### COMPROMISING RANDOM LINEAR NETWORK CODING AS A CIPHER

**SRAVYA BETHU**

### Bachelor of Technology in Information Technology

Teegala Krishna Reddy Engineering College May 2018

submitted in partial fulfillment of the requirements for the degree **MASTER OF COMPUTER AND INFORMATION SCIENCE**

at the

### CLEVELAND STATE UNIVERSITY

May 2022

We hereby approve this thesis for SRAVYA BETHU

Candidate for the Master of Computer and Information Science degree for the Department of Electrical Engineering and Computer Science and the CLEVELAND STATE UNIVERSITY

College of Graduate Studies by

Thesis Chairperson, Ye Zhu, PhD

Department & Date

Sunnie Sun Chung, PhD

Department & Date

Sathish Kumar, PhD

Department & Date

Siu-Tung Yau, PhD

Department & Date

Student’s Date of Defense: May 7, 2021

### DEDICATION

I dedicate this thesis to my family, cousins and many friends. A special feeling of gratitude to my parents, brother Anvesh, friends Sahithi and Kiran for constantly encouraging and helping me at every step of my life. I would also like to dedicate this to my grandparents, aunts and uncle for their kindheartedness and blessings.

### ACKNOWLEDGMENTS

First and foremost, I would like to praise and thank God, the Almighty who has granted countless blessings including the opportunity to study in the United States. Foremost thanks to my parents whose selfless sacrifices and constant support is the prime reason I am here today. Moreover, I am very thankful to my elder brother Anvesh and uncle Jagadeesh here in the US who were always welcoming and helped me get acclimatize to the US education system and culture.

I would like to express my deepest appreciation and indebtedness to countless individuals. First, I would like to express my deepest gratitude to my thesis advisor, Dr. Ye Zhu, for providing me a challenging opportunity, invaluable advice and pa- tience at every stage of the research project. I would also like to thank my committee members, Dr. Sunnie Sun Chung, Dr. Sathish Kumar, Dr. Siu-Tung Yau for their time in reviewing my thesis and providing insightful comments. I thank my follow members at Network Security research group for their assistance in different ways.

Finally, I am grateful to all my friends and family for their unwavering support and encouragement throughout my studies as well as during the research and writing this thesis.

### COMPROMISING RANDOM LINEAR NETWORK CODING AS A CIPHER

**SRAVYA BETHU ABSTRACT**

Due to its potential improvement on network throughput, network coding has attracted considerable research interests. Random Linear Network Coding (RLNC), as a branch of research on network coding, is proposed as a cipher to protect the confidentiality of packets possibly on the future Internet due to its features such as packet mixing in a decentralized approach. In this paper, we propose attacks to compromise RLNC as a cipher for confidentiality protection. The attacks are based on the blind source separation (BSS) technique, a statistical signal processing technique designed to recover original signals based on mixtures of original signals. We also design the scaling step to filter out artifacts generated by BSS algorithms and the cross-checking steps to increase confidence on packets recovered by the proposed attacks. Our extensive experiments on packets collected from the Internet and a campus network show that the attacks can successfully recover about 1% of original packets.

Keywords - Random Linear Network Coding, Independent Component, Blind Source Separation.

### TABLE OF CONTENTS

Page ABSTRACT vi

[LIST OF FIGURES ix](#_TOC_250015)

CHAPTER

I. INTRODUCTION . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1

II. BACKGROUND . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

1. NETWORK MODEL AND THREAT MODEL . . . . . . . . . . . . . . 5
2. COMPROMISING RLNC AS A CIPHER . . . . . . . . . . . . . . . . . 7

[4.1 Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7](#_TOC_250014)

[4.2 Separation Step . . . . . . . . . . . . . . . . . . . . . . . . . . . 8](#_TOC_250013)

[4.3 Scaling Step . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9](#_TOC_250012)

[4.4 Cross-Checking Step 11](#_TOC_250011)

1. EVALUATION 13
   1. [Experiment Setup 13](#_TOC_250010)
   2. [Performance Metrics 14](#_TOC_250009)
   3. [Packet Length 15](#_TOC_250008)
   4. [Comparison on Different Packet Types 22](#_TOC_250007)
      1. [Ethernet Packets vs. WIFI Packets 22](#_TOC_250006)
      2. [Comparison on Applications 26](#_TOC_250005)
   5. [Check Patterns 31](#_TOC_250004)
   6. [Generation Size 49](#_TOC_250003)
   7. [Dependence Between Packet Payload 50](#_TOC_250002)
   8. [Finite Field Size 58](#_TOC_250001)
2. DISCUSSION 67
3. RELATED WORK 68
4. CONCLUSION 71

[BIBLIOGRAPHY 72](#_TOC_250000)

### LIST OF FIGURES

Figure Page

1 Random Linear Network Coding (*gs* = 3) . . . . . . . . . . . . . . . . 3 2 Experiment Setup 13

1. Effect of Packet Length (Duplication Method, *gs* = 3, *nr* = 2) 16
2. Effect of Packet Length (Reduction Method, *gs* = 3, *nr* = 2) 17
3. Actual Byte Recovery Rate for Declared Packets (Duplication Method,

*gs* = 3, *nr* = 2) 17

1. Actual Byte Recovery Rate for Declared Packets (Reduction Method,

*gs* = 3, *nr* = 2) 18

1. Similarity (Packet length: 1500 bytes, *gs* = 3, *nr* = 2) 19
2. Similarity (Different Types of Packets, *gs* = 3, *nr* = 2) 20
3. Effect of Packet Length. (Duplication Method, *gs* = 3, *nr* = 2,

*patterns* : *CK, patternc* : *NF* ) 20

1. Actual Byte Recovery Rate for Declared Packets (Duplication Method,

*gs* = 3, *nr* = 2) 21

1. Ethernet Packets (*gs* = 3, *nr* = 2, Duplication Method) 22
2. WIFI Packets (*gs* = 3, *nr* = 2, Duplication Method) 23
3. Comparison between Ethernet and WIFI Packets (*gs* = 3, *nr* = 2, Duplication Method, *patterns* : *CK, patternc* : *NF* ) 23
4. Similarity (Ethernet Packets, *gs* = 3, *nr* = 2) 24
5. Similarity (WIFI Packets, *gs* = 3, *nr* = 2) 24
6. Actual Byte Recovery Rate for Declared Packets (Ethernet Packets, Duplication Method, *gs* = 3, *nr* = 2) 25
7. Actual Byte Recovery Rate for Declared Packets (WIFI Packets, Du-

plication Method, *gs* = 3, *nr* = 2) 25

1. Skype Video Call Packets (*gs* = 3, *nr* = 2, Duplication Method) 26
2. Skype Voice Call Packets (*gs* = 3, *nr* = 2, Duplication Method) 27
3. File Downloading Packets (*gs* = 3, *nr* = 2, Duplication Method) 27
4. Comparison between Skype Video and Voice calls and File Download- ing Packets (*gs* = 3, *nr* = 2, Duplication Method, *patterns* : *CK*,

*patternc* : *NF* ) 28

1. Similarity (Video packets, *gs* = 3, *nr* = 2) 28
2. Similarity (Voice packets, *gs* = 3, *nr* = 2) 29
3. Similarity (File Downloading Packets, *gs* = 3, *nr* = 2) 29
4. Actual Byte Recovery Rate for Declared Packets (Video Packets, Du- plication Method, *gs* = 3, *nr* = 2) 30
5. Actual Byte Recovery Rate for Declared Packets (Voice Packets, Du- plication Method, *gs* = 3, *nr* = 2) 30
6. Actual Byte Recovery Rate for Declared Packets (File Downloading Packets, Duplication Method, *gs* = 3, *nr* = 2) 31
7. *patterns* : *CK* (*gs* = 3, *nr* = 2, Duplication Method) 32
8. *patterns* : *NF* (*gs* = 3, *nr* = 2, Duplication Method) 33
9. *patterns* : *LR* (*gs* = 3, *nr* = 2, Duplication Method) 33
10. *patterns* : *UP* (*gs* = 3, *nr* = 2, Duplication Method) 34
11. Actual Byte Recovery Rate for Declared Packets (*patterns* : *CK*, Du- plication Method, *gs* = 3, *nr* = 2) 34
12. Actual Byte Recovery Rate for Declared Packets (*patterns* : *NF* , Du- plication Method, *gs* = 3, *nr* = 2) 35
13. Actual Byte Recovery Rate for Declared Packets (*patterns* : *LR*, Du- plication Method, *gs* = 3, *nr* = 2) 35
14. Actual Byte Recovery Rate for Declared Packets (*patterns* : *UP* , Du- plication Method, *gs* = 3, *nr* = 2) 36
15. *patterns* : *CK* (*gs* = 3, *nr* = 2, Duplication Method) 36
16. *patterns* : *NF* (*gs* = 3, *nr* = 2, Duplication Method) 37
17. *patterns* : *LR* (*gs* = 3, *nr* = 2, Duplication Method) 37
18. *patterns* : *UP* (*gs* = 3, *nr* = 2, Duplication Method) 38
19. Actual Byte Recovery Rate for Declared Packets (*patterns* : *CK*, Du- plication Method, *gs* = 3, *nr* = 2) 38
20. Actual Byte Recovery Rate for Declared Packets (*patterns* : *NF* , Du- plication Method, *gs* = 3, *nr* = 2) 39
21. Actual Byte Recovery Rate for Declared Packets (*patterns* : *LR*, Du- plication Method, *gs* = 3, *nr* = 2) 39
22. Actual Byte Recovery Rate for Declared Packets (*patterns* : *UP* , Du- plication Method, *gs* = 3, *nr* = 2) 40
23. Comparison between Pattern used for Cross-checking (*gs* = 3, *nr* = 2, Duplication Method) 40
24. Comparison between Pattern used for Cross-checking, Packet length:

1500 bytes (*gs* = 3, *nr* = 2, Duplication Method) 41

1. *patterns* : *CK* (*gs* = 3, *nr* = 2, Reduction Method) 41
2. *patterns* : *NF* (*gs* = 3, *nr* = 2, Reduction Method) 42
3. *patterns* : *LR* (*gs* = 3, *nr* = 2, Reduction Method) 42
4. *patterns* : *UP* (*gs* = 3, *nr* = 2, Reduction Method) 43
5. Actual Byte Recovery Rate for Declared Packets (*patterns* : *CK*, Re- duction Method, *gs* = 3, *nr* = 2) 43
6. Actual Byte Recovery Rate for Declared Packets (*patterns* : *NF* , Re- duction Method, *gs* = 3, *nr* = 2) 44
7. Actual Byte Recovery Rate for Declared Packets (*patterns* : *LR*, Re- duction Method, *gs* = 3, *nr* = 2) 44
8. Actual Byte Recovery Rate for Declared Packets (*patterns* : *UP* , Re- duction Method, *gs* = 3, *nr* = 2) 45
9. *patterns* : *CK* (*gs* = 3, *nr* = 2, Reduction Method) 45
10. *patterns* : *NF* (*gs* = 3, *nr* = 2, Reduction Method) 46
11. *patterns* : *LR* (*gs* = 3, *nr* = 2, Reduction Method) 46
12. *patterns* : *UP* (*gs* = 3, *nr* = 2, Reduction Method) 47
13. Actual Byte Recovery Rate for Declared Packets (*patterns* : *CK*, Re- duction Method, *gs* = 3, *nr* = 2) 47
14. Actual Byte Recovery Rate for Declared Packets (*patterns* : *NF* , Re- duction Method, *gs* = 3, *nr* = 2) 48
15. Actual Byte Recovery Rate for Declared Packets (*patterns* : *LR*, Re- duction Method, *gs* = 3, *nr* = 2) 48
16. Actual Byte Recovery Rate for Declared Packets (*patterns* : *UP* , Re- duction Method, *gs* = 3, *nr* = 2) 49
17. *gs* = 4 (*nr* = 3, Reduction Method, WIFI Packets) 50
18. *gs* = 5 (*nr* = 4, Reduction Method, WIFI Packets) 51
19. *gs* = 4 (*nr* = 3, Duplication Method, WIFI Packets) 51
20. *gs* = 5 (*nr* = 4, Duplication Method, WIFI Packets) 52
21. Comparison between Generation Numbers (Reduction Method, *patterns* :

