Cryptanalysis of Stream Ciphers

By

Santu Pal

Enrolment Number: MATH11201604005

National Institute of Science Education and Research, Bhubaneswar
Odisha - 752050, India

A thesis submitted to the
Board of Studies in School of Mathematical Sciences

In partial fulfillment of requirements
for the Degree

of

DOCTOR OF PHILOSOPHY

of

HOMI BHABHA NATIONAL INSTITUTE

June, 2021

Homi Bhabha National Institute

Recommendations of the Viva Voce Committee

As members of the Viva Voce Committee, we certify that we have read the dissertation prepared by **Santu Pal** entitled **"Cryptanalysis of stream ciphers"** and recommend that it may be accepted as fulfilling the thesis requirement for the award of Degree of Doctor of Philosophy.

Chairman - Dr. Anil Karn	21-10-2021
Guide / Convener - Dr. Deepak Kumar Dalai	21/10/2021
Member 1 - Dr. Kamal Lochan Patra	21-10-2021
Member 2 - Dr. Rishiraj Bhattacharyya	21/10/2021
External Member - Dr. Sourav Mukhopadhyay	21/10/2021

Final approval and acceptance of this thesis is contingent upon the candidate's submission of the final copies of the thesis to HBNI.

I hereby certify that I have read this thesis prepared under my direction and recommend that it may be accepted as fulfilling the thesis requirement.

Date: 21/10/2021

Place: Satna

Dr. Deepak Kumar Dalai

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at Homi Bhabha National Institute (HBNI) and is deposited in the Library to be made available to borrowers under rules of the HBNI.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the Competent Authority of HBNI when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

Santu Pal

DECLARATION

I, Santu Pal, hereby declare that the investigation presented in the thesis has been carried out by me. The work is original and has not been submitted earlier as a whole or in part for a degree / diploma at this or any other Institution / University.

Santu Pal

List of publications arising from the thesis

Published/ Accepted/ Communicated Journal paper:

1. Deepak Kumar Dalai, Subhamoy Maitra, Santu Pal and Dibyendu Roy. " Distinguisher and Non-randomness of Grain-v1 for 112, 114 and 116 initialization rounds with multiple-bit difference in IVs", *IET Information Security* 13.6 (2019), pp. 603-613.

2. Deepak Kumar Dalai, Santu Pal and Santanu Sarkar. " Some Conditional Cube Testers for Grain-128a of Reduced Rounds", *IEEE Transactions on computers*, DOI: 10.1109/TC.2021.3085144 (2021).

3. Deepak Kumar Dalai, Santu Pal and Santanu Sarkar. " A State Bit Recovery Algorithm with TMDTO Attack on Lizard and Grain-128a", *Design, codes and cryptography*, Springer. (Communicated)

LNCS conference proceedings:

1. Deepak Kumar Dalai and Santu Pal. " Recovering Internal States of Grain-v1". In: *Information Security Practice and Experience, ISPEC-2019, Kuala Lumpur, Malaysia*. Vol. 11879. Lecture Notes in Computer Science. Springer, 2019, pp. 325-337.

2. Deepak Kumar Dalai and Santu Pal. " Wip:Degree Evaluation of Grain-v1". In: *Information Systems Security, ICISS-2019, Hyderabad, India*. Vol. 11952. Lecture Notes in Computer Science. Springer, 2019, pp. 239-251.

Santu Pal

Dedicated

to

Grandmother and Parents,

Also teachers, and friends who helped me to come to this point.

ACKNOWLEDGEMENTS

First, I would like to thank my supervisor Dr. Deepak Kumar Dalai, for helping me carry out the research. His constant support helps me to reach this point. I would also like to thank Dr. Santanu Sarkar for discussing some research problems and helping in SAGE programming. Last but not least, I would like to thank Prof. Subhamoy Maitra and Dr. Dibyendu Roy.

Contents

Title page i

SUMMARY xix

List of Figures xxi

List of Tables xxiii

1 Introduction 1

1.1 Cryptography . . . 3

1.1.1 Different kind of cryptographic primitives . . . 3

1.2 Cryptanalysis . . . 8

1.3 Motivation of research . . . 11

1.4 Organization of thesis . . . 12

2 Preliminaries 15

2.1 Mathematical Background . . . 15

2.1.1 Polynomial over Finite field . . . 15

2.1.2 Finite dimensional vector space over finite field . . . 16

2.1.3 System of linear equations . . . 18

2.1.4 Boolean function . . . 20

2.1.5 Probability distributions and hypothesis testing . . . 22

2.2 Cryptographic Background . . . 27

2.2.1 Design of cyptographic scheme 31
2.2.2 Design specification of Grain-v1 33
2.2.3 Design specification of Grain-128 34
2.2.4 Design specification of Grain-128a 36
2.2.5 Design specification of Lizard 37
2.3 Cryptanalysis techniques 40
2.3.1 Tools for cryptanalysis 40
2.3.2 Differential Attack 41
2.3.3 Cube attack 43
2.3.4 Time memory data trade-off(TMDTO) attack 46

