Proceedings of the 2007 IEEE
Workshop on Information Assurance
United States Military Academy, West Point, NY 20-22 June 2007

Keyless Jam Resistance

Leemon C. Baird III, William L. Bahn, Michael D. Collins, Martin C. Carlisle, Sean C. Butler
Leemon.Baird@usafa.af.mil
United States Air Force Academy

Abstract—
Traditionally, omnidirectional, radio frequency (RF) communication has been made resistant to jamming by the use of a secret key that is shared by the sender and receiver. There are no known methods for achieving jam resistance without that shared key. Unfortunately, wireless communication is now reaching a scale and a level of importance where such secret-key systems are becoming impractical. For example, the civilian side of the Global Positioning System (GPS) cannot use a shared secret, since that secret would have to be given to all 6.5 billion potential users, and so would no longer be secret. So civilian GPS cannot currently be protected from jamming. But the FAA has stated that the civilian airline industry will transition to using GPS for all navigational aids, even during landings. A terrorist with a simple jamming system could wreak havoc at a major airport. No existing system can solve this problem, and the problem itself has not even been widely discussed.

The problem of keyless jam resistance is important. There is a great need for a system that can broadcast messages without any prior secret shared between the sender and receiver. We propose the first system for keyless jam resistance: the BBC algorithm. We describe the encoding, decoding, and broadcast algorithms. We then analyze it for expected resistance to jamming and error rates. We show that BBC can achieve the same level of jam resistance as traditional spread spectrum systems, at just under half the bit rate, and with no shared secret. Furthermore, a hybrid system can achieve the same average bit rate as traditional systems.

I. Introduction

Jam resistance is becoming increasingly important to modern communication. It is increasingly important to the military, as operations become more dependent on real-time, wireless communication, and even brief denial of service attacks become more damaging. It is increasingly important in the civilian sector, now that cheap, tiny cell phone jammers are becoming available. It is becoming more important for civilian GPS signals, because they are used by airlines and others, so jamming could cause serious problems. As the world moves toward software defined radios, the tools to jam police and emergency frequencies will become increasingly widespread. Jam resistance is a critical issue.

Traditionally, jam resistance has been achieved through spread spectrum communication. Each of the three common forms of spread spectrum can be made resistant to jamming if the sender and receiver share a secret key. Much attention has been given to performance against jamming and mitigation techniques in spread spectrum systems both generally [1], [2], [3] and in specific subfamilies including direct sequence [4], [5], [6], [7], [8], [9], [10], frequency hopping [4], [11], and pulse-position systems [12], [13], [14].

In *frequency hopping*, the sender and receiver communicate on a single frequency for a short period, then jump to another frequency. Bluetooth uses frequency hopping, and changes frequencies every 0.625 ms. If the sequence of frequencies is a cryptographically-secure, pseudorandom sequence determined by a secret key, then an attacker will not know which frequency to jam at any given time. The attacker must therefore flood every possible frequency with energy to guarantee that every part of the message is destroyed. This requires an enormous amount of energy. A smart jammer can usually succeed by destroying only a portion of the message, which requires fewer frequencies to be jammed, and so requires less energy. But it still requires the attacker to expend far more energy than the legitimate users. This can be expensive, can require large devices, and can make it easier to detect and triangulate the location of the attacker. This asymmetry in energy usage is the standard goal for a jam-resistant wireless communication system.

In *direct sequence*, a sender uses all frequencies simultaneously, by combining the message with a pseudorandom bitstream, generated according to some key. This is used in CDMA (Code Division Multiple Access) cell phones [15], [10], [16], [17], [9], [18]. If the key is known, then the attacker can predict the sequence, and can broadcast a strong signal using that sequence, thus masking the signal sent by the legitimate user. If the key is secret and a strong sequence is used, an attacker is forced to expend far more energy than the legitimate sender. A related approach is Orthogonal Frequency Division Multiplexing (OFDM) CDMA spread spectrum [19], [20] which uses multiple, orthogonal frequencies simultaneously.

In *pulse-based* systems, the sender broadcasts a series of very short pulses, each of which essentially spreads noise over the entire spectrum. The message is encoded in the exact timing of the pulses, and this encoding can be dependent upon a secret key if jam resistance is desired. This system is used in many of the new Ultra Wide Band (UWB)

1-4244-1304-4/07/$25.00 ©2007 IEEE

systems that are in the early stages of development and deployment [21], [22], [23], [24], [25], [26], [27], [28]. Although the pulse contains a fairly small amount of total energy, it is so short that the power is very high. This makes it difficult for a jammer to mask the pulse without using an enormous amount of energy to broadcast continuously. If the key is known, then the attacker can easily send pulses corresponding to the zeros in the message, and the receiver will receive a message consisting entirely of 1 bits. If the key is unknown, then the attacker must resort to sending pulses during every time period to destroy all of the message. Again, a smart jammer can succeed by only destroying a portion of the message, but it can still require far more energy than is used by the legitimate sender.

In each of these three systems, it is possible for multiple users to transmit without jamming each other, as long as each user is polite and uses a different channel (a different seed for the pseudorandom generator). However, an attacker will not be polite, and can easily jam the system if the channel is known. In each of these three systems, jam resistance has only been possible by using a secret key to control the sequence of frequencies, or the chip sequence, or the timing of the pulses. If the secret is known, then an attacker can concentrate energy in the channel being used (defined by the key), and easily jam the communication with little energy expenditure. That is why the channel must remain a secret.

