The Theory of Concurrent Codes with Application to Omnidirectional Jam-Resistant Communications without Shared Secrets

A Dissertation Proposal for the Ph.D. Degree in Electrical Engineering

William L. Bahn
Department of Electrical and Computer Engineering
University of Colorado at Colorado Springs
Colorado Springs, Colorado 80918
Email: wbahn@uccs.edu

Abstract

(Proposed dissertation abstract)

As commercial and military activities become increasingly dependent on wireless communications, the issue of jam-resistance likewise becomes increasingly important. To date, jam-resistance in wireless communications is accomplished either by using highly directional signals (e.g., high-gain antennas, lasers, or physical cable) or by employing modulation techniques involving symmetric keys (i.e., shared secrets). While directionality is generally the more effective of the two, it is not always practical. For instance, the United States military is becoming increasing dependent on highly mobile, battlefield ad hoc networks as part of the Global Information Grid (GIG). Such networks will almost certainly contain large numbers of omnidirectional wireless links and, at present, the jam resistance of those links is reliant on symmetric keys.

In general, cryptographic key management has been identified as a challenging problem in realizing the GIG. Most of the attention, however, has been focused on keys used to encrypt and decrypt data and much progress has been made utilizing asymmetric cryptography and a Public Key Infrastructure (PKI). Unfortunately, the same cannot be said for keys used to protect physical communication layers from hostile jamming.

There appears to be an unstated assumption that jam-resistance for such omnidirectinal links requires symmetric keys. These keys are then used to modulate the signal, generally via spread spectrum techniques. While shared-secret schemes are workable in small networks, the scale and nature of theater-wide, mobile, ad hoc wireless networks will quickly inundate any practical key management strategy.

Furthermore, public-access systems - such as the civilian side of the Global Positioning System (GPS) - preclude reliance on secret keys since, by definition, the pool of authorized users includes every person on the planet. Yet while these systems are recognized as having little to no jam resistance, the present Federal Radionavigation Plan calls for increased reliance on GPS for civil aviation operations, including en-route navigation and precision instrument landing approaches.

The key management problem can be greatly alleviated if a suitable asymmetric system for the physical layer can be developed, but little attention has been directed at this problem and no such system exists. Presented here is a new coding theory - the theory of concurrent codes - that permits the construction of such systems. One particular concurrent algorithm, the BBC algorithm, is explored in depth and serves as the foundation for a family of concurrent codecs implementable in several different suitable physical layer channels.

The jam-resistance afforded by these codecs is explored within the context of several potential attack algorithms and their performance compared to more traditional shared-secret based systems is discussed, both with and without a compromised key.

Finally, several prototype systems have been developed that demonstrate the capabilities of concurrent codecs using audio, image, and radio frequency transmission channels.

CONTENTS

I Preface 3

II Introduction 4
- II-A The Growing Need for Jam Resistance . . . 4
- II-B Defining Jamming and Jam-Resistance . . . 5
- II-C The Use of Directionality to Achieve Jam Resistance . . . 6
- II-D The Use of Shared Secrets to Achieve Jam Resistance . . . 7
- II-E The Key Management Problem . . . 8
- II-F Analogy to Public-Key Cryptography . . . 9
- II-G Avenue of Research . . . 10

III Literature Search 12
- III-A Relevant Work on Jam-Resistance . . . 12
- III-B Relevant Work Superimposed Codes . . . 12

IV Work to Date 16
- IV-A Concurrent Codecs 101 . . . 16
- IV-B The Basic Requirements of a Concurrent Codec . . . 20
- IV-C The Relevant Performance Metrics . . . 20
- IV-D Concurrent Codecs in the Context of the OSI Network Model . . . 23
- IV-E The Physical Layer Requirements . . . 23
- IV-F The Inadequacy of Conventional Coding Theory . . . 25
- IV-G The Performance of a Straight-through Codec . . . 26
- IV-H The Performance of a One-Hot Codec . . . 26
- IV-I The Performance of a Superimposed Codec . . . 27
- IV-J The BBC Algorithm . . . 29
  - IV-J.1 The Core Coding/Decoding Algorithm . . . 30
  - IV-J.2 The Use of Checksum Bits to Enhance Orthogonality . . . 34
  - IV-J.3 Suppression of Terminal Hallucinations . . . 36
  - IV-J.4 Working Hallucinations and Critical Density . . . 37
  - IV-J.5 Interior Checksum Bits and Extended Critical Density . . . 38
- IV-K The Visualization Demonstration . . . 40
- IV-L The Sonification Demonstration . . . 40
- IV-M Publications, and Conferences . . . 41
  - IV-M.1 Published or Accepted Papers . . . 42
  - IV-M.2 Publications In Progress . . . 42
  - IV-M.3 Papers in Preparation . . . 43
  - IV-M.4 Future Papers Planned . . . 43

V Research Plan 43

VI Conclusion 44

Appendix I: Glossary of Terms and Symbols 45

II Expanded Explanations of Terms 46
- II-A BBC Codec Parameters - m, c, e, s, k, α, β . . . 46
- II-B Packet Parameters - P, M, μ . . . 48
- II-C Sender, Receiver, and Attacker - S, R, A . . . 48
- II-D Attacker and Receiver Effort - E, ζ . . . 48

References 50

I. PREFACE

As a dissertation proposal, this document has the very narrow focus of establishing the author's qualifications to pass the Ph.D. Comprehensive Examination by demonstrating preparedness to continue work toward the final dissertation defense. To accomplish that, the following must be established in turn: (1) an adequate definition of the problem being explored; (2) that the problem is both real and relevant; (3) that the scope of the problem, and the work performed to address it, is of adequate depth to warrant the degree sought; (4) that the work is original and expands the body of scientific knowledge; and (5) that sufficient exploratory results have been obtained to permit the development of a reasonable plan for the remainder of the research effort.

To achieve the above goals, this proposal is partitioned as follows:

The **Introduction** will provide an overview of how wireless communication is increasingly being used and why that translates into a corresponding increase in the importance of providing jam-resistant communications. It will then provide a description of the present techniques used to achieve jam-resistance and their shortcomings, both intrinsically and in light of how these communications are increasingly being used. The emphasis will shift to omnidirectional wireless communications, as these are the types of communications that present capabilities are least able to protect and that, consequently, are the focus of the proposed research. The section will end with a discussion of how that research will permit the mitigation of some of these shortcomings.

The **Literature Search** will discuss the body of work that presently exists relating to superimposed codes, of which concurrent codes are a new subset, and make clear that concurrent codes and the algorithms proposed possess properties that represent extensions to that body of knowledge.

The **Work to Date** will provide a summary of the results obtained to this point, including theoretical work, simulation results, and preliminary physical prototypes. Furthermore, a review of the efforts to obtain peer review of the work will be provided. These results establish the basic validity of the avenue of research and support the assertion that the remaining work is both realistic and attainable.

The **Research Plan** will detail the work that remains to be done and the proposed schedule for its completion.

II. Introduction

A. The Growing Need for Jam Resistance

Reliable communication of data - be it voice, imagery, telemetry, financial, command and control, or an almost countless variety of other types - has become critical to modern economies and also to modern warfare[1], [2]. Long gone are the days when interfering with an opponent's communications seldom proved more than an annoyance or caused more than relatively minor delays in the flow of information. Today we are faced with many examples of communication systems in which even short term interruptions can prove catastrophic: an airliner low on fuel on final approach in bad weather is in a very precarious situation if the navigation signals are interfered with; similarly a battlefield engagement can go horribly awry if the communications used to distinguish friend from foe are compromised. These are just two examples where malicious jamming of a communications resource at a critical time can result in the loss of life.

To further expand on these two examples, the 1996 Federal Radionavigation Plan[3] specifically called for the transition to the Global Positioning Satellite (GPS) system as the primary means for enroute and terminal navigation for civil aircraft operations within the National Airspace System (NAS) with the phaseout of traditional VOR[1]/DME[2], NDB[3], and ILS[4] services to begin in 2005 and be complete in 2010. Further, it called for the use of GPS as a "sole-means" navigation service for all phases of flight, meaning that aircraft would be permitted to operate using GPS navigation without any requirement for a backup means. The present 2005 Federal Radionavigation Plan[4] reveals significant reflection on the wisdom of using GPS as a sole-means system in light of the recognized vulnerabilities in the system, including the lack of jam-resistance. Despite this, the plan still calls for significant reductions in the installed base of other systems as use shifts to GPS and a more gradual phase out of many of these systems is slated to begin between 2010 and 2015. The expectation is for these systems to be progressively designated as back-up facilities and taken off-line through attrition.

On the battlefield, the United States is presently pursuing the doctrine of net-centric operations (NCO) (a.k.a., net-centric warfare) with the goal of realizing the Global Information Grid (GIG)[5], [6] that will encompass users at all levels including top-level commanders, soldiers in the field, and conceivably even individual pieces of equipment such as rifles and artillery shells. Parts of the GIG have already been implemented and proven to be of significant utility. As the GIG is further realized, significant gains in efficiencies can potentially be achieved. For instance, if (among many other pieces of data) every rifle is reporting its ammunition usage in realtime via the GIG, then the flow of battle can more effectively

[1]VHF Omnidirectional Range - Allows aircraft to determine their absolute direction from the station.
[2]Distance Measuring Equipment - Allows aircraft to determine their distance from a VOR station.
[3]Non-directional Beacon - Allows aircraft to determine their relative direction from the station.
[4]Instrument Landing System - Presently the primary means for performing precision instrument landing approaches to an airport.

be monitored and managed by everyone from squad leaders to theater commanders. In addition, logistics personnel can more rapidly predict the need for specific supplies and more quickly transport them to where they will be needed. The result is a "force multiplier" in which fewer people with less equipment and fewer supplies will be able to accomplish the same missions as before. However, it also creates a critical reliance on the GIG where any significant disruption can threaten mission accomplishment.

Clearly, any link that forms a critical path in any mission-critical communication system will need to be designed and employed with jam-resistance in mind. This is particularly true in adversarial environments, such as anti-terrorism or tactical military operations, were malicious players exist whose objectives would be furthered by the disruption of those links.

B. Defining Jamming and Jam-Resistance

To appreciate what "jam-resistant" means, it is necessary to establish what "jamming" means. Within the framework of the four classic goals of secure communications (availability, confidentiality, authenticity, and integrity), jamming directly attacks the integrity of a communications resource, frequently as a means of indirectly attacking its availability.

Jamming typically involves artificially injecting a second signal into a communications channel that combines with the legitimate signal in such a way that the receiver's ability to recover the correct information is significantly impaired. This second signal can be as simple and brute force as wideband noise transmitted with sufficient power to overwhelm the legitimate signal, or it might be very carefully crafted signals intended to mislead the receiver into misinterpreting the combined signal it receives.

Brute force jamming with broadband noise is arguably the surest way to jam a radio broadcast. It also tends to be one of the most costly by several measures. First, the raw energy requirements are an issue, especially if the jamming platform is small, remotely deployed, and battery-powered. Second, the power required to effect such jamming is often sufficient to permit the defenders to detect and localize the jamming source. In most civilian environments this serves to increase the risk of apprehension and prosecution; on the battlefield, consequences tend to be more direct and permanent.

Attackers therefore prefer jamming methods that minimize energy expenditure, if for no other reason than self-preservation. The ideal method permits a minuscule amount of energy, significantly less than that used by the legitimate sender, transmitted in a single burst to corrupt a legitimate message just enough to prevent its accurate reception. In other words, they seek to marginally increase the channel's BER (bit error rate). If the receiver is able to identify a message as being corrupted beyond recovery they must request a retransmission (which may also be initiated by the sender if an expected acknowledgement fails to arrive). This reduces the effective bandwidth, and hence availability, of the channel.

It is important to avoid equating jam-resistant with jam-proof. If the attacker is willing and able to commit sufficient energy and resources, they will always be able to sufficiently interfere with a legitimate signal so as to render it useless - this is true even for hardwired networks. As in many adversarial environments, the goal is not to create a situation in which the opponent absolutely cannot triumph, but rather one in which the cost of doing so is prohibitive.

The cost of jamming is a complex computation involving such factors as time, effort, resources, energy, and risk. While the defender, in theory, only has to make the cost of one component sufficiently high, the attacker can respond by trading reliance on that component for others. Hence the classic game of counter-measures, counter-counter-measures, and so on in an endless cycle.

With this understanding comes the realization that characterizing the jam-resistance of any communications channel is neither trivial nor static. In general, it must take into account the type of attacks that are possible, the various costs associated with them, and the type of counter-measures that are available. Most of that is well beyond the scope of this work and hence the measure of jam-resistance that will be used will necessarily be much more narrow and abstract.

A reasonable measure of jam resistance relates the effort required by the receiver to recover the original message to the effort expended by the attacker. As previously noted, however, "effort" is a very elusive term in this context. For our purposes, it is reasonable to define "effort required by the receiver" to be a measure of the computational effort required to recover the original message while "effort expended by the attacker" will generally refer to how much energy the attacker is transmitting relative to the legitimate sender. The primary motive for this definition is to permit quantitative analysis and comparison of the various jam-resistant schemes considered.

C. The Use of Directionality to Achieve Jam Resistance

A high degree of jam resistance can be attained by the use of highly directional signal propagating elements - if the attacker can only interfere with the signal from a limited number locations, then the defenders gain a significant advantage by controlling where those locations are and ensuring that the attackers do not have easy access to them. For instance, satellite downlinks can be made quite jam-resistant by using high-gain receiving antennas. By having a very narrow reception arc that is directed well above the horizon, the attacker's problem of injecting a malicious signal into that arc is a challenging one, particularly if the receiver is placed on board an aircraft. While not as effective, purely terrestrial channels can be made quite jam-resistant by employing very narrow beams of energy, such as radio-frequency or laser light. This also includes the growing range of beam-forming and steering technologies, both for transmission and reception. Another very effective means is the use of wired connections, such

as copper or fiber optic, to carry the traffic. While not typically thought of in those terms, it is difficult to image anything more highly-directional than a piece of wire. Similarly, a physical message courier can be categorized as highly-directional.

It's worth noting that channels that base their jam-resistance on high directionality are really relying on preventing the attacker from injecting the jamming signal at all - they are not necessarily any more successful than channels that claim no particular jam-resistance at tolerating those jamming signals that do get injected.

While highly-directional links offer real benefits, they are not without serious shortcomings as well. In particular, they are typically not very scalable or flexible. A significant amount of preparation is usually involved establishing them, the number of users they can accommodate is relatively limited, and they generally cannot be readily adapted to changing needs.

D. The Use of Shared Secrets to Achieve Jam Resistance

Reliance on directionality for jam-resistance is not always an option. In particular, jam-resistance must extend to omnidirectional radio links as well because their scalability, simplicity, and flexibility will virtually assure their use, possibly even dominance, in the types of large-scale, flexible, highly-mobile battlefield ad hoc networks that will be found in tactical environments.