3 Conditional Differential Attacks on Grain-v1 of Reduced KSA Rounds 51
3.1 Motivation 51
3.1.1 Our Contributions 52
3.1.2 Organisation 52
3.2 Conditional differential attack on NFSR based stream cipher 54
3.2.1 A randomness test of the difference function $\Delta_a f_K$ 56
3.3 Existing conditional differential attacks 58
3.4 Distinguisher for 112 KSA round of Grain-v1 61
3.4.1 Function reduction method 70
3.5 Distinguisher for 114 KSA round of Grain-v1 71
3.5.1 Non-randomness of Grain-v1 with 116 KSA round with one bit difference in keys 76
3.6 Comparison 79
3.7 Conclusion 79

4 Conditional Cube Testers for Grain-128a of Reduced KSA Rounds 81
4.1 Motivation 81
4.1.1 Our Contributions 82

4.1.2 Organisation 84
4.2 Cube tester by using a cube searching method 85
4.2.1 Cube searching techniques 85
4.2.2 Conditional cube tester 88
4.2.3 Our method 88
4.3 Cube tester for Grain-128a 92
4.3.1 Structure observation of Grain-128a 92
4.3.2 Cube searching for Grain-128a 93
4.3.3 Results for Grain-128a 97
4.4 Cube tester for Grain-128 101
4.4.1 Study for Grain-128 101
4.5 Comparison 103
4.6 Conclusion 105

5 Time-Memory-Data Trade-off (TMDTO) Attack by using state bit recovery attack 107

5.1 Motivation 107
5.1.1 Our Contribution 108
5.1.2 Organisation 110
5.2 Bit recovery of Grain-v1 by observing algebraic structure 111
5.2.1 Analysis of the Non-linear Filter Function of Grain-v1 111
5.2.2 Guess and Determine Strategy 112
5.3 State bit recovery of FSR based stream ciphers 119
5.3.1 General idea of state bit recovery 119
5.3.2 An algorithm for state bit recovery 121
5.3.3 Implementation of Algorithm on ciphers 137
5.4 State Recovery by TMDTO Attack Approach 143
5.4.1 Conditional TMDTO attack 144
5.4.2 Results based on our TMDTO curve equation and criteria . . . 147

5.4.3 TMDTO attack on Lizard . . . 149
5.4.4 TMDTO attack on Grain-128a . . . 153
5.4.5 TMDTO Attack on Grain-v1 . . . 154
5.5 Conclusion . . . 156

6 Degree Evaluation of Grain-v1 159
6.1 Motivation . . . 159
6.1.1 Our Contribution . . . 160
6.1.2 Organization . . . 161
6.2 Degree Evaluation of Grain-v1 . . . 161
6.2.1 Calculating Repeated Bits . . . 162
6.3 Calculating the Degree of Feedback and Output Bits . . . 167
6.4 Conclusion . . . 171

7 Conclusions 173

Appendix A 177

Appendix B 179

SUMMARY

Data security plays a vital role in digitalization due to the rapid increase of science and technology. Cryptology deals with the security of data through the design and analysis of secure algorithms (ciphers). When two parties communicate through an insecure channel, a third party (adversary) may try to know the secret message between the two parties. Cryptanalysis explores the weaknesses in the cipher, which may be repaired by the designer such that the adversary can not use the weakness for its purpose. The messages are encrypted and decrypted by some keys. The stream ciphers are widely used for efficiency, low cost, and simplicity. This thesis explores popular cryptanalysis of well-known stream ciphers like Grain-v1, Grain-128, Grain-128a, and Lizard. We have used differential cryptanalysis, cube tester, and Time-Memory-Data trade-off (TMDTO) attacks to analyze those stream ciphers. The differential attack is a well-known cryptanalysis method for a stream cipher. In previous literature, the attacker used the conditional differential cryptanalysis to attack Grain-v1 using a one-bit difference vector in IV (Initialization Vector). We have proposed a conditional differential attack on Grain-v1 by using two bits difference vector in IV. Using this, we can distinguish Grain-v1 of 112 KSA round from a random sources in a single key setup with a 99% success rate. Further, Grain-v1 of 114 and 116 KSA rounds can be distinguished from a random source in weak key setup with 73% success rate for 2^{78} weak keys and 62% success rate for 2^{75} related keys, respectively. In recent days, the cube tester is a very popular cryptanalysis tool to analyze a stream cipher. Several types of attacks, including cube tester, are used to analyze Grain-128a. We have proposed a conditional cube tester on Grain-128a. But searching for a good cube is a challenging task in cube tester. We have proposed a heuristic method using three strategies, maximum initial zero, maximum last zero, and maximum last α, to find good and smaller dimensional cubes for Grain-128a. A cube tester is designed using the heuristic to distinguish Grain-128a of 191 KSA round in single key setup and 201 KSA round in weak key setup. Then we focused on the TMDTO attack