*CK*, *patternc* : *NF* ) 52

1. Comparison between Generation Numbers (Duplication Method, *patterns* :

*CK*, *patternc* : *NF* ) 53

1. Actual Byte Recovery Rate for Declared Packets, *gs* = 4 (*nr* = 3, Reduction Method, WIFI Packets) 53
2. Actual Byte Recovery Rate for Declared Packets, *gs* = 5 (*nr* = 4, Reduction Method, WIFI Packets) 54
3. Actual Byte Recovery Rate for Declared Packets, *gs* = 4 (*nr* = 3, Duplication Method, WIFI Packets) 54
4. Actual Byte Recovery Rate for Declared Packets, *gs* = 5 (*nr* = 4, Duplication Method, WIFI Packets) 55
5. Actual Packet Recovery Rate for Packet size 1500 (*gs* = 3, *nr* = 2, Duplication Method, WIFI Packets) 55
6. Actual Packet Recovery Rate for Packet size 1200 (*gs* = 3, *nr* = 2, Duplication Method, WIFI Packets) 56
7. Actual Packet Recovery Rate for Packet size 1000 (*gs* = 3, *nr* = 2, Duplication Method, WIFI Packets) 56
8. Actual Packet Recovery Rate for Packet size 800 (*gs* = 3, *nr* = 2, Duplication Method, WIFI Packets) 57
9. Actual Packet Recovery Rate for Packet size 600 (*gs* = 3, *nr* = 2, Duplication Method, WIFI Packets) 57
10. Actual Packet Recovery Rate for Packet size 400 (*gs* = 3, *nr* = 2, Duplication Method, WIFI Packets) 58
11. GF(25) Finite Field (*gs* = 3, *nr* = 2, Reduction Method, WIFI Packets) 59
12. GF(26) Finite Field (*gs* = 3, *nr* = 2, Reduction Method, WIFI Packets) 59
13. GF(27) Finite Field (*gs* = 3, *nr* = 2, Reduction Method, WIFI Packets) 60
14. GF(28) Finite Field (*gs* = 3, *nr* = 2, Reduction Method, WIFI Packets) 60
15. GF(25) Finite Field (*gs* = 3, *nr* = 2, Duplication Method, WIFI Packets) 61
16. GF(26) Finite Field (*gs* = 3, *nr* = 2, Duplication Method, WIFI Packets) 61
17. Comparison between Finite Field Size (Reduction Method, *patterns* :

*CK*, *patternc* : *NF* ) 62

1. Comparison between Finite Field Size (Duplication Method, *patterns* :

*CK*, *patternc* : *NF* ) 62

1. GF(25) Actual Byte Recovery Rate for Declared Packets (*gs* = 3, *nr* =

2, Reduction Method, WIFI Packets) 63

1. GF(26) Actual Byte Recovery Rate for Declared Packets (*gs* = 3, *nr* =

2, Reduction Method, WIFI Packets) 63

1. GF(27) Actual Byte Recovery Rate for Declared Packets (*gs* = 3, *nr* =

2, Reduction Method, WIFI Packets) 64

1. GF(28) Actual Byte Recovery Rate for Declared Packets (*gs* = 3, *nr* =

2, Reduction Method, WIFI Packets) 64

1. GF(25) Actual Byte Recovery Rate for Declared Packets (*gs* = 3, *nr* =

2, Duplication Method, WIFI Packets) 65

1. GF(26) Actual Byte Recovery Rate for Declared Packets (*gs* = 3, *nr* =

2, Duplication Method, WIFI Packets) 65

### CHAPTER I

**INTRODUCTIO**N

Network coding has attracted considerable research interests due to its poten- tials to improve network throughput. Through network coding, a router can combine multiple packets into one packet for transmission and the combination leads to more efficient use of network bandwidth. Network coding has been proven effective in improving throughput of multicast traffic [1] and wireless traffic [2].

As a branch of network coding, random linear network coding (RLNC) achieves the throughput improvement with a decentralized approach. In RLNC, the coding vectors used for combining packets are generated locally and randomly [2]. A receiving node can recover original packets if enough linearly independent combinations of original packets are received. Because of the combination, a packet of interest will be combined with several other packets to form new combined packets. So RLNC is similar as the one-time pad cipher which combines message bits with bits of a pre- shared key by modular addition. One-time pad has perfect secrecy as the pre-shared key has the same length of the message to be sent and the key bits are generated randomly. Similarly RLNC may combine packets of similar sizes and the packets to be combined can be generated from independent senders. The similarity raised interests from researchers to use RLNC as a cipher [3].

In this paper, we propose an attack to break RLNC as a cipher. The attack is based on the blind source separation (BSS) technique, which is designed to recover original signals based on mixtures of original signals. The attack first applies the BSS technique to recover original packets based on the combined packets generated by RLNC. But the separation results can be only loosely related to the original packets. So we design the scaling step to filter out separated packets that are only loosely connected to original packets and find the right scaling factors to restore the reconstructed packets as original packets. In the final step, cross-checking on the reconstructed packets is performed to enhance the adversary’s confidence by checking known patterns in headers of reconstructed packets such as urgent pointer.

We implemented the attack and evaluated performance of the proposed attack on packets captured from our campus network, streaming services, telecommunica- tions applications, downloading iso files, etc.. In our experiments, we assume that the attacker can not obtain enough linearly independent combination of original packets to recover original packets as a normal receiver of RLNC. Our extensive experiment results on the captured real packets show that the attack can recover about 1% of original packets with exactly same bytes of original packets. Our experiments also show that the attacker can be confident on the packets that verified by cross-checking. On average, the proposed attacks can recover about 4% of bytes of original packets. Given the experiment results, we believe that further research is needed on using RLNC as a cipher.

The rest of the paper is organized as follows. We review related work in Section

2. The network model and threat model are described in 3. We present the detailed design of the BSS-based attack in Section 4 In Section 5, we present our experiments on real packets captured from the Internet. We discuss few points in Section 6 and conclude this paper in Section 8.

### CHAPTER II

**BACKGROUND**

Network coding was originally proposed to save bandwidth for multi-cast traf- fic [1]. In traditional networks, every intermediate node simply stores and forwards network packets. Network coding allows an intermediate node to mix multiple re- ceived packets according to a coding vector and then forward the combined packet. A receiving node can recover the original packets from the sending node by decoding the combined packets with Gaussian elimination [4].

Random linear network coding (RLNC) was proposed for robust and dis- tributed transmission [2] [5], [6]. Instead of having coding vectors centrally deter- mined as in the previous network coding schemes [1], RLNC allows each intermediate node to generate coding vectors randomly and locally. A practical scheme for RLNC including the packet format was proposed by Chou *et al.* [4]. In the scheme, a fixed

*in* *out*  *in*  *in*  *in*



*p*

*p* *c p* *c p* *c p*

1 1 11 1 12 2 13 3

*in* *out*  *in*  *in*  *in*

*p*

*p* *c p* *c p* *c p*

2 2 21 1 22 2 23 3

*in* *out*  *in*  *in*  *in*

*p*

*p* *c p* *c p* *c p*

3 3 31 1 32 2 33 3

Figure 1: Random Linear Network Coding (*gs* = 3)

number of original packets form a generation. The fixed number is defined as the gen- eration size in [7] and denoted as *gs* in this paper. As shown in Figure 1, a random coding vector [*c*11*, c*12*, c*13] is generated by the node from a Galois field. The incoming

packets *pin*, *pin*, and *pin* are combined according to the random coding vector to gen-

1 2 3

erate the combined packet *pout* = *c*11*pin* + *c*12*pin* + *c*13*pin* . With the matrix notations,

1 1 3 3

we can describe the combined packets as follows:

**P***out* = **C** · **P***in* (2.1)

where

 *pin* 

 *pout* 

 *c*11 *c*12 *c*13 

 1   1   

**P***in* = *pin*

, **P***out* = *pout*

, and **C** = *c*21 *c*22 *c*23

.

 2

  2

 



 *pin* 

 *pout* 

 *c*31 *c*32 *c*33 

The combined packets *pout*, *pout*, and *p out* will be tagged with the same gener-

3

3

1 2 3

ation number and then forwarded with the coding vectors according to the practical network coding scheme [7]. The next intermediate node will also generate random coding vectors locally and continues the coding process. The coding vector in the combined packets will be updated according to the random coding vectors. Even- tually the receiving node can recover the original packets by solving a linear system similar as Equation 2.1 where **C** is the matrix containing the overall coding vectors updated by each intermediate node with local random coding vectors and **Pout** are the combined packets received by the receiving node. The linear system is usually solved with Gaussian elimination.

### CHAPTER III NETWORK MODEL AND THREAT

**MODEL**

The goal of the research is to verify whether RLNC can be used as a free cipher to protect confidentiality of network packets. We assume the network model as follows: (1) RLNC is used in the network for packet transmission as described in Section 2. The coding vectors are generated locally and randomly for RLNC as described in [7]. (2) The format of packets generated by RLNC is exactly as specified in [4]. As described in [4], packets generated by RLNC with the same set of coding vectors are in the same generation and these packets are tagged with the same generation number. The number of packets belonging to the same generation is called as the generation size, denoted as *gs*. (3) The packets generated by RLNC may following possibly different paths to destinations. (4) We assume the network operators are replying on RLNC as a free cipher [3]. So no further encryption is in use to protect confidentiality of the packets.

We consider the adversary with the following capabilities: (1) The adversary is able to eavesdrop RLNC packets on network links. (2) Since no further encryption is in use, the attacker is able to classify RLNC packets in terms of the generation number.

(3) Since the adversary may be able to eavesdrop on multiple links in the network, the adversary may be able to collect multiple RLNC packets belonging to the same generation. However, we assume that the number of packets belonging to the same generation is less than *gs*, the generation size. The assumption is important, because if the adversary is able to collect *gs* RLNC packets belong to the same generation, the adversary can decode RLNC packets as a regular receiver using the Gaussian elimination [4].

### CHAPTER IV

**COMPROMISING RLNC AS A CIPHER**

### Overview

The main reason behind using RLNC as a cipher is the combination of packets with randomly-generated coding vectors. Because of the combination, a packet of interest will be combined with several other packets to form new combined packets. So RLNC as a cipher is similar as the one-time pad scheme which combines message bits with bits of a pre-shared key by modular addition. One-time pad has perfect secrecy as the pre-shared key has the same length of the message to be sent and the key bits are generated randomly. Similarly RLNC may combine packets of similar sizes and the packets to be combined can be generated from independent senders. The similarity raised interests from researchers to use RLNC as a cipher.

However, there is a significant difference between RLNC and one-time pad. To successfully decode combined packets of one generation, a receiver in an RLNC network needs to receive at least *gs* combined packets of the generation and at least *gs* combined packets among the received packets need to be linearly independent. The requirement of RLNC on multiple combined packets generated from packets from the same generation enables recovery of original packets.

