All Jelly No Jam
Mitigating Interleaving Targeted Jamming of IEEE802.11

Benjamin Davis
bdavis41@calpoly.edu
California Polytechnic State University
San Luis Obispo, California

Bruce DeBruhl
bdebruhl@calpoly.edu
California Polytechnic State University
San Luis Obispo, California

ABSTRACT
Recent jamming attacks against IEEE 802.11 (WiFi) have targeted the interleaving mechanism to greatly improve the power efficiency of such attacks, getting as good as 95% packet loss at 0.1% power of the transmitter. This paper proposes a novel method of modifying the interleaver to prevent this attack vector through using a shared secret to deterministically interleave the data in such a way that is not susceptible to targeted sub-carrier jamming. We implement and test this approach using software defined radio. We show it entirely negates the impact of interleaving targeted jamming, without making the signal more susceptible to more traditional jamming methods or additive Gaussian white noise. We show that our modified interleaver makes these attacks less effective than whole-channel jamming by as much as 15% lower packet loss.

CCS CONCEPTS
• **Security and privacy → Mobile and wireless security**; *Denial-of-service attacks*; • **Networks** → Network reliability;

KEYWORDS
WiFi, Jamming, Interleaver

1 INTRODUCTION
Wireless transmission technology is becoming increasingly more popular for a wide variety of applications. What at one point was nearly explicitly used for the transmission of data for defense applications has fully permeated into civilian day-to-day life. Technologies such as GPS, cellular data, and WiFi have become so common that many people could not imagine life without them. However, the very nature of wireless data transfer exposes them to host of vulnerabilities not practical in wired transmission. Chief concerns among these are data leakage, data spoofing, and denial-of-service (DoS) attacks. DoS attacks are particularly concerning because they are able to bring an entire data channel down with minimal effort. In the early 2010's a series of cellular and GPS jamming attacks in the wild prompted the FCC to further its own education and enforcement efforts against jammers[1]. Some companies have even been fined when it was determined they were deliberately jamming particular wireless networks[2].

ACSAC '19, Dec. 09–13, 2019, San Juan, Puerto Rico
© 2019 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Recently, jamming attacks against IEEE 802.11 (WiFi) have been a prominent area of research. This is likely because of the ubiquity and prevalence of WiFi in a connected world. Among these are attacks which target the preamble of WiFi and the Pilot channels to strong effect. Recently, Vo-Huu et al. published an attack paper which describes highly efficient jamming using the static nature of the interleaving mechanism of 802.11 to achieve 95% packet loss at 0.1% of the transmitter power(see Section 2.1)[21]. Interleaving addresses two problems of RF data transfer: bursty errors and channel bias. Bursty error occur when noise naturally manifests itself across a frequency and forces the data on that frequency to become invalid. Channel bias occurs through some mediums and in some environments where a RF signal might be naturally forced towards a specific value. This can in turn cause a sufficient number of bit errors to overwhelm any error correction coding.

In their paper Vo-Huu et al. show that they are able to obtain 100% packet error rate with a signal-to-jamming ratio of only 1%. This attack is particularly concerning because it can be easily implemented on inexpensive hardware at a high power to impact a wide area. Additionally, while the attack is conducted on WiFi it is not the only wireless data transfer protocol with a static interleaver meaning there are other systems which are vulnerable to this attack vector as well. Notable examples being LoRa and GSM Voice's use of a diagonal interleaver as data-link being employed in military applications[16]. Although there already exist a variety of anti-jamming techniques for use in modern RF data transfer. To the best of our knowledge none of these were designed with interleaver jamming in mind and many of them appear to be susceptible to interleaving targeted jamming.

In this paper we propose and test Jelly interleaving, a novel interleaving mechanism designed to mitigate the impact of interleaving jamming without negatively impacting speed or noise resistance. In our protocol, we seek to establish a shared secret and use it to construct an interleaving mechanism which distributes adjacent bits in a non-predictable pattern. Through testing we are able to show that our modifications completely negate this particular jamming attack, in fact reducing its effect so this attack is less power efficient than whole-channel jamming by 15%.

The remainder of the paper is structured as follows. In Section 2, we discuss the underlying technologies used in the attack, previous work on the subject, and technologies used in our mitigation design. In Section 3, we discuss the design of our modified interleaver and perform a analytic analysis of its security. We then discuss our testing setup in Section 4 and present our results in Section

5. Finally, we discuss further work on the topic and discuss the implications of our results in Sections 6 and 7 respectively.

2 BACKGROUND

In this section, we provide summaries of the critical technologies and techniques necessary for understanding the problem and our proposed solution.

2.1 Jamming Techniques

Jamming is simply the process of creating channel conditions which prevent RF transmission across the channel. Recently a host of techniques have emerged which provide novel methods of jamming. In this section we will briefly cover some of the most notable jamming attacks, including interleaving jamming.

The simplest jamming method is whole-channel jamming, also referred to as a constant jammer[14]. The principle behind this attack being if the channel is flooded with powerful enough noise it is possible to force the DSCs to exhibit too much noise for the error correction encoding to recover from. This attack does tend to be power hungry making it a less than ideal attack.

Another attack which particularly targets OFDM has been shown by Clancy[10]. This is a pair of attacks which target the pilot signals of the transmission. By deliberately jamming these signals it has been shown that the channel equalization model can be sufficiently disrupted to cause packet loss. A second related attack is proposed by the same author. In this attack rather than jamming the Pilot channels the attacker attempts to drive them as close to zero energy as possible. This disrupts the channel with a high level of efficiency, up to 4dB signal-to-jamming ratio (SJR).