The strict definition of "omnidirectional" requires that signal propagation (transmitter) and sensitivity (receiver) patterns be independent of direction. In practice, true omnidirectionality is nearly impossible to achieve. This is particularly true in radio-frequency systems, where a system is generally considered omnidirectional if it has little directional dependence within a given plane (generally horizontal).

Omnidirectional systems are, by nature, wireless (provided "wireless" is taken to mean "without signal directing elements" and applied generically to include non-electromagnetic communication systems, e.g., acoustic systems). The reverse, of course, is not true; wireless communication channels can most certainly be highly directional.

For this treatment, a system will be considered omnidirectional if the intended recipients have no preferential access to the transmitter's signal relative to unintended recipients and, similarly, if the receiver is as sensitive to an attacker's signal as they are to that coming from the legitimate sender. These are reasonable assumptions in many tactical environments where the rapid, unpredictable movements of ally and adversary alike make this a practical reality.

Omnidirectional alternatives generally bring extensive simplicity and flexibility to the situation, but suffer because the attacker has essentially the same access to the channel as the legitimate parties and, hence, can detect what the sender is transmitting and affect what the recipient is receiving fairly easily. To overcome

this disadvantage, legitimate parties can collude ahead of time and share special knowledge, known as a "shared secret" or "symmetric key", about how the channel will be used. If they are successful at depriving the attacker of this knowledge, they can shift the odds significantly in their favor. For instance, in the case of spread spectrum, the hop sequence, spreading code, or other information used to spread the spectrum is assumed to be such a secret and virtually all jam-resistance disappears if that secret is compromised.

E. The Key Management Problem

While effective, the use of symmetric keys also serves as an Achilles' heal. Should the attacker gain access to the key, they gain the ability to jam the channel. A large part of the defender's task therefore becomes key management - distributing keys to legitimate users, denying them to unauthorized parties, detecting when keys have been compromised, and ensuring that legitimate users do not use compromised keys.

There are several obstacles to achieving proper key management, the most significant of which are the distribution, security, and perishability problems. The logistics involved in simply distributing the keys becomes a major problem as the size of the network grows. It is one thing to give all the members of a special operations team or a flight of fighter aircraft the keys they will need to communicate amongst themselves just prior to a mission, it is quite another to ensure that the front-line soldiers from an allied nation have the keys necessary to access real-time target intelligence coming from yet another allied nation half a world away. Another major problem is keeping the keys secret - the more people that have access to the key, and the longer ahead of time they have that access, the greater the likelihood that the key will be compromised. To address this, it is necessary to have strict accountability for the keys. In small scale networks this is reasonably doable, but the problem quickly becomes untenable as the number of people with access grows. Related to this is the problem of detecting when keys have been compromised - perhaps the only thing worse than the enemy learning your secrets is for the enemy to do so while you still believe they are secure. While small teams of people are much more likely to know when their keys have been compromised (compared to large networks), it is still generally the case that sensitive keys are only used for a short period of time - in essence it is assumed that the adversary will obtain the keys after a certain period of time and hence they are thought of as perishable commodities. When all three of these problems are combined, it becomes self-evident that the need to rapidly distribute short-lived keys to large numbers of people in a secure fashion is not only challenging, but a practical impossibility for all but the smallest of networks.

In some instances key management is a moot point because reliance on shared secrets is simply incompatible with certain types of communications - in particular, public-access systems such as civilian-

side GPS have authorized user pools that include every person on the planet. To the degree that those systems are also omnidirectional, they have essentially no jam resistance at all. Since incorporating a high degree of directionality will not always be practical (or even possible) the need for jam-resistant omnidirectional communications between parties having no prior shared secret will only grow.

F. Analogy to Public-Key Cryptography

This situation is not without precedent and an understanding of a similar historical problem is quite useful for framing this discussion. Until the advent of asymmetric cryptography in the early 1970's,[7], [8], [9], [10] it was nearly universally accepted that secure encrypted communications required the prior exchange of keys via a secure unencrypted channel. For entities such as governments and large corporations, for whom the overhead and expense of distributing keys was an acceptable cost for securing their internal communications, such a limitation was merely a significant hindrance - not only did these organizations have the resources to deal with the distribution problem, but the number of parties to whom the keys had to be distributed was relatively small and manageable. In contrast, however, the cost and difficulty of securing large scale personal and consumer transactions via the secure distribution of keys quickly approaches the unimaginable.

Attempts to extend cryptographic protection to larger networks has included a variety of strategies, but all have required prior secure communications - if not with the other party than with a trusted third party. Perhaps the most evolved such system is Kerberos[11], [12], developed originally at MIT during the early 1980's. Kerberos permits secure communications between two parties who have no prior knowledge of each other over an insecure network by acting as a central clearing house for session encryption keys. However, the system requires that each party have a prior relationship (and shared secret) with the Kerberos server and that the Kerberos server be available at the beginning of the communication session. If either of those requirements is not met, communications cannot commence.

With appropriate asymmetric algorithms, however, it became possible to securely communicate with another party using only publicly accessible information. This capability enables the existence of a still-expanding Public Key Infrastructure (PKI) for the distribution of this public information thereby facilitating authentication and secure communications between parties having no prior relationship[13], [14], [15], [16]. Even without PKI, it is possible to securely exchange information over an insecure network using, for instance, Diffie-Hellman - what is lost is the ability to authenticate the other party.

It would be incorrect, however, to conclude that asymmetric algorithms supplanted symmetric ones. Present asymmetric algorithms are very computationally intensive and it is highly likely that symmetric algorithms will always be the more efficient of the two. This is simply due to the significantly greater

constraints placed on the behavior of an asymmetric algorithm compared to its symmetric brethren.

As a result of this disparity in computational expense, the data throughput for an asymmetrically encrypted channel is generally much lower - typically one or two orders of magnitude - than for a symmetrically encrypted one. To overcome this penalty while maintaining the independence from shared secrets, a hybrid approach is typically used whereby an asymmetric channel is used initially, but only for the purpose of exchanging the short-lived session keys necessary to set up an appropriate symmetric channel. In modern communications, the overwhelming use of asymmetric algorithms, such as Diffie-Hellman key exchange and RSA, is for the secure exchange of symmetric session keys between parties while symmetric ciphers such as DES3 and AES continue to carry the bulk of the traffic.

In practical terms, then, the key management issues still exist, but they have been partitioned to a hyperfine degree. Instead of a central authority generating keys that many parties will use for an extended period of time and devising a way to securely transfer them, the keys are generated on the fly, distributed to only two parties, only exist for the duration of a single conversation, and are securely transmitted over an insecure channel as part of the asymmetric handshaking protocol.

With regard to jam-resistance, the situation in the world of omnidirectional wireless communications is quite analogous to that faced by the cryptographic world prior to about 1970. Modulation schemes such as spread spectrum are almost direct counterparts to symmetric ciphers in several key respects: (1) efficient, high-bandwidth, jam-resistant communications are possible, provided a shared secret can be successfully exchanged; (2) shared secret distribution is a limiting factor in the large scale deployment of such jam-resistant channels; (3) what is needed is an analogous means of exchanging keys via an insecure channel; and (4) significant penalties in data throughput and processing cost can be tolerated while performing the key exchange.

G. Avenue of Research

Put simply, this avenue of research has the goal of devising a means of transferring a message from a sender to a receiver via omnidirectional broadcast in the face of significant hostile jamming even when the jammer possesses any and all knowledge that is common to both legitimate parties.

Traditionally, jamming has the effect of altering a message (an attack on message integrity). In a communications system, this takes the form of turning some zero-bits into ones and some one-bits into zeros. The receiver is then faced with the task of detecting that the message has been altered and either correcting it to recover the original message or requesting a retransmission (which is, of course, subject to being jammed as well). In most communications systems, a great deal of effort is expended to make the bit-error probabilities symmetric - that is to say that the probability that a zero-bit and a one-bit will

get corrupted are nominally the same. While this usually results in the lowest overall system bit-error rate (BER), this is only the case for jamming that is essentially random in nature. Jam resistance in such (omnidirectional) channels is reliant primarily in the security of the key and, should that be compromised, on the robustness of any error-correcting codes in use in the face of any heightened bit error rates due to the presence of a jamming signal.

The approach taken here is essentially the reverse. In recognition that symmetric bit error probabilities do very little to aid in keyless jam-resistance, we aim to develop protocols that exploit the inherent highly asymmetric error probabilities of certain physical transmission schemes so as to fundamentally change the effect that jamming has. For simplicity, we will call the bit symbol that has very low error probability a "mark" and assign it to be the one-bit while the bit symbol having a higher error probability we will call a "space" and assign as the zero bit. Hence the attacker can fairly easily change our spaces into marks, but has a very hard time changing our marks into spaces. If we then develop a message coding protocol that only relies on the successful reception of all (or nearly all) of the marks in the transmitted codeword, the attacker will have a very difficult time preventing reception of the message.

By forcing the attacker to operate under these constraints and making it very difficult for them to actually alter the original message, they must resort to obscuring it by flooding the channel with false messages (a direct attack on channel availability in lieu of being able to attack channel integrity). If the receiver is able to recover all of the messages that were combined in the transmission, regardless of source, then they have the opportunity to sift through them and identify the correct one. There are a number of potential ways of doing this, but perhaps the most attractive one is to include a digital signature and authentication certificate with each message in a fashion quite analogous to how present key-exchange protocols work in e-commerce.

For this strategy to work, however, we must devise a means of taking messages that have been transmitted concurrently - with arbitrary phase relationships - and separating them at the receiving end. If this can be achieved, the attacker must resort to inserting multiple complete messages into the channel to block it and their energy expenditures rise proportionately. Instead of using a fraction of the energy consumed by the legitimate sender, they may now have to commit many times the energy - with corresponding increases in cost.

Furthermore, if the protocols are sufficiently flexible, the receiver gains the option of tolerating higher levels of jamming at the expense of more processing overhead to extract the legitimate message from the greater number of false messages. While the attacker ultimately still has the ability to block the channel, the result is a communications protocol that continues to function in the presence of significant, malicious, interference well beyond the point at which present omnidirectional systems would succumb once their

keys are compromised.

For reasons that will be examined in the discussion on work done to date, existing branches of coding theory, namely error detecting and error correcting codes, are not up to this challenge. What is needed is a new coding theory that describes the way arbitrary messages behave when they are combined in what is termed a "multiple-access OR" channel. Upon that foundation can then be built suitable encoding and decoding (codec) algorithms that exploit those behaviors to achieve a predicted level of jam-resistance. That new theory is the theory of concurrent codes and the principle codec algorithm that has evolved from it is the BBC algorithm.

While a purely theoretical treatment of this new theory might suffice, it is also the intent of this research effort to demonstrate that practical systems based on concurrent codecs are realizable without exotic technologies and that they perform in close accordance with the theory.

III. Literature Search

The literature search for this project focussed on two areas: (1) What work has been done to enable jam-resistant communications in omnidirectional systems in the absence of a symmetric key, and (2) what is the existing body of knowledge relating to the behavior of binary sequences combined via a bitwise-OR.

A. Relevant Work on Jam-Resistance

Since one of the chief claims of this research is that it is the first (successful) effort to achieve significant jam-resistance in an omnidirectional system in the absence of an uncompromised symmetric key, this part of the search equated to attempting to prove a negative. Numerous papers were found that dealt with the jam-resistance of communication systems. For instance, in the case of spread spectrum systems, much attention has been given to performance against jamming and various mitigation strategies both generally [17], [18], [19] and in specific subfamilies including direct sequence [20], [21], [22], [23], [24], [25], [26], frequency hopping[20], [27], and pulse-position systems[28], [29], [30]. However, all employ either directionality or symmetric keys. Similarly, examples of fielded systems that claim significant jam-resistance were also not hard to find, but again all of them rely on directionality and/or symmetric keys.

Given the growing importance of jam-resistance in general, it is unlikely that all substantive work in this area would have evaded the major research publications and databases such as CiteSeer, IEEExplore, and Engineering Village (Compendex)

B. Relevant Work Superimposed Codes

In contrast, exploration of the behavior of binary sequences that have been combined using a bitwise-OR enjoys a rich history over the past six decades. However, it took quite a while for this to become

apparent because the topic has been studied by several highly disparate communities, ranging from pure math theorists to practical information retrieval system designers. These groups appear to have had little interaction with each other in this topic area and, as a result, a plethora of terminology exists and many different words and phrases are used for the same, or at least similar, concepts. The most widespread term for the bitwise combination of messages is *superimposed codes*[31], [32], [33], [34], although names such as *Bloom filters*[35], [36], *strongly-selective families*[37], and *r-cover free families*[38], [39], [40], [41], [42] have strong followers in the various communities. Many of these codes have been constrained in various ways to improve specific performance parameters[43], [44], [45], [46], [47], [48], [49], [50], [51], but none have focussed on providing an efficient means of decoding large collections of codewords that have been bitwise-OR'ed together. Typically, these codes are for fairly small codebooks, such as one codeword per user in a system or cell-culture in a study, so it's more likely to be on the order of thousands (or perhaps millions) of codewords rather than exponentially large codebooks such as 2^{1000} [52], [53], [54], [55], [56].

One indication of the overall significance of superimposed codes is that the U.S. National Institute of Standards and Technology (NIST) maintains a definition for a superimposed code: "A set of bit vectors such that no vector is a subset of a bitwise-OR of a small number of the others"[57].

In keeping with tradition, yet another name is introduced here, namely *concurrent codes*. However, concurrent codes are not just superimposed codes by another name, concurrent codes are a subset of superimposed codes possessing a property that has previously eluded other researchers. Specifically, concurrent codes are the subset of superimposed codes that can be efficiently decoded, preferably in linear time.

For simplicity, it is useful here to define a single vocabulary that will be used throughout this document and then use that vocabulary, wherever possible, even when discussing the work of a particular field that prefers a different nomenclature. Translating the NIST definition into this terminology, the "bit vectors" are "*codewords*" and the "bitwise-OR of a small number of (them)" is a "*packet*". If a codeword is a subset of a packet, it is said to be "*covered*" by or "*contained*" in the packet. Hence, the NIST definition of a superimposed code in our terminology would be, "A set of codewords such that no codeword is contained in a packet composed of a small set of other codewords."

What has become fairly apparent after reviewing the literature is that nearly all of the existing work has proceeded from a central premise - that the set of codewords is relatively small and, where necessary, fully enumerated. It is a simple matter to determine if a specific codeword is contained in a packet, whereas identifying all of the codewords that are contained in a packet is a much more challenging task. However, if the set of codewords is sufficiently small, it can be accomplished in an acceptable amount of time via

exhaustive search. In fact, the inability to extract codewords from a packet except via exhaustive search has been touted as a benefit for some applications.

