

RATELESS ERASURE CODES FOR SHORT  
MESSAGES TRANSMISSION

CHONG ZAN KAI

DOCTOR OF PHILOSOPHY IN ENGINEERING  
SCIENCE

LEE KONG CHIAN FACULTY OF ENGINEERING AND  
SCIENCE  
UNIVERSITI TUNKU ABDUL RAHMAN  
MACRH 2016



**RATELESS ERASURE CODES FOR SHORT MESSAGE  
TRANSMISSION**

By

**CHONG ZAN KAI**

A thesis submitted to the Department of Electrical and Electronic Engineering,  
Lee Kong Chian Faculty of Engineering and Science,  
Universiti Tunku Abdul Rahman,  
in partial fulfilment of the requirements for the degree of  
Doctor of Philosophy in Engineering  
March 2016

To my beloved wife, daughter and family.

## ABSTRACT

Rateless erasure code is a type of error-correction code for erasure channel. Given a message of  $k$  symbols, it generates potentially infinite number of encoded symbols, which enables the receiver to reconstruct the original message from any  $k(1+\epsilon)$  coded symbols with high probability of complete decoding (PCD), i.e., 99.9% success probability, where  $\epsilon$  denotes the decoding inefficiency.

Generally, the network traffic is dominated by short messages. The state-of-art rateless erasure codes – LT code and Raptor code work efficiently only for long messages. In response, this thesis proposes the rateless erasure codes that are efficient in transmitting short messages. Our studies show that the Random code – a rateless erasure code with generator matrix of randomly distributed 0 and 1, is able to reconstruct both short and long messages from any  $k+10$  coded symbols with high PCD even for small  $k$ . Utilising the invariant PCD of Random code over the message length, we propose Micro-Random code with improved high PCD using only  $k+1$  encoded symbols.

We also propose two pseudo-random codes that have better decoding complexities, i.e., systematic Random code and Stepping-Random (SR) code. If the first  $k$  encoded symbols are received intact, systematic Random code is able to reconstruct the original message with negligible decoding complexity while SR code requires  $O(k)$  instead. It is to be noted that, systematic

Random code is only suitable for point-to-point and point-to-multipoint transmissions while SR code works even in multipoint-to-point transmission.

## **ACKNOWLEDGEMENT**

I wish to express my gratitude to my supervisor, Prof. Goi Bok Min, co-supervisor, Prof. Ewe Hong Tat, mentors Prof. Hiroyuki Ohsaki, and Dr. Bryan Ng for their guidance. Their invaluable support and supervision have enabled the timely completion of this work. I would like to extend my sincere appreciation to many friends and lab members, to whom we have gone through many difficulties for the past few years. Last but not least, I would like to thank my parents for their understanding, my beautiful wife, Sophia and my lovely daughter, Ning. They are my motivation to complete the work.

## APPROVAL SHEET

This thesis entitled “**RATELESS ERASURE CODES FOR SHORT MESSAGE TRANSMISSION**” was prepared by CHONG ZAN KAI and submitted as partial fulfilment of the requirements for the degree of Doctor of Philosophy in Engineering at Universiti Tunku Abdul Rahman.

Approved by:

---

(Prof. Ir. Dr. GOI BOK MIN)                      Date:.....  
Professor/Supervisor  
Department of Electrical and Electronic Engineering  
Lee Kong Chian Faculty of Engineering Science  
Universiti Tunku Abdul Rahman

---

(Prof. Dr. EWE HONG TAT)                      Date:.....  
Professor/Co-supervisor  
Department of Internet Engineering and Computer Science  
Lee Kong Chian Faculty of Engineering Science  
Universiti Tunku Abdul Rahman



## DECLARATION

I hereby declare that the thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UTAR or other institutions.

Name \_\_\_\_\_  
(CHONG ZAN KAI)

Date \_\_\_\_\_

## TABLE OF CONTENTS

	<b>Page</b>
<b>ABSTRACT</b>	<b>iii</b>
<b>ACKNOWLEDGEMENT</b>	<b>v</b>
<b>APPROVAL SHEET</b>	<b>vi</b>
<b>DECLARATION</b>	<b>vii</b>
<b>LIST OF CONTENTS</b>	<b>viii</b>
<b>LIST OF TABLES</b>	<b>xii</b>
<b>LIST OF FIGURES</b>	<b>xiv</b>
<b>LIST OF ABBREVIATIONS</b>	<b>xviii</b>
<b>CHAPTER</b>	
<b>1.0 INTRODUCTION</b>	<b>1</b>
1.1 Error-Correction Code and Erasure Code	1
1.2 Emergence of Rateless Erasure Code	2
1.3 The Impact of Rateless Erasure Code to Network Revolution	4
1.3.1 Transmission Control Protocol (TCP)	4
1.3.2 Point-to-Multipoint and Multipoint-to-Point Transmissions	6
1.4 Rateless Erasure Code and Network Coding	7
1.5 Research Problem	8
1.6 Contribution and Outline	10
<b>2.0 LITERATURE REVIEW</b>	<b>13</b>
2.1 Fixed Rate Erasure Code	14
2.1.2 Low Density Parity Check (LDPC) Code	14
2.1.3 Tornado Code	16
2.2 Rateless Erasure Codes	18
2.2.1 Luby Transform (LT) Code	19
2.2.1.1 Encoding and Decoding Processes	19
2.2.1.2 Degree Distribution	22
2.2.2 Raptor Code	24
2.2.3 RaptorQ Code	25

2.3	Other Rateless Erasure Code for Short Message Transmission	27
2.5	Summary	29
<b>3.0</b>	<b>RANDOM CODE</b>	<b>31</b>
3.1	Random Matrix as Rateless Erasure Code	31
3.1.1	Linear System and Rateless Erasure Code	31
3.1.2	Random Code	32
3.1.3	An Example of Encoding and Decoding Processes	35
3.2	Probability of Complete Decoding (PCD) for Long Messages	37
3.3	Probability of Complete Decoding (PCD) for Short Messages	39
3.4	Fixed Weight Random Code – Windowed Code	44
3.5	The Challenges of Random Code	46
<b>4.0</b>	<b>MICRO-RANDOM CODE</b>	<b>47</b>
4.1	High Decoding Inefficiency in Transmitting Short Message	47
4.2	Micro-Random Code	48
4.2.1	Symbol Dimensioning in Brief	48
4.2.2	Improving Decoding Inefficiency with Micro Symbols	48
4.3	Performance Analysis	52
4.3.1	Probability of Complete Decoding (PCD)	53
4.3.2	Expected Extra Coded Symbols	54
4.3.3	Decoding Steps and Decoding Complexity	57
4.4	Simulation Results	57
4.5	Deploying Micro-Random Code in Resource-Constrained Devices	61
4.5.1	Experiment Setting	62
4.5.2	Experiment Result	63
4.6	Summary	64

<b>5.0</b>	<b>SYSTEMATIC RANDOM CODE</b>	<b>66</b>
5.1	Systematic Rateless Erasure Code	66
5.2	Systematic Random Code	67
5.2.1	Encoding Message	67
5.2.2	Message Reconstruction	68
5.3	Performance Analysis	71
5.3.1	Probability of Complete Decoding	71
5.3.2	Decoding Algorithm of Random code	73
5.3.3	Decoding Algorithm of SYSR Code	76
5.4	Numerical Result	79
5.5	Summary	82
<b>6.0</b>	<b>STEPPING-RANDOM CODE</b>	<b>84</b>
6.1	Non-Systematic Pseudo-Random Code	84
6.1.1	Issues of SYSR Code in Multipoint-to-Point Transmission	84
6.1.2	Stepping Code	86
6.1.3	Stepping-Random (SR) Code	87
6.2	Message Reconstruction	88
6.2.1	Decoding in Ideal Channel	89
6.2.2	Decoding in Lossy Channel	90
6.2.3	Decoding Complexity	93
6.2.4	Decoding in Multipoint-to-Point Transmission Scenario	94
6.3	Numerical Result	95
6.4	Summary	102
<b>7.0</b>	<b>RATELESS ERASURE CODES IN GF(2<sup>8</sup>)</b>	<b>106</b>
7.1	Random Code in GF(2 <sup>q</sup> )	106
7.2	Micro-Random Code in GF(2 <sup>q</sup> )	110
7.3	Systematic Micro-Random Code and Stepping-Micro- Random Code in GF(2 <sup>8</sup> )	113
7.4	Conclusion	116

<b>8.0</b>	<b>CONCLUSION AND FUTURE WORK</b>	<b>117</b>
8.1	Probability of Complete Decoding (PCD) and Overhead Symbols	117
8.2	Decoding Complexity	119
8.3	Transmission Scenarios	119
8.4	Conclusion	120
8.5	Future Work	121
8.5.1	Transmitting Long Message with Short Rateless Erasure Code	121
8.5.2	Protocol Design	122
8.5.2.1	Computational Specification	122
8.5.2.2	Delay Acknowledgement	123
	<b>BIBLIOGRAPHY</b>	<b>124</b>
	<b>ACHIEVEMENT</b>	<b>132</b>

## LIST OF TABLES

Table		Page
3.1	The probability (CDF and PMF) for a random matrix to achieve complete decoding with $m$ additional rows.	40
3.2	The probabilities of having the last ten rank numbers for $k-m=20$ and $k-m=30$ and $k$ is fixed as 30.	42
4.1	The PCD of micro Random code, $Q_{\text{mRC}}(m, \alpha)$ for various $m$ and $\alpha$ .	56
4.2	The PMF of micro-Random code for various overhead symbols $m$ , segmentation factor $\alpha$ and the expected value.	57
5.1	The PCD of SYSR code for $k=10$ and 50	76
5.2	The expected overhead symbols to achieve complete decoding in SYSR code for increasing $k$ and $\rho$ .	76
7.1	The failure probabilities of Micro-Random code at various $\text{GF}(2^q)$ and $m$ overhead symbols using $\alpha=10$ .	115
7.2	The failure probabilities of $\text{GF}(2^8)$ Micro-Random code at various $\alpha$ and overhead symbols, $m$ .	116
7.3	The PMF and the mean of $\text{GF}(28)$ Micro Random code at various $\alpha$ .	118
7.4	The failure probabilities of systematic Micro-Random code and Stepping-Micro-Random code for $m=0$ to 3 in channel of various erasure probabilities.	120
8.1	The total required coded symbols to achieve high PCD for each of the coding scheme.	124

8.2	The average coded symbols to achieve complete decoding for each of the coding schemes.	124
8.3	The decoding complexity for each coding scheme in various channel conditions.	125
8.4	The applicability for each coding scheme in various transmission scenarios.	126

## LIST OF FIGURES

Figures		Page
1.1	BEC.	1
1.2	The encoding and decoding processes of a typical erasure codes.	2
1.3	The encoding and decoding processes of rateless erasure codes.	4
1.4	TCP flow diagram.	5
1.5	(a) Point-to-multipoint transmission and (b) multipoint-to-point transmission.	7
1.6	Distribution of flow size of CAIDA and CERNET.	10
2.1	Category of erasure codes.	14
2.2	The message-passing algorithm reconstructs the erased bits in two iterations.	16
2.3	(a) Encoding process and (b) decoding processes process of Tornado code.	18
2.4	Tornado code in bipartite graph of multiple cascaded layers.	19
2.5	Example of LT decoding process.	22
2.6	Ideal Soliton distribution for $k=100$ .	24
2.7	Robust Soliton distribution for $k=100$ .	25
2.8	Bipartite graph of Raptor code.	26



2.9	The failure probabilities of RaptorQ code at zero overhead symbol for (a) $\rho=0.1$ and (b) $\rho=0.5$ .	28
2.10	The failure probabilities of RaptorQ code at one overhead symbol for (a) $\rho=0.1$ and (b) $\rho=0.5$ .	28
2.11	The failure probabilities of RaptorQ code at two overhead symbol for (a) $\rho=0.1$ and (b) $\rho=0.5$ .	28
4.1	The encoding process of Micro-Random code.	52
4.2	The decoding process of Micro-Random code.	54
4.3	PCD for messages of (a) $k=10$ and (b) $k=100$ symbols for various extra coded symbols and segmentation factor $\alpha$ .	62
4.4	The expected number of extra coded symbols for messages of $k=10$ symbols to achieve complete decoding for various segmentation factor $\alpha$ .	63
4.5	The average decoding time needed to reconstruct the original message that encoded with Micro-Random code.	64
4.6	The screen-shot of QPython running in Alcatel OT.	66
4.7	The average decoding time of micro-Random code running on Android devices for messages of (a) 2,000 bytes, and (b) 5,000 bytes.	68
5.1	The encoding and decoding processes of systematic rateless erasure code.	70
5.2	The flow chart of Random code to form upper triangular matrix.	79
5.3	The flow chart for SYSR code to form upper triangular matrix.	81

5.4	The PCD of Random code, SYSR code and SR code for messages of symbols in channels of various erasure probabilities.	84
5.5	The average overhead symbols for SYSR code to achieve complete decoding in channels of various erasure probabilities.	85
5.6	The average XOR row operations for Random code, SYSR code and SR code to form (a) upper triangular matrix, and (b) backward substitution in channels of increasing erasure probabilities.	87
6.1	Multipoint-to-point transmission.	89
6.2	The performance of Random code, Windowed code and SR code in the channels of channel erasure probability $\rho=0.00$ for message of $k=100$ symbols.	101
6.3	The performance of Random code, Windowed code and SR code in the channels of channel erasure probability $\rho=0.01$ for message of $k=100$ symbols.	101
6.4	The performance of Random code, Windowed code and SR code in the channels of channel erasure probability $\rho=0.02$ for message of $k=100$ symbols.	102
6.5	The performance of Random code, Windowed code and SR code in the channels of channel erasure probability $\rho=0.03$ for message of $k=100$ symbols.	102
6.6	The performance of Random code, Windowed code and SR code in the channels of channel erasure probability $\rho=0.04$ for message of $k=100$ symbols.	103
6.7	The performance of Random code, Windowed code and SR code in the channels of channel erasure probability $\rho=0.05$ for message of $k=100$ symbols.	103

6.8	The performance of Random code, Windowed code and SR code in the channels of channel erasure probability $\rho=0.5$ for message of $k=100$ symbols.	104
6.9	The performance of Random code, Windowed code and SR code in the channels of channel erasure probability $\rho=0.7$ for message of $k=100$ symbols.	105
6.10	The performance of SR code in the channels of channel erasure probability $\rho=0.5$ for messages of length 100, 300, 500 and 1000 symbols.	105
6.11	Total message symbols that are reconstructed with sequential addition and Gaussian elimination in channels of various of erasure probabilities.	106
6.12	The achievable rankness by selecting $k$ Part I coded symbols randomly from various number of sources for messages of (a) $k=10$ (b) $k=50$ symbols.	109
6.13	The PCD of randomly selected $k+10$ Part I-II coded symbols from various number of sources.	110
7.1	The failure probabilities of $GF(2^q)$ random matrices at various additional rows.	114
7.2	The failure probabilities of $GF(2^q)$ Micro-Random code at increasing number of overhead symbols.	116
7.3	The failure probabilities of $GF(28)$ micro Random code at various $\alpha$ and overhead symbols, $m$ .	117
7.4	The failure probabilities of $GF(28)$ systematic Micro-Random code and Stepping-Micro-Random code at overhead symbols of $m=0$ to 3 in channels of various erasure probabilities.	121

## LIST OF ABBREVIATIONS

BEC	Binary erasure channel
BSC	Binary symmetric channel
CDF	Cumulative density function
CRC	Cyclic redundancy check
MR Code	Micro-Random Code
PCD	Probability of complete decoding
PMF	Probability mass function
SMR Code	Stepping-Micro-Random Code
SR Code	Stepping-Random code
SYSR Code	Systematic Random Code
SYSMR Code	Systematic Micro-Random Code
TCP	Transmission control protocol

# CHAPTER 1

## INTRODUCTION

This chapter introduces the rateless erasure codes and their impact to the network evolution. Then, issues will be discussed before forming the research problem. The thesis outline and contributions will be presented at the end of the chapter.

### 1.1 Erasure Code

Binary erasure channel (BEC) is a communication model, where each input bit has an equal probability  $\rho$  of erasure as shown in Figure 1.1 and  $\rho$  is also known as the *channel erasure probability*. Such channel is common in the Internet, where the packets are dropped due to Cyclic Redundancy Code (CRC) checking failure (packet error) or network congestion.

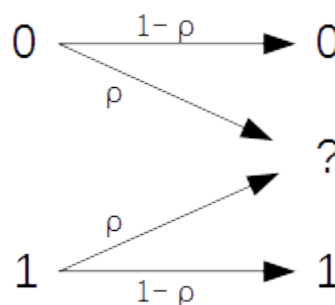


Figure 1.1: BEC.

Generally, the *erasure code* is an error-correction code for BEC. Given a message of  $k$  symbols (symbol is a sequence of bits), the erasure code generates  $n$  encoded symbols, where  $n > k$ . Correspondingly, one can reconstruct the original message with any  $k$  out of  $n$  encoded symbols, irrespective of the sequence as shown in Figure 1.2. Note that the blank circles

in the figure denote the message symbols and the hatched circles are the encoded symbols. The ratio  $k/n$  is also known as the code rate.

Erasure codes such as Reed-Muller code have been deployed in spacecraft and Reed-Solomon code in compact discs and digital communication (Reed, 2000; Wicker and Bhargava, 1999). For the past decades, the application of erasure codes has been studied in computer networks (Rizzo, 1997; McAuley, 1990), wireless sensor networks (Kim et al., 2004; Wen et al., 2007) and other wireless networks (Wang et al., 2005), etc.

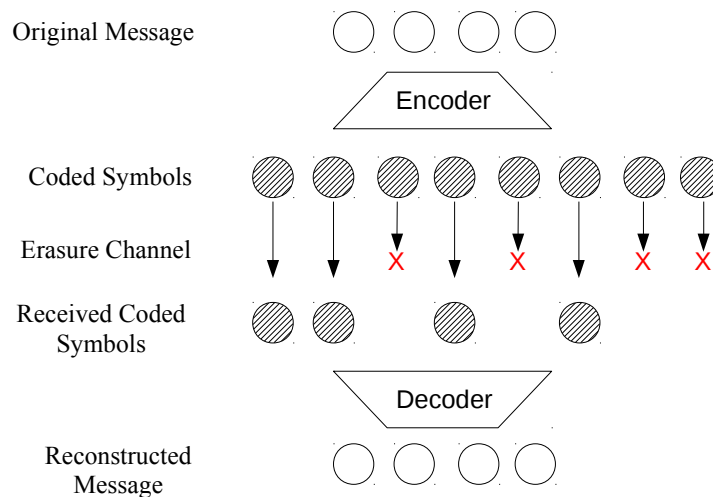


Figure 1.2: The encoding and decoding processes of typical erasure codes.

## 1.2 Emergence of Rateless Erasure Codes

Generally, the erasure codes in Section 1.1 do not generate the encoded symbols dynamically. Given a message of  $k$  symbols, the sender needs to pre-determine the total encoded symbols  $n$  to generate wisely such that at least  $k$  encoded symbols are received by the receiver. Basically, addressing the problem with a large  $n$  value is undesirable as it wastes the computational resource if the channel erasure probability  $\rho$  is overestimated.

Moreover, the estimation is difficult in wireless networks, where  $\rho$  may change drastically over a short time.

To address the inflexibility in erasure codes, Byers et al. (1998) proposed *rateless* erasure codes (in the name of digital fountain codes), where a message of  $k$  symbols is encoded into a potentially infinite number of encoded symbols. Then, the receiver reconstructs the original message from any  $k(1+\epsilon)$  encoded symbols, where  $\epsilon \in \mathbb{R}$  denotes the decoding inefficiency and  $\epsilon \geq 0$  as shown in Figure 1.3. As compared to the erasure codes in Section 1.1, the correct estimation of the channel erasure probability is unnecessary. The sender may keep generating the encoded symbols until the receiver has sufficient encoded symbols (i.e.,  $k(1+\epsilon)$ ) to reconstruct the original message.

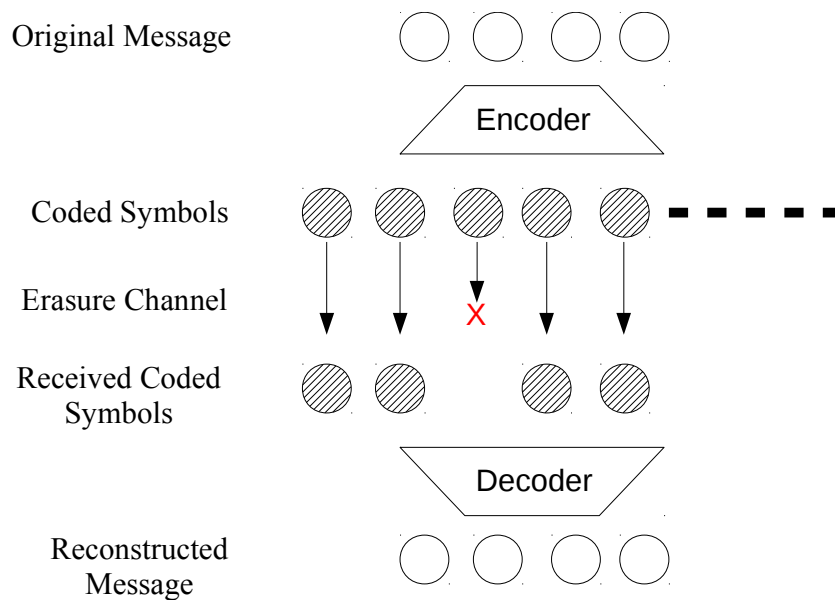


Figure 1.3: The encoding and decoding processes of rateless erasure codes.

### **1.3 The Impact of Rateless Erasure Codes to Network Revolution**

This section discusses the varying impacts of rateless erasure codes to the networks in brief.

#### **1.3.1 Transmission Control Protocol (TCP)**

Transmission Control Protocol (TCP) is a common transport layer protocol in the Internet. It ensures the reliable packets delivery based on the following control mechanisms:

- Flow control – it limits the transmission rate based on advertised window.
- Sequence control – it ensures the packets are delivered in order based on packet acknowledgement.
- Error control – it detects the error packets based on checksum and lost packets are retransmitted.
- Congestion control – it adapts to the available capacity with appropriate transmission rate.



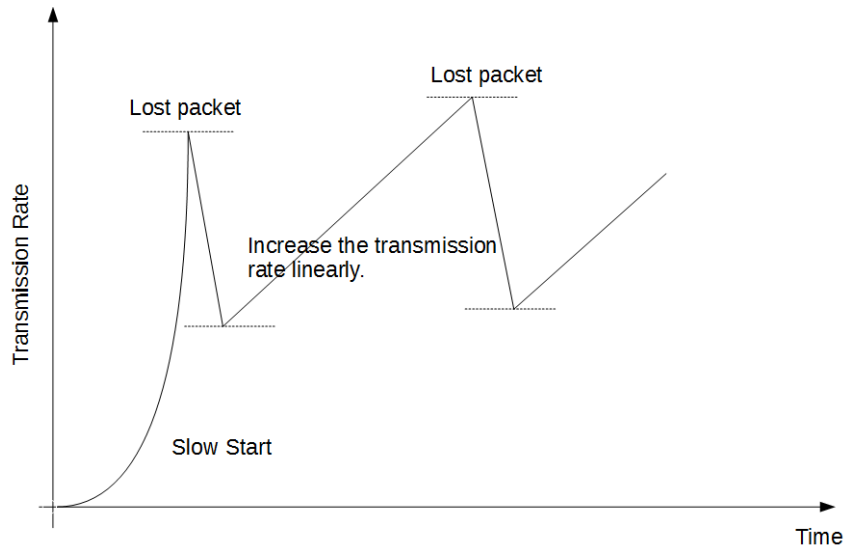


Figure 1.4: TCP flow diagram.

Figure 1.4 explains the transmission mechanism of TCP. Initially, the sender sends the message at a minimum rate. Whenever a packet is sent, an acknowledgement is expected from the receiver as the signal of successful delivery and then the transmission rate will be increased. Any unacknowledged packet will be retransmitted after the time-out period with the transmission rate halved. This process continues until all the packets have been delivered successfully. Note that all the packets must be delivered in order sequence. Any out-of-sequence packet is considered a packet loss event and the sender has to retransmit from the last acknowledged packet.

The aforementioned transmission mechanism can be simplified with the emergence of rateless erasure codes. Instead of identifying and retransmitting the lost packets, the rateless erasure codes enable TCP to reconstruct the message from any packets irrespective of the sequence, as demonstrated in Botos et al. (2010), Molnár et al. (2013), Molnár et al. (2014) and Móczár et al.(2014).

Generally, TCP attempts to utilise the available bandwidth without overwhelming the networks with excessive packets. Taking a different perspective, Raghavan and Snoeren (2006) argue that network congestion may not be as bad as we think. By assuming the existence of perfect rateless erasure codes, the authors suggest that every user will have fair use of bandwidth and no complicated congestion avoidance algorithm is required.

The idea is further studied by Bonald et al. (2009), Chong et al. (2012a, 2012b) and even adopted by Global Environment for Network Innovations (GENI) in designing future network architecture (Clark et al., 2007). Their counter-intuitive results dispute the common belief that a network will fall into congestion collapse without congestion control.

### **1.3.2 Point-to-Multipoint and Multipoint-to-Point Transmissions**

Consider the case, where a sender intends to transmit a message to a large group of receivers in the networks as shown in Figure 1.5(a). Since each channel has different erasure probability, the sender will be flooded with acknowledgements from the receivers for different sequence of packets. Such phenomenon is commonly known as *feedback explosion*. Rateless erasure codes address this issue effectively as demonstrated by Abdullah et al. (2011), Mammi et al. (2011), Du et al. (2013) and Abdullah et al. (2013).

Additionally, rateless erasure codes also enable a message to be received from multiple sources at the same time as shown in Figure 1.5(b) (Zhu et al. 2010; Qadri et al., 2010; Bursalioglu et al.,2011).

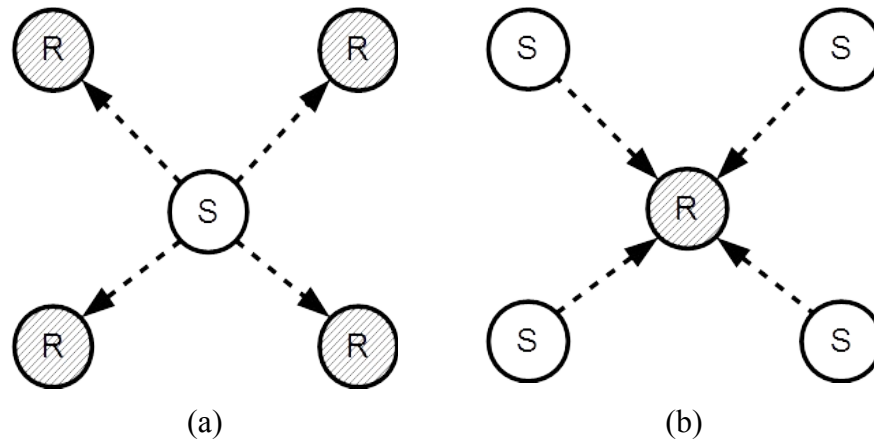


Figure 1.5: (a) Point-to-multipoint transmission and (b) multipoint-to-point transmission.

#### 1.4 Rateless Erasure Codes and Network Coding

Ahlsweede et al. (2000) presented a perspective-changing seminar article detailing network coding. Generally, both rateless erasure codes and network coding share many similarities as they provide reliable data transfer with coding theory. Rateless erasure codes require nodes at only two ends (i.e., sender and receiver) to perform the encoding and decoding processes. On the other hand, network coding needs intermediate nodes, source and sink to run the coding operations. Due to that, network coding is found naturally in networks that require persistent cooperation among the nodes, e.g., wireless sensor networks (Ou et al., 2012; Rout and Ghosh, 2013), wireless ad hoc networks (Mehta and Narmawala, 2012; Ebrahimi-Ghiri and Keshavarz-Haddad, 2013), delay-tolerant networks (Zeng et al., 2012; Sheu et al., 2013), etc.

Although the development of the Internet is much influenced by the end-to-end argument of Saltzer et al. (1984) (i.e., the network design should be kept simple while leaving the core functions at the both ends), the dispute has never stopped and we need a better Internet that meets the challenge of future demand (Blumenthal et al., 2001; Moors, 2002). Many research projects have been carried out to study the clean slate design of the future Internet (Feldmann, 2007; Pan et al., 2011; Fisher, 2014), and resulted in the development of future Internet testbeds like GENI (Berman et al., 2014) and FIRE (Schwerdel et al., 2014), Information-Centric Networking (ICN) (Ahlgren et al., 2012; Xylomenos et al., 2013), network encoded TCP (Sundararajan et al., 2011; Kim et al., 2011; Chen et al., 2012), etc.

No one knows if the future Internet will be totally different from today's host-centric communication model. Therefore, in this thesis we use rateless erasure codes in providing reliable data transmission, rather than using the network coding method.

## **1.5 Research Problem**

We review the existing characteristics of the current network traffic before forming the research problem.

Generally, the network traffic characteristics change as computer technology advances. Zhang and Qiu (2000) observe that the Internet traffic is

dominated by small messages of 10-20 kbytes, but the large portion of bandwidth is consumed by traffic of long messages, which is in the minority. After two years, Brownlee et al. (2002) discovered that the average message size has increased to 50 kbytes. Following that, Benson et al. (2010) discovered that 80% of the messages in the data centres are smaller than 10 kbytes. Most of the packet size are either in the range of 200 bytes or 1.4 kbytes.

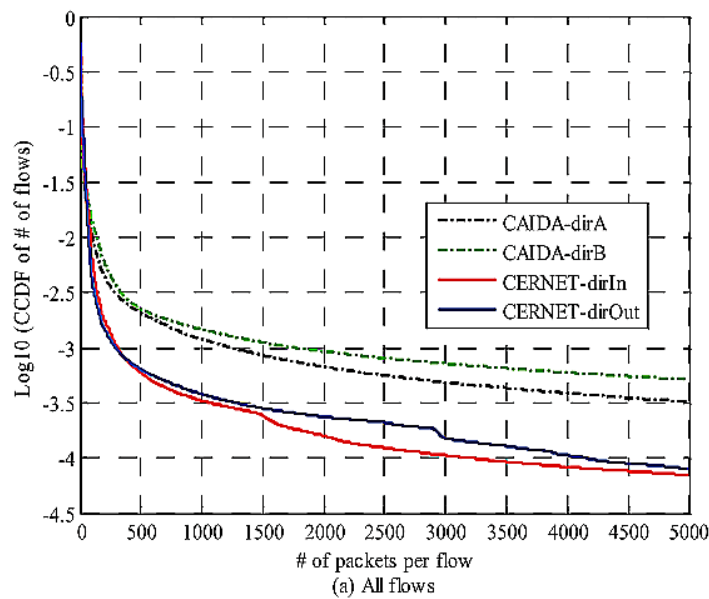


Figure 1.6: Distribution of flow size of CAIDA and CERNET from Zhang and Ding (2012).

Recently, Zhang and Ding (2012) discovered that 90% of the messages in China Education and Research Network (CERNET) are not larger than 7 kbytes and 80% of them less than 10 packets as shown in Figure 1.6. Additionally, 90% of messages are less than 2 kbytes in the Center for Applied Internet Data Analysis (CAIDA) and 80% of them are not larger than 5 packets.

We learn two things from the aforementioned observations on network characteristics. First, the Internet traffic changes insignificantly for the past 10 years. Second, the majority of the messages are transmitted in less than 10 packets.

We assume that the symbol size can occupy the maximum payload of the packet and the symbol size of  $l$  bits implies the symbol is the concatenation of multiple of 1 bit. For clarity, we consider messages of less than 500 symbols as short messages and those greater as long messages in the thesis. Generally, the state-of-the-art rateless erasure codes (e.g., LT code and Raptor code in Section 2.2) require short overhead symbols (i.e.,  $k \epsilon$ ) in transmitting long messages and they cater to the needs of a minor group of network users. Hence, we define the research problem of the thesis as the following:

*“To design  $GF(2)$  rateless erasure codes that achieve high PCD with short overhead symbols.”*

## **1.6 Contribution and Outline**

In this chapter, we have introduced rateless erasure codes and discussed their potential to revolutionise the traditional reliable transmission approach. Nonetheless, rateless erasure codes are not widely deployed in the Internet due to the state-of-the-art rateless erasure codes being only efficient in transmitting long messages and the majority of the network traffic are short messages instead.

To highlight the research problem, we review the state-of-the-art rateless erasure codes for both long and short messages in Chapter 2. Most of the short rateless erasure codes are derived from the LT code, in which they obtain better performance by optimising the degree distribution.

Taking a different approach, we utilise the mathematical properties of Random code in designing short rateless erasure codes. Chapter 3 introduces Random code, a rateless erasure code with a generator matrix of random distributed binary values. Given a message of  $k$  symbols, Kolchin's theorem shows that Random code (random matrix) is able to achieve a high *probability of complete decoding* (PCD), i.e., a 99.9% success probability to reconstruct the original message with  $k+10$  encoded symbols for a long message. To demonstrate the applicability in transmitting short messages, we prove that the high PCD persists for short message as well. We will use these theorems to construct short rateless erasure codes in the later chapters.

The unique decoding capability of Random code appears to be inefficient for a very short message, e.g.,  $k=10$  symbols. In this case, it requires  $k+10=20$  encoded symbols to achieve high PCD – that is double of the total original message symbols. To address the issue, Chapter 4 proposes *Micro-Random code*, a variation of Random code that achieves high PCD with  $k+1$  encoded symbols by dimensioning the symbol size appropriately before and after the encoder and decoder. Such gain is obtained with the trade-off of high decoding complexity and more decoding steps.

