REBUF: Jam **Re**sistant **BBC** based **U**ncoordinated **F**requency Division Multiplexing

Jaemin Ashley
EECS Department
United States Military Academy
West Point, NY
jaemin.ashley@westpoint.edu

Joshua Groen
EECS Department
United States Military Academy
West Point, NY
joshua.groen@westpoint.edu

Michael Collins
Laboratory for Advanced Cybersecurity Research
Annapolis Junction, MD
mdcolli@tycho.ncsc.mil

***Abstract*—One of the central tenants of information security is availability. One common form of attack against the availability of information in wireless networks is jamming. Currently, the most common techniques to provide jam-resistant communication, such as frequency-hopping spread spectrum (FHSS), are based on the use of a symmetric shared secret. However, there are theoretical approaches to jam resistance without a pre-shared secret. One theoretical approach using concurrent codes, called the BBC algorithm, was developed at the United States Air Force Academy.**

We developed and tested the effectiveness of REBUF, a Jam Resistant BBC based Uncoordinated Frequency Division Multiplexing (FDM) system. REBUF is the first known implementation of the BBC algorithm in a simultaneous frequency division multiplexing system. The contributions of this paper include: demonstrating the practical use of a BBC based FDM system, the ability of such a system to jam traditional orthogonal frequency division multiplexing (OFDM) systems, and the resilience of such a system to some common forms of jamming.

***Index Terms*—Spread Spectrum, Jam resistant, Random access, MAC Layer, FDM, BBC**

I. Introduction

A. Challenges of Using Preshared Identical Secrets

Wireless networks have traditionally relied on spread spectrum techniques to ensure the availability of network resources [1]–[4]. A common example of this is the military's use of frequency-hopping spread spectrum (FHSS) where the transmitter and receiver rapidly hop from one frequency channel to another in a synchronized and unpredictable matter. In order for successful FHSS communication to be achieved, an initial secret of a pseudo-random sequence of frequencies must be shared between the communicating devices and must be unknown to all other devices [5], [6]. The sharing of an initial secret is the defining aspect of all currently implemented spread spectrum communication techniques. This need for an initial secret for successful communication in spread spectrum communication systems can be problematic.

Systems based on shared keys have poor scalability. Communication systems with a large user-group must either provide a single-shared secret to every user or give each user a unique shared key [7]. If large user-groups rely on a single shared key, the attacker only needs to obtain one such key to be able to jam the whole system [2]. If each member of a large-user group had their own unique-shared key, the system requires storing every shared key and listening on millions of channels simultaneously, which is impractical.

There is little flexibility in re-establishing jam-resistant communication if the secret key is compromised. In order to re-establish jam-resistance, a new key must be shared between the system users, typically requiring an out of band communication method. Integration of the REBUF system into military-grade communication devices would allow users an additional method of establishing jam-resistant communication. REBUF requires no initially shared key in order to establish jam-resistant communication.

B. Concurrent Codes and the BBC Algorithm

The BBC algorithm allows for a jam-resistant communication system without the need for an initial shared secret [6], [8]. The BBC algorithm is the first efficient algorithm utilizing concurrent coding to achieve this [8]. Concurrent coding theory is quite complex; an in-depth description of concurrent codes can be seen in the Concurrent Coding Theory section of [9]. REBUF implements concurrent codes through the BBC algorithm through a form of frequency division multiplexing, originally described as Simultaneous Frequency Hopping in [8]. The BBC algorithm consists of two sub-algorithms: an encoding algorithm and a decoding algorithm.

The BBC encoding algorithm is based on the ability to create indelible marks [8]. For a given message, the actual locations of the marks are found by taking the hash of each prefix of the message. For example, the word "1011" will have four hashes. The prefixes to be hashed from "1011" are "1", "10", "101", and "1011". The hashing function used in REBUF is the Glowworm hash [10]. The locations of these marks determine the specific frequencies that will be turned on to create that packet's signal. In this way, a packet can be visualized as a string of 0's and 1's, where the value of 1 represents the location of the indelible marks and value of 0 represent no mark.