That so much attention has been devoted to superimposed codes without a corresponding amount of effort spent attempting to devise a means to efficiently decode packets containing them may seem odd at first. However, a brief overview of the types of applications they arose from sheds quite a bit of light on this apparent oddity. The earliest use of what would become superimposed codes appears to have originated by Robert Dorfman and David Rosenblatt in the early days of the United States involvement in World War II[58]. Faced with a need to screen millions of service inductees in order to identify a few thousands of cases of syphilis, Dorfman and Rosenblatt, economists by training, devised a means of significantly reducing the total number of tests that must be performed by pooling blood samples and testing only the pools. If, they asserted, the initial screening was performed by grouping the blood of five inductees into a single test sample, then if the test returned negative there was no need to perform further tests on any of those five people. If, however, the test was positive then the individual(s) infected could be identified by retesting just those five people (using blood held back for that purpose). Since most of the pools would return negative results, the total number of tests performed would only be a little over one-fifth of the number needed for individual testing. As it turned out, this method of screening for syphilis was never put into practice for a variety of reasons, however Dorfman published his proposal[59] and hence was started the field of "group testing".

One of the most frequently cited works in the field of superimposed codes is the 1964 paper by Kautz and Singleton[31], in fact many papers confer upon them the honor of inventing superimposed codes. However, Kautz and Singleton themselves, in that initial work, refer to the 1958 work of Schultz[60]. Schultz's work involved the effort to categorize and cross reference the unprecedented amount of technical literature that had developed during the World War II. A variety of schemes had been proposed, some practical and some not, of performing data searches on large volumes of information. One such method involved taking a large index card with the reference information for of a publication and punching a line of small holes along one edge. Then the publication would be classified according to a list of categories and for each category in which it belonged a subset of the holes assigned to that category would be turned into slots extending completely to the card edge. Each card would therefore end up with a set of holes and slots which served to identify all of the categories to which the publication belonged. However, since there were many more categories than there were holes, it was necessary for the same slot to be used for multiple categories. When a search needed to be performed, a stack of cards would be assembled and small rods would be passed through the holes associated with category of interest. Then the rods would be held

up and any cards that had slots at the locations of all the rods would drop out of the stack. Any card which did not drop would be known to not belong to the category being tested, although it was possible that some "false-drops" would occur meaning that some cards might drop that did not belong to the category. As computers, then very much in their early days, were developed, this manual index-card based system lent itself quite naturally to the migration to computer-read punched cards. Much of the subsequent development of superimposed codes arose from devising various ways to minimize the probability of false drops while maximizing the number of categories that could be encoded using a limited number of holes.

As (relatively) modern computers continued to find new applications, superimposed codes gained interest in a variety of venues, but nearly all had at their core the goal of membership testing within a group. Perhaps the archetypical example, generally associated with Bloom filters[35], is the notion of a dictionary. In this example each word in the dictionary is encoded into a codeword and then all of the codewords are combined into a packet. At this point, a spell check can be performed by taking each word in the document, encoding it, and testing if the codeword is covered by the packet. If it isn't, then it is guaranteed that the word is not contained in the dictionary. If it is, then the word probably is in the dictionary, but there is a slight chance that it is not. Bloom and others spent considerable effort devising codes that minimize the probability of false hits, but the premise has always remained that the areas of use would continue to involve membership testing. A consequence of this is that, while it is easy to test if a particular word is in the dictionary, it is impossible to take the dictionary packet and generate a list of all the words in the dictionary. This one-way behavior has been offered as being a feature in the sense of the following example: All of the social security numbers of all of the people in a particular group, say people that the government owes money to, are encoded and combined into a packet. The packet is now made public and any person can easily test whether the government owes them money. However, it is impossible to generate the list of social security numbers contained in the packet except by exhaustive search (which, today of course, would not be much of a hurdle, but the idea is there).

The inability to efficiently decode an arbitrary packet in the face of an exponentially large set of codewords has not been of major consequence for most of the work described in the literature. Most of it has been concerned with determining the likelihood that a given codeword will be incorrectly determined to be contained within a packet even when it was not one of the codewords used to form the packet. This is not to say that the desirability of efficiently decoding packets drawn from very large codebooks has never been recognized. However, as recently as 2003 Cormode and Muthukrishnan concluded that no method short of exhaustive search existed for decoding[61]. It's not known if these authors were aware of work published in 1988 that did present a relatively efficient (i.e., polynomial time) algorithm for

decoding packets containing fairly small numbers of codewords[62], [63], [64], [65], [66]. Even if such algorithms were known to them, it's quite conceivable that they would have been deemed inadequate since the algorithms were of complexity $O(n^3)$ or $O(n^4)$, hence packets containing 1000 codewords could have required as many as a trillion iterations of the algorithm. In contrast to this, the BBC algorithm developed here decodes packets in linear time, $O(n)$.

IV. Work to Date

The key results that have been achieved to date are summarized in this section. First, we give a highly simplified example of just what a concurrent codec is, including some of the issues and related terminology. We then examine the basic requirements of a concurrent codec, and establish that conventional coding theory is inadequate to address those basic requirements. After briefly considering where concurrent codecs fit within a traditional network stack, we proceed to examine three unsatisfactory codec algorithms: straight-through, one-hot, and random. The first, while completely inadequate, helps establish a useful discussion framework by illustrating most of the mechanics and issues involved. In contrast, the other two do address the jam-resistant requirements of a concurrent codec, but are not scalable in practice. Their examination is useful because they establish some important performance bounds.

Following this preliminary work, we examine a codec based on superimposed codes that possesses all of the necessary characteristics to provide a high degree of jam-resistance provided it can be efficiently decoded. We then present concurrent codes and the BBC algorithm that has been developed specifically to efficiently encode messages and decode concurrent packets. Finally, we discuss two demonstrations that have been developed, one involving visualization and the other sonification. In the interest of brevity, the details of most derivations will be omitted but will appear in the final dissertation.

A. Concurrent Codecs 101

A transmitting codec (coder/decoder) takes a message and encodes it into a codeword. This codeword is then transmitted, possibly becomes corrupted in the transmission channel, and is finally received by the receiver in its corrupted form. A conventional receiving codec then attempts to recover the original message from the corrupted codeword, or at least detect that corruption has occurred. While codecs based on error-correcting codes can be made quite robust against non-intelligent noise, they are easily defeated with intelligently crafted noise. In essence, error-correcting codes are too clever for their own good. What is needed is a codec that makes no attempt to figure out what the true original message was but, instead, simply generates a list of all possible messages that are consistent with the received packet regardless of the amount of corruption. Such a codec is defined as being a *concurrent codec*. In the case of a transmission channel with symmetric error probabilities, every codeword is consistent with any received packet (under

these conditions) and the resulting message list would always contain every conceivable message. However, with a sufficiently asymmetric channel, only those messages whose codewords are *covered* by the received packet are included in the list. To be covered a codeword must have all of its marks in the packet.

One consequence of an (ideal) asymmetric channel is that the symbol having near-zero probability of error becomes, in essence, an *indelible mark* in that the attacker cannot make that symbol disappear (i.e., not be received). Another consequence is that codebooks (i.e., the collection of all possible codewords) become hierarchically organized. This implies several things: (1) there is a "root packet" which is a packet that could be created by any legitimate codeword given sufficient corruption; (2) channel noise can only move packets toward the root packet, never away from it; and (3) the list of codewords generated by a concurrent codec in response to receiving a packet is simply all codewords that can be reached starting from the packet and moving outward away from the root packet - or, in the jargon of graph theory, all of the children that have the received packet as a parent.

As with conventional coding theory, the nature of the codebook cannot be separated from the characteristics of the channel. However, in conventional coding theory, this is a matter of making codes that are more effective. With concurrent coding, it is a matter of making codes that work at all. Once again, this is a consequence of dealing with an active attacker. We accept that an attacker can corrupt the packet, but we cannot permit the attacker to corrupt the packet in any arbitrary fashion of their choosing. Instead, we exploit the asymmetry of the channel to constrain the nature of that corruption and then devise a corresponding codebook that exploits that constraint.

Clearly the root packet for a concurrent codec is a packet containing all marks. If the attacker can force us to receive the root packet then the channel is completely jammed. In practice, the channel becomes effectively jammed if the attacker forces us to receive a packet that is too close to the root packet. Hence our anti-jamming strategy can be concisely summarized as needing to be able to operate as close to the root packet as feasible while simultaneously making it too expensive for the attacker to force us to operate any closer.

For illustration purposes, let's consider a toy example having four-bit messages with an *expansion* of eight (i.e., the codewords are eight times as long as the message, thirty-two bits in this example) and a *mark factor* of one (i.e., there is one mark in the codeword per message bit, four marks per codeword in this example).

Each of the sixteen possible messages maps to a unique codeword. The mapping is performed by the encoder portion of a codec using some algorithm. For our present purposes it is sufficient to simply list the messages and the corresponding codewords, the algorithm used is irrelevant (for those interested, the algorithm was simply "randomly" placing marks in in each codeword making sure that each one received

exactly four marks and that the last few were placed in columns that didn't already have at least one). The resulting codebook is shown in Table I.

TABLE I
EXAMPLE (4,8,1) CODEBOOK

#	Message	Codeword
0	0000	00010000 00100000 00000010 01000000
1	0001	00000000 00010000 00001010 00010000
2	0010	01000000 01000000 01001000 00000000
3	0011	00001000 00010000 00000000 10000001
4	0100	00000000 10000010 00010000 00010000
5	0101	00010000 00000001 10000000 00000100
6	0110	00100000 00010000 00010010 01000000
7	0111	00011000 00001000 00000000 00100000
8	1000	00000000 00100100 00000000 00100100
9	1001	00010010 00010000 00000100 00000000
10	1010	00000000 00000100 00000000 00101010
11	1011	00100100 01000000 00010000 00000000
12	1100	00010000 00001000 00000001 00001000
13	1101	10000001 00000100 00010000 00000000
14	1110	00010100 00100000 00100000 00000000
15	1111	00001100 00000000 00001000 00000010

Because all codewords are unique, any time a single message is encoded and transmitted only that message will be decoded at the other end unless some form of corruption occurs. As marks are added to the packet, either by noise or an attacker, eventually additional codewords will be covered by the packet and the codec will add the corresponding messages to the list. Since it is assumed that the attacker has access to the same information as the legitimate sender, they will always be able to force a concurrent codec to add additional messages to the list while exerting no more energy per message than was required by the legitimate sender, simply by using the same process.

Consider a packet formed by the legitimate sender transmitting message 3 and the attacker sending messages 6, 9, and 12. Assuming the codewords are perfectly aligned results in the packet shown in Table II. When these messages are decoded (by brute force examination of each codeword in the codebook in this case) it is found that the packet covers messages 3, 6, 9, and 12 and, hence, these are the messages in the codec's output list.

In contrast, consider a packet formed using the legitimate message 2 and attack messages 6 and 14 (See Table III). Because of the interaction between marks contributed by the individual codewords, the resulting packet covers additional messages, namely 0 and 11, that were not part of the generating list.

As the previous examples illustrate, a concurrent codec can produce output messages in addition to

TABLE II

EXAMPLE WITHOUT HALLUCINATIONS

#	Message	Codeword
Generating Message List		
3	0011	00001000 00010000 00000000 10000001
6	0110	00100000 00010000 00010010 01000000
9	1001	00010010 00010000 00000100 00000000
12	1100	00010000 00001000 00000001 00001000
Packet		00111010 00011000 00010111 11001001
Decoded Message List		
3	0011	00001000 00010000 00000000 10000001
6	0110	00100000 00010000 00010010 01000000
9	1001	00010010 00010000 00000100 00000000
12	1100	00010000 00001000 00000001 00001000

TABLE III

EXAMPLE WITH HALLUCINATIONS

#	Message	Codeword
Generating Message List		
2	0010	01000000 01000000 01001000 00000000
6	0110	00100000 00010000 00010010 01000000
14	1110	00010100 00100000 00100000 00000000
Packet		01110100 01110000 01111010 01000000
Decoded Message List		
0	0000	00010000 00100000 00000010 01000000
2	0010	01000000 01000000 01001000 00000000
6	0110	00100000 00010000 00010010 01000000
11	1011	00100100 01000000 00010000 00000000
14	1110	00010100 00100000 00100000 00000000

those sent by the genuine sender. These additional messages can come about in one of two ways - either because the attacker explicitly inserts them or because of interactions between messages (and other noise) in the packet. Intentionally inserted codewords are called *rogues* to distinguish them from those transmitted by the legitimate sender. Spontaneously appearing codewords are called *hallucinations* because they don't really exist but, like any good hallucination, are hard to distinguish from the real thing.

From the receiver's standpoint the distinction between rogues and hallucinations is largely, though not totally, superfluous. The concurrent codec must extract hallucinations and well as rogues but, since hallucinations are essentially random messages, the application may be able to identify many of them as invalid without expending a significant amount of effort. Whether this distinction matters in the end depends on where the processing bottleneck occurs - the concurrent codec or the application.

However, from the attacker's perspective the distinction is critical - they have to pay for rogues but hallucinations are free. If they can identify a set of rogue messages that produce a significant number

of hallucinations, then they can greatly reduce their average energy per message and hence have a much higher probability of jamming the channel. Their ultimate goal is to identify small sets of rogue marks that are highly hallucinogenic.

B. The Basic Requirements of a Concurrent Codec

To be acceptable for use in a practical implementation, a concurrent codec must be jam-resistant, efficient, and scalable. Clearly jam-resistance is the most important consideration, for without that the whole exercise is moot. But even an ideal system is worthless if it cannot be fielded in practice. Fortunately, the jam-resistance does not have to be perfect, it merely has to be good enough. This permits the designer to trade off jam-resistance in favor of attaining acceptable efficiency and scalability. Efficiency will almost certainly be important to most applications. While each particular application will be different, in general the implementation will have to be designed with processing, energy, space, and cost efficiencies in mind. Most applications are likely to be real-time in nature, but this is not to say that significant amounts of processing effort can't still be expended. For example, in the case of key-exchange, the transactions are very limited in both scope and duration and although the time available to arrive at a solution is still measured in seconds, as opposed to weeks or even minutes, this still permits a great deal of processing burden compared to a system that must be responsive on the microsecond or millisecond timescale. Other systems are likely to involve mobile units, perhaps including small, unmanned aerial vehicles, where space, weight, and power requirements drive the design. Still others might involved distributed sensor networks made up of thousands, perhaps millions, of remote sensors that have to be extremely low cost, and probably small and low powered, to be viable. However, the asymmetry of the responsibilities between sender and receiver might permit the receiving station to carry a load burden that the sender could not accommodate. Finally, in addition to being scalable with respect to cost, the solution must also be scalable with message length. In order to be suitable for key exchange purposes, it will be necessary to transfer messages that are on the order of several hundred to perhaps a few thousand bits. Algorithms that are not of polynomial time or better are categorically unsuitable. In fact, the limitations of algorithms that are not linear, or near-linear, in complexity are likely to prove too severe for realistic message lengths.