Unfortunately, secret keys are not scalable to large systems. If the civilian GPS signal used a key, then that key would have to be distributed to all 6 billion potential users, including the attackers. There is no way to use a secret key to protect such a signal that is meant to be public.

Handheld jammers for cell phones can now be bought on the open market for under $200 [29]. Theoretically, cell phones could be made jam resistant by using a separate secret key for each customer. But then each cell phone tower would have to store the secret keys for every customer in the world who might enter that cell. The tower would also have to continuously monitor all of those millions of channels. The cell phone system is at least based on fixed locations, but the increasing use of ad-hoc networks in sensitive and mission-critical roles raises significant issues in how those networks will be protected against jamming and other malicious attacks [30], [31], [32], [33], [34], [35].

The scaling problem is even worse for military applications. The future of the US Air Force is said to be net-centric, joint, and coalition. The current vision is that the entire battlefield will be a single ad-hoc wireless network, with continuous packet forwarding between devices owned by different services, and even by different countries. That would require that a single secret key be used by all of the equipment in the entire theater. If an enemy captured even a single handheld radio or micro-UAV, it would allow jamming of wireless communication throughout the entire the theater until new keys could be reloaded.

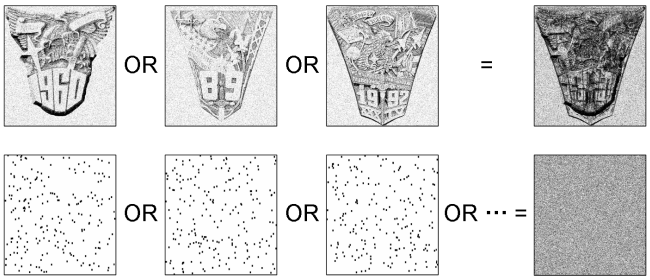

Fig. 1. Superimposed pictures (top) and BBC codewords (bottom)

The current situation is analogous to the state of cryptography in the early 1970s. Symmetric ciphers based on secret keys were believed to be secure, but the infrastructure for key management did not scale well. The invention of asymmetric cryptography [36], [37], [38], [39] allowed the development of a Public Key Infrastructure (PKI) which did scale well and allowed efficient key distribution [40], [41], [42], [43]. There is now a critical need for the equivalent of PKI for jam resistance.

Specifically, there needs to be some way that two strangers with no shared secret can exchange messages over a wireless medium in such a way that an attacker positioned between them cannot easily prevent those messages from arriving. Ideally, the attacker should be able to inject new messages into the stream, but not block or modify any legitimate messages. Attacks on traditional shared-secret jam-resistant communication require the attacker to expend far more energy than the legitimate user, flooding the entire spectrum with energy. This can be difficult for the attacker to achieve, and it makes it easier to locate and deal with the attacker. The need now is for an equivalent level of jam resistance without the shared secret.

No existing system achieves this goal. One might imagine building a system based on error correcting codes, but existing error correcting codes cannot even decode two messages that have been sent simultaneously and are combined with a bitwise OR. Various other codes such as locally decodable codes [44] or X codes [45] or separating codes [46] appear to be even less useful in this case. Superimposed codes deal with that situation, but traditionally have not allowed efficient decoding. If many messages are encoded and then combined with a bitwise OR, it generally appears difficult to recover all of them, especially when the space of possible messages is exponentially large. The difficulty of this problem is illustrated in figure I.

We propose the first system for jam-resistant, omnidirectional, wireless communication without a shared secret. It works by defining codewords that can be combined with a bitwise OR, and then reliably and efficiently decoded.

II. The BBC Algorithm

The general BBC algorithm is given in Algorithm 1 (for sending) and 2 (for receiving). The broadcast algorithm tells how to encode and send a single message. The decoding algorithm assumes that multiple, encoded messages were sent simultaneously, combining their binary strings with a bitwise OR, to form a combined string known as a *packet*. In addition, random noise (or the results of an active attack) will flip some of the packet's bits from 0 to 1 (but never from 1 to 0). Under these assumptions, the packet will have the same number of bits as a codeword, but typically many more of the bits will be 1 in the packet than in a single codeword.

The constant k controls the number of checksum bits to use (the bits themselves are simple zeros, but they force the hash function to create a strong checksum). The hash function H maps an arbitrary-length bit string to a location. The general BBC algorithm requires the making of *indelible marks* at *locations* chosen by hashes of all prefixes of the message to be sent. The exact definition of *indelible mark* and *location* depends on the system being used, but the requirement is that both the sender and attacker can make such marks, but neither can erase them. So it is equivalent to a long string of bits that are initially all zero, and the sender and attacker can each change any 0 to 1, but cannot change a 1 to 0. The attacker could jam the communication by setting all the bits to 1, but if there is an energy cost associated with each 1 bit, and if the legitimate sender is setting only a thousandth or a millionth of the bits to 1, then the attacker will be forced to expend a thousand or million times more energy than the sender.

Algorithm 1 BBCbroadcast(M)

This function broadcasts an m-bit message $M[1 \ldots m]$ adding k checksum bits to the end of the message. H is a hash function. The definition of H and the values of m and k are public (not secret). The definition of "indelible mark" and "location" are specific to the physical instantiation of BBC used.