For OFDM to function properly it is critical that the signals are synchronized in respect to time and frequency. To capitalize on this fact methods of jamming which disrupt this have been proposed[17, 19]. In WiFi this synchronization occurs during the preamble earning these attacks the nickname of preamble attacks. Because these attacks do not need to spend energy jamming the entire message packet this provides an efficient method of attacking OFDM.

The final jamming attack which we discuss is the one we are seeking to mitigate through our work. Vo-Huu et al. proposed a jamming method which targets the DSC of 802.11 based on the static nature of the interleaving mechanism which we further discuss in section 2.3 [21]. It was noted that adjacent bits are guaranteed to be three sub-carriers apart from each other. So a jamming method which focused solely on jamming every third sub-carrier was developed and implemented. By targeting adjacent bits the attacker is able to overwhelm the corrective capabilities of the forward error correction coding. Through this method they were able to achieve 100% packet loss at 1% SJR.

2.2 Anti-Jamming Techniques

In this section, we present several existing methods of jamming mitigation which have been proposed.

Pietro et al. have proposed an anti-jamming technique which focuses on ensuring successful broadcast of data across and entire connected network[12]. In this technique, probabilistic transfer patterns are used in conjunction with symmetric keys to ensure the propagation of broadcast packets through the entire network. This system does have the weakness that it requires pre-connection physical setup in the form of key-exchange. This makes this mitigation method less suitable for a dynamic networking environment such as WiFi.

Direct Sequence Spread Spectrum (DSSS) has long been a method used for mitigating jamming. It relies on spreading the channel information over a much larger bandwidth than is necessary to deliver the data. In order to do this the sender and receiver need to have a shared secret spreading code. Liu et al. propose an extension of this technology which is even more resistant to jamming[18]. Using a large set of spreading codes with a low correlation between the codes they are able to encode '1' using the same code resulting in a high calculable correlation at the receiver, while '0' is encoded using two distinct codes resulting in a much lower correlation. Again this mitigation technique requires the sharing of an initial set of secret data making it difficult to implement for WiFi.

A separate jamming mitigation method has been proposed by Cassola et al.[8]. This scheme has the advantage that it does not rely on any shared secrets. It uses DSSS with a randomly generated spreading code. They show that given a sufficient amount of time this code can be determined based on reducing key entropy. However, to determine this key the end of the message must be possessed. This prevents a jammer from determining the key until it is too late and the packet has already been fully broadcast.

2.3 802.11 Interleaving Mechanism

IEEE 802.11 seeks to solve the problems of bursty errors and channel bias by using a two part static interleaver[21]. WiFi uses a 20MHz channel width which implements orthogonal frequency division multiplexing (OFDM). OFDM divides a channel into multiple narrow-band frequency sub-carriers which are each used to transmit a portion of the data within the entire channel. The goal of this interleaver is to spread bits across many sub-carrier frequencies. In 802.11 there are 52 OFDM sub-carriers each of which are 312.5 KHz wide[15]. Four of these are called pilot channels and are used during the equalization process by the receiver. This leaves 48 data sub-carriers (DSC) responsible for carrying packet data. WiFi's static interleaver spreads adjacent bits across DSC so bursty errors are easily able to be recovered through the forward-error coding scheme.

Modulation Scheme	*bpsc*	*m*
BPSK	1	48
QPSK	2	96
16-QAM	4	192
64-QAM	6	288
256-QAM	8	384

Table 1: Relationships Between Modulation Scheme, Bits per Sub-Carrier, and Message Size[21]

WiFi supports several modulation schemes, each of which vary in the number of bits they are able to transmit per DSC. Table 1 shows the relationships between the various supported modulations, bits per sub-carrier (bpsc), and message size. The lower message size modulation schemes are more often used in noisy environments as they are more noise resistant, however they are limited in bandwidth so the QAM modulation variants are preferred[15].

The data stream then goes through two steps of interleaving. The first is designed to prevent bursty errors and spreads adjacent bits across non-adjacent DSC. This is a permutation performed through:

$$P' = (P \, mod \, 16)\frac{m}{16} + \left\lfloor \frac{P}{16} \right\rfloor$$

Where P, P', and m are the initial bit index, the interleaved bit index, and the message size (Table 1) respectively. This places adjacent bits within the input stream into every third DSC within the output stream[21].

The second stage of interleaving in IEEE 802.11 is used to counter any bias in the channel. This is done by shifting the bits within each DSC so previously adjacent bits are no longer at the same index within different DSCs. The permutation used to perform this operation is:

$$P'' = s\left\lfloor \frac{P'}{s} \right\rfloor + \left(P' + m - \left\lfloor 16\frac{P'}{16} \right\rfloor\right) mod \, s$$

where $s = max(b/2, 1)$, b is the bits per sub-carrier, and P'' if the final bit index. Note that for $s = 1$ (BPSK and QPSK) this stage has no impact. For QAM it can be thought of as a circular shift of each half of a particular DSC.

2.4 Deterministic Random Bit Generators

Deterministic random bit generators (DRBG) are pseudo-random number generators which are considered sufficiently random for security purposes. We use DRGB to in order to provide a random sequence in our proposed solution as discussed in Section 3. The most common source of these is from the NIST SP 800-90a rev 1[5]. There are three main classes of DRBG: hash, block-cipher, and number theoretic. As the names imply each of these rely on a different underlying cryptographic structure. In the rest of this section we will focus on hash based DRBG (Table 2) since this is the DRBG used in our testing.

