EXTENDING CRITICAL MARK DENSITIES IN CONCURRENT CODECS THROUGH THE USE OF INTERSTITIAL CHECKSUM BITS

William L. Bahn
Leemon C. Baird III
United States Air Force Academy

Abstract— *The advent of concurrent coding theory allows omnidirectional communication systems to enjoy a level of keyless jam-resistance comparable to the keyed jam-resistance of traditional spread spectrum systems, all of which rely on shared secrets. To achieve this, concurrent codecs possess the ability to efficiently separate multiple legitimate codewords that have been superimposed. Doing so requires the ability to efficiently discriminate between true codewords and hallucinations (which are codewords or partial codewords that materialize due to the interaction of transmitted codewords, both with each other and with channel noise). Hallucinations that survive the decoding process through the end of the encoded message can be exponentially exterminated by appending checksum bits to the end of the message prior to encoding. While these terminal checksum bits can reduce the final hallucination rate to an arbitrarily low level using a very reasonable number of checksum bits, they do not prevent the computational effort of the receiver from increasing exponentially as the packet mark density rises above fifty percent. However, by embedding checksum bits within the interior of the message body, this exponential blow-up can be postponed to arbitrarily high packet mark densities. This paper presents the theory behind the use of these interstitial checksum bits and analyzes their impact on receiver performance and jam-resistance.*

I. The Need for Keyless Jam-Resistance

Wireless networks operating in hostile environments require high degrees of jam-resistance to ensure the availability of network resources. The two traditional means of providing this are highly-directional links or spread spectrum. The exclusive use of directional links in highly dynamic mobile ad hoc networks poses practical challenges that virtually guarantee that omnidirectional links will continue to play a significant role in such networks. However, spread spectrum techniques are only as secure as the shared-secrets (i.e., symmetric keys) upon which their jam-resistance is based; this, in turn, is limited by the classic key distribution problem associated with such keys. Therefore a means is needed of ensuring the availability of the channel in large omnidirectional wireless networks that does not rely on symmetric keys.

A seemingly obvious alternative are asymmetric keys since ways of using them to ensure the other classic security goals (i.e., confidentiality, integrity, and authenticity) are well understood. Unfortunately, these all assume and require that data can be successfully exchanged between parties, implying that the communications channel is already available. Consequently, the conclusion is that a means is needed of ensuring the availability of the channel in large omnidirectional wireless networks that does not rely on secret keys at all.

This conclusion can be reached independently by considering a different category of application altogether – namely public-access systems. One example is civilian GPS where communications are one-way and where the pool of authorized users is literally every person on the planet. In this case, secret keys of any kind are precluded since, by definition, hostile parties are authorized users with access to the same keys as everyone else.

For an unkeyed system – a system with no secret keys – to be jam-resistant, it must be capable of dealing with multiple overlapping transmissions of legitimate waveforms: some from friendly sources and some from hostile sources. This problem is closely related to the field of superimposed codes; historically such codes have been limited to applications in which *membership testing* is sufficient since, until recently, no efficient means of fully decoding a superimposed transmission has existed. This situation changed fundamentally with the advent of concurrent codes; by definition, these are superimposed codes that can be efficiently decoded. In fact, although only one realized construction of a practical concurrent code presently exists, its decoding complexity is linear-time with respect to the product of the length and number of transmitted messages.

II. A Brief Review of Concurrent Coding Theory

A detailed explanation of concurrent coding theory is available in [1] and [2], but is not yet widely disseminated, thus a brief summary is provided here. The foundation upon which concurrent codes rest is the decades-old field of superimposed codes[3], of which Bloom filters[4] are an example. The heart of a superimposed code is an encoding algorithm whereby a packet containing several different messages can be constructed by performing a bitwise-OR of the codewords corresponding to the chosen messages. Going the other way, a packet is considered to contain a particular message if it "covers" that message, meaning that all of the marks (i.e., bits that are HI) in the codeword for that message are contained in the packet. Ideally, a packet should only cover those messages used to construct it. In practice, this will only be true up to a point; if too many messages are placed into a packet then it will begin covering additional messages. These additional messages go by a variety of different names; here they are referred to as *hallucinations*. In general, parameters in the coding and decoding (i.e., *codec*) algorithms permit the user to control, at least statistically, how many messages can be contained in a packet before hallucinations start to appear. The extreme case of this occurs when the packet consists solely of marks, in which case it covers every possible message and, consequently, contains absolutely no information.

