CODING THEORY FOR SECURITY AND RELIABILITY IN WIRELESS NETWORKS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Anna Kacewicz

February 2010

CODING THEORY FOR SECURITY AND RELIABILITY IN WIRELESS NETWORKS

Anna Kacewicz, Ph.D.

Cornell University 2010

Wireless networks hold many applications and are an integral part of our lives. Security and reliability are extremely important in wireless networks. These networks must be reliable so that data can be conveyed from transmitters to receivers. Data sent across wireless networks must be kept confidential from unintended users and it is necessary that false packets generated by illegitimate users are rejected by the receiver. Another important task is for the network to determine which network components can be trusted and to what degree.

The work presented in this dissertation addresses the security and reliability issues in wireless networks through the use of coding theory. The network is composed of numerous nodes and we consider a classical point to point communication problem. We explore the network reliability issue and develop two algorithms (exponential and polynomial time) which determine minimum redundancy and optimal symbol allocation to assure that the probability of successful decoding is greater than or equal to a specified threshold. The performance of the algorithms is compared with each other, and MDS, LT, and Raptor codes are compared using the exponential algorithm. We also consider the security problem of keeping a message confidential from an illegitimate eavesdropper in a multiple path network. Carefully crafted Raptor codes are shown to asymptotically achieve perfect secrecy and zero-error probability, and a bit allocation method across the paths is developed. Lastly, we look into the problem of determining the integrity of nodes in the network. In particular, we show how

the malicious nodes can be localized in the network through the use of Reed-Muller codes. The Reed-Muller codes represent the paths that are necessary in the network. For the case where a path is not realizable according to the network connectivity matrix, we conceived an algorithm to treat the non-realizable paths as erasures and decode to localize malicious nodes. The performance of the algorithm is compared to several techniques.

BIOGRAPHICAL SKETCH

Anna Kacewicz received her B.S. in Electrical and Computer Engineering from the University of Texas at Austin in 2005. She joined Cornell University as a M.S/Ph.D student in Electrical and Computer Engineering in 2005. At Cornell she worked with Professor Stephen B. Wicker in his Wireless Intelligence Systems Laboratory (WISL) group. Her research interests include security, reliability, wireless networks, and coding and information theory.

This document is dedicated to my parents and my twin sister Ewa, who have always supported and had faith in me.

ACKNOWLEDGMENTS

I would like to thank my advisor Professor Stephen B. Wicker for accepting me into his group and guiding me to interesting research. His constant encouragement raised my confidence level and without that I would not have been able to finish.

I would also like to thank my committee members Professor Adam Bojanczyk and Professor Kevin Tang for their guidance and support.

Thanks to all the fellow/previous members of WISL and ECE graduate students for all the great times relaxing from research as well as discussing research. Without them, graduate school would have been much more frustrating and dull.

Finally I would like to thank my friends, my parents Marek and Malgorzata Kacewicz and my twin sister Ewa Kacewicz. For all the days of my life, my family has loyally stood by my side providing comfort, support and friendship. Also I would like to thank Robbie for his companionship.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In recent years, the demand and applications for wireless networks have proliferated. The nature of wireless networks causes transmitted data to be particularly susceptible to noise. The noise, which results from either signal degradation caused by obstacles or traversing long distances, or to an illegitimate user tampering with the data, yields corrupted data. Thus, it is vital that means are found to assure that the message is received reliably and securely at the decoder. In particular, the message needs to be encoded in a manner that allows the receiver to correctly decode the message with the channel induced errors. Also, this encoding should prevent any malicious presence from having the ability to deduct anything about the original message. A related problem is how to assess the reliability and integrity of a wireless network.

1.1 Wireless Networks

Wireless networks are composed of multiple nodes connected without the use of wires. These nodes are used to communicate data between two end points which could be any one of the nodes. There are various forms of wireless networks which can be divided into two classes, decentralized and centralized networks. Decentralized networks are referred to as ad hoc networks, and unlike centralized networks, they do not depend on a pre-existing infrastructure. This means that they have frequent changes in topology due to link failures, physical obstructions, network intrusions, etc. Dependability on these sort of networks requires dynamic algorithms which quickly determine routing, redundancy, etc.

for deviations in the network. Centralized networks depend on routers or regulated access points. Examples of centralized networks include wireless personal area network (WPAN), wireless local area network (WLAN), and mobile device networks. Three types of wireless ad hoc networks include mobile ad hoc networks, wireless mesh networks, and wireless sensor networks.

Wireless personal area networks are networks that connect devices in a small area (normally around a person). A Bluetooth earpiece which connects to a cell phone is an example of WPAN.

Wireless local area networks connect computers in a home, office, school, etc. setting. WiFi is an example of a WLAN which allows devices with WiFi capabilities to connect with one another and also to connect to internet. Wireless metropolitan area networks connect multiple WLAN's together in a metropolitan area.

Mobile device networks connect mobile devices to the internet and to one another. Cellular networks are mobile device networks which allow cell phones to surf the internet, send/receive data, and hold conversations.

Mobile ad hoc networks are networks in which the nodes in the network are able to change their location, they are self-configuring. This mobility results in frequent network connectivity changes causing the need to develop techniques for dynamic forwarding of data so that it arrives at the desired location. There are many examples of the use of mobile ad hoc networks, a popular one being WiFi connections with laptops/communication devices. The growing use of wireless communication devices has caused these networks to receive a large amount of attention.

Wireless mesh networks are built with nodes in a mesh structure and unlike most traditional wireless networks, only one node needs to be wired to the internet (such as a DSL modem). Mesh clients are devices such as cell phones and laptops, and mesh routers forward the data through the network. The cluster of nodes which can communicate with one another are referred to as a connectivity cloud. Mesh networks have a variety of important applications one being linking WiFi hot spots together and keeping an entire city connected. This allows people to check their email at places such as a park, coffee shop or bus, and also keeps emergency workers connected to the network when cell phones are not working. Another important application of mesh networks is in developing countries that lack wired infrastructure. The nodes, dispersed in the country, can be solar powered and connected to cellular or satellite internet connections.

Wireless sensor networks are formed with wireless nodes that collect data in a distributed fashion. The sensor nodes are generally small and cheap, and they aggregate some type of information (temperature, sound, pressure, etc.) around them. Sensor networks can have nodes that have enough computational power to analyze the received data, and they can also be formed with nodes that just measure and forward data to a central estimation node which performs the data analysis. Sensor networks can be used in a large amount of applications ranging from battle field to medical purposes. In a battle field they can be used to detect the presence of enemies. They can also be used to monitor the health of an elder person. A person can wear sensors in a variety of places on his/her body, and the sensor network could detect if the person fell, or had any sort of problem and convey this information to medical doctor, emergency services, etc.

1.2 Network Attacks

Wireless networks have been the target of many sorts of attacks. The following are several types of attacks on wireless networks:

- **Eavesdropping:** When an illegitimate user spies on packets transmitted through the network.
- **Denial of Service (DoS):** When a malicious user prevents data communication, network functionality.
- **Intrusion:** Unauthorized user joins network and uses network resources.

Without proper security measures, eavesdropping may result in an adversary obtaining a users private information. Denial of service attacks result in the ceasing of network use for a legitimate user. Denial of service attacks could be caused by RF jammming, i.e. sending bursts of power through the medium and preventing legitimate messages from getting through, refusing to route a message to the correct location, sending incorrect packets instead of legitimate messages, etc. Intrusion attacks reduce available bandwidth for paying customers. It is vital that networks are protected from such attacks and there is a significant amount of research currently being done in this area.

1.3 Network Reliability

An important attribute that all the types of wireless networks must have is data reliability. Data transmitted from one point to another must be received reliably at the receiving point hence enabling communication between the transmitter

and receiver. Cellular networks need to allow cell phones to communicate reliably with one another even in adverse conditions. Wireless networks are prone to all sorts of errors due to the medium through which their information travels and hence it is necessary to protect this data, to add more information about the message in the case of errors. An error in a message could change it to a different set of bits or it could delete parts of the message. DoS attacks cause parts of or entire packets to be lost. A common approach to mitigate errors caused by a channel is through the use of codes.

1.4 Network Security

Wireless networks must also guarantee a level of security in the data communication. The messages must be kept confidential between the sender and receiver. The destination must also be able to tell the legitimacy status of a message, whether it has been tampered with. The research in this work primarily focuses on keeping a message confidential from an adversary. Proper encryption measures make it extremely tough for an eavesdropper to extract any sensitive information about the data. If an eavesdropper attains private information, which could be a password to a bank or email account, this would cause significant negative impact to the compromised user. Similarly, consider a body sensor network that measures the health and actions of an elder. If something happens to the person and this information is attained by a health insurance company, they could charge the elder higher rates. Similarly, if a crook attains information that the elder has fallen down the stairs or is debilitated, he/she can use this to their advantage and break into the house. The network must also be protected from unauthorized use or any malicious users who tamper-

with/destroy data. False packets and not forwarding legitimate packets results in lower network throughput, thus it is necessary for methods to take care of these types of attacks.

1.5 Network Integrity

The message must be kept secret from an adversary, as well as protected from any damage that the adversary may cause. Determining the location where the malicious nodes reside is another extremely important task since it allows the network to take appropriate action against them. This could mean removing the node from use or lowering the "trustworthiness" of the node. The localization task is tough since it requires close monitoring of all of the nodes in the network since the final destination of a message will not know exactly where the attack occurred. Having each node check the received packets from all of its neighbors is computationally inefficient and requires higher processing on the nodes. There has been a large quantity of research done in determining the "trustworthiness" of nodes ranging from game theoretical solutions to nodes rating their neighbors.

1.6 Contribution

The work in this dissertation addresses research done by the author in the three important wireless network topics mentioned earlier: network reliability, network security, and network integrity. Suggested solutions to these issues are all based on the use of coding theory, in particular, using certain types of error-

correction codes.

The first piece of contributed work found also in [9] and [11] considered the network reliability problem. A multiple path wireless network is assumed through which a source-destination pair would like to communicate. Moreover, the presence of adversaries on some paths may cause information to be lost or corrupted. The presence of malicious nodes and noise are modeled as an erasure channel. The questions that were asked were: "how should the data be routed and how much redundancy should be added to the original message to guarantee at least a specified level of success?". The authors solutions to these problems include the design of two algorithms MRAET (exponential time) and MRAPT (polynomial time) to determine minimum redundancy and optimal symbol allocation to attain a probability of success. The performance of the algorithms are compared with respect to each other and the desired success level. The algorithms are tested and compared on three different error-correction codes, namely MDS, LT, and Raptor codes. The MDS, LT, and Raptor code parameters are designed to be compatible with the algorithms. The performance of the codes is evaluated using the MRAET algorithm.

The next subject researched was related to network secrecy. Specifically, the network is assumed to be a multiple path wireless network in the presence of an eavesdropper. The question that was answered is: "What kind of coding scheme can be used which not only adds redundancy to correct channel errors but also guarantees that the eavesdropper cannot extract anything about the original message given the information she has received?". In particular, the introduced method uses Raptor codes which guarantees the desired destination to have zero error probability asymptotically and to have perfect secrecy against

an eavesdropper. Additionally, a method to efficiently route the data across multiple, simultaneous paths is introduced [10].

The last piece of work in this thesis considers the network integrity issues. The assumption on the network is that it is a multiple node wireless network with a point to point communication problem. The question that needed to be answered in this work was: "What kind of technique can be developed to determine the locations of malicious nodes in the network?". A technique is developed which determines the locations of malicious nodes in the network using error control codes. Reed-Muller codes are applied to paths in the network and the decoding at the destination node reveals the byzantine nodes [12]. The method depends on the probability of a node being malicious as well as the number of nodes in the network. An algorithm is introduced which deals with the case when all the paths are not realizable in the network and this is followed by comparison of other techniques.

1.7 Thesis Organization

The thesis will continue next with Chapter 2 on important background theory, such as coding and information theory, which is necessary to understand the contributed work. Following, Chapter 3 will discuss related work to the research found in this dissertation. Chapter 4 gives a detailed analysis and description of the two algorithms MRAPT and MRAET which assure network reliability. The method to assure network secrecy and reliability using raptor codes is covered in Chapter 5. The derived tactic which uses Reed-Muller codes to determine the locations of adversary nodes is described along with simula-

tions in Chapter 6. Finally, Chapter 7 concludes the thesis.

CHAPTER 2

BACKGROUND

2.1 Information Theory

Mathematical theory of communication systems, or information theory, was first formulated by Claude Shannon, who is now known as the father of information theory, in 1948 in [30]. Shannon considered the encoding of a message prior to being transmitted across a channel. He developed a metric which conveys the amount of information present in a message. Basic Shannon theory will be discussed next.

A metric which represents the amount of information/ level of uncertainty present in a random variable (r.v.) is called *entropy*. Assume that $p_X(x)$ is the probability distribution for $x \in \mathcal{X}$.

Definition 1 ***Entropy***

$$H(X) = -\sum_{x \in \mathcal{X}} p_X(x) \log p_X(x)$$

The entropy for two r.v.'s X and Y is defined as follows.

Definition 2 ***Joint Entropy***

$$H(X, Y) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p_{X,Y}(x, y) \log p_{X,Y}(x, y)$$

The conditional entropy of a r.v. given another is defined as:

Definition 3 ***Conditional Entropy***

$$H(X|Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p_{X,Y}(x,y)\log p_{X|Y}(x|y)$$

Mutual Information, $I(X;Y)$, is a mathematical function which quantifies the amount of information between two r.v.'s. If two r.v.'s X and Y are independent then their mutual information is zero, since having one r.v. gives absolutely no information about the other.

Definition 4 ***Mutual Information***

$$\begin{aligned} I(X;Y) &= \sum_{x\in\mathcal{X},y\in\mathcal{Y}} p_{X,Y}(x,y)\log_2\left(\frac{p_{X,Y}(x,y)}{p_X(x)p_Y(y)}\right) \\ &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \end{aligned}$$

Assume that $M \in \mathcal{M}$ represents the message random variable and $X \in \mathcal{X}$ the codeword r.v. The decoder receives $Y \in \mathcal{Y}$ which is the altered version of X after it traverses through a channel with probability distribution $p(X|Y)$. Let $p_X(x)$ represent the distribution from which the r.v. X comes from.

For discrete memoryless channels, the maximum rate which a message can be transmitted at, which results in arbitrarily low error probability in the reconstructed message, is called the *channel capacity*.

Definition 5 ***Channel Capacity***

$$C = \max_{p(x)} I(X;Y)$$

2.2 Galois Fields

Finite fields were come across by French Mathematician Evariste Galois, and hence also referred to as *Galois Fields*. Galois fields are useful in the construction of algebraic codes such as Reed-Solomon codes. A Galois field with q elements is labeled GF(q), and q is of the form p^m where p is a prime number and m is a positive integer. Next some relevant definitions found in [38] will be mentioned in order to fully understand the structure of finite fields.

Definition 6 ***Groups***

*A **group** G is a set with binary operation "$\cdot$" defined. The binary operation between any two elements in G ensures the result is also in G, closure. The operation also guarantees the following properties:*

1. *Associativity:* $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
2. *Identity:* $\forall a \in G, \quad \exists\, e \in G \;:\; a \cdot e = a$
3. *Inverse:* $\forall a \in G, \quad \exists a^{-1} \;:\; aa^{-1} = e$

   A commutative group has one more property:

4. *Commutativity:* $a \cdot b = b \cdot a \quad \forall a, b \in G$

Definition 7 ***Field***

*A **field** F is a set of elements with two binary operations, $\cdot$ and $+$, with the following properties:*

1. *F forms a commutative group under $+$, with additive identity element* 0.
2. *$F\backslash\{0\}$ forms a commutative group under $\cdot$,* 1 *being the multiplicative identity.*

3. *Distributive property under* $\cdot$ *and* $+$, $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$

Definition 8 ***Finite Field***

A ***finite field*** *is a field* F *with finite cardinality represented as* $GF(q)$*, where* $GF(q) = \{0, 1, \ldots, q-1\}$*.* $GF(q)$ *has cardinality* q*.*

Definition 9 ***Order***

The ***order*** *of an element* $\beta \in GF(q)$ *is the smallest* $m \in \mathbb{Z}^+$ *such that* $\beta^m = 1$*.*

Definition 10 ***Primitive Element***

A ***primitive element*** *of a finite field* $GF(q)$ *is an element which has order* $q - 1$*.*

It can be shown that the first element to repeat in a finite field is always 1, hence the non-zero elements of GF(q) can be completely represented by the powers of a primitive element α since $\{\alpha^{q-1} = 1, \alpha^1, \alpha^2, \ldots, \alpha^{q-2}\}$ are $q - 1$ distinct elements.

2.3 Coding Theory

Communication systems use three different classes of codes: source codes, secrecy codes, and channel codes. The background theory behind these classes is mentioned in the next several sections. Channel codes are most often referred to as error-control or error-correction codes (ECC). The research in this thesis focuses only on secrecy and channel codes.

2.3.1 Source Codes

Source codes map a message source to another alphabet so that the message source can be perfectly decoded (lossless source coding) or so that the message can be decoded within a specified distortion level (lossy source coding). Source codes are primarily used for data compression and these codes usually reduce the size of the original message.

Shannon introduced a metric which determines the minimum number of symbols needed to be able to reconstruct a source message with negligible error. He proved that if there were any less symbols, then information would be lost. This result is called the source coding theorem and is shown next.

Theorem 1 ***Source Coding Theorem***
Consider a source of n independent identically distributed (i.i.d) variables $X_1, \ldots X_n$ *over probability distribution* $p(X)$*. The source can be reliably encoding with arbitrarily small error probability with* $nH(X)$ *bits or at a rate R (bits per source symbol) if*

$$H(X) \leq R$$

Otherwise the error will be bounded away from zero.

2.3.2 Secrecy Codes

A secrecy code is one which removes information about the original message. In simple terms, the codeword reveals little to no information about the message. In 1949, Shannon first introduced communication secrecy in information-theoretic terms [31]. The code is an invertible mapping from the message space

to another space, the ciphertext space. Going from the ciphertext space to the message space requires a pre-known key, and unique deciphering is essential.

The key is chosen and distributed to the destination points. Then a message is chosen, encrypted and transmitted to the destination. It is possible that the ciphertext is intercepted or seen by an adversary prior to reaching the desired recipient. It is assumed that the adversary has a priori knowledge of the key and message probabilities. Using these probabilities and the intercepted ciphertext, the enemy cryptanalyst can calculate the a posteriori probabilities that a particular message and key were used to generate the ciphertext.