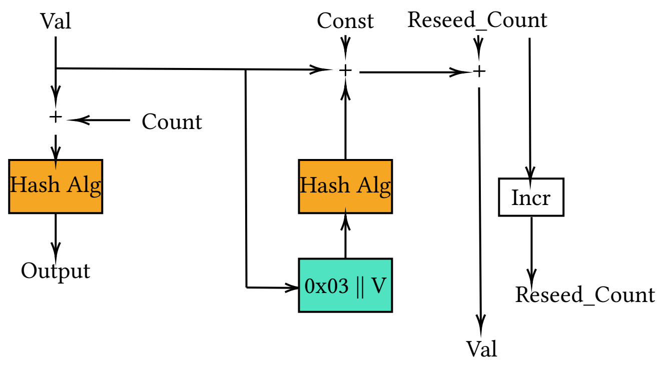

Figure 1: Hash Based DRBG Flow for Bit Generation Request[5]

When a hash-based DRGB is instantiated several steps of setup are required[5]. First the true seed is calculated as a hash of *entropyInput*||*nonce*||*personalizationString* with the personalization string being optional. This seed is set to the initial value of the DRGB internal state. The constant is then generated by hashing the seed with a byte of zeros preceding it. Finally, the reseed count is set to 1.

When random data is requested the current value gets added to an internal counter which is initially zero (see Figure 1). This is then hashed to generate an output block of bits. If more bits were requested then the internal counter is incremented and the next set of bits is generated. This proceeds until the correct number of bits have been returned. Then to update the value for the next run it is concatenated to 0x03 and hashed. This is added to the initial value and the constant. Finally this is added to the reseed counter to get the value to use for the next random bit generation.

By generating output bits in this way it is known that this is a deterministic process and because of the one-way function of hash algorithms the bits generated would be sufficiently hard to guess without knowledge of the system's state[9].

	SHA-1	SHA-256	SHA-512
Security Strength	128	256	256
Output Block Length	160	256	512
Seed Length	440	440	888
Max Bits per Request	$\leq 2^{19}$	$\leq 2^{19}$	$\leq 2^{19}$
Reseed Interval	$\leq 2^{48}$	$\leq 2^{48}$	$\leq 2^{48}$

Table 2: Summary of Hash Based DRBG Properties

2.5 Uncoordinated Frequency Hopping

We have previously covered methods mitigating jamming attacks. However, in this section we will perform a more in depth look at jamming resistant key exchange (JRKE) through uncoordinated frequency hopping (UFH)[20]. We do this because our proposed solution uses it as the method of establishing a shared secret which is pivotal for the successful mitigation of jamming attacks.

Most schemes of jamming resistance rely on the two parties holding a shared secret which they can use in a way which coordinates their efforts to circumvent the jammer's malicious efforts. However, in purely wireless systems this proves to be a circular dependency; it is difficult to establish a shared secret over a jammed channel, so the parties switch to using a jamming resistant protocol, but the parties then need to establish a shared secret for this to work. JRKE through UFH seeks to eliminate this redundancy by eliminating the need for the parties to hold a shared secret between them.

Uncoordinated frequency hopping is a method of avoiding jamming by randomly switching the channels which is being transmitted and listened on. As shown in figure 2 each time a message is transmitted there is a small but positive chance that the receiver is listening on the same channel. If this is the case then that packet of the message has been successfully received. However, because of the low probability of a packet being receiver at any given time it is necessary to resend the message many times to ensure with a high likely-hood that the receiver saw each packet of the message.

By transmitting in this way it prevents the jammer from being able to jam a single channel and forces them to either engage in costly multi-channel jamming or switch attack methods.

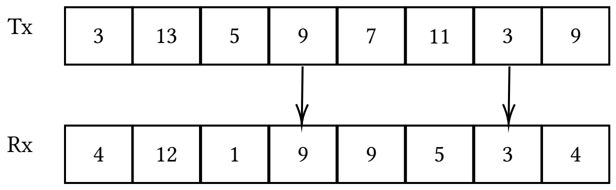

Figure 2: Uncoordinated Frequency Hopping Packet Transfer

On its own this method of UFH works to allow the transmission of data around a jammer. However, it does not prevent the attacker from inserting their data within valid transmissions from the sender. To this end a hash-link chain has been developed to allow JRKE over UFH to avoid an attacker from inserting malicious data into an otherwise valid packet. Each packet sent contains an *id* which is used to identify the message sequence, a packet number used to identify the order of the packets since it is not guaranteed that the packets will arrive in order, a portion of the message data, and a hash of the next packet's message data (see Figure 3). By linking the message fragments in this way it is possible to ensure the packets which are present in this chain are the packets that are intended to be there.

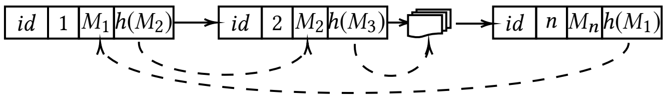

Figure 3: JRKE through UFH Hash-Link Chain[20]

However, a smart attacker may attempt to send entire malicious packets. JRKE through UFH does not inherently provided a method of preventing this behaviour. However, the key exchange used is not fixed and there are key-exchange methods which provide packet validation through a variety of methods such as certificates and packet signing. This is just one method and the actual prevention of packet spoofing is left to the implementer to choose what is considered a sufficiently secure key-exchange algorithm.

3 JELLY INTERLEAVER DESIGN

In this section, we describe our suggested approach to mitigating interleaver targeted. As suggested by Vo-Huu et al. we use a shared secret between the transmitter and receiver in order to cryptographically randomize the underlying interleaving pattern[21]. This prohibits the attack of specific sub-carriers in order to create post de-interleaving jamming. This shared secret is determined and shared at connection time through uncoordinated frequency hopping.

The rest of this section will be organized as follows. We will first discuss the additional steps required to establish a connection in section 3.1. In section 3.2 we will then discuss the modified interleaving algorithm, with section 3.3 discussing the de-interleaving mechanism. Finally, we perform an analytic analysis of Jelly interleaving in section 3.4.