Historically, superimposed codes have been used where it is sufficient to ask if a particular message, or small set of messages, is covered by the given packet. This is done by performing a "membership test" which involves verifying that all of the marks in the message being tested are covered by the packet. Membership tests can be performed very efficiently; conceptually they involve nothing more than a bitwise-AND between the codeword and the bitwise complement of the packet with any non-zero result declared a failure.

However, if it is necessary to generate a list of all messages covered by the packet - referred to here as *decoding* the packet - then an exhaustive search spanning the entire message space is generally needed. As recently as 2003, it has been claimed by at least one researcher [5] that no efficient means of decoding arbitrary codewords from very large code books yet exists. Decoding therefore requires an exponential amount of time as a function of the message length. While acceptable for applications having a sufficiently small message space as well as sufficient time and processing power to perform the work, it is totally infeasible given the message spaces, processing capabilities, and time constraints found in typical communication systems.

Thus, for communication systems, a subset of superimposed codes is needed in which the entire list of messages can be extracted from a packet in an "efficient" manner. This requirement is the defining characteristic of a concurrent code compared to the broader set of superimposed codes.

III. A Review of the BBC Encoding and Decoding Algorithms

The BBC algorithms are a pair of algorithms that encode and decode data into concurrent message packets. Like concurrent coding theory itself, extensive details about them and their behavior are now in the open literature[1], [2] and only a brief description is provided here. The underlying basis for the BBC algorithms is that m-bits of data are first transformed into a message and this message is then encoded one bit at a time, using progressively longer prefixes, to construct a c-bit codeword. This allows for the efficient decoding of the packet since the messages can be extracted one bit at a time by looking for progressively longer message prefixes and data extracted from the recovered message.

The BBC algorithms are a pair of algorithms that encode and decode data to and from concurrent message packets. Like concurrent coding theory itself, extensive details about them and their behavior are now in the open literature[1], [2] and only a brief description is provided here. The underlying basis for the BBC algorithms is that m-bits of data are first transformed into a message which is then encoded, one bit at a time, using progressively longer prefixes to construct a c-bit codeword. This allows for the efficient decoding of the packet since messages can be recovered one bit at a time by looking for progressively longer message prefixes. Finally, the data can be extracted from the recovered message.

In practice decoding a packet is most efficiently done using a depth-first search of the message space; however, it is easiest to describe in terms of a breadth first search, as depicted in Algorithm 2.

The dominant parameter in the algorithm is the expansion factor, e, which is the ratio of the length of the codeword, c, to the length of the original data, m (i.e., $c = me$). The expansion factor determines how the output of the hash function is interpreted; specifically, it is interpreted as being one of the c possible locations in the codeword. The expansion factor is roughly equivalent to the processing gain of a traditional spread spectrum system, both in terms of bandwidth spreading and in terms of the degree of jam-resistance.

One important point to note about concurrent codes – and true of all superimposed codes – is that any one codeword is relatively sparse, consisting almost entirely of spaces with a few pseudorandomly scattered marks. The fraction of a

Algorithm 1 BBC Encode(D)

```
*This function takes $m$-bits of data, pads it with $k$ checksum bits at locations specified by the protocol in use to produce the message $M$, and then places marks in the codeword $P$ at locations determined by hashes of all possible prefixes of the padded message.*

$M$ = $D$ with $k$ spaces added as specified by the protocol.
$P \leftarrow 0$ (i.e., $P$ is a $c$-bit vector initialized to all spaces)
for ($n \leftarrow 1 \ldots m+k$ ($m+k$ = length($M$))) do
    Compute $L = Hash(M[1 \ldots n])$ such that $1 \leq L \leq c$
    Place a mark in $P$ at location $L$
end for
return $P$

NOTES:
1) $M[1 \ldots n]$ is the $n$-bit prefix of $M$.
2) $1 \leq Hash(X) \leq c$.
```

codeword (or packet), that consists of marks is referred to as the *mark density*; this plays a key role in determining decoder performance.