Shannon wanted to determine how safe a system is from cryptanalysis if the adversary has unlimited time and power. Suppose that our sender is Alice and she wants to send a message to the receiver Bob. The adversary Eve wishes to obtain the message sent from Alice to Bob. Eve is aware of the message alphabet and the probability distribution over the messages prior to receiving transmitted data. Assume that the message comes from a set $\mathcal{X}$ with a priori distribution $p(X)$, and the ciphertext come from a set C. The messages are mapped to the ciphertext through the use of a transformation T_K which is a function of the key K. The key is chosen independently from the message and the transformation. From the intercepted ciphertext, Eve can calculate the a posteriori probability for all the messages in the set. Assume this a posteriori distribution is $p_C(X) = p(X|C)$.

Shannon came up with the notion of *perfect secrecy* in a system, which represents the case when the encoded data stolen by an illegitimate user does not give any information about the original message. In particular, it means that for all C the a posteriori distribution is the same as the a priori distribution over the

messages independent of C. Mathematically, $p_C(X) = p(X)$. The uncertainty in X is $H(X)$ and the uncertainty in the key is $H(K)$. Shannon showed that perfect secrecy required that $H(X) \leq H(K)$ implying that Alice and Bob have to share a key that is at least as long as the message.

Another important concept Shannon introduced is called *equivocation* which represents the amount of uncertainty one has given the ciphertext. Two important equivocation expressions include the message equivocation $H_C(X)$ and the key equivocation $H_C(K)$. It can be shown that $H_C(X) \leq H_C(K)$. For perfect secrecy, $H_C(X) = H(X|C) = H(X)$ or equivalently $I(X;C) = 0$, each key must be equally likely, and $H_C(k) = H(k)$.

More realistically in a wireless channel both Bob and Eve will receive corrupted versions of the codeword C. Wyner [40] extended Shannon's notion of communication secrecy to the scenario where Bob and Eve have a channel between them and Alice. Their channels are different and Bob's channel is called the main channel. Both security and reliability have to be considered in this model. Bob and Eve receive Y, Z respectively and $X \rightarrow C \rightarrow (Y, Z)$ form a Markov chain. In this case perfect secrecy is attained if $I(X;Z) = 0$. Eve's equivocation is represented as $\Delta = \frac{1}{k}H(X|Z)$, where X is a k-bit vector and Z an n-bit vector. It is desirable to have the largest equivocation possible. Wyner showed that if the capacity of the eavesdroppers channel is less than that of Bob's (i.e. Eve's channel is worse than Bob's) then there exists a coding method which guarantees both reliability and security.

Secrecy capacity represents the maximum possible transmission rate between Alice and Bob such that perfect secrecy from Eve is achieved. It can be shown that for a simple wiretap channel, where the eavesdroppers channel is

worse than the main channel, the secrecy capacity is:

$$C_s = \max_{p_X(x)} [I(X;Y) - I(X;Z)] \tag{2.1}$$

2.3.3 Channel Codes/Error-Correction Codes

Channel codes are applied to a message to protect it from errors caused by the channel through which it will travel. Channel codes are often referred to as error-control/correction codes. Error-control codes are beneficial tools for either correcting or flagging errors incurred by a message. An (n, k) error-correction code maps a k-symbol message into an n-symbol codeword, where $k < n$ and the $n - k$ extra symbols contain redundant information about the message symbols. Given the code C, the extra symbols give the decoder the capability of correcting and detecting a certain number of errors. Generally, codes are able to detect more errors than they can correct. Next, the notation and definitions used in analyzing error-correction codes will be introduced.

Definition 11 ***Rate***

The rate of an (n, k) *code is defined to be* $r = \frac{k}{n}$.

As mentioned earlier, the upper bound on the code rate varies depending on the channel through which the message traverses and is referred to as the *channel capacity*.

Definition 12 ***Weight***

The ***Weight*** *of a code word is the number of non-zero symbols.*

Definition 13 ***Hamming Distance***

The ***Hamming Distance*** *between two codewords* $\mathbf{u}, \mathbf{v}$ *of length* n *is the number of positions in which they differ or,*

$$d_{Hamming}(\mathbf{u}, \mathbf{v}) = d(\mathbf{u}, \mathbf{v}) = |\{i | u_i \neq v_i,\ i = 0, 1, \ldots, n-1\}|$$

Definition 14 ***Minimum Distance of a Code***

The ***Minimum Distance*** *,* $d_{\min}$ *of a code is the minimum Hamming distance between all distinct codewords in the codebook.*

An (n, k) code is one which starts with a message of length k and encodes it to a codeword of length n, or adds $n - k$ redundant symbols. We call the ratio $\frac{n}{k} = \gamma$.

Theorem 2 ***Singleton Bound***

The minimum distance $d_{\min}$ *for an* (n, k) *code is bounded by*

$$d_{\min} \leq n - k + 1$$

A code with minimum distance $d_{\min}$ can detect all error vectors with weight less than or equal to $d_{\min} - 1$ and can correct all error patterns with weight less than or equal to $t = \lfloor \frac{d_{\min}-1}{2} \rfloor$.

2.3.4 Maximum-Distance Separable Codes

Definition 15 ***Maximum-Distance Separable Code***

Maximum-distance separable (MDS) codes meet the Singleton Bound with equality, or:

$$d_{\min} = n - k + 1$$

From [38] it is known that for any (n, k) MDS code in systematic form, any combination of k out of the n codeword symbols allows for perfect recovery of the original message. This attribute serves well in erasure channel scenarios because, with complete certainty, a message can be decoded with up to $n - k$ erasures.

An important and well known MDS code is the Reed-Solomon (RS) code. RS codes are thus excellent candidates for channels with bursts of error such as CD's, DVD's, and the newer blu-ray discs. The reason for this is the fact that these data discs get errors when the disc gets scratched or dirty in a specific area resulting in error bursts. The next section provides a detailed review of Reed-Solomon codes.

Reed-Solomon Codes

Reed-Solomon codes were invented in 1959 by Irving S. Reed and Gustave Solomon at MIT Lincoln Laboratory and their work was published in 1960 [38]. Reed-Solomon Codes are a special case of BCH codes [38], they are non-binary BCH codes, specifically, q^m-ary BCH codes of length $q^m - 1$. BCH codes were discovered by two different research groups at approximately the same time, A. Hocquengham in 1959 followed by Bose and Ray-Chaudhuri's publications. The relationship between BCH and Reed-Solomon Codes was discovered through work published by Gorenstein and Zierler on the extension of BCH codes to arbitrary field sizes. There are three approaches for forming RS codes: the original approach, the generator polynomial approach, and finally the Galois Field Fourier Transform (FFT) approach [39]. The original approach is the method which Reed and Solomon introduced. The generator approach

construction of RS codes was realized after the publication of Gorenstein and Zierler's work.

The original approach is a fairly simple approach. Consider a k-symbol message $\mathbf{m} = \{m_0, m_1, \ldots m_{k-1}\}$, where each symbol $m_i \in \mathrm{GF}(q)$. A polynomial can be constructed from the message symbols in the following fashion: $P(x) = m_0 + m_1 x + \ldots + m_{k-1}x^{k-1}$. Then, using the original approach, the $n = q$-symbol codeword for this message can be formed using a primitive element $\alpha \in \mathrm{GF}(q)$ as: $\mathbf{c} = (c_0, \ldots, c_{q-1}) = (P(0), P(\alpha), \ldots P(\alpha^{q-1}))$. All possible codewords can be found by considering all possible k-symbol messages.

The generator approach is based on *cyclic codes* and is the most popular form in coding literature [39]. *Cyclic codes* are codes ones in which if $\mathbf{c} = (c_0, \ldots, c_{n-1})$ is a codeword then $\mathbf{c}' = (c_1, \ldots c_{n-1}, c_0)$ is also a codeword. An (n, k) cyclic code can be represented with a generator polynomial $g(x) = g_0 + g_1 x + \ldots + g_{n-k}x^{n-k}$. In this construction, the codeword $\mathbf{c}$ is seen as a codeword polynomial or $c(x) = c_0 + c_1 x + \ldots c_{n-1}x^{n-1}$. A codeword $\mathbf{c}$ is part of a code with generator $g(x)$ only if $c(x)$ is a multiple of $g(x)$. Assume that the message polynomial is $m(x) = m_0 + m_1 x + \ldots m_{k-1}x^{k-1}$. The codeword in the code with generator $g(x)$ for message $m(x)$ is: $c(x) = m(x)g(x)$. The generator polynomial is generated as follows. For a t error correcting code, take a primitive element $\alpha \in \mathrm{GF}(q)$ and the generator polynomial is:

$$g(x) = \prod_{i=1}^{2t}(x - \alpha^i)$$

More detailed theory can by found in [39]

The Galois FFT (GFFT) approach is the classical approach in terms of fourier transforms. Consider the codeword $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$ and its fourier trans-

form, written $\mathcal{F}\{\mathbf{c}\} = (C_0, C_1, \ldots C_{n-1})$. Where C_i is defined as:

$$C_j = \sum_{i=0}^{n-1} c_i \alpha^{ij} \quad \text{where } j = 0, \ldots, n-1$$

It can be shown that the GFFT approach is the dual to the generator polynomial approach, and leads to efficient encoders and decoders [39]. There are many algorithms for decoding Reed-Solomon codes, we refer the reader to [39] for detailed descriptions.

Compact Disc's (CD's) were the first application of RS codes which were mass produced in 1982. In particular, the CD uses a form of RS code called Cross-Interleaved RS codes or CIRC. CIRC is composed of the concatenation of two layers of an RS code separated by an interleaver. Each symbol consists of 8-bits and a message block or frame contains 24 symbols. The first layer takes a message block and encodes it into 28 symbols using a $(28, 24)$ RS code. These 28 symbols are then sent through an interleaver which disperses the information in this frame among all the 109 frames. Next, the second layer $(32, 28)$ code is applied to each frame out of the 109 frames. The newly formed frames have 32 symbols and they are sent through a different interleaver. The decoding is applied in the opposite fashion. The interleaving is useful because it disperses the errors among the frames resulting in higher error-correction. These codes are also used for forward error correction in data transmission. Due to the minimum distance of RS codes, they are extremely useful for erasure channels. As mentioned above, any k out of n symbols can be used to to generate a lossless reconstruction of the original message. Another application includes satellite communications. RS codes concatenated with convolutional codes were used in the Voyager to encode digital pictures. They were also used on the Mars Pathfinder, Galileo, Mars Exploration Rover, and Cassini missions in concatenation with convolutional codes.

Reed-Solmon codes are extremely useful since their codewords are maximally spaced apart, though there is a trade-off between this property and running time complexity. The encoding and decoding times for an RS code are quadratic with the codeword size. This implies that these codes are optimal for small message and codeword size. The problem is that most applications require fairly large source sizes. In a following section, LT codes are introduced, which are almost MDS codes and have reduced running times. For further theory on the Reed-Solomon code we encourage the reader to go to [39].

2.3.5 Low-Density Parity-Check codes

Robert Gallager invented Low-Density Parity-Check (LDPC) codes in 1963 [7]. As the name implies, an LDPC code has an extremely sparse parity matrix, meaning that the rows and columns have a large amount of '0's and small amount of '1's. These codes are linear and are derived from sparse bipartite graphs. LDPC codes have received a lot of recent attention due to the fact that they are capacity approaching codes on symmetric memoryless channels. Before getting into the details of constructing LDPC codes, some applications will be mentioned.

LDPC codes are used in the 10 Gigabit Ethernet standard, the fastest ethernet standard with data speeds up to 10 Gbps as the name implies. In 2003, LDPC codes were chosen for the Digital-Video Broadcasting-Satellite-Second generation (DVB-S2), which is the standard for the satellite transmission of digital tv. ITU-T picked LDPC codes for the G.hn standard, the next generation home network technology (currently being developed) which supports networking over

power lines, phone lines, and coaxial cables with data rates up to 1 Gbps.

Consider a sparse bipartite graph with n left nodes and r right nodes. The left nodes are called the message nodes and the right ones are called check nodes. This graph can be represented by a matrix H, where a '1' in spot (i, j) represents a connection between the ith message node and jth check node. H is the parity matrix of the code. Regular LDPC codes are ones in which all the columns have the same number of '1's and all the rows have the same number of '1's. The codewords $\mathbf{c} = \{c_1, \ldots, c_n\}$ for the LDPC code are such that $H\mathbf{c}^T = \mathbf{0}$.

LDPC codes are decoded using message passing algorithms, most commonly the Belief Propagation algorithm. Message passing algorithms are iterative algorithms in which a message is passed between the message and check nodes. The Belief Propagation algorithm is discussed in the next section.

Belief Propagation Decoding Algorithm

The belief propagation algorithm is an iterative algorithm in which the messages passed between check and message nodes are probabilities. A message from a message node v to a check node r would represent the probability that the v has a specific value given the values of other incident check nodes received in the preceding round. The message transmitted from r to v would contain information about the data that r received from message nodes other than v in the previous round.

Assume that H is the parity matrix and that $H_{i,j}$ represents the item in the ith row (check bits) and jth column (message bits). Let n denote the set of message bits and r the check bits. The set $\mathcal{N}(m) = \{n : \ H_{m,n} = 1\}$ denotes the set of

message bits which participate in check m and $\mathcal{N}(n) = \{m : \ H_{n,m} = 1\}$ the set of check bits which message bit n participates in. Assume the codeword $\mathbf{c}$ was transmitted and $\mathbf{r} = \mathbf{c} + \mathbf{e}$ is received. Call $\mathbf{z} = H\mathbf{r}^T$ the *syndrome*. If there is no error then $z = 0$, otherwise $z \neq 0$. To proceed with the algorithm it is necessary to know the noise distribution and the prior probability of the input symbols. If $z \neq 0$ then we know that $z = He^T$ since $Hc^T = 0$, so the goal is to find the variable $\mathbf{x} = \{x_1 \dots x_n\}$ such that $H\mathbf{x}^T = \mathbf{z}$. In particular, pick the $\mathbf{x}$ that maximizes $P(\mathbf{x}|H\mathbf{x}^T = \mathbf{z})$. Define $q^x_{mn} = P\{x_n = x|$ info from check bits $\mathcal{N}(n)\backslash m\}$ and $r^x_{mn} = P\{$check bit m is satisfied$|x_n = x$ and $q_{mn'} : n' \in \mathcal{N}(m)\backslash n\}$.

- **Initialization**

  1. $p^0_n = P(x_n = 0)$, $p^1_n = P(x_n = 1) = 1 - p^0_n$
  2. $\forall(n, m)$ with $H_{m,n} = 1$ set $q^0_{mn} = p^0_n$, $q^1_{mn} = p^1_n$

- **Horizontal Step** (running through checks)

$$r^0_{mn} = \sum_{\{x_{n'} : n' \in \mathcal{N}(m)\backslash n\}} P\left(z_m | x_n = 0, \ \{x_{n'} : n' \in \mathcal{N}(m)\backslash n\}\right) \prod_{n' \in \mathcal{N}(m)\backslash n} q^{x_{n'}}_{mn'}$$

$$r^1_{mn} = \sum_{\{x_{n'} : n' \in \mathcal{N}(m)\backslash n\}} P\left(z_m | x_n = 1, \ \{x_{n'} : n' \in \mathcal{N}(m)\backslash n\}\right) \prod_{n' \in \mathcal{N}(m)\backslash n} q^{x_{n'}}_{mn'}$$

- **Vertical Step**

$$q^0_{mn} = \alpha_{mn} \prod_{m' \in \mathcal{M}(n)\backslash m} r^0_{m'n}$$

$$q_{mn}^1 = \alpha_{mn} \prod_{m' \in \mathcal{M}(n) \setminus m} r_{m'n}^1$$

where α_{mn} is chosen to satisfy $q_{mn}^0 + q_{mn}^1 = 1$.

$$q_n^0 = \alpha_n p_n^0 \prod_{m' \in \mathcal{M}(n)} r_{mn}^0$$

$$q_n^1 = \alpha_n p_n^1 \prod_{m' \in \mathcal{M}(n)} r_{mn}^1$$

The q_n^x are used to form an estimate for x_n. If $q_n^1 > 0.5$ set $\hat{x}_n = 1$. If $\hat{\mathbf{x}}$ is such that $H\hat{\mathbf{x}}^T = \mathbf{z}$, then halt algorithm, otherwise return to horizontal step and halt after a maximum amount of iterations.

2.3.6 Digital Fountain Codes

Digital Fountain Codes are a class of sparse-graph codes designed for erasure channels. Unlike typical codes used for erasure channels, such as Reed-Solomon codes, fountain codes are *rateless* codes which implies that the symbols can be determined on the fly. Fountain codes are almost MDS meaning that with k input symbols, the decoder needs about $k(1 + \epsilon)$ symbols for successful decoding. Typical network protocols often use a feedback channel to mitigate the effects of an erasure channel, through which the receiver notifies the sender of any lost packets. Classic Shannon theory has shown that the capacity of a discrete memoryless channel with feedback is equivalent to the same channel without feedback, implying that feedback channels are wasteful and unnecessary for this class of channels. For high erasure probabilities, there is a significant amount of information transmitted over the erasure channel [19]. Rateless codes do not

depend on a rate implying that as an unlimited amount of output symbols can be transmitted. This is due to the fact that each output symbol does not depend on a previous or future output symbol. Feedback channels are completely unnecessary for rateless codes since as many output symbols can be transmitted as needed to successfully decode the message.

Luby Transform (LT) Codes

Luby Transform (LT) codes, popular fountain codes for erasure channels, were invented by Michael Luby in 1998 and formally published in [18] in 2002. These codes are based off of sparse bipartite graphs. A simple analogy to LT codes is holding a bucket and catching drops until a sufficient amount of drops are caught. The codes can be thought of as the balls and bins problem, where the bins represent the input symbols and the balls being the encoded symbols. The question is, how many balls does one have to throw so that with probability $1-\delta$ each bin has at least one ball. Given that there are k bins, the number of balls should be around $k \log \frac{k}{\delta}$ [18]. This yields small encoding and decodings times both on the order of $k \log \frac{k}{\delta}$. We will first introduce the encoding and decoding algorithms for the LT codes.

Suppose we have a source message $m_1 m_2 \dots m_k$, and we wish to form an output symbol c_i. The encoding of the message goes as follows:

1. Pick d_n from a degree distribution $\rho(d)$. The specifics of $\rho(d)$ will be revealed below.
2. Choose d_n distinct input packets uniformly at random and call the set of their indices I_{d_n}. Then using modulo 2 arithmetic, $c_n = \sum_{j:\, j \in I_{d_n}} m_j$.

These codes are sparse because the constructed degree distribution results in a mean degree that is considerably smaller than k. There are many different ways to relay the degree and edge connectivity of the graph to the decoder such as synchronized clock, sending header containing a key, etc.