Append k zero bits to the end of M
for $i \leftarrow 1 \ldots m+k$ **do**
  Make an *indelible mark* at the *location* given by $H(M[1 \ldots i])$
end for

Perhaps the simplest instantiation of BBC is when pulses are used, as in the most recent Ultra Wide Band (UWB) systems. The sender broadcasts very short, high-power bursts of radio frequency noise at certain times. The message is encoded in the timing. An attacker can also broadcast such pulses, but cannot erase any existing pulses. Such pulses are difficult to erase because they are very short, very high power, and consist of unpredictable random noise with energy spread over the entire spectrum. No known system exists that can detect and analyze such pulses, and then send out an inverse waveform to cancel them. There simply isn't time during the brief period it takes for the pulse to pass the attacker.

Algorithm 2 BBCdecode(n)

This recursive function can be used to decode all the messages found in a given packet by calling BBCdecode(1). There must be a global $M[1 \ldots m+k]$ which is a string of $m+k$ bits. The number of bits in a message is m, and the number of checksum zeros appended to the message is k. H is a hash function. The definition of H and values of m and k are public (not secret). The definition of "indelible mark" and "location" are specific to the physical instantiation of BBC used.

if $n = m + k + 1$ **then**
  output "One of the messages is:" $M[1 \ldots m]$
else
  if $n > m$ **then**
    $limit \leftarrow 0$
  else
    $limit \leftarrow 1$
  end if
  for $i \leftarrow 0 \ldots limit$ **do**
    $M[n] \leftarrow i$
    if there is an *indelible mark* at *location* $H(M[1 \ldots n])$ **then**
      BBCdecode(M,$n+1$)
    end if
  end for
end if

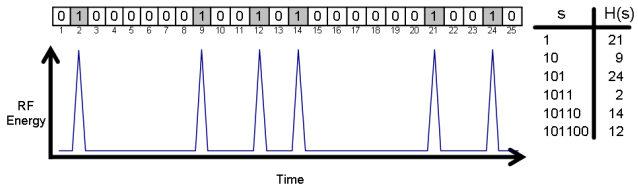

Fig. 2. BBC broadcast using UWB pulses. $M = 1011$, $k = 2$, an *indelible mark* is a radio pulse, and its *location* is the time when the pulse occurs relative to the start of the message. The table on the right shows part of the definition of the hash function $H(x)$.

Figure 2 shows how to broadcast using *BBC-Pulse*, an implementation of BBC for pulse-based broadcast. In this example, the message $\mathbf{M = 1011}$ is padded with $k = 2$ zero bits to get 101100. All prefixes of this are hashed with the hash function defined by the table on the right. A pulse is sent at the time defined by the hash of each prefix of 101100. For example, $H(1) = 21$, so a pulse is sent at time 21, and $H(10) = 9$, so another pulse is sent at time 9.

Figure 3 shows the decoding of that same message for

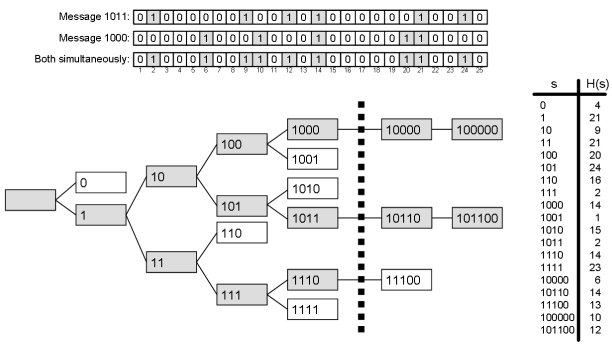

Fig. 3. Decoding tree for *BBC-Pulse* broadcasts when two messages were sent (bottom left), the times when the pulses were sent (top), and part of the hash function definition (right).

BBC-pulse. The receiver starts at the root of the tree (on the far left), which is an empty string. The receiver then repeatedly tries appending both a 0 and 1 bit to each string being considered, and calculates the hash of each new string generated. So on the first round, the receiver calculates H(0)=4 and H(1)=21. Since a pulse was not received at time 4, the receiver knows that no messages starting with 0 could have been sent. Since a pulse was received at time 1, the receiver knows that at least one message was sent that started with 1. These results are shown on the tree by coloring the 0 box white and the 1 box gray.

This process then continues. The string 0 is not expanded because no pulse was detected at time H(0). The string 1 is expanded both ways to get 10 and 11, and the receiver calculates H(10) and H(11). Both of these hashes yield times at which pulses were received, so both boxes are colored gray, and both will be expanded on the next round. This process continues until the entire 4-bit message has been decoded (at the dotted line). Note that at this point, both of the legitimate messages have been found (1011 and 1000), but an additional "message" has also been found: 1110. This third message is a *hallucination*. This hallucination occurred because all three of the strings 11, 111, and 1110 hashed to locations that just happened to contain a pulse for other reasons. This is why the k checksum bits are important. To the left of the dotted line, every box is expanded by appending both 0 and 1. To the right of the line, only 0 bits are appended. This is because every legitimate message is known to have k zero bits at the end. Both of the legitimate messages survive through all k rounds of checking. The hallucination, however, is eliminated because the string 11100 does not hash to a location that happens to contain a pulse.