C. The Relevant Performance Metrics

To evaluate codec designs, it is necessary to devise suitable measures for their performance against the three often competing requirements of jam-resistance, efficiency, and scalability. Unfortunately, as previously discussed, the truly relevant metrics cannot be developed in isolation to the intended application. We can, however, develop a set of abstract metrics that are useful in comparing potential codec designs relative to each other. Since jam-resistance is the primary requirement, we will develop a quantitative metric

for that measure. For the other two, we will simply note that the efficiency and scalability are generally related to the amount and type of processing that must be performed and also to the degree to which that processing can be parallelized.

Any suitable metric for jam-resistance must relate the amount of effort expended by the attacker to the detrimental effect that effort has on the receiver. In traditional systems, this generally takes the form of determining the effect that the jammer's average transmission power has on the receiver's bit error rate (BER). In a system based on concurrent codes, the receiver's BER is expected to be very nearly a step function remaining very low until some threshold is reached at which point the receiver will quickly become completely jammed. While identifying this threshold as a function of attack effort is important, it does not suffice as a complete metric. As the jammer increases their effort, they can force the receiver to perform additional work and hence do have a detrimental effect even while the BER remains low. In fact, in many systems the amount of additional work the receiver can handle will determine the performance limit of the system and not the point at which the BER rises precipitously. As a result, the most relevant metric relates the receiver's effort to recover a message to the attacker's effort to interrupt that recovery.

The attack effort is most naturally defined in relative terms: the ratio of the energy broadcast by the attacker to the energy broadcast by the message originator. For OOK systems, the energy transmitted is proportional to the *degree* of the packet (which, by definition, is equal to the number of marks in the packet). This, in turn, is proportional to the packet mark density. Hence any of the ratios in Eqn. 1 will suffice. We will generally express this ratio in deciBels which is reasonable since it is a measure of relative energy.

$$\zeta_A \equiv \frac{E_A}{E_S} = \frac{d_A}{d_S} = \frac{\mu_A}{\mu_S} \tag{1}$$

However, we note that this may or may not be a suitable definition for a given physical channel depending on the modulation scheme used. Furthermore, it does not take into account differences in transmitter capabilities - for instance, if the jammer it nearby, they might be able to create a mark in the packet using considerably less energy than the legitimate sender. Conversely, if they are further away they may have to use considerably more energy to do so. However, this shortcoming is the same for the analysis of traditional systems as well and the normal approach is to discuss the problem in terms of the noise generated at the receiver's input leaving it to the person solving a particular problem to convert that into the actual power ratios between the transmitters. That same concept applies here.

Measuring the receiver's effort also has a natural definition, although it is somewhat more arbitrary. A concurrent codec must recover a list of all possible messages that are covered by the received packet.

The receiver must then process those messages to discriminate which, if any, are genuine. The channel becomes jammed if either the concurrent codec requires more time or resources than available to generate the message list or if the list is too long for the receiver to process within its time and resource budget. In most cases, the processing effort required for both tasks, recovery and discrimination, scales with the length of the message list and hence the ratio of the number of messages recovered, M_R, to the number of messages that would have been recovered in the absence of jamming, $M_R^{'}$, will be used as the primary metric (Eqn. 2). In most cases, the number of messages that would be recovered in the absence of jamming is simply the number of genuine messages sent; where this is not the case it will be dealt with appropriately. Like the attacker's effort, the receiver's effort will also usually be expressed in deciBels, even though the relationship to energy ratios is only inferred. In cases where this metric is not suitable, additional appropriate metrics will be defined.

$$\zeta_R \equiv \frac{M_R}{M_R^{'}} = \frac{M_R}{M_S} \tag{2}$$

If we plot the relative receiver effort versus the relative attack energy we obtain a performance characteristic for that concurrent codec. As mentioned previously, both the receiver and the attacker have associated limits - the receiver has a maximum amount of effort they can afford to expend to recover the message while the attacker has a maximum amount of energy they can commit to the attack. The intersection of these two limits serves as the performance criterion for the system and determines the acceptability (for jam-resistance) of the codec. If the criterion lies above the characteristic, then the receiver can process any amount of jamming the attacker is willing to generate. Conversely, if the criterion lies below the curve, then there is an amount of jamming the attacker is willing to produce that will overwhelm the receiver's ability to cope.

Quantitatively, it is impossible to place exact, or even static, limits on either the amount of relative effort the receiver can afford to expend or the amount of energy the attacker is willing to commit. In fact, concurrent codecs that have fixed performance curves are probably not desirable. Instead, we want a codec that is extensible so that, by changing a few parameters, the curve can be shifted to a more advantageous location - usually in exchange for more absolute effort on the part of both the originator and the recipient. This is analogous to using a cryptographic system that permits the users to select longer length keys to gain cryptographic security in exchange for longer encryption and decryption efforts.

We can, however, gain some appreciation for the performance graphs by noting that if a receiver has to process ten thousand messages to find the one original message, it is expending an additional 40 dB of effort. This is a processing burden likely to be well within the capabilities of even moderately capable

machines for many applications. Conversely, if the attacker is transmitting a hundred times as many marks as the originator, then they are expending 20 dB of relative energy. Even if they can afford the energy and the equipment burden to operate at this level, such transmissions are highly likely to attract unwanted attention.

D. Concurrent Codecs in the Context of the OSI Network Model

While not of strong relevance to the work here, some may find it useful to envision where a concurrent codec fits within the traditional framework of the OSI seven-layer network model. In theory, the jam-resistant behavior afforded by a concurrent codec can be implemented via the interaction of elements operating at several layers and, as with most network implementations, the actual partitioning of tasks is not rigidly defined and so some implementations will allocate some tasks to different layers than other implementations. Accordingly, the discussion here presents merely one possible partitioning.

Ideally, all elements of the codec would be implemented in the physical and data transport layers. The physical layer, being responsible for the actual transmission and reception of the data packets, dictates the channel error properties and how asymmetric they are. Therefore it plays a major role in the physical realization of the type of jam-resistant channels under consideration and cannot be ignored. The data link layer is responsible for ensuring data integrity between nodes (as opposed to end-to-end data integrity) and since our purpose is to permit continued communications between a sender and a receiver across a single wireless link, we are intrinsically discussing a node-to-node protocol. Therefore it is most appropriate that the bulk of the codec, the portion that extracts the list of legitimate codewords and/or their corresponding messages from the packet, be implemented in this layer.

However, an integral part of the overall protocol is the discrimination of false messages. While this should ideally be done at the data-link layer, it might be more convenient to pass the entire extracted message list up to a higher layer. That determination will primarily be driven by how false messages are discriminated against.

E. The Physical Layer Requirements

As already noted, the strategy employed here to obtain jam resistance is fundamentally tied to the error probabilities of the physical communications channel used. The goal is to achieve a "multiple access OR"[5] wherein the received message is simply the bitwise-OR of all transmitted messages plus channel noise. In the ideal case the mark error probability (the probability of receiving a space when a mark was transmitted) will be zero, meaning that all transmitted marks will be received unaltered. Most of our development will

[5]This term appears to be the most widely used ([67] for instance), but others such as multiaccess OR[63], or just simply an *OR channel*[68] are common as well.

be based on the assumption of an ideal OR-channel, meaning only that the actual mark-error probability is sufficiently low as to be unlikely to affect a given packet. However, methods of tolerating higher mark-error probabilities will be explored as well.

To date, OR-channels have generally been considered only in terms of data storage structures in which codewords are combined via a bitwise-OR. These are the classic superimposed codes (by all their various names) explored in great depth in the literature. However, little attention has been given to physical channels that exhibit this behavior. This is not to say that such channels do not exist, but merely that this aspect of their behavior has apparently never before merited close examination. In the wired world one popular communications protocol used in many micro-controller based systems is I^2C (Inter-Integrated Circuit) whereby players are only allowed to assert marks; spaces only appear when no player is asserting a mark. This is generally implemented by using components having an open collector/drain output in a "wired-OR" arrangement. However, there is no effort to exploit this mark/space asymmetry to achieve jam resistance - in fact the protocol, like the majority of protocols, assumes that all players will play nice and obey the rules. In I^2C, the purpose of using a wired-OR connection is to attain collision detection and bus arbitration and the rules state that any player attempting to transmit a space but who senses a mark instead must assume that someone else is transmitting and immediately relinquish the bus. Such a protocol is, of course, trivially jammed by a malicious player with very little effort.

In the wireless world, on-off keying (OOK) has been in use since the beginning of radio. In its simplest form, the transmitter broadcasts wideband noise (such as from a spark-gap generator in the early days) to transmit a mark and silence to transmit a space. The receiver then simply looks for the presence of noise above a certain threshold and calls that a mark while noise below the threshold is called a space. Provided the threshold is not changed as a result of the jammer's actions, the jammer can only convert spaces to marks. In practice however, just using OOK does not yield a viable OR-channel from the jam-resistance point of view. This is because a jammer normally only has to corrupt a relatively small number of spaces to interrupt the signal. To compensate, the receiver increases their discrimination threshold in the presence of even mild jamming and it quickly rises to the point that mark-errors as well as space-errors are received. Like I^2C, jammers can create this situation with very little relative power.

The underlying reason why the OOK receiver must raise its discrimination threshold is because present codecs rely on both the marks and the spaces in the received packet. This is largely forced on them by the desire/need to work with physical channels having symmetric bit-error probabilities and it's consequences carry over to channels that happen to be asymmetric. However, if (1) the channel is highly asymmetric, (2) the receiver does not rely on the presence of any (particular) spaces in the codeword, and (3) they can

tolerate a large number of additional marks (space-errors), then the situation changes fundamentally. Under these conditions, the receiver can leave their discriminator threshold alone and accept the false marks as part of the received packet.

Suitable OOK transmission protocols are possible with each of the three major forms of traditional spread spectrum - direct sequence, frequency hopping, and time hopping. The easiest one to visualize is time hopping spread spectrum such as used in impulse-based ultra-wideband radio[69], [70], [71], [72], [73], [74], [75], [76]. Using this approach, very short but high power pulses are produced at times, relative to the start of the packet, dictated by the codec's encoding algorithm. Pulse durations of fractions of a picosecond are achievable with low cost pulse generators[77].

F. The Inadequacy of Conventional Coding Theory

At first glance, it is tempting to consider the use of conventional, or even specially developed, error correcting codes to impart the necessary jam-resistance to an insecure channel. Certainly conventional coding theory via the use of error correcting codes to mitigate non-malicious channel noise is a well established and developed field that has been extensively applied to mitigating the effects of jamming in channels where the security of the key remains intact. This success can lead to the misperception that error correcting codes can be used to combat jamming in general. A subtle point, however, is that an intact key robs the attacker of the information needed to create a devastating low-power attack signal; in essence, the key turns any attack signal into essentially random noise and error correcting codes are very good at dealing with random noise.

However, error correcting codes cannot function well in the presence of noise that has been carefully crafted by a hostile party who possesses all of the information that the genuine sender does (i.e., has obtained the key). To see that this is the case, one need only consider the basic goal of an error correcting code and its subsequent limitations. That goal is to recover the originally transmitted codeword from a data stream that has undergone some form of corruption. This, in turn, is done by examining the received packet and determining which codeword is the most likely one to have been transmitted. But if the attacker simply transmits a second perfectly valid codeword, the receiver will be left with a packet that is some combination of the two (regardless of what type of communications channel is used). Even if the receiving codec could determine that two codewords were combined (something that would be highly unlikely), it would still have to identify which two and, even then, would be unable to determine which codeword was the genuine one and which was the rogue.

The bottom line is that error correcting codes are designed to recover a single codeword corrupted by essentially random noise, but an attacker that corrupts the original codeword by transmitting additional

legitimate codewords on top of it is employing noise that is anything but random.

G. The Performance of a Straight-through Codec

A straight-through concurrent codec performs no encoding - the transmitted codeword and packet are identical to the original message. With no encoding, even in the absence of any noise (malicious or otherwise), the decoder will generate a list of 2^d potential messages where d is the degree of the packet. Thus codewords of degree one or greater are intrinsically *autohallucinogenic* (i.e., they cover other codewords even without corruption).

Even worse, every additional mark inserted by an attacker is highly hallucinogenic since it further doubles the length of the message list. Hence an attacker can expend far less energy than the sender while forcing the receiver to do exponential work. The performance characteristic for this codec is given by:

$$\zeta_R = 2^{(\mu_S(1-\mu_S))(m\zeta_A)} \ (\zeta_R \leq 2^m) \tag{3}$$

To emphasize how inadequate this codec is, for a performance criterion of 20 dB of attack effort and 40 dB of receiver effort, a message with 50% marks would have to be shorter than half a bit in length!.

Another point to keep in mind is that our definition of receiver effort specifically excludes the baseline effort required in the presence of no jamming signal and that effort is also exponential in terms of the degree of the legitimate message. It's not too much of an exaggeration to proclaim the genuine sender as the receiver's worst enemy.

H. The Performance of a One-Hot Codec

The obvious flaw of the straight-through concurrent codec is that legitimate codewords are autohallucinogenic. To overcome this, we need to construct a codebook in which no codeword is a subset of any other codeword - or, at least, that this is a vanishingly rare occurrence.

Ideally, our codewords would be completely orthogonal to each other such that a given mark can never be used as part of two distinct codewords. If this is the case, then an attacker must exert as much energy as the sender to inject a rogue message and the length of the list grows only in direct proportion to the amount of energy expended - it is impossible to generate hallucinations. Thus the attacker cannot force the receiver to do a disproportionate amount of work. However, this also establishes a best-case performance bound since the receiver can always force the receiver to perform work in direct proportion to the amount of energy expended.

Such a codebook is actually trivial to construct. The simplest concurrent codec that exhibits this property is a one-hot concurrent codec in which an m-bit message is reduced to a single bit located in a unique

position within a 2^m-bit codeword. The originator can therefore send M_S messages concurrently by asserting the associated M_S marks in the packet. As before, we assume that the energy required is proportional to the number of marks which, in this case, is also proportional to the number of messages. The resulting performance characteristic is therefore:

$$\zeta_R = \frac{M_R}{M_S} = 1 + (1 - \frac{M_S}{2^m})\zeta_A \tag{4}$$

As expected, the attacker can only force the receiver to perform linear effort. It might be tempting to note that, since M_S will almost certainly be small compared to the number of bins in the packet, the attacker will have to expend nearly as much energy as the receiver. However, it must be remembered that our definitions of "energy" for the two are quite different, hence at the very least there is a proportionality constant involved.