The attack on RLNC as a cipher is based on the blind source separation (BSS) technique, which is designed to recover original signals based on mixtures of original signals. The attack can be divided into three steps: the separation step based on the BSS technique, the scaling step designed to search for a correct scaling factor, and the optional cross-checking step designed to increase confidence on attack results. The details of each step are described in the rest of this section.

### Separation Step

BSS algorithms were originally designed to solve the cock-tail party problem, in which a number of microphones positioned at different locations collect mixtures of voice signals from party participants. The goal of the cock-tail party problem is to recover voice signals from individual participants based on the mixtures. In its simplest form, the cocktail-party problem assumes *n* individual voice signals *V*1(*t*)*,*· · · ,

V*n*(*t*) and *n* mixtures of the individual voice signals *M*1(*t*)*,*· · · , M*n*(*t*) where

*Mi*(*t*) = Σ*n cijVj*(*t*) and *cij* denotes the mixing coefficients In the cock-tail

*j*=1

part problem, the mixing coefficient *cij* is largely determined by the attenuation of the voice signal from the *j*th participant to the *i*th microphone.

BSS, as a methodology used in statistical signal processing, assumes indepen- dence among the individual signals *Vj*(*t*). BSS algorithms reconstructs the individual signals *Vj*(*t*) by maximizing independence among individual signals recovered from the mixtures *Mi*(*t*). The algorithms are called “blind” to emphasize that both the mixing coefficients *cij* and the individual voice signals *Vj*(*t*) are unknown to the algorithms. The common approaches used in BSS include maximization of likelihood [8] [9] [10], and minimization of mutual information [11] [12] [13]. When individual signals *Vj*(*t*) are dependent, timing-structure based algorithms [14] can be used for separation.

Similar as the microphones in the cock-tail party problem collecting mixtures

of individual voice signals *Vj* (*t*), RLNC combines original packets to form combined packets. So BSS algorithms can be used to recover original packets based on the com- bined packets collected by eavesdroppers. In comparison with the traditional cock-tail party problem, the recovery of original packets based on the combined packets has two major differences: (1) The combination of RLNC is through operations defined in a Galois field. So we use AMERICA [15] [16] [17], a BSS algorithm designed for Galois fields, to recover original packets. (2) As described in Section 3, the num- ber of combined packets available to the adversary is usually less than the number of original packets, i.e., the generation size *gs*. So the recovery of original packets is usually classified as an overcomplete separation problem, in which the number of mixtures is less than the number of individual signals. When we apply the AMER- ICA algorithm to recover original packets, we randomly duplicate combined packets so that the number of combined packets, i.e, the number of mixtures, is the same as the number of the original packets, i.e., the number of individual signals or we reduce the number of packets to be recovered to the number of combined packets. In other words, the number of original packets is assumed to be the same as the number of combined packets. In the rest of the paper, we call the two methods as *duplication method* and *reduction method* respectively. Since in general overcomplete separation problems are harder, the separation performance can degrade.

### Scaling Step

The purpose of applying the AMERICA BSS algorithm is to reconstruct orig- inal packets based on the combined packets generated by RLNC. But the separation results can be only loosely related to the original packets. The output of the sepa- ration step, i.e., the separated packets can be classified into four categories: (1) The reconstruction is successful. In other words, a separated packet can be correlated to

one of the original packets. Usually the separated packet in this category is a scaled version of the original packet. (2) A separated packet can be correlated to an aggre- gate of more than one original packets. It means that the BSS algorithm does not fully separate the combined packets into individual packets. Usually overcomplete problems are more likely to generate output in this category. (3) A separated packet can be correlated to multiple original packets in different portions of the packet. It means that the BSS algorithm only partially separated out original packets. (4) The output can also contain separated packets representing noise.

**input :** SP - an array of separated packets, i.e., the output of the AMERICA BSS algorithm; *offsetbegin*, *offsetend* - the two offsets are used to extract header bits/bytes specified by the two offsets from a packet; *patterns* - the expected pattern for specific header bits/bytes;

**output:** - an array of reconstructed packets ;

RP

### foreach *p* do

∈ SP

**foreach** *possible scaling factor s in the Galois Field* **do**

*tmp* ← *s* × *p*; // The multiplication is defined on the Galois Field.

*bstring* ← header bits/bytes specified by the two offsets *offsetbegin*

and *offsetend* extracted from packet *tmp*;

**if** *pattern on bstring is consistent with patterns* **then**

add packet *tmp* into

RP

### end end

**end**

**Algorithm 1:** Scaling Step

The goal of the scaling step is to filter out separated packets in the last three categories as described above and find the right scaling factors to restore the recon- structed packets as original packets. The pseudocode of the scaling step is shown in Algorithm 1. The algorithm determines a successfully recovered packet by checking patterns on header bytes or bits specified by the two offsets *offsetbegin* and *offsetend*, from the beginning of IP packets. For example, if the header field selected is the 16-bit IPv4 header checksum field, then the expected pattern *patterns* should be that the

bytes of the checksum field match the checksum of the packet header. Unused header bytes or header bytes with fixed patterns can also be used for pattern checking. A list of header fields and corresponding expected patterns is shown in Table I.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *patternexp* | Header Field | Number  of Bits | Protocol | Pattern Description |
| CK | Header check-  sum | 16 | IP | *offsetbegin* =  80*, offsetend* = 95*,* Pattern: The bits in this range should match the checksum of IP  headers |
| NF | IP flags and  fragment off- set | 8 | IP | *offsetbegin* =  48*, offsetend* = 55. Pattern: 0x40 (IP flags: Reserved bit: 0; Don’t fragment (DF): 1; More fragments (MF): 0.  Fragment offset (initial 5 bits): all 0 bits.) |
| UP | Urgent  pointer | 16 | TCP | *offsetbegin* =  304*, offsetend* = 319*,* Pattern: 0x0000 (Urgent  pointer (16 bits): all 0 bits.) |
| LR | Data offset,  reserved field and TCP Flag | 8 | TCP | *offsetbegin* =  256*, offsetend* = 263*,* Pattern: 0x80 or 0x50 (Data offset (4 bits): 1000  or 0110; Reserved (3 bits):  all 0 bits; TCP flags: Nonce (NS) bit: 0.) |

Table I: Header Fields and Corresponding Expected Patterns

### Cross-Checking Step

The cross-checking step is designed to enhance the adversary’s confidence on the reconstructed packets in RP, i.e., the output of Algorithm 1. In the previous step, a scaling factor is determined by checking a pattern on one header field. But the pattern can possibly exist in a packet not successfully reconstructed. To increase

|  |  |  |  |
| --- | --- | --- | --- |
| Packet  Length (Bytes) | Transport  Layer Proto- col | Content | Encryption |
| 162 | TCP | Video stream-  ing | No encryption |
| 225 | UDP | Web browsing | SSL |
| 557 | TCP | Web browsing | TLS |
| 900 | TCP | Web browsing | No encryption |
| 1086 | TCP | Web browsing | TLS |
| 1188 | UDP | File download | TLS |
| 1320 | TCP | Web browsing | No encryption |
| 1500 | TCP | Web browsing | TLS, SSL |
| 2948 | TCP | Web browsing | SSL |

Table II: Packet Types

the adversary’s confidence, this step checks each packet in RP for more patterns listed in Table I. The patterns to be used for checking may depend on the prior information that the adversary has on the original packets. For example, if the adversary knows that the original packets are TCP packets, then the pattern on the urgent pointer field of the TCP protocol, as listed in Table I, can be used for cross- checking. Obviously if more prior information that the adversary has on the original packets, more patterns can be used for cross-checking and the adversary will be more confident on the recovered packets.

### CHAPTER V

**EVALUATION**

### Experiment Setup

*in*

## p



1

*in*

*p*

2

*in gs*

*p*

Figure 2: Experiment Setup

*p out*

1

*out*

# p

2

*pout*

*gs*

To evaluate the performance of the proposed attack, we collected packets from both WIFI networks and wired networks with Wireshark (version 3.2.7). As shown in Table II, we collect packets generated by various protocols for the performance evaluation. The captured packets are combined as shown in Figure 2. The node first generates random coding vectors and then use the random coding vectors to combine the captured packets [*pin, pin,* · · · *, pin*] where *gs* denotes the generation size

1

2

*gs*

as described in Section 2.. If not specified, the finite field used for the combination in the following experiments is *GF* (24). We assume that the adversary is able to

capture a *strict subset* of the combined packets belonging to the same the generation [*pout, pout,* · · · *, pout*]. In other words, the number of combined packets belonging to the

1

2

*gs*

same generation captured by the adversary is less than the generation size *gs*. The assumption is necessary because if the adversary can capture all the packets belonging to the same generation, the adversary can recover the original packets as a normal receiving node. The assumption is valid when the adversary does not have access to all the outgoing links from the node for eavesdropping.

We can also view the node in Figure 2 as a *super-node*, representing a network of RLNC nodes. As described in Section 2, nodes in an RLNC network generate coding vectors randomly and independently. After the coding process at each intermediate RLNC node, the combined packets can be written as shown in Equation 2.1 where the matrix containing the overall coding vectors denoted as **C**. The coding vectors are updated by each intermediate RLNC node according to locally generated coding vectors. So the RLNC network resulting the matrix **C** is equivalent to a super- node which generates the coding vectors in **C** directly. In the super-node case, the assumption described above is valid when the adversary does not have access to all the RLNC nodes for eavesdropping.

### Performance Metrics

We evaluate performance of the proposed attacks with the following metrics:

*Declared Packet Recovery Rate:* This rate is defined as the percentage of pack- ets that are declared as recovered by the proposed attack, among all the original packets. A packet can be declared as recovered according to one or multiple patterns shown in Table I.

*Actual Packet Recovery Rate:* We defined the actual packet recovery rate as the percentage of packets that are successfully recovered by the proposed attacks, among all the original packets. A packet is determined as successfully recovered when every byte of the recovered packet is exactly the same as the corresponding byte in an original packet. Since an adversary does not have access to the original packets, the adversary can only declare packets as recovered based on the patterns shown in Table

I. Usually the declared packet recovery rate is higher than the actual packet recovery rate as some recovered packets that may not have every byte recovered successfully may still carry some patterns shown in Table I. The difference between the declared packet recovery rate and the actual recovery rate indicates the adversary’s confidence on packets declared as recovered by the proposed attack.

*Byte Recovery Rate:* We defined the rate as the percentage of bytes in a re- covered packet that are successfully recovered by the proposed attack. To obtain the byte recovery rate, we first compare bytes in the recovered packet with bytes of each original packet to calculate the percentage of recovered bytes. The highest percentage among all the calculated percentages is determined as the byte recovery rate. The comparison with each original packet is needed because the BSS algorithm may gen- erate a separated packet which can be correlated to an aggregate of more than one original packets or correlated to multiple original packets in different portions of the packet.

The results of the following experiments are averaged from at least 20,000 attack attempts.

### Packet Length

We design the set of experiments to evaluate the effect of packet length on the attack performance. In this set of experiments, we assume that the generation

0.04

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  | |
|  |  |  | |  |  | |
|  | Decl | ared Packet | Recovery Rate |  |
|  |  |  | (patt  Decl (patt | erns: CK, pa  ared Packet erns: CK, pa | tternc: None)  Recovery Rate tternc: NF) |  |
|  |  |  | Actu  Actu | al Packet Re al Byte Reco | covery Rate very Rate |  |
|  |  |  | |  |  | |
|  |  |  | |  |  | |
|  |  |  | |  |  | |

0.035

0.03

Recovery Rate

0.025

0.02

0.015

0.01

0.005

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 3: Effect of Packet Length (Duplication Method, *gs* = 3, *nr* = 2)

size is three, i.e., *gs* = 3 and the adversary receives only two RLNC packets of the same generation,i.e., *nr* = 2 where *nr* denotes the number of RLNC packets of the same generation that are collected by the adversary. The original packets are web browsing packets collected on a campus Ethernet network. To evaluate the effect of packet length, we truncate original packets of 1500 bytes into different lengths to generate the original packets. The truncation is used in the following experiments that are focusing only one type of packets.