The BBC decoding algorithm is responsible for deriving all possible messages from a packet. An efficient implementation of the decoding algorithm is a depth-first search of the message space utilizing the locations of the indelible marks within the packet [8], [11]. As the BBC decoding algorithm progresses, it will search through a binary tree representing the message

978-1-7281-4695-9/20/$31.00 ©2020 IEEE

being constructed. Each node in the tree represents a "prefix." The decoding algorithm has three available actions at a node: append a 0, append a 1, or drop the current message because it is not valid. Both appending options symbolize constructing the next potential prefix. The newly constructed prefix will then be hashed to obtain a location. The validity of the prefix constructed so far is determined by the presence or lack of an indelible mark in the location [9]. Schweitzer does a fantastic job visualizing the decoding process in [7].

C. Indelible Mark Density

An underlying assumption of the BBC algorithm is that all marks are indelible. An indelible mark is defined as the inability for attackers or external influences to delete existing marks within a packet. An attacker has two options to effectively attack a packet: remove marks or add marks. To completely erase a mark, an attacker must send the inverse of the transmitted signal at the correct time to cancel out the original sine wave. This is assumed to be too sophisticated for an attacker to achieve in real time. A more detailed discussion of indelible marks can be found in [8].

Mark density is the ratio of the number of marks to the total packet size. The encoding algorithm used for the current system allows the option to choose a certain mark density. Each packet can only transmit a message up to size x, which is determined by the number of possible marks and the mark density. If the total message size, n, is larger than the specified size, x, the message is fragmented into $\frac{n}{x}$ fragments of size x and a packet is devoted for each fragment. During a transmission, there are two ways the mark density of a packet can increase: environmental noise and jamming. Both factors introduce energy at some given frequency, which has the potential to turn certain locations in a packet into a mark.

D. Hallucinations and BBC Vulnerabilities

The consequences of high mark density include hallucinations and overwhelming the decoder. Hallucinations are defined as messages decoded by the receiver that were never intentionally encoded by the transmitter [8]. There are three classifications of hallucinations: working, terminal, and realized. Working hallucinations only exist during the decoding process and are commonly generated and eliminated. Terminal hallucinations are those that survive the decoding process. These hallucinations can be eliminated at an exponential rate by adding known checksum bits to the end of each packet. Realized hallucinations are those that survive the decoding process and the checksum bit validation [12].

The greater consequence of high mark density is decoding overload. BBC can always successfully decode the message even with high mark density, though there may be Realized hallucinations as well as the original message. It can be shown that as long as the packet density remains below a critical density (around 50%), the hallucination density remains steady and the decoding time stays linear. However, the time to decode increases exponentially above this critical density point [9], [12], [13].

E. Related Works

There are numerous works on the BBC algorithm including [6]–[12], [14]–[16]. In particular, Baird describes a hashing function specifically designed for the BBC algorithm, the Glowworm Hash [10]. A good visual demonstration of the effectiveness of BBC in decoding messages in a noisy environment can be seen in [14]. Bahn studies the effect of extending the critical mark density in BBC encoded packets through the use of checksum bits [12]. This work found that checksum bits enabled critical mark density to be increased to a higher level.

There is also a wide array of studies conducted on jamming and jam-resistant radio networks outside of the BBC algorithm. Mpitziopoulous introduces the jamming vulnerabilities of many Wireless Sensor Networks (WSNs) and performed an analysis on the performance of traditional jam-resistant techniques [1]. Press explores the theory of using "hypercarrier parallelism," a concept similar to the FDM implementation of BBC [17]. This work builds some theoretical foundations to analyze the non-symmetic properties of binary on/off keying. Popper created a communication technique called Uncoordinated Direct-Sequence Spread Spectrum that would establish jam-resistance within broadcast communication without the need for a shared secret [3], [18], [19]. Di Benedetto proposed a MAC layer protocol that provides uncoordinated, wireless medium access for Ultra Wide Band (UWB) radio networks [20] without a shared secret. Xio proposes a system using collaborative broadcast and uncoordinated frequency hopping without a shared secret [21], [22].

II. Design of REBUF

A. BBC based Uncoordinated FDM

While different implementations of the BBC algorithm are possible, there is currently no known use of the BBC algorithm for frequency division multiplexing. REBUF is a form of Jam *Re*sistant *B*BC-based *U*ncoordinated *F*requency Division Multiplexing simulated and implemented in a laboratory environment using GNU-Radio and USRP Software Defined Radios (SDRs). The indelible marks of REBUF are pure sinusoidal signal's at a given frequency with random phase. The total frequency bandwidth is split into a fixed number of sub carriers. Each sub carrier represents the possible location of a mark.