by using a state bit recovery attack to analyze a lightweight stream cipher Lizard and the ciphers of the Grain family, Grain-v1 and Grain-128a. We can recover some state bits of the cipher by fixing and guessing the remaining state bits in the state bit recovery attack. Then we used the information regarding numbers of recovering, fixing, and guessing bits to design a TMDTO curve. Then we proposed a general algorithm to recover the maximum number of state bits from the same number of keystream bits by fixing fewer state bits of any cipher. For Grain-v1, 33 state bits are recovered by fixing 45 state bits. For Lizard, $10, 11, 12, 13, 14, 15, 16, 17$ state bits are recovered by fixing $9, 10, 11, 12, 14, 18, 22, 25$ state bits respectively. For Grain-128a, 35 and 48 state bits are recovered by fixing 34 and 54 state bits, respectively. We have proposed conditional TMDTO curves for three different conditions $D < T = M$; $T = 2^{-f-2r}D^2, M = D$ and $D = T = M$. Using these, we got some good results for Lizard as (i) $T = M = 2^{54}, D = 2^{46}$; (ii) $T = 2^{52}, M = D = 2^{53}$; (iii) $T = 2^{49.68}, M = D = 2^{54.33}$ and (iv) $T = 2^{42}, M = D = 2^{60}$. For Grain-128a, we got $T = 2^{110}, M = 2^{111}, D = 2^{107}$ and $T = 2^{114}, M = 2^{113}, D = 2^{113}$. Finally, we used Jiao et al.'s TMDTO curve to attack Grain-v1 and got the result as $T = 2^{61}, M = 2^{66}, D = 2^{78}$. Finally, we have focused on calculating the algebraic degrees of the output and update functions of a cipher in a systematic manner. As a result, we claimed that how many iterations are required for a proper mix of NFSR and LFSR bits of Grain-like cipher. Further, we can find the degree of the IV bits (key bits are taken as a constant) to mount a cube attack on a Grain-like cipher. We have proposed an algorithm to find the algebraic degree of NFSR update functions of Grain-v1, which helps us calculating the degrees of LFSR update and output functions. Using the algorithm, we could calculate the degrees of the output and state update functions up to 54 rounds for Grain-v1.

List of Figures

1.1 Data exchange procedure 2
1.2 Classification of cryptology 2
1.3 Private key cryptography 4
1.4 Encryption and Decryption of Block cipher 7
1.5 Public key cryptography 7

2.1 KSA of Grain Family 32
2.2 PRGA of Grain Family 32

3.1 Distinguisher for Grain-v1 with 112 KSA rounds 69

5.1 Pictorial presentation of TMDTO curve of Lizard (Parameters are in $\log_2(L)$ forms, $L = T, M, D$) 153

List of Tables

3.1 Comparison Table (In single key model) 53
3.2 Comparison Table (In weak, related key model) 53
3.3 Differential status of Grain-v1 for round 0 to round 41 66

4.1 Comparison of the number of rounds in different attacks 83
4.2 Comparison with previous attack on Grain-128 84
4.3 Conditions for 1-dimensional cubes from B_α 94
4.4 The 1-dimensional cube(s) satisfying the conditions 95
4.5 The 2-dimensional cube(s) satisfying the conditions 95
4.6 The 3-dimensional cube(s) satisfying the conditions 95
4.7 The 4-dimensional cube(s) satisfying the conditions 96
4.8 The 5-dimensional cube(s) satisfying the conditions 97
4.9 Conditions for different cube 99
4.10 Result of cube testers on Grain-128a in the single key setup 100
4.11 Result of cube testers on Grain-128a in the weak key setup (Scenario I) 100
4.12 Result of cube testers on Grain-128a in the weak key setup (Scenario II) 101
4.13 Cube selection for Grain-128 102
4.14 Attacks on Grain-128 103

5.1 Relations of state bits in Grain-v1 as the subfunctions of h 113
5.2 The state bits involved to calculate the linear feedback bits 115
5.3 The state bits involved to calculate the non-linear feedback bits 115

5.4 Recovery of state bits . . . 117
5.5 Recovery of state bits continued . . . 118
5.6 Comparison of our result with previous results . . . 118
5.7 Intermediate steps of Algorithm 9 (using Note 6) to recover state bits of Lizard . . . 140
5.8 Recovering of 7 to 24 state bits of Lizard . . . 141
5.9 Comparison of number of fixing state bits of Lizard . . . 142
5.10 Intermediate steps of Algorithm 7 (using Note 4) to recover state bits of Grain-128a . . . 143
5.11 Recovering of (up to) 48 state bits of Grain-128a . . . 143
5.12 The parameters for Lizard satisfying $D^2 = T = M$ . . . 149
5.13 The parameters for Lizard satisfying $D'^2 = T'$ . . . 150
5.14 The parameters for Lizard satisfying $T = 2^{-f-2r}D^2, D = M$ . . . 151
5.15 The parameters for Lizard satisfying $D = T = M$ . . . 151
5.16 Comparison with previous result for Lizard . . . 152
5.17 TMDTO attack parameters on Grain-128a . . . 153
5.18 Comparison of our result with previous results . . . 156

6.1 ANF of h function of Grain-v1 at different rounds . . . 163
6.2 Order difference Δ^k of the indices . . . 164
6.3 Table of Common bits according to differences . . . 164
6.4 Table of degrees of $B_k C_{k+j}, i \geq 80$ . . . 165
6.5 Table of Repeated Common bit with difference Trail . . . 166
6.6 Table of differences (i) between terms m_{k+i} and n_k contain a common bit 168
6.7 Table of Calculating Degrees of NFSR bit b_{117} . . . 170
6.8 Table of Degrees of different non-linear functions . . . 171

Chapter 1

Introduction

The modern era can be considered as an era of digitalization due to the heavy dependency of human society on digital communication and rapid development in science and technology. It is effortless to send any information digitally rather than send the item physically. Cashless transactions are one of the best examples of digitalization used everywhere, like shopping malls, banking sectors, online services, etc., rather than carrying cash or checkbook. The world is currently going through the CORONA pandemic, where the virus spreads due to physical contact. Due to this pandemic situation, the meetings, teaching, money transactions are happening through digital communication. Hence, the world is almost entirely dependent on digital communication. Further, the communication between physical devices, i.e., the Internet of Things (IoT), is the evolving technology in smart networking. In the IoT network, the devices communicate digitally very frequently in an open channel. This communication of information can be private to someone. Else another person will access his data unconditionally. Similarly, two or a group of parties can exchange their private information digitally. Therefore, secure data transfer in an open channel is essential in many situations, specifically to the banking sector, the nation's defense organizations, health sector, and government organizations. In Figure 1.1, Alice and Bob are exchanging their secret messages with each other. Another person, Eve, tries to read

their messages.