This system is intended to be used in situations where a very small fraction of the time positions contain pulses. But consider a worst case, where a full third of all positions contain pulses. If a good cryptographic hash function is used, then these pulses will be scattered pseudorandomly.

BBC-pulse
mark - a very short pulse of radio frequency noise
location - the time at which the pulse occurs
BBC-frequency-hopping-simultaneous
mark - a pure sinusoidal signal (all marks sent simultaneously)
location - the frequency of the signal
BBC-frequency-hopping-sequential
mark - a pure sinusoidal signal for a short duration
location - $H(x)$=frequency, length(x)=start time

BBC-direct-sequence
mark - a pseudorandom radio frequency waveform (e.g. a chirp sequence)(all marks sent simultaneously)
location - the seed to use for the waveform generator

Fig. 4. Instantiations of the BBC algorithm for each of the major spread spectrum techniques

Then during the checksum process, at each step a given hallucination will have a probability of only 1/3 of surviving, because its hash will choose a pseudorandom location that has a probability of only 1/3 of containing a pulse. If there are k checksum bits, then the probability of a hallucination surviving until the end is only $(1/3)^k$, which quickly becomes vanishingly small for even moderate values of k. The number of potential hallucinations will be small (as is proved in a later section), therefore, the receiver is very unlikely to end up with any hallucinations at all. This, of course, assumes only random noise and accidental interference between legitimate messages. The analysis of an active attack is more complex, and is given in a later section.

All of the above analysis is for the pulse-based instantiation of BBC, but it applies essentially unchanged to the other instantiations as well. Figure 4 lists instantiations of BBC for each of the three major spread spectrum techniques, including both sequential and simultaneous approaches.

The two simultaneous instantiations (BBC-frequency-hopping-simultaneous and BBC-spread-spectrum) have several interesting properties. In BBC-frequency-hopping, the locations are frequencies, and a mark is a pure sine wave broadcast on the appropriate frequency. In the simultaneous version of it, all of the sine waves are broadcast simultaneously. That would be difficult on older hardware that uses physical circuits or crystals for each frequency, but it can be done easily on modern software defined radios, which are ideal for trying new communications schemes and for both attack and defense [47]. It could also be done easily with a single, custom chip, where a separate part of the circuit is devoted to generating each frequency. Either way, it has interesting properties not found in any system currently in use.

This approach sends information *holographically*. In a

long broadcast, the entire message is encoded in each short period of time. Theoretically, the receiver can listen for a very short period and capture the entire message at once. In practice, the length of the period will depend on the level of background noise and the quality of the receiving hardware. If there is a high level of noise, and the receiver is a small antenna with low-quality amplifiers etc, then the receiver will need to capture and process data from a long period in order to recover the message. But as soon as the noise drops, or the receiver upgrades to a larger antenna and better hardware, the message will be clear much sooner. This suggests an interesting approach sending messages in a way that adapts to available bandwidth. The sender can simply broadcast a message continuously until an acknowledgement is received. It also suggests reduced synchronization problems, since the receiver can start receiving in the middle of a message and still recover the entire message. The direct sequence instantiation of BBC also has this property, with an additional property that the receiver will be able to determine the exact instant that the sender must have started the broadcast, even if the receiver started receiving slightly later. This would be useful for time-based systems such as GPS.

III. Hybrid Communication Systems

Traditionally, public key cryptography (or *asymmetric* cryptography)is more powerful than symmetric cryptography, in the sense that it allows strangers to communicate securely without having previously established a shared key. However, the fastest known public key ciphers are far slower than the best symmetric ciphers. Therefore, cryptographers typically build hybrid systems. When Alice wants to talk to Bob, she first generates a random session key. She then encrypts this with Bob's public key and sends it to him. She might even digitally sign it with her own private key, to prove it comes from her. All this uses a slow, asymmetric cipher, but that is acceptable, since there are only a few hundred bytes to be sent. Once Bob has decrypted the session key, all future communication is done using that session key with the symmetric cipher, which is far faster.

A similar hybrid system should be used for jam-resistant communication, for much the same reasons. If Alice wants to communicate with a stranger Bob, she would first generate a random session key, encrypt it with Bob's public key, sign it with her private key, and send it to him in a jam-resistant way using BBC. Then all future packets in that session would be sent using traditional jam-resistant, spread spectrum techniques, using the session key. The BBC communication may be slightly slower than traditional spread spectrum for any given degree of jam resistance (about half the speed) and it requires more computation. This is perfectly acceptable, since only the first few hundred bytes are sent using BBC. Thus, for long messages, a hybrid system will send most of its bits using traditional, secret-key jam resistant methods, and so will have an average bit rate almost equal to traditional jam resistance.