BBC is fundamentally an uncoordinated coding scheme, meaning there is no time synchronization required between the sender and receiver [8]. This property holds for REBUF. A sender can initiate a message at any time. The sender must create the marks in a given packet for some finite amount of time, ϵ, but that amount of time can change. This can be expressed as a packet sending rate of $\frac{1}{\epsilon}$. As long as the receiver collects samples of the spectrum at a rate greater than $\frac{1}{\epsilon}$ all the packets can be recovered with no need to synchronize the sending and receiving clocks.

B. Signal Space Description

REBUF can also be thought of as a system that maps the input message to a signal space consisting of all possible signals of dimension N, where N is the maximum BBC code word size. We further restrict the code word to have a certain maximum density, as discussed in section I-C. The receiver must decide what was most likely sent, given what it received. REBUF is built on the underlying assumption that no marks are lost (no 1's become 0's). Instead of using a traditional maximum likelihood decoder to find the nearest valid signal to what was received, the receiver returns all possible code words that are subsets of what was received.

As an illustration, let $N = 3$ with maximum density of $\frac{1}{3}$. The set of valid code words that could be transmitted for this example are shown in equation 1.

$$S_{valid} = \begin{Bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{Bmatrix} \tag{1}$$

However, the set of possible code words that could be received include all possible signals of dimension 3, as shown in equation 2.

$$S_{possible} = \begin{Bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{Bmatrix} \tag{2}$$

In this example, if the receiver detected $\begin{Bmatrix} 1 & 0 & 1 \end{Bmatrix}$ it would return all valid code words that are subsets of what was received: ie, those shown in equation 3.

$$S_{receive} = \begin{Bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \end{Bmatrix} \tag{3}$$

As long as the received packet density is below the critical mark density threshold, the original message can always be decoded in a reasonable amount of time [12].

C. Implementation

We used USRP N310 software defined radios (SDR) to implement the new physical layer and allow for rapid development and testing. The USRPN310 SDR has a maximum instantaneous bandwidth of 100 MHz. However, the limiting factor for the bandwidth was the network connection speed between the computer and the SDR which provided us 5 MHz of usable bandwidth. We used a center frequency of 2.452 GHz. While any frequency could be used, we chose this frequency because it is the center frequency of 802.11 WiFi channel 9. This allowed us to directly compare the jam resistance of REBUF to OFDM based WiFi.

D. Transmitter

We used GNU Radio to interface with the SDR which allowed us to write our code in Python. REBUF is configured to use BBC to encode messages into vectors of 1024 bits with a mark density of 11 percent. This bit vector is used to generate the frequency tones transmitted by taking the IFFT of the bit vector. We designed our system to avoid a DC tone in the base-band. In order to accurately create the tones, the IFFT must include both the positive and negative frequency components. This required us to duplicate the original vector, flip the order of the bits, and prefix it to the original vector. Figure 1 illustrates the bit positions of the final vector that is passed to the IFFT function. The output of the IFFT is a sum of cosines at the specified frequencies. This is then shifted to the transmission center frequency of 2.452 GHz and sent to the SDR via a GNU Radio sink. The GNU Radio block diagram for the sender can be seen in figure 2.

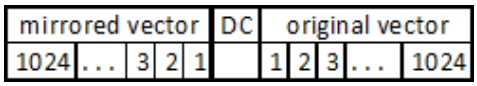

Fig. 1. Creating the bit vector that is used by the IFFT

E. Receiver

The receiver uses the SDR as the source in GNU Radio. The input is passed through a FFT to extract the frequency components. We utilized the magnitude squared of the FFT to detect the energy peaks present. The GNU Radio block diagram for the receiver is shown in figure 3. To determine which energy peaks represent a mark, we used a two pass detection logic. First we used a global threshold to separate peaks from the background noise. Then we used a sliding window average to ensure any identified peaks are actually higher than the local noise threshold. We also eliminated any peaks that are not at a legitimate frequency. The detection logic can be seen in listing 1. The result of the peak detection algorithm is a vector of 1's (mark present) and 0's (no mark). This bit vector is then passed to the BBC decoder which produces a list of all possible code words as the output. If a peak is missed during the detection portion of REBUF, it is essentially violating BBC's underlying assumption that marks cannot disappear. A single missed peak is enough to corrupt the packet and the decoding algorithm will fail to produce the original code word.

```
import numpy as np

vectorSig = Queue.get()
vectorPos = vectorSig[8191:]

# Global threshold
ind = np.where(vectorPos > np.mean(vectorPos)/20)
ind2 = list(ind[0])

# Sliding threshold
peak = []
for l in ind2:
    if l%7 == 0:
        ll = l + 8191
    if valPos[l] == np.max(vectorSig[ll-6:ll+7]) and
     valPos[l] > np.max(vectorSig[ll-82:ll+82]/17):
        peak.append(l)
```

