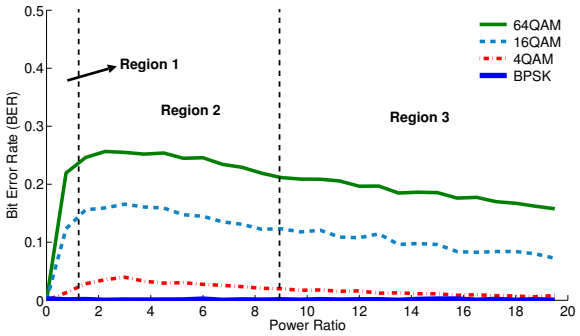


Figure 4—Performance of an optimal hypothesis-testing adversary. The figure shows the Bit Error Rate (BER) for different modulations as a function of the ratio of the jamming power to the transmitter power at the eavesdropper. The graph can be divided into three regions: Region 1 where the power from 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.

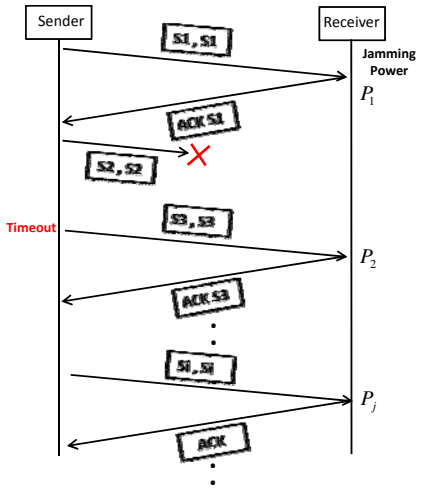


Figure 5—Making iJam location independent. The sender transmits a salt and its repetition and waits for acks. In the absence of an ack, the sender timeouts, discards the unacked salt and transmits a new random salt.

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 $1 < \frac{P_{j\to e}}{P_{s\to e}} < 9$. 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., $\frac{P_{j\to e}}{P_{s\to e}} \leq 1$.) 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, where $\frac{P_{j\to e}}{P_{s\to e}} \leq 1$. Specifically, an 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. Yet, the eavesdropper needs to xor both salts to get the key.

(b) Dealing with Region 3 (i.e., $\frac{P_{j\to e}}{P_{s\to e}} \geq 9$.) 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 obtain a low BER. So the solution to this problem is to reduce the jamming power so that the ratio $\frac{P_{j\to e}}{P_{s\to e}}$ stays relatively small. The problem however is that for any power value that the jammer picks, there exist eavesdropping locations for which the ratio $\frac{P_{j\to e}}{P_{s\to e}}$ is too high and other locations 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 using a power $P_2 = \frac{P_1}{9}$, and the third salt using a power $P_3 = \frac{P_2}{9}$, and so on until it jams the L^{th} salt 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 cannot 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, $1 \leq \frac{P_{j\to e}}{P_{s\to e}} \leq 9$.

We note the following two points:

- First, since the receiver may fail to decode a salt, we need the receiver to acknowledge every salt and the sender to continue sending salts until the receiver acknowledges L such salts. The key is then generated by xor-ing only the acked salts.
- 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 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.[4]

[4] 802.11 transmits around 15dBm and have a noise floor around -95dBm [4]. This translates to about 10-11 different power levels.

5.3 BER Magnification

Now that we have made iJam location independent, we address the next 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. Say, 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 $(1-x)^M$. This exponential trend enables us to quickly increase the BER at the eavesdropper to about 50%.

As in the previous section, if the receiver fails to decode a particular salt, it does not acknowledge the salt. Unacked salts are discarded, and the key is created by xor-ing only 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 its low BER. So 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 uses 4-QAM over OFDM. However, since 4QAM has a much higher BER 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, an iJam sender 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.

In our implementation a sub-salt is 128 bits. In §9.4, we show that using this value, 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 [8].

5.5 Summary

To summarize, say Alice and Bob want to exchange a secret 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, an 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.

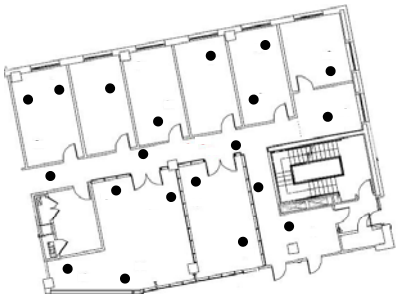


Figure 6—Locations of Testbed Nodes

6 Stitching Samples 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 the oscillators at the sender and receiver tend to have small differences that result in a frequency offset, Δf [14]. The frequency offset changes the phase of the received signal over time. In particular, the phase of the OFDM signal increases by $2\pi\Delta f$ every sample. Say that a transmitted OFDM symbol has the samples: $y_1, y_2, \cdots, y_N$. The frequency offset causes the symbol to be received as (after channel compensation):