Parameter Choices			Random Codebook			BBC Codebook		
m	e	$\lg(h)$	M_S	R	k	M_S	R	k
8	23	-24	3	0.1577	24	3	0.1339	16
8	15	-24	2	0.1490	23	2	0.1358	15
100	100	-60	42	0.4297	60	28	0.2877	40
1000	100	-60	65	0.6504	60	38	0.3894	41
10000	100	-60	68	0.6855	59	40	0.4038	41
1000	5	-60	2	0.5818	59	1	0.3904	38
1000	10	-60	6	0.6186	59	3	0.3902	39
1000	100	-60	65	0.6504	60	38	0.3894	41
1000	1000	-60	653	0.6536	59	388	0.3886	43
1000	10000	-60	6538	0.6539	60	3878	0.3879	45
1000	100000	-60	65390	0.6539	59	38708	0.3871	47
1000	100	-120	61	0.6154	119	37	0.3758	79
1000	100	-60	65	0.6504	60	38	0.3894	41
1000	100	-30	66	0.6695	29	39	0.3966	22
1000	100	-20	67	0.6761	19	39	0.3991	15
1000	100	-10	68	0.6828	9	40	0.4016	9

Fig. 5. For particular choices of m, e, and h, this table gives the performance of random codebooks and BBC codebooks, as measured by the number of messages that can be sent simultaneously, M_S, the bitrate efficiency, R, and the number of 1 bits per codeword in excess of the message size, k. Choices are gray when they differ from the example of $m = 1000, e = 100, \lg(h) = -60$.

IV. Analysis of BBC Codes

Recall that the BBC algorithm works by making *indelible marks* at *locations* chosen by hashes of prefixes of a padded message (padded with some number of zero bits at the end). Abstracting away from the radio frequency details, we can define a *BBC code* to be a set of codewords, where the codeword corresponding to a message is a binary string of mostly 0 bits, with 1 bits in the positions chosen by a hash of each prefix of the padded message. In other words, for messages of length m, codewords of length em, and k padding bits, the codeword C corresponding to the message M has its ith bit defined as:

$$C_i = \begin{cases} 1 & \text{if } i = H(M'_{1..j}) \text{ for some } j \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where M' is the message M with k zero bits appended to the end.

A random codebook is a hypothetical codebook where every codeword is created randomly. It assumes infinite computational power, and so is only useful as a theoretical comparison. The performance analysis for the random codebook will not be included here, because of space limitations. Only the analysis of BBC will be given in detail.

Figure 5 gives the results for both a random codebook and a BBC codebook. In this table, the first three columns give various choices for the parameters, and the next 6 columns give the performance that results from those choices. The message is m bits long, the codeword is e times as long as the message, and the goal is to limit the expected number of hallucinations to h. The gray cells show parameter choices that differ from the example case of $m = 1000$, $e = 100$, $\lg(h) = -60$. This gives an idea of how the performance is affected by each of the three parameters.

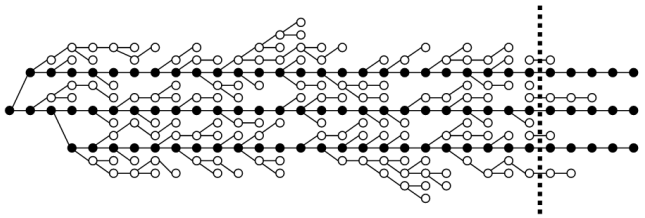

Fig. 6. An example decode tree for BBC showing true messages (black circles) and hallucinations (white).

BBC is illustrated in figure 6, which is an example BBC decoding tree. In this example, there are 3 intentional messages (black circles) and a number of partial hallucinations (white circles), which are all eliminated by the end (far right edge). In this case, the code successfully eliminated hallucinations at the end, but the receiver still had to perform calculations for every partial hallucination. In this diagram, for any given column in the center region of the tree, there are an average of 6 partial hallucinations and 3 messages. So in an average column, there are 9 nodes, 3 of which are the actual message, so the amount of calculation required to consider each node is $9/3 = 3$ times the computational cost as if only messages were considered. This implies a *workload factor* of $w = 3$, meaning there are $w = 3$ strings to consider (both messages and partial hallucinations) per intentional message, on average, for most of the tree.

Note that the tree has 3 regions. In the center region, each column has an average of wM_S nodes. At the far left, there is a small region where each column has fewer than that many nodes. This is because some messages may have the same first few bits, and the number of hallucinations takes some time before its exponential growth reaches its steady state value. At the far right, past the dotted line, the algorithm switches from decoding the message itself to checking the zeros that were appended to the message. From that point on, the number of hallucinations cannot grow, and will on average shrink exponentially, with any given hallucination dying out with probability $1 - \mu_p$ at each step. Since the left and right regions will typically be small compared to the middle region, the computational cost for decoding will be $\theta(mwM_S)$.

It is easy to calculate the steady state workload factor w, from the fact that in the steady state, the number of partial hallucinations (white nodes in one column) should on average remain unchanged in the next column. If a particular column has on average wM_S nodes, of which M_S are message prefixes and $(w-1)M_S$ are partial hallucinations, then the next column to the right will have those same M_S messages plus several hallucinations. Each hallucination node at one time step gives rise to an average of $2\mu_p$ nodes on the next step, because there is a probability μ_p that appending a 0 will hash to a location containing a mark, and a probability μ_p that appending a 1 will hash to a mark location. Each message gives rise to a hallucination with probability of only μ_p, since the other possible bit to append gives a legitimate message prefix. If this is at steady state, then w should remain unchanged on the next time step, so we can equate the number of hallucinations on one time step to that on the next step:

$$(w-1)M_S = M_S\mu_p + 2(w-1)M_S\mu_p \tag{2}$$

Solving that for w gives:

$$w = \frac{1-\mu_p}{1-2\mu_p} \tag{3}$$

Note two important properties of this equation. First, when $\mu_p = 1/2$, the value of w goes to infinity. This means that the number of partial hallucinations will grow exponentially and never reach a steady-state limit, whenever $\mu_p \geq 1/2$. Second, when $\mu_p = 1/3$, the workload factor is a very reasonable $w = 2$. This means that the receiver will have to process an equal number of message nodes and hallucination nodes, and so the computational requirements are merely double the case of no hallucinations at all.