In order to simplify the decoding complexity, Chapter 5 proposes the *systematic Random code* that reconstructs original message with negligible decoding complexity when the first  $k$  encoded symbols are received intact. High PCD can be obtained with  $k+10$  encoded symbols in lossy channel.

Chapter 6 proposes *Stepping-Random (SR) Code* that works in point-to-point, point-to-multipoint and multipoint-to-point transmission scenarios. It has the decoding complexity of  $O(k)$  if the first  $k$  encoded symbols are received intact. Like systematic Random code, SR code requires a total of  $k+10$  encoded symbols in lossy channel with decoding complexity of  $O(k^3)$ .

In Chapter 7, we extend the proposed GF(2) rateless erasure codes to higher order in order to compare with RaptorQ code, a variant of Raptor code that is built on the hybrid of GF(2) and GF( $2^8$ ). Note that, the focus of the thesis is still on GF(2) and this chapter demonstrates the strength of our rateless erasure codes in higher order finite field.

In Chapter 8, we discuss the performance of the proposed GF(2) rateless erasure codes from different aspects. Finally, we draw a conclusion and discuss the future work.



## CHAPTER 2

### LITERATURE REVIEW

This chapter reviews the fixed rate erasure codes and the state-of-the-art rateless erasure codes that are efficient in transmitting short messages or long messages, or both. The corresponding category can be found in Figure 2.1.

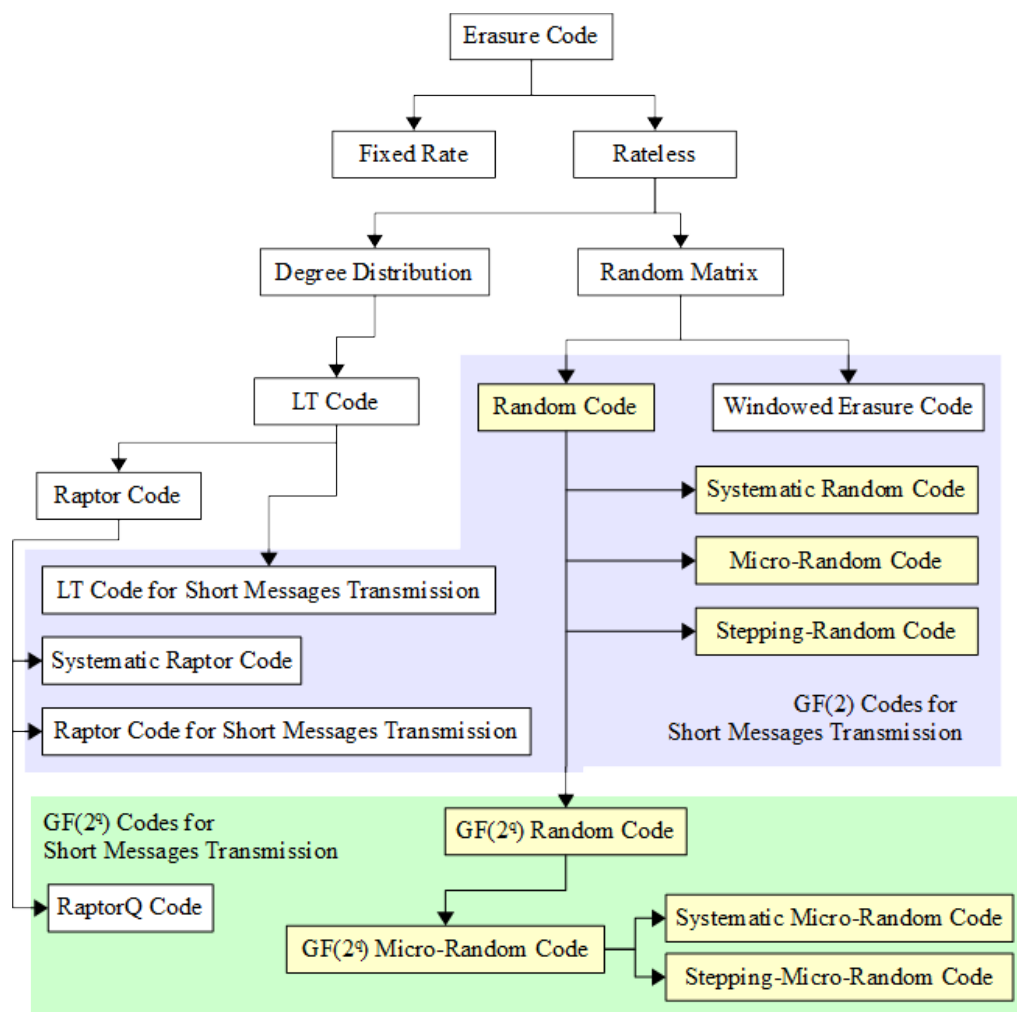


Figure 2.1: Category of erasure codes.

## 2.1 Fixed Rate Erasure Codes

Given a message of  $k$  symbols, a fixed rate erasure code generates  $n$  encoded symbols, where  $n > k$ . Then, the original message can be reconstructed with any  $k$  out of  $n$  encoded symbols and  $n - k$  is also known as overhead symbols. As the number of generated encoded symbols,  $n$  has to be pre-determined, this type of erasure codes is also known as *fixed rate* erasure codes. Generally, a *systematic* erasure code uses the original message symbols as part of the encoded symbols. Otherwise, it is called *non-systematic* erasure code.

### 2.1.1 Low Density Parity Check (LDPC) Code

Low-Density Parity-Check (LDPC) code is an error-correction code that was first proposed by Gallager (1962). His coding scheme, which is known as Gallager Code, was found to be impractical to the computational technologies at that time. The code has been neglected for decades until McKay and Neal (1995) rediscovered it as LDPC code.

Generally, LDPC code contains a very low ratio of non-zero entries. The sparsity enables LDPC code to reconstruct a message with message-passing algorithm, where the decoding complexity is linear with increased message size,  $k$ . An example of reconstructing a message with message-passing algorithm is presented in a Tanner graph in Figure Error: Reference source not found (a). The *message bits* are the circles at the bottom, where three of them are erased (denoted as  $e$ ). The *constraints* are represented by

the squares at the top of the figure, and they constrain the summation of all the connected nodes to be value 0.

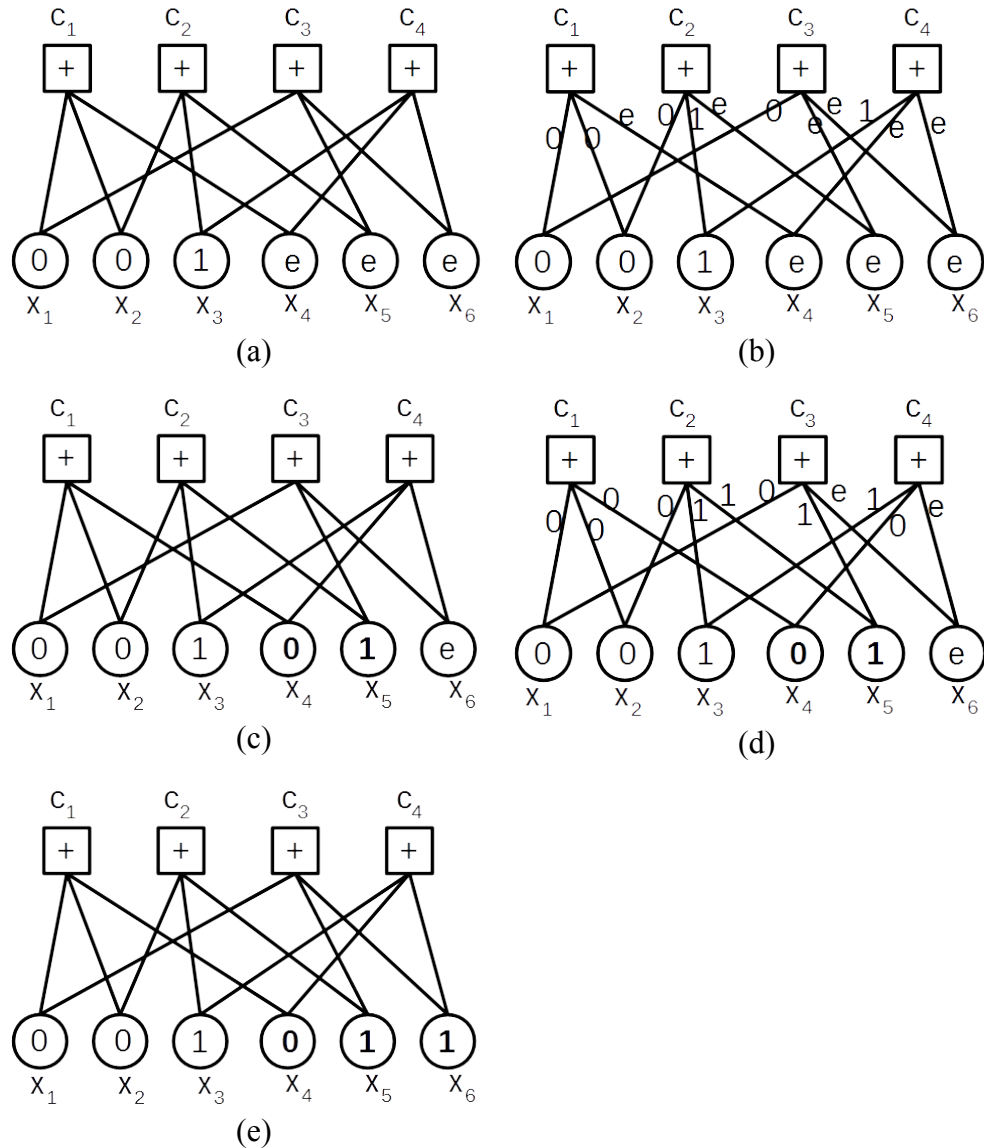


Figure 2.2: The message-passing algorithm reconstructs the erased bits in two iterations.

Initially, all message bits send their respective values to each of their connected constraints (Figure 2.2 (b)). The values are either 0, 1 or  $e$ . If a constraint receives only a single  $e$  among the neighbouring message bits, then the value of  $e$  can be determined instantly. For example, the constraint

$c_1$  receives  $(0,0,e)$  from the connected message bits  $(x_1,x_2,x_4)$ . Therefore, it deduces the value of  $x_4$  to be 0 (Figure 2.2 (c)). The same principle applies to  $c_2$ , where it determines  $x_5$  to be 1 based on the values from  $x_2$  and  $x_3$ . Note that the value of  $x_6$  cannot be determined in current state because  $c_4$  receives two  $e$  from  $x_3$  and  $x_5$ . The process is repeated until all the erased bits are recovered, or maximum iterations have been attempted.

### 2.1.2 Tornado Code

Tornado code (Luby et al., 2001) is the preliminary design of the state-of-the-art rateless erasure code – Luby Transform (LT) Code, which we will review in a later section. The encoding and decoding processes of Tornado code can be explained with the bipartite graph that is shown in Figure 2.3. The  $k$  circles on the left columns are the message bits and  $\beta k$  check bits are shown on the right hand side, where  $0 < \beta < 1$ . Note that the check bits are presented by square symbols in some papers.

The encoding and decoding processes is similar to the message-passing algorithm in Section 2.1.1. Set each check bit to be the XOR of all the neighbouring message bits. Then, all the message bits and the check bits are sent to the receiver. For example, in Figure 2.3(a) the value of  $c_1$  is the sum module-2 of  $x_1$  and  $x_2$  and the missing bit  $x_3$  can be reconstructed with  $c_1 + x_1 + x_2$  as shown in Figure 2.3(b).

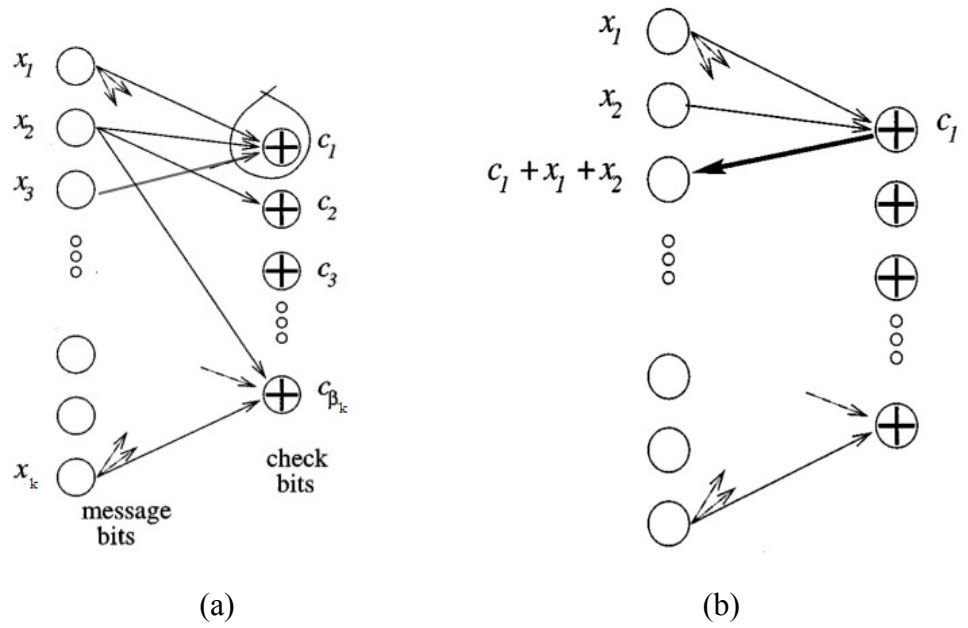


Figure 2.3: (a) Encoding process and (b) decoding process of Tornado code. Figure from Luby et al. (2001).

Tornado code employs multi-layer bipartite graph as shown in Figure 2.4. Generally, the encoder generates  $\beta k$  check bits from  $k$  message bits, denoting them as layer-1 check bits. Then, the process is repeated again to generate  $\beta^2 k$  check bits (layer 2) from the layer 1 check bits. The process is repeated continuously until the final layer, where a conventional erasure code is used.

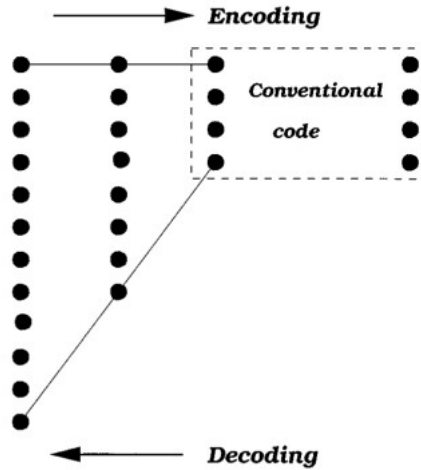


Figure 2.4: Tornado code in bipartite graph of multiple cascaded layers (Figure from Luby et al. 2001).

## 2.2 Rateless Erasure Codes

The fixed rate erasure codes in Section 2.1 require prior knowledge of the channel condition such that the sender generates sufficient encoded symbols to the receiver. Byers et al. (1998) proposed the idea of rateless erasure codes, where a potentially infinite number of encoded symbols are generated from a message of  $k$  symbols. Then, one may reconstruct the original message from any  $n$  encoded symbols with high success probability, where

$$n = k(1 + \epsilon), \quad (1)$$

$\epsilon$  denotes the decoding inefficiency and  $k\epsilon$  the overhead symbols. The state-of-the-art rateless erasure codes are presented in the following section.

## 2.2.1 Luby Transform (LT) Code

Luby (2002) proposed the first practical rateless erasure (digital fountain code), namely Luby Transform (LT) code. It has a well designed degree distribution that optimises the encoding and decoding complexities.

### 2.2.1.1 Encoding and Decoding Processes

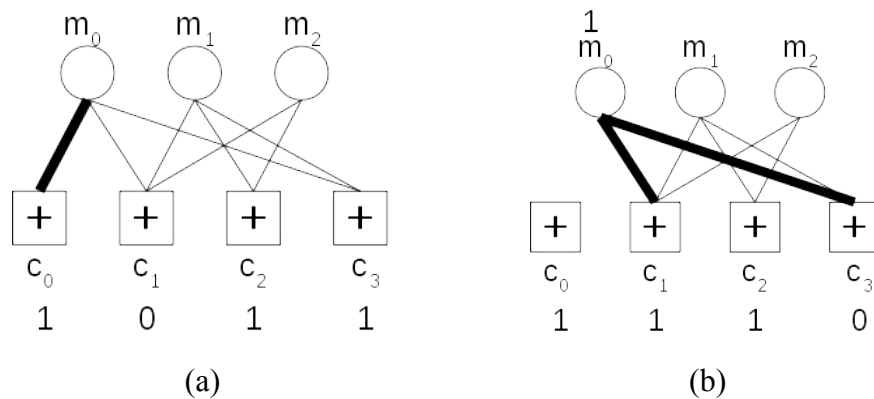
LT code generates the encoded symbols by selecting the degree  $d$  randomly from the degree distribution. Next, we select  $d$  distinct message symbols at random, add them together (XOR operation) and an encoded symbol of degree  $d$  is generated. For example, given a message of  $k=10$  symbols, i.e.,  $\mathbf{M}=(m_0, m_1, \dots, m_9)$  and a degree  $d=3$  is chosen from a degree distribution. Then, three unique message symbols are chosen from  $\mathbf{M}$  randomly to produce an encoded symbol  $x_0$ , where  $x_0=m_i \oplus m_j \oplus m_k$  and  $i \neq j \neq k$ .

Representing the encoded symbols and message symbols in bipartite graph, the original message can be reconstructed in the following steps.

- **Step 1:** All the degree one encoded symbols propagate their values to their connected message symbols respectively.
- **Step 2:** The connections between these degree one encoded symbols and the respective message symbols are removed.
- **Step 3:** The message symbols that are involved in Steps 1 and 2 will hold the latest values and then further propagate the latest values to all their connected encoded symbols respectively.

- **Step 4:** The encoded symbols that are involved in Step 3 will update their own values with the received values (XOR operation) respectively.
- **Step 5:** The connections that are involved in Steps 3 and 4 are removed.
- **Step 6:** Repeat from Step 1 until all the message symbols have been reconstructed.

Figure 2.5 demonstrates the decoding process. The message has three symbols –  $m_0$ ,  $m_1$  and  $m_2$ , in which they are located at the top of the bipartite graph in circle (see Figure 2.5(a)) and their values are unknown. The square (i.e.,  $c_0$ ,  $c_1$ ,  $c_2$  and  $c_3$ ) at the bottom are the received encoded symbols of values 1, 0, 1 and 1, respectively. The  $c_0$  is a degree one encoded symbol that links with  $m_0$  and  $c_1$  is a degree three encoded symbols that links with all three message symbols. Both  $c_2$  and  $c_3$  are degree two encoded symbols, in which their connections can be learned from the bipartite graph.





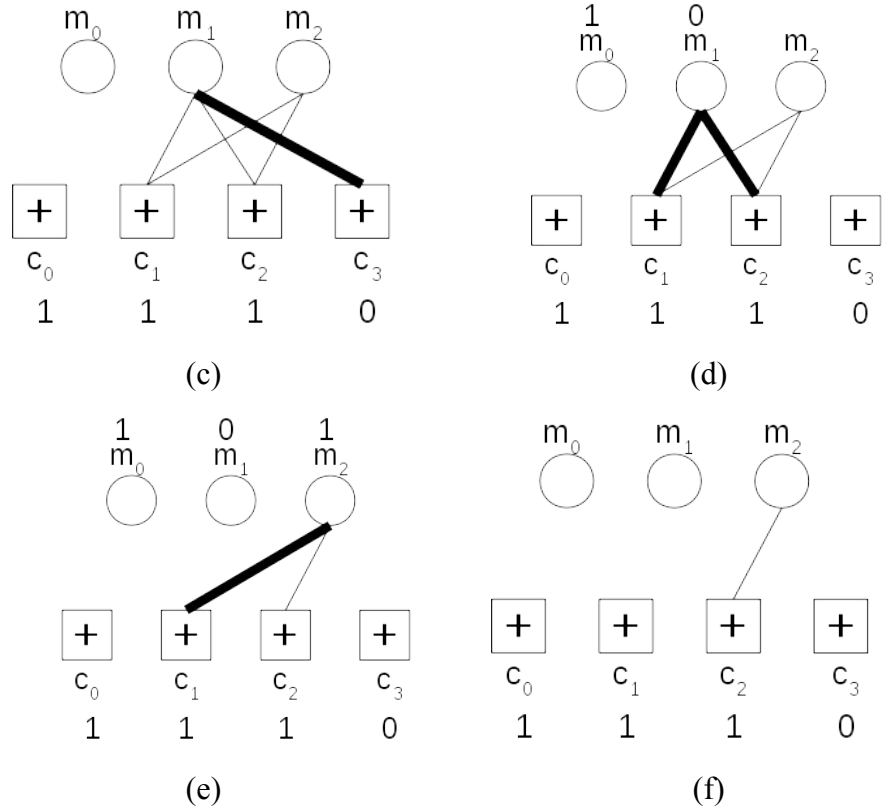


Figure 2.5: Example of LT decoding process.

We have a degree one encoded symbol (i.e.,  $c_0$ ) that links with  $m_0$  (Step 1). Then,  $c_0$  propagates its value directly to its neighbour and the corresponding connection is removed (Step 2). Then,  $m_0$  propagates its new value to  $c_1$  and  $c_3$  (Step 3), where  $c_1 \leftarrow c_1 \oplus m_0$  and  $c_3 \leftarrow c_3 \oplus m_0$  (Step 4, see Figure 2.5(b)). Then, the corresponding connections are removed (Step 5). As a result, we have a new degree one encoded symbol (i.e.,  $c_3$ ) that links with  $m_1$ . Repeating from Step 1 – the  $c_3$  propagates its value to its neighbour, updating the new value and removing the corresponding connection (see Figure 2.5(c)). Then,  $m_1$  propagates the new value to the connected encoded symbols (i.e.,  $c_1$  and  $c_2$  in Figure 2.5(d)). This process continues until all the

values of the message symbols have been reconstructed (see Figure 2.5(e) and (f)).

### 2.2.1.2 Degree Distribution

The number of degree one encoded symbols in the each decoding iteration is termed as “ripple”. The performance of LT code relies on the careful design of degree distribution, where there must be at least a degree one encoded symbol in the ripple. At the same time, the ripple size must be kept small to minimize the decoding complexity.

Luby proposes the Ideal Soliton distribution,  $\rho(d)$  that fulfils the aforementioned requirements, where

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

$d$  denotes the degree and  $k$  the total message symbols.

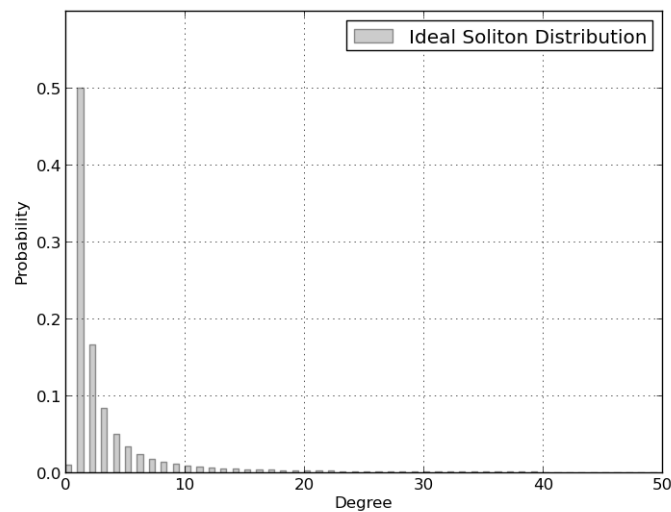


Figure 2.6: Ideal Soliton distribution for  $k=100$ .

The example of the Ideal Soliton degree distribution is presented in Figure 2.6 for  $k=100$  with degree  $d>50$  truncated due to the probabilities approaching zero. As observed, the Ideal Soliton degree distribution has the highest probability for degree two encoded symbols and the probability gradually reduces as the degree increases. A small probability is assigned to degree one encoded symbols in order to kick-start the decoding process (note that the decoding process requires degree one encoded symbol to initiate, see Section 2.2.1.1).

Generally, the Ideal Soliton distribution is susceptible to the channel erasure probability. A small variation in the degree distribution will cause ripple flats (zero degree one encoded symbol) in the middle of the decoding process. In response, Luby proposed the Robust Soliton distribution  $\mu(d)$ , where

$$\mu(d) = \frac{\rho(d) + \tau(d)}{\beta}, \quad (3)$$

for  $d=1,2,\dots,k$ ,  $\beta = \sum_{d=1}^k (\rho(d) + \tau(d))$ ,  $R = c \ln(k/\delta) \sqrt{k}$  and

$$\tau(d) = \begin{cases} \frac{R}{dk} & \text{for } d=1,\dots,k/R-1 \\ R \ln(R/\delta)/k & \text{for } d=k/R \\ 0 & \text{for } d=k/R+1,\dots,k \end{cases}, \quad (4)$$

for some constant  $c>0$  and success probability  $1-\delta$ .

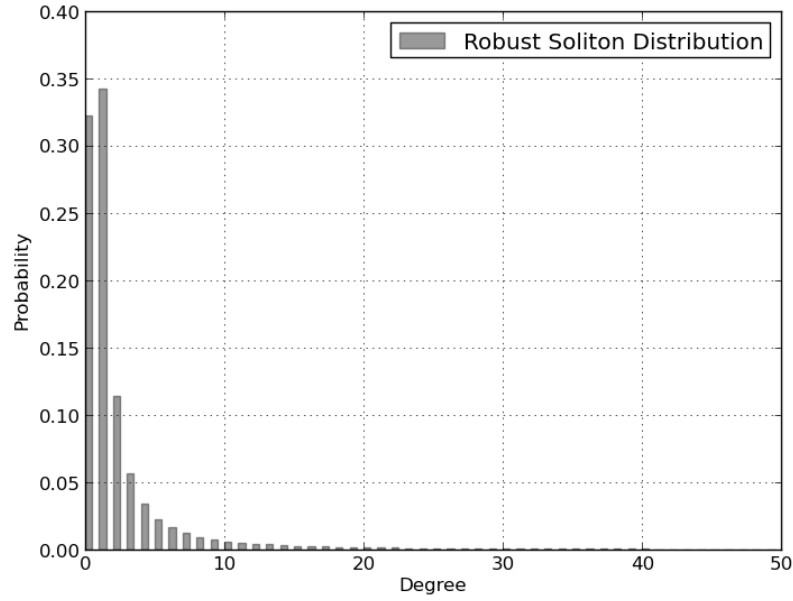


Figure 2.7: Robust Soliton distribution for  $k=100$  .

An example of Robust Soliton degree distribution for  $k=100$  ,  $c=1$  and  $1-\delta=0.95$  is presented in Figure 2.7. Note that the probability of degree one encoded symbol has been increased.

### 2.2.2 Raptor Code

The aforementioned LT code requires decoding complexity of  $O(k \log(k/\delta))$  to reconstruct the original message of  $k$  symbols with success probability  $1-\delta$  . Shokrollahi (2006) further improved the decoding complexity to  $O(k \log(1/\epsilon))$  by pre-coding the message in his Raptor Code.

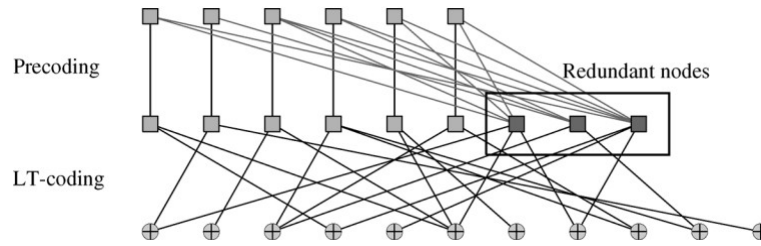


Figure 2.8: Bipartite graph of Raptor code. Figure from Shokrollahi (2006).

Let the original message be  $M$  and the pre-code's encoded symbols be  $M'$ . Correspondingly, we do not need to reconstruct every symbol of the original message but only a portion of  $M'$ . For example, the message bits (the squares in the top row) in Figure 2.8 bipartite graph are precoded with systematic erasure code and mapped to precoded message bits in the second row. Then, they are encoded with LT code to produce the final encoded bits in the last row.

Raptor code has excellence performance in transmitting long messages, i.e., decoding inefficiency of  $\epsilon \approx 0.03$  for  $k=100,000$  symbols and  $\epsilon \approx 0.04$  for  $k=65,535$  symbols. However, there is no reported result for messages of less than 1,000 symbols.

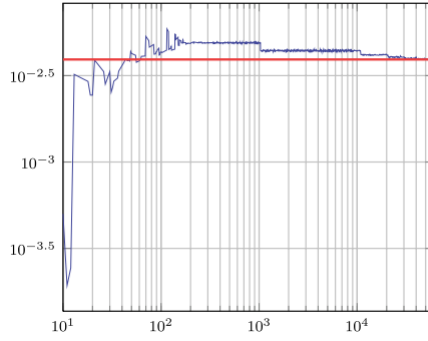
### 2.2.3 RaptorQ Code

RaptorQ code (or RaptorQ) is a variant of systematic Raptor code that is patented by Qualcomm and is constructed based on the hybrid of both  $GF(2)$  and  $GF(2^8)$  and it works for messages of tens to thousands of symbols. Generally, RaptorQ code consists of two pre-coding stages. In the first pre-

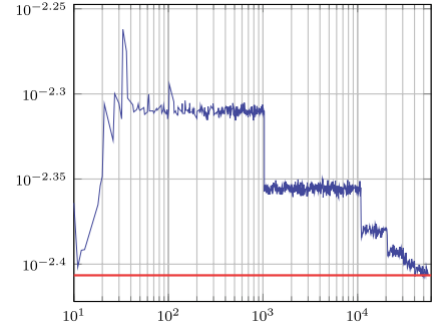
coding stage, LDPC code generates most of the redundant symbols (intermediate symbols) for the overall pre-coding stage in GF(2). In the second pre-coding stage, a small number of the redundant symbols are generated with high density parity check (HDPC) code in GF(2<sup>8</sup>).

To facilitate the discussion, we define *failure probability* as the probability that a code fails to reconstruct the original message with  $m$  overhead symbols (i.e., 1 - PCD) and express it in exponents of ten. According to the numerical results in Shokrollahi and Luby (2011), the failure probabilities of RaptorQ code fluctuate at about the same values as GF(2<sup>8</sup>) Random code, i.e.,  $10^{-2.407}$ ,  $10^{-4.815}$  and  $10^{-7.223}$  at zero, one and two overhead symbols, respectively as shown in the red lines in Figure 2.9 to 2.11, where the x-axes and y-axes refer to message length and failure probabilities. Nonetheless, RaptorQ code has a better encoding and decoding complexities than Random code.

Generally, our proposed rateless erasure codes in Chapter 3 to 6 are constructed in GF(2). In order to have a fair comparison with RaptorQ code, the proposed codes are extended to GF(2<sup>8</sup>) in Chapter 7 and compared with RaptorQ code in terms of failure probabilities.

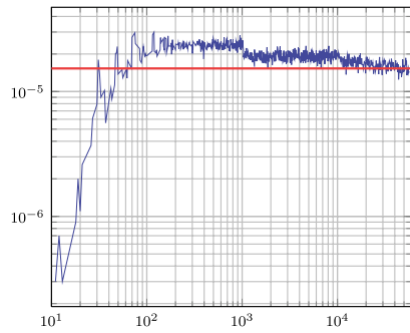


(a)

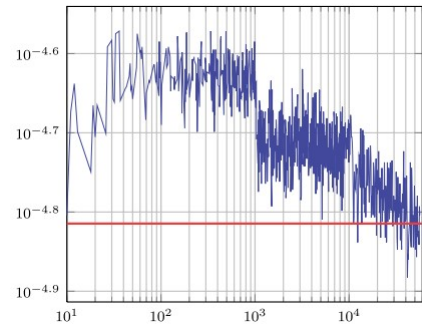


(b)

Figure 2.9: The failure probabilities of RaptorQ code at zero overhead symbol for (a)  $\rho=0.1$  and (b)  $\rho=0.5$ .

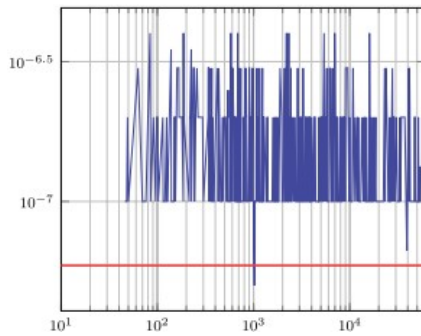


(a)

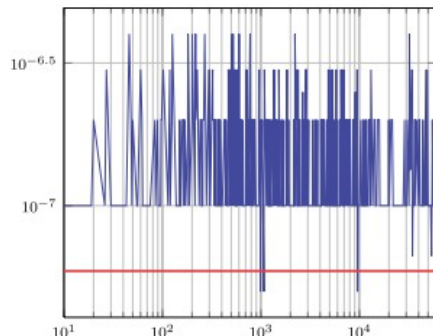


(b)

Figure 2.10: The failure probabilities of RaptorQ code at one overhead symbol for (a)  $\rho=0.1$  and (b)  $\rho=0.5$ .



(a)



(b)

Figure 2.11: The failure probabilities of RaptorQ code at two overhead symbol for (a)  $\rho=0.1$  and (b)  $\rho=0.5$ .

### 2.3 Other Rateless Erasure Code for Short Message Transmission

The state-of-the-art GF(2) rateless erasure codes in Section 2.2 (except RaptorQ code, which is not GF(2) code) are not efficient in

transmitting short messages, i.e., they require more overhead symbols to achieve complete decoding. In this section, we review the rateless erasure codes that are dedicated for short messages transmission and name them as short rateless erasure codes in the rest of the chapters.