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

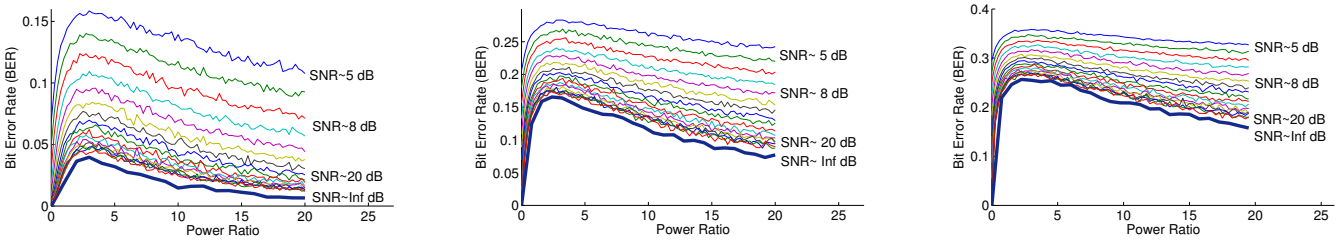
If the symbol repetition is D samples away from the original symbol, the repetition is received as:

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

By comparing the above two equations, it is clear that one cannot simply stitch samples from a symbol and its repetition without correcting for phase differences. We note however 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, using standard correlation techniques [11]. After correcting for this phase, the iJam receiver can combine samples across a symbol and its repetition. The resulting signal now looks like a proper OFDM signal which it can be decoded by a standard OFDM decoder.

7 Practical Jamming

An iJam 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. iJam however does not require such synchronization. Since an OFDM symbol and its repetition are sent back-to-back, all that an iJam jammer needs to know is the boundaries between OFDM symbols. Symbol boundaries are naturally detected by existing OFDM decoding algorithms. Once it locates symbol boundaries, iJam can pair a sample with its repetition because they are separated by the duration of a symbol.



(a) 4QAM (b) 16QAM (c) 64QAM

Figure 7—Effectiveness of Jamming at eavesdropper. For all modulation schemes, the BER is maximized when the power ratio is between 1 and 9.

8 Implementation

We implement iJam using USRP2. We use the RFX2400 daughterboards which operate in the 2.4 GHz range. We build our prototype on top of the GNU Radio software. We use an 802.11-like physical layer, with 64 OFDM sub-carriers. We implement a receiver-cum-jammer by keeping both the transmit and receive chains running and connected to the antenna for the duration of each 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.[5]

9 Results

We evaluate iJam in a representative indoor testbed. The testbed has 20 nodes in both line-of-sight and non-line-of-sight locations, as shown in Fig. 6.

9.1 Do empirical results match analysis and simulation?

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 in the testbed, 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 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 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 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 power ratio, $\frac{P_{j\to e}}{P_{s\to e}}$. 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 figure 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, and this works independent of the modulation used.
- Second, interestingly the adversary's measured bit error rates (the thin lines in Fig. 7) are larger than the simulated/analytical BERs (the thick lines in Fig. 7). The reason is that the analysis/simulation ignores channel noise which increases the BER created by jamming and improves iJams ability to prevent the attacker from decoding.

[5]Note that a receiver-cum-jammer does not mean that the wireless radio can transmit and receive concurrently. In particular, 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.

9.2 How well does iJam magnify the adversary's BER?

In §5.3, we provided a mechanism that allows iJam to magnify the adversary's BER 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 most powerful eavesdropper, *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 compute the maximum M, we only consider these eavesdropper locations. In each experiment, the sender transmits 10000 salts. Fig. 8 shows the value of the BER as a function of M. It 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 expected because, the original BER for 16-QAM and 64-QAM is much higher than 4-QAM, and hence it takes more transmissions with 4-QAM to achieve the same BER.
- 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 512 bits, multiple salts can fit in a single 1500 byte 802.11 packet. Thus, iJam requires around 14 packets for 64-QAM, 28 packets for 16-QAM and 84 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 14-84 ms. Since the final secret key is generated by XOR-ing the

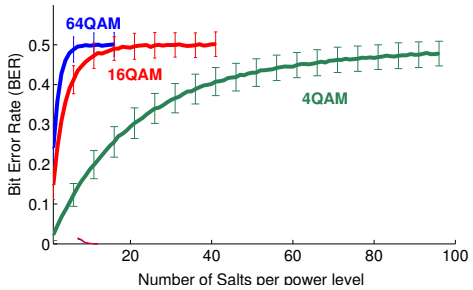


Figure 8—Error Magnification. For every modulation, iJam can magnify the Bit Error Rate (BER) to 50% by picking an appropriate value of M.

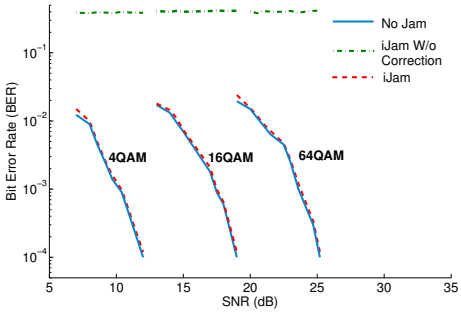


Figure 9—Comparison of Receiver Bit Error Rate (BER). The figure shows that, for all modulations and SNRs, the BER at the intended receiver, in the presence of jamming is similar to that without jamming. The figure also shows that phase correction is critical for iJam to work.

salts in the two directions, iJam can generate a 512 bit secret key in 28-168ms. This corresponds to a secrecy rate of 3 to 18 Kb/s.