Figure 3 and Figure 4 show the attack performance with the duplication method and reduction respectively. We use *pattens* and *pattenc* to denote the pat- terns used in the scaling step and the cross-checking step respectively. The pattern abbreviations can be found in Table I. The results in both Figure 3 and Figure 4 indicate that both methods can achieve around 1% actual packet recovery rates. In other words, both methods can successfully recover about 1% of original packets. The actual packet recovery rates are not very high mainly because of the degradation in separation performance on the overcomplete problem as described in Section 4. Re- sults in both figures show the gap of about 3% between declared packet recovery rates

0.055

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  | |
|  |  |  | |  |  | |
|  |  |  | Dec  (pat | lared Packet  terns: CK, pa | Recovery Rate  tternc: None) |  |
|  |  |  | Dec (pat | lared Packet terns: CK, pa | Recovery Rate tternc: NF) |  |
|  |  |  | Actu  Actu | al Packet Re al Byte Reco | covery Rate very Rate |  |
|  |  |  | |  |  | |
|  |  |  | |  |  | |
|  |  |  | |  |  | |
|  |  |  | |  |  | |

0.05

0.045

0.04

Recovery Rate

0.035

0.03

0.025

0.02

0.015

0.01

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 4: Effect of Packet Length (Reduction Method, *gs* = 3, *nr* = 2)

0.26



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

0.25

Actual Byte Recovery Rate

0.24

0.23

0.22

0.21

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 5: Actual Byte Recovery Rate for Declared Packets (Duplication Method,

*gs* = 3, *nr* = 2)

0.275

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

0.27



0.265

Actual Byte Recovery Rate

0.26

0.255

0.25

0.245

0.24

0.235

0.23

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 6: Actual Byte Recovery Rate for Declared Packets (Reduction Method, *gs* = 3, *nr* = 2)

and corresponding actual packet recovery rates. The gap indicates the adversary’s confidence on packets recovered by the proposed attack. Figure 3 and Figure 4 also show that the cross-checking with pattern on fragmentation, i.e., the *NF* pattern as show in Table I can reduce the gap slightly i.e., increase the confidence slightly.

We can also observe that the byte recovery rate decreases when the packet length increases. We believe the trend is because of the similarity between original packets, defined as percentage of same bytes when comparing two original packets. As shown in Figure 7, similarity decreases when packet length increases. The decrease is because header bytes of original packets are very similar and payload data bytes can be very different for different original packets. When the packet length increases, the percentage of header bytes also decreases. In turn, the similarity between original packets also decreases. The actual byte recovery rate decreases with the similarity because separation on the same bytes is more likely to be successful. Our results show that the actual byte recovery rate on header bytes actually are around 2% higher than payload data bytes.

0.07



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

0.06

0.05

Similarity

0.04

0.03

0.02

0.01

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 7: Similarity (Packet length: 1500 bytes, *gs* = 3, *nr* = 2)

Figure 5 and Figure 6 shows the actual byte recovery rates for only packets that are declared recovered. Both figures show much higher byte recovery rates for declared packets than those for all packets as show in Figure 3 and Figure 4. The results mean that the adversary can possibly recover over 20% of bytes from the declared packets. The results also shows the actual byte recovery rate decreases with the similarity.

The results shown in the previous figures are averaged over 10,000 repetitions. But actual packet recovery rate fluctuates because: (a) The actual packet recovery rate is close to 1%, which is relatively small. (b) When the length of packet increases, more data available to the BSS algorithm can possibly lead to better separation performance. However when the packet length increases, the chance of having at least one error byte increases.

To further verify the relationship between the similarity and actual byte re- covery rate, we collect various types of packets as shown in Table II. As shown in Figure 8, shows similarity of different types of packets. Similarity in Figure 8 does not

0.4



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

0.35

0.3

0.25

Similarity

0.2

0.15

0.1

0.05

0

162

557

900

1086

1320

1500

2948

Packet Length (Bytes)

Figure 8: Similarity (Different Types of Packets, *gs* = 3, *nr* = 2)

0.35

Declared Packet Recovery Rate (patterns: CK, patternc: None) Declared Packet Recovery Rate (patterns: CK, patternc: NF) Actual Packet Recovery Rate Actual Byte Recovery Rate

0.3

0.25

Recovery Rate

0.2

0.15

0.1

0.05

0

162

557

900

1086

1320

1500

2948

Packet Length (Bytes)

Figure 9: Effect of Packet Length. (Duplication Method, *gs* = 3, *nr* = 2, *patterns* :

*CK, patternc* : *NF* )

0.4



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

0.35

Actual Byte Recovery Rate

0.3

0.25

0.2

0.15

162

557

900

1086

1320

1500

2948

Packet Length (Bytes)

Figure 10: Actual Byte Recovery Rate for Declared Packets (Duplication Method,

*gs* = 3, *nr* = 2)

decrease monotonically as shown in Figure 7 because similarity of different transport layer and application layer protocol headers in different types of packets can vary. Even the payload data bytes of these different types of packets can have different similarity.

Figure 9 shows the performance of the proposed attack on different types of packets. The strong correlation between the similarity shown in Figure 8 and the actual byte recovery rate shown in Figure 9 indicates that higher similarity can lead to higher actual byte recovery rate. Figure 9 also shows: (1) The actual packet recovery rates for different types of packets are around 1%. (2) The gap between the actual packet recovery rate and the declared packet recovery rates, indicating the adversary’s confidence on packets recovered by the proposed attack, is around 4%. Similar correlation between the similarity and the actual byte recovery rate can also be observed from Figure 10, which focuses on packets that are declared as recovered only.

0.045



0.04

0.035

0.03

Recovery Rate

0.025

lared Packet Recovery Rate tterns: CK, patternc: None) lared Packet Recovery Rate tterns: CK, patternc: NF)

ual Packet Recovery Rate ual Byte Recovery Rate

Act

Act

Dec

(pa

Dec

(pa

0.02

0.015

0.01

0.005

400 600 800 1000 1200

Packet Length (Bytes)

Figure 11: Ethernet Packets (*gs* = 3, *nr* = 2, Duplication Method)

### Comparison on Different Packet Types

For fair comparison on the packet types, we truncated packets of different types into the various fixed lengths to generate original packets.

### Ethernet Packets vs. WIFI Packets

Figure 11 and Figure 12 show the attack performance on Ethernet and WIFI packets respectively. Figure 13 shows the comparison of the attack performance on the two types of packets. For both Ethernet and WIFI packets, the actual packet recovery rate can achieve around 0.8% and 0.9% respectively. The results also show that the gaps between actual packet recovery rates and the corresponding declared packet recovery rates (*patterns* = *CK*, *patternc* = *NF* ) are smaller for Ethernet packets. In other words, the adversary can be more confident on Ethernet packets recovered from the proposed attack. The observations are based on 2 0,000 repetitions of experiments on the packets that we collected from Browsing Web. We plan to conduct further larger-scale experiments on typical Ethernet and WIFI packets. The similarity for

0.05

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | |  |  | |
|  |  | |  |  | |
|  |  | De (pa | clared Packet Re  tterns: CK, patte | covery Rate  rnc: None) |  |
|  |  | De (pa | clared Packet Re  tterns: CK, patte | covery Rate  rnc: NF) |  |
|  |  | Ac  Ac | tual Packet Reco tual Byte Recove | very Rate ry Rate |  |
|  |  | |  |  | |
|  |  | |  |  | |
|  |  | |  |  | |
|  |  | |  |  | |

0.045



0.04

0.035

Recovery Rate

0.03

0.025

0.02

0.015

0.01

0.005

400 600 800 1000 1200

Packet Length (Bytes)

Figure 12: WIFI Packets (*gs* = 3, *nr* = 2, Duplication Method)

0.04



0.035

0.03

Recovery Rate

0.025

0.02

Actual Packet Recovery Rate (Ethernet Packets) Actual Pac

Rate (WIFI Actual Byte Rate (Ethe Actual Byte Rate (WIFI Gap (Ethe Gap (WIFI

rnet Packets) Packets)

rnet Packets)

Recovery Packets)

ket Recovery

Packets) Recovery

0.015

0.01

0.005

400 600 800 1000 1200

Packet Length (Bytes)

Figure 13: Comparison between Ethernet and WIFI Packets (*gs* = 3, *nr* = 2, Dupli- cation Method, *patterns* : *CK, patternc* : *NF* )

0.05



0.045

0.04

Similarity

0.035

0.03

0.025

400 600 800 1000 1200

Packet Length (Bytes)

Figure 14: Similarity (Ethernet Packets, *gs* = 3, *nr* = 2)

0.07



0.06

0.05

Similarity

0.04

0.03

0.02

400 600 800 1000 1200

Packet Length (Bytes)

Figure 15: Similarity (WIFI Packets, *gs* = 3, *nr* = 2)

0.355



0.35

Actual Byte Recovery Rate

0.345

0.34

0.335

0.33

400 600 800 1000 1200

Packet Length (Bytes)

Figure 16: Actual Byte Recovery Rate for Declared Packets (Ethernet Packets, Du- plication Method, *gs* = 3, *nr* = 2)

0.23



0.22

Actual Byte Recovery Rate

0.21

0.2

0.19

0.18

400 600 800 1000 1200

Packet Length (Bytes)

Figure 17: Actual Byte Recovery Rate for Declared Packets (WIFI Packets, Dupli- cation Method, *gs* = 3, *nr* = 2)

0.045



0.04

0.035

0.03

Recovery Rate

0.025

Declared Packet Recovery Rate (patterns: CK, patternc: None) Declared Packet Recovery Rate (patterns: CK, patternc: NF)

Actual Packet Recovery Rate Actual Byte Recovery Rate

0.02

0.015

0.01

0.005

0

400 600 800 1000 1200

Packet Length (Bytes)

Figure 18: Skype Video Call Packets (*gs* = 3, *nr* = 2, Duplication Method)

Ethernet and WIFI packets are shown in Figure 14 and Figure 15. Corresponding to similarity Figure 16 and Figure 17 shows actual byte recovery rates of Ethernet and WIFI packets that are declared recovered respectively.

### Comparison on Applications

Our focus in this set of experiments is on the effect of application data on attack performance. We collect voice, video, and file downloading packets generated by Skype voice calls, Skype video calls, and Windows operating system ISO file downloading respectively as original packets.