Successful decoding depends on the degree distribution used and the edges in the graph G. The decoding is quite simple due to the encoding structure and it goes as follow:

1. Find output symbol c_n that is connected to only one input symbol m_i. If there is no such output node then the decoding fails causing incomplete information about the input message.
   (a) Let $m_i = c_n$
   (b) Let C_{m_i} be the set of indices of output symbols which are connected to m_i , or all j such that $G_{i,j} = 1$, and set
   $$c_j = c_j + m_i \qquad \forall\ j \in C_{m_i}$$
   (c) Remove the edges connected to the input symbol m_i
2. Go back to step 1 until all the input symbols are determined.

This decoding is a simplified version of the belief propagation algorithm also known as the sum-product algorithm [19]. Using terminology as in [13], the *ripple* represents the set of output symbols of degree one. If the decoding algorithm reaches a point where the ripple is empty prior to decoding all the input symbols, then the receiver fails to obtain the source message. Hence, it is vital to design a degree distribution which assures with high probability that

the ripple is always nonempty prior to decoding all the input symbols, and also, that all the input symbols are connected to at least one output symbol. In theory, the ideal degree distribution is called the *ideal soliton distribution*,

$$\begin{aligned} \rho(1) &= \frac{1}{k} \\ \rho(d) &= \frac{1}{d(d-1)} \quad \text{for } d = 2, 3, \ldots, k \end{aligned}$$

which behaves as it should in expectation. The expected degree of this distribution is about $\log k$. The problem with this distribution is that even the smallest oscillation around the expected degree results in the failure of the decoding algorithm because there is no degree one check node. To take care of this issue, Luby introduced a slightly altered degree distribution which he calls the *robust soliton distribution* [18]. This distribution avoids the problem of not having an output node with degree one by guaranteeing that the expected number of degree one outputs is approximately $R = c \log(k/\delta) \sqrt{k}$ ($c > 0$). The robust soliton distribution is:

$$\mu(d) = \frac{\rho(d) + \tau(d)}{Z}$$

where $Z = \sum_d \rho(d) + \tau(d)$ and $\tau(d)$ is defined as follows,

$$\tau(d) = \begin{cases} \frac{R}{kd} & \text{for } d = 1, 2, \ldots, \frac{k}{R} - 1 \\ \frac{R}{k} \log(\frac{R}{\delta}) & \text{for } d = \frac{k}{R} \\ 0 & \text{for } d > \frac{k}{R} \end{cases}$$

Using this distribution there need to be at least $n = kR$ encoded symbols so that with probability $1-\delta$ we can successfully decode the message [18]. Raptor codes, discussed next, are an extension of LT codes which have a linear encoding and decoding time [33].

Raptor Codes

Raptor codes were developed by Amin Shokrollahi in 2001. Raptor codes are a class of fountain codes which have the property that encoding and decoding have a constant cost (in the per symbol sense). LT codes have a per symbol decoding on the order of $O(\log(k))$ since the decoding graph must have roughly $k\log(k)$ edges to ensure that the input symbols are all covered with high probability. Raptor codes diminish this condition to having a certain fraction of recoverable input nodes, which results in constant decoding cost [33]. Since the goal is to be able to recover all the input symbols, Raptor codes first apply a classical erasure code to the input symbols, followed by the LT code. A Raptor code is of the form $(k, C, \Omega(x))$, where C is the first layer code and $\Omega(x)$ is the output symbol distribution. The encoding goes as follows:

1. Using the code C, encode the message (m) of length k symbols into a codeword (c) of n symbols.
2. Apply the LT code algorithm to the codeword c resulting in another codeword c' which is slightly larger than c. The LT algorithm should use the specified output distribution $\Omega(x)$.

The choice of the code C effects the encoding and decoding costs, and also the decoding algorithm. Choices for C include Tornado codes, LDPC codes, extended Hamming codes, etc [20]. In our paper we use a regular Gallager LDPC code for C.

2.3.7 Reed-Muller Codes

In 1954, Muller discovered what are now known as Reed-Muller (RM) codes using a "Boolean net function" language. Later that year, Reed recognized that Muller's codes could be mapped into multinomials over the binary field, which resulted in Reed-Muller codes [38]. RM codes have had few applications with comparison to other codes but they are still an important type of error-correction codes. The applications are discussed next.

First-order RM codes of length 32 were used for error-correction on all *Mariner* Mars spacecraft missions from 1969 to 1977 [38]. Another more recent application of RM codes is in the third generation (3G) cellular wideband CDMA standard. They are used in the dedicated physical control channel (DPCCH) which is one of the two uplink channels. Specifically, each 10-milisecond length frame of the DPCCH is composed of 15 time slots of 10 bits each. These 10-bits are time-multiplexed over four fields, one of which is the transport format combination indicator (TFCI). The RM codes are used to encode the TFCI bits [8]. An advantage of these codes is that they have an incredibly fast maximum likelihood decoding algorithm developed by Reed, the Reed decoding algorithm [38]. The structure of these codes allows us to develop a method to find the adversary nodes. We will discuss this in a later section. The theory behind RM codes, found in more detail in [38], is discussed next.

Consider m binary linearly independent vectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_m}$. Let M be a set that contains all possible boolean functions of these m vectors that are repre-

sented by a monomial term,

$$M = \{\mathbf{1}, \mathbf{v_1}, \ldots, \mathbf{v_m}, \mathbf{v_1v_2}, \ldots, \mathbf{v_{m-1}v_m}, \mathbf{v_1v_2v_3}, \ldots, \mathbf{v_{m-2}v_{m-1}v_m}, \ldots, \mathbf{v_1v_2} \cdots \mathbf{v_m}\}$$

From [38] it is known that there is a unique Boolean function f for each vector $\mathbf{f}$ composed as:

$$\mathbf{f} = a_0\mathbf{1} + a_1\mathbf{v_1} + \ldots + a_m\mathbf{v_m} + a_{12}\mathbf{v_1v_2} + \ldots + a_{12\ldots m}\mathbf{v_1v_2} \cdots \mathbf{v_m} \tag{2.2}$$

Definition 16 ***Reed-Muller Codes*** $\mathcal{R}(r, m)$

The Reed-Muller code $\mathcal{R}(r, m)$ *of order* r *and length* 2^m *consists of all vectors* $\mathbf{f}$ *associated with all Boolean functions* f *that are polynomials of degree less than or equal to* r *in* m *variables [38].*

For example an $\mathcal{R}(2, 4)$ code would consist of $\mathbf{f} = a_0 + a_1\mathbf{v_1} + \ldots + a_4\mathbf{v_4} + a_{12}\mathbf{v_1v_2} + \ldots + a_{34}\mathbf{v_3v_4}$. In an $\mathcal{R}(r, m)$ code, the m vectors $\mathbf{v_1}, \ldots, \mathbf{v_m}$ are formed by considering an $n \times 2^m$ matrix with $\mathbf{v_i}$ as rows, and filling each of the 2^m columns with a different m-tuple. Combining all the columns results in all possible m-tuples. An $\mathcal{R}(r, m)$ RM-code has minimum distance $d_{\min} = 2^{m-r}$ [38].

The above information leads to the actual encoding process. Consider an $\mathcal{R}(r, m)$ RM code and a message vector $\mathbf{s} = (s_1, s_2, \ldots, s_m, s_{12}, \ldots, s_{m-r+1\ldots m-1m})$. Take

the vector $\mathbf{v_i}$ and form a generator matrix:

$$\mathbf{G} = \left[\begin{array}{c} \mathbf{1} \\ \hline \mathbf{v_1} \\ \vdots \\ \mathbf{v_m} \\ \hline \mathbf{v_1 v_2} \\ \vdots \\ \mathbf{v_{m-1} v_m} \\ \hline \vdots \\ \mathbf{v_{m-r+1} \cdots v_{m-1} v_m} \end{array}\right] = \left[\begin{array}{c} \mathbf{G}_0 \\ \hline \mathbf{G}_1 \\ \hline \mathbf{G}_2 \\ \hline \vdots \\ \hline \mathbf{G}_r \end{array}\right] \tag{2.3}$$

The message vector s can be grouped into order groups as: $\mathbf{s}_0 = s_0$, $\mathbf{s}_1 = (s_1, s_2, \ldots, s_m)$, $\mathbf{s}_2 = (s_{12}, \ldots, s_{m-1m})$, $\ldots$, $\mathbf{s}_r = (s_{12\ldots r}, \ldots, s_{m-r+1\ldots m-1m})$. Then the encoding is the following simple operation [38]:

$$\begin{aligned} \mathbf{c} &= (c_0, c_1, \ldots, c_{2^m-1}) \\ &= [\mathbf{s}_0|\mathbf{s}_2|\ldots|\mathbf{s}_r] \left[\begin{array}{c} \mathbf{G}_0 \\ \hline \mathbf{G}_1 \\ \hline \mathbf{G}_2 \\ \hline \vdots \\ \hline \mathbf{G}_r \end{array}\right] \end{aligned} \tag{2.4}$$

The decoding process was discovered by Reed and is called the Reed Decoding Algorithm. This is a majority logic technique which happens to be the maximum likelihood method. Before discussing the decoding technique, a few definitions will be introduced. Consider a codeword $(c_0c_1c_2c_3c_4) = 10011$, this vector is the incidence vector for the points $\{P_0, P_3, P_4\}$ since there is a '1' in bit positions 0,3 and 4. P_i is equivalent to the one's complement of the binary version of the integer i ($P_1 = (1000)$) . Define a set $I = \{1, \ldots, m\}$ which contains the

indices of the first order basis vectors. The higher order vector indices can be formed using the indices in I ($\mathbf{v_1 v_2}$ corresponds to $\{1, 2\}$). The entire decoding process for an $\mathcal{R}(r, m)$ RM code with a received word $\mathbf{r}$ goes as follows [38]:

- **Step 1:** Start with the message bits that have the highest order, r, and let $j = r$ and $\hat{\mathbf{c}}$ be the received vector.
- **Step 2:** For all the $\begin{pmatrix} m \\ j \end{pmatrix}$ of the order j message bits perform:
  1. Pick message bit $s_{i_1 i_2 \ldots i_j}$
  2. Consider associated incidence vector $\mathbf{v_{i_1} v_{i_2}} \ldots \mathbf{v_{i_j}}$
  3. Let $S = \{P_{i_1}, P_{i_2}, \ldots, P_{i_j}\}$
  4. Let $I \backslash \{i_1, i_2, \ldots, i_j\} = \{k_1, k_2, \ldots, k_{m-j}\}$ and $T = \{P_{k_1}, P_{k_2}, \ldots P_{k_{m-j}}\}$
  5. Form k_{m-j} translations of T by translating $\{P_{i_1}, P_{i_2}, \ldots, P_{i_j}\}$ by each $P_{k_l},\ l \in \{1, \ldots, m-j\}$
  6. Take one of the translations $P_{t_1}, \ldots, P_{t_j}$, and form first message bit estimate $\hat{s}^{(1)}_{i_1 i_2 \ldots i_j} = \hat{c}_{t_1} + \hat{c}_{t_2} + \ldots + \hat{c}_{t_j}$ (all in GF(2)). Do this for all the translations.
  7. Let $\hat{s}_{i_1 i_2 \ldots i_j} = \text{maj}\{\hat{s}^{(1)}_{i_1 i_2 \ldots i_j}, \ldots, \hat{s}^{(k_{m-j})}_{i_1 i_2 \ldots i_j}\}$ (where maj represents taking the most often occurring value of the argument).
  8. Go to step 3.
- **Step 3:**
  - If all message bits of order j have been estimated then form $\hat{\mathbf{s}}_r$ from the estimates of the message bits of order j and let $j = j - 1$.
    - If $j >= 0$, then let $\hat{\mathbf{c}} = \hat{\mathbf{c}} - \hat{\mathbf{s}}_r \mathbf{G}_r$. Go to step 2.

        * Else decoding process is done.

    - Else go to step 2.

Please see [38] for more detailed theory on Reed-Muller codes.

CHAPTER 3

RELATED WORK

There has been a variety of work done in assuring network reliability, network security, and network integrity which is related to the research in this thesis. There are numerous works on these topics, and for realistic length constraints only the most closely related works are referenced. In the next few sections, brief details are mentioned about the research related to each of the authors contributions.

3.1 Network Reliability

There is a vast array of work done on routing in multiple path channels. In [15], the authors introduce a method to find maximally disjoint paths. We assume that the independent paths in our setup were found in a similar fashion. In their paper they transmit information down these paths assuming that they have the same security level. In [35], the authors discuss a multiple path routing technique over equally reliable links. They devise a method to optimally allocate channel coded packets down the paths by maximizing the success probability. The problem with this method is the low network efficiency. In another paper [36], the authors remove the assumption that the paths have the same performance. They determine an approximation of the success probability for the network and then allocate packets down each path as to maximize this function. They do not consider the question of how much redundancy to add to the original message, and just assume that it is a pre-determined number.

Redundancy is vital in erasure channels since it allows perfect decoding even with some erasures. Maximum distance separable (MDS) codes are important examples of erasure codes since they have the property that only a set the size of the input symbols is required to perfectly decode the message. Our algorithms are based on the structure of MDS codes. In [25], the authors also use redundancy in the form of an erasure code which is based off of Rabin's algorithm [26]. A widely used MDS code is the Reed-Solomon code [39].

[6] develops a multipath routing algorithm between a single source and destination which ranks each path based on three metrics: energy, delay and reliability. The paths between the source and destination are found by sending simple query and reply messages. They also suggest the use of forward error-correction (FEC), in particular block codes, to account for packet losses. This work does not focus on FEC and hence no error-correction code is specified.

In [25], the authors suggest a protocol for secure message transmission in a multiple path channel. They generate a method to assess the "trustworthiness" of each path based on previous behavior, and the paths which are trusted above a certain threshold are put in the active path set (APS). The "trustworthiness" level of a path is directly linked to the probability of successful transmission. The paths in the APS are then used to transport messages.

Wireless networks such as mobile ad hoc networks or sensor networks have frequent changes in topology due to link failures, physical obstructions, network intrusions, etc. Dependability on these sort of networks requires dynamic algorithms which quickly determine routing, redundancy, etc. for deviations in the network. In [9], the authors introduce algorithms to dynamically determine the redundancy and message dispersion among the path.

3.2 Network Security and Reliability

As mentioned earlier, in 1975 Aaron Wyner introduced the wiretap channel in [40]. This initial work has lead to a large amount of research related to wiretap channels. Ozarow and Wyner discuss system design to protect a wiretap channel from revealing information to an intruder in [23].

The authors in [32] show the maximum possible rate to achieve perfect secrecy using MDS codes on the wiretap channel. The channels between Bob and Eve are assumed to each have a specified number of erasures, unlike the work in this document which assumes a probabilistic channel. The authors introduce a nested coding scheme based off of MDS codes which achieves secrecy capacity.

The application of LDPC codes to wiretap channels are introduced in [34]. These authors prove that capacity achieving codes can attain secrecy capacity for any wiretap channel. For the binary erasure and binary symmetric channels, specific codes are shown which result in perfect secrecy. Our research on secrecy is an extension of this work.

McEliece introduces the idea of using algebraic codes for security in [22]. He generates a public-key cryptosystem by permuting the generator matrix of an algebraic code (Goppa Code) and using the permutation as the public key. It is shown that the technique is hard to decode if one only knows the public key. The method does not have perfect secrecy.

In [21], the authors introduce a block cipher code called High Diffusion (HD) cipher which corrects errors and encrypts a message simultaneously. The setup of the codes are introduced, though not shown how to be formed. Security

capabilities are analyzed with respect to linear and differential cryptanalysis as well as the square attack. Error-correction capabilities are derived and these codes are shown to be MDS. These codes do not attain perfect secrecy but most realizable encryption methods do not.

3.3 Detection of Malicious Nodes in Network

In [14], the authors first introduce the well known Byzantine Generals problem in which loyal and disloyal generals communicate with one another through a messenger. The primary focus of this paper is for reliability in computer systems, showing when reliable communication can occur, i.e. when traitors can be detected.

The authors in [5], describe an information theoretic method to detect Byzantine adversaries using random network coding. The method adds overhead to the messages in the form of hash value and shows how malicious packets can be detected, but does not find the origin of these malicious packets.

A secure routing protocol which is robust to malicious nodes is developed in [1]. Our method, on the other hand, assumes that a standard encryption algorithm is used and determines the exact location of the adversary. The location of malicious nodes is important in determining which nodes can be trusted or the level to which each node can be trusted. If a node is constantly being hacked into by an illegitimate user, then for the sake of the integrity of the network, this node should be thrown out/not used.

The location of nodes is found in [37], but their method requires the interior

nodes to converse and monitor suspected malicious nodes. In our method the only job of the nodes is to forward packets.

[24] suggest the use of the secure traceroute protocol to detect and locate routers which fail to correctly forward traffic. They assume that the malicious routers misdirect the data. We assume that adversary nodes direct the data correctly but with fake packets, hence not allowing the destination to receive desired data.

CHAPTER 4

RELIABILITY FOR MULTIPLE PATH NETWORK

4.1 System Model for reliability

We assume that we have a source node who wishes to convey information to a specific destination node. These two nodes are separated by multiple wireless paths, each acting like an independent erasure channel. An erasure on a particular path could be caused by a malicious node which is stealing or tampering with data. The data is encrypted with a standard encryption algorithm which allows the destination to determine if an adversary has corrupted a path. Hence each path is associated with a pre-determined erasure probability which represents its "trustworthiness" level. Let path i have a probability of success p_i. Without loss of generality, we assume that $p_1 \geq p_2 \geq \ldots \geq p_N$ given that there are N paths. The transmitting node knows the statistics of the channels between itself and the receiving node, and hopes the message can be decoded within probability of success p^*. The source node needs an algorithm that increases the length of the message and allocates symbols down the paths so that p^* is achieved. Increasing redundancy reduces throughput and increases bandwidth thus it is important to find the minimum necessary redundancy. Since the security level on the paths changes over time, it is desirable that this algorithm is dynamic.

4.1.1 Channel Probability Distribution

Each path i out of the N paths acts as an erasure channel with erasure probability $1-p_i$. An erasure channel with erasure probability p is one in which each symbol is erased with probability p [4]. In our model we assume that the destination node either receives all the symbols down a particular path or receives nothing. This assumption is feasible since data cannot be trusted if it has been tampered with by an adversary. Figure 4.1 shows the network model.