In the case of a random codebook, the optimal number of final hallucinations was reached at exactly $\mu_p = 1/2$. In a BBC codebook, "optimal" performance is achieved at a μ_p slightly below 1/2. The difference is because the computational cost is taken into account (as well as the algorithm being inherently less efficient). As μ_p grows from 1/3 to 1/2, the workload factor grows from a small $w = 2$ to $w = \infty$. There is therefore a tradeoff: the sender can manage to send more bits in exchange for more computation on the receiver's side. But increasing μ_p from 1/3 to 1/2 gives only a 50% increase in bitrate, with an enormous increase in computational cost that is exponential in the size of the message. Similarly, a 25% bitrate increase from $\mu_p = 1/3$ to $\mu_p = 1.25/3$ increases the workload to $w = 3.5$, and a 40% bitrate increase from $\mu_p = 1/3$ to $\mu_p = 1.4/3$ increases the workload to $w = 8$. The user will have to determine how important bitrate is compared to the cost of computation. But typically, BBC will only be used to send one or two hundred bytes (e.g. a signed, random, session key), before communication switches to traditional, shared-secret spread spectrum. Therefore, we recommend using a target of $\mu_p = 1/3$ for BBC codes, which comes close to maximizing bitrate, while keeping computational costs at a very reasonable $w = 2$. All further analysis here will make that assumption. Therefore, the performance of BBC can summarized by the following theorem.

Theorem IV.1: For a BBC codebook, choosing $w = 2$, and for any given choice of m, e, and h, where m and e are large (e.g. in the hundreds or larger), the average performance of the system for randomly-chosen messages sent simultaneously will be described by M_S, R, and k, which will be:

$$M_S = \frac{\ln(2/3)\ln(3)}{\ln\left(1-\frac{1}{em}\right)W\left(\frac{3^m \ln(2/3)\ln(3)}{h\ln(1-\frac{1}{em})}\right)} \quad (4)$$

$$R = \frac{M_S}{e} \quad (5)$$

$$k = \log_3\left(\frac{\ln(2/3)\ln(3)}{h\ln\left(1-\frac{1}{em}\right)W\left(\frac{3^m \ln(2/3)\ln(3)}{h\ln(1-\frac{1}{em})}\right)}\right) \quad (6)$$

where the function W is the product-log function. `ProductLog[y]`.

Proof:

The final hallucination rate, h, is now easy to calculate. At the dotted line, the column where decoding finishes with the message and starts processing the appended zeros, there will be an expected $(w-1)M_S$ hallucinations. After that point, no new hallucinations can be generated, and each of the existing ones will survive each additional time step with probability μ_p. Therefore the expected number of hallucinations remaining after all k zero bits have been processed is:

$$h = (w-1)M_S(\mu_p)^k \quad (7)$$

Solving that for k and substituting our choice of $\mu_p = 1/3$ and $w = 2$ gives:

$$k = \log_3\left(\frac{M_S}{h}\right) \quad (8)$$

Given this value for k, it is now possible to calculate the number of messages M_S that can be sent simultaneously to obtain the chosen $\mu_p = 1/3$. Each codeword is generated by starting with an em-bit codeword of all zeros, and then $m+k$ times setting a randomly-chosen bit to 1. So if M_S messages are sent simultaneously, the packet can be thought of as being generated by starting with an em-bit packet of all zeros, and then $(m+k)M_S$ times setting a randomly-chosen bit to 1. During a single one of those rounds, a given bit will be chosen to become 1 with probability $1/(em)$, and so will remain unchanged with probability $1-1/(em)$. So after all $(m+k)M_S$ rounds, a given bit will still be zero at the end with probability $(1-1/(em))^{(m+k)M_S}$. Therefore the final density of 1 bits will be 1 minus that value:

$$\mu_p = 1 - \left(1-\frac{1}{em}\right)^{(m+k)M_S} \quad (9)$$

Substituting in the expression for k in equation 8 and the choice of $\mu_p = 1/3$ and $w = 2$ and solving the resulting equation for M_S yields:

$$M_S = \frac{\ln(2/3)\ln(3)}{\ln\left(1-\frac{1}{em}\right)W\left(\frac{3^m \ln(2/3)\ln(3)}{h\ln(1-\frac{1}{em})}\right)} \quad (10)$$

where the function $W(y)$ is the product-log function, which is defined as the result of solving the equation $y = xe^x$ for x in terms of y, and is implemented in Mathematica by the function `ProductLog[x]`. Note that the logs are now natural logarithms because of the definition of W. Then, as before, $R = M_S/e$:

$$R = \frac{\ln(2/3)\ln(3)}{e\ln\left(1-\frac{1}{em}\right)W\left(\frac{3^m \ln(2/3)\ln(3)}{h\ln(1-\frac{1}{em})}\right)} \quad (11)$$