A one-hot codec would appear to be an ideal solution to the jam-resistance problem. In fact, it marks the best possible performance of any conceivable concurrent codec in an OR-channel facing optimal hostile jamming. The problem is scalability.

To send a message consisting of m bits, the packet must contain 2^m bits. In a hypothetical system capable of transmitting packets at a rate of one gigabit per second, a sixty-four bit message would require over half a millennia to transmit. Hence the exponential scaling of the packet length makes this approach unworkable, at least in this most basic form.

I. The Performance of a Superimposed Codec

The one-hot concurrent codec attains the ultimate level of jam-resistance because every codeword is completely orthogonal to every other codeword. However, to achieve this the degree jam-resistance, both the sparseness of the codebook and the sparseness of each codeword is at the ultimate level of one mark per codeword. Recognizing this, plus that fact that we do not need the ultimate performance, only performance that is good enough, we can trade some jam-resistance for scalability improvements.

Consider a codec that maps m-bit messages onto c-bit codewords of degree d. If the expansion factor, e, (i.e, ratio of codeword length to message length) is even moderately high, then the codebook becomes extremely sparse. If, in addition, the codeword degree is relatively small then each codeword is also very sparse. The combination of these two effects can be used to make the codewords highly orthogonal to each other.

Two approaches to constructing the codebook are possible: (1) devise an algorithm specifically designed to produce optimally orthogonal codewords, or (2) use a pseudorandom process to generate codewords and determine if codec parameters (expansion and degree) can be found that make the codewords orthogonal

enough. The first, while attractive, may be difficult to achieve, especially since the packets must also be efficiently decodable. There is no guarantee that the second approach will work or that the resulting codewords will be short enough to be practical, but it leaves the avenue open for first developing an efficient means of encoding and decoding packets and then determining if the resulting codebook is sufficiently jam-resistant.

In developing the performance characteristic for this codec, we must consider the best way to construct attack packets. The attacker has three basic options: (1) they can transmit random marks, (2) they can construct packets by combining legitimate codewords, or (3) they can construct packets specifically designed to produce a disproportionately large number of hallucinations with a minimal number of marks. The first option is clearly the simplest to implement, but assuming that the attacker will do so should a better option exist is not realistic, especially since the entire motivation for developing concurrent codecs is because we have assumed that an attacker that obtains the key for a traditional jam-resistant system is willing to put forth the effort to exploit it. The second option at least permits the attacker to inflict a baseline level of harm on the receiver since they can force the receiver to recover and process the messages they've included. The third option is clearly preferred, but one of the goals of codebook design will obviously be to either make such packets nonexistent or make them extremely hard to find. Making them nonexistent is clearly achievable in the limit - a one-hot codec achieves this. Making ones that do exist extremely hard to find is achievable by using cryptographically sound one-way hash functions in the generation of the codewords. Thus, for now, we will assume that this option has been rendered infeasible.

Beginning with our working definitions of attacker and receiver effort and assuming (1) that an adequate pseudorandom function is used to generate the codewords and (2) that the attacker uses the second method to construct their attack packets, the performance characteristic is given by the following somewhat horrific equation:

$$\zeta_R = 1 + \frac{1}{M_S}\left\{\left[\zeta_A \frac{\ln(1-\beta)}{\ln\left(1-\frac{s}{e}\right)}\right] + \left[2^m - \left(M_S + \zeta_A \frac{\ln(1-\beta)}{\ln(\alpha 1)}\right)\right]\left[(\alpha)^{me}\right]^{\left\{1-\beta\left[1+\zeta_A \alpha^{M_S}\right]\right)}\right\} \tag{5}$$

where

$$\alpha = \left(1 - \frac{s}{e}\right)$$

and

$$\beta = \left(1 - \alpha^{M_S}\right)$$

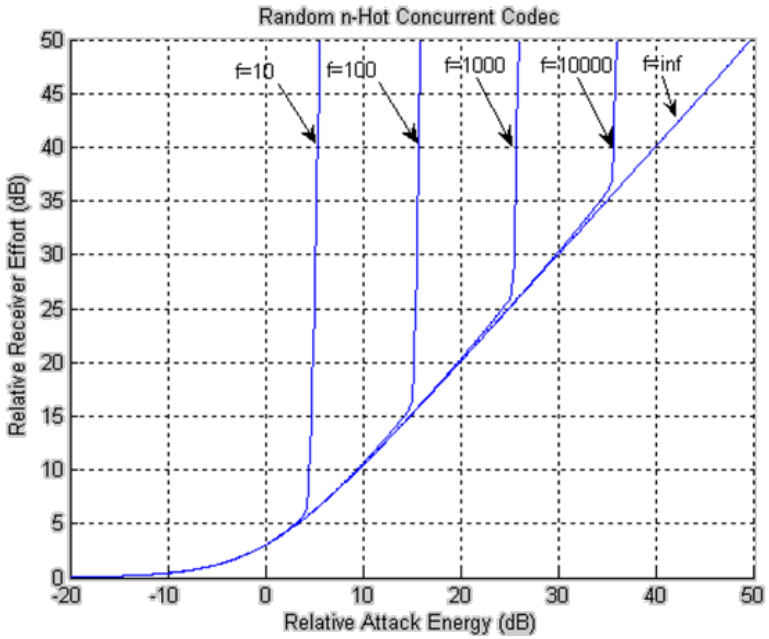

Fig. 1. Performance Characteristic of a superimposed concurrent codec.

The performance characteristic (Fig. 1) is very insensitive to the message length or the mark factor while the expansion factor sets an upper limit on the performance. The superimposed concurrent codec's characteristic for infinite codeword factor is identical to that of the one-hot concurrent codec, which was shown to represent the theoretical performance limit. For finite codeword factors, the attacker is constrained to the characteristic for the one-hot codec until they have committed an amount of energy that causes hallucinations to be produced in significant numbers. However, once this break-over energy level is reached, the hallucinations grow exponentially creating a near vertical wall effectively jamming the channel immediately.

J. The BBC Algorithm

Although the performance of a superimposed codec is acceptable, it's practicality is dependent upon efficient encoding of codewords and, more importantly, decoding of packets. As the literature search suggests, no such efficient algorithm has existed up to this time. This most likely owes more to the lack of a driving need for one rather than to its intrinsic difficulty. None-the-less, what is presented here is believed to be the only codec algorithm whose performance is of linear complexity in terms of both the message length and the number of messages (codewords) in a packet.

One approach to efficient decoding is to construct an algorithm that permits decoding arbitrary codewords in a fashion nearly identical to how packets are tested for specific codewords in most superimposed coding schemes. That approach simply looks to see if each mark in the codeword is present in the packet. As

soon as a needed mark is not found, the answer is known. On the other hand, if the search progresses to the end of the codeword without finding a missing mark, then the codeword is covered by the packet.

What has prevented efficient decoding to date is that codewords are constructed based on the entire message all at once. For instance, Bloom filters generate codewords by running the message through d different hash functions to produce the d necessary mark positions. However, if we construct codewords from the message one bit at a time, we can decode the packet the same way. More precisely, we must construct codewords in such a way that progressively larger subsets of codeword marks can be determined using progressively larger subsets of message bits. Implied in this description is that each subset must contain all prior subsets.

1) The Core Coding/Decoding Algorithm: The simplest and most obvious way to build a set of progressively larger subsets of the message to be encoded is to use progressively longer prefixes of that message. For example, if the 4-bit message 0110 is to be encoded, the series of prefixes that would be used would be {0, 01, 011, 0110}. Each prefix would be used to generate one or more mark positions. This is easily done using an appropriate hash function. To be appropriate, the hash function must be able to accept inputs of arbitrary length and must generate outputs that cover at least the range of the number of bits in the codeword. If the output spans a larger range, it must be mapped onto the range. In addition, the hash function should be reasonably random, even after any mapping of the range is performed.

Thus a codeword is generated using the following algorithm:

```
ENCODE(m, H, M) =
    for i = 1 to m
        Place a mark at position H(M[1:i])
```

Where $M[1:i]$ is the first i-bits of the message and $H(x)$ is a hash function that outputs a value from 1 to c (the number of bits in the codeword).

The decoding can be done using a recursive depth first search:

```
DECODE(m, H, M, i) =
    if (i EQ m)
        Add M to output message list
    else
        if there is a mark at H({M[i:1],'0'})
            DECODE(m, H, {M[1:i],'0'}, (i+1))
        if there is a mark at H({M[i:1],1})
            DECODE(m, H, {M[1:i],'1'}, (i+1))
```

Where the notation $\{M[1:i], \text{`0'}\}$ means to concatenate the string $M[1:i]$ and the string '0', in other words, add a 0 to the end of the partial message string.

The essence of the decoding algorithm is as follows: After decoding the partial message consisting of the first i bits of a message, determine where the encoder would have placed a mark if the next message bit were a zero. If a mark is found at that location, add a zero to the partial message and proceed to decode the next bit. Repeat this process for the case of the next message bit being a one.

The following examples show how the encoding algorithm works. The hash function for these examples is shown in Table IV.

```
Example 1
    Encode message: 0110

    Prefix #1:    0    H(0) = 12
        (Codeword = 00000000 00010000 00000000 00000000)
    Prefix #2:   01   H(01) = 15
        (Codeword = 00000000 00010010 00000000 00000000)
    Prefix #3:  011  H(011) = 20
        (Codeword = 00000000 00010010 00010000 00000000)
    Prefix #4: 0110 H(0110) = 25
        (Codeword = 00000000 00010010 00010000 10000000)

    Message: 0110 => Codeword: 00000000 00010010 00010000 10000000

Example 2
    Encode message: 1110

    Prefix #1:    1    H(1) = 10
        (Codeword = 00000000 01000000 00000000 00000000)
    Prefix #2:   11   H(11) = 14
        (Codeword = 00000000 01000100 00000000 00000000)
    Prefix #3:  111  H(110) =  1
        (Codeword = 10000000 01000100 00000000 00000000)
    Prefix #4: 1110 H(1110) = 21
        (Codeword = 10000000 01000100 00001000 00000000)

    Message: 1110 => Codeword: 10000000 01000100 00001000 00000000

Example 3
    Encode message: 0010

    Prefix #1:    0    H(0) = 12
        (Codeword = 00000000 00010000 00000000 00000000)
    Prefix #2:   00   H(00) = 19
        (Codeword = 00000000 00010000 00100000 00000000)
    Prefix #3:  001  H(001) = 23
        (Codeword = 00000000 00010000 00100010 00000000)
    Prefix #4: 0010 H(0010) = 19
        (Codeword = 00000000 00010000 00100010 00000000)

    Message: 0010 => Codeword: 00000000 00010000 00100010 00000000
```

Example #1 and #2 demonstrates that just because two messages are nearly identical does not indicate that their codewords will be related at all. Although these two messages differ only in the first bit, the resulting codewords are completely disjoint. However, two codewords that share the first n-bits will have at least the marks resulting from the first n prefixes in common. Fortunately, after the first prefix in which two messages differ the remaining codeword marks for those two messages are independent and unrelated (assuming a perfect hash function). None-the-less, this artifact is a potentially serious shortcoming of this encoding algorithm as it amounts to a significant reduction in the orthogonality of the codewords.

This loss of orthogonality is actually quite severe. Every codeword has at least one other codeword that differs from it by at most one mark. Plus, there is the very real possibility of having two codewords being identical. Put another way, the entire codebook can be partitioned into pairs of codewords that are separated by a Hamming distance of at most one. Furthermore, the codebook can be partitioned into groups of four codewords that are separated by a Hamming distance of at most two, groups of eight that are separated by a Hamming distance of of most three, and so forth.

Example #3 shows a different shortcoming, namely that it is not guaranteed that all of the marks generated for a given codeword will be disjoint. In this case, two different prefixes of the same message produced the same mark position resulting in a codeword of degree three instead of the expected degree four. Even worse, since the redundant mark position also happens to be the last mark position, the codeword for message 0011 will also contain all of the marks for message 0010 and hence be autohallucinogenic.

While both of these shortcomings decrease the level of jam resistance, they do so in favor of permitting efficient encoding/decoding without which we could not obtain jam resistance at all. The key will be in determining if the surviving level of jam resistance is likely to be good enough for the intended applications and, if not, whether modifications to the algorithm can be made to recover sufficient jam resistance to make it good enough.

Turning our attention to the decoding algorithm, the following examples also use the hash function from Table IV.

```
Example 4
    Codeword: 00000000 00010010 00010000 10000000

    H(0) = 12 ? Y
        H(00) = 19 ? N
        H(01) = 15 ? Y
            H(010) = 10 ? N
            H(011) = 20 ? Y
                H(0110) = 25 ? Y (Message: 0110)
                H(0111) =  2 ? N
```

TABLE IV
EXAMPLE HASH TABLE

#	w	$H(w)$	x	$H(x)$	y	$H(y)$	z	$H(z)$	#
0	0	12	00	19	000	6	0000	6	0
1							0001	15	1
2					001	23	0010	19	2
3							0011	14	3
4			01	15	010	10	0100	7	4
5							0101	5	5
6					011	20	0110	25	6
7							0111	2	7
8	1	10	10	28	100	11	1000	13	8
9							1001	3	9
10					101	8	1010	30	10
11							1011	29	11
12			11	14	110	28	1100	12	12
13							1101	17	13
14					111	1	1110	21	14
15							1111	9	15

```
    H(1) = 10 ? N

    Message list: {0110}

Example 5
    Codeword: 00000000 01111100 00100010 00010000

    H(0) = 12 ? Y
        H(00) = 19 ? N
            H(000) =  6 ? N
            H(001) = 23 ? Y
                H(0010) = 19 ? Y (Message: 0010)
                H(0011) = 14 ? Y (Message: 0011)
        H(01) = 15 ? N
    H(1) = 10 ? Y
        H(10) = 28 ? Y
            H(100) = 11 ? Y
                H(1000) = 13 ? Y (Message: 1000)
                H(1001) =  3 ? N
            H(101) =  8 ? N
        H(11) = 14 ? Y
            H(110) = 28 ? Y
                H(1100) = 12 ? Y (Message: 1100)
                H(1101) = 17 ? N
            H(111) =  1 ? N

    Message list: {0010, 0011, 1000, 1100}
```

Example #4 shows the basic decoding tree for a single codeword with no complications. Example #5, on the other hand, shows the decoding tree for a packet that was generated by combining the codewords

for messages 1000 and 0011. Since it has already been established that 0011 is autohallucinogenic, it is not surprising that 0010 also appears in the final message list. In addition, the marks in the two codewords combine to cover yet a fourth message, 1100.