The short rateless erasure codes presented in the following are derived from LT code due to its ameliorable degree distribution. Hyyti et al. (2007) modeled the LT decoding process with Markov chain and optimised the degree distribution accordingly. Nonetheless, the method is only suitable for very short messages due to state-space explosion. Given a message of symbol size  $k$ , the encoder can generate a total of  $2^{k-1}$  unique encoded symbols. Accordingly, these unique encoded symbols may form a total of  $2^{2^{k-1}}$  states in the Markov chain. For example, for  $k=3$  the state-space will be 128, and for  $k=4$ , the state-space will be 32,768.

On the other hand, Bodine and Cheng (2008) improved the LT code in transmitting short messages of  $k=10$ , 50 and 100 symbols and near 100% redundant symbols (i.e.,  $\epsilon \approx 1.0$ ) are required for them to achieve a complete decoding. With success probability of 99.5%, Zhu et al. (2007) managed to achieve a complete decoding for messages of  $k=2,000$  symbols with no extra redundant packets (i.e.,  $\epsilon=0$ ). However, there is no reported result for message of  $k < 1,000$  symbols.



Zhang and Hranilovic (2009) introduced a short-length Raptor code, where a message of  $k=64$  symbols required decoding inefficiency of  $\epsilon \approx 0.3$  in order to achieve a complete decoding. As for message of  $k=256$  symbols, decoding inefficiency  $\epsilon \approx 0.2$  is imposed.

Lu et al. (2013) proposed a new decoding process, namely the LT-W decoding that improves the LT code. Whenever a ripple flats, the Wiedemann algorithm is triggered to revive the ripple such that the LT decoding process can be continued. Such method preserves the low decoding complexity of LT code while improving the achievable high PCD with 10 extra encoded symbols.

Generally, our proposed short rateless erasure codes are derived from Random code (see Chapter 3) instead of LT code. To our best understanding, Windowed code (Studholme and Blake, 2006) is the only erasure code that is derived from Random code. We will introduce Windowed code right after explaining Random code in Chapter 3.

## 2.4 Summary

This chapter reviews the state-of-the-art rateless erasure codes that are efficient for long messages or short messages or both. Despite their excellent performance, our proposed rateless erasure codes outperform them in the following aspects:

1. Our proposed GF(2) codes are derived from Chapter 3's Random code. In general, they are able to reconstruct the original message from any  $k+10$  encoded symbols with high PCD, but with the trade-off of  $O(k^3)$  decoding complexity. From the literature review, we found Lu et al. (2013) has the identical performance in the simulation results but there is no explicit equation that demonstrates its high PCD with ten overhead symbols.
2. Chapter 4's Micro-Random code achieves high PCD with  $k+1$  encoded symbol, but with the trade-off of high computational complexity. To our best understanding, there is no GF(2) rateless erasure code that can achieve such result.
3. We propose two pseudo-random codes – systematic Random code and Stepping-Random code in Chapter 5 and 6, respectively. They have better decoding complexities than Random code in channels of low erasure probabilities.
4. We extend the proposed codes to GF(2<sup>8</sup>) codes in Chapter 7. They achieve much lower failure probabilities than RaptorQ code using the same amount of overhead symbols.



Each coefficient  $a_{ij}$  has equal probability to be 0 or 1. Given the values of the coefficients and the dependent variables  $x_i$ , we can find out the unknown variables with Gaussian elimination if and only if we have sufficient equations to describe the system.

Recall that a rateless erasure code is able to encode a message of  $k$  symbols into a potentially infinite number of encoded symbols. And, the receiver reconstructs the original message with any  $k(1+\epsilon)$  encoded symbols irrespective of the sequence. The aforementioned multivariate linear system resembles a rateless erasure code of  $k=3$  symbols (bits), i.e.  $m_0$ ,  $m_1$  and  $m_2$ . Each dependent variable (i.e.,  $x_i$ ) is analogous to an encoded symbol of a rateless erasure code. Therefore, to determine the values of dependent variables (i.e., to reconstruct the original message), we need at least  $k=3$  independent linear equations (encoded symbols) such that a full rank matrix is obtained.

### 3.1.2 Random Code

Given that a message of  $kl$  bits is segmented into  $k$  symbols with each symbol size  $l$  bits. Accordingly, these symbols can be represented as a matrix  $\mathbf{M}^{k \times l}$ , where

$$\begin{aligned}
\mathbf{M}^{k \times l} &= \begin{bmatrix} m_{0,0} & m_{0,1} & \cdots & m_{0,l-1} \\ m_{1,0} & m_{1,1} & \cdots & m_{1,l-1} \\ \vdots & \vdots & \ddots & \vdots \\ m_{k-1,0} & m_{k-1,1} & \cdots & m_{k-1,l-1} \end{bmatrix} \\
&= \begin{bmatrix} m_0^{1 \times l} \\ m_1^{1 \times l} \\ \vdots \\ m_{k-1}^{1 \times l} \end{bmatrix}.
\end{aligned} \tag{6}$$

The  $m_{i,j}$  represents the entry at  $i$  row  $j$  column and  $m_i^{1 \times l}$  is the  $i$ -th row that consists of  $l$  elements. Note that the row  $m_i^{1 \times l}$  can be treated as the  $i$ -th message symbol of  $l$  bits. We denote the dimensions of the matrix in superscript for clarity.

Random code is a rateless erasure code, where an encoded symbol is be generated by combining the message symbols randomly. Correspondingly, Random code has a generator matrix of randomly distributed binary values,  $\mathbf{G}^{n \times k}$ , where

$$\begin{aligned}
\mathbf{G}^{n \times k} &= \begin{bmatrix} g_{0,0} & g_{0,1} & \cdots & g_{0,k-1} \\ g_{1,0} & g_{1,1} & \cdots & g_{1,k-1} \\ \vdots & \vdots & \ddots & \vdots \\ g_{n-1,0} & g_{n-1,1} & \cdots & g_{n-1,k-1} \end{bmatrix} \\
&= \begin{bmatrix} g_0^{1 \times k} \\ g_1^{1 \times k} \\ \vdots \\ g_{n-1}^{1 \times k} \end{bmatrix}.
\end{aligned} \tag{7}$$

Each entry  $g_{i,j}$  has an equal probability to be 0 or 1. Then,  $n$  encoded symbols (denoted as  $\mathbf{X}^{n \times l}$ ) can be generated by multiplying  $\mathbf{G}^{n \times k}$  with  $\mathbf{M}^{k \times l}$ , i.e.,

$$\begin{aligned}
\mathbf{X}^{n \times l} &= \mathbf{G}^{n \times k} \times \mathbf{M}^{k \times l} \\
&= \begin{bmatrix} X_{0,0} & X_{0,1} & \cdots & X_{0,l-1} \\ X_{1,0} & X_{1,1} & \cdots & X_{1,l-1} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n-1,0} & X_{n-1,1} & \cdots & X_{n-1,l-1} \end{bmatrix} \\
&= \begin{bmatrix} X_0^{1 \times l} \\ X_1^{1 \times l} \\ \vdots \\ X_{n-1}^{1 \times l} \end{bmatrix}.
\end{aligned} \tag{8}$$

Note that a new encoded symbol can always be generated by multiplying a new generator row matrix  $g^{1 \times k}$  with  $\mathbf{M}^{k \times l}$ . Since the encoded symbols can be generated dynamically without a pre-determined  $n$ , Random code is said to be rateless.

In the perspective of linear algebra, the encoded symbols are a list of linear equations that combined the  $k$  message symbols randomly. Since the encoded symbols are constructed with  $\mathbf{G}^{n \times k} \times \mathbf{M}^{k \times l} = \mathbf{X}^{n \times l}$ , we can reconstruct the original message if the sub-matrix of  $\mathbf{G}^{n \times k}$ , denoted as  $\tilde{\mathbf{G}}^{k \times k}$ , is non-singular (i.e.,  $\tilde{\mathbf{G}}^{k \times k}$  is invertible) using Gaussian elimination of computational complexity  $O(k^3)$ . Though the decoding complexity is relatively high, Random code is able to reconstruct the original message from a fixed number of overhead symbols irrespective of the message length (see Sections 3.2 and 3.3).

### 3.1.3 An Example of Encoding and Decoding Processes

Given that a message of three symbols is represented as

$$\mathbf{M}^{3 \times 3} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}, \quad (9)$$

where each symbol consists of three bits. Say the sender has the first generator row matrix  $g_0^{1 \times 3} = [1 \ 1 \ 0]$  and the corresponding encoded symbol can be obtained with

$$\begin{aligned} g_0^{1 \times 3} \times \mathbf{M}^{3 \times 3} &= x_0^{1 \times 3} \\ &= [1 \ 0 \ 1]. \end{aligned} \quad (10)$$

Given that the second and third generator row matrices as  $g_1^{1 \times 3} = [0 \ 1 \ 1]$  and  $g_2^{1 \times 3} = [1 \ 0 \ 1]$ , the corresponding encoded symbols are  $x_1^{1 \times 3} = [1 \ 1 \ 0]$  and  $x_2^{1 \times 3} = [0 \ 1 \ 1]$ , which are obtained with the same method in Eq. (10). Then, the overall generator matrix and encoded matrix are

$$\mathbf{G}^{3 \times 3} = \begin{bmatrix} g_0^{1 \times 3} \\ g_1^{1 \times 3} \\ g_2^{1 \times 3} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}, \quad (11)$$

and

$$\mathbf{X}^{3 \times 3} = \begin{bmatrix} x_0^{1 \times 3} \\ x_1^{1 \times 3} \\ x_2^{1 \times 3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}. \quad (12)$$

The original message can be reconstructed if  $\mathbf{G}^{3 \times 3}$  is a full rank matrix (i.e.,  $\text{rank}(\mathbf{G}^{3 \times 3}) = 3$ ). However,  $\mathbf{G}^{3 \times 3}$  in Eq. (11) has only rank 2

because the third row is not independent. Hence, an additional encoded symbol is required.

Say, we obtain a new encoded symbol from the sender –  $g_3^{1 \times 3} = [0 \ 1 \ 0]$  and  $x_3^{1 \times 3} = [0 \ 1 \ 1]$ . The resulting generator matrix and encoded matrix are represented as

$$\mathbf{G}^{4 \times 3} = \begin{bmatrix} g_0^{1 \times 3} \\ g_1^{1 \times 3} \\ g_2^{1 \times 3} \\ g_3^{1 \times 3} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad (13)$$

and

$$\mathbf{X}^{4 \times 3} = \begin{bmatrix} x_0^{1 \times 3} \\ x_1^{1 \times 3} \\ x_2^{1 \times 3} \\ x_3^{1 \times 3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}. \quad (14)$$

We noticed that the sub-matrix (i.e.,  $g_0^{1 \times 3}$ ,  $g_1^{1 \times 3}$  and  $g_3^{1 \times 3}$ ) of  $\mathbf{G}^{4 \times 3}$  is full rank (i.e., rank 3). Representing the sub-matrix as

$$\tilde{\mathbf{G}}^{3 \times 3} = \begin{bmatrix} g_0^{1 \times 3} \\ g_1^{1 \times 3} \\ g_3^{1 \times 3} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad (15)$$

and their corresponding encoded symbols,

$$\tilde{\mathbf{X}}^{3 \times 3} = \begin{bmatrix} x_0^{1 \times 3} \\ x_1^{1 \times 3} \\ x_3^{1 \times 3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}. \quad (16)$$

Since the inverse of  $\tilde{\mathbf{G}}^{3 \times 3}$  is,



$$[\tilde{\mathbf{G}}^{3 \times 3}]^{-1} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \quad (17)$$

applying  $[\tilde{\mathbf{G}}^{3 \times 3}]^{-1}$  to  $\tilde{\mathbf{X}}^{3 \times 3}$  and we have reconstructed the original message  $\mathbf{M}^{3 \times 3}$  successfully, i.e.,

$$\mathbf{M}^{3 \times 3} = [\tilde{\mathbf{G}}^{3 \times 3}]^{-1} \times \tilde{\mathbf{X}}^{3 \times 3} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}. \quad (18)$$

### 3.2 Probability of Complete Decoding (PCD) for Long Messages

Generally, Random code reconstructs the original message with high PCD from  $k+10$  encoded symbols, or on average 1.6 overhead symbols. We explicate the aforementioned statement for long messages transmission using the following Kolchin's theorem and short messages transmission in Section 3.3.

Let  $\mathbf{G}^{(k+m) \times k}$  be a random matrix of size  $(k+m) \times k$ . Kolchin's theorem (Theorem 3.2.1 in Kolchin, 1998) states that the probability for random matrix  $\mathbf{G}^{(k+m) \times k}$  to have rank  $k-s$  is

$$\Pr(\text{rank}(\mathbf{G}^{(k+m) \times k}) = k-s) \rightarrow Q_{\text{kol}}(s, m), \quad (19)$$

when  $k \rightarrow \infty$ , where

$$Q_{\text{kol}}(s, m) = 2^{-s(m+s)} \prod_{i=s+1}^{\infty} \left(1 - \frac{1}{2^i}\right) \prod_{i=1}^{m+s} \left(1 - \frac{1}{2^i}\right)^{-1}, \quad (20)$$

for  $s \geq 0$ ,  $m \in \mathbb{Z}$  and  $m+s \geq 0$ .

Eq. (20) can be expressed as

$$Q_{\text{kol}}(m) \equiv Q_{\text{kol}}(0, m) = \prod_{i=m+1}^{\infty} \left(1 - \frac{1}{2^i}\right), \quad (21)$$

for calculating the probability to have full rank matrix and the numerical values are presented in Table 3.1. As observed, the probability for a random matrix to achieve complete decoding with exactly  $k$  encoded symbols is  $Q_{\text{kol}}(0) = 0.2888$ . To achieve high PCD (99.9% successful decoding), ten extra encoded symbols (i.e., ten overhead symbols) are required, i.e.,  $Q_{\text{kol}}(10) = 0.9990$ .

Generally, Eq. (21) is a cumulative distribution function (CDF). Let  $P_{\text{kol}}(m) = Q_{\text{kol}}(m) - Q_{\text{kol}}(m-1)$  be the probability mass function (PMF) for a random matrix to reach full rank with  $m$  extra rows and  $P_{\text{kol}}(0) = Q_{\text{kol}}(0)$ . Accordingly, the expected extra rows (expected overhead symbols) for the random code to reach full rank is

$$\begin{aligned} E(\text{rank}(\mathbf{G}^{(k+m) \times m}) = k) &= \sum_{m=0}^{\infty} m P_{\text{kol}}(m), \\ &\approx 1.6067 \dots \end{aligned} \quad (22)$$

i.e., 1.6 extra rows are needed on average for random matrix to reach full rank.

Table 3.1: The probability (CDF and PMF) for a random matrix to achieve complete decoding with  $m$  additional rows.

$m$	$Q_{\text{kol}}(m)$	$P_{\text{kol}}(m)$	$m$	$Q_{\text{kol}}(m)$	$P_{\text{kol}}(m)$
0	0.288788	0.288788	11	0.999512	0.000488
1	0.577576	0.288788	12	0.999756	0.000244
2	0.770102	0.192526	13	0.999878	0.000122
3	0.880116	0.110014	14	0.999939	0.000061
4	0.938791	0.058675	15	0.999969	0.000030
5	0.969074	0.030283	16	0.999985	0.000016
6	0.984456	0.015382	17	0.999992	0.000007
7	0.992208	0.007752	18	0.999996	0.000004
8	0.996099	0.003891	19	0.999998	0.000002
9	0.998048	0.001949	20	0.999999	0.000001
10	0.999024	0.000976			

### 3.3 Probability of Complete Decoding (PCD) for Short Messages

Kolchin's theorem is an asymptotic equation. For small values of  $k$  (short messages), it is unclear if such an asymptotic equation is useful. In this section, a theorem extending Kolchin's work is formulated in order to explain the PCD of Random code for short messages.

We are aware of the similar work in MacKay (2005) and Shokrollahi (2006), whereby the upper bound of the PCD is given as a function of overhead symbols. However, these papers do not explicitly deal with Random code arguing that the encoding and decoding costs are prohibitive. Our work herein departs from these works in that we use the exact PCD to dimension Random code for short messages. Moreover, our work extends Kolchin's theorem to short messages in support of Random code as a good rateless erasure code. We present the theorem starting with the following lemma.

**Lemma 3.3.1.** Given that a random matrix  $\mathbf{G}^{(n-1) \times k}$  of dimensions  $(n-1) \times k$  has rank  $(n-1)$ , where  $0 < n \leq k$ , then the probability of achieving rank  $n$  with an extra row is

$$p(n, k) = \prod_{i=1}^{n-1} (1 - 2^{i-k}). \quad (23)$$

**Proof.** Let  $g^{1 \times k}$  be a new row matrix with  $2^k - 1$  possible combinations excluding the row matrix of all zeros. In order to have rank  $n$  in  $\mathbf{G}^{n \times k}$ ,  $g^{1 \times k}$  must be independent of the other. Therefore,  $g^{1 \times k}$  is limited to  $\tau(n-1, k)$  possible combinations, where

$$\begin{aligned} \tau(n-1, k) &= 2^k - 1 - \sum_{i=1}^{n-1} \binom{n-1}{i} \\ &= 2^k - 1 - (2^{n-1} - 1) \\ &= 2^k - 2^{n-1}, \end{aligned} \quad (24)$$

and  $\binom{\cdot}{\cdot}$  is the binomial coefficient.

Each of the generator row matrices is generated independently. Correspondingly, the probability of  $\mathbf{G}^{(n-1) \times k}$  attaining rank  $n$  with  $g^{1 \times k}$  is

$$\begin{aligned}
p(n, k) &= p(n-1, k) \times \frac{\tau(n-1, k)}{2^k} \\
&= p(n-2, k) \times \frac{\tau(n-2, k)}{2^k} \times \frac{\tau(n-1, k)}{2^k} \\
&= \frac{\tau(1, k)}{2^k} \times \frac{\tau(2, k)}{2^k} \times \dots \times \frac{\tau(n-1, k)}{2^k} \\
&= \prod_{i=1}^{n-1} \frac{\tau(i, k)}{2^k} \\
&= \prod_{i=1}^{n-1} \frac{2^k - 2^i}{2^k} \\
&= \prod_{i=1}^{n-1} (1 - 2^{i-k}).
\end{aligned} \tag{25}$$

■

We study Lemma 3.3.1 with different matrix dimensions. Surprisingly, we find that the probability of having higher rank numbers (i.e., rank =  $k - 10, k - 9, \dots, k$ ) in matrices of different column dimensions has a negligibly small difference. For example, the probability to have rank 20 (i.e.,  $30 - 10$ ) in  $\mathbf{G}^{20 \times 30}$  is identical to the case of having rank 30 (i.e.,  $40 - 10$ ) in  $\mathbf{G}^{30 \times 40}$ . For different values of  $k$ , we tabulated the PCD in Table 3.2 (using Eq. (23) of Lemma 3.3.1) and find that the probabilities converge to five decimal digits after  $k=30$ . The precise study of the aforementioned observation is given in the following Lemma 2.

Table 3.2: The probabilities of having the last ten rank numbers for  $k-m=20$  and  $k-m=30$  and  $k$  is fixed as 30.

$m$	$p(k-m, k)$	$m$	$p(k-m, k)$
10	0.99902	4	0.93879
9	0.99805	3	0.88012
8	0.99610	2	0.77010
7	0.99221	1	0.57758
6	0.98446	0	0.28879
5	0.96907		

**Lemma 3.3.2.** Let  $m \in \mathbb{Z}^+$  and  $0 \leq m < k$ . Then, the probabilities to have rank  $k - m$  in matrices of dimensions  $(k - m) \times k$  for different  $k$  are similar, i.e.,

$$|p(k - m + 1, k + 1) - p(k - m, k)| < \delta, \quad (26)$$

and  $\delta \rightarrow 0$  exponentially fast with finite and bounded increment  $k$ .

**Proof.** By induction on  $k$ , we show that  $\delta < \frac{1}{2^k} \prod_{x=m+1}^{k-1} \left( \frac{2^x - 1}{2^x} \right)$ . First, we note that

$$|p(k - m + 1, k + 1) - p(k - m, k)| < |p(k - m, k) - p(k - m - 1, k - 1)|. \quad (27)$$

We may express  $|p(k - m, k) - p(k - m - 1, k - 1)|$  as

$$\begin{aligned} & |p(k - m, k) - p(k - m - 1, k - 1)| \\ &= \left| \prod_{i=1}^{k-m-1} (1 - 2^{i-k}) - \prod_{i=1}^{k-m-2} (1 - 2^{i-(k-1)}) \right| \\ &= \left| (1 - 2^{1-k})(1 - 2^{2-k}) \dots (1 - 2^{-m-1}) - (1 - 2^{-k})(1 - 2^{1-k}) \dots (1 - 2^{-m-1}) \right| \\ &= \left| (1 - 2^{1-k})(1 - 2^{2-k}) \dots (1 - 2^{-m-1}) [1 - (1 - 2^{-k})] \right| \\ &= \left| (1 - 2^{1-k})(1 - 2^{2-k}) \dots (1 - 2^{-m-1}) \left[ \frac{1}{2^k} \right] \right| \\ &= \left| \left( \frac{1}{2^{k-1}} \right) \left( \frac{1}{2^{k-2}} \right) \dots \left( \frac{1}{2^{m+1}} \right) \left[ \frac{1}{2^k} \right] \right| \\ &= \left| \left( \frac{2^{k-1} - 1}{2^{k-1}} \right) \left( \frac{2^{k-2} - 1}{2^{k-2}} \right) \dots \left( \frac{2^{m+1} - 1}{2^{m+1}} \right) \left[ \frac{1}{2^k} \right] \right| \\ &= \left| \left[ \frac{1}{2^k} \right] \prod_{x=m+1}^{k-1} \left( \frac{2^x - 1}{2^x} \right) \right|. \end{aligned} \quad (28)$$

Let  $\delta$  denote  $|p(k - m + 1, k + 1) - p(k - m, k)|$ . Then, by Eq.( 28),

$$\delta < \left| \left[ \frac{1}{2^k} \right] \prod_{x=m+1}^{k-1} \left( \frac{2^x - 1}{2^x} \right) \right|. \quad (29)$$

Since  $\left(\frac{2^x-1}{2^x}\right) < 1$  and  $0 \leq m < k$  for all  $\delta \in \mathbb{R}^+$  and

$$\inf \left\{ \left[ \frac{1}{2^k} \prod_{x=m+1}^{k-1} \left( \frac{2^x-1}{2^x} \right) \right] \right\} = 0, \text{ by monotone convergence we have } \delta \rightarrow 0$$

driven by an exponential term  $\frac{1}{2^k}$ .

■

To show that the probability of achieving complete decoding with  $m$  extra rows is the same as  $p(k-m, k)$ , we use the well known result of matrix rank invariance under transposition. We state the following lemma without proof.

**Lemma 3.3.3.** The probabilities to have rank  $k-m$  in  $\mathbf{G}^{(k-m) \times k}$  and  $\mathbf{G}^{k \times (k-m)}$  are identical.

Lemma 3.3.3 asserts that the rank of a matrix is invariant to matrix transposition. For example, the probability for  $\mathbf{G}^{(k-1) \times k}$  to have rank  $k-1$  is 0.57758 as stated in Table 3.2. Then, the probability to have rank  $k-1$  in  $\mathbf{G}^{k \times (k-1)}$  is also 0.57758.

Finally, building upon Lemma 3.3.1 and 3.3.3, we propose the following theorem.

**Theorem 3.3.4.** The probability to have rank  $k$  in matrix of dimensions  $(k+m) \times k$  for  $m \geq 0$  is

$$\begin{aligned} \Pr(\text{rank}(\mathbf{G}^{(k+m) \times k})=k) &= p(k-m, k) \\ &= \prod_{i=1}^{k-m-1} (1-2^{i-k}). \end{aligned} \quad (30)$$

For the ease of the explanation, we will use  $Q_{\text{RC}}(m)$  to denote the PCD of Random code with  $m$  overhead symbols, where

$$Q_{\text{RC}}(m) = \Pr(\text{rank}(\mathbf{G}^{(k+m) \times k})=k). \quad (31)$$

The probability to achieve complete decoding with  $m$  overhead symbols is the same as  $p(k-m, k)$  in the Table 3.2 for  $m=0, 1, \dots, 10$ . and  $Q_{\text{RC}}(10) = Q_{\text{kol}}(10) = 0.99902$ .

Pairing Kolchin's theorem (see Section 3.2) with Theorem 3.3.4, we conclude that Random code is able to achieve high PCD with  $k+10$  encoded symbols irrespective of the message length.

### 3.4 Random Code Variant – Windowed Code

Studholme and Blake (2006) proposed a fixed weight pseudo-random code, namely Windowed code, which has a better decoding complexity (i.e.,  $O(k^{3/2})$ ) than Random code but with minor trade-off in PCD.



Let  $\sigma$  be the row weight of Windowed code, where

$$\sigma = \lceil 2 \log k \rceil_{\text{odd}}, \quad (32)$$

i.e., the lowest odd integer greater than or equal to  $2 \log k$ . An entry  $i$  is chosen randomly from the row and its value is set to 1. The entry  $i$  is also known as the *initial 1*. Then,  $\sigma - 1$  1's are placed randomly within the next  $w = 2\sqrt{k}$  entries (window) with the last entry linked to the first entry.

$$\begin{bmatrix} \mathbf{1} & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & \mathbf{1} & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \quad (33)$$

An example of Windowed code's generator matrix of  $k=10$  is presented in Eq. (33). The row weight,  $\sigma = \lceil 2 \log k \rceil_{\text{odd}} = 3$  and the window  $w = 2\sqrt{k} = 6.3245 \approx 7$ . Say the initial 1 of the first row is the first entry. So, the rest of  $\sigma - 1 = 2$  1's must be placed within the second to the seventh entries. The initial 1 of the second row starts at third entries and rest of 1's must be placed within fourth to ninth entries. The initial 1 of third row is located at the seventh entry and the rest of 1's are placed in the next three entries and the first three entries.

Though Windowed code has a better decoding complexity than Random code, its PCD is not explained using rigorous mathematics. In contrast, the PCD of Random code in transmitting long and short messages are explained mathematically in Sections 3.2 and 3.3 respectively.

### 3.5 The Challenges of Random Code

Though Random code is able to reconstruct the original message with high PCD with  $k+10$  encoded symbols, it is challenged by the following issues.

1. Decoding inefficiency is high for short messages. For example, a message of  $k=10$  symbols requires  $k+10=20$  encoded symbols in order to reach high PCD. The decoding inefficiency is  $\epsilon=1$  in this case.
2. The decoding complexity of Random code is relatively high, i.e.  $O(k^3)$ .
3. Random code is a non-systematic rateless erasure code. The receiver takes time to reconstruct the original message even if all the encoded symbols are received intact.

We address the first issue by proposing Micro-Random Code in Chapter 4, where a high PCD is achievable with  $k+1$  encoded symbol. The second and third issues are addressed with the proposals of systematic and non-systematic pseudo-random codes in Chapters 5 and 6, where they have better decoding complexities and PCD.

## CHAPTER 4

### MICRO-RANDOM CODE

This chapter proposes a variant of Random code that achieves high PCD with  $k+1$  encoded symbols, i.e., one overhead symbol instead of  $k+10$  encoded symbols as in Random code. As for the trade-off, such improvement comes with a stringent requirement on computational resource.

#### 4.1 High Decoding Inefficiency in Transmitting Short Messages

Recall that Chapter 3's Random code is able to reconstruct the original message with high PCD at ten overhead symbols, irrespective of message length  $k$ . Manipulating Eq. (1), the decoding inefficiency of Random code can be expressed as

$$\epsilon_{RC} = \frac{n}{k} - 1 = \frac{10}{k}, \quad (34)$$

with  $n=k+10$ .

Generally, Random code has a high decoding inefficiency when a message is short, e.g., in ten-digits. For example, the decoding inefficiency is  $\epsilon_{RC}=0.01$  for a message of  $k=1,000$  symbols, but  $\epsilon_{RC}=1$  for  $k=10$  symbols. It requires  $k+10=20$  encoded symbols in order to reconstruct a short message of  $k=10$  symbols with high PCD and the total required encoded symbols is double that of the total message symbols.

## 4.2 Micro-Random Code

We explain the concept of symbol dimensioning before addressing the aforementioned issue with Micro-Random code.

### 4.2.1 Symbol Dimensioning in Brief

Sections 3.2 and 3.3 have demonstrated that PCD of Random code is a function of the total received overhead symbols,  $m$  and is invariant to the message length,  $k$ . Say we have a message of 1,000 bits. Encoding the message in  $k_1=10$  symbols with each symbol size  $l_1=100$  bits (Case 1) is no different than encoding the message in  $k_2=20$  symbols of symbol size  $l_2=50$  bits (Case 2) in term of PCD. Both cases require  $k+10$  encoded symbols in order to reconstruct the original message with high PCD. Nonetheless, the ten overhead symbols in Case 1 contribute the redundancies of  $10 \times l_1 = 1000$  bits but Case 2 only requires  $10 \times l_2 = 500$  bits.

The aforementioned example suggests that the decoding inefficiency can be improved if we dimension the symbol size appropriately during the encoding and decoding processes. In the rest of the chapter, we will use the terms *micro symbols*, *encoded micro symbols* and *encoded symbols* to indicate the symbols at different stages of the encoding and decoding processes.

### 4.2.2 Improving Decoding Inefficiency with Micro Symbols

Let a message of  $kl$  bits be segmented to  $k$  symbols of size  $l$  and represented as a matrix

$$\mathbf{M}^{k \times l} = \begin{bmatrix} m_0^{1 \times l} \\ m_1^{1 \times l} \\ \vdots \\ m_{k-1}^{1 \times l} \end{bmatrix}, \quad (35)$$

where  $m_i^{1 \times l}$  represents the  $i$ -th row matrix of dimension  $l$  or the  $i$ -th symbol of the original message with symbol size  $l$ . We denote the dimensions in superscript for clarity.

According to Chapter 3, an encoded symbol can be generated by multiplying a generator row matrix  $\mathbf{G}^{1 \times k}$  with  $\mathbf{M}^{k \times l}$ , that is  $\mathbf{X}^{1 \times l} = \mathbf{G}^{1 \times k} \mathbf{M}^{k \times l}$ . Moreover, we need  $k+10$  encoded symbols (each has size  $l$  bit) in order to reconstruct the original message with high PCD. To reduce the decoding inefficiency, we dimension each message symbol into  $\alpha$  symbols of smaller size, namely micro symbols as shown in Figure 4.1 (left hand side). Correspondingly, the message is transformed from a matrix of dimensions  $k \times l$  into  $(\alpha k) \times (l/\alpha)$ , where  $\alpha \in \mathbb{N}$  and  $\alpha$  divides  $l$ . In other words, we shrink the column size (symbol size) from  $l$  to  $l/\alpha$  while increasing the row size (total message symbol) from  $k$  to  $\alpha k$ . Then, the new matrix is expressed as

$$\mathbf{M}^{k\alpha \times l/\alpha} = \begin{bmatrix} m_0^{1 \times (l/\alpha)} \\ m_1^{1 \times (l/\alpha)} \\ \vdots \\ m_{\alpha k - 1}^{1 \times (l/\alpha)} \end{bmatrix}. \quad (36)$$

Note that both  $\mathbf{M}^{k\alpha \times l/\alpha}$  and  $\mathbf{M}^{k \times l}$  represent the same message but in different dimensions.

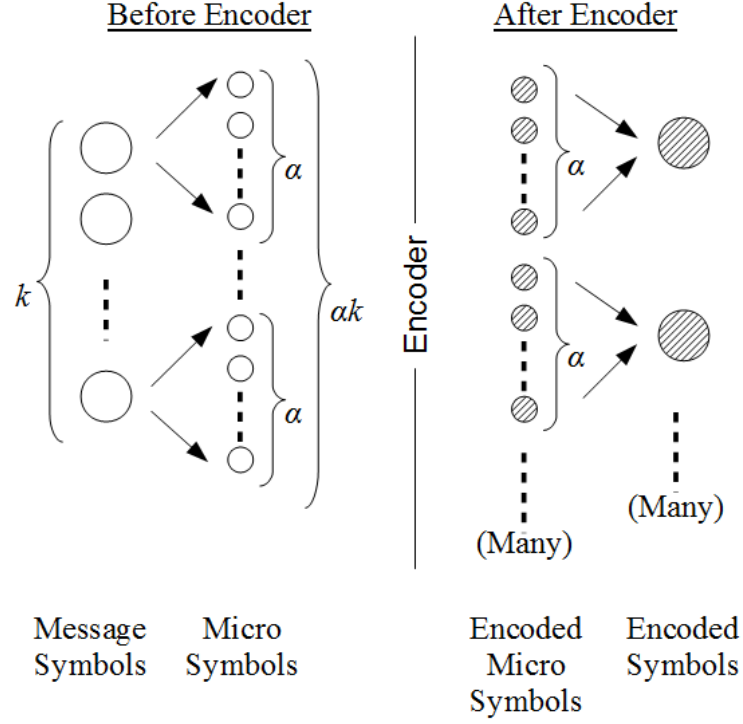


Figure 4.1: The encoding process of Micro-Random code.