IV. The Role of Checksum Bits

Checksum bits kill hallucinations. They do this by inserting k marks into the codeword at locations that are a function of the entire message or, in the case of interstitial checksum bits, the message up to the point at which they were inserted. Since the decoder knows a priori the value of all checksum bits, the list of covered prefixes cannot grow when decoding a checksum bit. Furthermore, any hallucinations that exist in the list will survive the decoding of a checksum bit purely by chance. In general, the fraction of hallucinations that survive the decoding of a checksum bit is equal to the packet's mark density (since that is the probability that the mark corresponding to the checksum bit will coincide with an unrelated mark in the packet). A detailed analysis of the impact of both terminal and interstitial checksum bits appears later in this paper.

V. The Physical OR-channel

Concurrent codes, like their superimposed brethren, require an OR-channel if they are to retain necessary properties. An OR-channel is one in which all of the codewords present in a channel are combined via a bitwise-OR operation. From a practical standpoint, this means that the receiver should produce a mark whenever any of the transmitters are broadcasting a mark and should produce a space only when all of the transmitters are broadcasting a space. This is difficult to achieve with most modern binary modulation schemes because they are designed to produce symmetric channels having nearly identical bit error probabilities, regardless of whether a mark or space is transmitted, as this minimizes the overall bit error rate (BER). However, by using a highly asymmetric channel, such as old-fashioned On-Off Keying (OOK), an OR-channel can be implemented with a very high degree of fidelity by setting the detection threshold sufficiently

Algorithm 2 BBC Decode(P)

```
*This function decodes all of the data in a given packet by first identifying all messages covered by the packet and then extracting the data from each covered message. At any given stage of the process, a list of message prefixes that are covered by the packet is maintained. The prefixes are lengthened one bit at a time and only those still covered are retained. If the next message bit is a checksum bit, then the prefix extended with a '0' bit is considered. However, if the next message is a data bit then the prefix extended with a '1' bit is also considered.*

$M \leftarrow \{\}$ (i.e., $M$ is an empty message list)
for ($n \leftarrow 1 \ldots (m+k)$) do
    Expand partial messages in list to $n$-bits
    for (Each partial message, $M_j$, in $M$) do
        Remove $M_j$ from list $M$
        Add $M_j : 0$ to list $M$ (i.e., append a '0')
        if $n$ corresponds to a data bit then
            Add $M_j : 1$ to list $M$ (i.e., append a '1')
        end if
    end for
    Prune messages from list
    for (Each partial message, $M_j$, in $M$) do
        Compute $L = Hash(M_j)$
        if $P$ does not contain a mark at location $L$ then
            Remove $M_j$ from list $M$
        end if
    end for
end for
Extract the data from the messages
for (Each message, $M_j$, in $M$) do
    Strip all checksum bits from $M_j$
end for
```

low. Since such a channel is not symmetric, it cannot be characterized by a single error rate. Instead, it has a mark error rate (MER) and a space error rate (SER). The MER is the probability that a space will be received even though one or more marks were transmitted and the SER is the probability that a mark will be received even though no marks were transmitted. In general, lowering the detection threshold will lower the MER at the expense of raising the SER.

VI. Jam-resistant Nature of Concurrent Codes in and OR-channel

At this point it is possible to understand how concurrent codecs provide high degrees of assurance that the channel will remain available even in the presence of considerable hostile jamming. If a sender transmits a message that is encoded with a concurrent codec, then as long as all transmitted narks are received, the message will be contained in the list of messages produced by the receiving codec. By contrast, successful reception of all spaces is not necessary; in fact, a significant fraction can be received erroneously as marks without significant impact on decoder performance. The attacker thus has two options available to them: (1) they can try to force the receiver to make a mark error, or (2) they can try to flood the receiver with space errors (false marks).