To a large degree, the problems shown by these examples disappear for messages and codewords of practical interest, namely messages that are hundreds of bits long and codewords that are tens of thousands of bits long. One problem that does not substantially disappear is the problem of small Hamming distances for lots of small groups of codewords, however this is quite easily addressed by the addition of checksum bits to the message.

2) The Use of Checksum Bits to Enhance Orthogonality: The loss of orthogonality for messages that differ in only the last bit is a simple consequence of differing in only the last bit. As prior examples demonstrate, two messages can differ by only one bit and still be completely orthogonal provided that bit is the first bit. In general, the more message bits that remain after the first prefix where two messages first differ, the greater the Hamming distance will (probably) be between the two eventual codewords. Therefore, we need to append additional bits that result in additional codeword marks such that the locations of those marks are a function of the entire message. In traditional coding, one way of doing this would be to compute and append a set of checksum bits to the message. In the BBC algorithm it is actually sufficient to simply append zero bits to the message because the hash value that results will automatically reflect the entire message as well as each additional bit in turn.

TABLE V
EXAMPLE HASH TABLE W/CHECKSUM BITS

#	w	$H(w)$	x	$H(x)$	y	$H(y)$	z	$H(z)$	k_1	$H(k1)$	k_2	$H(k2)$	#
0	0	12	00	19	000	6	0000	6	00000	2	000000	3	0
1							0001	15	00010	32	000100	31	1
2					001	23	0010	19	00100	8	001000	9	2
3							0011	14	00110	17	001100	30	3
4			01	15	010	10	0100	7	01000	30	010000	20	4
5							0101	5	01010	7	010100	26	5
6					011	20	0110	25	01100	3	011000	27	6
7							0111	2	01110	26	011100	22	7
8	1	10	10	28	100	11	1000	13	10000	26	100000	29 8	
9							1001	3	10010	16	100100	16	9
10					101	8	1010	30	10100	3	101000	19	10
11							1011	29	10110	11	101100	10	11
12			11	14	110	28	1100	12	11000	1	110000	30	12
13							1101	17	11010	4	110100	2	13
14					111	1	1110	21	11100	8	111000	27	14
15							1111	9	11110	31	111100	13	15

The hash table in Table V includes two checksum bits for each message. The encoding algorithm is essentially the same except that the message is first modified by appending k checksum bits and then $(m + k)$ passes are made through the mark generation algorithm:

```
ENCODE(m, k, H, M) =
    M = {M,0(k)}
    for i = 1 to m+k
        Place a mark at position H(M[1:i])

Alternatively:
ENCODE(m, k, H, M) =
    for i = 1 to m
        Place a mark at position H(M[1:i])
    for i = 1 to k
        Place a mark at position H({M,0(i)})
```

Where $0(x)$ represents a string of x zero bits.

The decoding algorithm is also nearly the same, except that it leverages the knowledge that, once all of the basic message bits have been recovered, only zeros are possible for legitimate messages:

```
DECODE(m, H, M, i) =
    if (i EQ m+k)
        Add M to output message list
    else
        if (i < m)
            if there is a mark at H({M[i:1],0})
                DECODE(m, H, {M[1:i],0}, (i+1))
            if there is a mark at H({M[i:1],1})
                DECODE(m, H, {M[1:i],1}, (i+1))
        else
            if there is a mark at H({M,0(i-m)})
                DECODE(m, H, {M,0(i-m)}, (i+1))
```

Repeating Example #5, in which the messages 1000 and 0011 were combined, we end up with the following decode tree:

```
Example 6
    Codeword: 00000000 01111100 10100010 01011100

    H(0) = 12 ? Y
        H(00) = 19 ? N
            H(000) =  6 ? N
            H(001) = 23 ? Y
                H(0010) = 19 ? Y
                    H(00100) = 8 ? N
                H(0011) = 14 ? Y
                    H(00110) = 17 ? Y
```

```
                    H(001100) = 30 ? Y (Message: 0011)
    H(01) = 15 ? N
H(1) = 10 ? Y
    H(10) = 28 ? Y
        H(100) = 11 ? Y
            H(1000) = 13 ? Y
                H(10000) = 26 ? Y
                    H(10000) = 30 ? Y (Message: 1000)
            H(1001) =  3 ? N
        H(101) =  8 ? N
    H(11) = 14 ? Y
        H(110) = 28 ? Y
            H(1100) = 12 ? Y
                H(11000) = 1 ? N
            H(1101) = 17 ? Y
                H(11010) = 4 ? N
        H(111) =  1 ? N

Message list: {0011, 1000}
```

Notice that the addition of the checksum bits eliminated both hallucinations, including the autohallucination associated with 0011. Further note, however, that there are costs associated with the checksum bits beyond the additional complexity and necessary computation. For instance, the checksum bit placed at location 17 as a result of encoding 0011 forced the decoder to continue working potential message 1101 beyond the point where it originally expired and, in fact, required the action of its own checksum bits to prevent it from becoming an hallucination.

3) Suppression of Terminal Hallucinations: As already shown, the addition of checksum bits increases the orthogonality of the codebook and hence eliminates hallucinations that have otherwise survived the decoding process. Such hallucinations are called "*terminal hallucinations*". In designing a BBC-based concurrent codec, one parameter that must be chosen is the number of checksum bits to add. To make this determination, it is necessary to understand quantitatively how the number of checksum bits translates into protection from terminal hallucinations.

Assume we are working with a packet having a mark density μ_P. If those marks are randomly distributed (or if the mark location we test is randomly distributed), then the probability that we will find a mark when we test for one at a particular location is equal to the mark density. A terminal hallucination must survive k such tests, all of which are independent (given a suitable hash function), in order to make it to the final message list. As a result, the number of terminal hallucinations that are expected to survive is given by:

$$M_H = (M_{HT}) \left(\mu_P^k\right) \tag{6}$$

Where M_{HT} is the number of terminal hallucinations.

It is probably not excessive to consider adding a number of checksum bits that is on the order of 10% of the message length, so for messages that are, say, four hundred bits in length, forty checksum bits would be unlikely to result in an unacceptable increase in processing overhead. At a packet density of 50%, those forty checksum bits would translate into fewer than one in a trillion terminal hallucinations being added to the message list. If packet densities are not expected to exceed 33% and it is deemed adequate to allow one terminal hallucination in a million through to the final message list, then as few as thirteen checksum bits will suffice. It should be noted that the number of checksum bits needed for a given level of protection is dependent only on the packet density; in particular, it is independent of the message length or the expansion factor.

As anecdotal evidence of the checksum bit effectiveness, a packet was generated with four legitimate messages as sufficient random noise to raise the packet density to approximately 53%. As expected, the decoding process was reduced to a crawl, requiring over twelve hours to complete. The program used generates an ASCII log file permitting the reconstruction of the decoding tree by printing out the message fragment each time a mark is found at a test location. The resulting file for this run was over 27 GB in size indicating approximately 250 million nodes in the decoding tree (in comparison, at 33% density there were approximately 2000 nodes). The theory predicts that approximately four million terminal hallucinations would result and this is in rough agreement with the number of hallucinations implied by the log file length. To eliminate this many hallucinations would require twenty-five checksum bits, so while it was reassuring, it was not surprising that the forty checksum bits actually used were adequate to eliminate all of them leaving only the four genuine messages in the final list.

4) Working Hallucinations and Critical Density: While knowing what fraction of terminal hallucinations are expected to survive to the final message list is clearly important, it only has meaning if it is known how many terminal hallucinations are expected to exist in the first place. To answer this question, we can examine the behavior of the partial messages at an arbitrarily chosen point in the decoding tree. Those partial messages can be categorized as either belonging to a legitimate message which is expected to survive the remainder of the decoding process and those that are "*working hallucinations*" meaning that if they survive it will only be due to coincidental marks from other sources.

Assuming that the system achieves steady state, the number of partial messages at one level of the decoding tree will be (statistically) equal to the number of partial messages at the next.

$$M_{HW} = (M_S + M_A)\left(\frac{\mu_P}{1 - 2\mu_P}\right) \tag{7}$$

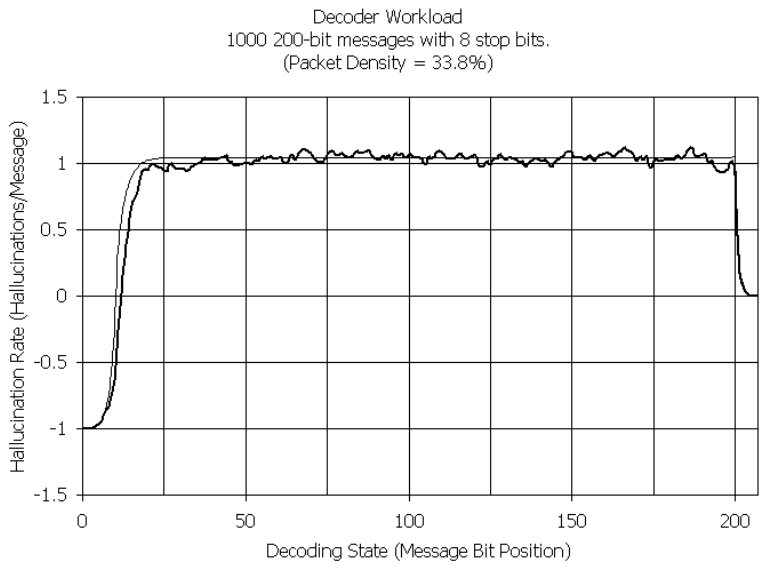

Fig. 2. Correlation between theoretical predictions and actual results. Note that in the initial stages of decoding, the highly limited number of possible codeword prefixes results in a mathematical artifact that manifests itself as "negative" hallucinations.

Where M_{HW} is the number of working hallucinations in steady state. Examination of the above relation shows that a packet mark density of 50% is expected to result in an infinite number of working hallucinations. In fact, 50% is the "*critical density*" below which a steady state hallucination level exists and above which the number of hallucinations grows exponentially without bound the further that processing continues.

Note that the number of working hallucinations is directly proportional to the number of legitimate messages in the packet, be they from the genuine sender or the attacker. In essence, each legitimate message is acting as a seed for the generation of new hallucinations while existing hallucinations tend to expire exponentially just as terminal hallucinations do.

Surprisingly, as Fig 2 demonstrates, the number of working hallucinations is quite tame until very near the critical density. For a density of 33% there is but a single working hallucination for each legitimate message (genuine and rogue) and even at a density of 47.6% there are only ten working hallucinations per legitimate message. As a result, we can reasonably expect our codec to continue functioning until very close to the critical density and to fail very quickly at or above it.

5) Interior Checksum Bits and Extended Critical Density: It is reasonable to ask if it is possible to increase the critical density by placing checksum bits within the message body in addition to placing them at the end. For packet densities below the normal critical density of 50%, such interior checksum bits have virtually no utility because even if they reduce the working hallucinations to zero at that point, the working hallucinations will rise back to the expected steady state level quite quickly. In fact, the addition

of these bits is detrimental because they only serve to increase the packet density, albeit marginally, and hence the number of working hallucinations will only tend to increase.

But the situation becomes very different if packet densities are expected to exceed 50%. Here we can utilize interior checksum bits to hold the exponential growth to acceptable limits. To do this, we decode a fragment of message, during which the number of hallucinations grows exponentially, and then we decode a string of checksum bits that clamp the hallucinations back down to a tolerable level. The resulting expected number of working hallucinations at the end of the message (and at the peak levels during message decoding) is then given by:

$$M_{HW} = (M_S + M_A)\left(\frac{\mu_P}{1-2\mu_P}\right)\left(\frac{\left[1-(2\mu_P)^{\beta}\right]}{\left[1-\mu_P^{\alpha}(2\mu_P)^{\beta}\right]}\right) \tag{8}$$

Where α zero bits are prepended to each fragment of β message bits. The factor $(1-2\mu_P)$ in the denominator that previously dictated the critical density now factors into the numerator and hence is a removable singularity. Therefore the remaining factor in the denominator determines the "*extended critical density*", given by:

$$\mu_{crit} = \frac{1}{2^{\frac{\beta}{\alpha+\beta}}} \tag{9}$$

If one checksum bit is added prior to each message bit, then the extended critical density is 70.7% while if ten checksum bits are inserted prior to each message bit the extended critical density increases to 93.9%. In theory the extended critical density can be made arbitrarily close to unity, however at some point the fact that the numbers involved, such as number of hallucinations and the number of remaining spaces in the packet, are intrinsically discrete quantities comes in to play. However, simulations have demonstrated that efficient decoding can remain possible even with densities driven to 99.5% by prepending 128 checksum bits to each message bit and appending 500 checksum bits to the end. Clearly, however, an extreme penalty must be paid to achieve this level of noise immunity.

It should be noted that being able to deal with packet densities above 50% has only limited utility in the case of an anti-jamming application. If the attacker is capable and willing to force the receiver's packet density to 50%, it is highly likely that they are capable and willing to force it another factor of two higher and completely jam the communications channel. Therefore the ability to extend the critical density is probably only of use when dealing with non-intelligent jamming sources (including environmental noise) and if that is the situation, then traditional jam-resistant approaches once again become a consideration, provided the necessary keys have been previously distributed or are publicly known. Hence about the only

application where extended critical densities might be useful, outside of academic circles, would involve initial exchange of keys in high noise environments without intelligent jamming.

K. The Visualization Demonstration

To better communicate the impact and degree of jam resistance afforded by concurrent codes a visual demonstration was developed that represents packet data as a Windows bitmap (BMP) file. The image is created by walking along the scan lines (which, for BMP files, begin in the lower left corner) and encoding each pixel white for a space and black for a mark. The program itself is ANSI-C compliant and, hence, cannot render the BMP file directly. Instead, once the BMP file is generated it may be opened in any of the readily available image viewers/editors (such as Windows Paint). In addition to viewing the packet data to gain an appreciation for how sparse and random the packet really is, the User can also modify the image to add additional marks. Once finished, they can use the program to retrieve the packet from the BMP file, decode it, and display any messages found.

The demonstration has proven very effective in conveying the concept and power of concurrent codes to numerous audiences consisting of people from both technical and non-technical backgrounds. In addition, a decoding log capability was added that permits graphing of the number of calls to the hash function made at each step of the decoding process. This acts as a good measure of the amount of effort performed by the receiver. Comparing this to the amount of effort predicted by the theory shows them to be in remarkably close agreement, as already shown in Fig. 2.

To further improve the utility of the demonstration, Dr. Dennis Schweitzer[6] developed a Java-applet that permits the User to perform all of these tasks using a single GUI. In addition, Dr. Schweitzer added a graphical representation of the decoding tree that clearly reveals the amount of additional work performed by the receiver as a function of the amount of corruption added by the attacker. The addition of this visualization element resulted in the somewhat startling realization of just how little additional work is forced on the receiver until the attack has become quite severe. While this is fully predicted by the theory, the degree and impact of it was not obvious until the tree could be visualized in realtime.