By feeding  $\mathbf{M}^{k\alpha \times l/\alpha}$  into the encoder, it generates many encoded micro symbols of size  $l/\alpha$  bits, that is

$$\mathbf{X}^{(\cdot) \times (1/\alpha)} = \begin{bmatrix} x_0^{1 \times l/\alpha} \\ x_1^{1 \times l/\alpha} \\ \vdots \end{bmatrix}. \quad (37)$$

We use  $(\cdot)$  when the quantity is not known. Next, we regroup each  $\alpha$  encoded micro symbols into one encoded symbol of  $l$  bits (Figure 4.1 right hand side), that is

$$\mathbf{X}^{(\cdot) \times l} = \begin{bmatrix} x_0^{1 \times l/\alpha} & x_1^{1 \times l/\alpha} & \dots & x_{\alpha-1}^{1 \times l/\alpha} \\ x_{\alpha}^{1 \times l/\alpha} & x_{\alpha+1}^{1 \times l/\alpha} & \dots & x_{2\alpha-1}^{1 \times l/\alpha} \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}. \quad (38)$$

Each  $l$  bits encoded symbol comprises  $\alpha$  encoded micro symbols of  $l/\alpha$  bits. Alternatively, we can denote each encoded symbol  $x_i^{1 \times l}$  as

$$x_i^{1 \times l} = \left[ g_{\alpha i}^{1 \times \alpha k} \mathbf{M}^{\alpha k \times l/\alpha} \quad g_{\alpha i+1}^{1 \times \alpha k} \mathbf{M}^{\alpha k \times l/\alpha} \quad \dots \quad g_{\alpha i+(\alpha-1)}^{1 \times \alpha k} \mathbf{M}^{\alpha k \times l/\alpha} \right], \quad (39)$$

where  $g_j^{1 \times \alpha k}$  are the corresponding generator row matrices.

Generally, one encoded symbol consists of  $\alpha$  encoded micro symbols, and  $1/\alpha$  encoded symbol represents one encoded micro symbol, i.e.,

$$\begin{aligned} 1 \text{ encoded symbol} &= \alpha \text{ encoded micro symbols} \\ 1/\alpha \text{ encoded symbol} &= 1 \text{ encoded micro symbol.} \end{aligned} \quad (40)$$

Since  $\alpha k$  micro symbols are used in the encoding process, the receiver requires  $\alpha k + 10$  encoded micro symbols in order to achieve high PCD. In other words,

$$\begin{aligned} \alpha k + 10 \text{ encoded micro symbols} &= \frac{1}{\alpha} (\alpha k + 10) \text{ encoded symbols} \\ &= k + \frac{10}{\alpha} \text{ encoded symbols,} \end{aligned} \quad (41)$$

i.e.,  $k + \frac{10}{\alpha}$  encoded symbols are needed. As we cannot have a fractional encoded symbol, the number of extra encoded symbols required to achieve high PCD is

$$n_{\text{mRC}} = k + \left\lceil \frac{10}{\alpha} \right\rceil. \quad (42)$$

Once  $n_{\text{mRC}}$  encoded symbols have been received, the receiver extracts  $\alpha n_{\text{mRC}} = \alpha k + 10$  encoded micro symbols as shown in Figure 4.2. These  $\alpha k + 10$  encoded micro symbols are sufficient to reconstruct the original message with high PCD. Since  $n_{\text{mRC}} = k + \left\lceil \frac{10}{\alpha} \right\rceil = k(1 + \epsilon_{\text{mRC}})$ , the decoding inefficiency of Micro-Random code is

$$\epsilon_{\text{mRC}} = \left\lceil \frac{10}{\alpha} \right\rceil \frac{1}{k}. \quad (43)$$

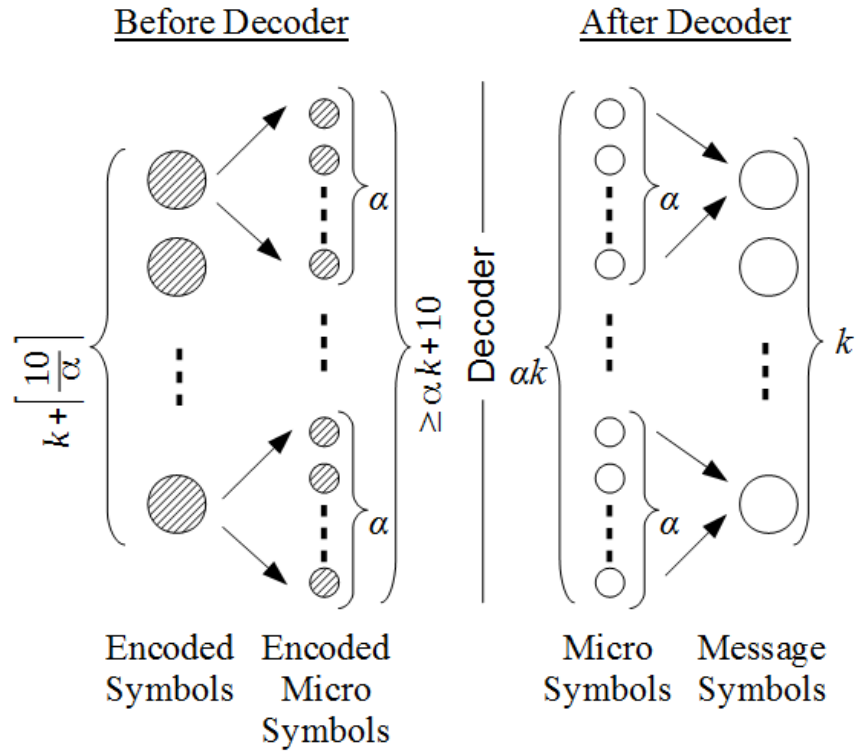


Figure 4.2: The decoding process of Micro-Random code.

For example, let the symbol size be  $l=10$  bits. Then, a message of size 100 bits can be represented in  $k=10$  symbols of size  $l=10$  or in a matrix of size  $10 \times 10$ . To improve the decoding inefficiency, we represent the message in 20 micro symbols, i.e.,  $\alpha=2$  and each encoded symbol of size  $l=5$  bits. Then, we group each pair of the encoded micro symbols to form a regular encoded symbol of 10 bits. Subsequently, the receiver needs only  $k + \lceil 10/2 \rceil = 15$  encoded symbols to reconstruct the original message as they have contributed 30 encoded micro symbols already. In this case,  $\epsilon_{\text{mRC}} = 0.5$ .

### 4.3 Performance Analysis

We analyse the performance of Micro-Random code in terms of the PCD and the expected overhead symbols to achieve complete decoding.



### 4.3.1 Probability of Complete Decoding (PCD)

Each encoded symbol of Micro-Random code has  $\alpha$  encoded micro symbols. Hence,  $m$  overhead symbols contributes  $m\alpha$  extra encoded micro symbols. Modifying Eq. (31), the PCD of Micro-Random Code with  $m$  overhead symbols can be expressed as

$$\begin{aligned} Q_{\text{mRC}}(m, \alpha) &= Q_{\text{RC}}(\alpha m) \\ &= \prod_{i=1}^{k-\alpha m-1} (1-2^{i-k}), \end{aligned} \quad (44)$$

and the numerical values are presented in Table 4.1, where “>0.999999” denotes a numerical value larger than 0.999999 but less than 1.0. According to the table, as  $\alpha$  increases, lesser overhead symbols ( $m$ ) are required to achieve high PCD (i.e.,  $Q_{\text{mRC}}(m, \alpha) \geq 0.999$ ). In particular, we achieve high PCD with only one overhead symbol if  $\alpha \geq 10$ .

To examine whether it can reach high PCD at zero overhead symbol, we substitute  $m=0$  into Eq. (44),

$$\begin{aligned} Q_{\text{mRC}}(0, \alpha) &= Q_{\text{RC}}(\alpha \times 0) \\ &= Q_{\text{RC}}(0) \\ &= \prod_{i=1}^{k-1} (1-2^{i-k}), \end{aligned} \quad (45)$$

and  $Q_{\text{mRC}}(0, \alpha) = 0.28879$ . Apparently,  $\alpha$  has no influence to  $Q_{\text{mRC}}$  when  $m=0$  and there is no way to achieve high PCD with zero overhead symbol (i.e.,  $m=0$ ) with large value of  $\alpha$ . Achieving high PCD with  $k+1$  encoded symbols (for  $\alpha=10$ ) is the best performance we can get.

Table 4.1: The PCD of Micro-Random code,  $Q_{\text{mRC}}(m, \alpha)$  for various  $m$  and  $\alpha$ .

		$\alpha$					
		1	2	4	8	10	20
m	0	0.288788	0.288788	0.288788	0.288788	0.288788	0.288788
	1	0.577576	0.770102	0.938791	0.996099	0.999024	0.999999
	2	0.770102	0.938791	0.996099	0.999985	0.999999	>0.999999
	3	0.880116	0.984456	0.999756	>0.999999	>0.999999	>0.999999
	4	0.938791	0.996099	0.999985	>0.999999	>0.999999	>0.999999
	5	0.969074	0.999024	0.999999	>0.999999	>0.999999	>0.999999
	6	0.984456	0.999756	>0.999999	>0.999999	>0.999999	>0.999999
	7	0.992208	0.999939	>0.999999	>0.999999	>0.999999	>0.999999
	8	0.996099	0.999985	>0.999999	>0.999999	>0.999999	>0.999999
	9	0.998048	0.999996	>0.999999	>0.999999	>0.999999	>0.999999
	10	0.999024	0.999999	>0.999999	>0.999999	>0.999999	>0.999999

Alternatively, the aforementioned result can also be deduced from Eq.

(42), i.e.,  $n_{\text{mRC}} = k + \left\lceil \frac{10}{\alpha} \right\rceil$ . Each increment in  $\alpha$  will reduce the total required encoded symbols to achieve high PCD until  $\alpha = 10$ . No improvement can be made for  $\alpha > 10$  as  $\lceil n_{\text{mRC}} \rceil_{\alpha=10,11,\dots} = k + 1$ .

### 4.3.2 Expected Overhead Symbols

Let PMF to achieve high PCD be expressed as

$$P_{\text{mRC}}(m, \alpha) = Q_{\text{mRC}}(m, \alpha) - Q_{\text{mRC}}(m-1, \alpha), \quad (46)$$

and

$$P_{\text{mRC}}(0, \alpha) = Q_{\text{mRC}}(0, \alpha). \quad (47)$$

Then, the expected overhead symbols can be expressed as

$$E_{\text{mRC}}(\alpha) = \sum_{m=1}^{\infty} m P_{\text{mRC}}(m, \alpha), \quad (48)$$

and the numerical values are presented in Table 4.2. Note that “<0.00001” denotes a non-zero positive value that is less than 0.00001.

Table 4.2: The PMF of Micro-Random code for various overhead symbols  $m$ , segmentation factor  $\alpha$  and the expected value.

$m$	$\alpha$			
	1	2	4	10
0	0.28879	0.28879	0.28879	0.28879
1	0.28879	0.48131	0.65000	0.71024
2	0.19253	0.16869	0.05731	0.00098
3	0.11001	0.04567	0.00366	<0.00001
4	0.05867	0.01164	0.00023	<0.00001
5	0.03028	0.00292	0.00001	<0.00001
6	0.01538	0.00073	<0.00001	<0.00001
7	0.00775	0.00018	<0.00001	<0.00001
8	0.00389	0.00005	<0.00001	<0.00001
9	0.00195	0.00001	<0.00001	<0.00001
10	0.00098	<0.00001	<0.00001	<0.00001
$E_{\text{mRC}}(\alpha)$	1.60666	1.02307	0.77658	0.71219

With  $\alpha=10$ , Micro-Random code requires about 0.7 overhead symbols on average to achieve complete decoding, as compared with Random code ( $\alpha=1$ ), which requires 1.6 overhead symbols. No further improvement can be made for  $\alpha>10$  as it explained in Section 4.3.1.

### 4.3.3 Decoding Steps and Decoding Complexity

Recall that Micro-Random code dimensions the symbol size in order to gain higher PCD with fewer overhead symbols. In particular, its generator matrix has more rows but less columns, i.e.,  $\mathbf{G}^{(\alpha k+10)\times\alpha k}$  as compared to Random code, i.e.,  $\mathbf{G}^{(k+10)\times k}$ . Therefore, it is non-trivial to examine the process to invert  $\mathbf{G}^{(\alpha k+10)\times\alpha k}$ .

We present the pseudo-code of Gaussian elimination in Algorithm 4.1, where it solves the augmented matrix  $[\mathbf{G}^{(\alpha k+10)\times\alpha k} | \mathbf{X}^{(\alpha k)\times(l/\alpha)}]$  by forming

upper triangular matrix in Step 1 to 9 and backward substitution in Step 10 to 16 and the total XOR operations is expressed as

$$\text{Step}_{\text{GE}} = \sum_{i=1}^{\alpha k} \sum_{j=i+1}^{\alpha k+10} \left( \alpha k + \frac{l}{\alpha} \right) + \sum_{i=2}^{\alpha k} \sum_{j=i+1}^{\alpha k} \frac{l}{\alpha}. \quad (49)$$

Note that the  $\text{Step}_{\text{GE}}$  may be slightly different for other optimised versions of Gaussian elimination.

Substituting  $\alpha=1$  and  $\alpha=10$  in  $\text{Step}_{\text{GE}}$  respectively, we obtain

$$[\text{Step}_{\text{GE}}]_{\alpha=1} = \frac{1}{2}k^3 + k^2l + \frac{19}{2}k^2 + 8kl + l, \quad (50)$$

and ,

$$[\text{Step}_{\text{GE}}]_{\alpha=10} = 500k^3 + 10k^2l + 950k^2 + 8kl + \frac{1}{10}l. \quad (51)$$

Clearly, the decoding complexity does not change for different  $\alpha$ , i.e., it stays  $O(k^3)$ . Increasing  $\alpha$  introduces more symbols (micro symbols) to the decoding process without affecting the complexity of the algorithm.

Taking the difference in between Eq. 51 and Eq. 50,

$$[\text{Step}_{\text{GE}}]_{\alpha=10} - [\text{Step}_{\text{GE}}]_{\alpha=1} = \frac{999}{2}k^3 + \frac{1881}{2}k^2 + l\left(9k^2 - \frac{9}{10}\right), \quad (52)$$

yields a positive function for all positive integers  $k$  and  $l$ . Therefore,  $\alpha=10$  imposes more decoding steps as compared with  $\alpha=1$ .

Algorithm 4.1: Gaussian elimination for Micro-Random code.

**Form upper triangular matrix**

```
1: for  $i=1,2,\dots,\alpha k$ 
2:    $s$  = Row number that has a leading non-zero  $i$ -entry.
3:   Exchange Row  $s$  and Row  $i$ .
4:   for  $j=i+1,i+2,\dots,\alpha k+10$ 
5:     if Entry  $i$  of Row  $j$  is non-zero
6:       Row  $j$   $\leftarrow$  Row  $j$   $\oplus$  Row  $i$ 
7:     end if
8:   end for
9: end for
```

**Backward substitution**

```
10: for  $i=\alpha k,\alpha k-1,\dots,2$ 
11:   for  $j=i-1,i-2,\dots,1$ 
12:     if Entry  $i$  of Row  $j$  is non-zero then
13:       Row  $j$   $\leftarrow$  Row  $j$   $\oplus$  Row  $i$ 
14:     end if
15:   end for
16: end for
```

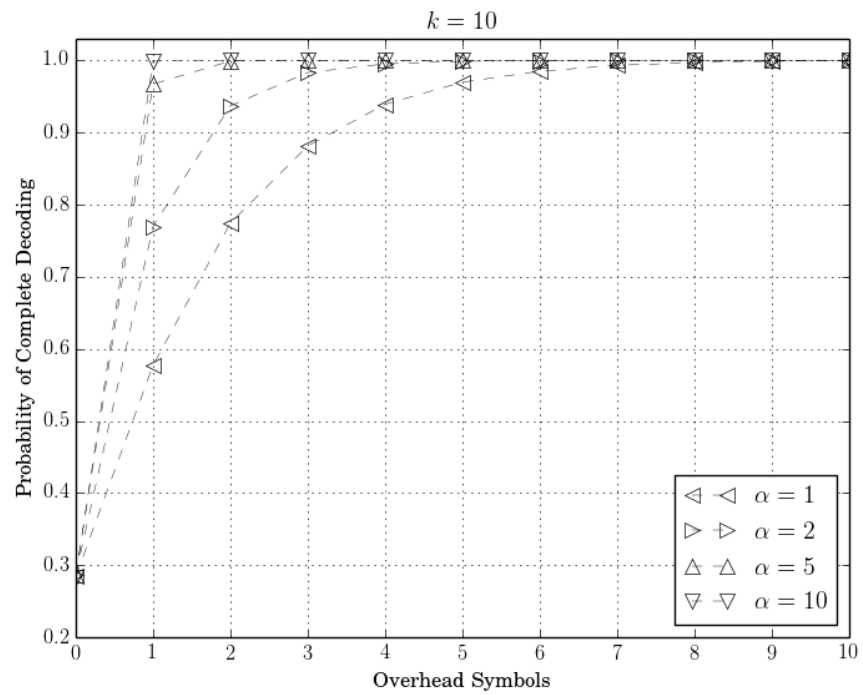
#### 4.4 Simulation Results

We study the performance of Micro-Random code for message length  $k$  and segmentation factor  $\alpha$  with simulation. The symbols are represented with Galois Field GF(2) matrices. The erasure probability of the channel is not important as all the encoded symbols are generated independently with random matrix (generator matrix). All the simulation scenarios have been repeated 1000 times and the average values are presented.

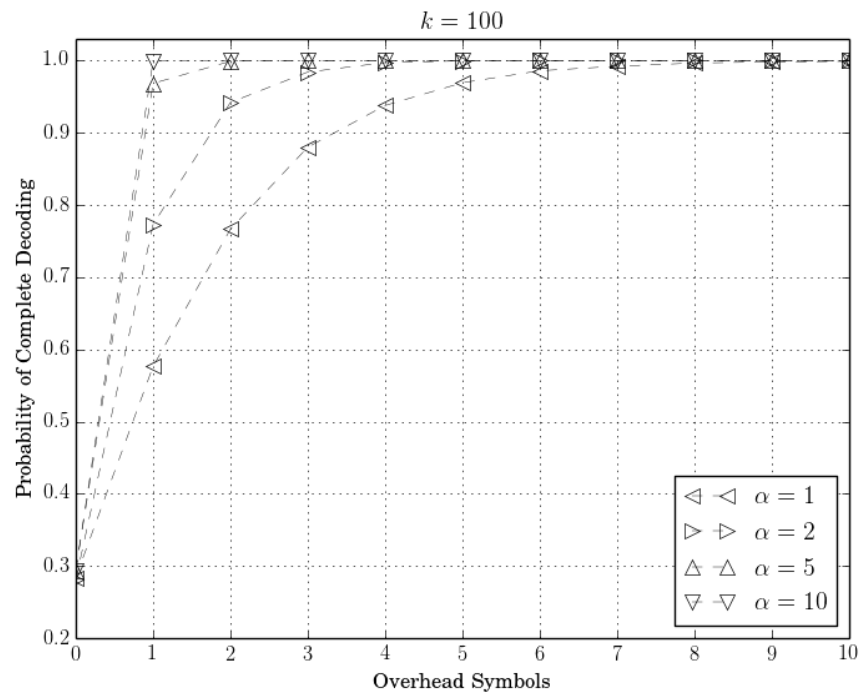
Let the symbol sizes be fixed to 120 bits such that they can be segmented to micro symbols of different sizes ( $\alpha = 1, 2, 4, 5, 8, 10, 12$ , and

15). Then, the message size is fixed to 1200, 12,000, 30,000 and 60,000 bits, which represent messages of 10, 100, 250 and 500 symbols respectively. Note that  $\alpha=1$  also represents Random code, where all the message symbols, micro symbols, encoded micro symbols and encoded symbols have the same dimension.

Figure 4.3 shows PCD for each overhead symbol with different  $\alpha$  for messages (a)  $k=10$  and (b)  $k=100$ . Both graphs look identical. The performance of Random code can be learned from the line  $\alpha=1$ , where it achieves high PCD at ten overhead symbols. As for  $\alpha=2$ , each encoded symbol consists of two encoded micro symbols. Thus, it achieves high PCD at five overhead symbols. The same observation goes for  $\alpha=4$  to 10 – the higher the  $\alpha$ , the faster it reaches the high PCD. However, no significant improvement is observed after  $\alpha=10$ . Such results are consistent with the discussion in Section 4.3.1. We have tried the messages of  $k=50$ , 250 and 500 symbols and they have identical results as shown in Figure 4.3. Overall, the PCD of Micro-Random code does not change with the number of message symbols  $k$  but only the segmentation factor  $\alpha$ .



(a)



(b)

Figure 4.3: The PCD for messages of (a)  $k=10$  and (b)  $k=100$  symbols at various overhead symbols and segmentation factor  $\alpha$ .

Figure 4.4 shows the average number of overhead symbols to achieve complete decoding for messages of  $k=10$  symbols. It shows a decreasing trend with increasing  $\alpha$ , but saturates at about 0.7 when  $\alpha=10$ . The observation supports our previous discussion. There is no way to achieve the high PCD with zero overhead symbols in Micro-Random code. Identical results are observed for messages of  $k=50$ , 250 and 500 symbols.

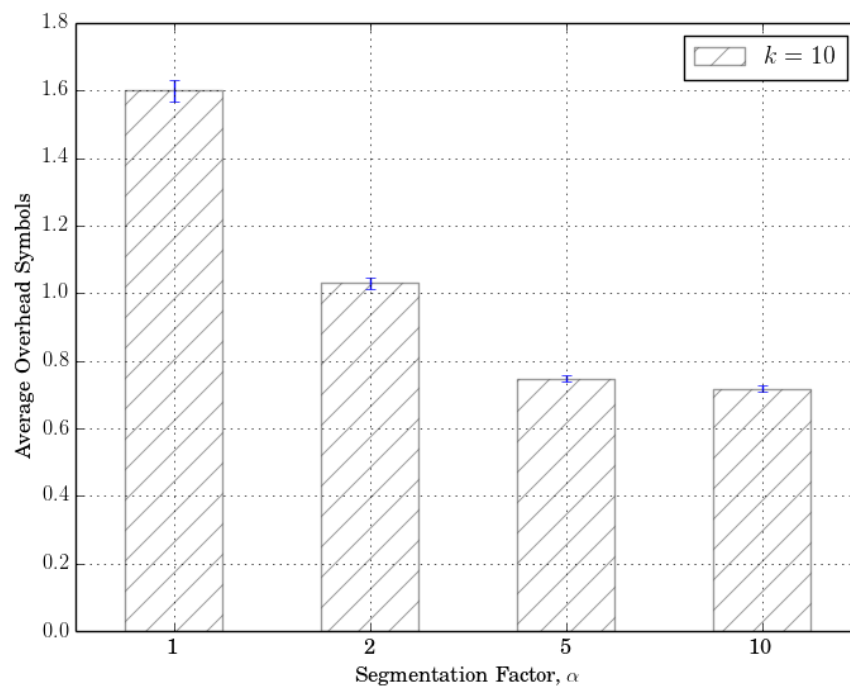


Figure 4.4: The expected overhead symbols for messages of  $k=10$  symbols to achieve complete decoding at various segmentation factor  $\alpha$ .

As mentioned in Chapter 3, Random code reconstructs the original message with Gaussian elimination of decoding complexity  $O(k^3)$ . In order to achieve high PCD with only one overhead symbol, Micro-Random code dimensions each message symbol into  $\alpha=10$  micro symbols. Hence,  $10k$  micro symbols are involved in the encoding and decoding processes. Though



the decoding complexity is still  $O(k^3)$  as  $O((10k)^3) = O(k^3)$ , it is found that the decoding time is relatively higher than Random code as observed in Figure 4.5, i.e., the decoding time increases exponentially with increasing  $\alpha$  as more micro symbols sets involved in the decoding process. Note that we measure the running time with a computer that uses i3 processor and inversion of the matrices of dimensions  $10 \times 10$ ,  $100 \times 100$  and  $250 \times 250$  are done using unoptimised code.

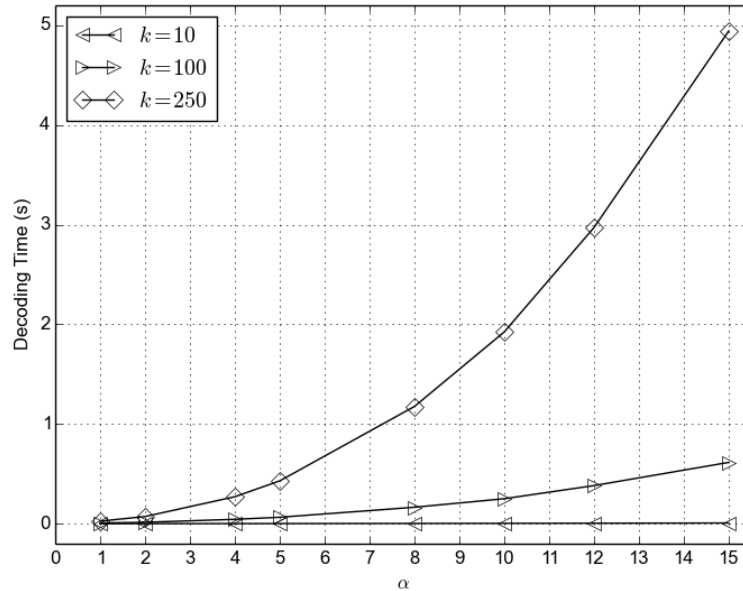


Figure 4.5: The average decoding time needed to reconstruct the original message with Micro-Random code.

#### 4.5 Deploying Micro-Random Code in Resource-Constrained Devices

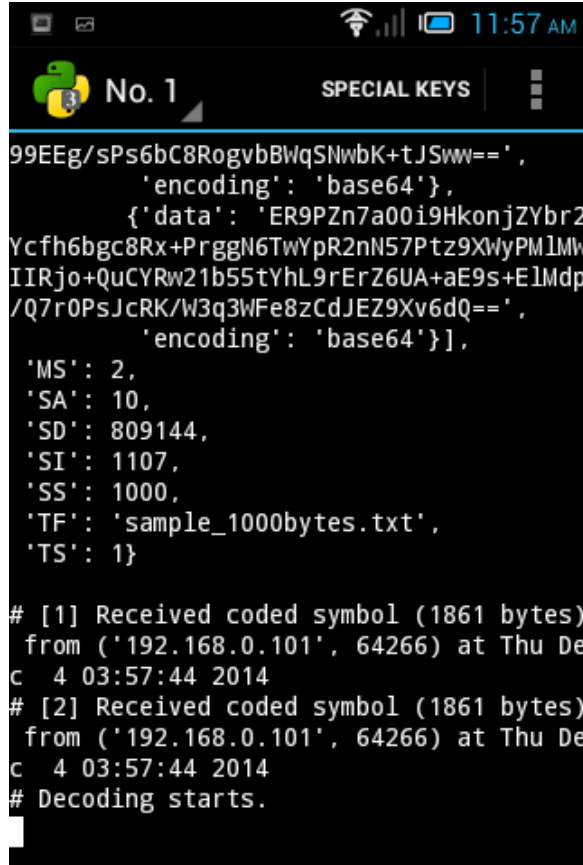
Though Micro-Random code requires extensive computational resources, experiments we have done demonstrate that the time to reconstruct short messages is acceptable in resource-constrained devices.

### 4.5.1 Experiment Setting

In this experiment, the computer encodes the messages with Micro-Random code and broadcast them to two resource-constrained devices (Android devices) via Wi-Fi. Two sample files of 2,000 bytes and 5,000 bytes are used. The symbol size (packet size) is fixed to 1,000 bytes with segmentation factors of  $\alpha=1, 2, 5$  and 10. In other words, messages of  $k=2$  and 5 symbols are used in the experiment. The receivers will gather  $k+\lceil 10/\alpha \rceil$  encoded symbols before reconstructing the original messages with Gaussian elimination. Each scenario was repeated for 100 times.

The computer runs Windows 7 on 64 bits i7 processor with 4GB RAM. The two Android devices are Asus Nexus 7 and Alcatel One Touch (OT). Asus Nexus 7 is a mid-range Android device that runs on Quad-Core 1.2GHz with 1GB RAM (Asus Nexus 7: Hardware specification, 2014). On the other hand, Alcatel OT is a low-end Android device with 1GHz CPU and 512MB RAM (Alcatel One Touch: Hardware specification, 2014).

We use Python 3.4 (Python, a programming language, 2014) as the main scripting language. Additionally, external python modules like bitstring (bitstring, 2014) and serpent (Serpent, 2014) are used in order to manipulate the bit arrays and serialisation. We also use CRCmod (CRCmod, 2014) to compute the integrity of the packets. The Gaussian elimination algorithm can be found in (Gaussian elimination, 2014).



```
99EEg/sPs6bC8RogvbBWqSNwbK+tJSww==',
  'encoding': 'base64'},
  {'data': 'ER9PZn7a00i9HkonjZYbr2
Ycfh6bgc8Rx+PrggN6TwYpR2nN57Ptz9XWyPMlMW
IIRjo+QuCYRw21b55tYhL9rErZ6UA+aE9s+ElMdp
/Q7r0PsJcRK/W3q3WFe8zCdJEZ9Xv6dQ==',
  'encoding': 'base64'}],
'MS': 2,
'SA': 10,
'SD': 809144,
'SI': 1107,
'SS': 1000,
'TF': 'sample_1000bytes.txt',
'TS': 1}

# [1] Received coded symbol (1861 bytes)
from ('192.168.0.101', 64266) at Thu De
c 4 03:57:44 2014
# [2] Received coded symbol (1861 bytes)
from ('192.168.0.101', 64266) at Thu De
c 4 03:57:44 2014
# Decoding starts.
```

Figure 4.6: The screen-shot of QPython running in Alcatel OT.

We run the Python scripts on Android devices by using QPython framework (QPython, 2014) as shown in Figure 4.6. It is a script engine that that runs Python scripts seamlessly on Android devices.

#### 4.5.2 Experiment Result

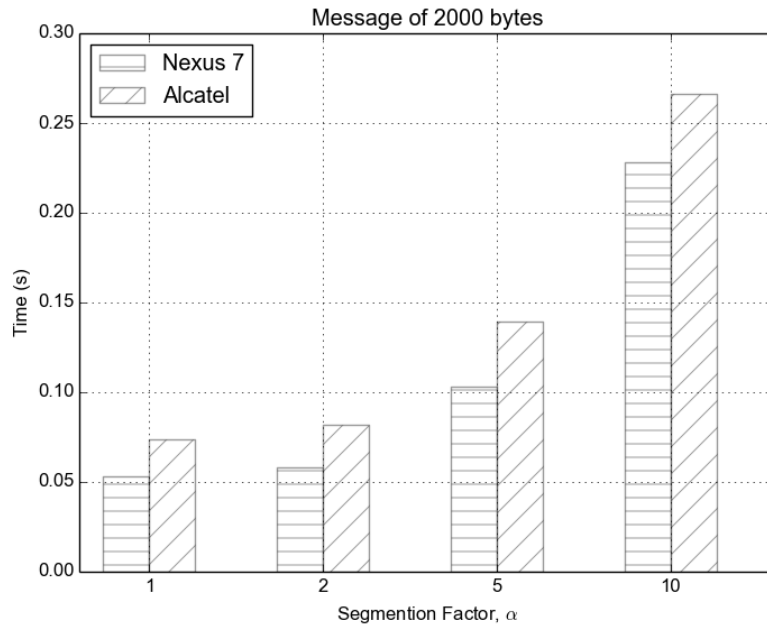
Figure 4.7 presents the average decoding time for messages of (a) 2000 bytes and (b) 5000 bytes in Asus Nexus 7 and Alcatel OT. Generally, the larger the message size, the longer is the time needed to reconstruct the original messages. Additionally, as the segmentation factor  $\alpha$  increases, Gaussian elimination needs to work on a larger matrix (in term of rows) and therefore a longer time is needed to reconstruct the original messages. For

example, the time needed to reconstruct original messages of 2,000 bytes and 5,000 bytes with  $\alpha=10$  is much longer than the case where  $\alpha=1$ . Such results are consistent with our findings in Section 4.4.

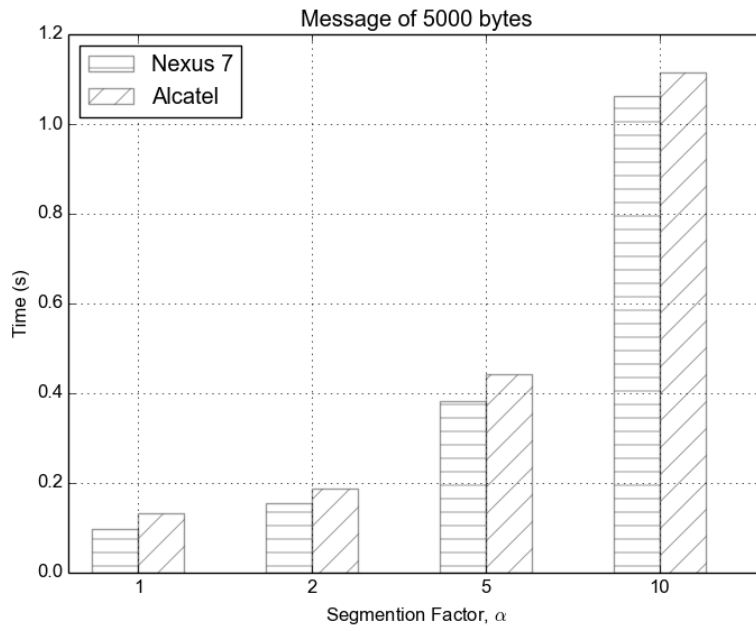
The Gaussian Elimination that we use does not utilise the multi-core technology of the devices. Hence, both Android devices have about the same decoding time though Alcatel OT appears to be a low-end device.

#### **4.6 Summary**

Utilising the previous rigorous results and bounds given in Chapter 3, Micro-Random code achieves a better PCD at fewer overhead symbols by dimensioning the symbols appropriately, but with the trade-off of higher decoding time.



(a)



(b)

Figure 4.7: The average decoding time of Micro-Random code for messages of (a) 2,000 bytes, and (b) 5,000 bytes on Android devices.

## CHAPTER 5

### SYSTEMATIC RANDOM CODE

This chapter proposes a variant of Random code, i.e., systematic Random (SYSR) code that achieves better decoding complexity than Random code on average.