Figure 18, Figure 19, and Figure 20 show the attack performance on packets generated by different applications. Figure 21 compares the performance on packets generated by different applications. The actual packet recovery rates for all the three applications are between 0.4% and 0.9%. The actual byte recovery rates for the three types of applications all decrease when the packet length increases. The decrease is largely because of the decrease in similarity when the packet length increases as

0.045

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | |  |  | |
|  |  | |  |  | |
|  | De | clared Packet Re | covery Rate |  |
|  |  | (p  De | atterns: CK, patt  clared Packet Re | ernc: None)  covery Rate |  |
|  |  | (p  Ac | atterns: CK, patt  tual Packet Rec | ernc: NF)  overy Rate |  |
|  |  | Ac | tual Byte Recov | ery Rate |  |
|  | |  |  | |
|  |  | |  |  | |
|  |  | |  |  | |
|  |  | |  |  | |

0.04

0.035

0.03

Recovery Rate

0.025

0.02

0.015

0.01



0.005

400 600 800 1000 1200

Packet Length (Bytes)

Figure 19: Skype Voice Call Packets (*gs* = 3, *nr* = 2, Duplication Method)

0.05

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | |  |  | |
|  |  | |  |  | |
|  |  | |  |  | |
|  | De | clared Packet Re | covery Rate |  |
|  |  | (p  De | atterns: CK, patte  clared Packet Re | rnc: None)  covery Rate |  |
|  |  | (p  Ac | atterns: CK, patte  tual Packet Reco | rnc: NF)  very Rate |  |
|  |  | Ac | tual Byte Recove | ry Rate |  |
|  | |  |  | |
|  |  | |  |  | |
|  |  | |  |  | |
|  |  | |  |  | |

0.045

0.04



0.035

Recovery Rate

0.03

0.025

0.02

0.015

0.01

0.005

400 600 800 1000 1200

Packet Length (Bytes)

Figure 20: File Downloading Packets (*gs* = 3, *nr* = 2, Duplication Method)

0.04

0.035

0.03

0.025

Recovery Rate

0.02

0.015

Actual Packet Recovery Rate (Video Packets)

Actual Packet Recovery Rate (Voice Packets)

Actual Packet Recovery

Rate (File Downloading Packets)

Actual Byte Recovery Rate (Video Packets)

Actual Byte Recovery Rate (Voice Packets)

0.01

0.005

0

400 600 800 1000 1200

Packet Length (Bytes)

Actual Byte Recovery

Rate (File Downloading Packets) Gap (Video Packets)

Gap (Voice Packets)

Gap (File Downloading Packets)

Figure 21: Comparison between Skype Video and Voice calls and File Downloading Packets (*gs* = 3, *nr* = 2, Duplication Method, *patterns* : *CK*, *patternc* : *NF* )

0.08



0.07

0.06

0.05

Similarity

0.04

0.03

0.02

400 600 800 1000 1200

Packet Length (Bytes)

Figure 22: Similarity (Video packets, *gs* = 3, *nr* = 2)

0.055



0.045

0.035

Similarity

0.025

0.015

400 600 800 1000 1200

Packet Length (Bytes)

Figure 23: Similarity (Voice packets, *gs* = 3, *nr* = 2)

0.08



0.07

0.06

0.05

Similarity

0.04

0.03

0.02

400 600 800 1000 1200

Packet Length (Bytes)

Figure 24: Similarity (File Downloading Packets, *gs* = 3, *nr* = 2)

0.44



0.435

0.43

Actual Byte Recovery Rate

0.425

0.42

0.415

0.41

0.405

0.4

400 600 800 1000 1200

Packet Length (Bytes)

Figure 25: Actual Byte Recovery Rate for Declared Packets (Video Packets, Dupli- cation Method, *gs* = 3, *nr* = 2)

0.435



0.43

0.425

Actual Byte Recovery Rate

0.42

0.415

0.41

0.405

0.4

0.395

400 600 800 1000 1200

Packet Length (Bytes)

Figure 26: Actual Byte Recovery Rate for Declared Packets (Voice Packets, Duplica- tion Method, *gs* = 3, *nr* = 2)

0.032



0.03

0.028

Actual Byte Recovery Rate

0.026

0.024

0.022

0.02

0.018

400 600 800 1000 1200

Packet Length (Bytes)

Figure 27: Actual Byte Recovery Rate for Declared Packets (File Downloading Pack- ets, Duplication Method, *gs* = 3, *nr* = 2)

shown in Figure 22, Figure 23, and Figure 24.

The major differences among the three types of applications are on the gaps between actual packet recovery rates and the corresponding declared packet recovery rates (*patterns* = *CK*, *patternc* = *NF* ). As shown in Figure 21, the adversary’s confidence on video packets and file downloading packets is the highest and the lowest among the three types of applications respectively. Figure 25, Figure 26 and Figure 27 shows the rate of Actual bytes recovered for packets that are declared correct for video, voice and file downloading packets decreases with decrease in similarity.

### Check Patterns

In this set of experiments, we varies the order of patterns used in the scaling step and the cross-checking step. The packets used in this set of experiments are TCP packets as shown in Table II. We are focusing on TCP packets because the patterns on LR and UP listed in Table II are on TCP protocol headers. It also means that

0.35



(patterns: CK, patternc: None) Declared Packet Recovery Rate (patterns: CK, patternc: NF) Declared Packet Recovery Rate (patterns: CK, patternc: NF,LR) Declared Packet Recovery Rate (patterns: CK, patternc: NF,LR,UP) Actual Packet Recovery Rate Actual Byte Recovery Rate

red Packet Recovery Rate

cla

De

0.3

0.25

Recovery Rate

0.2

0.15

0.1

0.05

0

162

557

900

1086

1320

1500

2948

Packet Length (Bytes)

Figure 28: *patterns* : *CK* (*gs* = 3, *nr* = 2, Duplication Method)

the adversary is assumed to know that the original packets are TCP packets in this set of experiments.

Figure 28, Figure 29, Figure 30 and Figure 31 show attack performance with different combinations of scaling patterns and cross-checking patterns using Duplica- tion method. The results from these figures show that the CK pattern, as a scaling pattern is significantly more effective in filtering out packets that are not successfully recovered than the other three patterns. Figure 32, Figure 33, Figure 34 and Figure 35 shows that the actual bytes recovery rate for declared packets.

Figure 36, Figure 37, Figure 38, and Figure 39 show attack performance on WIFI packets with different combinations of scaling patterns and cross-checking pat- terns. The results from these figures also indicate that the CK pattern is much more effective in filtering out packets that are not successful recovered. Figure 40, Figure 41, Figure 42 and Figure 43 shows that the actual bytes recovery rate for declared packets.

Figure 44 and Figure 45 shows the comparison between the actual packet

0.7



Declared Packet Recovery Rate (patterns: NF, patternc: None) Declared Packet Recovery Rate (patterns: NF, patternc: CK) Declared Packet Recovery Rate (patterns: NF, patternc: CK,LR) Declared Packet Recovery Rate (patterns: NF, patternc: CK,LR,UP) Actual Packet Recovery Rate Actual Byte Recovery Rate

0.6

0.5

Recovery Rate

0.4

0.3

0.2

0.1

0.0075

162

557

900

1086

1320

1500

2948

Packet Length (Bytes)

Figure 29: *patterns* : *NF* (*gs* = 3, *nr* = 2, Duplication Method)

0.7

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | |
|  |  |  |  |  |  | |
|  | Decl | ared Packet Recovery Rate |  |
|  |  |  |  | (patt  Decl | erns: LR, patternc: None)  ared Packet Recovery Rate |  |
|  |
|  |  |  |  | Decl (patt | atterns: LR, patternc: CK)  ared Packet Recovery Rate erns: LR, patternc: CK,NF) |  |
|  |
|  |  |  |  | Decl  (patt | ared Packet Recovery Rate  erns: LR, patternc: CK,NF,UP) |  |
|  |
|  |  |  |  | Actu Actu | al Packet Recovery Rate al Byte Recovery Rate |  |
|  |
|  |
|  |  |  | |
|  |  |  |  |  |  | |

0.6

0.5

0.4 (p



Recovery Rate

0.3

0.2

0.1

0.0035

162

557

900

1086

1320

1500

2948

Packet Length (Bytes)

Figure 30: *patterns* : *LR* (*gs* = 3, *nr* = 2, Duplication Method)

1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | |  |  | |
|  |  | Dec | lared Packet Recovery Rate |  |
|  |  |  |  |  | (pat  Dec (pat | terns: UP, patternc: None)  lared Packet Recovery Rate terns: UP, patternc: CK) |  |
|  |
|  |  |  |  |  | Dec  (pat  Dec (pat | lared Packet Recovery Rate  terns: UP, patternc: CK,NF)  lared Packet Recovery Rate tern : UP, pattern : CK,NF,LR) |  |
|  |
|  |
|  |  |  |  |  | Actu Actu | s c  al Packet Recovery Rate al Byte Recovery Rate |  |
|  |
|  |
|  | |  |  | |
|  |  |  |  | |  |  | |

0.8

Recovery Rate

0.6

0.4

0.2



0.0004

162

557

900

1086

1320

1500

2948

Packet Length (Bytes)

Figure 31: *patterns* : *UP* (*gs* = 3, *nr* = 2, Duplication Method)

0.4



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

0.35

Actual Byte Recovery Rate

0.3

0.25

0.2

0.15

162

557

900

1086

1320

1500

2948

Packet Length (Bytes)

Figure 32: Actual Byte Recovery Rate for Declared Packets (*patterns* : *CK*, Dupli- cation Method, *gs* = 3, *nr* = 2)

0.45

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

0.4



Actual Byte Recovery Rate

0.35

0.3

0.25

0.2

0.15

162

557

900

1086

1320

1500

2948

Packet Length (Bytes)

Figure 33: Actual Byte Recovery Rate for Declared Packets (*patterns* : *NF* , Dupli- cation Method, *gs* = 3, *nr* = 2)

0.45

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

0.4



Actual Byte Recovery Rate

0.35

0.3

0.25

0.2

0.15

162

557

900

1086

1320

1500

2948

Packet Length (Bytes)

Figure 34: Actual Byte Recovery Rate for Declared Packets (*patterns* : *LR*, Duplica- tion Method, *gs* = 3, *nr* = 2)

0.45



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

0.4

Actual Byte Recovery Rate

0.35

0.3

0.25

0.2

0.15

0.1

162

557

900

1086

1320

1500

2948

Packet Length (Bytes)

Figure 35: Actual Byte Recovery Rate for Declared Packets (*patterns* : *UP* , Dupli- cation Method, *gs* = 3, *nr* = 2)

0.04

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  | |
|  |  |  | |  |  | |
|  |  |  | |  |  | |
|  | Declar (patter | ed Packet Re ns: CK, patt | covery Rate ernc: None) |  |
|  |  |  | Declar (patter | ed Packet Re ns: CK, patte | covery Rate rnc: NF) |  |
|  |  |  | Declar  (patter  Declar | ed Packet Re  ns: CK, patte ed Packet Re | covery Rate  rnc: NF,LR) covery Rate |  |
|  |  |  | (patter  Actual | ns: CK, patte Packet Reco | rnc: NF,LR,UP)  very Rate |  |
|  |  |  | Actual | Byte Recove | ry Rate |  |
|  | |  |  | |

0.035



0.03

Recovery Rate

0.025

0.02

0.015

0.01

0.005

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 36: *patterns* : *CK* (*gs* = 3, *nr* = 2, Duplication Method)

0.7



0.6

0.5

Recovery Rate

0.4

0.3

0.2

0.1

Declar (pattern

Declar (pattern

ed Packet Rec

s: NF, pattern

ed Packet Rec

s: NF, pattern

ed Packet Rec

s: NF, pattern

overy Rate

c: CK,LR)

overy Rate

c: CK)

overy Rate

c: None)

ed Packet Rec

s: NF, pattern

Declar (pattern

Declar (pattern

overy Rate

c: CK,LR,UP)

Actual Packet Recovery Rate Actual Byte Recovery Rate

0.007

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 37: *patterns* : *NF* (*gs* = 3, *nr* = 2, Duplication Method)

0.7



0.6

0.5

(pattern : LR, pattern : None)

covery Rate

d Packet Re

Declare

s c

Recovery Rate

0.4

0.3

0.2

0.1

Declared Packet Recovery Rate (patterns: LR, patternc: CK) Declared Packet Recovery Rate (patterns: LR, patternc: CK,NF) Declared Packet Recovery Rate (patterns: LR, patternc: CK,NF,UP)