Listing 1. Detection Logic

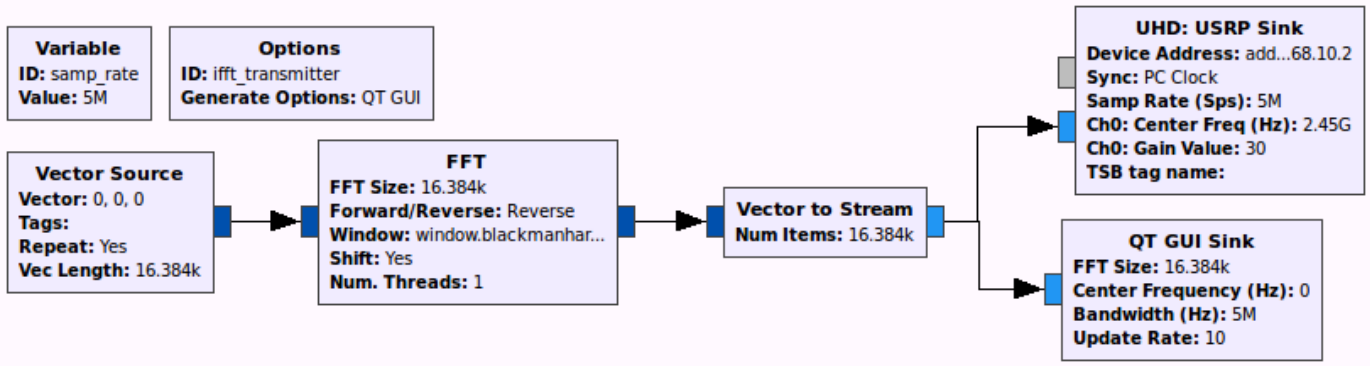

Fig. 2. GNU Radio block diagram for the sender

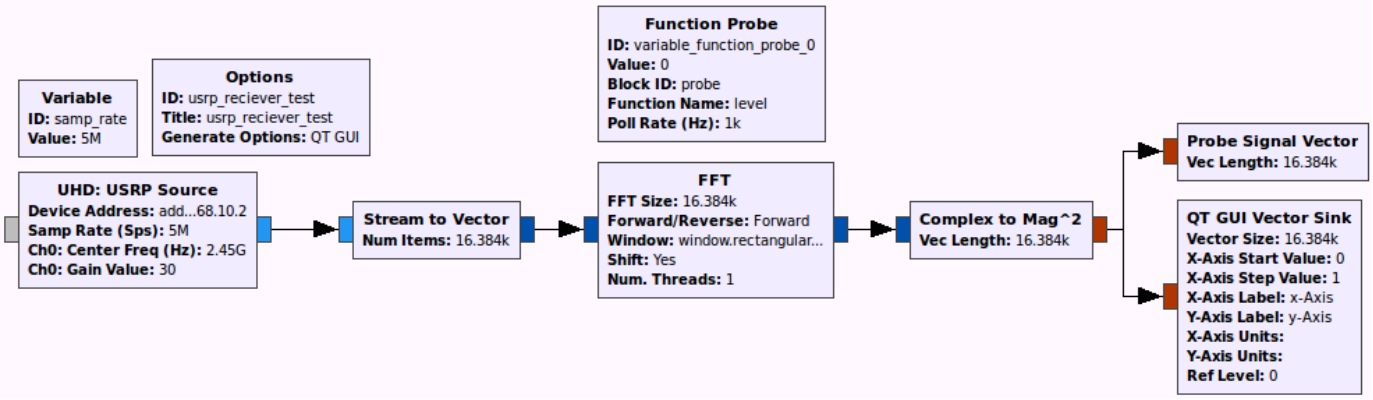

Fig. 3. GNU Radio block diagram for the reciever

The receiver waits until it detects a typical REBUF packet. The receiver then records all received packets as raw signal vectors into a queue. After the receiver detects an end of transmission signal, it begins decoding every packet in the queue in order of receipt. Throughout our experimentation we found that the encoder was significantly faster than the decoder. We manually reduced the sending rate to ensure the decoder had enough time to process the packet. Future work should focus on improving the decoder and eliminating this manually imposed throttle. While these solutions are far from optimal, our main focus was to test the packet loss rate of the system in the presence of jamming. Future work for optimization of the system is discussed further in section V.