Finally, substituting equation 10 into equation 8 gives the final result for k:

$$k = \log_3\left(\frac{\ln(2/3)\ln(3)}{h\ln\left(1-\frac{1}{em}\right)W\left(\frac{3^m \ln(2/3)\ln(3)}{h\ln(1-\frac{1}{em})}\right)}\right) \quad (12)$$

This is the final equation that was to be proved. ■

Equations 10, 11, and 12 give the expected performance for BBC, and were used to generate the table in figure 5. Note that the performance of BBC is comparable to that of an ideal, random codebook in most cases. Whereas a random codebook achieves a bitrate efficiency of about 65% in most cases, a BBC codebook achieves about 40%. It is acceptable to send data at half the rate of non-jam-resistance communication, especially if this halving of the bit rate only occurs while sending a session key, and for the rest of the session all communication will be at the full rate.

V. Conclusions

Although the problem of keyless jam resistance has not been addressed to date, it is of vital importance. We propose the BBC algorithm to achieve this, analyze its resistance to noise, and show that it can achieve the same type of jam resistance as traditional spread spectrum (which uses secret keys), with almost half the bit rate. A hybrid system allows the average bit rate to almost equal that of traditional spread spectrum. As the importance of wireless networks continues to grow, the importance of this problem will grow apace.

References

[1] A. Belouchrani and M. Amin, "Jammer mitigation in spread spectrum communications using blind source separation," *Signal Processing*, vol. 80, pp. 723–729, Apr. 2000.

[2] L. B. Milstein, "Interference rejection techniques in spread spectrum communications," *Proc. IEEE*, vol. 76, pp. 657–671, June 1998.

[3] G. J. Saulnier, Z. Ye, and M. J. Medley, "Performance of a spread-spectrum ofdm system in a dispersive fading channel with interference," in *Proc. MILCOM Conf.*, pp. 679–683, 1998.

[4] J. P. F. Glas, "On multiple access interference in a ds/ffh spread spectrum communication system," in *Proc. of the Third IEEE International Symposiumon Spread Spectrum Techniques and Applications*, (Oulu, Finland), July 1994.

[5] E. G. Kanterakis, "A novel technique for narrowband/broadband interference excision in ds-ss communications," in *MILCOM '94*, vol. 2, pp. 628–632, 1994.

[6] L. Li and L. Milstein, "Rejection of pulsed cw interference in pn spread-spectrum systems using complex adaptive filters," *IEEE Trans. Commun.*, vol. COM-31, pp. 10–20, Jan. 1983.

[7] L. B. Milstein, "Interference suppression to aid acquisition in direct-sequence spread-spectrum communications," *IEEE Transactions on Communications*, vol. 36, pp. 1200–1207, Nov. 1988.

[8] H. V. Poor and L. A. Rusch, "Narrowband interference suppression in spread spectrum cdma," *IEEE Personal Communication Magazine*, vol. 1, pp. 14–27, Aug. 1994.

[9] T. Ristaniemi, K. Raju, and J. Karhunen, "Jammer mitigation in ds-cdma array system using independent component analysis," in *Proc. IEEE Int. Conf. on Communications*, (New York, USA), Apr. 2002. To appear.

[10] M. Davis and L. Milstein, "Implementation of a cdma receiver with multiple-access noise rejection," in *Proceedings of the Third IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC '92)*, pp. 103–107, oct 1992.

[11] 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*, vol. 6.

[12] I. Bergel, E. Fishler, and H. Messer, "Low complexity narrowband interference suppression in impulse radio," in *Proceedings of the 2003 International Workshop on Ultra Wideband Systems (IWUWBS), Oulu, Finland*, June 2003.

[13] I. Bergel, E. Fishler, and H. Messer, "Narrow-band interference suppression in time-hopping impulse-radio systems," in *Proceedings of the Conference on Ultra Wideband Systems and Technologies, Baltimore, MD, May 20-23*, pp. 303–307, May 2002.

[14] R. Blazquez and A. P. Chandrakasan, "Architectures for energy-aware impulse uwb communications," *ICASSP*, Mar. 2005.

[15] G. Ahlstrm and D. Danev, "A class of superimposed codes for cdma over fiber optic channels," Tech. Rep. LiTH-ISY-R-2543.

[16] S. V. Mariac and V. K. Lau, "Multirate fiber-optic cdma: System design and performance analysis," *Journal of Lightwave Technology*, vol. 16, Jan 1998.

[17] L. Rusch and H. V. Poor, "Narrowband interference suppression in cdma spread spectrum communications," *IEEE Transactions on Communications*, vol. 42, pp. 1969–1979, Apr 1994.

[18] K. Raju, T. Ristaniemi, J. Karhunen, and E. Oja, "Jammer suppression in ds-cdms arrays using independent component analysis," *IEEE Transactions on Wireless Communications*, vol. 5, Jan 2006.

[19] Z. Ye, D. Lee, G. Saulnier, and M. Medley, "Anti-jam, anti-multipath spread spectrum ofdm system," in *Conference Record of the Thirty-Second Asilomar Conference on Signals, Systems & Computers, 1998*, vol. 2, pp. 1793–1797, nov 1998.