Figure 1.1: Data exchange procedure

The word *cryptology* comes from the Greek words *kryptos* and *logos*, which means hidden and word, respectively. Cryptology deals with the security of the secret messages transferred between two parties in an open channel. Cryptology is classified into two parts, as in Figure 1.2.

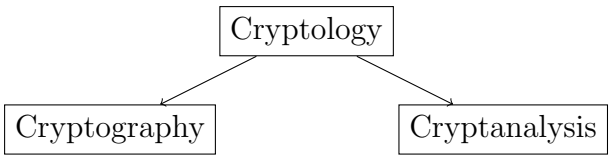

Figure 1.2: Classification of cryptology

The first one, *cryptography*, deals with encoding the original message (or *plaintext*) to the hidden message (or *ciphertext*) called *encryption* and decoding the hidden message to the original message called *decryption*. First, the plaintext is encrypted using public or private (secret) key, and then the ciphertext is decrypted by using a secret key. The ciphertext is transferred through a public channel. The latter, *cryptanalysis*, deals with analyzing the weakness and strength of the encryption algorithm without knowing the secret key. The challenge is to find the secret key with or without using the encryption (or decryption) oracle. The encryption (or decryption) oracle outputs a ciphertext (or plaintext) when one inputs a plaintext (or ciphertext). The leakage of any information about the plaintext or key is also associated with cryptanalysis. The ciphertext transferred in the public channel should look like a random sequence (called a pseudo-random sequence) of a secure cipher. The details of the above two parts are in the following sections.

1.1 Cryptography

During the transmission of messages, the involved parties may require the following criteria:

1. Confidentiality: The most fundamental criterion ensures that the message is secret between the involved parties by encrypting the message.

2. Data integrity: It ensures that any other parties can not modify the message except the authorized parties.

3. Authentication: It guarantees the involved parties' genuineness, i.e., the parties are sure about each other's identity.

4. Non-repudiation: It assures the commitment of both parties. That is, a party can not deny the authenticity and integrity of a message after a transmission.

We will now discuss the cryptographic algorithms by which a message m can be sent from one party A to another party B. Let $\mathcal{M}$ denotes the set of messages or plaintexts, and $\mathcal{C}$ denotes the set of ciphertext. Consider that the set $\mathcal{M} = \mathcal{C} = \{0,1\}^n$, a collection of n-bit strings.

1.1.1 Different kind of cryptographic primitives

1.1.1.1 Unkeyed cryptography

These primitives require no key. An example of such a type of primitives is the hash function which inputs a string of arbitrary length and outputs a fixed-length string. Moreover, the function has a domain with a large size and a range with a fixed size. A good cryptographic hash function should have the following properties:

- **First pre-image resistance:** From any given output value, it is computationally difficult (i.e., infeasible) to get its corresponding input. It is deterministic

and fast to compute the hash value for any given input.

- **Collision resistance:** It is computationally infeasible to find two distinct inputs whose corresponding hash values are the same.
- **Second pre-image resistance:** For a given hash value, finding another input with the same hash value is infeasible.

The hash function is primarily used to resist data integrity as two different input strings can not have the same hash value.

1.1.1.2 Private key cryptography

In this cryptosystem, the same key is used for encryption and decryption, or one key is easily derivable from another. The key needs to be communicated privately. Therefore, the cipher is called a private key or secret key, or symmetric cipher. Figure 1.3 gives the pictorial presentation. Stream ciphers and block ciphers belong to this category. The motivation for stream cipher comes from the encryption scheme

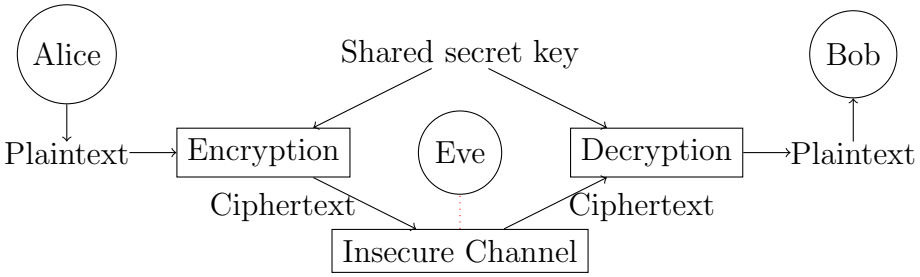

Figure 1.3: Private key cryptography

one-time pad, which Vernam patented in 1917. In 1949, Claude Shannon proved that this scheme is unbreakable [1]. The encryption algorithm of this scheme is $c = p \oplus z$, where c, p and z are the strings of ciphertext, plaintext and key, respectively. Therefore, the decryption algorithm is $p = c \oplus z$. But the major disadvantage of this scheme was that the key size is as long as the plaintext size, and one key is used for only one encryption. The symmetric cipher comes with a solution by replacing the long key