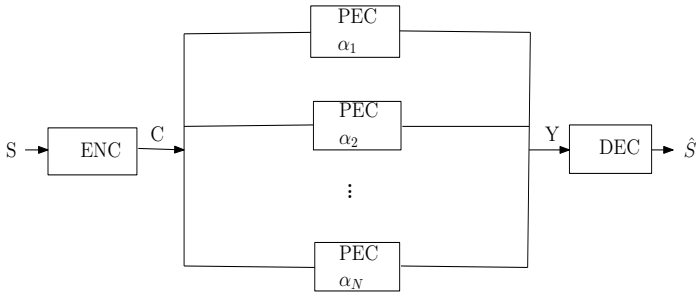

Figure 4.1: Multiple path network receiver sees each path as a packet erasure channel

A codeword of length n is dispersed among the N paths, with path i receiving f_i symbols forming a vector $\mathbf{f} = \{f_1, f_2, \ldots, f_N\}$. This means that $\sum_{i=1}^{N} f_i = n$. We can form a vector $\mathbf{s}$ of length N composed of '1's and '0's with a '1' in spot i representing a non-erasure on path i and a '0' representing an erasure. If we use an MDS code, then with complete certainty the message can be decoded if k out of n symbols are received. Thus, if we construct a matrix S composed of all possible combinations of '0's and '1's our probability of successful decoding for a specified $\mathbf{f}$ becomes:

$$P_{\text{success}}(\mathbf{f}) = \sum_{\mathbf{s}\in S} \prod_{i=1}^{N} p_i^{s_i}(1-p_i)^{1-s_i} u(\mathbf{s}\cdot\mathbf{f} - k) \tag{4.1}$$

where $u(\cdot)$ represents the unit step function. The matrix S is filled with all possible vectors $\mathbf{s}$ implying that there are 2^N rows. Hence if we run through all the

rows in S to calculate P_{success}, this results in an exponential running time with relation to the number of paths. If the number of paths is not too large and one wishes for extreme precision, this calculation is not too arduous. Otherwise it is necessary to find a close alternative which we will discuss next.

Each path has a Bernoulli distribution since it receives the exported symbols (1) with probability p_i or an erasure (0) with probability $1 - p_i$. This implies that the sum of multiple transmissions across path i has a Binomial distribution. For large sample sizes, the Binomial distribution can be approximated using the Gaussian distribution, and the authors in [36] suggest this approximation to calculate the probability of success.

We consider a random variable that represents the average number of successful transmission attempts out of f_i on path i. Using the observation mentioned above, this random variable is distributed binomially. We approximate this random variable using a Gaussian distribution and it is known that the distribution of the sum of independent Gaussian random variables is also Gaussian. This indicates that the distribution of the sum of the paths is also Gaussian. Hence, the approximation on each path is Gaussian distribution $\sim \mathcal{N}\left(f_i p_i, f_i^2 p_i(1-p_i)\right)$, and the distribution of the sum of the paths becomes $\sim \mathcal{N}\left(\sum_{i=1}^{N} f_i p_i, \sum_{i=1}^{N} f_i^2 p_i(1-p_i)\right)$. Then, integrating this distribution over the scenario that k or more symbols are received in total yields a probability of success function:

$$P_{\text{success}}(\mathbf{f}) \approx \frac{1}{2} + \frac{1}{2}\text{erf}\left(\frac{\sum_{i=1}^{N} f_i p_i - k + \frac{1}{2}}{\sqrt{2\sum_{i=1}^{N} f_i^2 p_i(1-p_i)}}\right) \tag{4.2}$$

where $\text{erf}(x) = \frac{2}{\pi}\int_0^x e^{-y^2} dy$. It can be seen that computation of this success probability approximation is significantly simpler than that of the true success probability.

The source node wants to assure that his message is received intact with probability p^* at the destination. The success probability functions along with the security level of each path can be used to determine message redundancy and symbol allocation. The original data is k symbols long, and if an MDS code is used to extend the k to n symbols, there are a few observations we can make. As mentioned earlier, any of the k of the n symbols can decode the original message. Let $\gamma = \frac{n}{k}$ represent the redundancy ratio. Below are some initial observations:

- $p_1 \geq p_2 \geq \ldots \geq p_N$ implies that $f_1 \geq f_2 \geq \ldots \geq f_N$
- If $\gamma \geq N$ then an optimal approach is to send $f_1, f_2, \ldots, f_N \geq k$
- It is not optimal to send more than k symbols down any path
- If $p_1 \geq p^*$ then k symbols should be sent down path 1. In this case $\gamma = 1$

These observations have led to developing two optimal redundancy and symbol allocation algorithms described in [9], and presented in a later section.

4.2 Algorithms

4.2.1 Optimal Symbol Allocation and Minimum Redundancy for $N = 3$

To simplify our analysis, in [9], we began with the situation with $N = 3$ paths between the source and destination. A brute force method can be used to find optimal symbol allocation and minimum redundancy. Assume that the desired

success probability is p^* and that paths 1, 2, 3 have erasure probabilities $1-p_1, 1-p_2, 1-p_3$. The optimal symbol allocation vector $\mathbf{f}$ and redundancy ratio γ are as follows:

- If $p_1 + p_2 - p_1p_2 < p^* \leq p_1 + p_2 + p_3 - p_1p_2 - p_2p_3 - p_1p_3 + p_1p_2p_3$
  $\Rightarrow \gamma_{\min} = 3$ and $f_1, f_2, f_3 = k$
- If $\max\{p_1p_2 + p_2p_3 + p_1p_3 - 2p_1p_2p_3, p_1\} < p^* \leq p_1 + p_2 - p_1p_2$
  $\Rightarrow \gamma_{\min} = 2$ and $f_1, f_2 = k, \quad f_3 = 0$
- If $p_1 < p_1p_2 + p_2p_3 + p_1p_3 - 2p_1p_2p_3$ and $p_1 < p^* \leq p_1p_2 + p_2p_3 + p_1p_3 - 2p_1p_2p_3$
  $\Rightarrow \gamma_{\min} = \frac{3}{2}$ and $f_1, f_2, f_3 = \frac{k}{2}$
- If $0 < p^* \leq p_1$
  $\Rightarrow \gamma_{\min} = 1$ and $f_1 = k, f_2, f_3 = 0$

Fig. 4.2 shows a plot of minimum redundancy versus the target success probability for $p_1 \geq p_1p_2 + p_2p_3 + p_1p_3 - 2p_1p_2p_3$. Fig 4.3 shows the case where

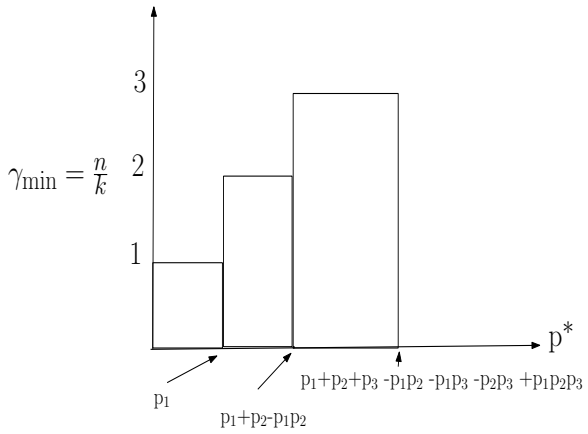

Figure 4.2: Minimum Redundancy for $N = 3$ when $p_1 \geq p_1p_2 + p_2p_3 + p_1p_3 - 2p_1p_2p_3$

$p_1 < p_1p_2 + p_2p_3 + p_1p_3 - 2p_1p_2p_3$. Results in figures 4.2 and 4.3 represent all

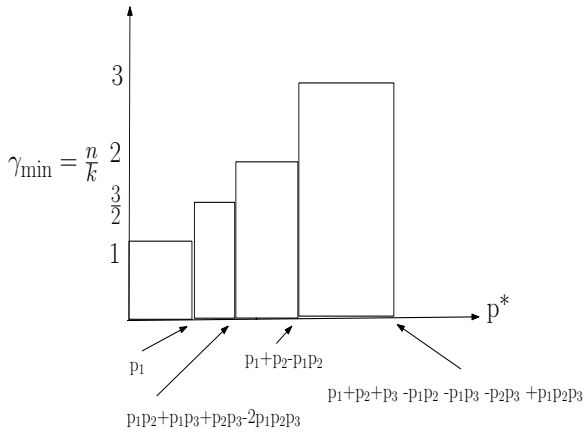

Figure 4.3: Minimum Redundancy for $N = 3$ when $p_1 < p_1p_2 + p_2p_3 + p_1p_3 - 2p_1p_2p_3$

possible combinations for the symbols across the paths. Ordering the paths simplifies the analysis so assume that $p_1 \geq p_2 \geq \ldots \geq p_N$. We know that the redundancy γ is such that $1 \leq \gamma \leq 3$. Considering $\gamma = 1$ then there are three different possible symbol allocations: $\mathbf{f} = k, 0, 0$, $\mathbf{f} = \frac{k}{2}, \frac{k}{2}, 0$, and $\mathbf{f} = \frac{k}{3}, \frac{k}{3}, \frac{k}{3}$. It is known that a success occurs if the sum of the received symbols across all the paths is greater than or equal to k and since $p_1 \geq p_2 \geq p_3$, this implies that we only need to look at three situations: 1) when only the first path is needed for success, 2) when the first and second path are needed for success, and 3) when all three paths are needed for success. The reason for having only three symbol allocations is based on the following observation. Consider the case when only two paths are required and assume that $\frac{k}{2}$ symbols are sent down each path, then it can be seen that probability of success is exactly the same as when $\frac{(i-1)k}{i}$ symbols were sent down the first path and $\frac{k}{i}$ symbols sent down the second path. Note that p_1, p_1p_2, and $p_1p_2p_3$ are probabilities of success for scenarios 1, 2 and 3 respectively. So clearly in this scenario, $\mathbf{f} = k, 0, 0$ yields the highest success probability. Continuing in this manner for different possible values of γ we obtain the results shown above. Unfortunately, there are not always going

to be $N = 3$ paths, so next we devise a method to take care of the instance where there are an arbitrary amount of paths.

4.2.2 Algorithms for Arbitrary Number of Paths

Following [9], the results can be extended to an arbitrary number of paths. Since the APS set size varies with path security, it is important to design an algorithm that dynamically determines all parameters. A closed form expression for the network's probability of success is difficult to obtain, thus, we developed heuristic algorithms which determine parameters. One of these algorithms has exponential running time and is called Minimum Redundancy Algorithm in Exponential Time or MRAET. Though MRAET has an exponential running time, we prove that in several special cases it's optimal. In cases when exponential running time is unacceptable, we introduce an algorithm with polynomial running time. We call it Minimum Redundancy Algorithm in Polynomial Time (MRAPT).

Both MRAET and MRAPT algorithms consist of two steps. Part 1 is to reduce dimensionality of the space due to the fact that the search redundancy/symbol dispersion space is extremely large. This reduction results in a shorter search time required to determine the desired parameters. Given p^*, part 1 first assigns the symbol allocation vector $\mathbf{f} = [k, 0, \ldots, 0]$. $\mathbf{f}$ is then plugged into probability expression (Eq. 4.1, Eq. 4.2 in MRAET, MRAPT respectively) and compared with p^*. If the value is greater than or equal to p^* the algorithm moves on to part 2. Otherwise it sets $\mathbf{f} = [k, k, 0, \ldots, 0]$ and repeats the procedure. Part 1 is exited when $\mathbf{f}$ is found that generates a probability greater than or equal to p^*.

Part 2 picks off where part 1 left off, and it checks possible symbol allocations which have a smaller redundancy than that found in part 1 and also result in a success probability greater than p^*. The algorithm terminates when there are no options left. Taking the value $j = \gamma_{\min}$ from the first part of the algorithm, the second part of the algorithm starts with the case were the first $j-2$ paths have k symbols assigned to them. The algorithm then steps through different combinations for the rest of the $N-(j-2)$ paths to see if there is a combination which results in a lower redundancy and also meets the target success probability requirement. An example of a combination which part two of the algorithm will attempt would be $f_1, f_2, \ldots, f_{j-2} = k, f_{j-1}, f_j, f_{j+1} = \frac{k}{2}, f_{j+2}, \ldots, f_N = 0$. Some new notation will be mentioned before introducing both parts of the algorithm.

$$P^A_{\text{success}} = \sum_{\mathbf{s}\in A} \prod_{i=1}^{N} p_i^{s_i}(1-p_i)^{1-s_i}$$

where A is some submatrix of S (S is a matrix filled with all possible length N binary vectors, with the first row being the all zero vector and the last row being the all one vector). $\sum_{\mathbf{s}\in A}$ represents a sum which begins with the first row vector of A and terminates with the vector that is the last row of A. Let,

$$Z^j_{(i,s)} = \{z \in \{1, \ldots, 2^{N-(j-2)}\} \mid \sum_{l=j-1}^{j-2+i} S_{z,l} \geq s\}$$

for some integers s, i, j. Where $S_{z,l}$ represents the element of S in the z^{th} row and l^{th} column. Let $S_{((i:j),(1:l))}$ ($i \geq j$ and $l \geq 1$) represent a submatrix of S composed of all the rows $i, i+1, \ldots, j$ of S up to the column l.

Minimum Redundancy Algorithm in Exponential Time

Part 1:

Step 1: Assign $j = 1$ and go to *step 2*.

Step 2: Let $A = S_{((2^{N-j}+1:2^N),(1:N))}$ and go to *step 3*.

Step 3: Calculate P^A_{success}

If $P^A_{\textbf{success}} \geq p^*$

$f_1, \ldots, f_j = k, f_{j+1}, \ldots, f_N = 0$

$\gamma_{\min} = j, P_{\text{temp}} = P^A_{\text{success}}$

Go to *Part 2* of the algorithm

else let $j = j + 1$

if $j > N$ move on to *Part 2*

else return to *Step 2*

If $j < 2$ then we have an optimal allocation and we are done. Otherwise:

Part 2:

Let $i = 2$ and $j = \gamma_{\min}$

Step 1: Let $i = i + 1$

if $i > N$ or $j - 2 + i > N$ then terminate Part 2

else go to *Step 2*

Step 2: Let $s = 2$ and go to *step 3*

Step 3:

if $j - 2 + \frac{i}{s} \leq j$

Let A denote the subset of matrix S composed of rows whose indices are in $Z^j_{(i,s)}$ (where $Z^j_{(i,s)}$ is the set defined above) followed with rows $(2^{N-(j-2)} + 1 : 2^N)$ of the matrix S, or

$$A = \begin{bmatrix} S_{\left((Z^j_{(i,s)}),(1:N)\right)} \\ S_{\left((2^{N-(j-2)}+1:2^N),(1:N)\right)} \end{bmatrix}$$

Go to *step 4*

else Go to *step 6*

Step 4: Calculate P^A_{success}

if ($P^A_{\text{success}} \geq p^*$ with $j - 2 + \frac{i}{s} < j$) or ($j - 2 + \frac{i}{s} = j$ and $P_{\text{temp}} < P^A_{\text{success}}$)

Go to *step 5*

else Go to *step 6*

Step 5: Let $P_{\text{temp}} = P^A_{\text{success}}$, $\gamma_{\min} = j - 2 + \frac{i}{s}$, and

$f_1, \ldots, f_{j-2} = k$

$f_{j-1}, \ldots f_{j-2+i} = \frac{k}{s}$

$f_{j-1+i}, \ldots, f_N = 0$

Go to *step 6*

Step 6: Let $s = s + 1$

if $s > i$ Go to *step 1*

else Go to *step 3*

This algorithm is optimal for several of cases. One case is when we have $N = 3$ paths.

Theorem 3 ***MRAET is optimal when*** $N = 3$

Proof: After Part 1 of the algorithm, we have 3 options for j, $j = 1, 2, 3$.

If $j = 1$, *then the algorithm terminates after part 1 since* $j < 2$. *We are left with* $\gamma_{\min} = 1 \Rightarrow n = k$

Thus $f_1 = k$, $f_2 = f_3 = 0$ *which is optimal.*

If $j = 2$, *then*

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

and $P_{temp} = p_1 + p_2 - p_1 p_2$

The algorithm then steps into part 2. It starts and ends with the scenario $i = 3, s = 2$ *since* $N = 3$. *It first checks is* $j - 2 + \frac{i}{s} \leq \gamma_{\min} = j$. *If so, then it searches through*

$$S = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

rows $(1, \ldots, 2^{N-(j-2)}) = (1, \ldots, 8)$ *such that the columns* $(j-1, \ldots, j-2+i) = (1, \ldots, 3)$

sum to greater than or equal to $s = 2$. *Then*

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}.$$

Then since $j - 2 + \frac{i}{s} = \frac{3}{2} < j = 2$, *MRAET checks if* $P_{success} = \sum_{\mathbf{s}\in A} \prod_{l=1}^{3} p_i^{s_i}(1-p_i)^{1-s_i} = p_1p_2 + p_2p_1 + p_1p_3 - 2p_1p_2p_3 \geq p^*$. *If so, then* $\gamma_{\min} = \frac{3}{2}$ *and* $f_1, f_2, f_3 = \frac{k}{s} = \frac{k}{2}$, *otherwise* $\gamma_{\min} = 2$ *and* $f_1 = f_2 = k, f_3 = 0$.

Lastly, if $j = 3$ *the algorithm stops before part 2 because* $j - 2 + 3 > 3$. *Thus,* $f_1 = f_2 = f_3 = k$, *which is the same allocation as we had above.* □

Next, we prove a theorem to help us show another optimal case.

Theorem 4 ***Suppose we have*** k ***symbols allocated to paths*** $1, \ldots, i-1$ ***and*** 0 ***symbols allocated to the remaining of the*** N ***paths, resulting with probability of success*** $\hat{p}_{i-1}$. ***Then, if we let path*** i ***have*** k ***symbols, the probability of success is***

$$\hat{p}_i = \hat{p}_{i-1} + p_i - \hat{p}_{i-1}p_i \tag{4.3}$$

Proof: By induction on the integer i.