3.1 Modified Connection Setup

IEEE 802.11 specifies a specific handshake used to establish a secure connection between the transmitter and receiver. In order to do this it sends packets over the air using the default interleaver[11]. This means these packets are also subjected to interleaver targeted jamming. Because of this a jammer could feasibly prevent any connection from ever being established. In order to establish the modified interleaver we propose in section 3.2 a secret needs to be shared between the two parties. Using traditionally interleaved packets for this task is again insufficient as these packets are likely to be jammed by a sufficiently motivated attacker.

In order to send and generate a shared secret seed between the two we proposed the use of UFH, as shown in figure 4. When a connection attempt is detected UFH and a secure key-exchange protocol should be used in order generate a shared secret key (note that it is not necessary for forward secrecy as we are only prevent real-time attacks). Then continuing to use UFH for our packet delivery we use the secret key to securely share sufficiently random data in order act as a seed for our DRBG. Because there are cases where the receiver and transmitter may not trust each other we propose two methods of exchanging high-entropy data. The first can be used where both parties claim to be concerned with the reliability of the channel. In this case both parties generate a bit string long enough to sufficiently seed the DRBG and the two of these strings are concatenated an used to seed the DRBG. The second method is when only one party claims to care about the reliability of the channel. In this case that party generates the bit stream and it is used without additional modification.

After the shared secret random seed has been established between the two ends, the interleaving algorithm described below is use to generate a well conditioned and random interleaving pattern. Now that the interleaver is no long vulnerable to targeted jamming, the traditional WiFi handshake can occur without concern of this method of jamming. This process will increase the time required for setup however it is necessary to prevent targeted jamming because of the slower transfer rate of UFH[20].

3.2 Interleaving Algorithm

In this section we discuss our proposed modification to the IEEE 802.11 interleaving scheme in order to mitigate interleaver targeted jamming attacks. From our UFH intial connection we we able to establish a DRBG which uses the same random seed on both sides of the communication channel. We divide our interleaver modifications into two separate stages: sub-carrier pattern generation and interleaving.

In the DSC pattern generation stage, as shown in Algorithm 1, we seek to use the DRBG to establish DSC offsets used to perform the interleaving later. To this end we generate a random DSC offset from between 1 and the number of data sub-carriers, inclusive. We then check that this value was not already present in the DSC offset pattern. If it is not present then we append this value to the DSC offset pattern. If it was already present then we simply regenerate

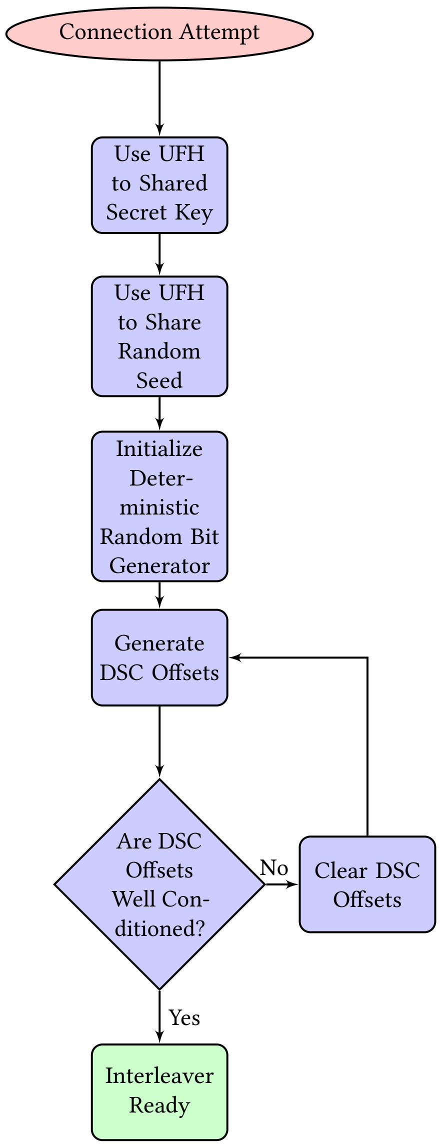
Connection Attempt
Use UFH to Shared Secret Key
Use UFH to Share Random Seed
Initialize Deterministic Random Bit Generator
Generate DSC Offsets
Are DSC Offsets Well Conditioned?
No
Clear DSC Offsets
Yes
Interleaver Ready

Figure 4: Interleaver Connection Process

a new DSC offset. Once we have a unique offset value for each bit within the sub-carrier, we perform a Fisher-Yates shuffle over the pattern to generate the final pattern[13]. It is then necessary to check that pattern is well conditioned to be an interleaver. This is done by ensuring there are no two values in the pattern which are adjacent sub-carriers, so there is at-least a sub-carrier worth of buffer between bits within the same initial DSC. If the pattern is not well conditioned we attempt to generate another patter using the DRBG in its current state. This continues until we find a sufficiently conditioned pattern or the DRBG exceeds its reseed interval at which point we share a new random seed between the two parties and continue.

Generating a DSC pattern only needs to be done once per connection. For improved security this pattern should time-out and be re-computed occasionally. After it has been established it can be used to perform the actual interleaving. We first divide the input stream into a 2D array of bits where each row represents a DSC and each column represents all the values at the particular index of each initial sub-carrier. Then the transformation performs a circular shift on each row where the shift distance is determined by the value in the corresponding position of the DSC pattern. This can be more linearly thought of as shifing each bit within each sub-carrier by the DSC pattern offset in the corresponding position, as we show in Algorithm 2. Because of the minimal computation requirements after initially generating the DSC pattern this algorithm has the potential to be implemented in hardware and operate at high speeds.