with a long keystream generated from a short key. The keystream generator takes an n-bit secret key as input and outputs a pseudo-random sequence of keystream bits as z. Since the same key always generates the same keystream sequence, the initialization vectors (IV) that the user chooses are used to generate different keystream sequences using one secret key. The literal meaning of stream cipher comes from the fact that it encrypts a continuous stream of characters. In general, the stream ciphers are preferred for fast encryption and low-cost design.
Since stream ciphers are the primary interest of this thesis, we explain the structure of classical stream ciphers. Feedback shift registers (FSR) and Boolean functions are the basic building blocks of most of the keystream generator because of the following properties:

- It is easily implemented into hardware.
- It produces a large period keystream sequence with good statistical property [2].

An FSR of length d is having d cells numbered by $0, 1, \ldots, d-1$ and a recursive relation on the content of the cells. Each cell stores one bit, and the content of the cells at a time is called a state of the FSR. When the FSR is clocked, the FSR outputs the content of cell 0; the remaining cells (i.e., from 1 to $d-1$) are shifted to cell numbered one less (i.e., 0 to $d-2$). The $(d-1)$th cell is updated with the feedback bit of the FSR. The feedback bit is calculated by a recursive relation obtained from the connection polynomial. If the recursive relation is linear, the FSR is called a linear feedback shift register (LFSR); otherwise, it is called a nonlinear feedback shift register (NFSR). The feedback bit and the keystream bit in each clock are Boolean functions that take d bit string from $\{0,1\}^d$ as input and produce one bit as output. The stream ciphers are classified into two categories:

- **Synchronous stream cipher:** In this cipher, the generation of keystream sequence is independent of plaintext and ciphertext.

- **Self-synchronizing stream cipher:** The generation of keystream bits of this cipher depends on the key and previous ciphertext bit.

The cryptanalysis performed on the ciphers in this thesis are synchronous ciphers. **The eSTREAM project [3]:** From November 2004 to April 2008, the eSTREAM project organized by the ECRYPT network was continually searching for new stream ciphers suitable for widespread adoption. Based on the applicability, the ciphers were categorized into two profiles:

- Profile 1: Stream ciphers for software applications with high throughput requirements;
- Profile 2: Stream ciphers for hardware applications with restricted resources such as limited storage, gate count, or power consumption.

The ciphers made up the eSTREAM portfolio are

- Profile 1: Grain-v1 [4], MICKEY 2.0 [5], Trivium [6].
- Profile 2: Salsa 2.0 [7], Sosemanuk [8], HC-128 [9], Rabbit [10].

The increase in popularity of IoT and mobile devices propels cryptographers to design lightweight ciphers for the security in the resource-constrained device, which has limited computing power, storage space, and energy source. As a result, the state size of lightweight ciphers has to be reduced. In 2015, NIST initiated a competition for lightweight cryptographic algorithms [11]. Grain-128AEAD [12], Lizard [13], Fruit [14] and Plantnet [15] are some important lightweight ciphers which are participating in the competition.

A block cipher encrypts a block (a sequence of bits of fixed length) of plaintext and outputs a block of ciphertext of the same size using a key for each block. Here the encryption function (E_K) is invertible. It takes plaintext (P) and key (K) as inputs and outputs corresponding ciphertext, i.e., $C = E_K(P)$. Figure 1.4 shows the encryption process of a block cipher. The decryption function(D_K) returns the

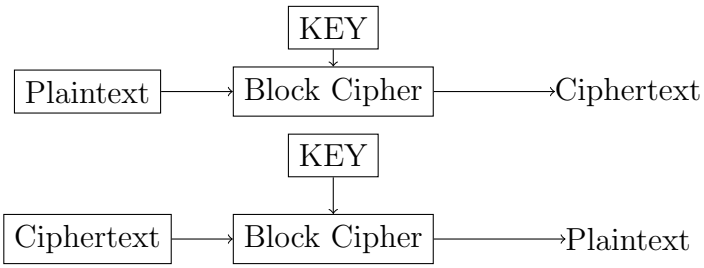

Figure 1.4: Encryption and Decryption of Block cipher

plaintext i.e. $P = D_K(C)$ as it is invertible. The decryption process of block cipher is pictorially presented in Figure 1.4. AES, DES are examples of popular block ciphers.

1.1.1.3 Public key cryptography

The drawback of a symmetric cipher is that the sender and receiver need to share the key secretly, which is solved by the public key cryptosystem. A pair of different keys are used in this cryptosystem. These keys are the public key and the private key of the receiver. Further, computing one key from another key must be computationally hard without knowing a piece of information called a trapdoor. The sender uses the public key for encryption, and the receiver uses the private key for decryption. For data integrity purposes, the sender can use its private key for signature, and the receiver can use the sender's public key for verification. RSA is one of the popular public-key primitives. The pictorial view of such primitive is presented in Figure 1.5.