Base Case: $i = 2$

We have $\hat{p}_{i-1} = \hat{p}_1 = p_1$, *since we only have success if path 1 succeeds. If we let path 2 have k symbols, then we have a success solely if path 1 succeeds, if path 2 is the only successful one, or if they both succeed. This is equivalent to:*

$$\hat{p}_2 = \hat{p}_1p_2 + \hat{p}_1(1-p_2) + (1-\hat{p}_1)p_2 = \hat{p}_1 + p_2 - \hat{p}_1p_2$$

Inductive Hypothesis: Suppose Eq. 4.3 holds $\forall i \leq m-1$

Inductive Step: Let $i = m$ *. Then by inductive hypothesis we know that* $\hat{p}_{m-1}$ *is the probability of success for the first* $m-1$ *paths having* k *symbols and the rest having* 0*. We can think of* $\hat{p}_{m-1}$ *as being the probability of success for one super path. Thus, if we let the* m^{th} *path have* k *symbols, then we have success if only the super path is successful, the* m^{th} *path is the only successful one, or if they are both successful. That is:*

$$\begin{aligned} \hat{p}_m &= \hat{p}_{m-1}p_m + \hat{p}_{m-1}(1-p_m) + (1-\hat{p}_{m-1})p_m \\ &= \hat{p}_{m-1} + p_m - \hat{p}_{m-1}p_m \end{aligned}$$

Hence the result holds $\forall i \in 2, \ldots N$*.* □

Theorem 5 ***MRAET is optimal when*** $j = N$

Proof: If j=N, then we know that A is equal to S, excluding the all zero first row, $P^{A}_{success} \geq p^{*}$*.*

By Thm. 4 we know that the probability of success for the first $N-2$ *paths having* k *symbols and the rest having* 0 *is:*

$$\hat{p}_{N-2} = \hat{p}_{N-3} + p_{N-2} - \hat{p}_{N-3}p_{N-2} < p^{*}$$

Thus, if we treat the first $N-2$ *paths as one super path with probability* $p_{super} = \hat{p}_{N-2}$*, then the current problem can be mapped to the case where* $N = j = 3$ *since super path is path 1,* $N-1$ *is path 2, and N is our third path.* □

Corollary 1 ***MRAET is optimal when*** $j = N-1$

This results follows from the theorem above, since $j = N-1$ *can be mapped to the case where* $N = 3$ *and* $j = 2$*.*

Minimum Redundancy Algorithm in Polynomial Time (MRAPT)

For MRAPT we proceed similarly to MRAET but we use the success probability approximation, eqn. 4.2. This means that there is no need to search through the matrix S and further spending exponential running time by calculating the true probability of error. Hence, Part 2 for the MRAPT algorithm excludes the S matrix search, and when it terminates a vector $\mathbf{f}$ is returned. Next we will analyze the running time of these algorithms.

Algorithm Running Time Analysis

We begin by analyzing the MRAET algorithm. We assume that S is computer offline. The matrix S has 2^N rows and the worst case scenario is if we have to search through the entire matrix. Using a common logarithmic search algorithm like the binary search mentioned in [3], the running time of this process becomes $O(\log_2(2^N)) = O(N)$. The worse case running time ($A = S$) calculating the probability of success is $O(2^N)$. Thus, inside the loop running time becomes $O(N + 2^N)$. It can be seen that the outer loop takes $< N$ iterations, hence the total worst case running time of part 1 is $O(N(N + 2^N))$. Part 2 has one more outside loop of N iterations but the analysis is pretty much identical. This implies that MRAET's total worst scenario running time is $O(N^2(N + 2^N))$. Without a doubt MRAET is an exponential time algorithm.

The running time of MRAPT is significantly reduced since it is not necessary to traverse matrix S or to calculate the true success probability. Using eqn. 4.2 as the probability of success function, it is necessary to calculate the mean and variance of the Gaussian distribution $\sim \mathcal{N}\left(\sum_{i=1}^{N} f_i p_i, \sum_{i=1}^{N} f_i^2 p_i(1-p_i)\right)$. . Assuming

that the values $p_i(1-p_i)$ are saved, the mean and variance calculations each take $O(2N)$ iterations which is the worst case running time. Similar to the analysis for MRAET, the second part overbears the running time with its two loops resulting in $O(N^2(4N)) = O(N^3)$ running time for MRAPT. MRAPT thus runs in polynomial time with respect to the number of paths.

4.3 Application to Different Codes

Although these algorithms were developed for MDS codes, we can also apply them to LT and Raptor codes. These codes are almost MDS codes implying that instead of needing to receive exactly k symbols to decode the message, around $k(1+\varepsilon)$ are needed for decoding. Although this increase in overhead is not desirable, these codes have several advantages over the classic MDS codes. As mentioned above, both LT and Raptor codes have smaller encoding/decoding time than MDS codes. There are many instances where the network efficiency is less important than the cost of the encoder/decoder. Another advantage of these codes is that they are fountain codes, meaning that they are rateless. This is extremely important because in the situation where the decoder does not receive enough output symbols to decipher the transmitted message, the encoder can very simply send some more data independently of the output symbols which were received. An RS code, for example, would either require knowledge of the exact locations of missing symbols or would have to re-encode and retransmit the original message.

4.3.1 LT Code Implementation

Theoretically in an LT code, with probability of $1 - \delta$, all k input symbols can be recovered from a set of $k(1 + \varepsilon)$, where overhead ε depends on δ. To implement LT codes, we first decide on the value of δ which is used to determine the parameter $R = c \log(k/\delta) \sqrt{k}$. The next step is to establish the overhead ε. We use the Robust Solition distribution mentioned in an earlier section for the output symbol degree distribution. In [18], the authors show that using this distribution and to ensure a probability $1 - \delta$ of successful decoding, the total output symbol size should be on the order of $K = k + O(\sqrt{(k)} \ln^2(k/\delta))$. In particular, $K = k + \sum_{i=1}^{k/R-1} \frac{R}{i} + R \ln(R/\delta)$ implying that $\varepsilon = \frac{1}{k} \left(\sum_{i=1}^{k/R-1} \frac{R}{i} + R \ln(R/\delta) \right)$. Our algorithm is used to determine total redundancy needed to ensure that the probability of success is above a certain threshold. This means that our algorithm passes $1 - \delta$ and p^* as parameters and determines the symbol allocation/redundancy so that the calculated success probability is at least as large as p^*.

4.3.2 Raptor Code Implementation

We implement a raptor code using a regular LDPC code combined with an LT code. The first step is to determine the overhead of both codes. Based on the fraction of input symbols, δ, which we would like to decode using the inner LT code, we determine ε_{LT} which specifies the LT redundancy. Then, following the method used in [2], we let $\varepsilon_{\text{raptor}} = 2\varepsilon_{\text{LT}}$. Since $(1 + \varepsilon_{\text{raptor}}) = (1 + \varepsilon_{\text{LDPC}})(1 + \varepsilon_{\text{LT}})$, $\varepsilon_{\text{LDPC}}$ becomes

$$\varepsilon_{\text{LDPC}} = \frac{\varepsilon_{\text{raptor}}}{2 + \varepsilon_{\text{raptor}}}$$

Next, using the raptor overhead, or $k(1 + \varepsilon_{\text{raptor}})$, the total redundancy and symbol allocation are determined using either MRAET or MRAPT. Once the redundancy and the symbol allocation is specified, it is necessary to determine the amount of redundant check nodes of each code. Since the previous overheads represented the k out of n symbols similar to MDS codes, these redundancies no longer match the total code size. So the initial overhead used to determine the total amount of check nodes and symbol allocation was:

$$k' = k(1 + \varepsilon_{\text{LDPC}})(1 + \varepsilon_{\text{LT}}) \tag{4.4}$$

In order to determine the new redundancies, we keep in mind that we would like the ratio of the original redundant nodes of each code to be the same. We introduce two new constants which we would like to solve for: γ_{LT} and γ_{LDPC}. These constants represent the increase in the overhead for the LDPC and LT code to reach the size of the final codeword n. We have,

$$n = k(1 + \gamma_{\text{LT}}\varepsilon_{\text{LT}})(1 + \gamma_{\text{LDPC}}\varepsilon_{\text{LDPC}}) \tag{4.5}$$

Initially there are $r_{\text{iLDPC}} = k\varepsilon_{\text{LDPC}}$ and $r_{\text{iLT}} = k(\varepsilon_{\text{LT}} + \varepsilon_{\text{LT}}\varepsilon_{\text{LDPC}})$ redundant nodes for the LDPC and LT code respectively. After finding the total codeword size n from the algorithm, the LDPC code has $r_{\text{LDPC}} = k\gamma_{\text{LDPC}}\varepsilon_{\text{LDPC}}$ redundant symbols and the LT code has $r_{\text{LT}} = k(\gamma_{\text{LDPC}}\varepsilon_{\text{LT}} + \gamma_{\text{LT}}\gamma_{\text{LDPC}}\varepsilon_{\text{LT}}\varepsilon_{\text{LDPC}})$ redundant symbols. We solve for $\gamma_{\text{LT}}, \gamma_{\text{LDPC}}$ by setting the ratio of the final redundant symbols equal to that of the initial ones, or:

$$\begin{aligned} \frac{r_{\text{LT}}}{r_{\text{LDPC}}} &= \frac{r_{\text{iLT}}}{r_{\text{iLDPC}}} = \frac{\gamma_{\text{LDPC}}\varepsilon_{\text{LT}} + \gamma_{\text{LT}}\gamma_{\text{LDPC}}\varepsilon_{\text{LT}}\varepsilon_{\text{LDPC}}}{\gamma_{\text{LDPC}}\varepsilon_{\text{LDPC}}} \\ &= \frac{\epsilon_{\text{LT}} + \epsilon_{\text{LT}}\epsilon_{\text{LDPC}}}{\epsilon_{\text{LDPC}}} \end{aligned} \tag{4.6}$$

Using equation 4.5 we obtain:

$$\gamma_{\text{LT}} = \frac{\frac{n}{k(1+\gamma_{\text{LDPC}}\varepsilon_{\text{LT}})} - 1}{\varepsilon_{\text{LT}}} \tag{4.7}$$

From eqn's 4.6 and 4.7 we acquire:

$$\gamma_{\text{LDPC}} = \frac{\frac{n}{k} - 1}{(\varepsilon_{\text{LDPC}} + \varepsilon_{\text{LT}} + \varepsilon_{\text{LDPC}}\varepsilon_{\text{LT}})} \tag{4.8}$$

Hence the LDPC code has $k\gamma_{\text{LDPC}}\varepsilon_{\text{LDPC}}$ redundant nodes. We use a regular Gallagher code, meaning that the $n - k \times n$ parity check matrix has three '1's per column. The belief propagation algorithm is used for decoding.

To determine the degree distribution of the LT code, we use an idea discussed in [33] and [17]. In [33], the authors discuss the design of the LT degree distribution for finite length raptor codes. Based on keeping the expected ripple size of the LT code at $c\sqrt{k(1-x)}$, they determine that the LT degree distribution should meet this inequality:

$$\Omega'(x) \geq \frac{-\ln\left(1 - c\sqrt{\frac{1-x}{k}}\right)}{1 + \varepsilon_{\text{raptor}}} \tag{4.9}$$

with $x \in [0, 1-\delta]$ and $\delta > c/\sqrt{(k)}$. The degree distribution can be solved as in [33], by discretizing x in the interval $[0, 1-\delta]$ and forming a linear program that must meet the constraints in 4.9. In particular, the linear program is a minimization problem where the expected degree, $\Omega'(1)$, is minimized, or:

$$\begin{aligned} \min_{\Omega} \quad & \Omega'(1) \\ \text{s.t} \quad & \Omega'(x) \geq \frac{-\ln\left(1 - c\sqrt{\frac{1-x}{k}}\right)}{1 + \varepsilon_{\text{raptor}}} \\ & \sum_{i=1}^{D} \Omega_i = 1 \\ & x \in [0, 1-\delta] \end{aligned} \tag{4.10}$$

where D represents the maximum degree. Using the overhead determined above for the LT code, we use this degree distribution in the final step of the encoding process to add redundancy to the LDPC code and forming the final codeword.

4.4 Simulations

All of our simulations are run over numerous Monte Carlo runs to accurately depict performance. We measure the performance of our algorithms by first comparing the success probability of MRAET and MRAPT in Fig. 4.4. Figure 4.4 also shows the desired probability level as well as the success probability approximation to allow for a full comparison. The parameters we use for this plot are $N = 7,\ k = 4,\ \mathbf{p} = [0.8000, 0.5901, 0.5338, 0.5261, 0.5203,$
$0.5107, 0.5000]^T$. MRAET algorithm performs very well and achieves a success probability that is higher or equal to p^*. MRAPT on the other hand seems to oscillate around p^*, but typically stays above p^*. This makes MRAPT particularly practical since it determines symbol allocation and redundancy in polynomial time. Comparing MRAPT to the approximation we used for the probability of success, it can be seen that this approximation seems to be off by a constant offset from the true performance.

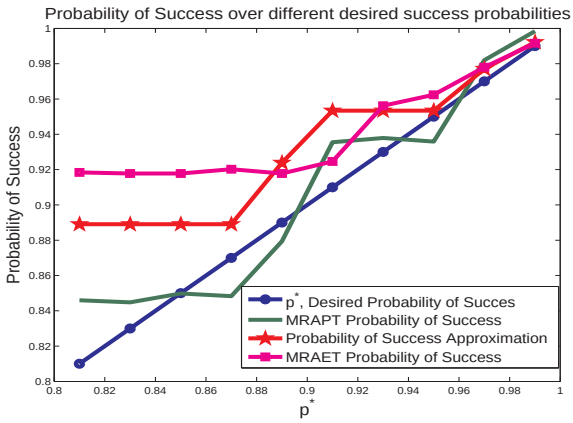

Figure 4.4: Probability of Success of MRAPT and MRAET

Using the same parameters as used for 4.4, Fig. 4.5 shows a comparison of the redundancy ratios of MRAET and MRAPT vs. p^*. Due to the sub-

optimality of MRAPT, the redundancy ratio of MRAPT is slightly higher than that of MRAET. The gap between the redundancy ratios is fairly small until p^* gets to be around 0.98 and there appears to be a large jump for the redundancy of MRAPT.

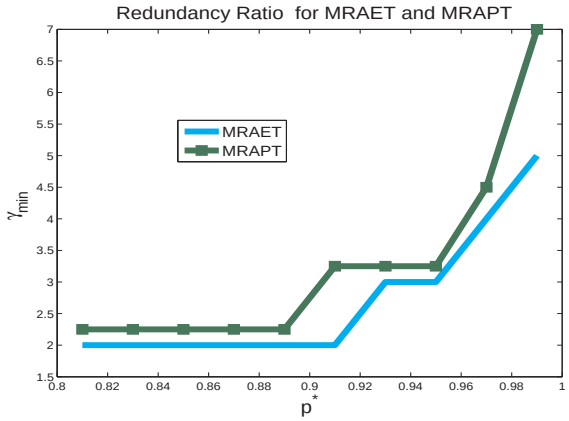

Figure 4.5: Redundancy Ratio for MRAPT and MRAET

Figures 4.6, 4.7, and 4.8 show the performance of MDS, LT, and Raptor codes using the MRAET algorithm for $k = 50$. In the remaining figures we change the number of input symbols to $k = 50$, and we evaluate the performance of MDS, LT, and Raptor codes using the MRAET algorithm. Fig. 4.6 shows the actual probability of successful decoding for MDS, LT, and Raptor codes as compared to the target probability of success. All of these codes have a successful decoding probability that is equal to or above p^*, making them excellent candidates for multipath channels. On average, the MDS code seems to have lower success probability than the other codes, though in Fig. 4.7 it can be seen that the MDS codes do have the minimum redundancy. The Raptor code has higher decoding success probability than the LT code for lower values of p^* but as p^* grows the LT code outperforms the Raptor code. Though in Fig. 4.7 LT code has significantly bigger codeword size than both Raptor and MDS codes.

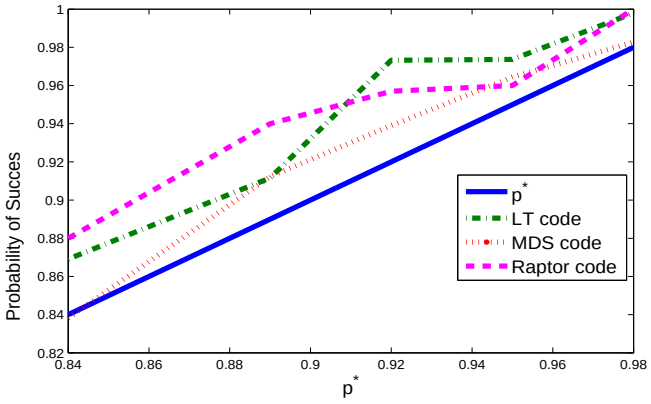

Figure 4.6: Probability of Success over Different Codes using MRAET

The Raptor code seems to have a constant amount of extra output symbols than the MDS code. The code has more redundancy added but it outperforms the MDS code significantly as shown in Fig. 4.8. Fig 4.8 depicts the average bit error probability for these three codes. The Raptor code has consistently lower bit error rate than both the other codes, and the LT code has lower bit error rate than the MDS code. The bit error rate for the Raptor code is extremely low and the LT error rate is higher by a relatively minute amount.

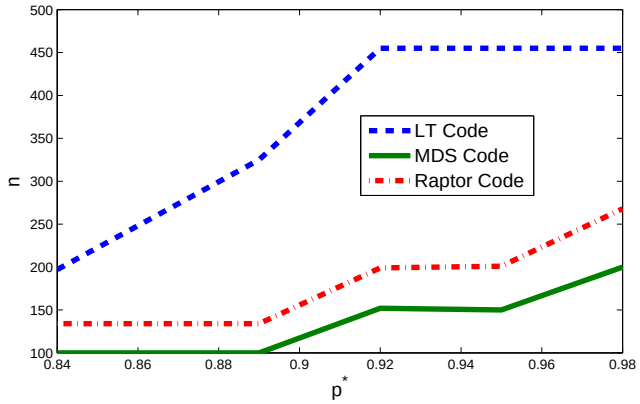

Figure 4.7: Total Codeword size for Different Codes using MRAET

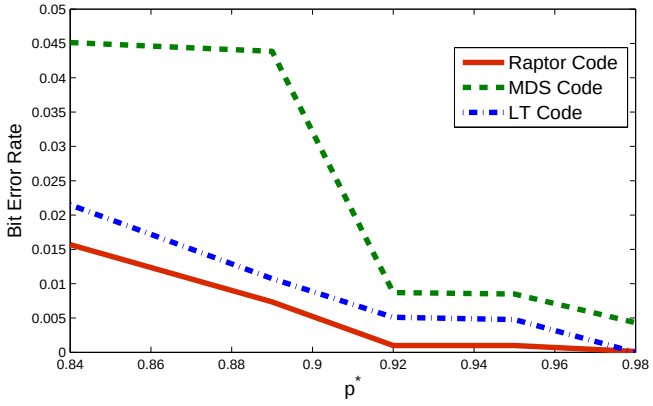

Figure 4.8: Bit Error Rate over Different Codes using MRAET

4.5 Summary

This chapter considered a network setup with a source and destination node and N independent paths separating them. These paths are erasure paths and they have a pre-determined "trustworthiness" level. We determined the minimum code length and message dispersal down the paths as to achieve a target success probability. Due to the inability to express the true success probability in closed form, we introduced two heuristic algorithms to determine the mentioned parameters. MRAET uses the actual success probability and results in an exponential running time. MRAPT on the other hand uses an approximation of the probability of success, which causes sub-optimal performance, though runs in efficient polynomial time. Both these algorithms are developed for the specific class of codes called MDS codes.

