Visually Understanding Jam Resistant Communication

Dino Schweitzer, Leemon Baird, and William Bahn,
Academy Center for Information Security (ACIS),
United States Air Force Academy

Abstract— **The primary goal of information security is to ensure the confidentiality, integrity, authenticity, and availability of information. Availability is often relegated to a discussion of denial of service attacks on network resources. Another form of denying availability is to prevent communication through the use of traditional jamming techniques. At the United States Air Force Academy Center for Information Security, we have been working on a new algorithm, BBC, which is based on a new type of coding theory known as concurrent codes, that is resistant to traditional jamming techniques. While the formal definition and proofs of concurrent codes can be daunting, the algorithm's effectiveness can be easily conveyed and appreciated through visual demonstration. This paper briefly introduces concurrent codes and describes an interactive applet that visually demonstrates the algorithm's effectiveness in a noisy environment.**

Index Terms—**Coding Theory, Jam Resistant Communication, Visualization.**

I. INTRODUCTION

A. The Problem

Traditional omni-directional techniques for resisting jamming all assume that the sender and receiver share a secret key that is unknown to the attacker [1]. They use spread spectrum communication methods, where there is a sequence of frequency hops, or a sequence of chips, or a sequence of pulses, which are all unpredictable to the attacker [2, 3, 4]. It is this unpredictability that provides resistance to jamming. To jam the signal, the attacker must use far more energy than the legitimate sender.

This approach with a shared secret works well on a small scale. If a small group of people need to communicate by radio, then they can share a secret key beforehand. However, it breaks down completely on a large scale. For example, civilian GPS signals have no jam resistance. This is unfortunate, because the FAA has stated that the commercial airline industry will transition to using civilian GPS for all of its navigational aids. If a terrorist jams the signal at a major airport, it could cause serious problems.

Another example is the civilian cell phone system. It is currently possible to buy a small cell phone jammer on the internet [5]. The cell phone system has no resistance to jamming whatsoever, because it would be unthinkable to have every subscriber share a single shared secret (which wouldn't remain secret for long), or to give every subscriber a different secret key, and then have all the millions of keys loaded into every cell phone tower, and have it listen on millions of channels simultaneously.

Another example is the fact that the Air Force has stated that the future of warfare is net-centric warfare [6]. This means that every vehicle and device in theater will be a node in a wireless, ad hoc internet. If the Air Force plans to rely heavily on this network, then it is critical that it be resistant to jamming. However, a jam resistance system based on a shared key will not scale up. It is not practical to create a single shared key, and to distribute it to every person, vehicle, radio, and device that will be deployed to the theater, and to assume it will remain secret. This scaling problem is why the Common Access Card (CAC) is based on public key cryptography rather than on symmetric cryptography that uses a shared secret. Just as the 1970s saw the invention of public key crypto, which scaled better than symmetric crypto, so we currently have a critical need for the equivalent for jam resistance [7].

B. BBC and Concurrent Codes

The first system ever proposed for this problem is the BBC algorithm. It allows jam resistance without a shared secret. Many radios can broadcast messages simultaneously, and the receiver will receive all of the messages without error. It can be built on top of any of the three most common forms of spread spectrum: frequency hopping, direct sequence, or pulse-based Ultra Wide Band.

The BBC algorithm is based on a new field of coding theory, concurrent codes. These are a subset of superimposed codes, and are very different from traditional error detecting or error correcting codes. A number of theorems have been proven about these codes, and about the BBC algorithm and other algorithms based on these codes. This math can be fairly complex, which is why some form of visualization is useful for this.

The BBC algorithm works as follows. First, there must be a way to send an *indelible mark* at chosen *locations*. If BBC is run on a pulse-based UWB system, then the *indelible mark* is a

short pulse of very high power RF noise that spans a large spectrum of frequencies. The *location* of the mark is the exact time of the pulse. There is no information encoded in the random noise of the pulse itself. All of the useful information is encoded in the exact timing of the pulse. Because the pulse is very short, very random, and very powerful, it is not possible, in any practical sense, for an attacker to erase a pulse, in the way a sine wave is erased by destructive interference when an identical wave is sent exactly out of phase with it. Therefore, an attacker can only create new pulses, not erase existing pulses.

In pulse-based BBC, the message is sent by appending several zeros which act as checksum bits to a binary message, hashing each possible prefix of the message, and sending one pulse for each prefix at a location determined by the hash. For example, to send the message 1011, it would first append zeros to get 1011000, then it would take each prefix of the message {1, 10, 101, 1011, 10110, 101100, 1011000}, then it would send 7 pulses, where the exact time of each pulse is determined by a secure hash of each of those 7 prefixes as shown graphically in Figure 1.
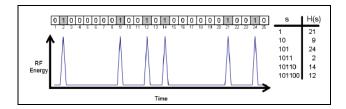

Figure 1. Encoding messages as pulses in time.