#### 5.1 Systematic Rateless Erasure Code

A *systematic* rateless erasure code uses the original message as part of the encoded symbols and the rest are generated with a sequence of bit operations. Correspondingly, the receiver reconstructs the original message instantly if the first  $k$  encoded symbols are received intact. In case of any lost symbols, the receiver will reconstruct the message from the subsequent encoded symbols.

Figure 5.1 illustrates the encoding and decoding processes of a systematic rateless erasure code on a message of  $k=4$  symbols. The white circles denote the original message symbols and the hashed circles are the generated encoded symbols. Note that the encoder generates potentially an infinite number of encoded symbols with the original message as part of the encoded symbols.

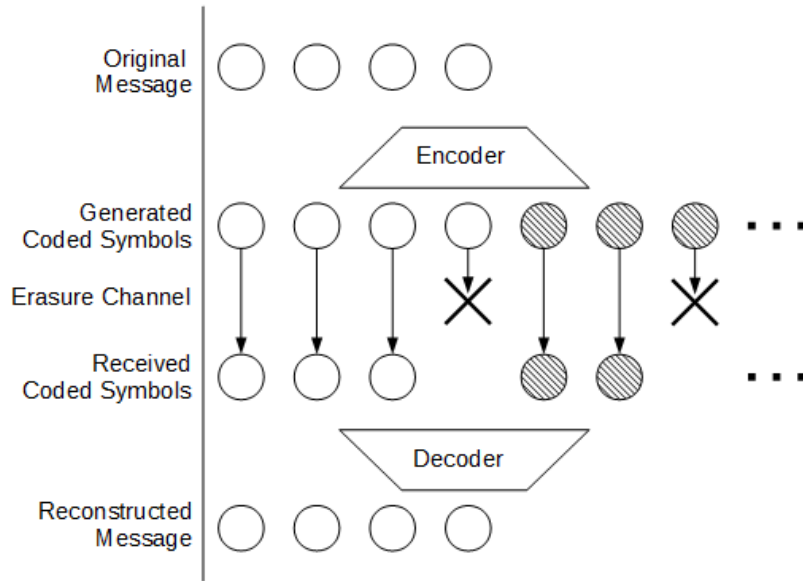


Figure 5.1: The encoding and decoding processes of systematic rateless erasure code.

## 5.2 Systematic Random Code

In this section, we propose a systematic rateless erasure code, namely systematic Random (SYSR) code that is built on top of a random matrix framework. SYSR code outperforms Random code in erasure channels of low error rate and performs as good as Random code in very lossy channels (e.g., where erasure probability,  $\rho=0.5$ ). We will elaborate on the decoding process with an example for the ease of explanation.

### 5.2.1 Encoding Message

Systematic Random code generates two type of encoded symbols.

- Part I encoded symbols are the first  $k$  encoded symbols of systematic Random code (i.e.,  $x_0, x_1, \dots, x_{k-1}$ ). They are also the symbols of the original message.

- Part II encoded symbol refers to the encoded symbols starting from  $(k+1)$ -th onwards (i.e.,  $x_k, x_{k+1}, \dots$ ) and they are generated with Random code.

A message of  $k$  symbols with each symbol size  $l$  bits is denoted as a matrix of dimensions  $k \times l$ , i.e.,  $\mathbf{M}^{k \times l}$ . The Part I encoded symbols are the original message symbols and its generator matrix is an identity matrix. Then, each Part II encoded symbol (coded row matrix)  $\mathbf{X}^{1 \times l}$  is independently generated by multiplying a random row matrix  $\mathbf{G}^{1 \times k}$  with  $\mathbf{M}^{k \times l}$ , i.e.,

$$\begin{aligned}
\mathbf{X}^{1 \times l} &= \mathbf{G}^{1 \times k} \times \mathbf{M}^{k \times l} \\
&= \begin{bmatrix} g_{0,0} & \dots & g_{0,k-1} \end{bmatrix} \times \begin{bmatrix} m_{0,0} & \dots & m_{0,l-1} \\ \vdots & \ddots & \vdots \\ m_{k-1,0} & \dots & m_{k-1,l-1} \end{bmatrix} \\
&= \begin{bmatrix} x_{0,0} & \dots & x_{0,l-1} \end{bmatrix}.
\end{aligned} \tag{53}$$

We assume that the receiver will receive both the encoded symbol  $\mathbf{X}^{1 \times l}$  and the corresponding generator matrix  $\mathbf{G}^{1 \times k}$  at the same time as they are embedded in the same packet during the transmission. Subsequently, the receiver reconstructs the original message with Gaussian elimination when it has a full rank generator matrix.

### 5.2.2 Message Reconstruction

Since the Part I encoded symbols are the original message symbols, the decoding process is completed instantly if they are received intact. Otherwise, at least  $k+10$  Part I and II encoded symbols are required in order to reconstruct the original message with high PCD.



Assume that the receiver has received  $a$  Part I encoded symbols and  $k+10-a$  Part II encoded symbols from the lossy erasure channel. Augmenting the encoded symbol matrices  $\mathbf{X}^{(k+10) \times l}$  with their respective generator row matrices  $\mathbf{G}^{(k+10) \times k}$ , we can express them as

$$\left[ \mathbf{G}^{(k+10) \times k} \mathbf{X}^{(k+10) \times l} \right] = \left[ \begin{array}{ccc|ccc} \tilde{\mathbf{g}}_{0,0} & \cdots & \tilde{\mathbf{g}}_{0,k-1} & \tilde{\mathbf{x}}_{0,0} & \cdots & \tilde{\mathbf{x}}_{0,l-1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathbf{g}}_{a-1,0} & \cdots & \tilde{\mathbf{g}}_{a-1,k-1} & \tilde{\mathbf{x}}_{a-1,0} & \cdots & \tilde{\mathbf{x}}_{a-1,l-1} \\ \mathbf{g}_{a,0} & \cdots & \mathbf{g}_{a,k-1} & \mathbf{x}_{a,0} & \cdots & \mathbf{x}_{a,l-1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{g}_{k-9,0} & \cdots & \mathbf{g}_{k-1,k-1} & \mathbf{x}_{k-9,0} & \cdots & \mathbf{x}_{k-1,l-1} \end{array} \right]. \quad (54)$$

where  $g_{i,j}$  ( $x_{i,j}$ ) denotes the  $i$ -th row  $j$ -th entry in the generator matrix (encoded symbol matrix). We use the tilde notations (e.g.,  $\tilde{\mathbf{g}}$  and  $\tilde{\mathbf{x}}$ ) to denote the Part I row matrices.

Before reconstructing the original message with Gaussian elimination, we apply the following steps to reduce the dimensions of the augmented matrix.

**Step 1:** Identify a Part I generator row matrix. Then, denote the position of the selected Part I generator as row  $i$  and its non-zero entry at column  $j$ .

**Step 2:** Add (XOR) row  $i$  of both generator and encoded symbols row matrices to those Part II row matrices, for which their  $j$  entries are non-zero.

**Step 3:** Remove row  $i$  and column  $j$  from the augmented matrix.

**Step 4:** Repeat from Step 1 until all the Part I row matrices are removed.

We illustrate the matrix reduction with an example as follows. Assume a message of  $k=4$  symbols and symbol size  $l=2$ . The receiver has  $a=2$  Part I encoded symbols and  $k+10-a=12$  Part II encoded symbols. Then, the augmented matrix can be represented as

$$[\mathbf{G}^{14 \times 4} | \mathbf{X}^{14 \times 2}] = \left[ \begin{array}{cccc|cc} 1 & 0 & 0 & 0 & \tilde{x}_{0,0} & \tilde{x}_{0,1} \\ 0 & 1 & 0 & 0 & \tilde{x}_{1,0} & \tilde{x}_{1,1} \\ 1 & 0 & g_{2,2} & g_{2,3} & x_{2,0} & x_{2,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & g_{13,2} & g_{13,3} & x_{13,0} & x_{13,1} \end{array} \right]. \quad (55)$$

Note that the first two row matrices belong to Part I and the rest are Part II. Some values are defined explicitly as 0 or 1 for ease of explanation.

In Step 1, we select the first Part I encoded symbol in the generator matrix and its first entry is non-zero, i.e.  $i=0$  and  $j=0$ . In Step 2, we add row  $i$  to those Part II rows, in which their  $j$  entries are non-zero. Then, row  $i$  and column  $j$  are removed as instructed in Step 3. The remaining augmented matrix becomes

$$[\mathbf{G}^{13 \times 3} | \mathbf{X}^{13 \times 2}] = \left[ \begin{array}{ccc|cc} 1 & 0 & 0 & \tilde{x}_{1,0} & \tilde{x}_{1,1} \\ 0 & g_{2,2} & g_{2,3} & x_{2,0} \oplus \tilde{x}_{0,0} & x_{2,1} \oplus \tilde{x}_{0,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & g_{13,2} & g_{13,3} & x_{13,0} & x_{13,1} \end{array} \right]. \quad (56)$$

We still have one Part I encoded symbol at second row ( $i=1$ ). Hence, we repeat Step 1 and Step 2, adding row  $i$  to the rest of Part II rows that have non-zero values in entries  $j$ . After Step 3, we have

$$[\mathbf{G}^{12 \times 2} | \mathbf{X}^{12 \times 2}] = \left[ \begin{array}{cc|cc} g_{2,2} & g_{2,3} & x_{2,0} \oplus \tilde{x}_{0,0} & x_{2,1} \oplus \tilde{x}_{0,1} \\ \vdots & \vdots & \vdots & \vdots \\ g_{13,2} & g_{13,3} & x_{13,0} \oplus \tilde{x}_{1,0} & x_{13,1} \oplus \tilde{x}_{1,1} \end{array} \right]. \quad (57)$$

Since there is no more Part I encoded symbol, Gaussian elimination will process the rest of the generator matrix of dimensions  $12 \times 2$  (augmented matrix of dimensions  $12 \times 4$ ) as usual.

### 5.3 Performance Analysis

This section analyses the performance of SYSR code in terms of PCD and its decoding algorithm (total number of XOR row operations).

#### 5.3.1 Probability of Complete Decoding

Recall that there are two cases in the decoding process:

- **Case I:** Receive all the Part I encoded symbols intact and the message is reconstructed instantly.
- **Case II:** Receive total of  $k+10$  Part I and II encoded symbols and the message is reconstructed with high PCD.

Given the channel erasure probability  $\rho$ , the probability that Case I occurs is a binomial function,  $\text{Binom}(0, k, \rho)$ . Correspondingly, the probability that Case II occurs is multiplication of  $1 - \text{Binom}(0, k, \rho)$  with Random code's PCD equation (Eq. (21)). Hence, given a message of  $k$  symbols, the probability for SYSR code to achieve complete decoding with  $m$  overhead symbols in channel with erasure probability,  $\rho$  is

$$Q_{\text{SYSR}}(m, k, \rho) = \text{Binom}(0, k, \rho) - [1 - \text{Binom}(0, k, \rho)]Q_{\text{kol}}(m). \quad (58)$$

Then, the PMF of SYSR code is

$$P_{\text{SYSR}}(m, k, \rho) = Q_{\text{SYSR}}(m, k, \rho) - Q_{\text{SYSR}}(m-1, k, \rho), \quad (59)$$

and

$$P_{\text{SYSR}}(0, k, \rho) = Q_{\text{SYSR}}(0, k, \rho). \quad (60)$$

Correspondingly, the expected overhead symbols can be expressed as

$$\sum_{m=0}^{\infty} m P_{\text{SYSR}}(m, k, \rho). \quad (61)$$

Table 5.1: The PCD of SYSR code for  $k=10$  and  $50$ .

	<b>k=10</b>			<b>k=50</b>		
	<b><math>\rho = 0.01</math></b>	<b><math>\rho = 0.1</math></b>	<b><math>\rho = 0.5</math></b>	<b><math>\rho = 0.01</math></b>	<b><math>\rho = 0.1</math></b>	<b><math>\rho = 0.5</math></b>
<b>m=0</b>	0.931995	0.536772	0.289483	0.719076	0.292454	0.288788
<b>m=1</b>	0.959609	0.724866	0.577989	0.833145	0.579753	0.577576
<b>m=2</b>	0.978018	0.850262	0.770326	0.909192	0.771286	0.770102
<b>m=3</b>	0.988537	0.921917	0.880233	0.952647	0.880734	0.880116
<b>m=4</b>	0.994147	0.960133	0.938850	0.975823	0.939106	0.938791
<b>m=5</b>	0.997043	0.979857	0.969104	0.987784	0.969233	0.969074
<b>m=6</b>	0.998514	0.989876	0.984471	0.993860	0.984536	0.984456
<b>m=7</b>	0.999255	0.994925	0.992215	0.996922	0.992248	0.992208
<b>m=8</b>	0.999627	0.997459	0.996103	0.998459	0.996119	0.996099
<b>m=9</b>	0.999813	0.998729	0.998050	0.999229	0.998058	0.998048
<b>m=10</b>	0.999907	0.999364	0.999025	0.999614	0.999029	0.999024

The PCD of SYSR code for  $k=10$  and  $50$  with increasing  $m$  are presented in Table 5.1. Generally, SYSR code has a higher PCD than Random code for the same  $m$  when both  $\rho$  and  $k$  are small (comparing with Table 3.1). Both codes have about the same PCD when  $\rho$  and  $k$  increase. Same observation is found on the expected overhead symbols presented in Table 5.2, where 1.6 overhead symbols are required when  $\rho$  and  $k$  increase.

Table 5.2: The expected overhead symbols to achieve complete decoding in SYSR code for increasing  $k$  and  $\rho$ .

	$\rho = 0.01$	$\rho = 0.1$	$\rho = 0.5$
<b>k=10</b>	0.153629	1.046475	1.605126
<b>k=50</b>	0.634635	1.598415	1.606695
<b>k=100</b>	1.018593	1.606652	1.606695
<b>k=200</b>	1.391431	1.606695	1.606695

### 5.3.2 Decoding Algorithm of Random Code

Generally, Gaussian elimination involves two algorithms – transforming the generator matrix (augmented matrix) into an upper triangular matrix and subsequently to an identity matrix with backward substitution. We will present the pseudo-codes for Random code and SYSR code to form upper triangular matrices in this section and Section 5.3.3, respectively. Note that both rateless erasure codes employ the same backward substitution and hence it will not be discussed here.

Algorithm 5.1 and its corresponding flow chart in Figure 5.2 present the process to form an upper triangular matrix. The `randMat` represents the generator matrix and `resultMat` in second line is a blank matrix, where the pivoting rows (the row with the first non-zero element at  $i$ -th entry) will be added later on.

The for-loop in lines 3 to 11 will loop through the  $k$  columns of the generator matrix in order to select the  $i$ -th pivoting row in each iteration (line 4). Then, the pivoting row will be moved from the `randMat` to the `resultMat` (line 5). Next, it searches the remaining rows in `randMat`, for which their  $i$ -th

entries are non-zero and XORs them with the pivoting row (line 8). At the end of line 12, the algorithm returns an upper triangular matrix.

Algorithm 5.1: The pseudo-code for forming an upper triangular matrix with Random code.

```
1: function FormUpperTriangularMatrix (randMat)
2:   resultMat = null matrix
3:   for  $i=0, 1, \dots, k-1$  do
4:     pivotRow = SearchPivotRow(  $i$  ,randMat)
5:     Move pivotRow from randMat to resultMat
6:     for row in randMat do
7:       if  $i$  -th entry of row is non-zero then
8:         row  $\leftarrow$  row  $\oplus$  pivotRow
9:       end if
10:    end for
11:  end for
12:  return resultMat
13: end function
```

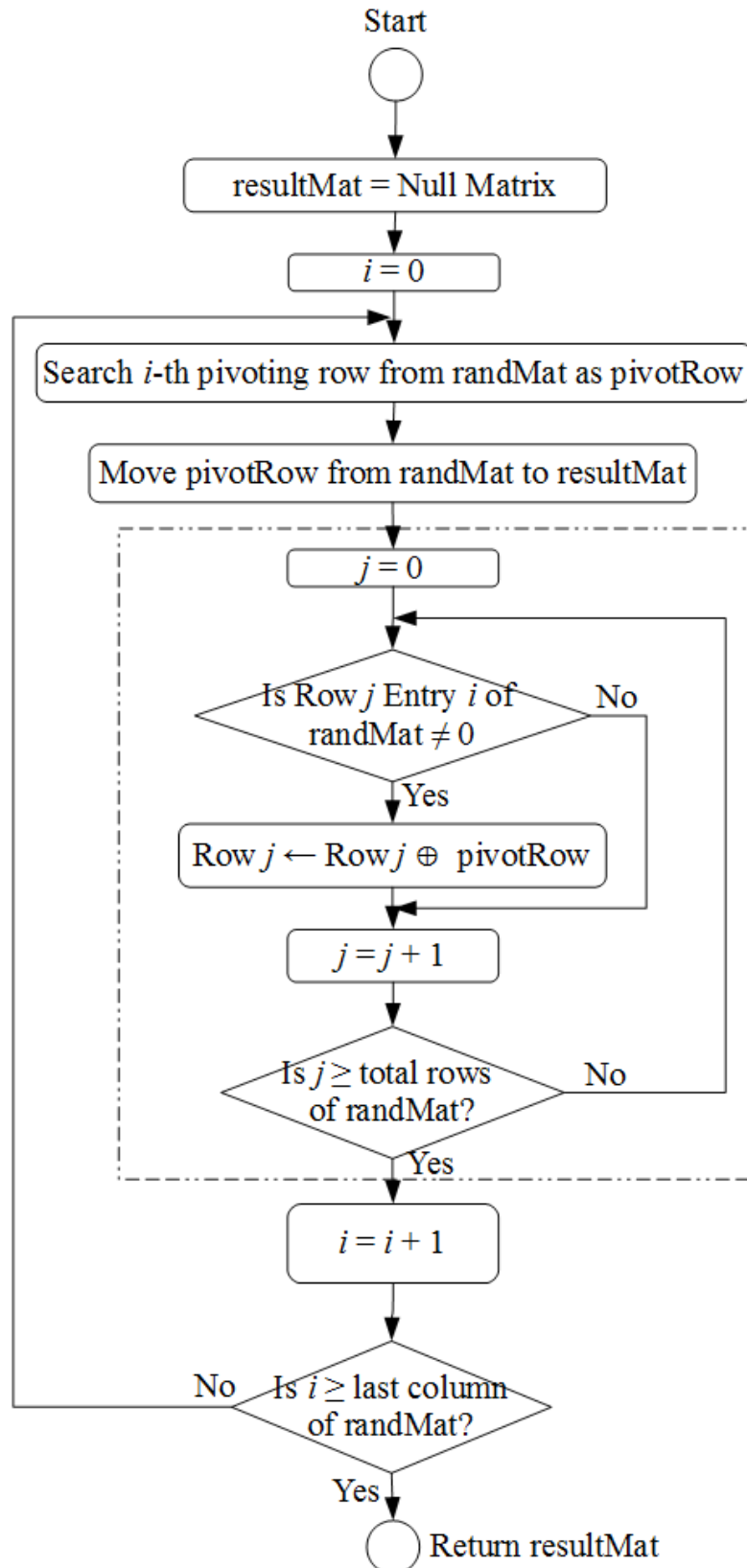


Figure 5.2: The flow chart of Random code to form upper triangular matrix.

### 5.3.3 Decoding Algorithm of SYSR Code

The pseudo-code for SYSR code to form upper triangular matrix is presented in Algorithm 5.2 and the corresponding flow chart in Figure 5.3. The  $idnMat$  and  $randMat$  denote the generator matrices of Part I and II encoded symbols, respectively. Basically, Algorithm 5.2 has a similar structure to Algorithm 5.1 except that the former will attempt to search the pivoting rows from  $idnMat$  first before getting them from the  $randMat$ , if not found (lines 4-10).

Algorithm 5.2: The pseudo-code for forming an upper triangular matrix with SYSR code.

```
1: function FormUpperTriangularMatrix (idnMat, randMat)
2:   resultMat = null matrix
3:   for  $i=0, 1, \dots, k-1$  do
4:     Search  $i$ -th pivoting row from idnMat as pivotRow
5:     if pivotRow  $\neq$  NOT_FOUND then
6:       Move pivotRow from idnMat to resultMat
7:     else
8:       Search  $i$ -th pivoting row from randMat as pivotRow
9:       Move pivotRow from randMat to resultMat
10:    end if
11:    for row in randMat do
12:      if  $i$ -th entry of row is non-zero then
13:        row  $\leftarrow$  row  $\oplus$  pivotRow
14:      end if
15:    end for
16:  end for
17:  return resultMat
18: end function
```



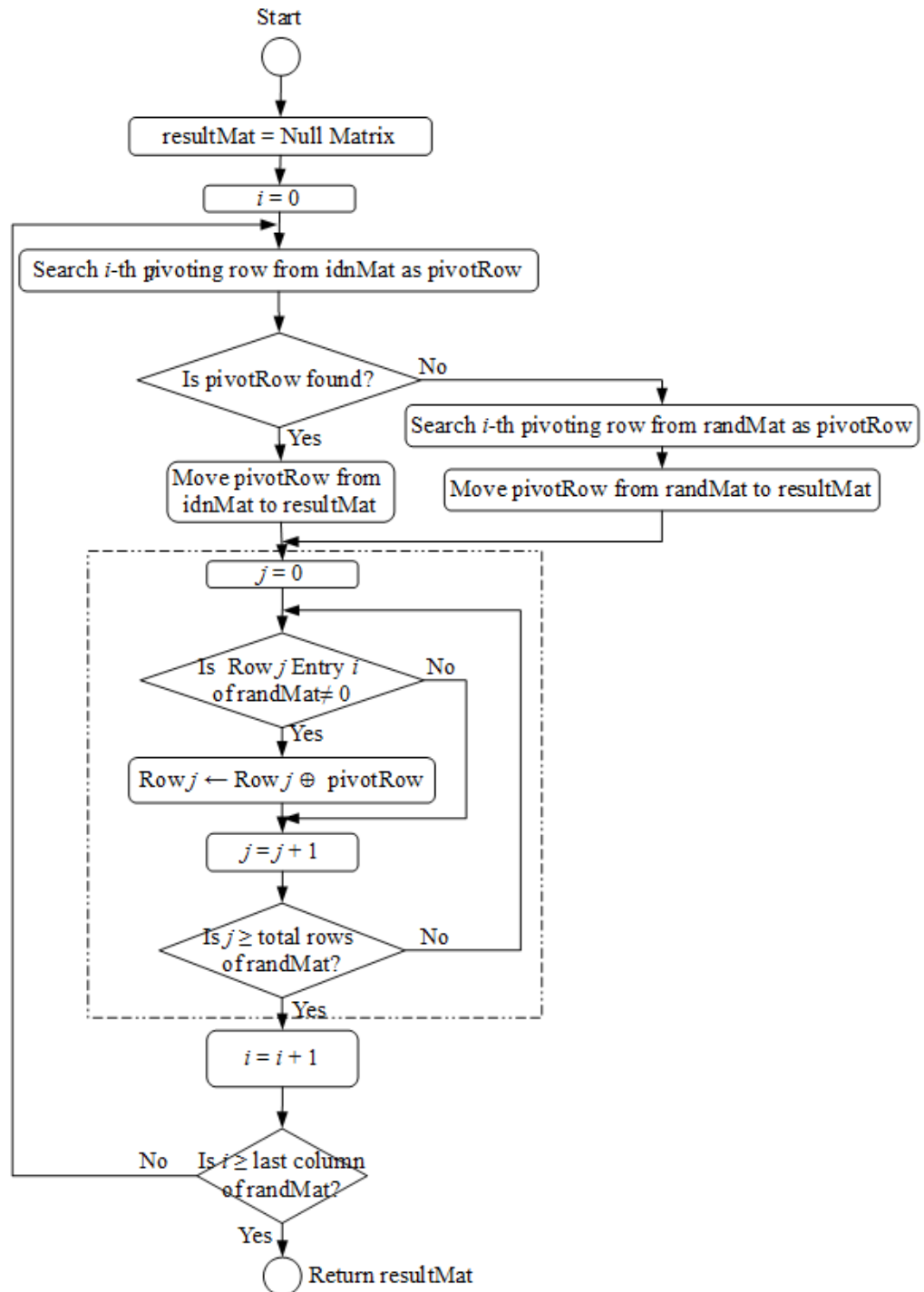


Figure 5.3: The flow chart for SYSR code to form upper triangular matrix.

To explain the improved decoding algorithm in SYSR code, assume  $a$  Part I encoded symbols are received and they need to XOR with  $k+10-a$

Part II rows, i.e.,  $\sum_{i=1}^a \sum_{j=1}^{k+10-a} (k+l)$ , where one row operation consists of  $k+l$

XOR operations. After that,  $k-a$  pivoting rows will be selected from randMat for similar XOR row operations and then removed (corresponding to

the expression  $\sum_{i=1}^{k-a} \sum_{j=i+1}^{k+10-a} (k+l)$ ). Hence, we denote the number of XOR

operations for SYSR code to form an upper triangular matrix as  $\chi$ , where

$$\begin{aligned}
\chi &= \sum_{i=1}^a \sum_{j=1}^{k+10-a} (k+l) + \sum_{i=1}^{k-a} \sum_{j=i+1}^{k+10-a} (k+l) \\
&= (k+l) \left[ \sum_{i=1}^a \sum_{j=1}^{k+10-a} (1) + \sum_{i=1}^{k-a} \sum_{j=i+1}^{k+10-a} (1) \right] \\
&= (k+l) \left[ \frac{1}{2} k^2 + \frac{19}{2} k + \frac{1}{2} (a - a^2) \right].
\end{aligned} \tag{62}$$

Since  $0 \leq a \leq k$ , Eq. 62 is a non-increasing function and each received Part I encoded symbols will reduce the number of XOR operations, except when  $a=1$ . Note that if all the Part I encoded symbols are received intact (i.e.,  $a=k$ ), the original message can be reconstructed instantly without forming an upper triangular matrix with Algorithm 5.2.

Eventually, if none of the Part I encoded symbols are received (i.e.,  $a=0$ ),

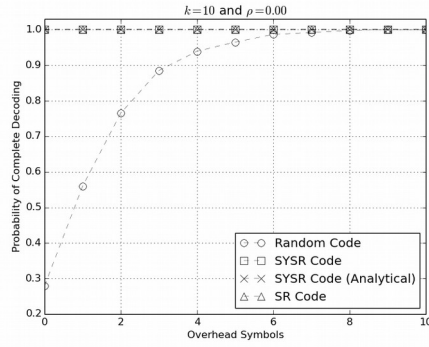
$$[\chi]_{a=0} = \frac{1}{2} k^3 + \frac{1}{2} k^2 l + \frac{19}{2} k^2 + \frac{19}{2} k l, \tag{63}$$

and SYSR code has the same decoding complexity as Random code in constructing an upper triangular matrix, i.e.,  $O(k^3)$ .

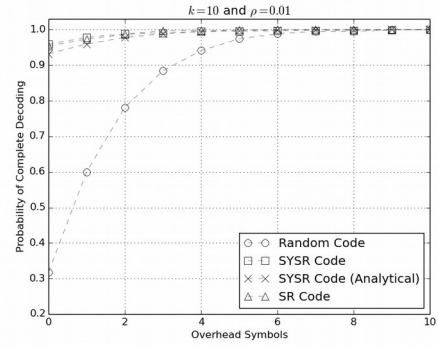
## 5.4 Numerical Result

In this section, we measure the performance of Random code, SYSR code and SR code (introduced in Chapter 7) in terms of PCD and the average number of XOR row operations to reconstruct the original messages (i.e., forming upper triangular matrix and backward substitution). The simulator consists of three main components – the sender (encoder), which generates the encoded symbols continuously, the erasure channel that drops the encoded symbols with erasure probability  $\rho$  and the receiver (decoder), which attempts to reconstruct the original message when it has  $k+10$  encoded symbols. Each simulation scenario is repeated for 1000 times and the averaged results are presented.

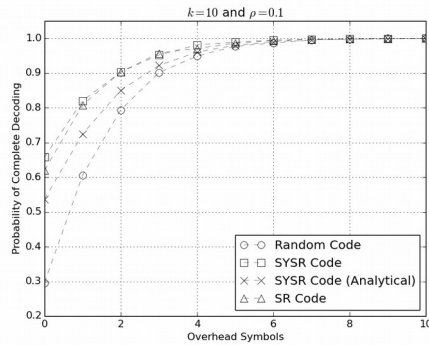
Figure 5.4 illustrates the PCD of SYSR code (simulation and analytical), Random code and SR code with incremental overhead symbols for message length of  $k=10$  symbols. The interpolated lines in the graphs are used to show the trends of the PCD with respect to each incremental overhead symbol. When  $\rho=0.00$ , both SYSR and SR code are able to reconstruct the original message with high PCD from the first  $k$  encoded symbols (Part I encoded symbols) and no overhead symbol is required (Figure 5.4(a)). However, as  $\rho$  increases they require at most ten overhead symbols to achieve high PCD (Figure 5.4 (b) to (d)). Meanwhile, Random code requires  $k+10$  encoded symbols to reach high PCD in all the cases.



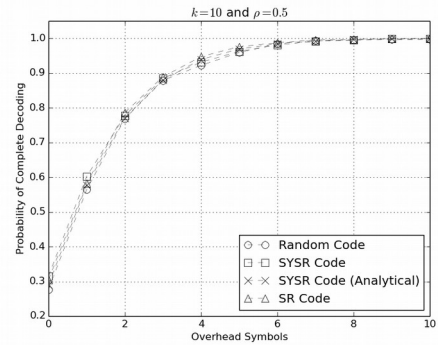
(a)



(b)



(c)



(d)

Figure 5.4: The PCD of Random code, SYSR code and SR code for messages of  $k=10$  symbols in channels of various erasure probabilities.

We observe that the PCD of SYSR code is similar to SR code in all the cases. They outperform Random code when the channel erasure probability is small and perform as good as Random code in a very lossy channel (e.g.,  $\rho=0.5$ ). Similar PCD results are obtained for messages with  $k=10$ , 50, 100 and 250 symbols in channels of various erasure probabilities. Since SYSR code is able to achieve high PCD with ten overhead symbols irrespective of the message length, it needs an average of 1.6 overhead symbols to achieve complete decoding in lossy channels as shown in Figure 5.5.

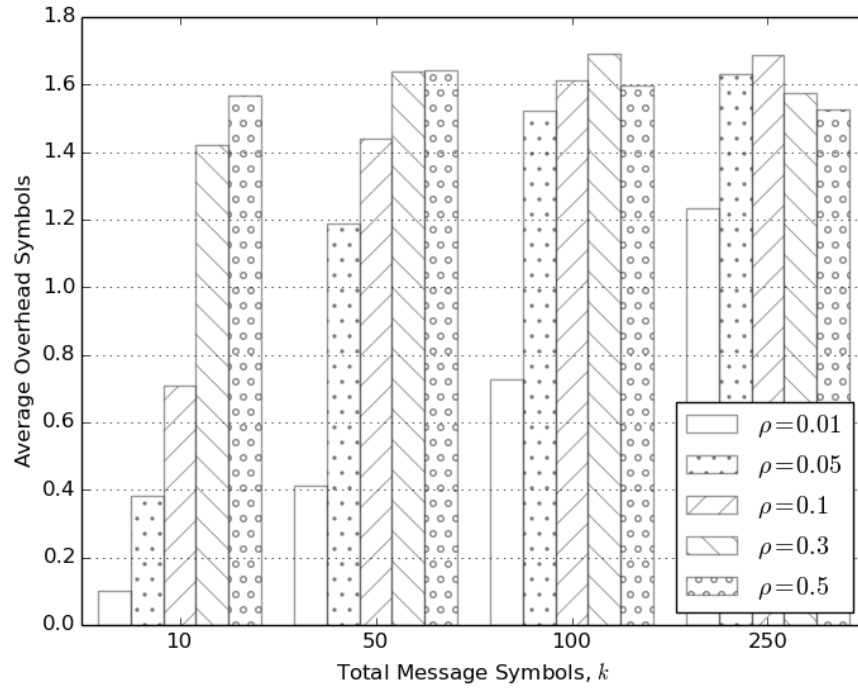


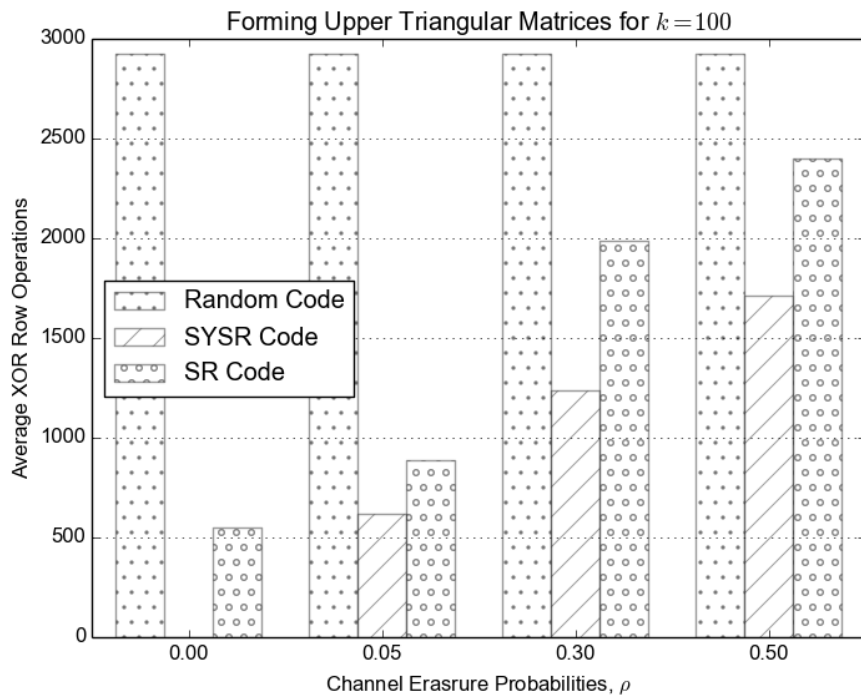
Figure 5.5: The average overhead symbols for SYSR code to achieve complete decoding in channels of various erasure probabilities.