We applied MRAET to three codes, MDS, LT, and Raptor codes. The simulations showed that the Raptor code appears to be the best candidate for our algorithm in a multipath channel. It has a high success probability, along with very low bit error rate, and the code does not have a huge amount of redun-

dancy difference over the MDS code. If one wishes to have higher network efficiency MDS codes would be the ideal choice since the redundancy is lower than the other codes. The problem with MDS codes is that the higher encoding/decoding time result in more complex and expensive encoders/decoders. Also, if an MDS code is used, and the necessary output symbols are not received then it is much more difficult to retransmit the missing information. Raptor and LT codes are rateless meaning it is very simple to re-encode and transmit missing output symbols. Another advantage of these codes is the fact that they have cheap encoding/decoding costs and result in low bit-error rate.

CHAPTER 5

SECURITY AND RELIABILITY FOR MULTIPLE PATH NETWORK

5.1 System Model for reliability and security

In this model, we want a method to be simultaneously secure and robust from an adversary. There are two legitimate users in the network, Alice and Bob, and Alice would like to convey a message to Bob through an imperfect channel. Unfortunately, there is an adversary, Eve, present in the network who would like to spy on the conversation between Alice and Bob. Alice wishes to keep the data private from Eve, but she also wants Bob to be able to reconstruct the message. We use Raptor codes to both encrypt the message and to allow the receiver to decode the message with probability 1.

Network Model

Our network is similar to the network models used in [36] and [11]. We assume that the network has a large amount of nodes and that a disjoint path algorithm has found N node disjoint paths between the source (Alice) and destination (Bob). These paths could be picked based on numerous parameters such as their reliability, length, or trustworthiness level. Each of these paths independently acts like a binary erasure channel (BEC). Additionally, an eavesdropper (Eve) has access to each path and also views them as BEC's. In particular, each path can be thought of as a wiretap channel. We assume that the distributions across both the legitimate and illegitimate channels are known. The main channel on path i has bit erasure probability α_i and the wiretap channel has erasure

probability ϵ_i. Figure 5.1 shows the network model.

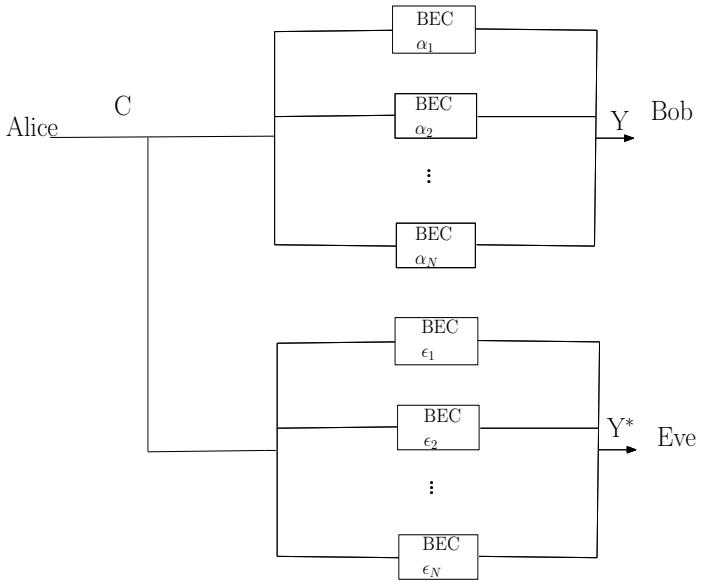

Figure 5.1: Multiple path network where both the legitimate and illegitimate users have binary erasure channel

Alice, the sender, wants to reliably broadcast a message to Bob without giving too much information to Eve. For each path i we assume that $\alpha_i < \epsilon_i$, since otherwise there would be no point in transmitting down that path. With out loss of generality, we assume $\alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_N$. Alice has a message S that is k bits long which she will encode into a length n codeword C using a Raptor code. We will introduce Raptor codes in the next section. In order to increase the probability of successful transmission, Alice will split the codeword into N groups of bits $\{f_1, f_2, \ldots, f_N\}$ and then she will transmit the first group of f_1-bits or $\mathbf{c}^{(1)} = (c_1^{(1)} c_2^{(1)} \ldots c_{f_1}^{(1)})$ down path 1, next f_2-bits, $\mathbf{c}^{(2)} = (c_1^{(2)} c_2^{(2)} \ldots c_{f_2}^{(2)})$, down path 2, and so on. Figure 5.2 shows how each path i looks individually. Alice wants to determine the minimum redundancy and the bit-allocation across each path to assure that the message can be decoded by Bob and that it is perfectly secure from Eve.

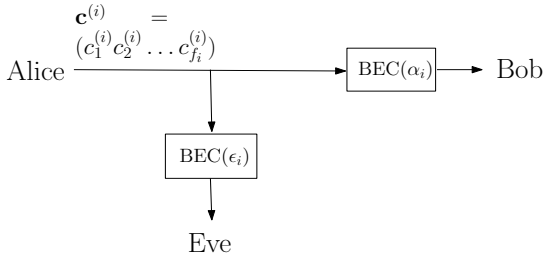

Figure 5.2: Wiretap channel on each path *i*

5.2 Asymptotic System Secrecy and Reliability

The goal is to develop an encoding scheme to assure a zero error probability for Bob and complete uncertainty for Eve. In [34], the authors show how LDPC codes can be applied to the wiretap channel to achieve asymptotic robustness and security. Previous work [11] has shown that Raptor codes perform well over multiple path erasure channels. We extend the work in [34] to multiple path wiretap channels, using Raptor cod es rather than LDPC codes. An advantage of Raptor codes is the fact that they are rateless, so if the receiver does not receive enough bits, then some extra output symbols can be transmitted on the fly. Also, these codes have linear encoding and decoding time.

5.2.1 Raptor Code Parameters

As mentioned above, Raptor codes are well known fountain codes which were first introduced in [33]. The structure of Raptor codes allows them to have high error correction as well as having properties of rateless codes. This is due to the fact that they are constructed using a high performing error-correction code as a pre-code followed by an LT code, which is a fountain code.

Assume that $c\sum_{i=1}^{N} f_i$ bits are transmitted over our network, where f_i represents the number of bits transmitted across path i and c the total number blocks of bits transmitted across paths (so there are c blocks of f_i-bits transmitted across path i). In our model, asymptotically (as $c \to \infty$), Bob and Eve will receive $c\sum_{i=1}^{N} f_i\alpha_i$ and $c\sum_{i=1}^{N} f_i\epsilon_i$ erasures respectively. This is true because the distribution of each main path (same analysis for the eavesdroppers path) is Bernoulli with parameter α_i, since we have two possible outcomes: an erasure with probability α_i and the transmitted bit with probability $1 - \alpha_i$. It is known that if one obtains n independent samples of a Bernoulli random variable with parameter p (representing an erasure), then as n gets large, the number of erasures approaches the mean of the random variable, np. Hence, if cf_i bits are transmitted down path i, then the number of erasures on this path as c tends to infinity will approach $cf_i\alpha_i$, implying that the total number of erasures across all the paths becomes $c\sum_{i=1}^{N} f_i\alpha_i$. Using the same LT code parameters, (n', Ω_D), as found in [33], we have a degree distribution:

$$\Omega_D(x) = \frac{1}{\mu + 1}\left(\mu x + \sum_{i=2}^{D} \frac{x^i}{(i-1)i} + \frac{x^{D+1}}{D}\right)$$

where $D = \lceil 4(1+\epsilon)/\epsilon \rceil$, $\mu = (\epsilon/2) + (\epsilon/2)^2$, and ϵ is the overhead of the entire encoding process. We restate an important result shown in [33] for clarity.

Lemma 1 *$\exists a(\epsilon) > 0$ such that any set of $n'(1+\epsilon/2)+1$ output symbols of an LT code with parameters (n', Ω_D) are sufficient to recover at least $(1-\delta)n'$ with an error probability at most $e^{-a(\epsilon)n'}$, where $\delta = (\epsilon/4)/(1+\epsilon)$.*

Given a choice of δ, Alice needs to determine the amount of bits to transmit. If $n'(1+\epsilon/2) = ck(1+\epsilon)$ output bits are sent, with cf_i transmitted on each path, then, asymptotically, Bob will receive $n'(1+\epsilon/2) - c\sum_{i=1}^{N} f_i\alpha_i$ bits. In order to

guarantee $1-\delta$ fraction of decoded bits, Alice needs to consider the erasures Bob's channel will incur. Thus she needs to send $ck(1+\epsilon) + c\sum_{i=1}^{N} f_i\alpha_i$ output bits to Bob. Mathematically this implies that Alice send $c\sum_{i=1}^{N} f_i = n$ bits to Bob, where $c\mathbf{f}^T(\mathbf{1}-\alpha) = ck(1+\epsilon)$. Then according to Lemma 1, and assuming that Bob uses the BP decoding algorithm, at least $(1-\delta)n'$ bits will be recovered after the LT decoding.

The next question we answer is how to determine the values of f_i, $i \in \{1,\ldots,N\}$. Starting with the value of δ we determine the overhead ϵ. We know that it is undesirable to transmit more than k-bits down each path and ideally we want to minimize the total number of transmitted bits. This is a simple linear optimization problem of the following form:

$$\begin{aligned} \min \quad & \mathbf{f}^T\mathbf{1} \\ \text{s.t} \quad & \mathbf{f}^T(\mathbf{1}-\alpha) = k(1+\epsilon) \\ & 0 \le f_i \le k, \quad \forall i \in \{1,\ldots,N\} \end{aligned}$$

It is assumed that Eve knows the connectivity graph of both the LDPC code and the LT code. Typically in information security papers it is assumed that the adversary has infinite computational power. Thus, instead of assuming that Eve uses the BP algorithm to decode the message, we assume that she uses the best possible graph decoding algorithm. It is known that Maximum Likelihood (ML) decoding is optimal, thus Eve cannot do any better than it.

In [29], the authors provide lower bounds on the bit error probability for codes on graphs. A bound found in this paper that is of interest to us is one that shows the minimum possible erasure probability after decoding a graph code which traversed a binary erasure channel (BEC). This bit erasure probability is lowest for all possible decoding algorithms, even the Maximum-Likelihood

(ML) algorithm, meaning that Eve cannot achieve a lower erasure probability. Before showing this lower bound we will define some variables. Assume that H is the parity check matrix for the LT code. A parity check matrix of an LT code is such that $HX^T = C^T$, where the row H_i is generated randomly using the output degree distribution (in our setup $\Omega_D(x)$) [15].

Definition 17 *The **density** $\Delta = \Delta(H)$ of a parity check matrix H is the normalized number of ones in H per information bit.*

Definition 18 *The **normalized density** $t = t(H)$ of H is:*

$$t = t(H) = \frac{R\Delta}{2 - R}$$

Where R is the rate of the code. Let P_b be the bit erasure probability and the erasure probability across the channel is ϵ. The bound is as follows:

$$h(P_b) \geq R - (1 - \epsilon) + (1 - R)(1 - \epsilon)^{\frac{(2-R)t}{1-R}} \tag{5.1}$$

where $h(\cdot)$ is the binary entropy function (logarithm to the base of 2). This bound is used in the following manner. Since $c \sum_{i=1}^{N} f_i = n$ bits are transmitted, this implies that as $c \to \infty$ Eve will have $c \sum_{i=1}^{N} f_i \epsilon_i$ erasures. Therefore, Eve will asymptotically have probability of bit erasure $\epsilon_{\text{avg}} = c \sum_{i=1}^{N} f_i \epsilon_i / n = \sum_{i=1}^{N} f_i \epsilon_i / \sum_{i=1}^{N} f_i$ over the combined paths. Using ϵ_{avg} as the erasure probability over Eve's entire channel, one can find the lower bound for the bit erasure probability after decoding the LT layer by solving Equation (5.1) for P_b. In particular, since P_b represents the minimum possible fraction of erasures after decoding with the best possible decoding algorithm, it can be used to determine an upper bound on the fraction of symbols Eve decodes, which is $(1 - P_b)n'$. The erasure probability P_b is associated with the case where Eve uses the best possible decoding algorithm over

the LT code, hence one needs to assure that δ is picked so that $\delta < P_b$. Since δ represents the upper bound for the number of erasures Bob will have after the LT decoding and P_b is the lower bound on Eve's erasures, this implies that a scheme around these parameters will consider the worst case scenario. Hence, such a scheme will be valid for any feasible decoding outcome.

We first mention some information about the LT decoding in order to properly determine the LDPC code parameters. The LDPC parameters are picked to assure perfect secrecy and reliability, which will be shown after the encoding process is mentioned. After decoding the received message using the LT code's tanner graph, both Bob and Eve's inner channel is transformed into a bit erasure channel (BEC), which can be seen in Figure 5.3. This means that the multiple path network is converted into a single path network following the first decoding process. Using Lemma 1 and Equation (5.1), the decoding of the LT code results in Bob having a BEC with erasure probability δ and Eve having a BEC with erasure probability P_b . We begin with the encoding process which is similar to that in [34].

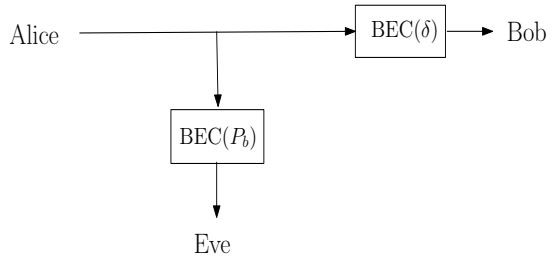

Figure 5.3: Channel Transformation after LT Decoding (BEC Wiretap Channel)

As mentioned above, the wiretappers inner BEC channel has erasure probability P_b and the legitimate receivers has erasure probability δ. First pick a length n' code C_1, with rate r_1 and parity matrix H_1, from an LDPC ensemble

with threshold P_b. Next, consider the dual space of C_1 and pick $n'(1-r_2)$ independent vectors, where $r_2 > r_1$. From these vectors form a $n'(1-r_2) \times n'$ parity matrix H_2, corresponding to code C_2. C_2 should be from the LDPC ensemble with threshold δ. Combine the rest of the independent vectors in the dual space of C_1 to construct a matrix $\bar{H_2}$. Capacity and security requirements yield the following inequalities:

$$\begin{aligned} 1-r_2 &\geq \delta \\ 1-r_1 &\geq P_b \\ 1-r_2 &< P_b \end{aligned}$$

Soon it will be shown how the parameters of the code result in secrecy and reliability, but first the encoding method will be discussed.

5.2.2 Raptor Encoding

Suppose Alice wants to send a secret message $\mathbf{S}$ which has $n'(r_2-r_1)$-bits. She first concatenates $n'(1-r_2)$ 0's with $\mathbf{S}$ resulting in an $n'(1-r_1)$-bit vector. Then, she randomly picks a vector $\mathbf{X}$ out of the vectors that form a solution to:

$$\begin{bmatrix} H_2 \\ \bar{H_2} \end{bmatrix} \mathbf{X}^T = [0 \cdots 0 \mathbf{S}]^T \tag{5.2}$$

It can be shown that using this encoding method, the secrecy rate becomes $r_2 - r_1$. Finally, $\mathbf{X}$ is encoded into an $n = \sum_{i=1}^{N} f_i$ bit vector, $\mathbf{C}$, using the LT code with parameters (n', Ω_D).

5.2.3 Raptor Decoding

Asymptotically, Eve receives $c\sum_{i=1}^{N} f_i\epsilon_i$ erasures, and after applying the optimal decoding using the LT code connectivity graph, by Equation (5.1) she will decode at most $(1 - P_b)n'$ bits. Thus in the long run, after decoding the LT code, Eve will have $P_b n'$ erasures. This results in $2^{n'(P_b-(1-r_2))}$ solutions to $H_2\mathbf{X}^T = 0$ which are equally likely. The code C_1 has threshold P_b implying that any submatrix formed from nP_b columns of its parity matrix H_1 will have full column rank [27]. This implies that each solution $\mathbf{X}$ maps to a specific value of $\mathbf{S}$. Hence Eve's uncertainty is $\Delta = n'(P_b - (1 - r_2))$, and using a capacity achieving code with $1 - r_1 = P_b$, her uncertainty becomes $\Delta = n'(r_2 - r_1)$, implying complete secrecy.

Bob acquires $c\sum_{i=1}^{N} f_i(1 - \alpha_i)$ bits of the codeword $\mathbf{C}$, and by lemma 1, he decodes $n'(1 - \delta)$ bits of the intermediate codeword $\mathbf{X}$. Since C_2 has erasure threshold δ, this means that as $n \to \infty$ Bob can decode the sent codeword, $\mathbf{X}$, using belief propagation with probability one. He can then form an estimate of $\mathbf{S}$ by solving $\bar{H}_2\mathbf{X}^T = \mathbf{S}$. Hence the probability of error approaches zero as n approaches infinity using this coding method.

5.2.4 Secrecy Capacity

It can be shown that the secrecy capacity of the inner channel is $C_s = P_b - \delta$. The rate of our encoding scheme is $r_2 - r_1 = r_2 - (1 - P_b)$. The scheme does not attain the secrecy capacity unless C_2 is also picked to be capacity achieving code with $r_2 = 1 - \delta$, yielding a rate of $r_2 - (1 - P_b) = P_b - \delta = C_s$.

5.3 Summary

In this chapter we have considered the problem of deterring a wiretapper in a multiple path wireless network. We have shown how Raptor codes can be applied to our network model to guarantee security and robustness. Specifically, we derived the number of erasures that the eavesdropper could correct from decoding the LT code, and then determined the necessary parameters for the LDPC layer. Also, we devised a method to route the codeword through a multiple path network, assigning a certain number of bits to each path. The routing scheme depends on the erasure probabilities and the LT code specifics.

CHAPTER 6

DETECTING MALICIOUS NODES

6.1 System Model for localizing malicious nodes

Our network is composed of $N + 2$ nodes with one source node and one destination node, let V be a set containing the nodes in the the network where $V = \{u_s, u_1, u_2, \ldots, u_N, u_d\}$ (u_s, u_d represent the source and destination node respectively). We assume that malicious nodes are present in the network. The source wishes to transmit a message to the destination node, and the destination node wants to determine the adversary node locations. The network can be represented by $G = (V, E)$, where E represents all the edges in the network. It is assumed that the source and destination node are impervious to infection while each of the internal nodes are not. By infected node we mean the node is being tampered with, i.e. a malicious or byzantine node. Note, this should not be confused with the network epidemic spreading problem since a malicious node will not transmit its "infection" to any of its neighbors. Internal node u_i has a probability p_i of becoming infected. If a node is overtaken by an adversary, the node will refuse to forward legitimate data and instead will transmit tampered with/fake packets.