The first option, while highly attractive since the decoder is extremely sensitive to mark errors, is not very practical because once the sender transmits energy into the spectrum, the attacker cannot easily remove that energy; they can add to it or corrupt it, but they cannot easily diminish it in any practically meaningful sense. The receiver, on the other hand, is not relying on being able to extract any information from the energy that was transmitted beyond the mere detection of its existence. If the receiver's detection threshold is sufficiently low, detection is virtually guaranteed. Furthermore, promising methods of tolerating modest mark error rates are presently being explored.

This leaves the second option which, while not particularly attractive since the decoder is highly insensitive to space errors, is never-the-less always possible. The hostile party can almost certainly force a sufficiently high space error rate so that packet is beyond the codec's ability to decode it with the available resources. However, doing so requires the attacker to expend considerably more energy in the attack than the genuine sender did thus exposing them to the network defenders. In order to successfully jam the channel the attacker must accept a commensurate level of risk; if that risk can be made unacceptable at the level of effort required to jam the channel, then the channel will most likely remain available.

It should be emphasized that concurrent codecs promise neither jam-proof communications nor a level of jam-resistance greater than that of uncompromised spread spectrum. What concurrent codecs provide is a comparable level of jam-resistance without the need for symmetric keys and the corresponding key management nightmare. Thus, from a practical perspective, concurrent codes can be viewed not as a means of improving jam-resistance, but as a tool for improving key management.

VII. Decoder Performance Metrics

Since the primary attack against a concurrent codec is to overload the processing capacity of the receiver's decoder, the primary measure of receiver effort is the computation workload per genuine message recovered. A very good proxy for this workload is the number of calls to the hash function. Since a fixed number of calls to the hash function are made for each entry in the message list at each stage of decoding, a near-direct measure of the workload is the average number of messages in the list over the entire decoding process divided by the number of genuine messages recovered. Thus to determine the workload we must examine the size of the message list throughout the decoding process.

After the n^{th} decoding stage, the total number of messages in the message list is

$$M_T(n) = M + H(n) = (M_S + M_A) + H(n) \quad (1)$$

where $M_T(n)$ consists of two parts: the number of intentional (i.e., properly encoded) messages, M, which is constant, and the number of hallucinations, $H(n)$, which is variable. The intentional messages, M, can be further broken into two subgroups, those transmitted by the genuine sender, M_S, and those transmitted by the attacker, M_A.

The attacker can always force the receiver to do an amount of work at least roughly proportional to the number of properly encoded messages, M_A, that the attacker transmits. This is inherent in the fact that the system has no secret keys meaning the attacker always has the ability to properly encode messages. Fortunately, itself this is not a serious threat since to force twice the workload on the receiver the attacker is forced to transmit roughly twice the energy and, in doing so, accepts a proportionately greater risk of detection and localization. Furthermore, although the attack messages will survive decoding and be included in the final message list, higher levels in the communications stack have the ability to discriminate against by authenticating digital signatures contained in the legitimate messages.

The attacker's goal is to force the production of large (preferably exponential) numbers of hallucinations per attack message because then they have the potential to overwhelm the decoder while remaining acceptably covert. In essence, the attacker must pay for attack messages but hallucinations are free. Thus the key to quantifying decoder workload is to understand the expected behavior regarding the production and extermination of hallucinations per properly formatted message.

VIII. General Hallucination Behavior

Hallucinations can be broken into three categories: working, terminal, and realized. Working hallucinations are those that exist during the decoding process and manifest themselves as false partial messages that are indistinguishable from (and must therefore be processed identically to) true messages. Working hallucinations are continually generated and extinguished as part of the normal decoding process. Those that exist once the last data bit has been decoded are the terminal hallucinations; these are exponentially exterminated by any terminal checksum bits. Any hallucinations that survive the final checksum bit are realized as false messages (hence the term "realized hallucinations") and passed to higher levels of the network stack for disposition; this is straightforward if legitimate messages contain authentication information.

As will be shown, terminal and realized hallucinations pose little threat because, as will be shown, even modest numbers of terminal checksum bits provide extremely high levels of discrimination against terminal hallucinations. Similarly, any realized hallucinations that seep through are effectively, additional attack messages that can be discriminated against using the same authentication process.