F. BBC as a MAC Layer Protocol

REBUF is implemented as a physical and MAC (or Data Link) layer protocol within the network stack. The MAC layer is primarily responsible for providing node-to-node connection and addressing, and providing bit level error detection and/or correction. REBUF uses the BBC algorithm at the MAC layer to provide a jam-resistant node to node connection. The current BBC implementation has an elementary addressing protocol in place. Messages encoded by the BBC algorithm are prefixed with BBC header that includes the packet ID, next hash, and current hash information. An opportunity for future work is the implementation of unique check bit patterns for each pair of nodes. This would simultaneously solve packet addressing and further protect against terminal hallucinations. The BBC algorithm also provides error detection and correction of data against noise and jamming, which is primarily handled in the decoding algorithm at the receiver.

III. Experimentation Results

A. Testing

BBC is a provably jam-resistant algorithm, given the conditions outlined in section I-C [6], [8], [9], [11]. However, the validity of the conditions listed in section I-C in a real system are less well known. To test the effectiveness of REBUF, we created a test environment that consisted of: a laptop connected via Ethernet to single WiFi access point, a WiFi client, a REBUF transmitter and receiver pair, and a malicious jamming source. We repeatedly sent a 10.3 KB text file from the sender to the receiver.

We wanted to compare the jam resistance against WiFi, so first we focused on finding an effective signal for jamming WiFi. We tried several possible signals, including AWGN noise spread across 20MHz, pure frequency tones at the OFDM sub-carriers, and a REBUF transmitter sending random data with a mark density of 5%. We ran IPERF3 between a laptop and WiFi client to observe the throughput and effect of jamming. There was little to no effect for the WiFi throughput

TABLE I
PACKET LOSS PERCENTAGE

Trial	No Jamming	With REBUF Jamming
1	.035%	.043%
2	.148%	.031%
3	.064%	.179%
4	.028%	.038%

for the first two jamming methods. However, the REBUF transmitter sending random data stopped all transmission, even causing the WiFi client to look for a new access point.

Next we tested the REBUF system in the presence of WiFi traffic. Unsurprisingly, the WiFi client was totally jammed and disconnected from the access point. REBUF did not suffer any performance degradation. Then we tested REBUF in the presence of each of the jamming signals. REBUF did not suffer any performance degradation for any of the signals. We conducted the most thorough testing for the random REBUF jamming signal because that was the most effective at jamming OFDM. Table I depicts the throughput and error rates of eight runs (4 runs in the presence of jamming and 4 runs without the presence of random REBUF jamming) of the system.

IV. ANALYSIS

A. Error Rate

Out of the 8 tests runs shown in table I, the average packet loss percentage in the presence of no jamming was .06875% and the average packet loss percentage in the presence of jamming was .07275%. We determined packet error rate by comparing the known file to the received file. Interestingly enough, there is an instance in trial 2 above where there was higher packet loss in the presence of no jamming than that in the presence of jamming.

The reason for the packet loss lies within the detection logic of the receiving SDR. As described in section II-E, if a peak is missed during the detection portion of REBUF it is enough to corrupt the packet and the decoding algorithm will fail to produce the original code word. Ironically, the presence of noise can help our system. Noise can add energy to a real mark that would otherwise fall below the detection threshold, giving that frequency enough energy to be correctly detected. While the packet loss percentage is extremely low, there are several possible methods to correct this issue. For example, we can further reduce the energy thresholds in one or each of the detection logic passes. While this would reduce the chance of not detecting a peak, it could also increase the decode time.

B. Throughput

The full encoding and transmission took an average of 29.5 seconds to send the 10.3KB file while the receiver took an average of 30.1 seconds to gather all packets. This gives a throughput of about 2.75 Kbps. While this data rate is extremely low, we discuss several optimization ideas in section V. Even after further optimization, the throughput of REBUF will not be able to compete with the traditional jam resistant means of communication discussed in section I-A. However, even low data rates are sufficient to exchange a new shared secret and move back to traditional jam resistant techniques.