The receiver decodes the message by first checking whether there are pulses at the locations determined by the hash of 0 and the hash of 1. If there is a pulse at time $H(1)$ but none at $H(0)$, then it knows there was at least one message sent starting with 1, but no messages sent starting with 0. It then checks for pulses at time $H(10)$ and $H(11)$ to find the second bit of the message. It will end up following an entire tree of possible decodings, which ultimately yields all the messages that were sent simultaneously.

A full understanding of the encoding and decoding of BBC, and all the properties of it, are not easily grasped from just the English description. To truly comprehend it, there needs to be an appropriate demonstration to allow a person to experiment with the algorithm and see the results. Further information on the algorithm and its properties can be found at [8].

C. Possible Implementation

The BBC algorithm can be implemented on a software radio using any of the three main types of spread spectrum. This is how it would most likely be used in practice. However, humans cannot directly perceive radio signals, doing so does not give much insight into the algorithm.

A slightly better approach is to implement it with sound waves instead of radio waves. Such a system allows the audience to hear the pulses, and demonstrates how two messages can be sent simultaneously without interference. But it still doesn't give much insight into how the system works internally.

Perhaps the best demonstration is a visual one. Instead of using radio frequency pulses for each mark, the marks can be represented as black pixels in a white image. This allows the user to actually see the encoding. The decoding tree can also be shown visually, so that the user understands the amount of computation being performed. It can also allow interactive use by the user, where the user draws additional noise on the picture in an attempt to jam it. This gives insight into the robustness of the system. For these reasons, the visual approach appears to be the most useful for gaining an understanding of the system.

II. UNDERSTANDING CONCURRENT CODES

A. The Challenge

While concurrent codes offer many advantages to traditional approaches for jam resistant communications, these benefits are not always readily understood by decision-makers and possible project funders. The basic algorithm for encoding and decoding can be quickly explained, but the resiliency to additional noise in the system is difficult to appreciate based solely on the mathematical analysis. This is especially true for a less technical audience, such as some higher level management and fiscal managers.

The obvious approach to demonstrate the effectiveness of the algorithm is to simply have two communication devices talking using the algorithm and show that they can continue to operate, even under noisy conditions. Unfortunately, having two software radios, or computers, communicate is not a very impressive demonstration. As discussed previously, the algorithm is "hidden" in the software, and the noise level is hard, if not impossible, to judge. The user simply sees messages being sent and received correctly. Even if additional information about noise levels, such as some visual presentation of frequencies, is presented, the user has a difficult time understanding the significance of what they are seeing.

To address this challenge in communicating the effectiveness of the BBC algorithm, the ACIS research group developed a demonstration that would be meaningful and understandable to a less technical audience. The goal was for viewers to easily understand how resistant to noise the algorithm was, and to be able to appreciate some of the more subtle characteristics of the algorithm, such as how computational costs grow with noise.

B. An Audio Solution

The first attempt at developing an understandable

demonstration of the BBC algorithm was to use sound pulses as the communication medium. Three laptop computers were set up close to each other with two sending messages while the other received them. To send a message, the desired message was encoded as a series of high-pitched "beeps" set in time based on the hash of the message prefix. At approximately the same time, each sending computer would start sending their encoded messages simultaneously as audio beeps transmitted through the laptop speaker.

The receiving laptop would be "listening" for beeps through the attached microphone. Each beep received would be recorded. After recording long enough to receive all of the beeps, the receiver would decode the received messages using the BBC algorithm interpreting each beep as a received bit in the message and display the results.

This demonstration was successful in demonstrating how two senders could simultaneously send messages that could be correctly decoded by a receiver. It also made the concept of "noise" easy to understand, since in addition to the two senders simultaneously beeping, the receiver was also recording any background noise such as people talking, equipment noise, etc. Another advantage of this demonstration is that it shows the algorithm's robustness to inexact timing. The two senders did not start at exactly the same moment in time, and the timing calculations for when to send beeps versus when beeps were received are slightly inexact for the three laptops. However, the receiver is able to correctly decode messages in spite of these variations.

While a powerful and useful demonstration, a disadvantage of the audio version is that it still does not give the viewer a clear understanding of the amount of noise in the system and exactly how resilient the algorithm is. It is easy to understand that there is additional noise, but not the amount or the effect of the noise to computation cost. To achieve this understanding, the group set out to develop a visual demonstration. The goal was to have a demonstration that was easy to understand, visually meaningful, and interactive.