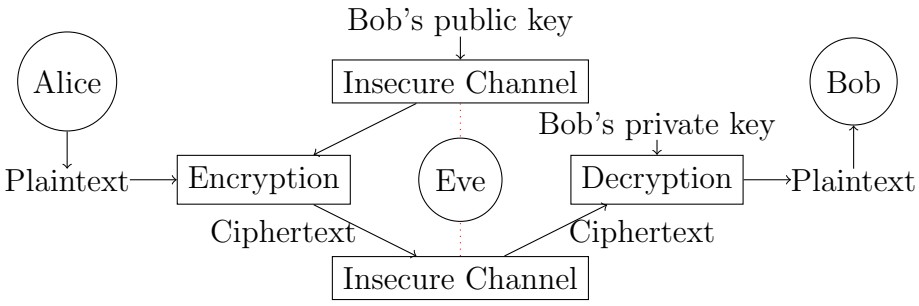

Figure 1.5: Public key cryptography

1.2 Cryptanalysis

Cryptanalysis is related to the security of cryptographic primitives. Cryptanalysis is the study of security aspects of cryptographic primitives. The straightforward approach to secure the message transmission is by hiding the cryptographic primitive from everyone except the involved parties. If any eavesdropper somehow gets the knowledge of primitive and finds out the weakness of the primitive, they can decrypt the ciphertext. Therefore, when someone designs the cryptographic primitives, Kerckhoffs' principle needs to be followed to overcome this scenario. In his two articles on La Cryptographie Militaire [16] in 1883, he stated that the security of a cryptosystem must lie in the choice of its keys only; everything else (including the algorithm itself) should be considered as public knowledge. The cryptosystem must be designed so that no one (even the designer) cannot break the cryptosystem without the secret key. In general, the attacks on different encryption schemes are classified into different categories:

- **The ciphertext only attack:** In this attack, the adversary uses the ciphertext only to deduce the plaintext or the secret key.

- **The known-plaintext attack:** The adversary has some plaintexts and corresponding ciphertexts to deduce the message or the secret key in this attack.

- **The chosen-plaintext attack:** In this attack, the adversary is given more power. He can choose the plaintexts and get their corresponding ciphertexts to deduce the message or the secret key.

- **The chosen-ciphertext attack:** The adversary can access the decryption algorithm that embeds the decryption key for some period. Therefore, the adversary can choose some ciphertexts and have their corresponding plaintexts to deduce the message or the secret key.

The security of different cryptographic primitives is defined under distinct models. Some models are as follows.

1. **Unconditional security:** In this case, the adversary cannot attack the primitives although he has infinite computational resources. A particular case is called perfect secrecy, where one cannot have any information from the ciphertext without having the key. A necessary condition for a symmetric cipher to be unconditionally secure is that the key size is at least as large as the plaintext size. An example of a scheme that is unconditional secure is the one-time pad.

2. **Provable security:** In this case, it can be proved that the adversary has to solve an underlying mathematical computationally hard problem in order to break the security of the cryptographic primitive. For example, solving the integer factorization problem implies breaking the security of the RSA cryptosystem.

3. **Empirical security:** If there is no efficient method rather than an exhaustive search, each key from the whole keyspace is tried to attack the cryptosystem until getting the right key. When designing a stream cipher, an initialization round of the cipher is declared as a security margin, although there is no theoretical proof about the fixed number of initialization rounds. Eavesdropper tries to attack the cipher on reduced initialization rounds. Most of the attacks in this thesis are implemented on reduced initialization rounds.

As our thesis includes cryptanalysis of some stream ciphers, we discuss the attacks on the stream cipher from now onwards. Next, we present the classification of attacks based on the goals of the adversary.

- **Key recovery attack:** This is the strongest attack where the adversary recovers a partial or complete secret key.

- **State recovery attack:** In this case, the adversary recovers a state of the cipher at a particular clock. Then, the adversary can recover the further states

of the cipher and the keystream in forwarding mode. Moreover, if the keystream generation algorithm is invertible, the state recovery leads to a key recovery attack.

- **Distinguishing attack:** This attack is comparatively weaker and easier to implement than the previous two attacks. The adversary tries to distinguish the keystream sequence of a cipher from a random source. In general, the adversary exploits some properties of the Boolean functions (e.g., unbalancedness, the presence of neutral or linear variables, the existence of low or high degree monomials) used in the cipher.

Different techniques are used to analyze the stream ciphers. The attacks can be classified into (a) based on the weakness in the algorithm and (b) based on the weakness in the physical implementation of the cryptographic primitive (e.g., a side-channel attack). As the thesis includes the cryptanalysis of the former type, we now introduce some popular attack methods of the former type.

- **Differential attack:** The adversary exploits the predictability of the keystream bits difference at some rounds on a difference in states of the cipher. Different statistical tests on the keystream bits difference function can distinguish the cipher of reduced initialization rounds from the random function. Sometimes conditions on some state bits (i.e., IV bits and key bits) are imposed to propagate the non-randomness of the keystream difference for larger initialization rounds. Such a type of attack is called a conditional differential attack.

- **Cube attack:** In this attack, a set of cube variables are chosen from the IV bits such that the polynomial generated by adding the output function for all possible cube values (called the superpoly) is a simple polynomial (e.g., linear or quadratic). In the preprocessing phase of the cube attack, the linear (or quadratic) expressions of the superpolies of corresponding cubes are collected using the linearity test (or quadratic test). In the online phase, for a fixed key,

the oracle is queried by setting the values of non-cube IV variables as zero, and the values of the expressions are collected. Finally, a system of equations is solved, and the corresponding key bits are recovered. Also, another variant of cube attack, cube tester, deals with the properties of the superpoly to implement the distinguishing attack. In conditional cube tester or conditional cube attack, conditions on some state bits are imposed.