Figure 5.6 presents the average number of XOR row operations for Random code, SYSR code and SR code to form upper triangular matrices and backward substitution for messages of  $k=100$  symbols subject to various channel erasure probabilities. Overall, SYSR code requires the least XOR row operations among the rest. In particular, when  $\rho=0.00$  SYSR code receives all the Part I encoded symbols intact and the original message can be reconstructed instantly. As  $\rho$  increases, more Part II encoded symbols are required in reconstructing the original messages due to missing Part I encoded symbols. Therefore, the total XOR row operations increases. Meanwhile, Random code requires about the same total XOR row operations in all the cases as its generator matrices only consist of randomly distributed binary values.

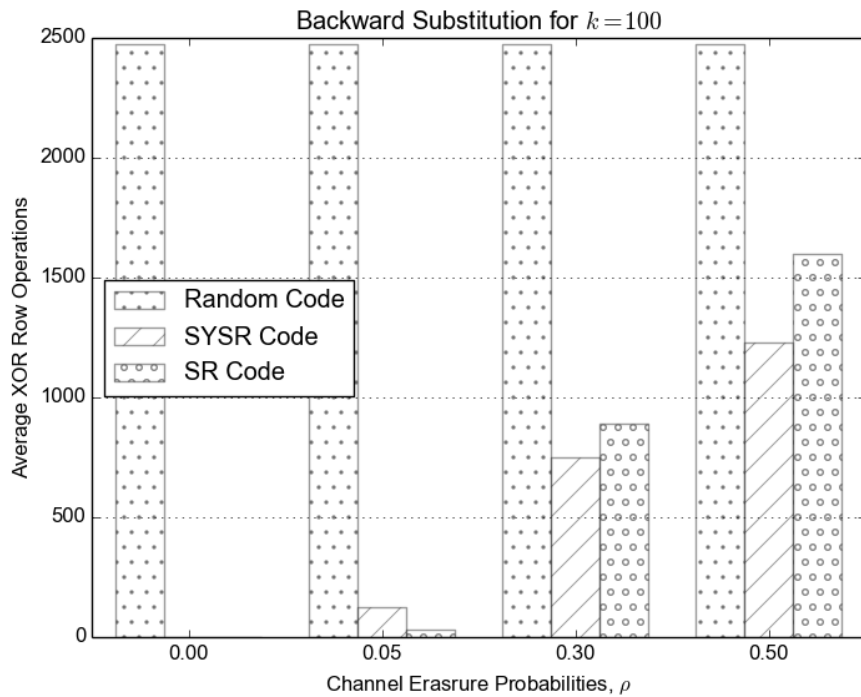
Generally, as  $\rho$  increases, more Part II encoded symbols are involved in the decoding process. Since the Part II encoded symbols are generated with a random matrix, the performance of SYSR code approaches that of Random code. Therefore, we omit the simulation results for  $\rho > 0.5$  in this chapter. SYSR code will achieve high PCD with  $k+10$  encoded symbols (with 1.6 overhead symbols) under very lossy channel conditions and the total number of required XOR row operations will be the same as Random code.

## 5.5 Summary

SYSR code is a systematic rateless erasure code that is built on top of random matrix. It achieves better PCD with fewer decoding steps than Random code in channels with low erasure probability.



(a)



(b)

Figure 5.6: The average XOR row operations for Random code, SYSR code and SR code to form (a) upper triangular matrix, and (b) backward substitution in channels of increasing erasure probabilities.

## CHAPTER 6

### STEPPING-RANDOM CODE

Chapters 3's Random code and Chapter 4's Micro-Random code are able to achieve high PCD with a fixed number of overhead symbols irrespective of message length  $k$ , but with the trade-off of high decoding complexity. Though Chapter 5's systematic Random (SYSR) code achieves better PCD and decoding complexity, it works efficiently only for point-to-point (i.e., one sender to one receiver) and point-to-multipoint (i.e., one sender to many receivers) transmissions. In this chapter, we propose a non-systematic pseudo-random code, namely Stepping-Random (SR) code that works in point-to-point, point-to-multipoint and multipoint-to-point (i.e., many senders to one receiver) transmissions.

#### **6.1 Non-Systematic Pseudo-Random Code**

In the following sections, we will explain the reason SYSR code performs inefficiently in multipoint-to-point transmission. Then, we propose a variant, namely Stepping-Random (SR) code that achieves similar PCD as SYSR code but with slightly higher decoding complexity.

##### **6.1.1 Issue of SYSR Code in Multipoint-to-Point Transmission**

In order to improve the network throughput, one may request the same message (if available) from multiple sources concurrently (multipoint-to-point transmission) as depicted in Figure 6.1. Both Random code and Micro-Random code are applicable in this transmission paradigm because all the



received encoded symbols form random matrices after all and the theorems in Sections 3.2 and 3.3 have assured their high PCD with a fixed number of overhead symbols.

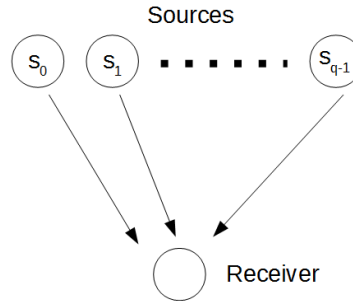


Figure 6.1: Multipoint-to-point transmission.

Generally, Chapter 5's systematic Random code imposes lower decoding complexity among all the codes proposed in this thesis. However, it does not work efficiently in multipoint-to-point transmission as each of its Part I encoded symbols represents exactly one message symbol. This statement can be explained using the classical balls and bins analysis (Luby, 2002) as follows.

Let a message of  $k$  bits be represented as  $k$  bins and the Part I encoded symbol as balls. The classical balls and bins analysis indicates that on average  $k \ln(k/\delta)$  balls (Part I encoded symbols) are needed for each of the  $k$  bins (message of  $k$  bits) to be covered by at least one ball with success probability  $1-\delta$ . For example, to have a success probability of  $1-\delta=0.99$ , a message of  $k=100$  bits requires  $100 \ln(100/0.01)=921$  Part I encoded symbols to achieve high PCD. It implies that the Part I encoded symbols of systematic Random code are less useful in multipoint-to-point transmission.

### 6.1.2 Stepping Code

Though the Part I encoded symbols of SYSR code is the main factor that improves the decoding complexity, it causes inefficiency in multipoint-to-point transmission. To address this issue, we suggest to generate the Part I encoded symbols with Stepping code.

Stepping code is a block code. Each of its generator row matrices has about the same weight as the Random code, i.e.,  $w \approx k/2$ . Instead of distributing the non-zero entries randomly, Stepping code organises them in the order that is similar to Gray code – each row differs from the previous one by only one bit.

The generator matrix of Stepping code is constructed as the followings. Let  $w = \lfloor k/2 \rfloor$ . Then, the first two rows,  $g_0^s$  and  $g_1^s$  have weights of  $w$  and  $w+1$  respectively and they can be represented as

$$g_0^s = [a_{0,0}(w) \ a_{0,1}(w) \ \dots \ a_{0,k-1}(w)], \quad (64)$$

and

$$g_1^s = [a_{1,0}(w+1) \ a_{1,1}(w+1) \ \dots \ a_{1,k-1}(w+1)]. \quad (65)$$

The notation  $a_{i,j}$  defines the value of each entry, where

$$a_{i,j}(w) = \begin{cases} 1 & \text{if } j < w \\ 0 & \text{otherwise} \end{cases}. \quad (66)$$

Then, the rest of the rows  $g_j$  are expressed as

$$g_j^s = \begin{cases} g_0^s \gg j/2 & \text{for } j=2,4,6,\dots \\ g_1^s \gg (j-1)/2 & \text{for } j=3,5,7,\dots \end{cases}, \quad (67)$$

where  $\gg$  is a right cyclic shift operator.

For example, the first and second row of the generator matrix for message of  $k=6$  symbols are  $g_0^s=[111000]$  and  $g_1^s=[111100]$ . Then, the third and fourth rows are obtained by applying the right shift operator to the first and second rows, i.e.,  $g_2^s=[011100]$  and  $g_3^s=[011110]$ . Using the same method, the first six generator row matrix is presented in Eq. (68).

$$\mathbf{G}_{\text{step}}^{k \times k} = \begin{bmatrix} g_0^s \\ g_1^s \\ g_2^s \\ g_3^s \\ g_4^s \\ g_5^s \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}. \quad (68)$$

The decoding method will be explicated together with SR code in Section 6.2.

### 6.1.3 Stepping-Random (SR) Code

We propose a pseudo-random code, namely Stepping-Random (SR) code that works in point-to-point, point-to-multipoint and multipoint-to-point transmissions. Given a message of  $k$  symbols, SR code generates two types of encoded symbols:

- First  $k$  encoded symbols, i.e.,  $x_0^s, x_1^s, \dots, x_{k-1}^s$  are generated with Stepping code and they are termed as Part I encoded symbols.

- The rest of the encoded symbols from  $k+1$  onwards, i.e.,  $x_k^r, x_{k+1}^r, \dots$  are generated with Random code and termed as Part II encoded symbols.

Generally, SR code's generator matrix can be expressed as

$$\mathbf{G}_{\text{SR}}^{(k+(\cdot)) \times k} = \begin{bmatrix} \mathbf{G}_{\text{step}}^{k \times k} \\ \mathbf{G}_{\text{rand}}^{(\cdot) \times k} \end{bmatrix}. \quad (69)$$

We use  $(\cdot)$  when the quantity is not known. An example of a generator matrix of  $k=6$  is presented as Eq. (70), where the first  $k$  rows are Stepping code's generator matrix and the rest are generated with Random code.

$$\mathbf{G}_{\text{SR}}^{(k+(\cdot)) \times 6} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ \hline 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}. \quad (70)$$

## 6.2 Message Reconstruction

The decoding algorithm for ideal channel will be discussed in Section 6.2.1 (decoding in ideal channel) and the full version (decoding in lossy channel) in Section 6.2.2.

### 6.2.1 Decoding in Ideal Channel

Recall that the Part I encoded symbols possess a Gray code like structure, where each row matrix is differed from the previous one by only one bit. This property enables SR code to reconstruct the original message with *sequential addition* algorithm as shown in the following.

Let  $x_i^s$  be the  $i$ -th Part I encoded symbol and its corresponding generator row matrix as  $g_i^s$ . Then, the sequential addition reconstructs a message symbol  $m_j$  by adding two encoded symbols in sequence. That is,

$$m_j = x_i^s + x_{i-1}^s, \quad (71)$$

where  $i=1,2,\dots,k-1$  and  $j$  is the index of non-zero entry in the addition of  $g_i^s$  and  $g_{i-1}^s$ . For example, adding the  $g_0^s = [1 \ 1 \ 1 \ 0 \ 0 \ 0]$  and  $g_1^s = [1 \ 1 \ 1 \ 1 \ 0 \ 0]$  in Eq. (70) will reconstruct the message symbol  $m_3$ .

The original message can be reconstructed from the first  $k$  encoded symbols (i.e., all the Part I encoded symbols). The sequential addition will reconstruct  $k-1$  message symbols and the last message symbol  $m_{w-1}$  (where  $w = \lfloor k/2 \rfloor$ ) with Gaussian elimination. For example, assuming all the Part I encoded symbols in Eq. (70) are received intact. Then, the sequential addition will reconstruct  $k-1$  message symbols (i.e., adding  $x_0$  and  $x_1$

reconstructs  $m_3$ , adding  $x_1$  and  $x_2$  reconstructs  $m_0$  and etc., except  $m_2$ ).

The last message symbol  $m_2$  can be reconstructed with Gaussian elimination.

Sequential addition reconstructs the  $k-1$  message symbols with  $k-1$  row operations, i.e., the computational complexity is  $O(k)$ . On the other hand, the last message symbol can be reconstructed with Gaussian elimination of computational complexity  $O(k^3)$ . The latter computational complexity can be ignored as it is used to construct only one symbol.

### 6.2.2 Decoding in Lossy Channel

In the previous section, we assume all the Part I encoded symbols are received intact and the original message is reconstructed with sequential addition. No Part II encoded symbols are needed. In this section, we assume a lossy channel and not all the Part I encoded symbols are received intact. Then, the receiver requires a total of  $k+10$  Part I-II encoded symbols in order to reconstruct the original message with high PCD.

The full decoding algorithm of SR code in lossy channel consists of the following steps:

- **Step 1:** Form the generator matrix from the received  $k+10$  Part I-II encoded symbols.
- **Step 2:** Perform the sequential addition to the generator row matrices that belongs to Part I encoded symbols.

- **Step 3:** Add the reconstructed message symbols in Step 2 to the rest of relevant encoded symbols. Removing the columns and rows that correspond to the reconstructed message symbols in Step 2.
- **Step 4:** Apply Gaussian elimination to reconstruct the remaining symbols of the original message.

We will explain the aforementioned algorithm with the example below.

$$\mathbf{G}_{\text{SR}}^{16 \times 6} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}. \quad (72)$$

Assuming a lossy channel transmission and we have received  $k+10$  Part I-II encoded symbols. The generator matrix is formed in Step 1 as shown in Eq. (72). Note that only two Part I encoded symbols (first two rows) are received in sequence. The third Part I encoded symbol is not in sequence with the second one. Reconstructing the fourth message symbol is done by adding the first and second encoded symbols, i.e.,  $m_3 = x_0 \oplus x_1$  in Step 2. The resulting generator matrix is presented in Eq. (73).

$$\mathbf{G}_{\text{SR}}^{(16) \times 6} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}. \quad (73)$$

In Step 3, we add the second encoded symbol to the rest of the encoded symbols, in which the fourth entries are non-zero. Then, the second

row and the fourth column of the generator matrix are removed as shown in Eq. (74). The resulting generator matrix will have smaller dimensions if we have more Part I encoded symbols that are in sequence.

$$\mathbf{G}_{\text{SR}}^{(15) \times 5} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \quad (74)$$

In Step 4, the first and the third Part I encoded symbols will combine with the rest of the encoded symbols to form a new generator matrix of dimensions  $15 \times 5$ . This new generator matrix appears to be a message of  $k=5$  symbols and we reconstruct the original message from the rest of ten overhead symbols with high PCD.

Another example, given a message of  $k=100$  symbols and we have received 50 Part I encoded symbols and 60 Part II encoded symbols. In Step 1, we form a generator matrix of dimensions  $110 \times 100$  that represents these 110 encoded symbols. Assume that only some of the Part I encoded symbols are in sequence. Consider the case that we manage to reconstruct  $(1-\beta)k=20$  message symbols in Step 2. Then, 20 rows and columns of the generator matrix will be eliminated and the remaining generator matrix has dimensions  $90 \times 80$  in Step 3. Generally, the resulting generator matrix represents the subset message of 80 symbols and we have  $80+10=90$  encoded symbols.



Therefore, in Step 4 we deploy Gaussian elimination to reconstruct the remaining message with high PCD.

### 6.2.3 Decoding Complexity

Assume a lossy channel and we have received  $k+10$  Part I-II encoded symbols. The sequential addition in Step 2 of the decoding process only works on the sequential Part I encoded symbols. Let  $\beta$  be the fraction of the Part I encoded symbols that are out of sequence for  $0 \leq \beta \leq 1$ . Then, sequential addition only works on  $(1-\beta)k$  Part I encoded symbols that are in sequential order – each of them contributing a single row operation and the decoding complexity becomes  $O((1-\beta)k)$ . Next, in Step 3 we shrink the matrix dimensions from  $(k+10) \times k$  to  $(\beta k+10) \times \beta k$ . The operation involves  $(1-\beta)k$  columns and each column involves  $(\beta k+10)$  row operations. Therefore the decoding complexity is  $O((\beta-\beta^2)k^2)$ .

Let  $K=\beta k$ , Gaussian elimination requires  $O(K^3)$  to solve a  $(K+10) \times K$  matrix. In other words, SR code requires computational complexity of  $O((\beta k)^3)$  in Step 4. Hence, the overall decoding complexity can be expressed as

$$O((1-\beta)k) + O((\beta-\beta^2)k^2) + O((\beta k)^3). \quad (75)$$

Note that  $\beta$  is a variable. If all the Part I encoded symbols are received intact ( $\beta=0$ ), the decoding complexity can be simplified to  $O(k)$ . On the other hand, if none of the Part I encoded symbols can be used in sequential addition, the decoding complexity will be  $O(k^3)$ .

#### 6.2.4 Decoding in Multipoint-to-Point Transmission Scenario

Unlike Chapter 5's systematic Random code, the receiver of SR code has the ability to reconstruct the original message by receiving the encoded symbols from multiple sources. We represent the message as  $\mathbf{M}^{k \times l}$  and the Stepping code generator matrix as  $\mathbf{G}_{\text{step}}^{k \times k}$ . Instead of generating the Part I encoded symbols with  $\mathbf{G}_{\text{step}}^{k \times k}$  directly, we randomise the columns of the generator matrix and denote the resulting generator matrix as  $\tilde{\mathbf{G}}_{\text{step}}^{k \times k}$ . Then, the Part I encoded symbols can be generated using  $\mathbf{X}_{\text{step}}^{k \times l} = \tilde{\mathbf{G}}_{\text{step}}^{k \times k} \mathbf{M}^{k \times l}$ . Note that such action will not affect the rankness of the matrix. Additionally, we assume that the receiver can learn the random seed from the identities of the encoded symbols.

Say the receiver receives  $q$  Part I encoded symbols from  $q$  sources, where  $q \geq k$ . Assume that these received Part I encoded symbols appear to be random from each other. Then, the receiver just needs to receive another extra 10 Part II encoded symbols from any source in order to achieve high PCD. Note that the sequential addition is not applicable in this case as all the Part I encoded symbols are generated from different sources. Hence, the original message has to be reconstructed with Gaussian elimination. The simulation results of SR code in multipoint-to-point transmission paradigm are presented in Figures 6.12 and 6.13.

### 6.3 Numerical Result

The simulation consists of three components – sender, erasure channel and receiver. The sender generates the generator row matrices in the encoding process, the erasure channel decides whether a generator row matrix should be dropped according to the channel erasure probability  $\rho$  and the receiver attempts to inverse the generator matrix. Each simulation has been repeated for 1000 times and the average results are presented in graphs.

Figures 6.2 to 6.7 illustrate the performance of SR code, Random code and Windowed code (see Section 3.4) in channel of erasure probabilities  $\rho=0.00$  to  $0.05$ . In general, SR code has a higher PCD as compared to Random code and Windowed code using the same amount of overhead symbols. Particularly, Figure 6.2 demonstrates that SR code is the only coding scheme that is able to achieve complete decoding with zero overhead symbols in an ideal channel (i.e.,  $\rho=0$ ). Nevertheless, the disparity between SR code and Random code becomes negligible when the channel erasure probability increases.

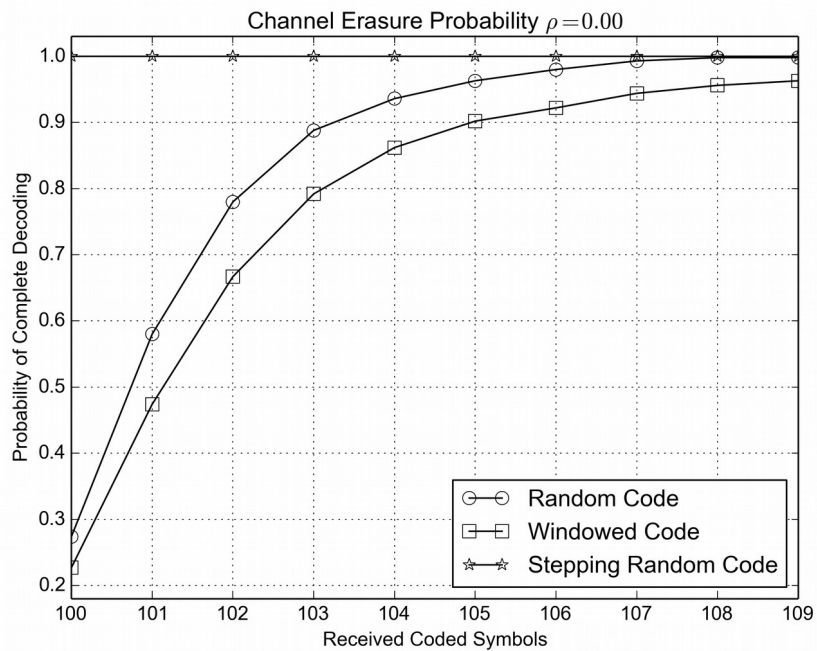


Figure 6.2: The performance of Random code, Windowed code and SR code in the channels of channel erasure probability  $\rho=0.00$  for message of  $k=100$  symbols.

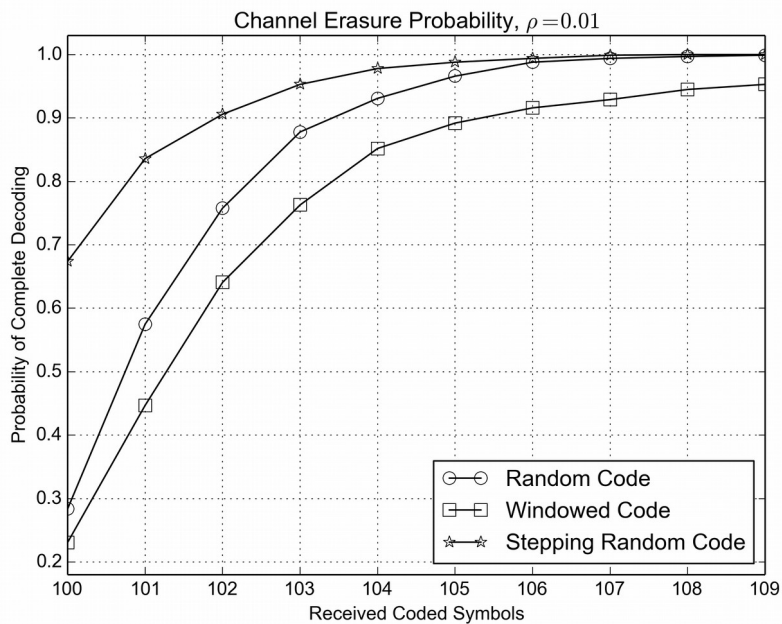


Figure 6.3: The performance of Random code, Windowed code and SR code in the channels of channel erasure probability  $\rho=0.01$  for message of  $k=100$  symbols.

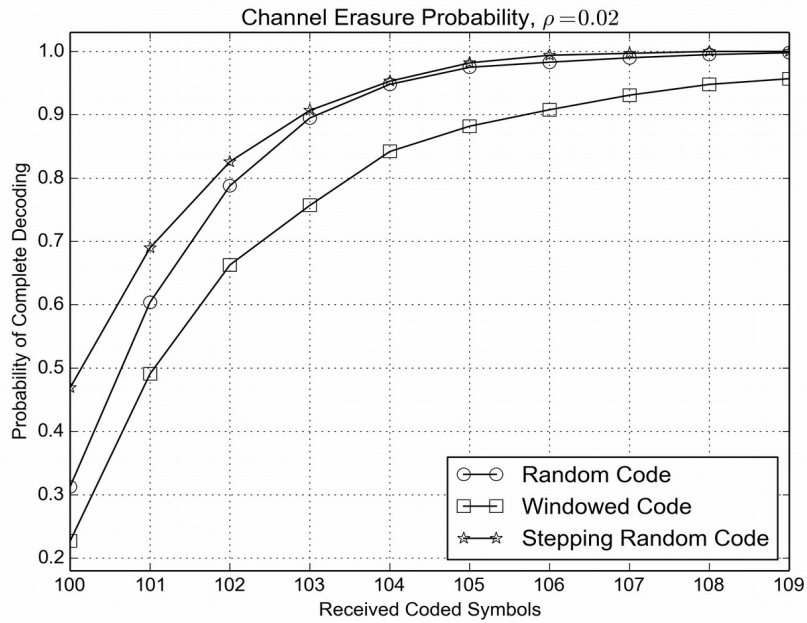


Figure 6.4: The performance of Random code, Windowed code and SR code in the channels of channel erasure probability  $\rho=0.02$  for message of  $k=100$  symbols.

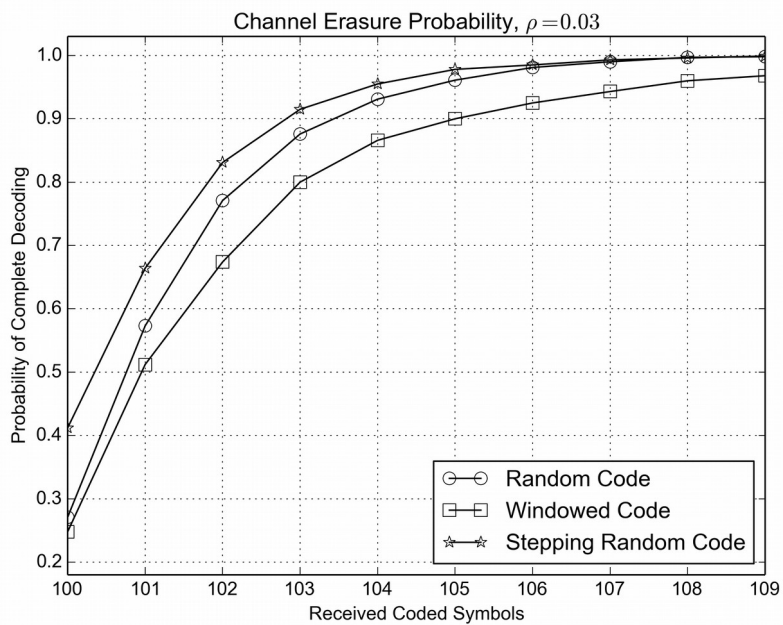


Figure 6.5: The performance of Random code, Windowed code and SR code in the channels of channel erasure probability  $\rho=0.03$  for message of  $k=100$  symbols.

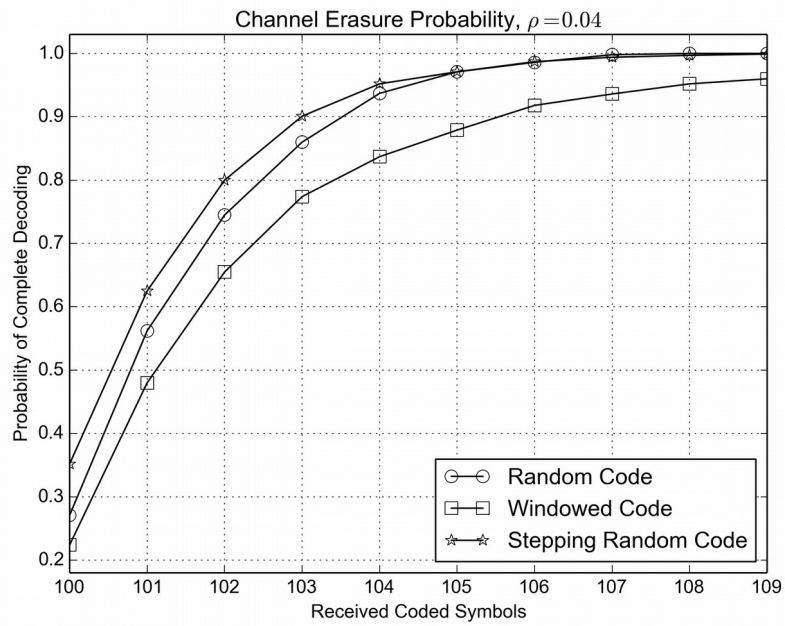


Figure 6.6: The performance of Random code, Windowed code and SR code in the channels of channel erasure probability  $\rho=0.04$  for message of  $k=100$  symbols.

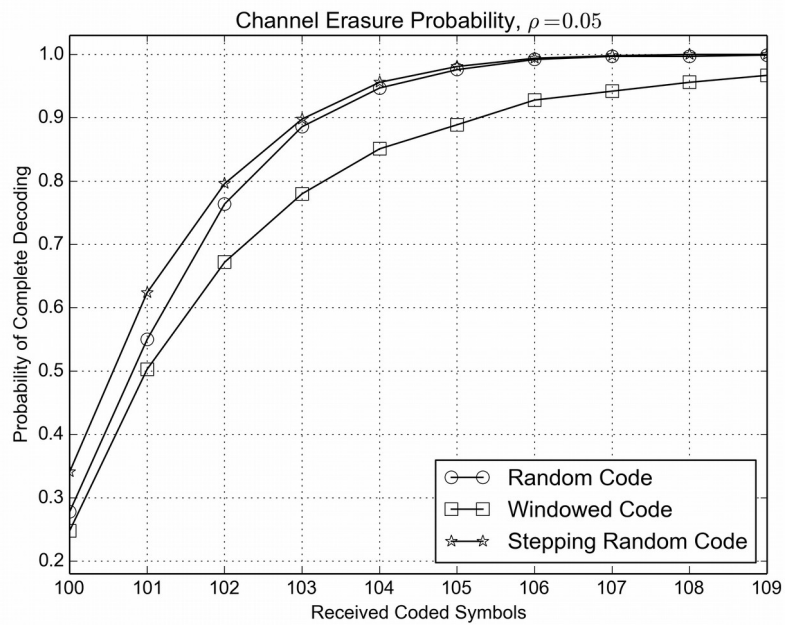


Figure 6.7: The performance of Random code, Windowed code and SR code in the channels of channel erasure probability  $\rho=0.05$  for message of  $k=100$  symbols.

We vary the channel erasure probabilities from  $\rho=0.1$  to 0.9 (very lossy channel). The results are similar among all three and part of the results ( $\rho=0.5$  and 0.7) are presented in Figures 6.8 and 6.9. We have tested the performance of SR code for messages of  $k=100$ , 300, 500 and 1000 symbols in channels of various channel erasure probabilities. The results are identical and one of the results  $\rho=0.5$  is shown in Figure 6.10.

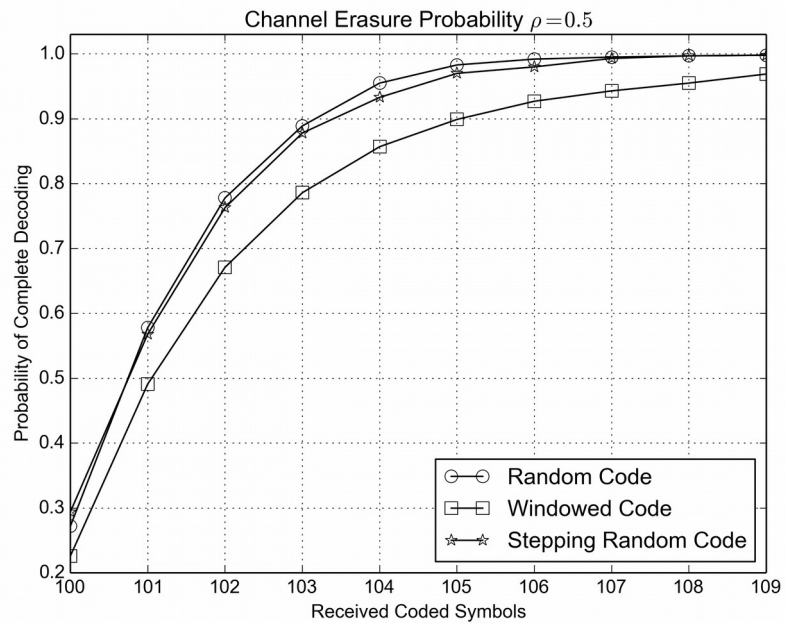


Figure 6.8: The performance of Random code, Windowed code and SR code in the channels of channel erasure probability  $\rho=0.5$  for message of  $k=100$  symbols.

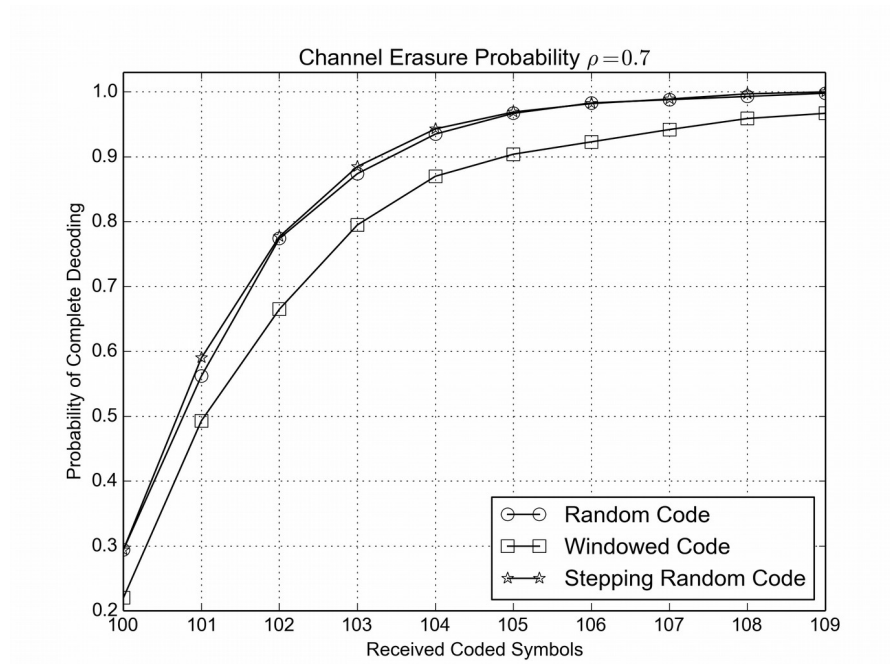


Figure 6.9: The performance of Random code, Windowed code and SR code in the channels of channel erasure probability  $\rho=0.7$  for message of  $k=100$  symbols.