On the other hand, working hallucinations are the dominant factor in determining the decoder workload. If the decoder can't keep up with the processing requirements, the channel becomes jammed or, at the very least, packets start being dropped as load-shedding begins. At any particular decoding stage the rate at which hallucinations are formed and extinguished depends directly on two things: the packet mark density, μ, and whether the bit being decoded is a data bit or a checksum bit. These relationships are shown in Figure 1.

If bit n is a data bit, then as each actual message is extended using the next bit in the actual data string there is an assumed 100% chance of finding the associated

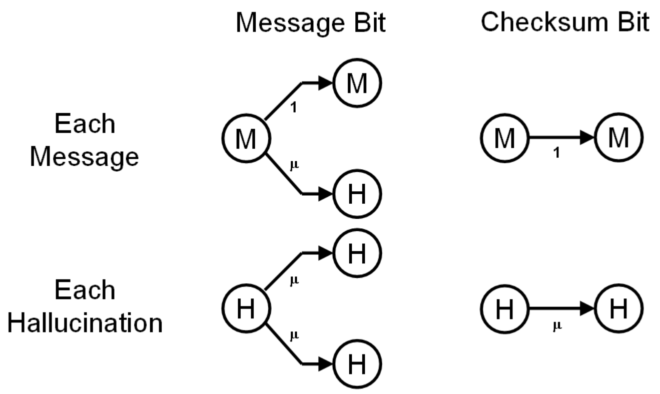


Fig. 1. Hallucination generation and propagation probabilities.

mark. However, making the worst-case assumption that all intentional messages have distinct prefixes by this stage, then when using the other choice for the next data bit there is still a chance, equal to μ, that the mark will be found thus producing a hallucination. Likewise, every hallucination that exists at this stage has the potential to producing two hallucinations for the next stage, each with probability μ. Thus the number of expected hallucinations after this stage of decoding is

$$H(n) = \mu M + 2\mu H(n-1) \text{ (n is a message bit)} \tag{2}$$

On the other hand, if bit n is a checksum bit then only one possible value for the next bit, namely 0, will be examined. In the case of actual messages, it is assumed that the corresponding mark will be found, but hallucinations only have a probability μ of surviving. Thus the number of expected hallucinations after this stage of decoding

$$H(n) = \mu H(n-1) \text{ (n is a checksum bit)} \tag{3}$$

IX. Hallucination Performance of an Unenhanced BBC-based Codec

For our purposes, an unenhanced BBC-codec forms messages by appending k terminal checksum bits to m-bits of data as the first step in Algorithm 1. Statistically speaking, as long as the packet density remains below some critical density, μ_c, the hallucination density – number of hallucinations per intentional message – quickly settles to a steady state value. The hallucination density remains near the steady state level until the terminal checksum bits are encountered at which time the remaining terminal hallucinations are exterminated exponentially. The steady state hallucination density, H_{SSD}, can be found by equating the hallucinations at two adjacent stages of the decoding process and, since it is a density, dividing by the total number of intentional messages. This yields a steady state hallucination density of

$$H_{SSD} = \frac{H(n)}{M} = \frac{\mu}{(1-2\mu)} \tag{4}$$

This result also allows us to determine the critical packet density since H_{SSD} becomes singular as the denominator approaches zero. Thus

$$\mu_c = 0.5 \tag{5}$$

For an $(m+k)$-bit message, the terminal checksum bits are reached at stage m, thus $H(m)$ represents the number of terminal hallucinations that must be exterminated by the terminal checksum bits; these are exterminated exponentially according to the relation

$$H(m+i) = H(m)\mu^i \text{ for } 0 \leq i \leq k \tag{6}$$

In particular, if there are k terminal checksum bits, then the expected density of realized hallucinations is

$$H_R D = \mu^k H_{SS} = \frac{\mu^{(k+1)}}{(1-2\mu)} \tag{7}$$

The results shown in Figure 2 validate the above predictions. In this test run, one thousand 200-bit messages were superimposed in the same packet bringing the packet density to 33.8%. The expected steady-state hallucination density at this packet density, per Equation 4, is 1.04 hallucinations per message. The predicted performance is shown by the thin line while the actual performance is represented by the bold line. As is evident, the actual performance tracks the predicted performance very closely throughout the decoding process, including the initial build up, in steady state, and during the terminal checksum decoding. With eight terminal checksum bits and a packet density of 33.8%, the predicted density of realized hallucinations is only 170 ppm; hence, even with 1000 messages spawning hallucinations and with only eight terminal checksum bits there is less than a one-in-five chance that a single hallucination will survive the decoding process.