- **Guess and determine attack:** Some state bits are guessed to make a system of simpler equations on other state variables in this attack. By solving the system of equations, the other state bits are recovered.

- **Time memory data tradeoff (TMDTO) attack:** This attack recovers the state bits of a cipher using a balanced trade-off between time, memory and data. In the preprocessing stage, a table stores the states of keystream sequences. The adversary collects some keystream sequences in the online stage and tries to find the keystream in the preprocessing table. Once it is found, its predecessor state is the correct state from which the keystream is generated. Sometimes conditions on state bits are used, then such an attack is called a conditional TMDTO attack.

The conditions on the state bits lead to two different setups as follows.

- **Single key setup**: A conditional attack where the conditions are imposed only on IV bits.

- **Weak key setup**: A conditional attack where at least one key bit is conditioned.

The detailed presentation of some important attacks is discussed in Chapter 2.

1.3 Motivation of research

Data security is an essential need in our daily lives as we exchange a large amount of data over the internet. Therefore, the world needs good cryptographic primitives to

secure the data. The cryptanalysis of the cryptographic primitive helps to find out the possible weaknesses in the primitives. Symmetric ciphers, especially the stream ciphers, are preferred for fast encryption, low-cost implementation, and resource constraint networks. We have cryptanalyzed some popular and contemporary stream ciphers. The chosen ciphers are candidates in competitions like the eSTREAM project by ECRYPT [3] and the lightweight design competition by NIST [11].

Grain-v1 is one of the famous ciphers in the hardware-based eSTREAM portfolio. The best attack on Grain-v1 was the conditional differential cryptanalysis used to attack the cipher in reduced rounds [17]. We improved the reduced rounds by using the same attack. We, too, implemented two more attacks, i.e., the guess and determine attack and the TMDTO attack on Grain-v1. Since the degree of the output function of Grain-v1 is six, finally, we try to evaluate the degree of Grain-v1 up to some rounds such that we can observe the ANF of the function more carefully.

Grain-128a is another famous cipher that belongs to the same family of Grain-v1. First, we have proposed cube testers on the reduced initialization rounds of Grain-128a. Also we applied the above attack on Grain-128. We have presented a deterministic algorithm to find the state bits that need to be fixed to recover some state bits. Using the algorithm, the state bits of Grain-128a are recovered with fewer fixing bits. Then conditional TMDTO attacks are implemented on Grain-128a.

Recently, lightweight cryptography is in demand for its requirement in resource constraint networks like IoT. Lizard is one of the famous lightweight ciphers. We used our deterministic algorithm to recover state bits of Lizard and implemented conditional TMDTO attacks on Lizard using the obtained results.

1.4 Organization of thesis

The following chapters are added to the thesis:

- **Chapter 1** introduces the cryptanalysis of stream ciphers and the motivation

for the research topics of the thesis.

- **Chapter 2** covers all technical information regarding the research presented in the thesis. In addition, the designs of the stream ciphers used in this thesis are presented, and some important cryptanalysis models on stream cipher are explained in this chapter.

- **Chapter 3** presents a conditional differential attack on Grain-v1. In this attack, distinguishers on Grain-v1 of 112 and 114 KSA rounds are proposed in a single key and weak key setup, respectively. The material of this chapter is taken from the paper [18].

- **Chapter 4** presents conditional cube testers on Grain-128a. First, a new cube searching method is presented. Using a five-dimensional cube obtained from the method, Grain-128a of 191 and 201 KSA rounds are distinguished in single and weak key setups, respectively. This method is also applied to Grain-128. This chapter is based on the paper [19].

- **Chapter 5** is divided into two parts. In the first part, a heuristic method is used to recover state bits of Grain-v1, and then for the first time, a deterministic algorithm is proposed to recover state bits with a minimal number of fixed bits of a stream cipher. The method is used on Grain-128a and Lizard. In the second part, a TMDTO equation is provided to mount conditional TMDTO attacks on the ciphers using the number of recovering and fixing bits of the ciphers. The material of this chapter is taken from the paper [20] and the communicated paper.

- **Chapter 6** presents a method to compute the algebraic degrees of the output and the feedback function of Grain-like ciphers. The method is implemented on Grain-v1 to compute the degree of the output function of Grain-v1 up to 54 rounds. The work is in progress. This chapter is based on the paper [21].

- **Chapter 7** concludes the thesis with a summary of our research works and the future research plan in this direction.

Chapter 2

Preliminaries

2.1 Mathematical Background

2.1.1 Polynomial over Finite field

It is known that for each prime p and each positive integer n there exists a finite field of order p^n uniquely, called the Galois field [22] and denoted by $GF(p^n)$ or $\mathbb{F}_{p^n}$.

Definition 1. *The generator of the multiplicative group $\mathbb{F}^*_{p^n}$ of the finite field $\mathbb{F}_{p^n}$ is called the **primitive element** [22] of the group, which is algebraic over $GF(p)$ of degree n, i.e., it satisfies a non-constant polynomial of degree n over $GF(p)$.*

Let $\mathbb{F}[x]$ be a polynomial ring such that $\mathbb{F}[x] = \{a_0 + a_1x + \cdots + a_nx^n + \cdots | a_i \in \mathbb{F}\}$, where $\mathbb{F}$ is a field.