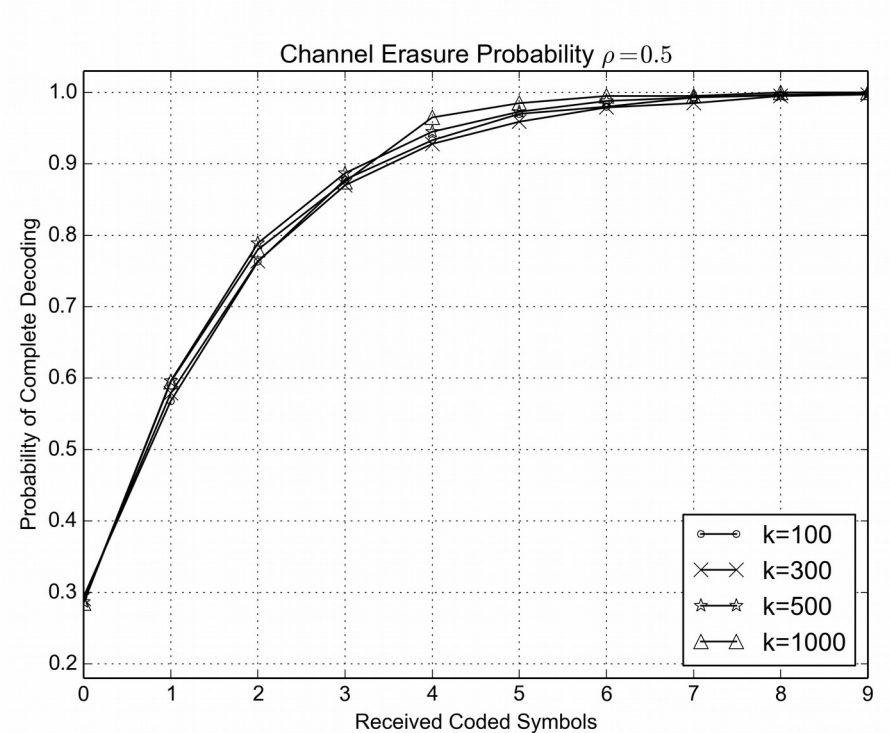


Figure 6.10: The performance of SR code in the channels of channel erasure probability  $\rho=0.5$  for messages of length 100, 300, 500 and 1000 symbols.



Figure 6.11 demonstrates the decoding complexity of SR code for various channel erasure probabilities. In the simulation, the sender continuously generates the encoded symbols for a message of  $k=100$  symbols and the receiver attempts to reconstruct the original messages with sequential addition and Gaussian elimination.

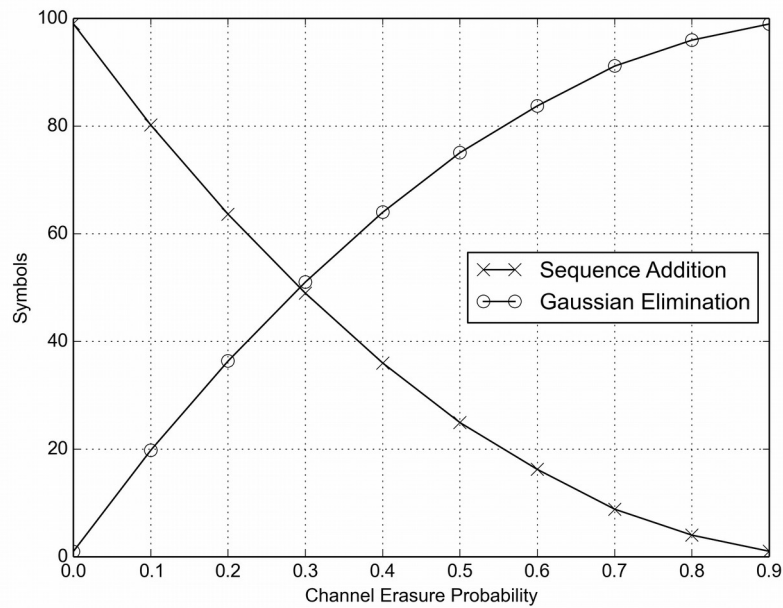


Figure 6.11: Total message symbols that are reconstructed with sequential addition and Gaussian elimination in channels of various erasure probabilities.

In the channel of zero or near-zero packet loss probability, the majority of the message symbols can be reconstructed with sequential addition. For example, in  $\rho=0.1$  about 80 out of 100 symbols were reconstructed with sequential addition and only about 20 symbols (i.e.,  $\beta=0.2$ ) needed to be reconstructed using Gaussian elimination. However, as the channel erasure probability increases, the major portion of the original message will be reconstructed with Gaussian elimination. On the other hand, in a very lossy channel ( $p=0.9$ ) SR code performs as good as Random code and it uses Gaussian elimination to reconstruct the messages most of the time.

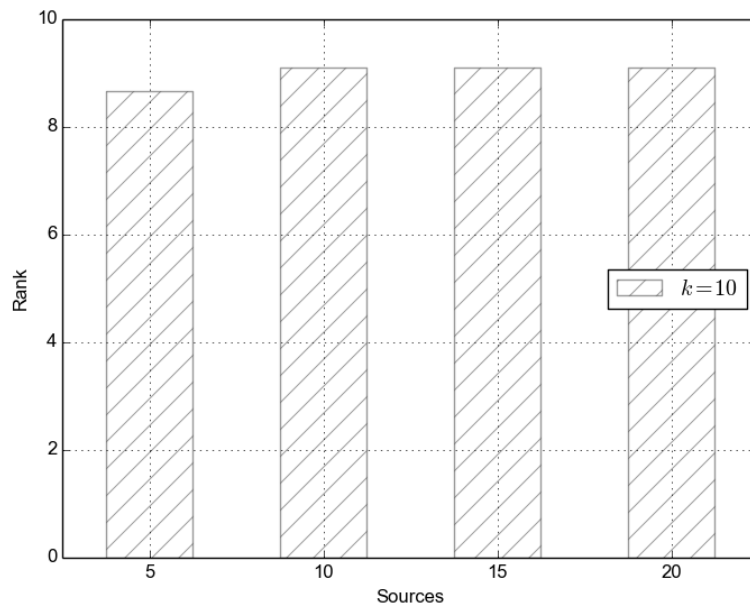
Figures 6.12 and 6.13 illustrate the performance of SR code in multipoint-to-point transmission paradigm with erasure probability  $\rho=0.1$ . Receiving Part I encoded symbols from  $q=5, 10, 15$  and  $20$  sources randomly will form a nearly full rank matrix as depicted in Figure 6.12(a) for message of  $k=10$  symbols and (b) message of  $k=50$  symbols. Additionally, doubling the total number of sources (i.e.,  $q=2k$ ) will not improve the rankness further. After all, the Part I encoded symbols are generated with Stepping code instead of Random code and the theorems in Sections 3.2 and 3.3 do not apply here (i.e., it is not able to achieve high PCD with  $k+10$  Part I encoded symbols).

Additionally, Figure 6.13 (a) and (b) illustrate that SR code is able to achieve high PCD by combining  $k$  Part I encoded symbols and 10 extra Part II encoded symbols (i.e., total of  $k+10$  Part I-II encoded symbols) for messages of  $k=10$  and  $k=50$  symbols. Note that, SR code deploys Gaussian elimination to reconstruct the original message in multipoint-to-point transmission paradigm – it has the same decoding complexity as Random code in this case.

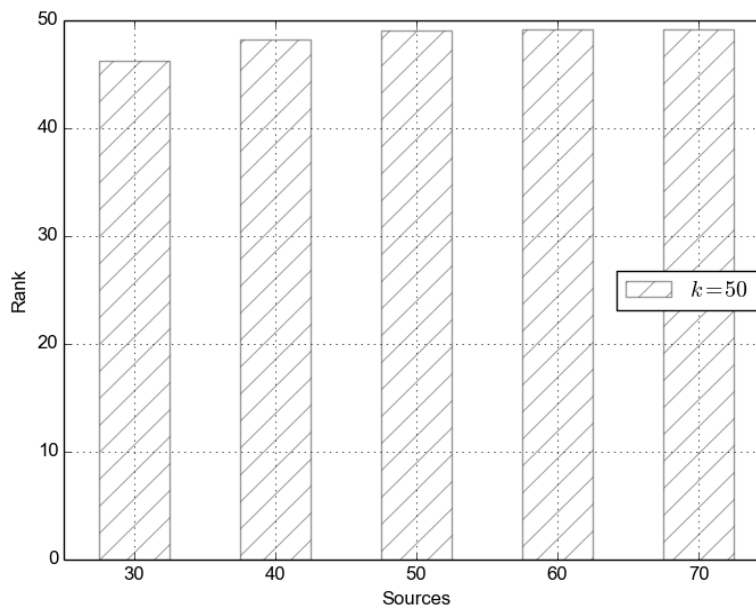
#### 6.4 Summary

In this chapter, we have proposed a non-systematic rateless erasure code, namely Stepping-Random (SR) code that demonstrates better PCD and decoding complexity as compared with Random code and Windowed code in channels of zero or near zero erasure probability. Additionally, it has identical

performance with Random code in a very lossy channel. Unlike the systematic Random code, which was proposed in the previous chapter, SR code works in multipoint-to-point transmission paradigm with the trade-off of non-negligible decoding complexity even if all the Part I encoded symbols are received intact.

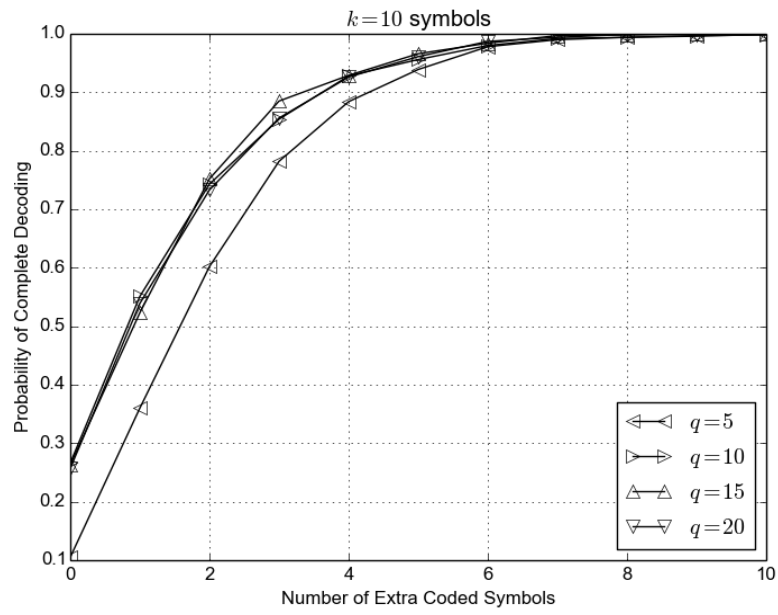


(a)

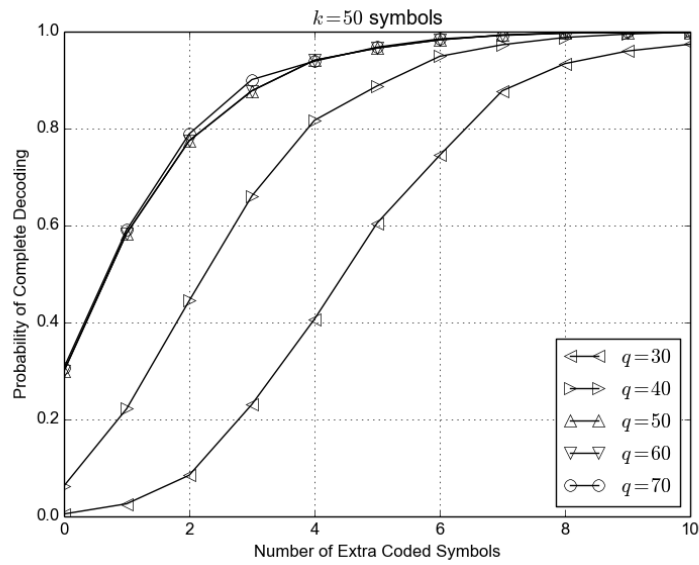


(b)

Figure 6.12: The achievable rankness by selecting  $k$  Part I encoded symbols randomly from various number of sources for messages of (a)  $k=10$  (b)  $k=50$  symbols.



(a)



(b)

Figure 6.13: The PCD of randomly selected  $k+10$  Part I-II encoded symbols from various number of sources.

## CHAPTER 7

### RATELESS ERASURE CODES IN GF(2<sup>8</sup>)

This chapter extends our proposed GF(2) rateless erasure code to higher order of finite field for gains in PCD. Though these GF(2<sup>8</sup>) rateless erasure codes are not the focus of the thesis, they are presented here in order to demonstrate the potential in attaining better PCD performance than RaptorQ code – a Raptor code variant that is built on the hybrid of GF(2) and GF(2<sup>8</sup>).

#### 7.1 Random Code in GF(2<sup>q</sup>)

Random code is the basic design of our proposed rateless erasure codes. In this section, we will first generalise Random code to GF(2<sup>q</sup>) and the resulting PCD equation will be used in the rest of the proposed GF(2<sup>q</sup>) rateless erasure codes. We start the explanation by first restating Lemma 3.3.1 in GF(2<sup>q</sup>) as the following.

**Lemma 7.1.1.** Given that a GF(2<sup>q</sup>) random matrix  $\mathbf{G}^{(q),(n-1)\times k}$  of dimensions  $(n-1)\times k$  has rank  $(n-1)$ , where  $0 < n \leq k$ . Then, the probability to attain rank  $n$  with an extra row is

$$p^{(q)}(n, k) = \prod_{i=1}^{n-1} (1 - 2^{q(i-k)}). \quad (76)$$

**Proof.** In GF(2<sup>q</sup>), a random vector  $\mathbf{g}^{(q),1\times k}$  of  $k$  elements has  $2^{kq} - 1$  possible combination, excluding the vector of all zeros. In order to have rank  $n$  in  $\mathbf{G}^{(q),n\times k}$ ,  $\mathbf{g}^{(q),1\times k}$  must be independent of all the rows in the previous

$\mathbf{G}^{(q),(n-1) \times k}$ . Therefore,  $g^{(q)1 \times k}$  is limited to  $\tau^{(q)}(n-1, k)$  possible combinations, where

$$\begin{aligned}\tau^{(q)}(n-1, k) &= 2^{kq} - 1 - \sum_{i=1}^{n-1} \binom{n-1}{i} (2^q - 1)^i \\ &= 2^{kq} - 1 - [(2^q)^{(n-1)} - 1] \\ &= 2^{kq} - 2^{q(n-1)},\end{aligned}\tag{77}$$

and  $\binom{\cdot}{\cdot}$  is the binomial coefficient. Then, the probability for  $\mathbf{G}^{(q),(n-1) \times k}$  to attain rank  $n$  with  $g^{(q),1 \times k}$  is

$$\begin{aligned}p^{(q)}(n, k) &= p^{(q)}(n-1, k) \times \frac{\tau^{(q)}(n-1, k)}{2^{kq}} \\ &= p^{(q)}(n-2, k) \times \frac{\tau^{(q)}(n-2, k)}{2^{kq}} \times \frac{\tau^{(q)}(n-1, k)}{2^{kq}} \\ &= \frac{\tau^{(q)}(1, k)}{2^{kq}} \times \frac{\tau^{(q)}(2, k)}{2^{kq}} \times \dots \times \frac{\tau^{(q)}(n-1, k)}{2^{kq}} \\ &= \prod_{i=1}^{n-1} \frac{\tau^{(q)}(i, k)}{2^{kq}} \\ &= \prod_{i=1}^{n-1} \frac{(2^{kq} - 2^{qi})}{2^{kq}} \\ &= \prod_{i=1}^{n-1} (1 - 2^{q(i-k)}),\end{aligned}\tag{78}$$

■

Generally, Lemma 3.3.1 uses  $i-k$  in the exponents, i.e.,  $2^{i-k}$  while the exponent in Eq. (78) is  $q(i-k)$  – they share the same exponential form but in different constant,  $q$ . Next, Lemmas 3.3.2 to 3.3.4 are restated in the followings without proof.

First, the probabilities of the last few ranks for random matrices of different dimensions differ insignificantly as stated below.

**Lemma 7.1.2.** Let  $m \in \mathbb{Z}^+$  and  $0 \leq m < k$ . The probabilities to have rank  $k-m$  in  $\text{GF}(2^q)$  random matrices of dimensions  $(k-m) \times k$  for different  $k$  are similar, i.e.,

$$|p^{(q)}(k-m+1, k+1) - p^{(q)}(k-m, k)| < \delta, \quad (79)$$

and  $\delta \rightarrow 0$  exponentially fast with finite and bounded increment  $k$ .

Secondly, the transposition of the random matrices will not change the rankness as stated below.

**Lemma 7.1.3.** The probabilities to have rank  $k-m$  in  $\mathbf{G}^{(q), (k-m) \times k}$  and  $\mathbf{G}^{(q), k \times (k-m)}$  are identical.

In short, we have determined the probability to attain rank  $k-m$  in  $\text{GF}(2^q)$  random matrix  $\mathbf{G}^{(q), (k-m) \times k}$  in Lemma 7.1.1. Then, Lemma 7.1.2 shows that such probability varies insignificantly for any two  $\text{GF}(2^q)$  random matrices of different dimensions with increasing  $k$ . Since the rankness will not change with transposition (Lemma 7.1.3), we conclude the following theorem:

**Theorem 7.1.4.** The probabilities to have rank  $k$  in  $\text{GF}(2^q)$  random matrix of dimensions  $(k+m) \times k$  for  $m \geq 0$  is

$$\begin{aligned} \Pr(\text{rank}(\mathbf{G}^{(k+m) \times k}) = k) &= p^{(q)}(k-m, k) \\ &= \prod_{i=1}^{k-m-1} (1 - 2^{q(i-k)}) \\ &= Q_{\text{RC}}^{(q)}(m), \end{aligned} \quad (80)$$



Let the failure probability be defined as  $P_{\text{fail}} = 1 - Q_{\text{RC}}^{(q)}(m)$ . The corresponding numerical results for GF(2<sup>q</sup>) random matrices of dimensions 100×100 with extra rows are presented in Figure 7.1. As to facilitate the representation of long numerical values, we state the failure probabilities in the exponents of 10 throughout the rest of this chapter.

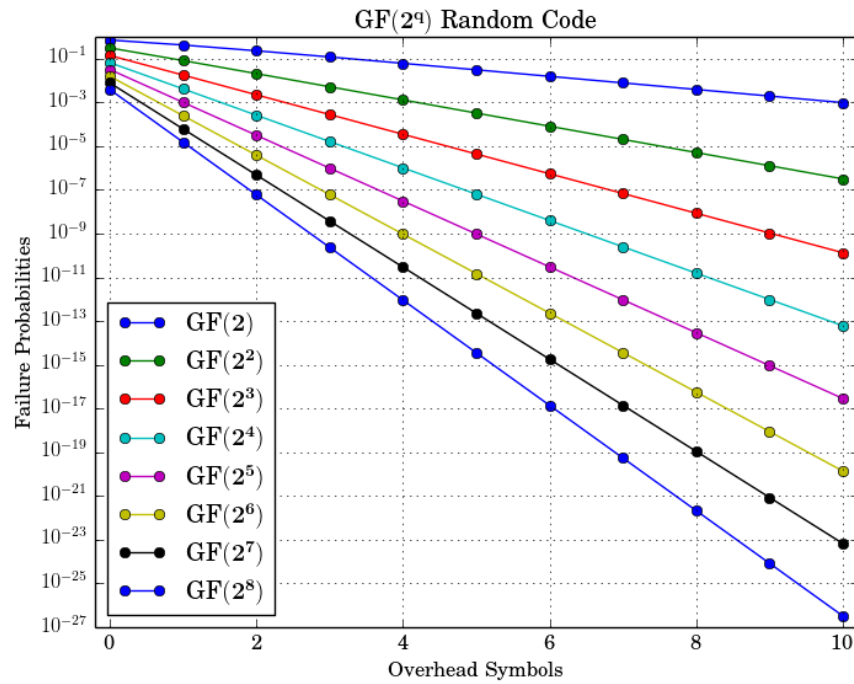


Figure 7.1: The failure probabilities of GF(2<sup>q</sup>) random matrices at various additional rows.

Overall, the probabilities improve significantly with higher  $q$ . For example, the failure probabilities at zero and ten overhead symbols for GF(2) are  $10^{-0.1480}$  and  $10^{-3.010}$ . However, in GF(2<sup>8</sup>) the failures probabilities improve to  $10^{-2.407}$  and  $10^{-26.49}$  respectively.

## 7.2 Micro-Random Code in GF(2<sup>q</sup>)

Generally, Micro-Random code attains higher PCD than Random code by using the micro symbols in the encoding and decoding processes.

Revising Eq. (44), the PCD of GF(2<sup>q</sup>) Micro-Random code is expressed as

$$\begin{aligned} Q_{\text{MRC}}^{(q)}(m, \alpha) &= Q_{\text{RC}}^{(q)}(\alpha m) \\ &= \prod_{i=1}^{k-\alpha m-1} (1 - (2^q)^{i-k}). \end{aligned} \quad (81)$$

and the failure probability is defined as  $p_{\text{fail}} = 1 - Q_{\text{MRC}}^{(q)}(m, \alpha)$ .

The failure probabilities of GF(2<sup>q</sup>) Micro-Random code with increasing overhead symbols using  $\alpha=10$  are presented in Table 7.1 and the corresponding graph in Figure 7.2. Additionally, the failure probabilities of GF(2<sup>8</sup>) Micro-Random code at various  $\alpha$  are presented in Table 7.2 and graphed in Figure 7.3.

Table 7.1: The failure probabilities of Micro-Random code at various GF(2<sup>q</sup>) and  $m$  overhead symbols using  $\alpha=10$ .

	GF(2)	GF(2 <sup>2</sup> )	GF(2 <sup>3</sup> )	GF(2 <sup>4</sup> )	GF(2 <sup>5</sup> )	GF(2 <sup>6</sup> )	GF(2 <sup>7</sup> )	GF(2 <sup>8</sup> )
<b>m=0</b>	10 <sup>-0.1480</sup>	10 <sup>-0.5066</sup>	10 <sup>-0.8520</sup>	10 <sup>-1.178</sup>	10 <sup>-1.492</sup>	10 <sup>-1.799</sup>	10 <sup>-2.104</sup>	10 <sup>-2.407</sup>
<b>m=1</b>	10 <sup>-3.010</sup>	10 <sup>-6.498</sup>	10 <sup>-9.876</sup>	10 <sup>-13.22</sup>	10 <sup>-16.54</sup>	10 <sup>-19.86</sup>	10 <sup>-23.18</sup>	10 <sup>-26.49</sup>
<b>m=2</b>	10 <sup>-6.021</sup>	10 <sup>-12.52</sup>	10 <sup>-18.91</sup>	10 <sup>-25.26</sup>	10 <sup>-31.59</sup>	10 <sup>-37.92</sup>	10 <sup>-44.25</sup>	10 <sup>-50.57</sup>
<b>m=3</b>	10 <sup>-9.031</sup>	10 <sup>-18.54</sup>	10 <sup>-27.94</sup>	10 <sup>-37.30</sup>	10 <sup>-46.65</sup>	10 <sup>-55.98</sup>	10 <sup>-65.32</sup>	10 <sup>-74.65</sup>
<b>m=4</b>	10 <sup>-12.04</sup>	10 <sup>-24.56</sup>	10 <sup>-36.97</sup>	10 <sup>-49.34</sup>	10 <sup>-61.70</sup>	10 <sup>-74.05</sup>	10 <sup>-86.39</sup>	10 <sup>-98.74</sup>
<b>m=5</b>	10 <sup>-15.05</sup>	10 <sup>-30.58</sup>	10 <sup>-46.00</sup>	10 <sup>-61.38</sup>	10 <sup>-76.75</sup>	10 <sup>-92.11</sup>	10 <sup>-107.5</sup>	10 <sup>-122.8</sup>
<b>m=6</b>	10 <sup>-18.06</sup>	10 <sup>-36.60</sup>	10 <sup>-55.03</sup>	10 <sup>-73.42</sup>	10 <sup>-91.80</sup>	10 <sup>-110.2</sup>	10 <sup>-128.5</sup>	10 <sup>-146.9</sup>
<b>m=7</b>	10 <sup>-21.07</sup>	10 <sup>-42.62</sup>	10 <sup>-64.06</sup>	10 <sup>-85.46</sup>	10 <sup>-106.9</sup>	10 <sup>-128.2</sup>	10 <sup>-149.6</sup>	10 <sup>-171.0</sup>
<b>m=8</b>	10 <sup>-24.08</sup>	10 <sup>-48.64</sup>	10 <sup>-73.09</sup>	10 <sup>-97.51</sup>	10 <sup>-121.9</sup>	10 <sup>-146.3</sup>	10 <sup>-170.7</sup>	10 <sup>-195.1</sup>
<b>m=9</b>	10 <sup>-27.09</sup>	10 <sup>-54.66</sup>	10 <sup>-82.12</sup>	10 <sup>-109.5</sup>	10 <sup>-137.0</sup>	10 <sup>-164.4</sup>	10 <sup>-191.8</sup>	10 <sup>-219.1</sup>
<b>m=10</b>	10 <sup>-30.10</sup>	10 <sup>-60.68</sup>	10 <sup>-91.15</sup>	10 <sup>-121.6</sup>	10 <sup>-152.0</sup>	10 <sup>-182.4</sup>	10 <sup>-212.8</sup>	10 <sup>-243.2</sup>

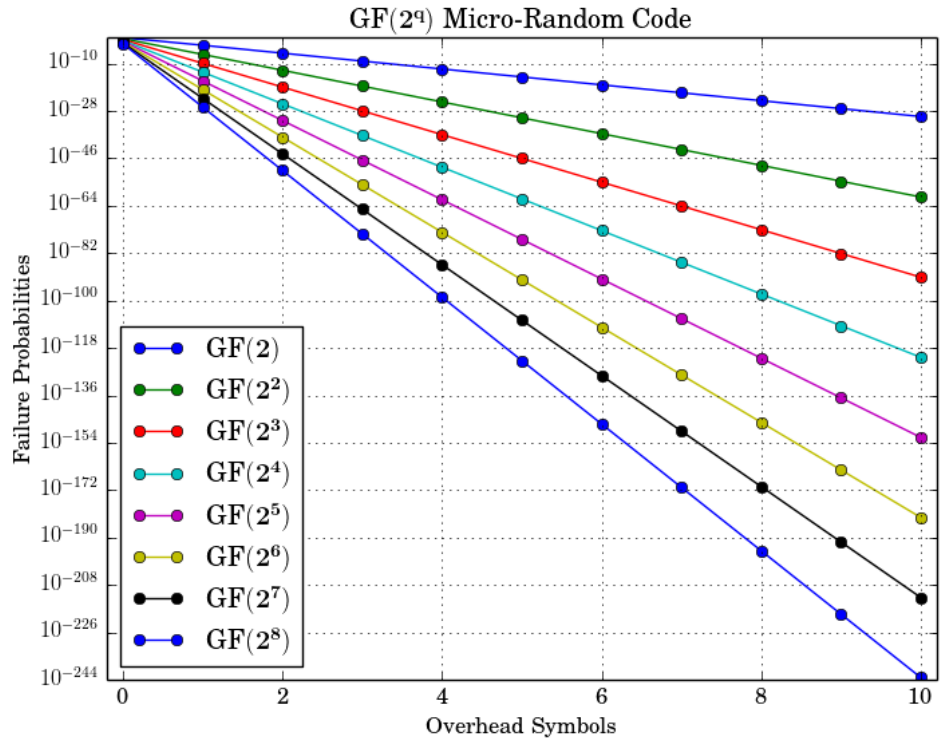


Figure 7.2: The failure probabilities of GF( $2^\alpha$ ) Micro-Random code at increasing number of overhead symbols.

Table 7.2: The failure probabilities of GF( $2^8$ ) Micro-Random code at various  $\alpha$  and overhead symbols,  $m$ .

	$\alpha=1$	$\alpha=2$	$\alpha=4$	$\alpha=6$	$\alpha=8$	$\alpha=10$
<b>m=0</b>	$10^{-2.407}$	$10^{-2.407}$	$10^{-2.407}$	$10^{-2.407}$	$10^{-2.407}$	$10^{-2.407}$
<b>m=1</b>	$10^{-4.815}$	$10^{-7.223}$	$10^{-12.04}$	$10^{-16.86}$	$10^{-21.67}$	$10^{-26.49}$
<b>m=2</b>	$10^{-7.223}$	$10^{-12.04}$	$10^{-21.67}$	$10^{-31.31}$	$10^{-40.94}$	$10^{-50.57}$
<b>m=3</b>	$10^{-9.631}$	$10^{-16.86}$	$10^{-31.31}$	$10^{-45.75}$	$10^{-60.20}$	$10^{-74.65}$
<b>m=4</b>	$10^{-12.04}$	$10^{-21.67}$	$10^{-40.94}$	$10^{-60.20}$	$10^{-79.47}$	$10^{-98.74}$
<b>m=5</b>	$10^{-14.45}$	$10^{-26.49}$	$10^{-50.57}$	$10^{-74.65}$	$10^{-98.74}$	$10^{-122.8}$
<b>m=6</b>	$10^{-16.86}$	$10^{-31.31}$	$10^{-60.20}$	$10^{-89.10}$	$10^{-118.0}$	$10^{-146.9}$
<b>m=7</b>	$10^{-19.26}$	$10^{-36.12}$	$10^{-69.84}$	$10^{-103.6}$	$10^{-137.3}$	$10^{-171.0}$
<b>m=8</b>	$10^{-21.67}$	$10^{-40.94}$	$10^{-79.47}$	$10^{-118.0}$	$10^{-156.5}$	$10^{-195.1}$
<b>m=9</b>	$10^{-24.08}$	$10^{-45.75}$	$10^{-89.10}$	$10^{-132.5}$	$10^{-175.8}$	$10^{-219.1}$
<b>m=10</b>	$10^{-26.49}$	$10^{-50.57}$	$10^{-98.74}$	$10^{-146.9}$	$10^{-195.1}$	$10^{-243.2}$

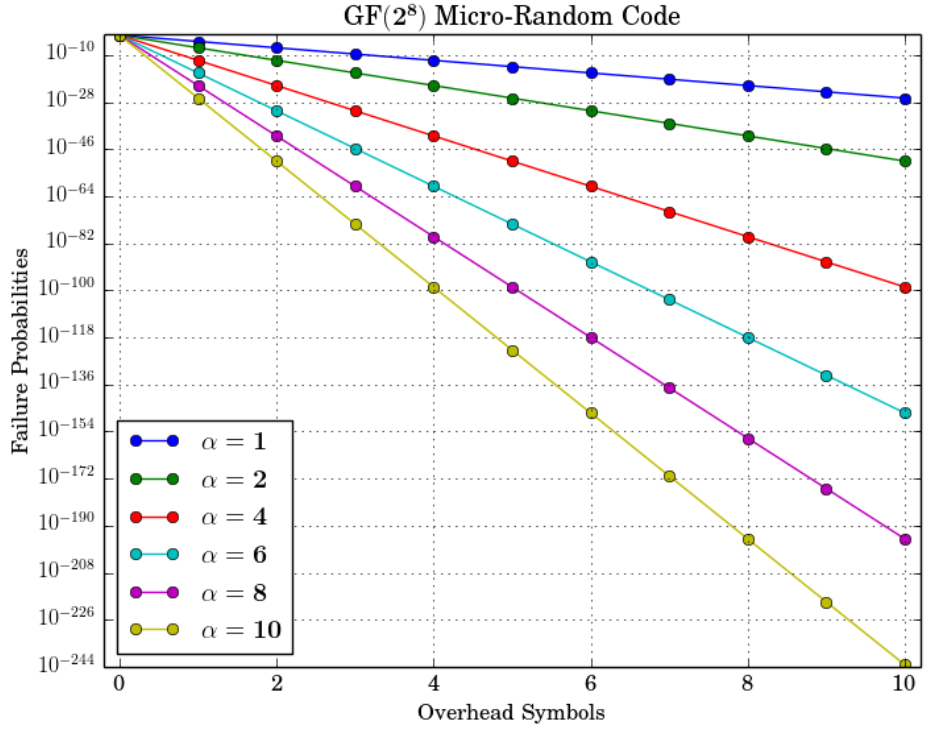


Figure 7.3: The failure probabilities of GF(2<sup>8</sup>) micro Random code at various  $\alpha$  and overhead symbols,  $m$ .

To find the PMF of each additional overhead symbols, we revise Eq.

(46) and (47) as

$$P_{\text{MRC}}^{(q)} = Q_{\text{MRC}}^{(q)}(m, \alpha) - Q_{\text{MRC}}^{(q)}(m-1, \alpha), \quad (82)$$

and

$$P_{\text{MRC}}^{(q)}(0, \alpha) = Q_{\text{MRC}}^{(q)}(0, \alpha). \quad (83)$$

Utilising them, the numerical values are presented in Table 7.3.

Table 7.3: The PMF and the mean of GF(2<sup>8</sup>) micro Random code at various  $\alpha$ .