[20] G. Saulnier, M. Mettke, and M. Medley, "Performance of an ofdm spread spectrum communications system using lapped transforms," in *Proceedings of the IEEE Military Communications Conference, 1997 (MILCOM 97)*, vol. 2, pp. 608–612, nov 1997.

[21] R. Blazquez, P. Newaskar, and A. Chandrakasan, "Coarse acquisition for ultra wideband digital receivers," in *Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, vol. IV, (Hong Kong, China), pp. 137–140, Apr. 2003.

[22] K. J. Negus, J. Waters, J. Tourrilhes, C. Romans, J. Lansford, and S. Hui, "Homerf and swap: Wireless networking for the connected home," *ACM Mobile Computing and Communications Review*, vol. 2, pp. 28–37, Oct. 1998.

[23] P. P. Newaskar, R. Blazquez, and A. P. Chandrakasan, "A/d precision requirements for an ultra-wideband radio receiver," in *SIPS 2002*, (San Diego, CA), pp. 270–275, oct 2002.

[24] M. Z. Win and R. A. Scholtz, "Impulse radio: how it works," *IEEE Communications Letters*, vol. 2, pp. 36–38, Feb. 1998.

[25] R. J.-M. Cramer, R. A. Scholtz, and M. Z. Win, "Evaluation of an ultra-wide-band propagation channel," *IEEE Transactions on Antennas and Propagation*, vol. 50, pp. 561–570, May 2002.

[26] M. Z. Win and R. A. Scholtz, "On the robustness of ultra-wide bandwidth signals in dense multipath environments," *IEEE Communications Letters*, vol. 2, pp. 51–53, Feb. 1998.

[27] M. Z. Win and R. A. Scholtz, "On the energy capture of ultra-wide bandwidth signals in dense multipath environments," *IEEE Communications Letters*, vol. 2, pp. 245–247, Sep 1998.

[28] M. Z. Win and R. A. Scholtz, "Ultra-wide bandwidth time-hopping spread-spectrum impulse radio for wireless multiple-access communications," *IEEE Transactions on Communications*, vol. 48, pp. 679–691, Apr 2000.

[29] "http://www.globalgadgetuk.com/personal.htm."

[30] F. Stajano and R. Anderson, "The resurrecting duckling: Security issues for ad-hoc wireless networks," in *Security Protocols, 7th International Workshop Proc., Lecture Notes in Computer Science*, 1999.

[31] A. D. Wood and J. A. Stankovic, "Denial of service in sensor networks," *Computer*, vol. 35, no. 10, pp. 54–62, 2002.

[32] L. Zhou and Z. J. Haas, "Securing ad hoc networks," *IEEE Network Magazine*, vol. 13, Nov. 1999.

[33] M. Burmester and T. V. Le, "Secure communications in ad hoc networks," in *Proceedings of the 2004 IEEE Workshop on Infromation Assurance and Security*, pp. 234–241, May 2004.

[34] M. Burmester and T. V. Le, "Secure multipath communications in mobile ad hoc networks," in *International Conference on Information Technology: Coding and Computing (ITCC 2004)*, Apr 2004.

[35] Y. Desmedt, R. Safavi-Naini, H. Wang, C. Charnes, and J. Pieprzyk, "Broadcast anti-jamming systems," in *ICON '99: Proceedings of the 7th IEEE International Conference on Networks*, (Washington, DC, USA), pp. 349–355, IEEE Computer Society, 1999.

[36] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, vol. IT-22, no. 6, pp. 644–654, 1976.

[37] M. E. Hellman, "An overview of public key cryptography," *IEEE Communications Magazine, 50th Anniversary Commemorative Issue*, pp. 42–49, May 2002.

[38] R. Merkle, *Secrecy, Authentication, and Public Key Systems*. PhD thesis, Stanford University, 1979.

[39] W. Diffie, "The first ten years of public-key cryptography," *Proceedings of the IEEE*, vol. 76, pp. 560–577, May 1988.

[40] M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution system," in *Advances in Cryptology - Pre-Proceedings of Eurocrypt '94*, 1995.

[41] M. Burmester and Y. Desmedt, "A secure and scalable group key exchange system," *Information Processing Letters,*, vol. 94, no. 3, pp. 137–143, 2005.

[42] Y. Desmedt and M. Burmester, "Towards practical proven secure authenticated key distribution," in *Proceedings 1st ACM Conference on Computer and Communication Security*, pp. 228–231, 1993.

[43] D. R. Stinson, T. van Trung, and R. Wei, "Secure frameproof codes, key distribution patterns, group testing algorithms and related structures," *Journal of Statistical Planning and Inference*, vol. 86, pp. 595–617, 2000.

[44] S. Yekhanin, "New locally decodable codes and private information retrieval schemes," *Electronic Colloquium on Computational Complexity*, vol. TR06, p. 127, 2006.

[45] S. Lumetta and S. Mitra, "X-codes: Theory and applications of unknowable inputs," Tech. Rep. CRHC-03-08 (also UILU-ENG-03-2217), UIUC Center for Reliable and High-Performance Computing.

[46] H. G. S. Grard D. Cohen, "Asymptotic overview on separating codes," Tech. Rep. 2003-248.

[47] S. Chuprun, C. Bergstrom, and B. Fette", ""sdr strategies for information warfare and assurance"," in *"MILCOM 2000. 21st Century Military Communications Conference Proceedings"*, vol. "2".