III. VISUAL REPRESENTATION OF MESSAGE SPACE

The first step in developing a visual demonstration was to come up with a meaningful representation for encoded messages that was visually compact and understandable. The description of the algorithm often uses the specific implementation of pulses on a timeline as an easy to understand way of encoding messages. Pulses represent bits that can be turned on, but not off. The user can envision this as a linear timeline with pulses at set points in time as was shown in Figure 1. Multiple messages are encoded by adding pulses at the appropriate point in time. Noise is represented as pulses that are not part of the encoded messages. The number of pulses, and thus density of the timeline, represents how noisy the message space is.

A variation of this visual analogy was used for the visual demonstration. The linear timeline is "wrapped around" as successive rows of a bit map. Rather than pulses represented

as visual "spikes", a single dot on the bit map represents the presence of a pulse at that point in time. Bits can be turned on as additional messages and noise are added to the system, but bits can not be turned off. Using this analogy, the visual denseness of the bit map image represents how noisy the message space is (see Figure 2).
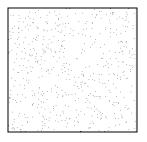

Figure 2. Bit map representation of bits in message space.

In addition to the compactness, simplicity, and understandability of this representation, there is another advantage to using the bit map approach. Adding noise to the system is equivalent to turning on random bits in the bit map. Familiar paint tools (pencil, spray can, and solid rectangle) can provide this capability to the user without needing lengthy explanation.

IV. BINARY TREE FOR DECODING

Along with representing the message space, it was desirable to show how much computation was required to decode the message(s), and the impact of multiple messages and noise in the system. The analogy of message decoding to searching a binary tree was used for this purpose. As each prefix of the message is decoded, there are two possible "next bits" in the message whose presence needs to be checked. Thus, decoding the message is analogous to searching a binary tree. If both bits are present in the message space, the tree continues to grow. If one of the bits is not present, that branch of the tree is disregarded for further consideration. The number of tree nodes represents how much computation is necessary to decode the message(s). As additional messages and noise are added to the message space, the size of the tree grows, both for actual messages as well as false paths to follow. A sense of the relative amount of computation can be quickly gleaned from the visual representation of the tree (see Figure 3).
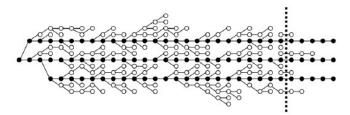

Figure 3. Binary decoding tree.

V. BBCVIS INTERACTIVE APPLET

The bit map message space representation and binary tree

computation metaphor is combined in BBCVis, an interactive applet for experimenting with and understanding concurrent codes. Figure 4 shows the initial layout for the applet. Messages are entered into the message box and encoded into the bit image. Decoded message are displayed below along with the associated decoding binary tree on the right. Figure 5 shows the applet after four messages have been entered. The density of the bit map image is updated as images are encoded, and represent less than 1% in the example shown. The decoding tree shows that very few additional nodes were searched beyond the actual messages (nodes resulting in terminated branches are shown as hollow circles). The four paths surviving to the bottom of the tree represent the four messages that were decoded. The dotted line on the decoding tree panel represents the start of the checksum bits which were appended to each message before encoding.
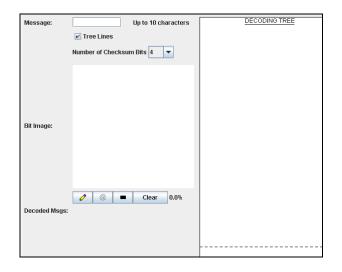

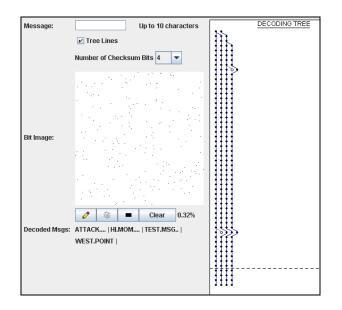Figure 4. Initial screen shot of the BBCVis applet.

Figure 5. Four encoded messages.

Figure 6 shows the effect of "adding noise" to the system. Standard bit map painting tools allow the user to add significant number of additional bits to the message space. As a result of this additional noise, the decoding tree has grown to account for the additional nodes that needed to be searched to find the final decoded messages. An indication of the robustness of the algorithm is that a significant increase in the bit density of the message space (25%) has resulted in only a modest increase in the number of binary tree nodes, less than twice as many nodes to search. This is easy to quickly see by the change in the size of the displayed tree. In addition, the four original messages are correctly recovered.
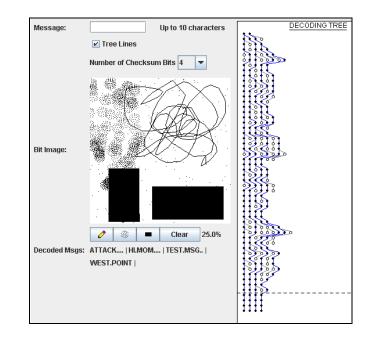

Figure 6. Encoded messages with noise added to message space.