Algorithm 1 DSC Interleaver Pattern Generator Algorithm

```
Require: $bpsc \in \{1, 2, 4, 6, 8\}$
  $DSCoffsets \leftarrow$ Empty Array of Size $bpsc$
  $i \leftarrow 1$

  while $i \leq bpsc$ do
    $randBits \leftarrow DRBG$
    $randDSC \leftarrow \lceil numSubCarriers * randBits / maxRandVal \rceil$
    if $randDSC \notin DSCoffsets$ then
      $DSCoffsets[i] \leftarrow randDSC$
      $i \leftarrow i + 1$
    end if
  end while

  fisherYatesShuffle($DSCoffsets$)

Ensure: $\forall i, j \in [1, bpsc], i < j :$
  $|DSCoffsets[i] - DSCoffsets[j]| \neq 1$
  return DSCoffsets
```

Algorithm 2 Bit Interleaver Algorithm

```
Require: $inputBits \equiv$ 2D Array of Size $bpsc$ X $dataSubCarriers$
  $outputBits \leftarrow$ Empty 2D Array of Size $bpsc$ X $dataSubCarriers$

  for $b \in [1, bpsc]$ do
    for $d \in [1, dataSubCarriers]$ do
      $newDSC \leftarrow mod((d + DSCoffsets[b]), dataSubCarriers)$
      $outputBits[b][newDSC] \leftarrow inputBits[b][d]$
    end for
  end for

  return outputBits
```

3.3 De-Interleaving

Our interleaving algorithm shifts the bit in each position by a certain value, so we are able to re-establish the original bit stream by pulling each bit from its final location. We show this in Algorithm 3.

Algorithm 3 Bit De-Interleaver Algorithm

```
Require: $inputBits \equiv$ 2D Array of Size $bpsc$ X $dataSubCarriers$
  $outputBits \leftarrow$ Empty 2D Array of Size $bpsc$ X $dataSubCarriers$

  for $b \in [1, bpsc]$ do
    for $d \in [1, dataSubCarriers]$ do
      $oldDSC \leftarrow mod((d + DSCoffsets[b]), dataSubCarriers)$
      $outputBits[b][d] \leftarrow inputBits[b][oldDSC]$
    end for
  end for

  return outputBits
```

3.4 Analytic Analysis of Interleaving Patterns

There are several tasks which an interleaver must achieve in order to be successful before we are able to evaluate the security of the system against targeted attacks. To this end, we ensure that our interleaver will provide protection against bursty errors on a given frequency. This is done by checking that the data sub-carrier offsets generated through the interleaving pattern generator algorithm are at least two sub-carriers apart for bits which are initially within the same sub-carrier. This means we can guarantee that there is a sub-carrier buffer between these so the noisy burst would have to span over an entire sub-carriers which is 312.5 KHz in 802.11's scheme of OFDM[3]. Additionally, interleavers for OFDM need to shuffle adjacent bits so they do not share the same position in sub-carriers. This is necessary to combat any bias distortion which could occur over the same bit of different sub-carrier OFDM. Our proposed interleaver naturally does this by leaving the bits in the same position within their shifted data sub-carrier as they were in their initial sub-carrier. Through this qualitative analysis it is possible to show that our interleaving mechanism meets these minimal requirements as an effective interleaver.

Of more concern is the security of this design against malicious attacks. We explore the security levels of each component within our design in the remainder of this section.

The first potential attack surface in our design is the process of using UFH to generate a shared secret which is then used to seed the DRBG. There are two main attacks we consider on this front: man-in-the-middle and secret leakage. Both of these would allow the attacker to determine a subset of sub-carriers which could be targeted to perform the same jamming effect. Strasser et al. were able to show that because of the nature of the hash-link chains used in UFH it becomes exponentially difficult for an attacker to insert or modify data within an individual message chain because of the difficulty in finding a hash collision for the a proceeding packet[20]. This is only as secure as the hash algorithm, but it is known that SHA-256 has a security level of 128 bits so this is not a vulnerability in the system against current computation resources[4]. However, UFH does not prevent against the malicious delivery or reception of packets during the key exchange performed over it. This must be accounted for within the key-exchange protocol. This is a solved problem and can be overcome by using digital signatures for authentication, time-stamps for freshness, and a strong key exchange such as Elliptic Curve Cryptography with the Diffie-Hellman key exchange for example[7, 23].

The next attack surface is the randomness of the shared secret and the security of the DRBG. The entropy of the shared secret is a difficult problem as it requires the generator to be truthful about their entropy. It is for this reason we proposed two methods of generating the shared secret. This allows the parties which claim to be concerned with the jamming resistance for the connection to contribute to the shared secret so they only need to trust themselves. In terms of the security of the DRBG, NIST further discusses the security considerations of each type of DRBG in SP800-90a revision 1[5]. Our implementation used the Hash-based DRNBG with SHA underlying and NIST considers this as secure as the digest size of the particular hash (up to 256 bits of security).

The final potential attack surface is the set of possible interleaver configurations. If this number were small it would be substantially more likely for an attacker to be able to randomly guess the interleaver configuration. To ensure the security we now show the number of possible configurations is large enough to not pose a security concern. We again let b be the number of bits per sub-carrier and we use 48 data sub-carriers as is standard for 802.11. We start by determining the number of unconditioned possibilities. This number is fairly small for BPSK and QPSK ($b = 1$ and 2 respectively), however it grows much larger for QAM. We now must determine the number of combinations which do not meet our conditioning requirements (see Appendix A for proof).

$$N \geq \prod_{i=0}^{b-1} 48 - 3 * i$$

Note that this is a strict lower bound on the number of possible combinations. Using this bound we find the number of unique configurations to be larger than 3,538,080 and 4,203,239,040 for 16QAM and 64QAM respectively. This provides an adequately large search space so that an attack is unlikely to properly choose the correct configuration at random.