Prior to transmitting a message, the source node will encode the message using a standard encryption technique. This implies that the receiver will be able to recognize a false or tampered with packet when the packet is decrypted. Thus, the destination can denote an uncorrupted packet by a '0' and a corrupted one by a '1'. The output of a particular path at the destination is a function of all the nodes that form the path. The destination node will receive n messages

from n different paths (messages not necessarily unique) and it represents the legitimacy status with a vector $\hat{\mathbf{c}}$, of length n, in the following manner:

$$\hat{c}_i = \begin{cases} 0 & \text{if message received on path } i \text{ is legitimate} \\ 1 & \text{if message received on path } i \text{ is illegitimate} \end{cases} \tag{6.1}$$

If there is an adversary on at least one node in a path, the false message from the node will be propagated across the rest of the path, resulting in the detection of the illegitimate message by the terminal node. In particular, path i (P_i) will have $\hat{c}_i = 1$ if at least one node on the path has been infected. Let '1' represent the event that a specified node is malicious and '0' if the node has not been tampered with. Form a variable $\mathbf{s} = \{s_1, s_2, \ldots, s_N\}$ expressing this information, or

$$s_i = \begin{cases} 1 & \text{if } u_i \text{ is malicious} \\ 0 & \text{otherwise} \end{cases} \tag{6.2}$$

The the value of $\hat{c}_i$ can be written as function of all s_j given that $u_j \in P_i$. The function for $\hat{c}_i$ is the 'or' operation over all the nodes in path i. Mathematically, the output of path i can be written:

$$\hat{c}_i = \bigvee_{j:u_j \in P_i} s_j \tag{6.3}$$

It can be seen that this expression can be re-formulated in terms of addition in the Galois field of order 2, GF(2), as

$$\hat{c}_i = \sum_{j:u_j \in P_i} s_j + \sum_{j:u_j \in P_i} \sum_{k>j:u_k \in P_i} s_j s_k + \ldots + \prod_{j:u_j \in P_i} s_j \tag{6.4}$$

6.2 Localizing Adversary Nodes

Our method is based on transmitting a message or many messages across a large number of paths, and using the path outputs to determine the locations

of adversaries in the network. The intuition behind it comes from ECC theory: by forming a set of specified combinations of the message bits into a codeword and transmitting across a channel, one can determine the points at which errors occurred as long as the number of errors is less than a specified number. With enough paths traversing through the nodes, it is possible to deduce the infected node locations. In particular, we treat the node status variable $\mathbf{s} = \{s_1, \ldots, s_N\}$ as a message vector. The reasoning behind this is explained in the next section. This allows us to determine the values of each s_i by using RM codes.

6.2.1 Mapping Paths to Reed-Muller Codes

As mentioned earlier, the variable representing each node's legitimacy status is considered to be the message bit, i.e. s_i is the ith message bit. For each path i, form an N-bit vector P_i which has a '1' in spot i if u_i belongs to path i and a '0' otherwise. Eq. (6.4) reveals that each received codeword bit $\hat{c}_i$ is formed by monomial combinations of the node status variables which belong to path i. The goal is to map this expression to the formation of RM codes.

In RM codes, given the generator matrix and message vector, the codeword is formed as in Eq.(2.4) in Chapter 2 resulting in:

$$\begin{aligned} \mathbf{c} &= (c_0, c_1, \ldots, c_{2^m-1}) \\ &= s_0\mathbf{1} + s_1\mathbf{v}_1 + \ldots + s_m\mathbf{v}_m + s_{12}\mathbf{v}_1\mathbf{v}_2 \\ &\quad + \ldots + s_{m-r+1\ \ldots\ m-1\ m}\mathbf{v_{m-r+1}} \cdots \mathbf{v_{m-1}}\mathbf{v_m} \end{aligned} \tag{6.5}$$

Where s_{m+1} has been renamed s_{12}, s_{m+2} has been renamed s_{13}, and so on until

s_N has been renamed $s_{12\ldots m}$. Consider a row in the above expression, or $c_i = s_0 + s_1 v_1(i) + \ldots + s_m v_m(i) + s_{12} v_1(i) v_2(i) + \ldots + s_{m-r+1\ldots\, m-1\; m} v_{m-r+1}(i) \cdots v_m(i) = \sum_{j:u_j \in P_i} s_j$, and let $P_i = (1,\ v_1(i),\ \ldots,\ v_m(i),\ v_1(i)v_2(i),\ \ldots,\ v_{m-r+1}(i) \cdots v_m(i))$. If there is a '1' in the jth spot then node u_j is present in path i, otherwise it is not. This implies that each column of the generator matrix represents a required path in the network and we call these paths RM-paths.

Comparing the two equations (6.4) and (6.5), it can be seen that they do not match. In particular, Eq.(6.5) is missing all terms of order greater than one. We assume that the message has been encoding using the RM code, as in Eq. (6.5), and the codeword $\hat{\mathbf{c}}$, corresponding to Eq.(6.4), is received. The extra terms found in Eq. (6.4) on each path i can be treated as the error in the ith codeword bit, so $\hat{c}_i = c_i + \epsilon_i$ where

$$\epsilon_i = \sum_{j:u_j \in P_i} \sum_{k>j:u_k \in P_i} s_j s_k + \ldots + \prod_{j:u_j \in P_i} s_j \tag{6.6}$$

Thus, it is necessary to analyze the probability that the higher order terms alter the true codeword value.

The only way that the higher order terms change the value of the first order terms in path i is if the sum of them results in the value '1' or $\epsilon_i = 1$. It can be shown that this occurs if an even number greater than zero of the message bits on path i have the value '1'. For simplicity, assume that with probability $p_i = p$ an interior node u_i becomes malicious. The following equations can be easily altered for the case where the nodes have different infection probabilities. Let l_i represent the number of nodes on path i, $n = 2^m$ represent the number of paths, and

$$N_e(l_i) = \begin{cases} l_i & \text{if } l_i \text{ is even} \\ l_i - 1 & \text{otherwise} \end{cases}$$

Then an error on path i occurs with probability:

$$P_{\text{error}} = \sum_{k=2,4,6\ldots,N_e(l_i)} \binom{l_i}{k} p^k (1-p)^{l_i - k} \tag{6.7}$$

For simplicity, since the structure of the paths has not yet been determined, it is assumed that the paths are independent from one another. Hence, the expected number of errors over all the paths approximately becomes:

$$\begin{aligned} & E\{\text{Number of errors}\} \\ \approx & \sum_{i=1}^{n} \sum_{k=2,4,\ldots,N_e(l_i)} \binom{l_i}{k} p^k (1-p)^{l_i - k} \\ = & \sum_{i=1}^{n} \left[\sum_{k=0,2,\ldots,N_e(l_i)} \binom{l_i}{k} p^k (1-p)^{l_i - k} - (1-p)^{l_i} \right] \\ = & \sum_{i=1}^{n} \left(\frac{1}{2} - (1-p)^{l_i}\right) \\ = & \frac{n}{2} - \sum_{i=1}^{n} (1-p)^{l_i} \end{aligned} \tag{6.8}$$

This expression will be used to determine the parameters for the RM code. In particular, the expected number of errors is used to determine the minimum distance of the code.

6.2.2 Designing the Reed-Muller Parameters

The length of each path vector P_i needs to be equal to the number of nodes N. This implies that m and r should be such that

$$N = \sum_{j=0}^{r} \begin{pmatrix} m \\ j \end{pmatrix}$$

The code must be able to "correct" at least $\frac{n}{2} - \sum_{i=1}^{n}(1-p)^{l_i}$ errors to determine the location of the adversaries. An error-correction code with minimum distance $d_{\min}$ can decode $\lfloor \frac{d_{\min}-1}{2} \rfloor$ errors. This implies that the minimum distance of the code must satisfy $d_{\min} = 2^{m-r} \geq 2(\frac{n}{2} - \sum_{i=1}^{n}(1-p)^{l_i}) + 1$. The larger the minimum distance, the longer the codeword, hence implying more paths. The values of l_i are not known, so it is necessary to form bounds on the number of errors. There are no more than N nodes on each path implying $l_i \leq N$ for all i, therefore,

$$2(\frac{n}{2} - \sum_{i=1}^{n}(1-p)^{l_i}) + 1 \leq 2(\frac{n}{2} - n(1-p)^{N}) + 1 \leq d_{\min}$$

Since $n = 2^{m-r}$ we have,

$$\begin{aligned} d_{\min} &= 2^{m-r} \geq 2^{m}(1-(1-p)^{N}) + 1 \\ &\Rightarrow 2^{-r} \geq 1 - (1-p)^{N} + 2^{-m} \\ &\Rightarrow r \leq -\log_2(1-(1-p)^{N} + 2^{-m}) \end{aligned} \tag{6.9}$$

Using this minimum distance, the code can localize all malicious nodes as long as there are less than $n(1-(1-p)^{N}) + 1$ errors.

6.2.3 Forming Paths in Network

These relationships between N, m and r give the possibility to obtain their values and form an $\mathcal{R}(r, m)$ RM-code. The code is formed as in Section (2.3.7) of Chapter

2. An issue occurs if all the RM-paths are not present in the network, thus next we discuss how to assure that the RM-paths formed from the code are realizable in the network. First, all the simple paths $\mathcal{P}$ (paths not containing a specific node more than once) between the source and destination must be found. To do this we use an algorithm, developed by Rubin, in [28] which finds all simple paths between each pair of nodes. Once all the paths are found, they are compared to the paths that were generated by the RM code. All the RM-paths P_i that belong to $\mathcal{P}$ are fine and can be used in localizing the malicious nodes. The remainder of the RM-paths, $P_i \notin \mathcal{P}$, must be formed by combinations of paths in $\mathcal{P}$. These combinations can be found by starting with combining two paths, comparing them to the particular path $P_i \notin \mathcal{P}$ and seeing if they match. If no two combinations match, then comparing them to combinations with three paths, and continuing in this fashion until a matching combination has been found. Suppose three paths, $P'_{k_1}, P'_{k_2}, P'_{k_3}$, are needed to represent a path P_k (composed of nodes $u_{k_1}, \ldots, u_{k_{l_k}}$) which was found with the RM code. This means that $P'_{k_1} + P'_{k_2} + P'_{k_3} = P_k$ and $c_k = s_{k_1} + s_{k_2} + \ldots + s_{k_{l_k}}$. The decoder will have $\hat{c}_{k_1}, \hat{c}_{k_2}, \hat{c}_{k_3}$ from paths $P'_{k_1}, P'_{k_2}, P'_{k_3}$ respectively. Then the decoder will combine the three outputs of the paths as $\hat{c}_k = \hat{c}_{k_1} + \hat{c}_{k_2} + \hat{c}_{k_3}$. This implies that the probability of error on paths that do not belong in $\mathcal{P}$, and hence must be formed through a combination of paths, is larger than that on the paths in $\mathcal{P}$ (shown in Eq. 6.7).

This error probability can be analyzed as follows. Assume that an RM-path P_k needs to be realized by using n different paths, so $P_k = \sum_{i=1}^{n} P'_{k_i}$. Then ,

$$\begin{aligned}\hat{c}_k &= \sum_{i=1}^{n} \hat{c}_{k_i} \\ &= \sum_{i=1}^{n} \left[\sum_{j:u_j \in P'_{k_i}} s_j + \sum_{j:u_j \in P'_{k_i}} \sum_{l>j:u_l \in P'_{k_i}} s_j s_l + \ldots + \prod_{j:u_j \in P'_{k_i}} s_j \right]\end{aligned}$$

Since P_k is formed through the modulus two addition of n paths, this implies that the first order message bits corresponding to the nodes that do not belong on path P_k are canceled out. Hence,

$$\begin{aligned}\hat{c}_k &= \sum_{j:u_j \in P_k} s_j + \sum_{i=1}^{n} \left[\sum_{j:u_j \in P'_{k_i}} \sum_{l>j:u_l \in P'_{k_i}} s_j s_l + \ldots + \prod_{j:u_j \in P'_{k_i}} s_j \right] \\ &= c_k + \sum_{i=1}^{n} \left[\sum_{j:u_j \in P'_{k_i}} \sum_{l>j:u_l \in P'_{k_i}} s_j s_l + \ldots + \prod_{j:u_j \in P'_{k_i}} s_j \right]\end{aligned}$$

Thus, the error term can be seen to be:

$$\epsilon'_k = \sum_{i=1}^{n} \left[\sum_{j:u_j \in P'_{k_i}} \sum_{l>j:u_l \in P'_{k_i}} s_j s_l + \ldots + \prod_{j:u_j \in P'_{k_i}} s_j \right] \tag{6.10}$$

An error occurs in the kth codeword bit only if $\epsilon'_k = 1$, which happens only if an odd number of the n paths have an even number of malicious nodes. Given the relationships between these n paths, the probability of error can be analyzed.

Once all the messages have been received, the destination node will combine the required paths, and then use the Reed Decoding algorithm to form an estimate of the message. If spot i has a '1', then this implies that node v_i has been compromised. This information allows the network to update the probability of each node being malicious, and if necessary remove certain nodes from the network. An RM path which is not present in $\mathcal{P}$ may result in a higher error

probability than that calculated in Eq.6.7, which may yield more codeword errors than expected and an incorrect message estimate. This uncertainty can be taken into account when updating the probability of a node being malicious.

6.2.4 Treating Missing RM-paths as Erasures

The resulting probability of bit error is higher when several paths are combined to form an RM-path, thus we consider another technique where the missing RM-paths are treated as erasures. We develop an algorithm which exploits the structure of our decoding process to mitigate erasures. Before introducing the algorithm, a few key observations are mentioned.

- If $\hat{c}_i = 0 \Rightarrow \hat{c}_i = c_i$ and $s_j = 0 \quad \forall s_j \in P_i$
- If $s_{i_1 i_2 \ldots i_j} = 0$ is known and one of its bit estimate expressions or $\hat{s}_{i_1 i_2 \ldots i_j} = \hat{c}_{t_1} + \hat{c}_{t_2} + \ldots + \hat{c}_{t_j}$ has exactly one k such that $\hat{c}_{t_k} = 1$ and has no erasures, then $\hat{c}_{t_k}$ has an error and $c_{t_k} = 0$.
- If $s_{i_1 i_2 \ldots i_j} = 0$ is known and one of its bit estimate expressions or $\hat{s}_{i_1 i_2 \ldots i_j} = \hat{c}_{t_1} + \hat{c}_{t_2} + \ldots + \hat{c}_{t_j}$ has exactly one k such that $\hat{c}_{t_k} = e$ (an erasure) and has no $c_{t_l} = 1, \quad \forall l \in \{1, \ldots, j\}$, then $c_{t_k} = 0$.

These observations are used to generate an algorithm which is used in combination with the Reed-decoding algorithm to form an estimate of the original message. The algorithm will be introduced next and then we will show some simulations showing the differences between using and not using the algorithm. Assume $\hat{c}_i = e$ represents an erasure and initialize an empty set, $M = \varnothing$. To simplify some notation, assume that for $1 \leq j \leq m$, $u_j \in P_i$ is analogous to $s_j \in P_i$,

$u_{m+1} \in P_i \Rightarrow s_{12} \in P_i$, $u_{m+2} \in P_i \Rightarrow s_{13} \in P_i$, ..., and $u_N \in P_i \Rightarrow s_{12\ldots m} \in P_i$. Also, the algorithm mention earlier is referred to as the Reed-Decoding algorithm and the following algorithm is referred as the RD-Erasures algorithm

RD-Erasures Algorithm

Step 1: $\forall\ \hat{c}_i \quad i \in \{1, \ldots, 2^m\}$ such that $\hat{c}_i = 0$ and $s_{i_1 i_2 \ldots i_j} \in P_i$ set to $\hat{s}_{i_1 i_2 \ldots i_j} = 0$ and add $\{i_1 i_2 \ldots i_j\}$ as a set into M. Set means that if $\{i_1 i_2 \ldots i_j\}$ is in M that does not necessarily imply that i_1 is in M. (M represents the message bits which are known with complete certainty).

Step 2: $\forall\ \hat{c}_i \quad i \in \{1, \ldots, 2^m\}$ such that $\hat{c}_i = e$ check if $\{i_1 i_2 \ldots i_j\} \in M,\ \forall s_{i_1 i_2 \ldots i_j} \in P_i$. If so, then set $\hat{c}_i = 0$.

Step 3: Let $j = r$ (highest order)

Step 4: Start with Step 2 of the Reed-Decoding algorithm as mentioned in an earlier section.

Step 5: Stop the Reed-Decoding algorithm directly before Part 6 of Step 2 for bit $s_{i_1 i_2 \ldots i_j}$, check to see if $\{i_1 i_2 \ldots i_j\} \in M$. If so keep going with this step knowing that $\hat{s}_{i_1 i_2 \ldots i_j} = s_{i_1 i_2 \ldots i_j}$, otherwise go to Step 6 of this algorithm.

1. If $\exists$ exactly one $\hat{c}_{t_l},\ l \in \{1, \ldots, j\}$ such that $\hat{c}_{t_l} = 1$ and the rest of the current codeword bits $\{\hat{c}_{t_1}, \hat{c}_{t_2}, \ldots, \hat{c}_{t_j}\} \backslash \hat{c}_{t_l}$ used for an estimate of $\hat{s}_{i_1 i_2 \ldots i_j}$ have value '0', then set $\hat{c}_{t_l} = 0$. Otherwise do nothing
2. Else if $\exists$ exactly one $\hat{c}_{t_l},\ l \in \{1, \ldots, j\}$ such that $\hat{c}_{t_l} = e$ and the rest of the current codeword bits $\{\hat{c}_{t_1}, \hat{c}_{t_2}, \ldots, \hat{c}_{t_j}\} \backslash \hat{c}_{t_l}$ used for an estimate of $\hat{s}_{i_1 i_2 \ldots i_j}$ have value '0', then set $\hat{c}_{t_l} = 0$. Otherwise do nothing

Go to Step 3 of the Reed-Decoding algorithm, and at the end of this step, if the decoding process is not done, go to Step 4 of the RD-Erasures algorithm.

Step 6: Continue with rest of Reed-Decoding algorithm starting at Part 6 Step 2. Go to Step 3 of the Reed-Decoding algorithm, and at the end of this step, if the decoding process is not done, go to Step 4 of the RD-Erasures algorithm.

In the next section, we simulate the performance of several methods in the presence of missing paths. In all the methods we assume that there are certain paths not realizable in the network. The details of the message are discussed next.

6.3 Simulations

The simulations show comparisons of bit-error probability over several methods. One of the methods uses the path combination method to take care of the paths not in the network and then follows with the Reed-Decoding Algorithm, we call it the "path-combination" technique. The next method treats the non-realizable paths as erasures and just ignores those codeword bits in the Reed-Decoding algorithm. This implies that in the presence of erasures each bit estimate is formed with a smaller number of values in the majority ruling. The final method also assumes that the non-realizable paths are erasures and uses the RD-Erasures algorithm to form a message estimate.