Packet Recovery Rate Byte Recovery Rate

Actual

Actual

0.0035

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 38: *patterns* : *LR* (*gs* = 3, *nr* = 2, Duplication Method)

1



0.8

0.6

Recovery Rate

0.4

Declared Packet Recovery Rate (patterns: UP, patternc: None) Declared Packet Recovery Rate (patterns: UP, patternc: CK) Declared Packet Recovery Rate (patterns: UP, patternc: CK,NF) Declared Packet Recovery Rate (patterns: UP, patternc: CK,NF,LR)

Actual Packet Recovery Rate

ry Rate

Byte Recove

Actual

0.2

0.0004

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 39: *patterns* : *UP* (*gs* = 3, *nr* = 2, Duplication Method)

0.29

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

0.28



Actual Byte Recovery Rate

0.27

0.26

0.25

0.24

0.23

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 40: Actual Byte Recovery Rate for Declared Packets (*patterns* : *CK*, Dupli- cation Method, *gs* = 3, *nr* = 2)

0.275

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

0.27



Actual Byte Recovery Rate

0.265

0.26

0.255

0.25

0.245

0.24

0.235

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 41: Actual Byte Recovery Rate for Declared Packets (*patterns* : *NF* , Dupli- cation Method, *gs* = 3, *nr* = 2)

0.265



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

0.26

Actual Byte Recovery Rate

0.255

0.25

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 42: Actual Byte Recovery Rate for Declared Packets (*patterns* : *LR*, Duplica- tion Method, *gs* = 3, *nr* = 2)

0.3



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

0.28

Actual Byte Recovery Rate

0.26

0.24

0.22

0.2

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 43: Actual Byte Recovery Rate for Declared Packets (*patterns* : *UP* , Dupli- cation Method, *gs* = 3, *nr* = 2)

0.05



Gap (patterns: NF) Gap (patterns: LR)

Gap (patterns: UP)

Actual Packet Recovery Rate (patterns: UP)

Gap (patterns: CK)

Rate (patterns: NF)

Actual Packet Recovery Rate (patterns: LR)

Actual Packet Recovery Rate (patterns: CK)

Actual Packet Recovery

0.04

0.03

Recovery Rate

0.02

0.01

0

162

557

900

1086

1320

1500

2948

Packet Length (Bytes)

Figure 44: Comparison between Pattern used for Cross-checking (*gs* = 3, *nr* = 2, Duplication Method)

0.03



0.025

0.02

Recovery Rate

0.015

0.01

0.005

Actual Packet Recovery Rate (patterns: CK) Actual Packet Recovery Rate (patterns: NF) Actual Packet Recovery Rate (patterns: LR) Actual Packet Recovery Rate (patterns: UP) Gap (patterns: CK)

Gap (patterns: NF) Gap (patterns: LR) Gap (patterns: UP)

0

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 45: Comparison between Pattern used for Cross-checking, Packet length: 1500 bytes (*gs* = 3, *nr* = 2, Duplication Method)

0.3



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  | Decla (patte | red Packet Recovery Rate rns: CK, patternc: None) |  |
|  |  |  |  | Decla (patte | red Packet Recovery Rate rns: CK, patternc: NF) |  |
|  |  |  |  | Decla (patte | red Packet Recovery Rate rns: CK, patternc: NF,LR) |  |
|  |  |  |  | Decla  (patte  Actua | red Packet Recovery Rate  rns: CK, patternc: NF,LR,UP) l Packet Recovery Rate |  |
|  |  |  |  | Actua | l Byte Recovery Rate |  |
|  |  |  | |
|  |  |  |  |  |  | |

0.25

0.2

Recovery Rate

0.15

0.1

0.05

0

162

557

900

1086

1320

1500

2948

Packet Length (Bytes)

Figure 46: *patterns* : *CK* (*gs* = 3, *nr* = 2, Reduction Method)

1



Declared Packet Recovery Rate (patterns: NF, patternc: None) Declared Packet Recovery Rate (patterns: NF, patternc: CK) Declared Packet Recovery Rate (patterns: NF, patternc: CK,LR) Declared Packet Recovery Rate (patterns: NF, patternc: CK,LR,UP) Actual Packet Recovery Rate Actual Byte Recovery Rate

0.8

0.6

Recovery Rate

0.4

0.2

0

162

557

900

1086

1320

1500

2948

Packet Length (Bytes)

Figure 47: *patterns* : *NF* (*gs* = 3, *nr* = 2, Reduction Method)

1

Recovery Rate

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | |
|  | Decl | ared Packet Recovery Rate |  |
|  |  |  |  | (patt  Decl (patt | erns: LR, patternc: None)  ared Packet Recovery Rate ern : LR, pattern : CK) |  |
|  |
|  |
|  |  |  |  | Decl (patt | s c  ared Packet Recovery Rate erns: LR, patternc: CK,NF) |  |
|  |
|  |  |  |  | Decl  (patt  Actu | ared Packet Recovery Rate  erns: LR, patternc: CK,NF,UP) al Packet Recovery Rate |  |
|  |
|  |
|  |  |  |  | Actu | al Byte Recovery Rate |  |
|  |  |  | |

0.8

0.6

0.4

0.2



0.005

162

557

900

1086

1320

1500

2948

Packet Length (Bytes)

Figure 48: *patterns* : *LR* (*gs* = 3, *nr* = 2, Reduction Method)

1



(patterns: UP, patternc: None) Declared Packet Recovery Rate (patterns: UP, patternc: CK) Declared Packet Recovery Rate (patterns: UP, patternc: CK,NF) Declared Packet Recovery Rate (patterns: UP, patternc: CK,NF,LR) Actual Packet Recovery Rate Actual Byte Recovery Rate

ed Packet Recovery Rate

clar

De

0.8

0.6

Recovery Rate

0.4

0.2

0.0005

162

557

900

1086

1320

1500

2948

Packet Length (Bytes)

Figure 49: *patterns* : *UP* (*gs* = 3, *nr* = 2, Reduction Method)

0.45

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

0.4



Actual Byte Recovery Rate

0.35

0.3

0.25

0.2

0.15

162

557

900

1086

1320

1500

2948

Packet Length (Bytes)

Figure 50: Actual Byte Recovery Rate for Declared Packets (*patterns* : *CK*, Reduc- tion Method, *gs* = 3, *nr* = 2)

0.4



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

0.35

Actual Byte Recovery Rate

0.3

0.25

0.2

0.15

162

557

900

1086

1320

1500

2948

Packet Length (Bytes)

Figure 51: Actual Byte Recovery Rate for Declared Packets (*patterns* : *NF* , Reduc- tion Method, *gs* = 3, *nr* = 2)

0.45

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

0.4



Actual Byte Recovery Rate

0.35

0.3

0.25

0.2

0.15

162

557

900

1086

1320

1500

2948

Packet Length (Bytes)

Figure 52: Actual Byte Recovery Rate for Declared Packets (*patterns* : *LR*, Reduc- tion Method, *gs* = 3, *nr* = 2)

0.45



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

0.4

Actual Byte Recovery Rate

0.35

0.3

0.25

0.2

0.15

0.1

162

557

900

1086

1320

1500

2948

Packet Length (Bytes)

Figure 53: Actual Byte Recovery Rate for Declared Packets (*patterns* : *UP* , Reduc- tion Method, *gs* = 3, *nr* = 2)

0.055



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | |
|  |  |  |  |  | |
|  |  | Declare (pattern | d Packet Re  s: CK, patte | covery Rate rnc: None) |  |
|  |  | Declare (pattern | d Packet Re  : CK, patte | covery Rate rn : NF) |  |
|  |  | Declare | s  d Packet Re | c  covery Rate |  |
|  |  | (pattern  Declare | s: CK, patte  d Packet Re | rnc: NF,LR)  covery Rate |  |
|  |  | (pattern  Actual | s: CK, patte  Packet Reco | rnc: NF,LR,UP)  very Rate |  |
|  |  | Actual | Byte Recove | ry Rate |  |
|  |  |  | |
|  |  |  |  |  | |

0.05

0.045

Recovery Rate

0.04

0.035

0.03

0.025

0.02

0.015

0.01

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 54: *patterns* : *CK* (*gs* = 3, *nr* = 2, Reduction Method)

1



0.8

0.6

Recovery Rate

0.4

0.2

Declared Packet Recovery Rate (patterns: NF, patternc: None) Declared Packet Recovery Rate (patterns: NF, patternc: CK) Declared Packet Recovery Rate (patterns: NF, patternc: CK,LR) Declared Packet Recovery Rate (patterns: NF, patternc: CK,LR,UP) Actual Packet Recovery Rate Actual Byte Recovery Rate

0

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 55: *patterns* : *NF* (*gs* = 3, *nr* = 2, Reduction Method)

1



0.8

0.6

Recovery Rate

0.4

0.2

Declared

Packet Rec

overy Rate

(pattern

c: None) overy Rate

c: CK)

s: LR, pattern ed Packet Rec

s: LR, pattern

Declar (pattern

Declared Packet Recovery Rate (patterns: LR, patternc: CK,NF) Declared Packet Recovery Rate (patterns: LR, patternc: CK,NF,UP) Actual Packet Recovery Rate Actual Byte Recovery Rate

0.006

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 56: *patterns* : *LR* (*gs* = 3, *nr* = 2, Reduction Method)

1



0.8

0.6

Recovery Rate

0.4

0.2

(patterns: UP, patternc: None) Declared Packet Recovery Rate (patterns: UP, patternc: CK) Declared Packet Recovery Rate (patterns: UP, patternc: CK,NF) Declared Packet Recovery Rate (patterns: UP, patternc: CK,NF,LR) Actual Packet Recovery Rate Actual Byte Recovery Rate

covery Rate

Packet Re

Declared

0.0006

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 57: *patterns* : *UP* (*gs* = 3, *nr* = 2, Reduction Method)

0.28



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

0.27

Actual Byte Recovery Rate

0.26

0.25

0.24

0.23

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 58: Actual Byte Recovery Rate for Declared Packets (*patterns* : *CK*, Reduc- tion Method, *gs* = 3, *nr* = 2)

0.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

0.29



Actual Byte Recovery Rate

0.28

0.27

0.26

0.25

0.24

0.23

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 59: Actual Byte Recovery Rate for Declared Packets (*patterns* : *NF* , Reduc- tion Method, *gs* = 3, *nr* = 2)

0.3



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

0.295

0.29

Actual Byte Recovery Rate

0.285

0.28

0.275

0.27

0.265

0.26

0.255

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 60: Actual Byte Recovery Rate for Declared Packets (*patterns* : *LR*, Reduc- tion Method, *gs* = 3, *nr* = 2)

0.34

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

0.32



Actual Byte Recovery Rate

0.3

0.28

0.26

0.24

0.22

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 61: Actual Byte Recovery Rate for Declared Packets (*patterns* : *UP* , Reduc- tion Method, *gs* = 3, *nr* = 2)

recovery and the gap curve for all the scaling patterns. Figure 46, Figure 47, Figure 48 and Figure 49 using Reduction Method. Figure 50, Figure 51, Figure 52 and Figure 53 shows that the actual bytes recovery rate for declared packets.

Figure 54, Figure 55, Figure 56, and Figure 57 show attack performance on WIFI packets with different combinations of scaling patterns and cross-checking pat- terns using Reduction Method. The results from these figures also indicate that the CK pattern is much more effective in filtering out packets that are not successful recovered. Figure 58, Figure 59, Figure 60 and Figure 61 shows that the actual bytes recovery rate for declared packets.