4 TESTING METHOD

For testing our modified interleaving mechanism we sought to stay close to the testing procedure of the initial attack paper by Vo-Huu et al. in order to obtain consistent results[21]. To this end we obtained the source code used by the attack paper to generate the jamming signal on a HackRF[22]. In this section, we discuss our testing procedure.

For our transmitter and receiver we used software defined radio as the interleaver is most often a hardware component and we needed to modify it. We elected to use the *gr-ieee802-11* implementation for GNU-Radio because of its open source nature and relatively high speeds[6]. We modified the underlying implementation to use our design with a fixed specified shared secret. We did not elect to perform the secret sharing and initialization because it is beyond the scope of this paper and there are other publications which have already explored that topic.

Our implementation for testing used a SHA-256 hash-based DRBG as our deterministic source of randomness. Over the course of testing we tested across 30 different initial seeds which were randomly generated using */dev/random* as our source of entropy.

A HackRF with the provided attack firmware was used as our jamming device, with SDRs for both our transmitter and reciever. All three of these devices were linked through SMA and a T-junction as shown in Figure 5 and 6.

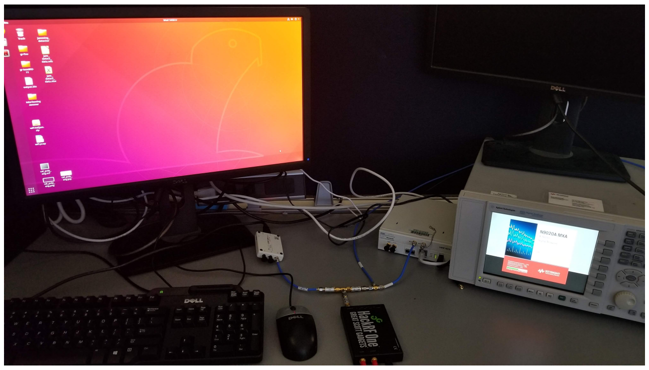

Figure 5: Test Bench Setup

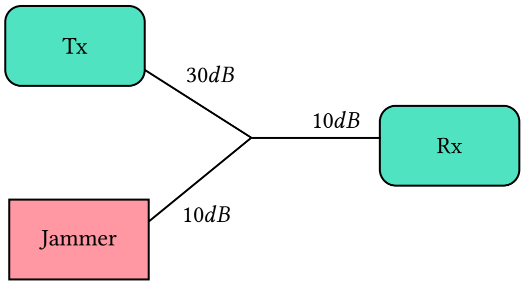

Figure 6: Testing Setup Diagram. Attenuation in dB of each connection is shown.

For every data run we first calculated our SJR using a spectrometer attached at the receiver to determine the average channel power. We then launched the attack and sent data packets from our transmitter. We then counted the number of packets received which had the correct check-sum. Each test was performed across several of the 2.4GHz channels, before the results were compiled.

5 RESULTS

In this section we evaluate our modified interleaving algorithm against interleaving targeted jamming. To evaluate the efficiency of a jamming attack we will use the packet error-rate (PER) metric. To attain a comparison between varying jamming techniques we use the SJR of the channel under attack.

5.1 Interleaving Jamming

We first test our Jelly interleaver and the 802.11 interleaver against Vo-Huu et al. interleaving jamming technique. For this set of tests we jammed on DSCs 0, 3, 6, 9, 12, 15, and 18 since this is the largest subset of DSCs targetted by the attack paper.

In Figure 7, we show the results for this test which clearly illustrate the impact which our re-designed interleaver is able to have.

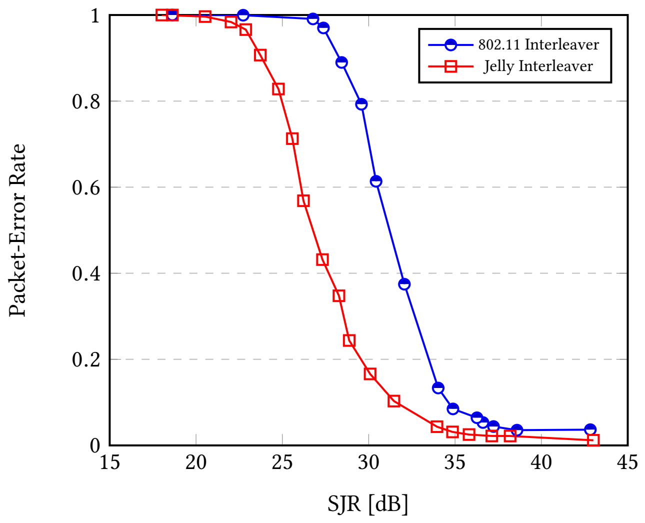

Figure 7: Interleaving Jamming of Modified and 802.11 Interleaver

Our interleaver increase the channel power requirement by as much as 4.5 dB in order to achieve the same jamming level (0.6 PER). This data shows that our modified design is effective at mitigating the interleaver targeted jamming attack.

5.2 Other Jamming Techniques

While our results are extremely positive for mitigating interleaving jamming this is not the only jamming method present in the wild. In this section we test our interleaver against two other types of jamming in-order to verify our design does not create any further vulnerabilities against these common attacks.