9.3 Can an iJam receiver decode while jamming?

Here we show that iJam's algorithm for stitching samples across a symbol and its repetition works. Specifically, we check that the receiver in the presence of jamming can match the BER of a jamming-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 using different modulations. The receiver transmits its jamming signal at maximum power as this is the worst case scenario for decoding.

Fig 9 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 9 shows that the symbols are completely undecodable when phase correction is not employed. This should not come as a surprise because in the absence of phase correction, the phases of the 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 accurately decode across the entire range of SNRs, even while the eavesdropper experiences a BER of around 50%.

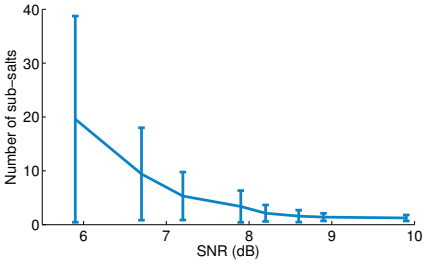


Figure 10—Number of sub-salt transmissions required at BPSK SNRs. The figure shows the number of sub-salts the sender transmits before the receiver at BPSK SNRs can correctly receive one sub-salt.

9.4 Does iJam work at BPSK SNRs?

As mentioned earlier, 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. Here, we check 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. 10 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) [8]. 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. The key point to note, however, is that iJam can use 4-QAM to confidentially deliver salts even to receivers which traditionally require BPSK.

9.5 Aggregate Results from the Testbed

Finally, in this section, we use the representative indoor testbed shown in Fig. 6 as a case study to investigate iJam's BER at an eavesdropper at various locations. Specifically, we randomly pick two nodes from the testbed to be Alice and Bob and run the complete protocol from §5.5. All the other nodes eavesdrop on the channel. For each Alice-Bob location pair, we run all three modulation schemes: 4-QAM, 16-QAM and 64-QAM over OFDM.

Fig. 11 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

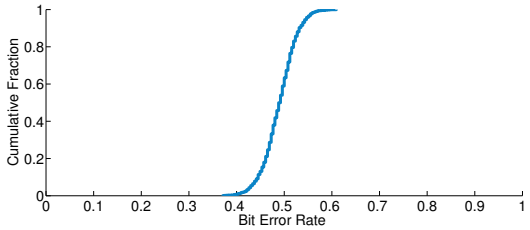


Figure 11—Eavesdropper Bit Error Rate (BER) for the whole Testbed. The median BER is 50% and the BER range is 40-60%, which shows that the eavesdropper's performance is close to a random guess.

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 secret key.

10 Conclusion

This paper presents iJam, a novel PHY technique that enables two wireless devices to communicate secret bits, in the presence of an eavesdropper, and without encryption. iJam works by strategically jamming the transmission so as to prevent an eavesdropper from getting any information about the secret key, while allowing only the intended receiver to decode the key accurately. We build a prototype of our design and show that it can provide orders of magnitude higher secrecy rates than existing schemes.

References

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

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

[3] V. Brik, S. Banerjee, M. Gruteser, and S. Oh. Wireless Device Identification with Radiometric Signatures. In *Mobicom 2008*.

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

[5] R. Bustin, R. Liu, H. V. Poor, and S. Shamai. An mmse approach to the secrecy capacity of mimo gaussian wiretap channel. *JWCN*, 2009.

[6] J. Croft, N. Patwari, and S. Kasera. Robust uncorrelated bit extraction methodologies for wireless sensors. *IPSN 2010*.

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

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

[9] E. Ekrem and S. Ulukus. Secrecy capacity of a class of broadcast channels with an eavesdropper. *JWCN*, 2009.

[10] S. Goel and R. Negi. Guaranteeing Secrecy using Artificial Noise. *IEEE Trans. on Wireless Comm.*, 2008.

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

[12] Z. Hao, S. Zhong, and L. E. Li. Towards wireless security without computational assumptions. *Buffalo*, 2006.

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

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

[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] M. Kobayashi, M. Debbah, and S. Shamai. Secured communication over frequency-selective fading channels: A practical vandermonde precoding. *JWCN*, 2009.

[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 artificial noise. *Proc. Vehic. Tech Conf*, 2005.

[23] A. Sayeed and A. Perrig. Secure wireless communications: Secret keys through multipath. *CMU*.

[24] C. Shannon. Communication theory of secrecy systems. *Bell Labs*, 1949.

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

[26] 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.

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

[28] S. Xiao, W. Gong, and D. Towsley. Secure wireless communication with dynamic secrets. In *Infocom 2010*.

[29] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. *MobiHoc*, 2005.

[30] R. Y. Z Li and W. Trappe. Secure communication with a fading eavesdropper channel. In *ISIT 2007*.