V. FUTURE WORK

REBUF demonstrates that jam resistant communication without a pre-shared secret is both possible and practical to implement. However, there are numerous areas for further research and optimization of the REBUF system.

A. Optimization

A large opportunity for future work is the optimization of REBUF. The current implementation does not allow for multi-threading or multi-processing so the receiver has to gather all encoded packets of a transmission before it can decode any packets. The implementation of multiprocessing would drastically cut down the decoding time by enabling parallel processing. Through the use of multiple cores the compilation of packets and the decoding of packets can occur simultaneously.

The underlying assumption that a mark is indelible is largely determined by the peak detection algorithm. We chose to implement a minimum threshold detection which worked well in experimentation. However, other detection thresholds are described in [8], [17]. Further study is required to fully quantify the energy costs required for an enemy to invalidate this assumption.

A more effective decoding algorithm would be able to logically separate the code words produced from packets with more than one code word. There is also a need for a better addressing system for the BBC algorithm as a whole in order to support multiple users or superimposed concurrent coding. Decoding packets with multiple code words introduces a new problem of ordering. Effective addressing of code words will be essential in order to know where to place them. A potential solution for addressing is the implementation of check bit patterns specific to each pair of nodes. This would enable identification of code word ordering within all those decoded from the super-imposed packet.

Future work should also explore increasing the systems total bandwidth. Using a larger frequency range would have a multitude of benefits, including increasing the necessary power requirements of a malicious attacker.

B. Higher Layer Networking Services

In our current experimental setup REBUF does not utilize any networking layers above Data Link layer. There is great value in implementing higher layer protocols, specifically those of the Network and Transport layers. Such an implementation could potentially improve the reliability, accuracy, and throughput of the current system. Implementing higher layers would also enable a greater expansion of the current system to include a larger communication network. It should be possible to implement higher layers using existing protocols, though more study of any cross-layer effects is needed.

1) Network Layer: The purpose of the network layer is the routing of packets from source to destination within a network. In the current system, there is no need for the network layer because we only have two nodes. The addition of more nodes into the current system will create a greater need for routing functionality from the network layer, especially in multiple hop scenarios. Using ad-hoc networks with multiple hops may actually help REBUF stay within the constraints discussed in section I-C. The current experimental set up does not support any of this type of functionality.

2) Transport Layer: The purpose of the transport layer is to enable end-to-end communication over a network. Functions of the transport layer include the following: reliable end-to-end transmission, ordered end-to-end transmission, and flow control. The current system does not have reliable or ordered end-to-end transmission and does not have any measures for flow control. Utilizing TCP at the transport layer would allow us to remove the manual sending rate on the transmitter. It would also allow us to trade off occasional packet loss for a higher average throughput. Using TCP to implement flow control would also decrease the probability of internal jamming within the network. However, more work would be needed to quantify the full effects of using existing congestion control algorithms within TCP.

C. Security Services

REBUF addresses a serious need in wireless communications for availability. However, the current implementation made no effort to address other security services such as confidentiality or integrity. These services are needed in many real world scenarios and should be addressed in future work.

VI. Conclusion

In this paper we presented REBUF, a Jam **Re**sistant **B**BC based **U**ncoordinated **F**requency Division Multiplexing system. Early implementation of REBUF demonstrates the successful transmission of data using the system as well as a significantly higher resilience to jamming than existing OFDM wave forms. While further work is required to determine the maximum throughput and further quantify the resilience to jamming, the implementation of REBUF presented in this paper is proof of concept for a jam resistant form of uncoordinated communication requiring no pre-shared secret.

Acknowledgment

The views expressed in this article are those of the authors and do not reflect the official policy or position of the Department of the Army, Department of Defense, or the U.S. Government.

References

[1] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, and G. Pantziou, "A survey on jamming attacks and countermeasures in wsns," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 4, pp. 42–56, 2009.

[2] J. T. Chiang and Y.-C. Hu, "Cross-layer jamming detection and mitigation in wireless broadcast networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 19, no. 1, pp. 286–298, 2011.

[3] C. Popper, M. Strasser, and S. Capkun, "Anti-jamming broadcast communication using uncoordinated spread spectrum techniques," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 5, pp. 703–715, 2010.