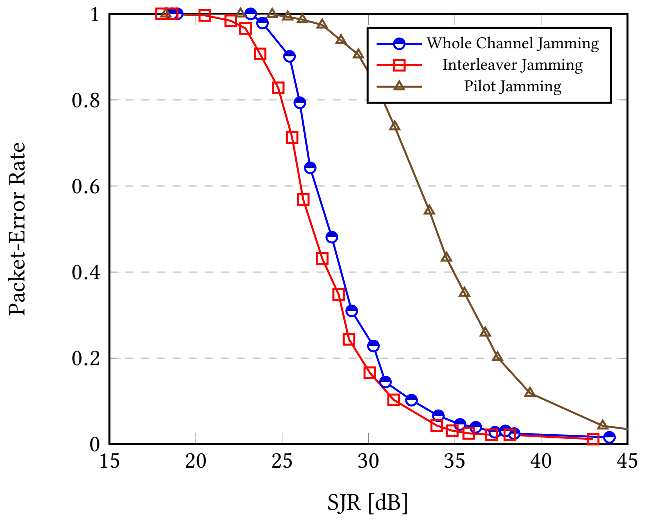

Figure 8: Interleaving, Whole-Channel, and Pilot Jamming of Jelly Interleaver

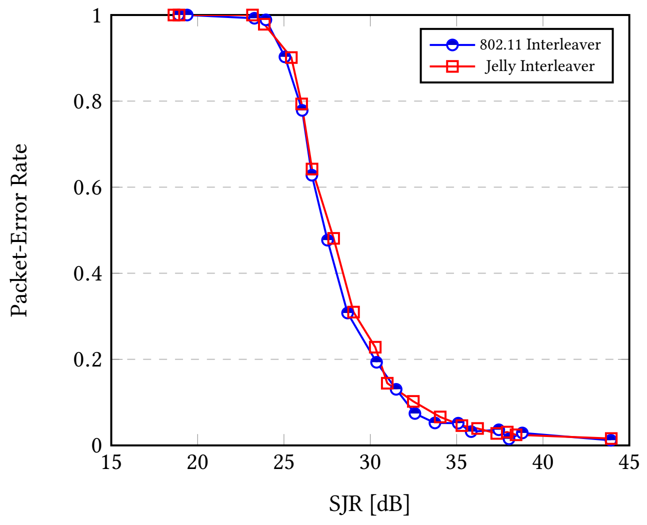

Figure 9: Whole-Channel Jamming of 802.11 and Jelly Interleaver

The first jamming method we test against is whole channel-jamming. In Figure 8, we show our interleaver modifications work so well against the attack that we reduce the power efficiency of the attack below the power efficiency of whole-channel jamming. This shows we have completely negated this jamming attack vector. Through comparing the results of our modified design against whole-channel jamming with the results of 802.11 jammer against the same attack we show, in Figure 9, our interleaver does not inherently exacerbate this attack vector as it performs as well against this common jamming technique.

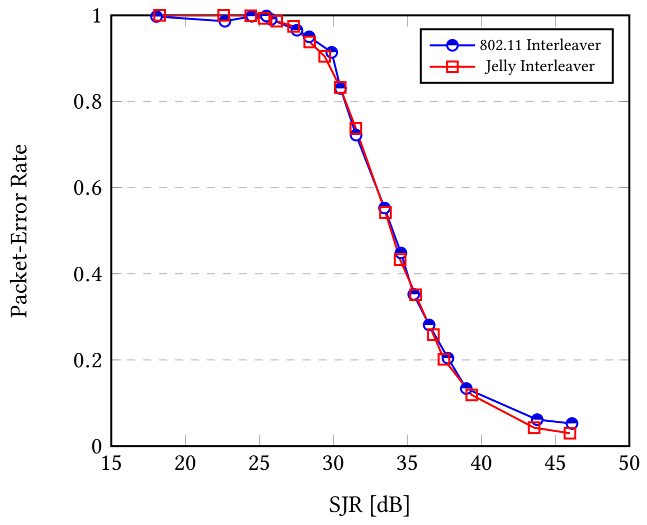

Figure 10: Pilot Jamming of 802.11 and Jelly Interleaver

We also tested our jammer against pilot jamming. As discussed earlier this attack is an attack on ODFM itself so we would not expected to see any change in the effect of this jamming technique. Our tests demonstrate this is the case as both the 802.11 interleaver and our Jelly interleaver are both nearly identically impacted by pilot jamming.

The results from our tests against other jamming methods demonstrate that our jamming mechanism does not expose the RF communication channel to any further risk from pre-existing jamming attacks than the 802.11 design. This in combination with the strong results from our interleaving jamming tests show that our Jelly interleaver successfully mitigates an interleaving targeted jamming attack.

6 FUTURE WORK

Our work has demonstrated the interleaver targeted vector attack vector is not unavoidable. Because of this it is important for this issue to be considered in the development of future wireless data protocols with a static interleaver. Other methods should also be explored and considered along side our proposed solution.

Further direct comparison and cost-benefit analysis of all potential strategies needs to be further experimentally and mathematically explored in order to determine which strategy provides the best mitigation without hindering bandwidth or usability to excess.

Continuing work on this particular method needs to be performed to ensure that our method does not expose another attack vector. Additionally, more work should be done to explore dynamic interleaving across time as well as connections. This problem appears to come with a non-negligible overhead requirement, so it could be beneficial to explore this as a game-theoretic problem to determine the transmitter and receiver's ideal reseed rate. Because of the high data-rates which IEEE 802.11 supports it will eventually be necessary to develop hardware which performs

7 CONCLUSION

In this paper, we presented and tested a novel method modifying the static interleaver of IEEE 802.11 in order to mitigate the interleaving jamming technique presented by Vo-Huu et al.. We use UFH in order to exchange a shared secret between the two parties. This shared secret is then used to generate a deterministic interleaving offset pattern across non-adjacent DSCs to maintain resilience to bursty attack noise. Using this pattern bits are interleaved using an algorithm which is designed to be easy to implement in high-speed hardware. We analytically analyzed this data to show it secure against malicious attack. Finally, we tested an implementation of 802.11 with our Jelly interleaver against several jamming attacks. We showed our modified jammer completely negates the interleaving targeted jamming attack. We also show that our Jelly interleaver does not introduce any additional vulnerabilities against whole-channel jamming and pilot jamming.