Assume that the number of nodes/message bits is 11 and that the number of codeword bits is 16 with code parameters $r = 2, m = 4$. The different curves

on the first two plots each represent a different number of path erasures. Figure 6.2 shows the bit-error probability over different malicious node probabilities for the method which assumes erasures and only uses the Reed-Decoding algorithm. Bit-error probability is analogous to the probability that a node is mistaken to be malicious/non-malicious in our setup. Figure 6.1 shows the bit-error probability over different malicious probabilities for the case with erasures and using RD-Erasures algorithm. It can be seen that for both methods when the number of erasures is between 1 and 4 the bit-error probability is almost the same, especially for the RD-Erasures algorithm. An interesting result occurs when there are no erasures, the bit-error probability is actually higher than when there are 1-4 erasures. This implies that it is better to have an erasure than an error. Another explanation for this if there is more than one non-realizable path, then one of these paths will most certainly be the path including all the nodes. It can be deduced that this path is the most prone to errors since it contains the most nodes. Similarly, as expected, in both plots the bit-error probability increases as the probability of a node being malicious increases. It can be seen in figure 6.3 that the method using the RD-Erasures algorithm outperforms (has a lower bit-error probability) than the method only using the Reed-Decoding algorithm. The RD-Erasures algorithm significantly outperforms only using the Reed-Decoding algorithm in the case of no erasures and the difference increases as the probability of a node being malicious grows. When the number of erasures gets large, the performance deteriorates, as seen when there are six erasures in both fig.'s 6.1 and 6.2. The RD-Erasure algorithm yields a low bit-error probability for most instances.

Next, it is assumed that we have a specific node connectivity matrix, implying that we have a set number of non-realizable paths. In particular, there are 4

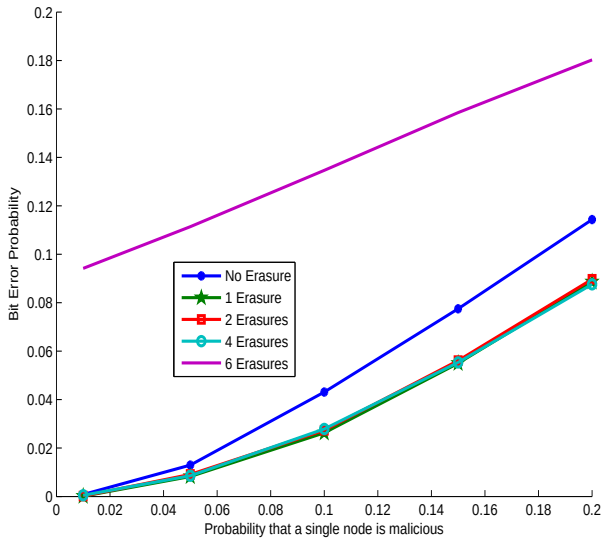

Figure 6.1: Performance of the RD-erasures algorithm over different number of erasures

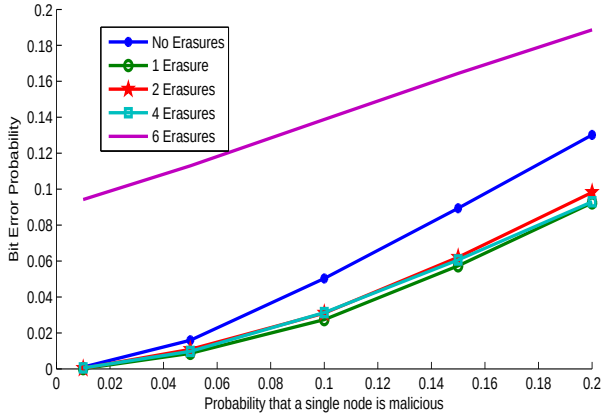

Figure 6.2: Performance of the Reed-Decoding algorithm over different number of erasures

non-realizable paths out of 16. Assume that the first node has a connection to the source node (otherwise we can re-order the nodes so that this condition is met).

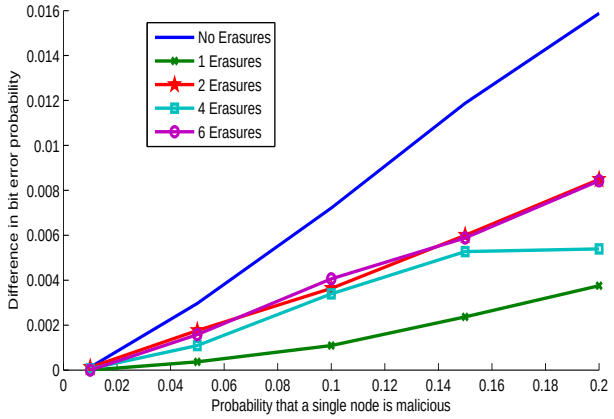

Figure 6.3: Difference between the bit-error probability of the Reed-Decoding algorithm and the RD-Erasures algorithm

Difference between the bit-error probability of the Reed-Decoding algorithm and the RD-Erasures algorithm over a different number of erasures

$$\begin{pmatrix}
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

Forming this matrix (each row represents a path) of realizable RM-paths. The

last node in a row vector with the value '1' is connected to the destination node:

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1
\end{pmatrix}$$

For the "combining-paths" technique, the paths that are needed to combine to form each non-realizable path are found. As mentioned earlier, the missing paths are handled as erasures for the RD-Erasure algorithm. The performance of the two methods is shown in figure 6.4, which compares the bit-error probability as the probability of a node being malicious increases. It can be seen that the RD-erasures algorithm significantly outperforms the "combining-paths" technique. The "combining-paths" technique seems to have a propagation of errors and hence our conclusion is that it is better to treat the non-realizable paths as erasures.

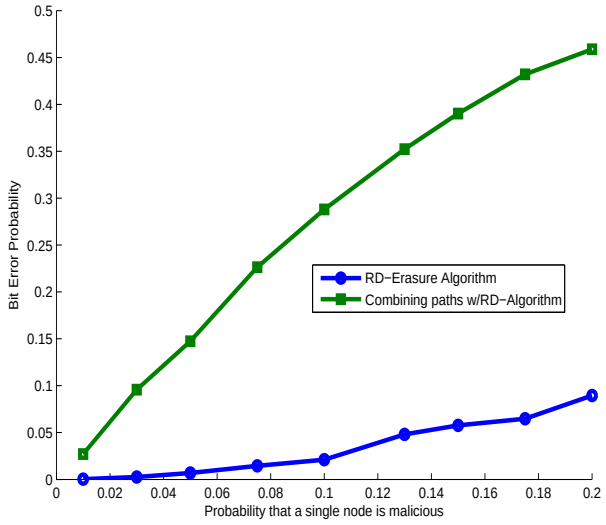

Figure 6.4: Comparing the "combine-paths" technique with RD-erasures algorithm

6.4 Summary

In this chapter we have considered the problem of detecting and localizing malicious nodes in a wireless network. We have shown the application of Reed-Muller codes for finding the adversary nodes. The minimum distance of these codes must be scaled with the expected number of malicious nodes in the network. The code parameters are found based on the probability of a node being malicious. Once the RM-code is known, network paths are derived from the generator matrix of the code and it is shown how the byzantine nodes are located. For the case of non-realizable paths, we derived an algorithm to correct for these paths. We compared the performance of several techniques and showed that our algorithm achieves the lowest bit-error probability.

CHAPTER 7

CONCLUSION

Applications of wireless networks are immense and there are many issues which come with transmitting data through the air (wirelessly). Data transmitted through the air is subject to noise and a variety of attacks. There are many possible approaches and solutions for solving these problems. The common goal between all of these approaches is to guarantee network reliability, network security, and network integrity. The network must have reliability in order to be able to communicate successfully in the presence of adverse conditions. Security in a network refers to preventing false packets from being mistaken for legit ones, making sure that adversaries in the network are not preventing the forwarding of legitimate packets, and keeping the data message secret from an illegitimate user. The integrity of a network is the degree to which a user can trust the network components.

We have three contributions related to wireless network reliability, security, and integrity. Each piece of research assumes a specific network and channel distribution and applies coding theory to find a solution.

7.1 Summary of Contributions

First the network reliability issue is addressed. We assume that the wireless network has multiple paths where each path acts like an erasure channel. Since MDS codes are known to perform well under erasure channels, properties of MDS codes are used to generate algorithms to guarantee a level of success. Two

algorithms are generated, an exponential one (MRAET) and a polynomial one (MRAPT) which take as input the probability of erasure on each path and the lowest allowed probability of successful decoding. Adding redundancy to the message is important for reliability but the higher the redundancy the more bandwidth is used which is undesirable. Thus, the algorithms aim at finding the minimum redundancy and optimal symbol allocation so that this success probability is attained. The symbol allocation determines how many symbols to transmit down each path. MRAET is proved to be optimal for three cases. The performancec of MRAET, MRAPT and the target success probability are compared and shown that the algorithms perform very well. MDS, LT, and Raptor codes are used in the MRAET algorithm and their performance is compared. The performance comparison yields the conclusion that Raptor codes are favorable over the other two.

The next contribution concerns message secrecy in a multiple path wireless network. An eavesdropper (Eve) is spying on the network and views the same multiple paths as the legitimate receiver (Bob) though has a noisier channel. Wyner [40] showed that a message could be kept secret if this was the case. Bob and Eve each have binary erasure channels. We generate a coding and bit allocation method which guarantees message secrecy and reliability. Specifically, it is shown how a Raptor code with carefully picked parameters can result in message secrecy and zero-probability of error as the message size tends to infinity. A technique to determine the number of bits to transmit down each path at each message round is introduced.

Finally we show how to determine the location of an adverse node in a wireless network. We assume that there is standard encryption on the messages

and that the receiver can correctly determine the legitimacy of a packet. The nodes each have a certain probability of being malicious. Paths in the network are formed so that each column of the generator matrix of a Reed-Muller code is represented, where a '1' in spot i of a vector represents node i present on a particular path. Given the node maliciousness probability, it is shown how to pick the Reed-Muller code parameters. The paths required by the Reed-Muller code are not necessarily realizable in the network, so we derive a decoding algorithm for this case. The algorithm treats the missing paths by erasures and goes from there. The performance of this algorithm is compared with several other methods and it can be shown that our algorithm outperforms the other methods.

7.2 Future Work

Possible future work for the reliability of wireless networks is generating a coding scheme for the network without using the assumption of independent paths. This could entail a routing method through the nodes generating node dependent paths and inventing a coding scheme for these dependent paths. A non-routing approach could be taken using network coding and a practical coding method could be developed which assures probability of successful decoding.

Future problems to solve in coding theory for secure wireless communications include developing realistic codes which meet the secrecy and reliability criterion. The scheme we developed in Chapter 5 depends on capacity achieving codes and on the codeword size tending to infinity which is unrealistic in the real world. Usually for security in system, the error-control codes are applied

after the encryption codes instead of a simultaneous encoding, having codes which both correct and secure a message would reduce this to one step. Further work can also consider the secrecy and reliability of a message when applied to different types of main and eavesdropper channels.

Wireless network integrity is an important ongoing research area with many prospective idea suggestions. Some ideas similar to our work are mentioned here. One idea is deriving an error-correction scheme from the network connectivity information. Instead of assuming a particular (n, k) code like in this document which might result in unattainable paths, the paths in the network would be used to determine the best code and parameters to guarantee a high probability of correct localization. Another option includes a probabilistic localization scheme which uses the network connectivity information and the probability of each node being malicious as input to help determine location of recent adversaries. Applying different types of error-correction codes for the localization problem is a simple extension of the authors work which will be their next step.

BIBLIOGRAPHY

[1] B. Awerbuch, D. Holmer, C. Nita-Rotaru and H. Rubens, "An On-Demand Secure Routing Protocol Resilient to Byzantine Failures," ACM Workshop on Wireless Security (WiSe), September 2002.

[2] P. Cataldi, M. P. Shatarski, M. Grangetto, E. Magli, "Implementation and performance evaluation of LT and Raptor codes for multimedia applications," Intelligent Information Hiding and Multimedia Signal Processing, 2006.

[3] Cormen, T.H, Leiserson, C.E, Rivest, R.L, Stein, L. *Introduction to Algorithms*. MIT Press; second edition , 2001.

[4] Thomas M. Cover and Joy A. Thomas, *Elements of Information Theory*.John-Wiley and Sons, Inc. 2nd edition, 2006.

[5] T. Ho, B. Leong, R. Koetter, M. Medard, M. Effros, and D. R. Karger, "Byzantine Modification Detection in Multicast Networks with Random Network Coding," IEEE Transactions onf Information Theory, June 2008.

[6] Chun Huang, Mainak Chatterjee, Wei Cui and Ratan Guha, "Multipath Source Routing in Sensor Networks Based on Route Ranking," Lecture notes in Computer Science: Distributed computing, 2005.

[7] R. Gallager, "Low-Density Parity-Check Codes" dissertation, 1963.

[8] L. M. A. Jalloul, "Multichannel Baseband Processor forWideband CDMA," EURASIP Journal on Applied Signal Processing, 17531768, 2005.

[9] A. Kacewicz and S.B. Wicker. "Optimizing Redundancy Using MDS Codes and Dynamic Symbol Allocation in Mobile Ad Hoc Networks," Conference on Information Sciences and Systems (CISS), Princeton University, March, 2008.

[10] A. Kacewicz, S.B. Wicker, "Secrecy and Reliability using Raptor Codes in the Presence of a Wiretapper in a Multiple Path Wireless Network," IEEE International Conference on Wireless Communications and Signal Processcing (WCSP), Nanjing, China, November 13-15, 2009.

[11] A. Kacewicz, S.B. Wicker, "Redundancy Minimizing Techniques for Robust

Transmission in Wireless Networks," Journal of Communication Networks Special Issue on Secure Wireless Networking, December 2009.

[12] A. Kacewicz, S.B. Wicker, "Application of Reed-Muller Codes for Localization of Malicious Nodes," submitted Sept. 2009.

[13] R. Karp, M. Luby, A. Shokrollahi, "Finite Length Analysis of LT Codes," International Symposium on Information Theory (ISIT), Chicago, June 27-July 2, 2004.

[14] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," ACM Transactions on Programming Languages and Systems, Vol. 4, No. 3, July 1982.

[15] S. J. Lee and M. Gerla, "Split multipath routing with maximally disjoint paths in ad hoc networks," in Proc. Int. Conf. Communications (ICC 2001), 2001, pp. 3201–3205.

[16] K. Lee, H. Radha, and B. Kim, "Kovalenkos Full-Rank Limit and Overhead as Lower Bounds for Error-Performances of LDPC and LT Codes over Binary Erasure Channels," CoRR abs/0901.1762.

[17] M. Luby, M.Mitzenmacher, A. Shokrollahi, D. Spielman, "Efficient erasure correcting codes," IEEE Trans. Inf. Theory, vol. 47, no. 2, pp.569–584, Feb. 2001.

[18] M. Luby, "LT-codes," in Proc. 43rd Annu. IEEE Symp. Foundations of Computer Science (FOCS), Vancouver, BC, Canada, Nov. 2002, pp. 271–280.

[19] David J.C. MacKay, *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press, 2003.

[20] E. Maneva and A. Shokrollahi, "New Model for Rigorous Analysis of LT Codes," ISIT 2006, Seattle, WA, July 2006.

[21] C. N. Mathur , K. Narayan, and K. P. Subbalakshmi, *High Diffusion Cipher: Encryption and Error Correction in a Single Cryptographic Primitive*, Applied Cryptography and Network Security, Springer 2006.

[22] R. J. McEliece, "A Public-key cryptosystem based on algebraic coding theory," DSN Progress Report, Jet Propulsion Laboratory, Pasadena, CA (Jan./Feb. 1978) pp. 114116.

[23] L. H. Ozarow, A. D. Wyner, "Wire-Tape Channel II," Springer, 1998.

[24] V. N. Padmanabhan and D. R. Simon, "Secure Traceroute to Detect Faulty or Malicious Routing,"ACM SIGCOMM Computer Communications, 2003.

[25] Papadimitratos, P., and Haas, Z.J., "Secure Message Transmission in Mobile Ad Hoc Networks,"Ad Hoc Networks,pp.193–209,2003.

[26] M.O. Rabin, Efficient dispersal of information for security, load balancing, and fault tolerance, J. ACM 36 (2) (1989) pp. 335–348.

[27] T. J. Richardson and R. Urbanke, "Efficient encoding of low-density parity-check codes," IEEE Trans. on Info. Theory, vol. 47, no. 2, pp.638656, February 2001.

[28] F. Rubin,"Enumerating all Simple Paths in a Graph," IEEE Transactions on Circuits and Systems, VOL. CAS-25, NO. 8 August, 1978.

[29] I. Sason and R. Urbanke, "Information-Theoretic Lower Bounds on the Performance of Codes on Graphs," ISIT 2003.

[30] C. E. Shannon, "A Mathematical Theory of Communication," Bell System Technical Journal, Vol. 27, pp. 379423, 623656, 1948.

[31] C. E. Shannon,"Communication theory of secrecy systems," B.S.T.J., vol. 28, pp. 656715, Oct. 1949.

[32] Arunkumar Subramanian, Steven W. McLaughlin, "MDS codes on the erasure-erasure wiretap channel," CoRR abs/0902.3286: (2009).

[33] A. Shokrollahi, "Raptor Codes," IEEE Trans. on Information Theory, vol. 52, no. 6, pp.2551–2567, June 2006.

[34] A. Thangaraj, S. Dihidar, A. R. Calderbank, S. McLaughlin, J. Merolla,"Application of LDPC codes to the Wiretap Channel" IEEE Trans. on Information Theory, vol. 53, no. 8, pp. 2933–2945, August 2007.

[35] Tsirigos, A., and Haas, Z.J., "Analysis of Multipath Routing, Part 1: The Effect on the Packet Delivery Ratio," IEEE Trans. on Wireless Communication ,Vol. 3, No. 1, January 2004.

[36] Tsirigos, A., and Haas, Z.J., "Analysis of Multipath Routing, Part 2: Mitigation of the Effects of Frequently Changing Network Topologies,"IEEE Trans. on Wireless Communication ,Vol. 3, No. 2, March 2004.

[37] W. Wang, J. Xu and J. Wang, "Detection and location of malicious nodes based on source coding and multipath transmission in WSN," 11th IEEE International Conference on High Performance Computing and Communications, 2009.

[38] S. B. Wicker. *Error Control Systems for Digital Communication and Storage*. Upper Saddle River,NJ. Prentice Hall, 1995.

[39] S. B. Wicker, V. K. Bhargava (editors), *ReedSolomon Codes and Their Applications*, Piscataway: IEEE Press, 1994.

[40] A. D. Wyner, "The Wire-Tap Channel," B.S.T.J., vol. 54, no. 8, pp. 13551387, Oct. 1975.