### Generation Size

We vary *gs*, the generation size, in this set of experiments. The number of received packets, denoted as *nr*, is set to *gs* − 1 so that the adversary can not simply recover original packets as a normal receiving node in an RLNC network. Both the

0.04



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  | |
|  |  |  | |  |  | |
|  | Decla | red Packet | Recovery Rate |  |
|  |  |  | (patte  Decla | rns: CK, pat  red Packet | ternc: None)  Recovery Rate |  |
|  |  |  | (patte  Actua | rns: CK, pat  l Packet Re | ternc: NF)  covery Rate |  |
|  |  |  | Actua | l Byte Reco | very Rate |  |
|  | |  |  | |
|  |  |  | |  |  | |
|  |  |  | |  |  | |
|  |  |  | |  |  | |

0.035

0.03

0.025

Recovery Rate

0.02

0.015

0.01

0.005

0

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 62: *gs* = 4 (*nr* = 3, Reduction Method, WIFI Packets)

reduction and duplication methods are used in the experiments for separation.

Figure 62, Figure 63, Figure 64 and Figure 65 show the attack performance of different generation size *gs* with the Reduction and Duplication methods respectively. Figure 66 and Figure 67 compares the attack performance of different generation sizes. The comparison shows that the actual packet recovery rate decreases when the generation size increases. This is consistent with intuition because in general, the separation performance can degrade when the number of original signals used to form mixtures is increasing. We can also observe similar decrease in actual byte recovery rates when the generation size increases.

Figure 68, Figure 69 Figure 70 and Figure 71 shows that the actual bytes recovery rate for declared packets.

### Dependence Between Packet Payload

We design this set of experiments to evaluate the effect of dependence be-tween payload of original packets and the attack performance. The experiments are

0.04



0.035

0.03

0.025

Recovery Rate

0.02

0.015

Declared Packet Recovery Rate (patterns: CK, patternc: None) Declared Packet Recovery Rate (patterns: CK, patternc: NF) Actual Packet Recovery Rate Actual Byte Recovery Rate

0.01

0.005

0.0005

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 63: *gs* = 5 (*nr* = 4, Reduction Method, WIFI Packets)

0.035

0.03

0.025

Recovery Rate

0.02

0.015

lared Packet Recovery Rate tterns: CK, patternc: None) lared Packet Recovery Rate tterns: CK, patternc: NF)

ual Packet Recovery Rate ual Byte Recovery Rate

Act

Act

Dec (pa

Dec (pa

0.01

0.005

0

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 64: *gs* = 4 (*nr* = 3, Duplication Method, WIFI Packets)

0.03



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  | |
|  |  |  | Decla (patte | red Packet rns: CK, pa | Recovery Rate tternc: None) |  |
|  |  |  | (patte  Actua | rns: CK, pa l Packet Re | tternc: NF) covery Rate |  |
|  |  |  | Actua | l Byte Reco | very Rate |  |
|  | |  |  | |
|  |  |  | |  |  | |
|  |  |  | |  |  | |

0.025

Declared Packet Recovery Rate

0.02

Recovery Rate

0.015

0.01

0.005

0

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 65: *gs* = 5 (*nr* = 4, Duplication Method, WIFI Packets)

0.04



0.035

0.03

0.025

Recovery Rate

0.02

0.015

Actual Packet Recovery Rate $gs=4$, $nr=$3 Actual Packet Recovery Rate $gs=5$, $nr=$4 Actual Byte Recovery Rate $gs=4$, $nr=$3 Actual Byte Recovery Rate $gs=5$, $nr=$4 Gap ($gs=4$, $nr=3$)

$, $nr=4$)

Gap ($gs=5

0.01

0.005

0.0005

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 66: Comparison between Generation Numbers (Reduction Method, *patterns* :

*CK*, *patternc* : *NF* )

0.035

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | |
| Actual | Packet Recovery |  |
|  |  |  | Rate $g  Actual Rate $g | s=4$, $nr=$3  Packet Recovery s=5$, $nr=$4 |  |
|  |  |  | Actual  Rate $g Actual | Byte Recovery  s=4$, $nr=$3 Byte Recovery |  |
|  |  |  | Rate $g  Gap ($g Gap ($g | s=5$, $nr=$4  s=4$, $nr=3$) s=5$, $nr=4$) |  |
|  |  |  |  |  | |
|  |  |  |  |  | |
|  |  |  |  |  | |

0.03



0.025

Recovery Rate

0.02

0.015

0.01

0.005

0

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 67: Comparison between Generation Numbers (Duplication Method,

*patterns* : *CK*, *patternc* : *NF* )

0.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

0.09



Actual Byte Recovery Rate

0.08

0.07

0.06

0.05

0.04

0.03

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 68: Actual Byte Recovery Rate for Declared Packets, *gs* = 4 (*nr* = 3, Reduc- tion Method, WIFI Packets)

0.06



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

0.055

0.05

Actual Byte Recovery Rate

0.045

0.04

0.035

0.03

0.025

0.02

0.015

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 69: Actual Byte Recovery Rate for Declared Packets, *gs* = 5 (*nr* = 4, Reduc- tion Method, WIFI Packets)

0.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

0.09



Actual Byte Recovery Rate

0.08

0.07

0.06

0.05

0.04

0.03

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 70: Actual Byte Recovery Rate for Declared Packets, *gs* = 4 (*nr* = 3, Dupli- cation Method, WIFI Packets)

0.055



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

0.05

0.045

Actual Byte Recovery Rate

0.04

0.035

0.03

0.025

0.02

0.015

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 71: Actual Byte Recovery Rate for Declared Packets, *gs* = 5 (*nr* = 4, Dupli- cation Method, WIFI Packets)

0.03

0.025

0.02

Recovery Rate

0.015

0.01

0.005

0

Mutual Information

Figure 72: Actual Packet Recovery Rate for Packet size 1500 (*gs* = 3, *nr* = 2, Duplication Method, WIFI Packets)

0.03

0.02

Recovery Rate

0.01

0

Mutual Information

Figure 73: Actual Packet Recovery Rate for Packet size 1200 (*gs* = 3, *nr* = 2, Duplication Method, WIFI Packets)

0.025

0.02

0.015

Recovery Rate

0.01

0.005

0

Mutual Information

Figure 74: Actual Packet Recovery Rate for Packet size 1000 (*gs* = 3, *nr* = 2, Duplication Method, WIFI Packets)

0.025

0.02

Recovery Rate

0.015

0.01

0.005

0

Mutual Information

Figure 75: Actual Packet Recovery Rate for Packet size 800 (*gs* = 3, *nr* = 2, Dupli- cation Method, WIFI Packets)

0.025

0.02

0.015

Recovery Rate

0.01

0.005

0

Mutual Information

Figure 76: Actual Packet Recovery Rate for Packet size 600 (*gs* = 3, *nr* = 2, Dupli- cation Method, WIFI Packets)

0.025

0.02

Recovery Rate

0.015

0.01

0.005

0

Mutual Information

Figure 77: Actual Packet Recovery Rate for Packet size 400 (*gs* = 3, *nr* = 2, Dupli- cation Method, WIFI Packets)

motivated by a general observation from the research on BSS algorithms: More inde- pendence between original signals can lead to better separation performance on signal mixtures. Since packets usually have similar IP and TCP layer headers, we measure the independence only on payload data of different packets. Mutual information, an information-theoretic metric is used to measure the dependence.

Figure 72, Figure 73, Figure 74, Figure 75, Figure 76, Figure 77 shows the attack performance on packets with different dependence between payload of different packets. The results indicates that the attack performance degrades when the mutual information increases, i.e., the dependence increases. The results are consistent with the general observation from the research on BSS algorithms.

### Finite Field Size

We vary the finite field used in the RLNC from *GF* (25) to *GF* (28) in this set of experiment. Figure 78, Figure 79, Figure 80, Figure 81, Figure 82 and Figure 83 shows

0.035



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  | |
|  |  |  | Decla  (patte | red Packet  rns: CK, pat | Recovery Rate  ternc: None) |  |
|  |  |  | Decla  (patte  Actua | red Packet  rns: CK, pat l Packet Re | Recovery Rate  ternc: NF) covery Rate |  |
|  |  |  | Actua | l Byte Reco | very Rate |  |
|  | |  |  | |
|  |  |  | |  |  | |
|  |  |  | |  |  | |
|  |  |  | |  |  | |

0.03

0.025



Recovery Rate

0.02

0.015

0.01

0.005

0

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 78: GF(25) Finite Field (*gs* = 3, *nr* = 2, Reduction Method, WIFI Packets)

0.04



0.035

0.03

red Packet

rns: CK, pat

red Packet

rns: CK, pat

Recovery Rate

ternc: None)

Decla

(patte

0.025

Actua

Actua

Recovery Rate

ternc: NF)

Decla

(patte

Recovery Rate

0.02

0.015

l Packet Recovery Rate l Byte Recovery Rate

0.01

0.005

0

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 79: GF(26) Finite Field (*gs* = 3, *nr* = 2, Reduction Method, WIFI Packets)

0.035



Recovery Rate

red Packet

Decla

0.03

0.025

Recovery Rate

0.02

(patterns: CK, patternc: None) Declared Packet Recovery Rate (patterns: CK, patternc: NF) Actual Packet Recovery Rate Actual Byte Recovery Rate

0.015

0.01

0.005

0.0001

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 80: GF(27) Finite Field (*gs* = 3, *nr* = 2, Reduction Method, WIFI Packets)

0.05

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  | |
|  |  |  | |  |  | |
|  | Declare (pattern | d Packet Re  s: CK, patte | covery Rate rnc: None) |  |
|  |
|  |  |  | Declare (pattern  Actual | d Packet Re  s: CK, patte Packet Reco | covery Rate rnc: NF)  very Rate |  |
|  |
|  |  |  | Actual | Byte Recove | ry Rate |  |
|  |
|  | |  |  | |
|  |  |  | |  |  | |

0.04



Recovery Rate

0.03

0.02

0.01



0.0002

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 81: GF(28) Finite Field (*gs* = 3, *nr* = 2, Reduction Method, WIFI Packets)

0.025



0.02

(pattern : CK, pattern : None)

Recovery Rate

red Packet

Decla

s c

0.015

Recovery Rate

Declared Packet Recovery Rate (patterns: CK, patternc: NF) Actual Packet Recovery Rate Actual Byte Recovery Rate

0.01

0.005

0.0001

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 82: GF(25) Finite Field (*gs* = 3, *nr* = 2, Duplication Method, WIFI Packets)

0.035

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  | |
|  |  |  | |  |  | |
|  | Dec (pa | lared Packet ttern : CK, pa | Recovery Rate ttern : None) |  |
|  |  |  | Dec (pa | s  lared Packet tterns: CK, pa | c  Recovery Rate tternc: NF) |  |
|  |  |  | Act Act | ual Packet Re ual Byte Rec | covery Rate overy Rate |  |
|  | |  |  | |
|  |  |  | |  |  | |
|  |  |  | |  |  | |
|  |  |  | |  |  | |

0.03

0.025

0.02

Recovery Rate

0.015



0.01

0.005

0

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 83: GF(26) Finite Field (*gs* = 3, *nr* = 2, Duplication Method, WIFI Packets)

0.04



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  | |  |
| Actual | | Packet |
|  |  |  | Recover Actual | | y Rate (GF(25)) Packet |
|  |  |  |  | Recover Actual | y Rate (GF(26)) Packet |
|  |  |  |  | Recover  Actual | y Rate (GF(27))  Packet |
|  |
|  |  |  |  | Recover  Gap (GF | y Rate (GF(28))  (25)) |
|  |
|  |  |  |  | Gap ((G  Gap ((G | F(26))  F(27)) |
|  |
|  |  |  | Gap ((G | | F(28)) |
|  | |  |
|  |  |  |  | |  |