The apparent negative hallucination density in Figure 2 during the initial decoding stages is an artifact of the math since, early on, there are insufficient bits in the partial messages to support as many messages as are known to exist in the packet, thus the computed number of hallucinations is negative.

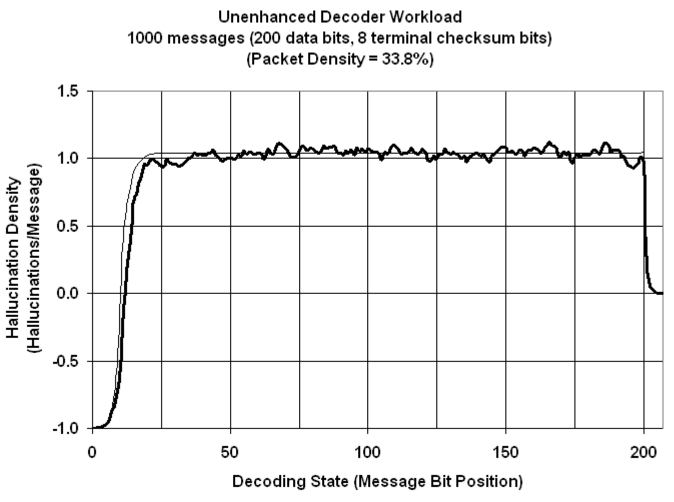


Fig. 2. Hallucination workload of an unenhanced BBC-codec.

X. The Use of Interstitial Checksum Bits

Early in the development of the BBC algorithms, the notion of distributing the checksum bits amongst the data bits was considered. However, once it was realized that the decoder quickly reached a steady-state hallucination level, it was apparent that the use of checksum bits within the message interior would provide only a transient reduction in the hallucination level and have no impact on the number of terminal hallucinations (unless, of course, they appeared very close to the end of the message itself). As a result, the number of terminal checksum bits needed would not be reduced and the use interstitial checksum bits did not appear justifiable.

However, owing purely to academic curiosity, a codec was written that permitted checksum bits to be inserted within the message body by breaking the data into fragments and prepending each fragment with a string of checksum bits. For simplicity the first test involved inserting one checksum bit prior to each data bit, thus effectively doubling the length of the message, the decoding chain, and the number of marks in each codeword. The simulator, already configured to produce about ten hallucinations per message (in an unenhanced codec, meaning that it was producing packets with a mark density of about 47%) yielded a very unexpected result. Even though there were twice as many decoding stages the enhanced decoder finished in well under half the time required by the unenhanced codec. This observation prompted a closer examination of the expected behavior.

XI. Predicted Hallucination Performance of an Enhanced BBC-based Codec

For our purposes, an enhanced codec is one where data is broken into fragments consisting of d data bits separated by groups of s interstitial checksum bits; for simplicity, these will be referred to as "fragment" and "clamping" bits, respectively, in this section. As with the unenhanced codec, we are primarily concerned with the steady state workload imposed on the decoder as a result of the presence of working hallucinations. However the enhanced codec oscillates between two hallucination levels; after decoding a fragment of data the working hallucinations will be at their highest level, H_H, and will then fall to their lowest level, H_L, after decoding the next group of clamping bits.