L. The Sonification Demonstration

While highly effective from many perspectives, the visualization demonstration is lacking in that it is not a tight analogy to the type of omnidirectional RF system envisioned as the eventual embodiment. To more closely model such a system in a way that still makes for a highly effective and portable demonstration, a sonification demonstration was also developed. The initial programs were developed jointly by Dr. Martin

[6]Director, Academy Center for Information Security (ACIS), United States Air Force Academy, Colorado Springs, CO.

Carlisle[7] and Capt. Sean Butler.[8] This demo consists of two programs, a sender and a receiver. The sender takes a message and converts it to codewords one character at a time (each character is a separate message in terms of the codec) and produces a Windows Wave (WAV) file for that character and sends the file to the sound card. The result is a series of clicks emanating from the speaker. The second program records the sound present at the computer's microphone until told to stop and then processes the recorded waveform to produce a packet, decodes the packet, and displays the recovered messages (characters) on the screen.

The "clicks" produced by the sending program consist of continuous sinusoidal tones of roughly 2.8kHz sine wave and two milliseconds in duration. The receiving program processes the waveform by applying a first order finite impulse response (FIR) wideband filter followed by a simple first order infinite impulse response (IIR) radiometer, followed by a Schmitt trigger comparator to produce a logic bitstream at the audio sampling rate (22.5kHz was used). This bitstream was then downsampled and converted to produce the packet for decoding.

The resulting demo allows quite effective realtime demonstrations by using two computers to send messages simultaneously while a third listens to the clearly overlapping series of apparently random, but identical, clicks. In order to permit the receiving computer to distinguish which computer a given message came from, the most significant bit (msb) in each ASCII character is set LO for one sender and HI for the other (mimicking a digital signature).

In addition to demonstrating the basic jam resistance properties, this demo also demonstrates that synchronization is not a problem (none of the three computers were synchronized in any fashion) and that oscillator differences were not overly critical (at least to the degree that the oscillators on different computers of different ages from different manufactures varied).

While this demo has been highly effective, it is not without significant limitations. It works well in moderately quite environments (it does not require highly silent surroundings) but is easily overcome by background noise. Part of this is intrinsic to the use of sound cards which have a very finite dynamic range and also to the automatic gain control built in to the sound card drivers. None-the-less, more robust performance should be possible by using longer message lengths and also by incorporating interior checksum bits to combat the intrinsically higher noise floors.

M. Publications, and Conferences

The theory of concurrent codes has potential applications far beyond jam-resistant communications. Some of those areas include cryptography, information retrieval, group testing, and MAC-less networks for such areas as RFID applications. In addition, the "pure theory" aspects are suitable for publication in

[7]Professor of Computer Science, United States Air Force Academy, Colorado Springs, CO.
[8]Instructor of Computer Science, United States Air Force Academy, Colorado Springs, CO.

several areas of mathematics, in particular any area in which superimposed codes have gained an audience. Because of the high diversity of topic areas, the potential for obtaining peer review from many different perspectives is very rich, but the ability to tap into that potential is limited to the rate at which appropriate venues, primarily conferences, become available and the ability of the people involved in the project to generate quality papers in each area. With a few exceptions, the initial focus is on gaining exposure to and feedback from the military communications community. Much of this effort has taken the form of briefing numerous senior officials in the Department of Defense and other government agencies in order to raise their awareness of this growing problem as well as offer a potential solution to at least part of it.

In addition to these "behind-the-scenes" efforts, it is also considered important that this theory and its potential applications be disseminated to the broader scientific and technical communities both in order to spread the knowledge but also to obtain peer review to validate or, if appropriate, refute the results obtained thus far. To that end, papers and abstracts presently exist in every stage of the publications pipeline: already published, accepted or submitted for publication, actively in process, or in the planning stages. The following describes most of those efforts and their status.

1) Published or Accepted Papers: The first paper that presents concurrent codes and the BBC algorithms is concerned with a topic area separate from jam-resistance communications but, rather, in information retrieval. This paper was prepared by Dr. Donald H. Kraft[9] and Lt Col Leemon C. Baird[10] and will be presented at a conference in Mexico in the summer of 2007. For the record, this author is not affiliated with that paper.

The sole publication that has actually been printed and distributed to date is the Air Force Academy tech report[78] on the topic published on 14 February 2007. This report also served as the initial Report to Sponsor from the Academy Center for Information Security (ACIS) to the Air Force Information Operations Center (AFIOC).

2) Publications In Progress: Since publication of the USAFA tech report several papers and/or abstracts have been submitted to conferences, each focussing on a different area of the subset. Two papers have been submitted to the Information Assurance Workshop held annually at West Point.[11] One[79] introduces the general topic, focussing on the general problem of jam-resistance in large scale ad hoc wireless networks and the use of BBC-based codecs as a potential solution while the other[80] focusses on the use of data visualization to present and understand the underlying concepts.

The abstract for another paper focussing principally on the threat to military communications at the edge

[9]Professor, Department of Computer Science, Louisiana State University, Baton Rouge, LA and the 2005 - 2006 Visiting Distinguished Professor, Department of Computer Science, United States Air Force Academy, Colorado Springs, Colorado.
[10]Associate Professor, Department of Computer Science, United States Air Force Academy, Colorado Springs, Colorado.
[11]8th Annual IEEE SMC Information Assurance Workshop, 20-22 June 2007, West Point, NY.

of the Global Information Grid and the potential for concurrent codes to help mitigate them[81] has been submitted to MilCom2007.[12]

Finally, the abstract of a paper focussing on the potential use of concurrent codes in undergraduate programming and signal processing education[82] has been submitted to the annual RMC/CSCC conference.[13]

3) Papers in Preparation: In addition to preparing the actual papers in cases where only abstracts have been so far submitted, plans to write a paper focussing on how to properly measure the performance of traditional spread spectrum systems and concurrent codec based systems so that comparisons are relevant and meaningful are well underway. This paper involves the participation of faculty from the Electrical Engineering departments both at the Air Force Academy and at the University of Colorado, Colorado Springs.

Another paper that is in the initial preparation stages involves a stochastic security proof addressing the question of how secure the BBC algorithm is against attacks based on finding highly hallucinogenic attack packets.

4) Future Papers Planned: Once the papers described above are finished, the plans are to prepare additional papers for submission to conferences and journals in the fields of information retrieval, group testing, and RFID applications.

V. Research Plan

Ideally, this Ph.D. proposal and the corresponding comprehensive examination would have taken place significantly earlier in the author's work. Unfortunately, since May 2006 the pace of the work has been driven by the expectations of the author's employer (the Academy Center for Information Security) and the sponsoring agency (the Air Force Information Operations Center) while the comprehensive examination could not be taken prior to the present semester due to Ph.D. program rules. As a result, the amount of remaining work that is planned for the Ph.D. is limited and consists of the following:

- Implement and demonstrate the BBC algorithm in a Software Defined Radio.
- Modify the BBC decoding algorithm to incorporate a limited tolerance for mark errors.
- Perform bit error rate (BER) calculations for spread spectrum systems used in an On/Off Keying mode appropriate to the implementation of appropriate multiple access OR channels.
- Devise a suitable measure of processing gain for a BBC-based system and compare it to traditional spread spectrum systems with and without compromised keys.

[12]2007 IEEE Military Communications Conference (MilCom2007), 29-31 October 2007, Orlando, Florida.

[13]Rocky Mountain Conference of the Consortium for Computing Sciences in Colleges (RMC/CSCC), 19-20 October 2007, Utah Valley State College, Orem, Utah.

Arguably, all of the above tasks are beyond the scope necessary to bring the research of a new coding theory to an acceptable level as they address practical implementation issues. They are included primarily because demonstrating that those issues do not render this work impractical to implement is of central importance to the program sponsor.

The time-line to complete the above work is keyed to making the final dissertation defense early in the 2007 Fall semester.

VI. Conclusion

This proposal has presented the relevant details of the author's involvement in a research effort of potentially far-reaching scope and consequence. The motivation for this work and its relevance to present and future communications in both the military and civilian communities has been presented. In addition, the relationship between this work and previous related work by others, such as could be identified, has been explored. Following that, the key results that have been obtained to date have been described and examined in some detail. Finally, efforts to obtain peer review of the work accomplished to date as well as a summary of the plans to complete the remaining work have been presented.

It is the author's belief (some might say fervent hope) that this proposal, in combination with the Comprehensive Exam, will provide convincing evidence to the author's advising committee that this avenue of research represents a significant and original contribution to the general body of knowledge that, when complete, will be commensurate with the awarding of a Ph.D. in Electrical Engineering.

APPENDIX I

GLOSSARY OF TERMS AND SYMBOLS

The following list and tables are short references of the terms and symbols used in this work. The following section provides more complete descriptions as well as the mathematical relationships between many of them.

sender	The party attempting to send a message to a receiver (a.k.a., genuine sender).
receiver	The intended recipient of a sender's message.
attacker	The party attempting to prevent the receiver from receiving the sender's message.
message	A binary bit string that is to be conveyed from sender to receiver.
codeword	The binary encoding of a message that is to be transmitted.
legitimate codeword	A binary bit string that can result from encoding a message.
codebook	The collection of all legitimate codewords.
multiple access OR channel	An environment were multiple codewords are combined via bitwise-OR.
packet	A collection of concurrent codewords, i.e., codewords that have been combined in a multiple access OR channel.
codec	In general, a device or algorithm that codes messages into codewords and decodes messages from packets.
concurrent codec	A codec specifically designed to work with packets that contain concurrent codewords.
mark	A binary symbol having (ideally) zero transmission error probability.
space	A binary symbol potentially having a significant transmission error probability.
hallucination	A message produced by decoding a packet that was not intentionally inserted into the packet.
covered codeword	A codeword all of whose marks are present in the packet being decoded.
terminal hallucination	A message that has survived the decoding process up to the point of decoding the checksum bits.
autohallucination	A codeword that, by itself, covers other codewords.
working hallucination	A message that exists at some intermediate point in the decoding process.
genuine message	A message that has been encoded and inserted into a packet by the genuine sender.
rogue message	A message that has been encoded and inserted into a packet by a hostile player (attacker).
legitimate message	A message that has been encoded and inserted into a packet, either by the genuine sender or by a hostile player (attacker) (i.e., not an hallucination).
density	The fraction of marks in a codeword or packet.
attack packet	The collection of marks inserted into a packet by an attacker.
critical density	The packet density at which decoding is expected to produce an exponential number of terminal hallucinations.
BBC	A concurrent coding algorithm based on sequential partial encoding/decoding.
checksum bits	In BBC, zero bits that are appended to the end of a message prior to encoding that serve much the same purposes as traditional checksum bits in other applications.
message fragment	In BBC, a portion of a message that has been partitioned for the purpose of inserting additional checksum bits (a.k.a., clamp bits) in the interior of a codeword.
clamp bits	In BBC, checksum (zero) bits that are inserted between message fragments

TABLE VI
GLOSSARY OF TERMS AND SYMBOLS

Codec Parameters		
m	=	message length (bits)
e	=	expansion factor (codeword bits per message bit)
c	=	codeword length (bits)
s	=	mark factor (nominal number of marks per codeword per message bit)
k	=	BBC checksum bits (number of zeros appended to a end of message)
α	=	BBC clamp bits (number of interior zero bits that prefix each fragment)
β	=	BBC fragment bits (number of message bits between groups of checksum bits)
Codeword and Packet measures		
d	=	degree of codeword/packet (number of marks in the codeword or packet)
μ_c, μ_p	=	mark density - (probability that a codeword/packet bit is a mark)
Performance-related Parameters		
M_S	=	Messages sent by the legitimate sender
M_A	=	Messages (or their equivalent) sent by the attacker
M_R	=	Messages output by the receiving codec
M_H	=	Messages that are hallucinations ($M_R - M_S or M_R - (M_S + M_A)$ as appropriate)
ζ_A, ζ_R	=	Relative effort of attacker/receiver

prior to encoding to extend the increase packet density.

II. EXPANDED EXPLANATIONS OF TERMS

This appendix provides fuller explanations of many of the terms used in this work and relates them to each other. While some of this is duplicated within the body of the text, it is believed that presenting them here in collected form will prove beneficial by minimizing the need to seek out the relevant descriptions within the text.

A. BBC Codec Parameters - m, c, e, s, k, α, β

A BBC-based codec, like many codecs, takes an *m*-bit messages and generates a *c*-bit codeword. As with error-correcting codecs, *c* is larger than *m* but, unlike error-correcting codes that use the additional bits to incorporate redundant information that can be used to detect errors and reconstruct the original codeword from a corrupted packet, in concurrent coding the additional bits are used to create sparseness in the codeword so as to accommodate the addition of undesired marks. None-the-less, a fair degree of common purpose, at least conceptually, exists and, like error-correcting codes, the greater the ratio between the codeword length and the message length, the greater the ability of the code to achieve its goal of recovering the correct message from a corrupted packet. This is such a key factor in the capability of the code that it is the dominant parameter in the description of the codec and is called the *expansion factor*, *e*.

$m \equiv$ message length (bits)

$$c \equiv \text{codeword length (bits)}$$

$$e = \text{expansion factor} \equiv \frac{\text{codeword length}}{\text{message length}} = \frac{c}{m}$$

The expansion factor is always based on the number of bits in the raw message and does not reflect any additional bits added to the message prior to encoding.

Another key codec parameter is how many marks appear in each codeword. This is defined as the *degree*, *d*, of the codeword (e.g., a codeword of degree 62 has 62 marks in it). Like the codeword length, this is best expressed in terms of the message length, hence the *mark factor*, *s*, is the ratio of the nominal number of marks in the codeword to the length of the original message.

$$s = \text{mark factor} \equiv \frac{\text{marks in codeword}}{\text{message length}} = \frac{d}{m}$$

In some contexts it is important to understand that a codeword may be may have a different *actual degree* than its *nominal degree*. This is particularly the case for stochastic generating algorithms such as BBC that may produce degenerate marks (i.e., multiple marks placed at the same location within the same codeword). Generally the context of the discussion is sufficient to determine which meaning is relevant.

The BBC algorithm incorporates the ability to reject hallucinations as part of the normal decoding process by appending checksum bits to the end of the message prior to encoding it. The checksum bits are simply zero bits and are not dependent on the contents of the message, however they place additional marks in the codeword at locations that are a function of the entire message, and hence serve the same purpose as checksum bits in more traditional contexts. The number of checksum bits, k, that are appended to the message determines what expected fraction of the terminal hallucination will survive to become hallucinations in the final message list for a particular packet density.

$$k = \text{number of zero bits appended to message to serve as checksum bits}$$

An extension of the basic BBC algorithm that permits the decoder to continue working with very dense packets is to partition the original message into a series of fragments and insert additional checksum bits to the beginning of each fragment. The parameter α is equal to the number of clamp bits that are prepended to each fragment prior to encoding. Like the checksum bits at the end of the message, the clamp bits are simply zero bits - a different name is used merely to distinguish the two, while $\beta$is equal to the number of message bits in each fragment (the last fragment may have fewer if the message length is not an integral multiple of the fragment length).