REFERENCES

[1] [n. d.]. FCC ENFORCEMENT BUREAU STEPS UP EDUCATION AND ENFORCEMENT EFFORTS AGAINST CELLPHONE AND GPS JAMMING. *FCC* ([n. d.]). https://www.fcc.gov/document/fcc-enforcement-bureau-steps-education-and-enforcement-efforts-against
[2] [n. d.]. Marriott to Pay $600K to Resolve WiFi-Blocking Investigation. *FCC* ([n. d.]). https://www.fcc.gov/document/marriott-pay-600k-resolve-wifi-blocking-investigation

[3] 2009. IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC)and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput. *IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009)* (Oct 2009), 1–565. https://doi.org/10.1109/IEEESTD.2009.5307322

[4] Elaine Barker and Quynh Dang. 2016. NIST Special Publication 800-57 Part 1, Revision 4. *NIST, Tech. Rep* (2016).

[5] Elaine Barker and John Kelsey. 2015. NIST Special Publication 800-90A Revision 1: Recommendation for Random Number Generation Using Deterministic Random Bit Generators. *NIST, June 20q5, http://nvlpubs. nist. gov/nistpubs/SpecialPublications/NIST. SP* (2015).

[6] Bastian Bloessl. 2013-2019. gr-ieee802-11. https://github.com/bastibl/gr-ieee802-11. (2013-2019).

[7] Ran Canetti and Hugo Krawczyk. 2001. Analysis of key-exchange protocols and their use for building secure channels. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 453–474.

[8] Aldo Cassola, Tao Jin, Guevara Noubir, and Bishal Thapa. 2012. Efficient spread spectrum communication without preshared secrets. *IEEE Transactions on Mobile Computing* 12, 8 (2012), 1669–1680.

[9] Donghoon Chang, Sangjin Lee, Mridul Nandi, and Moti Yung. 2006. Indifferentiable security analysis of popular hash functions with prefix-free padding. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 283–298.

[10] T Charles Clancy. 2011. Efficient OFDM denial: Pilot jamming and pilot nulling. In *2011 IEEE International Conference on Communications (ICC)*. IEEE, 1–5.

[11] Todor Cooklev. 2004. *Wireless Communication Standards: A Study of IEEE 802.11, 802.15, 802.16.* IEEE Standards Association.

[12] Roberto Di Pietro and Gabriele Oligeri. 2015. Freedom of speech: thwarting jammers via a probabilistic approach. In *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 4.

[13] Ronald A Fisher and Frank Yates. 1943. *Statistical tables for biological, agricultural and medical research.* Oliver and Boyd Ltd, London.

[14] Kanika Grover, Alvin Lim, and Qing Yang. 2014. Jamming and anti-jamming techniques in wireless networks: a survey. *International Journal of Ad Hoc and Ubiquitous Computing* 17, 4 (2014), 197–215.

[15] Steve Kapp. 2002. 802.11 a. More bandwidth without the wires. *IEEE Internet computing* 6, 4 (2002), 75–79.

[16] Matthew Knight and Balint Seeber. 2016. Decoding LoRa: Realizing a modern LPWAN with SDR. In *Proceedings of the GNU Radio Conference*, Vol. 1.

[17] Matthew J La Pan, T Charles Clancy, and Robert W McGwier. 2013. Phase warping and differential scrambling attacks against OFDM frequency synchronization. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2886–2890.

[18] Yao Liu, Peng Ning, Huaiyu Dai, and An Liu. 2010. Randomized differential DSSS: Jamming-resistant wireless broadcast communication. In *2010 Proceedings IEEE INFOCOM*. IEEE, 1–9.

[19] Christopher Mueller-Smith and Wade Trappe. 2013. Efficient OFDM denial in the absence of channel information. In *MILCOM 2013-2013 IEEE Military Communications Conference*. IEEE, 89–94.

[20] Mario Strasser, Christina Popper, Srdjan Capkun, and Mario Cagalj. 2008. Jamming-resistant key establishment using uncoordinated frequency hopping. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, 64–78.

[21] Triet Dang Vo-Huu, Tien Dang Vo-Huu, and Guevara Noubir. 2016. Interleaving jamming in Wi-Fi networks. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 31–42.

[22] Triet Dang Vo-Huu, Tien Dang Vo-Huu, and Guevara Noubir. 2016. WiFi-Intljam. http://www.ccs.neu.edu/home/vohuudtr/projects/wifi-intljam/wifi-intljam.zip. (2016).

[23] Qi Xie, Duncan S Wong, Guilin Wang, Xiao Tan, Kefei Chen, and Liming Fang. 2017. Provably secure dynamic id-based anonymous two-factor authenticated key exchange protocol with extended security model. *IEEE Transactions on Information Forensics and Security* 12, 6 (2017), 1382–1392.

APPENDIX A PROOF OF LOWER BOUND OF INTERLEAVING SCHEMES

Let N be the number of permutations of $[1, 48]$ into b such that no two elements are within 1 of each other.

Show:

$$N \geq \prod_{i=0}^{b-1} 48 - 3 * i$$

Start: By induction

Let n_i be the number in position i (1-indexed) of the permutation.
Let S_i be the set of potential values in position i.
Let $P_i = |S_i|$.
Let N_i be the number of permutations for $b = i$

For P_1 then $S_1 = [1, 48]$ so $N \geq 48$

For P_i the number of permutations can be any value other than those previously choose and the two values adjacent to those chosen.

So $S_i = S_{i-1} \backslash [n_i - 1, n_i + 1]$
So $P_i \geq P_{i-1} - 3$.
So $P_i \geq P_{i-1} * (48 - 3 * i)$

Q.E.D