 

0.035

0.03

0.025

Recovery Rate

0.02

0.015

0.01

0.005



0

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 84: Comparison between Finite Field Size (Reduction Method, *patterns* : *CK*,

*patternc* : *NF* )

0.035

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  | |
|  |  |  | |  |  | |
|  |  | Actual Pack | et |  |
|  |  |  |  | Recovery Ra Actual Pack | te (GF(25))  et |  |
|  |  |  |  | Recovery Ra | te (GF(26)) |  |
| Gap (GF(2 | 5)) |
|  |
|  |  |
|  |  |  |  | Gap ((GF( |  |  |
|  | |  |  | |
|  |  |  | |  |  | |
|  |  |  | |  |  | |

0.03



0.025

Recovery Rate

0.02

0.015

0.01

0.005

0



400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 85: Comparison between Finite Field Size (Duplication Method, *patterns* :

*CK*, *patternc* : *NF* )

0.33



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

0.32

Actual Byte Recovery Rate

0.31

0.3

0.29

0.28

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 86: GF(25) Actual Byte Recovery Rate for Declared Packets (*gs* = 3, *nr* = 2, Reduction Method, WIFI Packets)

0.075



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

0.07

0.065

Actual Byte Recovery Rate

0.06

0.055

0.05

0.045

0.04

0.035

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 87: GF(26) Actual Byte Recovery Rate for Declared Packets (*gs* = 3, *nr* = 2, Reduction Method, WIFI Packets)

0.065

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

0.06



Actual Byte Recovery Rate

0.055

0.05

0.045

0.04

0.035

0.03

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 88: GF(27) Actual Byte Recovery Rate for Declared Packets (*gs* = 3, *nr* = 2, Reduction Method, WIFI Packets)

0.07



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

0.06

Actual Byte Recovery Rate

0.05

0.04

0.03

0.02

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 89: GF(28) Actual Byte Recovery Rate for Declared Packets (*gs* = 3, *nr* = 2, Reduction Method, WIFI Packets)

0.07

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

0.06



Actual Byte Recovery Rate

0.05

0.04

0.03

0.02

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 90: GF(25) Actual Byte Recovery Rate for Declared Packets (*gs* = 3, *nr* = 2, Duplication Method, WIFI Packets)

0.026

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

0.024



Actual Byte Recovery Rate

0.022

0.02

0.018

0.016

0.014

0.012

400 600 800 1000 1200 1500

Packet Length (Bytes)

Figure 91: GF(26) Actual Byte Recovery Rate for Declared Packets (*gs* = 3, *nr* = 2, Duplication Method, WIFI Packets)

the attack performance on packets with different Finite Field using Reduction and Duplication methods. The results in the figures show that the attack can still recover original packets successfully based on a strict subset of RLNC packets combined in different finite fields. The figures also show that the attack performance degrades when the finite field size increases.

Figure 84 and Figure 85 compares Finite Field Size performance for Reduction and Duplication method. Figure 86, Figure 87, Figure 88, Figure 89, Figure 90, Figure 91 shows that the actual bytes recovery rate for declared packets for Reduction and Duplication methods.

### CHAPTER VI

**DISCUSSION**

In this paper, we assume that the adversary does not have access to all the RLNC packets in one generation because otherwise the adversary is able to recover all the original packets as a receiving node. For RLNC, since each node in the network generates coding vectors randomly and independently, there are possibilities that some RLNC packets received by a receiving node are dependent. When the number of innovative RLNC packets in one generation is less than the generation size *gs*, a normal receiver can not recover original packets based on the received RLNC packets. But in this situation, an adversary is still possible to recover original packets based on the received RLNC packets using the proposed attack.

### CHAPTER VII

**RELATED WORK**

The complex operation on packets, especially the mixing of packets from dif- ferent traffic flows, brings a new set of security challenges on network coding. Lima *et al.* [18] provide a taxonomy of security vulnerabilities of network coding. A number of research challenges such as resistance to Byzantine attacks targeting control traffic or network code are identified for applicability of secure network coding in practical networks.

The challenges brought by the mixing are also studied in [19]. The authors assume that malicious nodes can inject corrupted packets and possibly end up con- taminating all the information reaching a destination. Algorithms with polynomial- time complexity are proposed in [19] to defeat adversaries with different adversarial strengths.

Wiretapping by a wiretapper capable of accessing a subset of links of a network has been studied by many researchers. Cai and Yeung [20] establish a model of a com- munication system over a wiretap network (CSWN). A construction of secure linear network codes to protect information confidentiality from wiretapper is proposed in the paper.

Bhattad and Narayanan [21] extend the wiretap network model proposed in [20]

and focus on the case when the number of independent messages available to the eavesdropper is less than the multicast capacity of the network. The authors show that secure communication is possible without any loss in the rate in the case.

Cao *et al.* [22] develop a secure random linear network scheme on wiretap net- works where an adversary can access only limited number of channels. One-time pad is used in the scheme to protect message confidentiality without decreasing network throughput. Since the scheme is based on random linear network coding, the scheme can has a number of advantages such as uncentralized implementation of network coding and no restriction on encoding vectors.

A threat model focusing on curious nodes instead of wiretapping is proposed by Lima *et al.* [3]. The curious nodes are assumed compliant with communication protocols but curious to learn information from the packets that passes through them. Lima *et al.* develop an algebraic security criterion and find that algebraic security with network coding is topology-dependent. The authors plan to develop secure communication protocols based on random linear network coding as an almost free cipher.

Network coding is also proposed to secure content distribution and cloud stor- age. In [23], network coding is used as a lightweight security mechanism to secure content distribution. More redundant packet transmissions can increase the chance of successful decoding at destination and in the meantime increase the chance of eavesdropping. The trade-off between security and reliability is studied in the paper. Three optimization methods are proposed to in the paper to determine the number of packets transmitted through different paths. Zhao *et al.* [24] propose a signature scheme for network coding to defeat pollution attacks by malicious nodes in content distribution systems. The signatures can be easily checked with negligible overhead. A network coding based secure storage scheme is proposed in [25]. The scheme

is designed to improve performance of a cloud storage system in terms of storage cost and coding processing time. Design guidelines on performance tradeoff among security requirement, storage cost per node, and encoding processing time are suggested in the paper.

### CHAPTER VIII

**CONCLUSION**

In this paper, we propose attacks to compromise RLNC as a cipher for con- fidentiality protection. The attacks are based on the blind source separation (BSS) technique, a statistical signal processing technique designed to recover original sig- nals based on mixtures of original signals. Following the separation step based on the AMERICA BSS algorithm, the scaling step is designed to filter out artifacts gener- ated by BSS algorithms. The cross-checking step can increase confidence on packets recovered by the proposed attacks by checking known patterns on recovered pack- ets. Our extensive experiments on packets collected from the Internet and a campus network shows that the attacks can recover about 1% of original packets.

### BIBLIOGRAPHY

1. R. Ahlswede, Ning Cai, S. . R. Li, and R. W. Yeung, “Network information flow,”

*IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

1. S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, “Xors in the air: Practical wireless network coding,” *IEEE/ACM Transactions on Net- working*, vol. 16, no. 3, pp. 497–510, 2008.
2. L. Lima, M. Medard, and J. Barros, “Random linear network coding: A free cipher?” in *2007 IEEE International Symposium on Information Theory*, 2007, pp. 546–550.
3. P. A. Chou, Y. Wu, and K. Jain, “Practical network coding,” in *Allerton Confer- ence on Communication, Control, and Computing*, October 2003, invited paper.
4. T. Ho, M. Medard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, “A random linear network coding approach to multicast,” *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.
5. ——, “A random linear network coding approach to multicast,” *IEEE Transac- tions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.
6. T. Ho, R. Koetter, M. Medard, D. R. Karger, and M. Effros, “The benefits of coding over routing in a randomized setting,” in *IEEE International Symposium on Information Theory, 2003. Proceedings.*, 2003, pp. 442–.
7. M. Gaeta and J.-L. Lacoume, “Source separation without prior knowledge: The maximum likelihood solution,” *Proc. EUSIPCO*, pp. 621–624, 01 1992.
8. A. Hyvarinen, “Fast and robust fixed-point algorithms for independent com- ponent analysis,” *IEEE Transactions on Neural Networks*, vol. 10, no. 3, pp. 626–634, 1999.
9. A. Hyv¨arinen and E. Oja, “A fast fixed-point algorithm for independent compo- nent analysis,” *Neural Computation*, vol. 9, no. 7, pp. 1483–1492, 1997.
10. P. Comon, “Independent component analysis, a new concept?” *Signal Processing*, vol. 36, no. 3, pp. 287 – 314, 1994, higher Order Statistics. [Online].

Available: <http://www.sciencedirect.com/science/article/pii/0165168494900299>

1. A. Yeredor, “Ica in boolean xor mixtures,” in *Independent Component Analysis and Signal Separation*, M. E. Davies, C. J. James, S. A. Abdallah, and M. D. Plumbley, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.
2. Zhenya He, Luxi Yang, Ju Liu, Ziyi Lu, Chen He, and Yuhui Shi, “Blind source separation using clustering-based multivariate density estimation algorithm,” *IEEE Transactions on Signal Processing*, vol. 48, no. 2, pp. 575–579, 2000.
3. L. Tong, R. . Liu, V. C. Soon, and Y. . Huang, “Indeterminacy and identifiability of blind identification,” *IEEE Transactions on Circuits and Systems*, vol. 38, no. 5, pp. 499–509, 1991.
4. A. Yeredor, “Independent component analysis over galois fields of prime order,”

*IEEE Transactions on Information Theory*, vol. 57, no. 8, pp. 5342–5359, 2011.

1. H. W. Gutch, P. Gruber, A. Yeredor, and F. J. Theis, “Ica over finite fields—separability and algorithms,” *Signal Processing*, vol. 92, no. 8, pp. 1796– 1808, 2012, latent Variable Analysis and Signal Separation. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0165168411003513>
2. I. Nemoianu, C. Greco, M. Castella, B. Pesquet-Popescu, and M. Cagnazzo, “On a practical approach to source separation over finite fields for network coding applications,” in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 1335–1339.
3. L. Lima, J. Vilela, P. F. Oliveira, and J. Barros, “Network coding security: Attacks and countermeasures,” *ArXiv*, vol. abs/0809.1366, 2008.
4. S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Medard, “Resilient network coding in the presence of byzantine adversaries,” in *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, 2007, pp. 616–624.
5. N. Cai and R. Yeung, “Secure network coding,” in *Proceedings IEEE Interna- tional Symposium on Information Theory*, 2002.
6. K. Bhattad and K. Narayanan, “Weakly secure network coding,” 2005.
7. Z. Cao, S. Zhang, X. Ji, and L. Zhang, “Secure random linear network coding on a wiretap network,” *AEU - International Journal of Electronics and Communications*, vol. 69, no. 1, pp. 467–472, 2015. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1434841114002994>
8. P. Ostovari and J. Wu, “Fault-tolerant and secure data transmission using ran- dom linear network coding,” in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, 2017, pp. 1–9.
9. F. Zhao, T. Kalker, M. Medard, and K. J. Han, “Signatures for content distri- bution with network coding,” in *2007 IEEE International Symposium on Infor- mation Theory*, 2007, pp. 556–560.
10. Y.-J. Chen and L.-C. Wang, “An overflow problem in network coding for secure cloud,” *IEEE Transactions on Parallel and Distributed Systems Storage*, vol. 30, no. 4, pp. 789–799, 2019.