$\alpha =$ number of zero bits prepended to each message fragment

$\beta =$ number of message bits in each (full) message fragment

B. Packet Parameters - P, M, μ

A packet, *P*, is a collection of codewords (and/or packets) that have been combined by performing a bitwise logical-OR'ing of all the included codewords. Because of this, codewords and packets share many of the same descriptors, including message load, *M*, and mark density, μ.

The message load of a packet is the number of messages whose codewords have been combined to produce that packet. Depending on the context, it may include or exclude hallucinations.

The mark density is the fraction of bits in the codeword or packet that contain marks. In other words, it is the degree of the codeword or packet divided by the codeword or packet length.

$$\mu = \text{mark density} \equiv \frac{\text{marks in codeword}}{\text{codeword length}} = \frac{d}{c} = \frac{d}{me} \tag{10}$$

In the case of individual codewords where the degree is dictated by the mark factor, this can also be expressed, at least as an expectation value, as

$$\mu_c = \frac{d}{c} = \frac{ms}{me} = \frac{s}{e} \tag{11}$$

C. Sender, Receiver, and Attacker - S, R, A

Most discussions will require three actors: a sender, a receiver, and an attacker. We could use the archetypical actors Alice and Bob from the cryptographic community, adding in the less-well-known Mallory as our malicious, active attacker, but we have chosen not to. As such, parameters will frequently be subscripted with an *S*, an *R*, or an *A* to indicate that they apply to the sender, receiver, or attacker respectively.

D. Attacker and Receiver Effort - E, ζ

The end goal of much of the analysis in this work is to compare the amount of effort that the attacker can force the receiver to perform as a function of the amount of energy expended in the attack. As in most other treatments of communications jamming we will normally want to, at least eventually, normalize the receiver's effort to the amount of effort they would expend if there were no jamming to contend with.

Our definition of energy, *E*, is necessarily rather abstract so as to establish a well defined measure that can be applied consistently to the various concurrent codecs being analyzed, but the intent is that the relevant effects in practical applications should scale approximately in accordance.

The relative energy used by the attacker is defined as the ratio of the marks in the attack packet to the marks in the sent packet. In transmission schemes such as pulse-based ultra-wideband, this definition is particularly appropriate. In other schemes it may be less so.

$$\zeta_A \equiv \frac{E_A}{E_S} = \frac{d_A}{d_s} = \frac{\mu_A}{\mu_S} \tag{12}$$

The relative energy required by the receiver is defined as the ratio of the computational work needed to extract and process all of the messages (M_R) to the amount of computational work that would be required if no jamming were taking place($M_R^{'}$). In most cases, $M_R^{'}$ will be equal to the number of genuine messages sent (M_S). Our proxy for computational effort will be the number of messages that are contained in a packet, hence

$$\zeta_R \equiv \frac{M_R}{M_R^{'}} = \frac{M_R}{M_S} \tag{13}$$

Depending on the details of the codec, the relevant point at which to measure M_R may not be at the codec's output. In general, it will be at the point in the processing chain that dictates the capacity limits.

REFERENCES

[1] J. Foster and L. Welch. The evolving battlefield. *Physics Today*, 53(12):31, dec 2000.

[2] J. Gowens and Jr. K. Young. Fy2001 annual report of the communications and networks consortium. Army Research Laboratory Collaborative Technology Alliance Program.

[3] 1996 federal radionavigation plan. U.S. Gov't Printing Office, jul 1997.

[4] 2005 federal radionavigation plan. National Technical Information Service, Springfield, Virginia 22161, jul 2006.

[5] http://www.nsa.gov/ia/industry/gig.cfm?MenuID=10.3.2.2.

[6] Defense acquisitions - the global information grid and challenges facing its implementation. U.S. Government Accountability Office, 2004.

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

[8] Martin E. Hellman. An overview of public key cryptography. *IEEE Communications Magazine, 50th Anniversary Commemorative Issue*, pages 42–49, May 2002.

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

[10] Whitfield Diffie. The first ten years of public-key cryptography. *Proceedings of the IEEE*, 76(5):560–577, May 1988.

[11] B. Neuman and T. Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications*, 32(9):33–38, sep 1994.

[12] J. Kohl, B. Neuman, and T. Ts'o. *The Evolution of the Kerberos Authentication System*. IEEE Computer Society Press, 1994.

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

[14] M. Burmester and Yvo Desmedt. A secure and scalable group key exchange system. *Information Processing Letters,*, 94(3):137–143, 2005.

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

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

[17] A. Belouchrani and M. Amin. Jammer mitigation in spread spectrum communications using blind source separation. *Signal Processing*, 80(4):723–729, April 2000.

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

[19] 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.*, pages 679–683, 1998.

[20] 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.

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

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

[23] L. B. Milstein. Interference suppression to aid acquisition in direct-sequence spread-spectrum communications. *IEEE Transactions on Communications*, 36(11):1200–1207, November 1988.

[24] H. V. Poor and L. A. Rusch. Narrowband interference suppression in spread spectrum cdma. *IEEE Personal Communication Magazine*, 1:14–27, August 1994.

[25] 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, April 2002. To appear.

[26] 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)*, pages 103–107, October 1992.

[27] 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, pages 3426–3431, November 1998.

[28] 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.

[29] 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*, pages 303–307, May 2002.

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

[31] W. Kautz and R. Singleton. Nonrandom binary superimposed codes. *IEEE Transactions on Information Theory*, pages 363–377, 1964.

[32] D. Danev. Some constructions of superimposed codes in euclidean spaces. *Discrete Applied Mathematics*, 128(1):85–101, May 2003.

[33] G.T Ahlstrm and D Danev. A class of superimposed codes for cdma over fiber optic channels. Technical Report LiTH-ISY-R-2543, Linkoping University, 2003.

[34] Peter-Olof Anderson. Superimposed codes and bn-sequences. Technical Report LiTH-ISY-I-1108, Linkoping University, 1990.

[35] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, July 1970.

[36] Haoyu Song, Sarang Dharmapurikar, Jonathan Turner, and John Lockwood. Fast hash table lookup using extended bloom filter: an aid to network processing. In *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 181–192, New York, NY, USA, 2005. ACM Press.

[37] A. Clementi, A. Monti, and R. Silvestri. Distributed broadcast in radio networks of unknown topology. *Theor. Comput. Sci.*, 302(1-3):337–364, 2003.

[38] Z. Furedi Z. Furedi P. Erdos, P. Frankl. Families of finite sets in which in which no set is covered by the union of r others. *Israel Journal of Mathematics*, 51:75–89, 1985.

[39] M. Ruszinko. On the upper bound of the size of the r-cover-free families. In *Proceeding of the 1993 IEEE International Symposium on Information Theory*, page 367, 1993.
[40] Arkadii D'yachkov, Vladimir Lebedev, Paul Vilenkin, and Sergi Yekhanin. Cover-free families and superimposed codes: Constructions, bounds, and applications to cryptography and group testing. In *IEEE International Symposium on Information Theory*, 2001.
[41] D. R. Stinson and R. Wei. Generalized cover-free families. *Discrete Mathematics*, 279:463–477, 2004.
[42] R. Wei. On cover-free families. Technical report, Lakehead University, 2006.
[43] Arkadii D'yachkov, Anthony Macula, David Torney, Pavel Vilenkin, and Sergey Yekhanin. New results in the theory of superimposed codes. In *roceedings of International Conf. on Algebraic and Combinatorial Coding Theory (ACCT)*, pages 126–136, 2000.
[44] Sergy Yekhanin. Sufficient conditions of existence of fix-free codes. *IEEE International Symposium on Information Theory*, June 2001.
[45] A. Dyachkov and V. Rykov. Optimal superimposed codes and designs for renyi's search model. *Journal of Statistical Planning and Inference*, 100:281–302, 2002.
[46] A. de Bonis and U. Vaccaro. Constructions of generalized superimposed codes with applications to group testing and confict resolution in multiple access channels. *Theoretical Computer Science*, 306:223–243, 2003.
[47] A. D'yachkov, P. Vilenkin, and S. Yekhanin. Upper bound on the rate of superimposed (s,l) codes based on engel's inequality. In *Proceedings of the International Conf. on Algebraic and Combinatorial Coding Theory (ACCT)*, pages 95–99, 2002.
[48] S. Yekhanin. Some new constructions of optimal superimposed designs. In *Proceedings of International Conf. on Algebraic and Combinatorial Coding Theory*, pages 232–235, 1998.
[49] A. de Bonis and Ugo Vaccaro. Efficient constructions of generalized superimposed codes with applications to group testing and conflict resolution in multiple access channels. In *ESA*, pages 335–347, 2002.
[50] Thomas Erickson and Laszlo Gyorfi. Superimposed codes in rn. Technical Report LiTH-ISY-I-0818, Linkoping University, 1986.
[51] Tayuan Huang and Chih wen Weng. A note on decoding of superimposed codes. *Journal of Combinatorial Optimization*, 7:381–384, 2003.
[52] D. Awduche and A. Ganz. Mac protocol for wireless networks in tactical environments. In *IEEE Military Communications Conference, MILCOM-96*, October 1996.
[53] Lars-Inge Alfredsson. A class of superimposed codes for cdma over fiber optic channels. Technical Report LiTH-ISY-I-1045, Linkoping University, 1989.
[54] Fidel Cacheda and Ángel Viña. Superimposing codes representing hierarchical information in web directories. In *WIDM*, pages 54–60, 2001.
[55] Y. Desmedt. A high availability internetwork capable of accomodating compromised routers. *BT Technology Journal*, 24(4):77–83, July 2006.
[56] Janos Komlos and Albert Greenberg. An asymptotically nonadaptive algorithm for conflict resolution in mutliple-access channels. *IEEE Transactions on Information Theory*, IT-31(2):302–306, March 1985.
[57] http://www.nist.gov/dads/HTML/superimposedCode.html.
[58] D. Du and F. Hwang. *Combinatorial Group Testing and Its Applications*. World Scientific, 1993.
[59] R. Dorfman. The detection of defective members of large populations. *Annals of Mathematical Statistics*.
[60] C. Schultz. *Punched Cards, Their Applications to Science and Industry*, chapter 10 - An application of random codes for literature searching. Reinhold Publishing Corp., 1958.
[61] 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*, pages 296–306, 2003.
[62] P. de Laval and S. Abdu-Jabbar. Decoding of superimposed codes in multiaccess communication. In *Conference Proceedings on Area Communication, EUROCON 88*, pages 154–157, June 1988.
[63] Shakir Abdul-Jabbar and Peter de Laval. Constant weight codes for multiaccess channels without feedback. In *Conference Proceedings on Area Communication, EUROCON 88*, pages 150–153, June 1988.
[64] P de Laval. Decoding of superimposed codes. Technical Report LiTH-ISY-R-0958, Linkoping University, 1988.
[65] P. de Laval. Sliding window reduced search decoding for superimposed codes. Technical report, Linkoping University, 1989.
[66] P. de Laval. Superimposed code reservations systems- description and examples of possible implementation principles. Technical report, Linkoping University, 1989.
[67] Sandor Gyori. Signature codeing over multiple access or channel. In *Winter School on Coding and Information Theory*, 2003.
[68] C. Christian Jonsson. Anti-jamming on the or-channel. In *1994 IEEE International Symposium on Information Theory*, pages 480–480, Trondheim, Norway, July 1994.
[69] R. Blazquez, P. Newaskar, and A. Chandrakasan. Coarse acquisition for ultra wideband digital receivers. In *Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, volume IV, pages 137–140, Hong Kong, China, April 2003.
[70] Kevin J. Negus, John Waters, Jean Tourrilhes, Chris Romans, Jim Lansford, and Stephen Hui. Homerf and swap: Wireless networking for the connected home. *ACM Mobile Computing and Communications Review*, 2(4):28–37, October 1998.
[71] P. Newaskar, R. Blazquez, and A. Chandrakasan. A/d precision requirements for an ultra-wideband radio receiver. In *SIPS 2002*, pages 270–275, San Diego, CA, October 2002.
[72] M. Z. Win and R. A. Scholtz. Impulse radio: how it works. *IEEE Communications Letters*, 2(2):36–38, February 1998.
[73] R. Jean-Marc Cramer, Robert A. Scholtz, and Moe Z. Win. Evaluation of an ultra-wide-band propagation channel. *IEEE Transactions on Antennas and Propagation*, 50(5):561–570, May 2002.
[74] M. Z. Win and R. A. Scholtz. On the robustness of ultra-wide bandwidth signals in dense multipath environments. *IEEE Communications Letters*, 2(2):51–53, February 1998.
[75] M. Z. Win and R. A. Scholtz. On the energy capture of ultrawide bandwidth signals in dense multipath environments. *IEEE Communications Letters*, 2(9):245–247, September 1998.
[76] Moe Z. Win and Robert A. Scholtz. Ultra-wide bandwidth time-hopping spread-spectrum impulse radio for wireless multiple-access communications. *IEEE Transactions on Communications*, 48(4):679–691, April 2000.
[77] S. Yilmaz and I. Tekin. Ultra-wideband n-bit digitally tunable pulse generator. In *Ultra-Wideband, 2005. ICU 2005. 2005 IEEE International Conference on*, pages 438–441, sep 2005.

[78] L. Baird, W. Bahn, and M. Collins. Jam-resistant communication without shared secrets through the use of concurrent codes. Technical Report USAFA-TR-2007-01, United States Air Force Academy, 2007.
[79] L. Baird, W. Bahn, M. Collins, M. Carlisle, and S. Butler. Keyless jam resistance. In *8th Annual IEEE SMC Information Assurance Workshop*, page (paper submitted), jun 2007.
[80] D. Schweitzer, L. Baird, and W. Bahn. Visually understanding jam resistant communication. In *8th Annual IEEE SMC Information Assurance Workshop*, page (paper submitted), jun 2007.
[81] L. Baird, W. Bahn, and M. Collins. Jam resistant communications without shared secrets. In *2007 Military Communications Conference (MILCOM2007)*, page (abstract submitted), oct 2007.
[82] W. Bahn, L. Baird, M. Collins, M. Carlisle, and S. Butler. The use of concurrent codes in computer and digital signal processing education. In *Rocky Mountain Conference of the Consortium for Computing Sciences in Colleges (RMC/CSCC)*, page (abstract submitted), oct 2007.