The relationships described in Figure 1 and quantified in Equations 2 and 3 still apply. Using these, the hallucination level after decoding s clamping bits is

$$H_L = \mu^s H_H \tag{8}$$

while the hallucination level after decoding d fragment bits is

$$H_H = M\left[\mu\frac{1-(2\mu)^d}{1-2\mu}\right] + (2\mu)^d H_L \tag{9}$$

Combining these two results and dividing by the number of intentional messages to get the peak hallucination density, H_PD, we have

$$H_PD = \frac{H_H}{M} = \mu\left[\frac{1-(2\mu)^d}{1-2\mu}\right]\left[\frac{1}{1-(2\mu)^d\mu^{(s+d)}}\right] \tag{10}$$

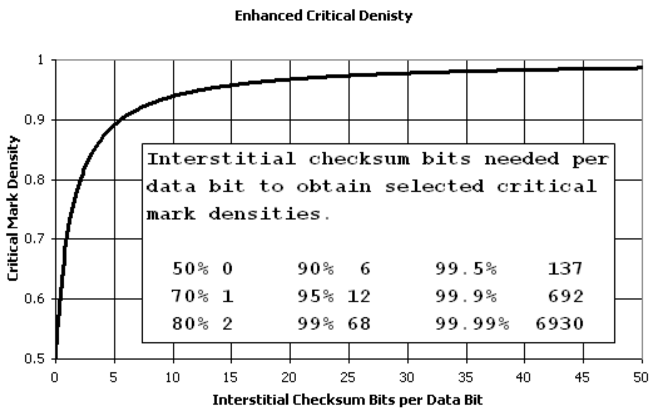
The denominator of the first fraction divides the numerator and is thus a removable singularity. Hence the critical density is governed by the denominator of the second fraction, yielding a critical density of

$$\mu_c = 0.5^{\left[\frac{d}{s+d}\right]} = 0.5^{\left[\frac{1}{1+\frac{s}{d}}\right]} \tag{11}$$

Solving for the ratio of clamping bits to fragment bits as a function of the desired critical packet density, we have

$$\frac{s}{d} = \frac{\log(0.5)}{\log(\mu_c)} - 1 \tag{12}$$

As Figure 3 shows, it is possible to drive the critical packet density to arbitrarily high values, although the number of clamping bits quickly becomes unreasonable. However, experiments using 128 clamping bits between each data bit ($\mu_c = 99.46\%$) have shown that even decoding packets with a mark density of 99% is remarkably fast and efficient.



Interstitial checksum bits needed per data bit to obtain selected critical mark densities.

50%	0	90%	6	99.5%	137
70%	1	95%	12	99.9%	692
80%	2	99%	68	99.99%	6930

Fig. 3. Critical density performance of an enhanced BBC-codec.

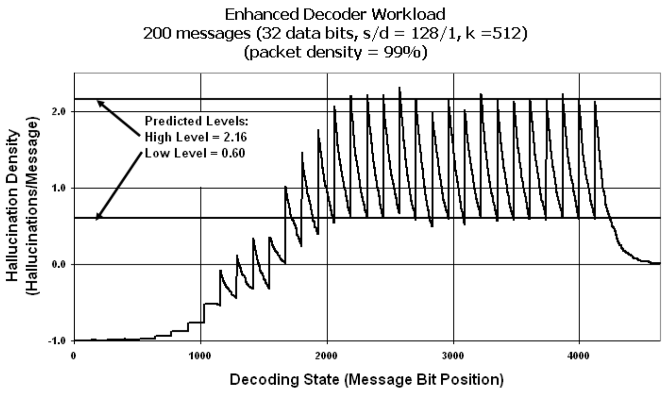


Fig. 4. Hallucination workload of an enhanced BBC-codec.

Figure 4 shows that an enhanced BBC codec performs in very close agreement with predictions. Note that unrealistically short messages (32-bits of data) were chosen only to clearly show the oscillatory behavior of the workload while displaying an entire decoding chain. Despite having over five hundred terminal checksum bits, the predicted realized hallucination rate of 1.25% is in close agreement with the zero to six such hallucinations observed over many trial runs.