[4] A. Liu, P. Ning, H. Dai, Y. Liu, and C. Wang, "Defending dsss-based broadcast communication against insider jammers via delayed seed-disclosure," in *Proceedings of the 26th Annual Computer Security Applications Conference*. ACM, 2010, pp. 367–376.

[5] L. Milstein, "Interference rejection techniques in spread spectrum communications," vol. 76, no. 6, 1988, pp. 657–671.

[6] W. L. Bahn, L. C. Baird III, and M. D. Collins, "Jam resistant communications without shared secrets," in *Proceedings of the 3rd International Conference on Information Warfare and Security (ICIW08)*, 2008, pp. 24–25.

[7] W. B. Dino Schweitzer, Leemon Baird, "Visually understanding jam resistant communication," *US Air Force Academy, Academy Center for Cyberspace Research, Tech. Rep. USAFA-TR-2009-ACCR-01*, 2008.

[8] L. Baird, W. Bahn, and M. Collins, "Jam-resistant communication without shared secrets through the use of concurrent codes," *United States Air Force Academy, Tech. Rep. USAFA-TR-2007-01*, 2007.

[9] W. L. Bahn, *Concurrent code spread spectrum: theory and performance analysis of jam resistant communication without shared secrets*. University of Colorado at Colorado Springs, 2012.

[10] L. C. Baird, M. C. Carlisle, W. L. Bahn, and E. Smith, "The glowworm hash: Increased speed and security for bbc unkeyed jam resistance," in *MILCOM 2012-2012 IEEE Military Communications Conference*. IEEE, 2012, pp. 1–6.

[11] L. C. Baird and B. Parks, "Exhaustive attack analysis of bbc with glowworm for unkeyed jam resistance," in *MILCOM 2015-2015 IEEE Military Communications Conference*. IEEE, 2015, pp. 300–305.

[12] W. L. Bahn and L. C. Baird III, "Extending critical mark densities in concurrent codecs through the use of interstitial checksum bits," *US Air Force Academy, Academy Center for Cyberspace Research, Tech. Rep. USAFA-TR-2008-ACCR-02*, 2008.

[13] M. D. Collins and W. B. Johnson, "Impulsive noise immunity of multidimensional pulse position modulation," *arXiv preprint arXiv:1805.08120*, 2018.

[14] L. C. Baird III and W. L. Bahn, "Parallel bbc decoding with little interprocess communication," *US Air Force Academy, Academy Center for Cyberspace Research*, 2009.

[15] W. L. Bahn and L. C. Baird III, "Hardware-centric implementation considerations for bbc-based concurrent codecs," *United States Air Force Academy, Academy Center for Cyberspace Research, Tech. Rep. USAFA-TR-2008-ACCR-03*, 2008.

[16] W. Bahn, L. Baird, and M. Collins, "The use of concurrent codes in computer programming and digital signal processing education," *Journal of Computing Sciences in Colleges*, vol. 23, no. 1, pp. 174–180, 2007.

[17] W. Press, W. Dally, D. Eardley, R. Garwin, and P. Horowitz, "An unconventional, highly multipath-resistant, modulation scheme," MITER, Tech. Rep., 1997. [Online]. Available: https://apps.dtic.mil/dtic/tr/fulltext/u2/a331647.pdf

[18] C. Pöpper, M. Strasser, and S. Capkun, "Jamming-resistant broadcast communication without shared keys." in *USENIX security Symposium*, 2009, pp. 231–248.

[19] M. Strasser, C. Popper, S. Capkun, and M. Cagalj, "Jamming-resistant key establishment using uncoordinated frequency hopping," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, 2008, pp. 64–78.

[20] M.-G. Di Benedetto, L. De Nardis, M. Junk, and G. Giancola, "(uwb) 2: uncoordinated, wireless, baseborn medium access for uwb communication networks," *Mobile networks and applications*, vol. 10, no. 5, pp. 663–674, 2005.

[21] L. Xiao, H. Dai, and P. Ning, "Jamming-resistant collaborative broadcast using uncoordinated frequency hopping," *IEEE transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 297–309, 2011.

[22] ——, "Mac design of uncoordinated fh-based collaborative broadcast," *IEEE Wireless Communications Letters*, vol. 1, no. 3, pp. 261–264, 2012.