	$\alpha=1$	$\alpha=2$	$\alpha=4$	$\alpha=6$	$\alpha=8$	$\alpha=10$
<b>m=0</b>	$10^{-0.001706}$	$10^{-0.001706}$	$10^{-0.001706}$	$10^{-0.001706}$	$10^{-0.001706}$	$10^{-0.001706}$
<b>m=1</b>	$10^{-2.408}$	$10^{-2.407}$	$10^{-2.407}$	$10^{-2.407}$	$10^{-2.407}$	$10^{-2.407}$
<b>m=2</b>	$10^{-4.816}$	$10^{-7.223}$	$10^{-12.04}$	$10^{-16.86}$	$10^{-21.67}$	$10^{-26.49}$
<b>m=3</b>	$10^{-7.225}$	$10^{-12.04}$	$10^{-21.67}$	$10^{-31.31}$	$10^{-40.94}$	$10^{-50.57}$
<b>m=4</b>	$10^{-9.633}$	$10^{-16.86}$	$10^{-31.31}$	$10^{-45.75}$	$10^{-60.20}$	$10^{-74.65}$
<b>m=5</b>	$10^{-12.04}$	$10^{-21.67}$	$10^{-40.94}$	$10^{-60.20}$	$10^{-79.47}$	$10^{-98.74}$
<b>m=6</b>	$10^{-14.45}$	$10^{-26.49}$	$10^{-50.57}$	$10^{-74.65}$	$10^{-98.74}$	$10^{-122.8}$
<b>m=7</b>	$10^{-16.86}$	$10^{-31.31}$	$10^{-60.20}$	$10^{-89.10}$	$10^{-118.0}$	$10^{-146.9}$
<b>m=8</b>	$10^{-19.27}$	$10^{-36.12}$	$10^{-69.84}$	$10^{-103.6}$	$10^{-137.3}$	$10^{-171.0}$
<b>m=9</b>	$10^{-21.67}$	$10^{-40.94}$	$10^{-79.47}$	$10^{-118.0}$	$10^{-156.5}$	$10^{-195.1}$
<b>m=10</b>	$10^{-24.08}$	$10^{-45.75}$	$10^{-89.10}$	$10^{-132.5}$	$10^{-175.8}$	$10^{-219.1}$
<b>Mean</b>	$10^{-2.41}$	$10^{-2.407}$	$10^{-2.407}$	$10^{-2.407}$	$10^{-2.407}$	$10^{-2.407}$

Generally, a lower failure probability is achievable with higher GF(2<sup>q</sup>),  $\alpha$  and  $m$ . The failure probability at zero overhead symbol improves with increasing order of GF(2<sup>q</sup>) and is invariant to  $\alpha$ . Additionally, GF(2<sup>8</sup>) Micro-Random code achieves failure probabilities of  $10^{-26.49}$  at one overhead symbol,  $10^{-50.57}$  at two overhead symbols and  $10^{-243.2}$  at ten overhead symbols.

On that account, we improve the performance of SYSR code and SR code in the following sections by replacing their Part II's GF(2) Random code with GF(2<sup>8</sup>) Micro-Random code.

### 7.3 Systematic Micro-Random Code and Stepping-Micro-Random Code in GF(2<sup>8</sup>)

We improve SYSR code and SR code by replacing their Part II's GF(2) Random code with GF(2<sup>8</sup>) Micro-Random code using segmentation factor  $\alpha=10$ . The resulting GF(2<sup>8</sup>) codes are named as *systematic Micro-Random (SYSMR) code* and *Stepping-Micro-Random (SMR) code*,

respectively. These codes will be compared with RaptorQ code, the variant of Raptor code that is built on the hybrid of GF(2) and GF(2<sup>8</sup>). In the following, we will study the failure probabilities of the proposed codes using various overhead symbols and channel erasure probabilities analytically.

By replacing the GF(2) Random code with GF(2<sup>8</sup>) Micro-Random code (using  $\alpha=10$ ), the PCD of SYSR, i.e., Eq. (58) is revised to

$$Q_{\text{SYSR}}^{(q)}(m) = B(0, k, \rho) + [1 - B(0, k, \rho)] Q_{\text{MRC}}^{(q)}(m, 10). \quad (84)$$

According to Section 5.4, SR code has the similar PCD as SYSR code. Hence,

$$Q_{\text{SR}}^{(q)}(m) \approx Q_{\text{SYSR}}^{(q)}(m), \quad (85)$$

the PCD analysis on SYSR code is applicable to SR code.

Defining failure probability as  $1 - Q_{\text{SYSRC}}^{(q)}(m)$ , the numerical results of both SYSMR code and SMR codes at  $m=0$  to 3 overhead symbols in channels of various erasure probabilities are presented in Table 7.4 and graphed in Figure 7.4. Both of them achieve failure probability of  $10^{-3.148}$  at zero overhead symbol with  $\rho=0.001$  and it increases to  $10^{-2.407}$  in a very lossy channel (e.g.,  $\rho=0.5$ ). At one overhead symbol, both systematic Micro-Random code and Stepping-Micro-Random code achieve much lower failure probabilities, i.e.,  $10^{-27.23}$  and  $10^{-26.49}$  with  $\rho=0.001$  and  $\rho=0.5$ , respectively. Two overhead symbols further improve the failure probabilities to  $10^{-51.31}$  and  $10^{-50.57}$  with the aforementioned channel erasure probabilities respectively. Meanwhile, they require zero overhead symbol to achieve complete decoding when  $\rho = 0$ .

Note that the improvement is not significant as compared to GF(2<sup>8</sup>) Random code from  $\rho = 10^{-1}$  to  $10^{-3}$  (See Figure 7.1). However, as discussed in Chapter 5 and 6, SYSMR code (SYSR code) and SMR code (SR code) have relatively better decoding complexities due to its Part I encoded symbols.

Comparing with the reported numerical results of RaptorQ code in Shokrollahi and Luby (2011), the failure probabilities of RaptorQ code fluctuate at about the same values as GF(2<sup>8</sup>) Random code, i.e.,  $10^{-2.407}$ ,  $10^{-4.815}$  and  $10^{-7.223}$  for zero, one and two overhead symbols, respectively as shown in Figure 2.9 to 2.11 of Section 2.2.3. Comparatively, SYSMR code SMR code have much more significant improvement in PCD starting at one overhead symbol onwards.

Table 7.4: The failure probabilities of systematic Micro-Random code and Stepping-Micro-Random code for  $m = 0$  to 3 in channel of various erasure probabilities.

	$\rho=0.001$	$\rho=0.005$	$\rho=0.01$	$\rho=0.05$	$\rho=0.1$	$\rho=0.5$
<b>m=0</b>	$10^{-3.148}$	$10^{-2.605}$	$10^{-2.469}$	$10^{-2.407}$	$10^{-2.407}$	$10^{-2.407}$
<b>m=1</b>	$10^{-27.23}$	$10^{-26.69}$	$10^{-26.55}$	$10^{-26.49}$	$10^{-26.49}$	$10^{-26.49}$
<b>m=2</b>	$10^{-51.31}$	$10^{-50.77}$	$10^{-50.63}$	$10^{-50.57}$	$10^{-50.57}$	$10^{-50.57}$
<b>m=3</b>	$10^{-75.40}$	$10^{-74.85}$	$10^{-74.72}$	$10^{-74.65}$	$10^{-74.65}$	$10^{-74.65}$

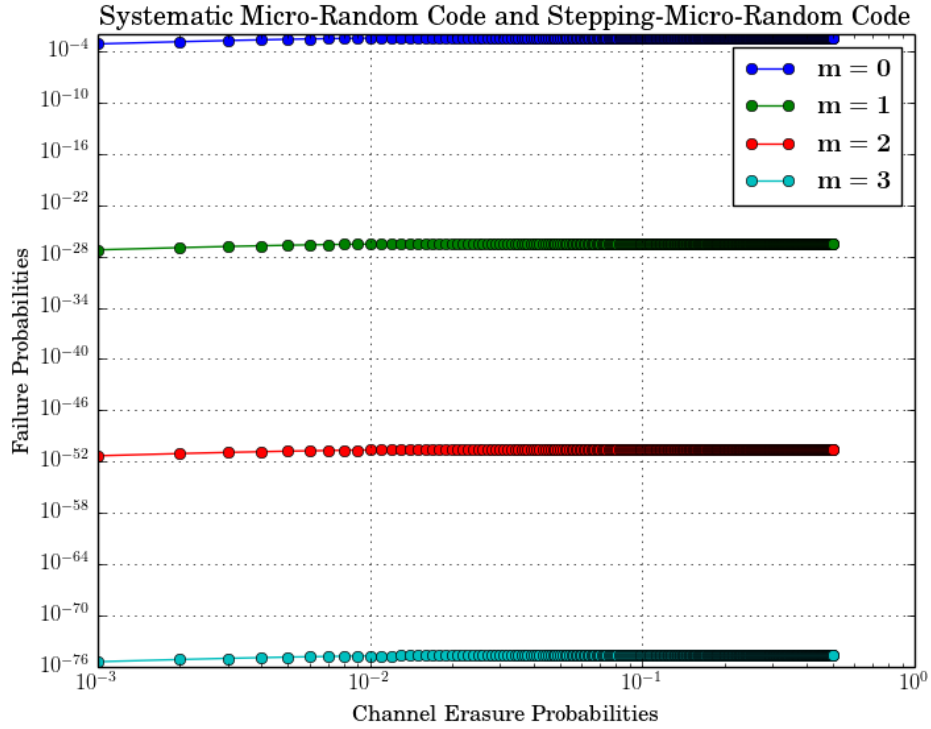


Figure 7.4: The failure probabilities of GF(2<sup>8</sup>) systematic Micro-Random code and Stepping-Micro-Random code at overhead symbols of  $m=0$  to 3 in channels of various erasure probabilities.

#### 7.4 Conclusion

We generalise both Random code and Micro-Random code to GF(2<sup>9</sup>) and construct systematic Micro-Random code and Stepping-Micro-Random code on GF(2<sup>8</sup>). The proposed GF(2<sup>8</sup>) rateless erasure codes have a much lower failure probabilities than RaptorQ code, i.e.,  $10^{-26.49}$  and  $10^{-50.57}$  at one and two overhead symbols, respectively. They can be used in the networks that have expensive retransmission cost. For example, the communications between low-earth-orbit satellites and ground stations are short due to rapid nodes mobility (i.e., satellites move out-of-sign if the messages are not delivered in time). Therefore, messages must be reconstructed with the least overhead symbols and a very low failure probability.



## CHAPTER 8

### CONCLUSION AND FUTURE WORK

To address the need in transmitting short messages with rateless erasure codes, we introduce Random code and propose its variants, namely MR code, SYSR code and SR code in the previous chapters. This chapter compares their performance in term of PCD, decoding complexities and applicable transmission scenarios altogether before drawing the conclusion and then discussing the future work.

#### 8.1 Probability of Complete Decoding (PCD) and Overhead Symbols

The required overhead symbols to achieve high PCD (i.e., 99.9% success probability to reconstruct the original message) are different for each proposed rateless erasure code. Referring to Table 8.1, Random code requires  $k+10$  encoded symbols (i.e., ten overhead symbols) to achieve high PCD no matter whether the channel is ideal or lossy. On the other hand, with a segmentation factor  $\alpha=10$ , Micro-Random code achieves the high PCD at  $k+1$  encoded symbols. On the other hand, both SYSR code and SR code have better performance when the channel erasure probabilities are low. They require zero overhead symbol in an ideal channel. Like Random code, both SYSR code and SR code require ten overhead symbols to achieve high PCD when the channels are lossy.

Table 8.1: The total required encoded symbols to achieve high PCD.

	Ideal Channel	Lossy Channel
Random Code	$k + 10$	$k + 10$
Micro-Random Code	$k + 1$	$k + 1$
Systematic Random Code	$k$	$k + 10$
Stepping-Random Code	$k$	$k + 10$

Table 8.2 illustrates the average number of encoded symbols for each code to achieve complete decoding. On average, Random code requires 1.6 overhead symbols to achieve complete decoding, irrespective of the message length as proven in Chapter 3. Micro-Random code outperforms Random code by achieving complete decoding with an average of 0.7 overhead symbols in both ideal and lossy channels. On the other hand, both SYSR code and SR code achieve complete decoding with zero overhead symbol if all their Part I encoded symbols are received intact in the ideal channel. They require an average of about 1.6 overhead encoded symbols in lossy channel just like Random code.

Table 8.2: The average encoded symbols to achieve complete decoding for each of the coding schemes.

	Ideal Channel	Lossy Channel
Random Code	$k + 1.6$	$k + 1.6$
Micro-Random Code	$k + 0.7$	$k + 0.7$
Systematic Random Code	$k$	$k + 1.6$
Stepping-Random Code	$k$	$k + 1.6$

## 8.2 Decoding Complexity

The decoding complexity of each code is presented in Table 8.3. Both Random code and Micro-Random code requires decoding complexity of  $O(k^3)$  in all the channel conditions – ideal and lossy. However, the analysis in Section 4.3.3 has suggested that Micro-Random code uses more time in reconstructing the original messages.

Generally, the decoding complexities of SYSR code and SR code vary according to the channel conditions (to be precise, the sequence of the received Part I encoded symbols). SYSR code reconstruct the original message with negligible decoding complexity if all the Part I encoded symbols are received intact, especially in an ideal channel condition or  $O(k^3)$  in lossy channel. On the other hand, SR code requires  $O(k)$  in an ideal channel condition and  $O(k^3)$  if the channel is lossy.

Table 8.3: The decoding complexity for each coding scheme in various channel conditions.

	Ideal Channel	Very Lossy Channel
Random Code	$O(k^3)$	$O(k^3)$
Micro-Random Code	$O((ak)^3)$	$O((ak)^3)$
Systematic Random Code	Negligible	$O(k^3)$
Stepping-Random Code	$O(k)$	$O(k^3)$

## 8.3 Transmission Scenarios

The transmission scenarios of the aforementioned rateless erasure codes are summarised in Table 8.4. Random code and Micro-Random code are

built on GF(2) random matrix. Hence, they are suitable for all the transmission scenarios as all the received encoded symbols forms a random matrix after all.

SYSR code has the smallest decoding complexity among the rest. However, it is applicable only at one-to-one and one-to-many transmission scenarios. The decoding complexity of SR code is slightly higher than SR code but SR code works in all transmission scenarios that listed in Table 8.4.

Table 8.4: The applicability for each rateless erasure code in various transmission scenarios.

	One-to-One	One-to-Many	Many-to-One
Random Code	✓	✓	✓
Micro-Random Code	✓	✓	✓
Systematic Random Code	✓	✓	✗
Stepping-Random Code	✓	✓	✓

#### 8.4 Conclusion

The state-of-art rateless erasure codes are efficient for long messages but the majority of the network traffic are short messages instead. To overcome this issue, this thesis proposes rateless erasure codes that are efficient in transmitting short messages.

Chapter 3's Random code is built on random matrix and it possesses a high decoding complexity. Nonetheless, the mathematical theorems in Sections 3.2 and 3.3 assure that Random code is able to achieve high PCD with  $k+10$  encoded symbols in transmitting both short and long messages

and in both ideal and lossy channels. These mathematical theorems are the key design to the rest of the proposed rateless erasure codes.

Chapter 4's Micro-Random code achieves high PCD at only  $k+1$  encoded symbols but with the trade-off of higher decoding complexity than Random code. Both SYSR code (Chapter 5) and SR code (Chapter 6) possess lower decoding complexity depending on the channel conditions. In particular, SYSR code has the lowest decoding complexity but it only works in one-to-one or one-to-many transmission scenarios. Though the decoding complexity of SR code is higher than SYSR code, it is applicable to all transmission scenarios.

## **8.5 Future Work**

In this section, we will discuss some future work to implement the proposed rateless erasure codes.

### **8.5.1 Transmitting Long Messages with Short Rateless Erasure Code**

Generally, short rateless erasure code can be used to transmit both short and long messages. We can transmit a long message with short rateless erasure code by segmenting the long message into multiple short messages and sending their respective encoded symbols in different sessions. To illustrate the idea, say we intend to transmit a message of  $k=1,000$  symbols to a receiver with Random code, in which it promises high PCD with ten overhead symbols. In this case, only 1% redundant encoded symbols is introduced and the decoding inefficiency  $\epsilon=0.01$ .

Assuming the receiver is a resource-constrained device. Instead of processing the 1000 symbols at one time, we segment the message into 10 short messages of 100 symbols and initiate the transmission (encoding) in ten sessions. Such a method requires less computational resources. However, ten overhead symbols are required in each session in order to reconstruct each short message with high PCD. In other words, a total of  $10 \times 10 = 100$  overhead symbols are needed in transmitting a message of  $k=1000$  symbols (segmented to ten short messages) with Random code. The decoding inefficiency becomes relatively high, i.e.,  $\epsilon=0.1$ . A novel method is required to address this issue.

## **8.5.2 Protocol Design**

In order for the short rateless erasure code to be deployed widely, we need to incorporate the short rateless erasure code into the TCP with minimum modification. We have listed a few issues in designing such protocol.

### **8.5.2.1 Computational Specification**

As we have suggested in previous section, the short rateless erasure code can be used to transmit the long messages. Therefore, both sender and receiver need to exchange the information about the computational resources during the initialization. For example, the receiver needs to specify the total message symbols and symbol size to transmit the message with no coding.

### **8.5.2.2 Delay Acknowledgement**

Instead of acknowledging each received packet, the receiver can acknowledge a group of them. This mechanism is commonly known as the delay acknowledgement in TCP research communities.

The delay acknowledgement influences the performance of the rateless erasure code aided TCP. The over-frequent acknowledgement introduces unnecessary delay to the network communication. In contrast, insufficient acknowledgement will increase the communication overhead caused by the short rateless erasure code as the sender unintentionally send excessive code symbols that may not be needed by the receiver.

We suggest the receiver send the first acknowledgement when the first 50% of the encoded symbols received. The second acknowledgement should be sent when 75% of the acknowledgement received. The frequency of the acknowledgement will be double and eventually the receiver will acknowledge the sender for every last few received encoded symbols. We leave the detailed study in future work.

## BIBLIOGRAPHY

1. Abdullah, N.F., Doufexi, A., and Piechocki, R.J., 2011. Raptor codes for infrastructure-to-vehicular broadcast services. *IEEE Vehicular Technology Conference*, pp. 1 – 5.
2. Abdullah, N.F., Doufexi, A., and Piechocki, R.J., 2013. Raptor codes-aided relaying for vehicular infotainment applications. *IET Communications*, 7(18), pp. 2064 – 2073.
3. Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D., and Ohlman, B., 2012. A survey of information-centric networking. *IEEE Communications Magazine*, 50(7), pp. 26 – 36.
4. Ahlswede, R., Cai, N., Li, S.Y., and Yeung, R.W., 2000. Network information flow. *IEEE Transactions on Information Theory*, 46(4), pp. 1204 – 1216.
5. Alcatel One Touch: Hardware specification [Online]. Available at <http://www.imei.info/phonedatabase/13112-alcatel-ot-4011x/> [Accessed: 1 Nov 2014].
6. Asus Nexus 7: Hardware specification [Online]. Available at [http://www.asus.com/Tablets\\_Mobile/Nexus\\_7/specifications/](http://www.asus.com/Tablets_Mobile/Nexus_7/specifications/) [Accessed: 1 Nov 2014].
7. Benson, T., Akella, A., and Maltz, D. A., 2010. Network traffic characteristics of data centers in the wild. *ACM SIGCOMM conference on Internet Measurement*, pp. 267 – 280.
8. Benson, T., Anand, A., Akella, A., and Zhang, M., 2010. Understanding data center traffic characteristics. *ACM SIGCOMM Computer Communication Review*, 40(1), pp. 92 – 99.
9. Berman, M., Chase, J.S., Landweber, L., Nakao, A., Ott, M., Raychaudhuri, D., Ricci, R., and Seskar, I., 2014. GENI: a federated testbed for innovative network experiments. *Elsevier Computer Networks*, 61, pp. 5 – 23.



10. Bitstring: a python module to construct, analyse and modify binary data [Online]. Available at <https://pypi.python.org/pypi/bitstring/3.1.3> [Accessed: 1 Nov 2014].
11. Blumenthal, M.S., and Clark, D.D., 2001. Rethinking the design of the Internet: the end-to-end arguments vs. the brave new world. *ACM Transactions on Internet Technology (TOIT)*, 1(1), pp. 70 – 109.
12. Bodine, E.A., and Cheng, M.K., 2008. Characterization of Luby transform codes with small message size for low-latency decoding. *IEEE International Conference on Communications*, pp. 1195 – 1199.
13. Bonald, T., Feuillet, M., and Proutiere, A., 2009, Is the "Law of the Jungle" sustainable for the Internet?, *IEEE INFOCOM*, pp. 28 – 36.
14. Botos, A., Polgar, Z. A., and Bota, V., 2010. Analysis of a transport protocol based on rateless erasure correcting codes. *IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, pp. 465 – 471.
15. Brownlee, N., and Claffy, K.C., 2002. Understanding Internet traffic streams: dragonflies and tortoises. *IEEE Communications Magazine*, 40(10), pp. 110 – 117.
16. Bursalioglu, O. Y., Caire, G., and Divsalar, D., 2011. Joint source-channel coding for deep space image transmission using rateless codes. *Information Theory and Applications Workshop (ITA)*, pp. 1 – 10.
17. Byers, J.W., Luby, M., Mitzenmacher, M., and Rege, A., 1998. A digital fountain approach to reliable distribution of bulk data. *ACM SIGCOMM Computer Communication Review*, 28(4), pp. 56-67.
18. Chen, C.C., Tahasildar, G., Yu, Y.T., Park, J.S., Gerla, M., and Sanadidi, M.Y., 2012. CodeMP: Network encoded multipath to support TCP in disruptive MANETs. *IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, pp. 209 – 217.
19. Chong, Z.K., Goi, B.M., Ewe, H.T., Tan, S.W., and Bryan, Ng .C.K.B., 2012a. Impact of fountain-based protocol in the uncongested and congested network. *International Conference on Computer and Information Science (ICCCIS)*, 2, pp. 747 – 754.

20. Chong, Z.K., Goi, B.M., Ohsaki, H., Ng, B.C.K., Ewe, H.T., 2013. Stepping-Random Code: A Rateless Erasure Code for Short-Length Messages. *IEICE Transactions on Communications*, E96-B (07), pp. 1764 – 1771.
21. Chong, Z.K., Ohsaki, H., Goi, B.M., Ewe, H.T., and Ng, C. K., 2012b. Performance analysis of fountain-based protocol in CSMA/CA wireless networks. International Symposium on Applications and the Internet (SAINT), pp. 184 – 189.
22. Clark, D., Shenker, S. and Falk, A., 2007. GENI research plan (Version 4.5). Available at <http://groups.geni.net/geni/raw-attachment/wiki/OldGPGDesignDocuments/GDD-06-28.pdf> [Accessed: 1 October 2014].
23. CRCmod: A python module to compute Cyclic Redundancy Check (CRC) [Online]. Available at <https://pypi.python.org/pypi/crcmod> [Accessed: 1 Nov 2014].
24. Du, Q., Ren, P., Lu, J., and Chen, Z., 2013. A novel cooperative multicast scheme based on fountain code. *Communications and Network*, 5(03), pp. 144 – 149.
25. Ebrahimi-Ghiri, R., and Keshavarz-Haddad, A., 2013. Novel broadcast schemes based on network codes for Ad Hoc networks with noisy wireless links. *IEEE Iranian Conference on Electrical Engineering (ICEE)*, pp. 1 – 6.
26. Feldmann, A. (2007). Internet clean-slate design: what and why?. *ACM SIGCOMM Computer Communication Review*, 37(3), pp. 59 – 64.
27. Fisher D., 2014. A look behind the future internet architectures efforts, *ACM SIGCOMM Computer Communication Review*, 44 (3), pp. 45 – 49.
28. Gaussian elimination: Solver for GF(2) matrix [Online]. Available at: [https://github.com/zkchong/gaussian\\_elimination\\_py](https://github.com/zkchong/gaussian_elimination_py) [Accessed: 1 Nov 2014].

29. Hyytia, E., Tirronen, T., and Virtamo, J., 2007. Optimal degree distribution for LT codes with small message length. *IEEE International Conference on Computer Communications*, pp. 2576 – 2580.
30. Kim, M., Médard, M., and Barros, J., 2011. Modeling network encoded TCP throughput: A simple model and its validation. *In Proceedings of the 5th International ICST Conference on Performance Evaluation Methodologies and Tools*, pp. 131 – 140.
31. Kim, S., Fonseca, R., and Culler, D., 2004. Reliable transfer on wireless sensor networks. *IEEE Sensor and Ad Hoc Communications and Networks*, pp. 449 – 459.
32. Kolchin, V.F. 1998. Random Graphs (Encyclopedia of Mathematics and its Applications). Cambridge University Press.
33. Lu, H., Lu, F., Cai, J., and Foh, C.H., 2013. LT-W: Improving LT Decoding With Wiedemann Solver. *IEEE Transactions on Information Theory*, 59(12), pp. 7887 – 7897.
34. Luby, M. G., Mitzenmacher, M., Shokrollahi, M. A., and Spielman, D. A., 2001. Efficient erasure correcting codes., *IEEE Transactions on Information Theory*, 47(2), pp. 569 – 584.
35. Luby, M., 2002. LT codes. *IEEE Symposium on Foundations of Computer Science*, pp. 271 – 280.
36. MacKay, D.J.C., 2005. Fountain codes. *IEE Proceedings-Communications*, 152(6), pp. 1062 – 1068.
37. MacKay, D.J.C., and Neal, R.M., 1995. Good codes based on very sparse matrices. *Cryptography and Coding*, pp. 100 – 111.
38. Mammi, E., Palma, V., Neri, A., and Carli, M., 2011. Fountain code based AL-FEC for multicast services in MANETs. *Wireless Telecommunications Symposium (WTS)*, pp. 1 – 7.
39. McAuley, A.J., 1990. Reliable broadband communication using a burst erasure correcting code. *ACM SIGCOMM Computer Communication Review*, 20(4), pp. 297 – 306.

40. Mehta, T., and Narmawala, Z., 2012. Performance enhancement of multimedia traffic over wireless ad hoc networks using network coding. *IEEE Nirma University International Conference on Engineering (NuiCONE)*, pp. 1 – 6.
41. Móczár Z., Molnar S., and Sonkoly B., 2014. Multi-platform performance evaluation of digital fountain based transport. *Science and Information Conference (SAI)*, pp. 690 – 697.
42. Molnár, S., Móczár, Z., and Sonkoly, B., 2014. How to transfer flows efficiently via the Internet?. *International Conference on Computing, Networking and Communications (ICNC)*, pp. 462 – 466.
43. Molnár, S., Móczár, Z., Temesváry, A., Sonkoly, B., Solymos, S., and Csicsics, T., 2013. Data transfer paradigms for future networks: Fountain coding or congestion control?. *In IEEE IFIP Networking Conference*, pp. 1 – 9.
44. Moors, T., 2002. A critical review of "end-to-end arguments in system design". *IEEE International Conference on Communications*, 2, pp. 1214 – 1219.
45. Ou, W., Yang, Z., Tang, L., and Zhongyang, G., 2012. Design of wireless sensor network based on random linear network coding. *IEEE International Conference on Computer Science and Service System (CSSS)*, pp. 986 – 990.
46. Pan, J., Paul, S., and Jain, R., 2011. A survey of the research on future internet architectures. *IEEE Communications Magazine*, 49(7), pp. 26-- 36.
47. Python: a programming language [Online]. Available at: <https://www.python.org/> [Accessed: 1 Nov 2014]
48. Qadri, N.N., Fleury, M., Altaf, M., and Ghanbari, M., 2010. Multi-source video streaming in a wireless vehicular ad hoc network. *IET Communications*, 4(11), pp. 1300 – 1311.
49. QPython: Python on Android [Online]. Available at <http://qpython.com/> [Accessed: 1 Nov 2014].

50. Raghavan, B., and Snoeren, A. C., 2006. Decongestion control. *In Proceedings of the Fifth Workshop on Hot Topics in Networks (HotNets-V)*, pp. 61 – 66.
51. Reed, I.S., 2000. A Brief History of the Development of Error Correcting Codes. *Computers and Mathematics with Applications*, 39(11), pp. 89 – 93.
52. Rizzo, L., 1997. Effective erasure codes for reliable computer communication protocols. *ACM SIGCOMM computer communication review*, 27(2), pp. 24 – 36.
53. Rout, R.R., and Ghosh, S.K., 2013. Enhancement of lifetime using duty cycle and network coding in wireless sensor networks. *IEEE Transactions on Wireless Communications*, 12(2), pp. 656 – 667.
54. Saltzer, J.H., Reed, D.P., and Clark, D. D., 1984. End-to-end arguments in system design. *ACM Transactions on Computer Systems (TOCS)*, 2(4), pp. 277 – 288.
55. Schwerdel, D., Reuther, B., Zinner, T., Müller, P., and Tran-Gia, P., 2014. Future internet research and experimentation: the G-Lab approach. *Elsevier Computer Networks*, 61, pp. 102 – 117.
56. Serpent: a python module to serialize object [Online]. Available at <https://pypi.python.org/pypi/serpent> [Accessed: 1 Nov 2014].
57. Sheu, J. P., Lee, C. Y., and Ma, C., 2013. An efficient transmission protocol based on network coding in delay tolerant networks. *IEEE International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, pp. 399 – 404.
58. Shokrollahi, A., 2006. Raptor codes. *IEEE Transactions on Information Theory*, 52(6), pp. 2551 – 2567.
59. Studholme, C., and Blake, I. 2006. Windowed erasure codes. *IEEE International Symposium on Information Theory*, pp. 509 – 513.
60. Sundararajan, J. K., Shah, D., Médard, M., Jakubczak, S., Mitzenmacher, M., and Barros, J., 2011. Network coding meets TCP: Theory and implementation. *Proceedings of the IEEE*, 99(3), pp. 490 – 512.

61. Wang, Y., Jain, S., Martonosi, M., and Fall, K., 2005. Erasure-coding based routing for opportunistic networks. *ACM SIGCOMM workshop on delay-tolerant networking*, pp. 229 – 236.
62. Wen, H., Lin, C., Ren, F., Yue, Y., and Huang, X., 2007. Retransmission or redundancy: transmission reliability in wireless sensor networks. *IEEE International Conference on Mobile Ad hoc and Sensor Systems*, pp. 1 – 7.
63. Wicker, S. B., and Bhargava, V. K., 1999. Reed-Solomon codes and their applications. John Wiley and Sons.
64. Xylomenos, G., Ververidis, C.N., Siris, V.A., Fotiou, N., Tsilopoulos, C., Vasilakos, X., Katsaros, K.V., and Polyzos, G.C., 2013. A survey of information-centric networking research. *IEEE Communications Surveys and Tutorials*, 16 (2), pp. 1024 – 1049.
65. Zeng, D., Guo, S., Jin, H., and Leung, V., 2012. Dynamic segmented network coding for reliable data dissemination in delay tolerant networks. *IEEE International Conference on Communications (ICC)*, pp. 63 – 67.
66. Zhang, W., and Hranilovic, S., 2009. Short-length raptor codes for mobile free-space optical channels. *IEEE International Conference on Communications*, pp. 1 – 5.
67. Zhang, X. and Ding, W., 2012. Comparative Research on Internet Flows Characteristics. *International Conference on Networking and Distributed Computing (ICNDC)*, pp. 114 – 118.
68. Zhang, Y., and Qiu, L., 2000. Understanding the End-to-End Performance Impact of RED in aHeterogeneous Environment. Cornell University.
69. Zhu, H., Zhang, C., and Lu, J., 2007. Designing of fountain codes with short code-length. *IEEE Signal Design and Its Applications in Communications*, pp. 65 – 68.
70. Zhu, X., Zhang, W., Li, B., and Zhang, L., 2010. Symmetric Distributed Joint Source-Channel Coding Using Raptor Codes. *IEEE*

*International Conference on Networking, Architecture and Storage (NAS)*, pp. 317 – 321.

## ACHIEVEMENT

### PUBLICATION

1. Chong, Z.K., Goi, B.M., Ohsaki, H., Ng, B.C.K., and Ewe, H.T., 2015. Improving The Probability of Complete Decoding of Random Code by Trading-off Computational Complexity. IET Communications. [Accepted on 1<sup>st</sup> September 2015].
2. Chong, Z.K., Goi, B.M., Ohsaki, H., Ng, B.C.K., and Ewe, H.T., 2015. Systematic Rateless Erasure Code for Short Messages Transmission. *Computers & Electrical Engineering*, Vol. 45, pp. 55 – 67.
3. Chong, Z.K., Goi, B.M., Ohsaki, H., Ng, B.C.K., and Ewe, H.T., 2015. Probability of Complete Decoding of Random Codes for Short Messages. *Electronics Letters*, 51(3), pp. 251 – 253.
4. Chong, Z.K., Goi, B.M., Ohsaki, H., Ng, B.C.K., and Ewe, H.T., 2013. Stepping-Random Code: A Rateless Erasure Code for Short-Length Messages. *IEICE Transactions on Communications*, E96-B (07), pp. 1764 – 1771.
5. Chong, Z.K., Goi, B.M., Ohsaki, H., Ng, B.C.K., and Ewe, H.T., 2012. Design of Short-Length Message Fountain Code for Erasure Channel Transmission. *IEEE Conference on Sustainable Utilization and Development in Engineering and Technology (STUDENT)*, pp. 239 – 241.
6. Chong, Z.K., Goi, B.M., Ewe, H.T., Tan S.W., and Ng, B.C.K., 2012. Impact of Fountain-based Protocol in the Uncongested and Congested Network. *International Conference on Computer and Information Sciences (ICCIS)*, 2, pp. 747 – 754.
7. Chong, Z.K., Ohsaki, H., Goi, B.M., Ng, B.C.K., and Ewe, H.T., 2012. Performance Analysis of Fountain-based Protocol in CSMA/CA Wireless Networks. *The 12th IEEE/IPSJ International Symposium on Applications and the Internet (SAINT)*, pp. 184 – 189.

### RESEARCH PROJECTS

1. Transmitting Long-Length Messages to Resource Constraint Mobile Computing Devices with Rateless Erasure Code, UTAR Research Fund 2014 – 2015.
2. Reliable Data Transfer for High Altitude Platform (HAP)'s Wireless Access Networks, UTAR Research Fund 2013 – 2014.
3. Evaluation Of The Performance Of Rateless encoded TCP In Small-Scale Networks., UTAR Research Fund, 2012 – 2013.
4. Qos And Over-Provisioning In Backbone Network, UTAR Research Fund, 2011.