XII. Practical uses for Interstitial Checksum Bits

While decoding signals from packets having nearly 100% mark densities sounds impressive, it is unlikely to prove useful in practice. Most metrics of jammer strength involve the ratio of average jamming signal power to average legitimate signal power at the receiving antenna. Where that is the relevant metric, extended critical mark densities are detrimental because of disproportionately higher legitimate

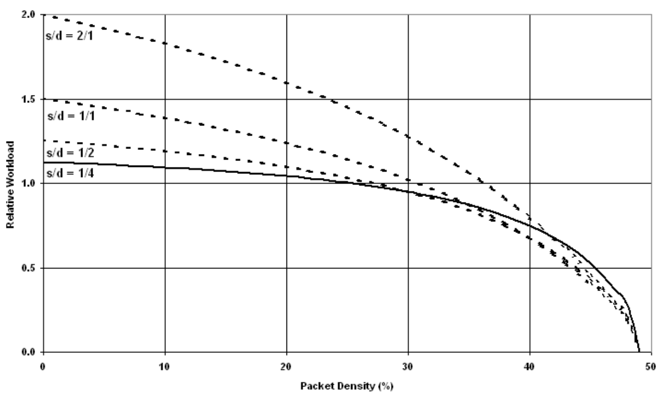


Fig. 5. Impact of Interstitial Checksum Bits on Decoder Workload.

transmitted signal energy. For instance, an unenhanced codec with an expansion factor of 1000 requires an attacker to expend approximately 500 times the energy to bring the packet to critical density. A single interstitial checksum bit per data bit may raise the critical mark density to nearly 71%, but the attacker now only has to expend approximately 350 times the sender's energy to reach that level. At a critical mark density of 99%, this is reduced to a mere factor of fourteen.

Energy-limited jammers, perhaps remotely-deployed and battery-powered, might conceivably be able to raise the packet density above 50% but lack the ability to jam continuously and drive it to 100%. The use of interstitial checksum bits to achieve high critical mark densities might bear fruit in this rather unlikely situation.

While the use of interstitial checksum bits is difficult to justify based on exploiting the increased critical mark density, the existence of a region having lower total workload per data bit provides some incentive for operating the codec with a light load of interstitial checksum bits. As Figure 5 shows, with one checksum bit before each four data bits (the solid line) the negative impact at low packet densities is limited to about 10% while workload is actually decreased for packet densities above 25% thus yielding a processing gain above this level. At the same time, only 10% of the jam-resistance is sacrificed because the critical mark density is improved to about 57% requiring the attacker to still expend about 450 times the energy to jam the channel (for the example used previously).

XIII. Results Using Software-Defined Radio

In addition to operating the codecs with computer-simulated packet data, experiments have been performed in hardware using software defined radios. While the data obtained thus far is limited, the observed impact on performance appears to match predictions quite closely.

XIV. Conclusion

While interstitial checksum bits allow the critical mark density of a concurrent codec to be pushed to arbitrarily high levels, and while this ability is quite interesting from an academic perspective, its utility in practical systems is questionable because the gain in absolute jam-resistance is more than offset by the increase in self-jamming. However, despite the increase in codeword mark density, the unexpected appearance of a region in which codec workload is actually decreased (in absolute terms) offers the ability to improve the processing gain in ways that may find practical application.

XV. Acknowledgements

This work was sponsored in part by the Air Force Information Operations Center (AFOIC), Lackland AFB, TX, and was performed principally at the Academy Center for Cyberspace Research (ACCR) at the United States Air Force Academy.

References

[1] 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.

[2] L. Baird, W. Bahn, M. Collins, M. Carlisle, and S. Butler, "Keyless jam resistance," in Proc. 8th Annual IEEE SMC Information Assurance Workshop (IAW), jun 2007, pp. 143–150.

[3] W. Kautz and R. Singleton, "Nonrandom binary superimposed codes," IEEE Transactions on Information Theory, pp. 363–377, 1964.

[4] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," Communications of the ACM, vol. 13, no. 7, pp. 422–426, Jul. 1970.

[5] G. Cormode and S. Muthukrishnan, "What's hot and what's not: Tracking most frequent items dynamically," in Proceedings of ACM Principles of Database Systems, 2003, pp. 296–306. [Online]. Available: "http://acm.org/sigmod/pods/proc03/online/210-cormode.pdf"