Figure 7 shows how the algorithm behaves under extreme noise conditions. With almost 50% density in the message space (half of the bit image covered), the number of nodes in the decoding tree, and thus computation time, has grown significantly. The original four messages are still recovered, but additional "hallucinations" or false messages are also decoded. One way of removing the hallucinations is to use additional checksum bits when encoding the messages. Figure 8 shows the same four original messages, along with the same amount of noise (~50%). However, by increasing the number of checksum bits from four to six, no hallucinations survived, and the original four messages were correctly decoded. The size of the tree, or amount of computation, is approximately the same in both cases. Figure 9 shows what happens at 100% noise level (completely black image). The tree shows the expected exponential growth (for practical purposes, the algorithm ceases building the tree after a predefined number of maximum nodes at any level occurs).
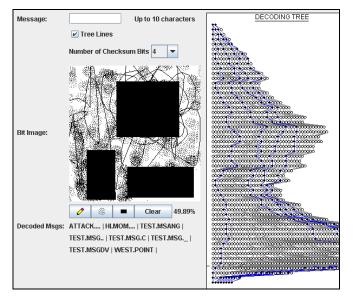
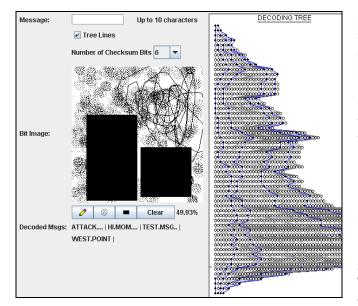Figure 7. Hallucinations occurring from excessive noise.

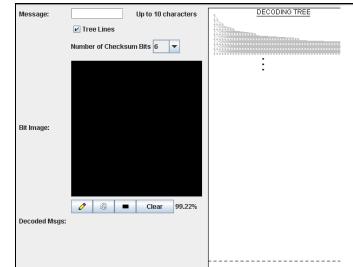Figure 8. Increased number of checksum bits eliminating hallucinations.

Figure 9. Maximum noise density.

In an effort to make the visualization as dynamic and informative as possible, we have made the decoding portion of the visualization occur in real-time as messages are entered, or noise is added to the bit image. This allows the user to immediately see the impact on computation as the message space is changed. Perhaps the most dramatic effect occurs when drawing a solid rectangle. The tree grows and shrinks as the rectangle is resized dynamically showing the sensitivity of the computation to the amount of noise in the system.

VI. OUR EXPERIENCE

The BBCVis applet has been demonstrated to several individuals at different levels of familiarity with the algorithm. The concept of a message encoded as bits in the image and decoded seems to be quickly grasped. The use of the drawing tools to add noise is also quickly understood. The significance of the tree is less intuitive without explanation. However, the concept that the number of tree nodes and visual density relates to the amount of computation time is rapidly appreciated.

In addition to the basic understanding, several viewers expressed similar reactions to interacting with the tool. For example, many are surprised at how much noise can be added without significantly affecting the amount of computation or corrupting the decoded messages. Similarly, many noted how the sensitivity increases dramatically at key noise densities, such as 50%.

In summary, the BBCVis applet is effective in describing the concurrent code algorithm and characteristics. Users are able to quickly understand the visual metaphors and effectively interact with the applet to experiment with and understand the underlying algorithm.

REFERENCES

[1] L. B. Milstein. Interference rejection techniques in spread spectrum communications. Proc. IEEE, 76(6):657–671, June 1998.

[2] E. G. Kanterakis. A novel technique for narrowband/broadband interference excision in ds-ss communications. In MILCOM '94, volume 2, pages 628–632, 1994.

[3] C. Bergstrom and J. Chuprun. Optimal hybrid frequency hop communication system using nonlinear adaptive jammer countermeasures and active fading mitigation. In IEEE Global Telecommunications Conference, 1998. GLOBECOM 98. The Bridge to Global Integration,volume 6.

[4] I. Bergel, E. Fishler, and H. Messer. Low complexity narrow-band interference suppression in impulse radio. In Proceedings of the 2003 International Workshop on Ultra Wideband Systems (IWUWBS), Oulu, Finland, June 2003.

[5] http://www.globalgadgetuk.com/Personal.htm.

[6] H. Raduege JR. Net-centric warfare is changing the battlefield environment. CrossTalk. http://www.stsc.hill.af.mil/crossTalk /2004/01/ 0401Raduefe.html, accessed March 2007.

[7] W. Diffie and M. E. Hellman. New directions in cryptography. IEEE Trans. Inform. Theory, IT-22(6):644–654, 1976.

[8] L. Baird, W. Bahn, and M. Collins, Jam-resistant communication without shared secrets through the use of concurrent codes, US Air Force Academy Technical Report, USAFA-TR-2007-01, February 2007.