Definition 2. *Let D be an integral domain. A polynomial $f(x) \in D[x]$ is said to be **irreducible** [23], if*

- *$f(x)$ is neither zero nor unit in $D[x]$.*
- *if $f(x) = g(x)h(x)$, then either $g(x)$ or $h(x)$ is a unit in $D[x]$, where $g(x)$ and $h(x)$ belong to $D[x]$.*

Definition 3. *The **minimal polynomial** [23] of an element α in the extension field over the base field $\mathbb{F}$ is the monic irreducible polynomial in $\mathbb{F}[x]$ such that α is a root.*

Theorem 1. *$\mathbb{F}[x]/\langle f(x)\rangle$ forms a field iff $\langle f(x)\rangle$ is an irreducible polynomial of degree n [24].*

An irreducible polynomial is used to construct a finite field $\mathbb{F}_{p^n}$ of order p^n over the base field $\mathbb{F}_p$ of order p and $\mathbb{F}_{p^n} \cong \mathbb{F}_p[x]/\langle f(x)\rangle$ as fields. It is known that for a polynomial $f(x) \in \mathbb{F}_p[x]$ of degree n with $f(0) \neq 0$, there always exists one positive integer $e \leq p^n - 1$ such that $f(x)$ divides $x^e - 1$ [22].

Definition 4. *For a non-zero polynomial $f(x) \in \mathbb{F}_p[x]$, the order of $f(x)$ is defined for the following two cases [22]:*

1. *If $f(0) \neq 0$, then the least positive integer e for which $f(x)$ divides $x^e - 1$ is said to be the order of $f(x)$.*
2. *If $f(0) = 0$, there exists uniquely $k \in \mathbb{N}$ and $g(x) \in \mathbb{F}_p[x]$ with $g(0) \neq 0$ such that $f(x) = x^k g(x)$, then the order of $f(x)$ is equal to the order of $g(x)$.*

It is known that, for a polynomial $f(x) \in \mathbb{F}_p[x]$ of degree n with $f(0) \neq 0$, the order of $f(x)$ is equal to the order of any element of $\mathbb{F}_{p^n}^*$, which is a root of $f(x)$ [22] [25]. This means that the order of any irreducible polynomial over $\mathbb{F}_p$, divides $p^n - 1$.

Definition 5. *The minimal polynomial of degree n over $\mathbb{F}_p$ of a primitive element of $\mathbb{F}_{p^n}$ is called a **Primitive polynomial** [22] of degree n. In another words, we can say that a monic polynomial $f(x) \in \mathbb{F}_p[x]$ of degree n with $f(0) \neq 0$ is said to be **Primitive**, if the order of $f(x)$ is equal to $p^n - 1$.*

2.1.2 Finite dimensional vector space over finite field

First, we discuss some basic definitions regarding vector spaces [26].

Definition 6. *Let $\mathbb{V}$ be a vector space of the field $\mathbb{F}$. Then,*

1. *A **linear combination** of a set of vectors $v_1, \ldots, v_m \in \mathbb{V}$ is a vector of the form $a_1v_1 + \cdots + a_mv_m$, where $a_i \in \mathbb{F},\ 1 \leq i \leq m$.*

2. *The set of all linear combinations of the vectors $v_1, \ldots, v_m \in \mathbb{V}$ is said to be the **linear span** of the set of $v_1, \ldots, v_m$ and it is denoted by $span(\{v_1, \ldots, v_m\})$.*

3. *A set of vectors $v_1, \ldots, v_m$ in $\mathbb{V}$ is said to be **linearly independent** if $a_1v_1 + \cdots + a_mv_m = 0$ implies that $a_i = 0\ \forall\ i$.*

Definition 7. *Let $\mathbb{V}$ be a vector space of the field $\mathbb{F}$. Then,*

- *The vector space $\mathbb{V}$ is said to be finite-dimensional vector space if there exists a finite subset S of $\mathbb{V}$ such that $span(S) = \mathbb{V}$.*

- *A basis of vector space $\mathbb{V}$ is a set of vectors in $\mathbb{V}$ which is linearly independent and spans $\mathbb{V}$.*

Example 1.

- *Every field is a vector space over it's subfield e.g. $\mathbb{F}_{2^n}$ is a vector space over it's subfileld $\mathbb{F}_2$. $\{1, \alpha, \ldots, \alpha^{n-1}\}$ is a basis of $\mathbb{F}_{2^n}$, where α is a cyclic generator of $\mathbb{F}^*_{2^n}$.*

- *For a fixed n, set of polynomials of degree n or less over a field $\mathbb{F}$, $\mathbb{P}_n(\mathbb{F}) = \{a_0 + a_1x + \cdots + a_nx^n | a_i \in \mathbb{F}\}$ is a vector space over $\mathbb{F}$. $\{1, x, \ldots, x^n\}$ is a basis of $\mathbb{P}_n(\mathbb{F})$.*

Theorem 2. *Let $\mathbb{V}$ be a vector space of the field $\mathbb{F}$. Then,*

- *a set of vectors $v_1, \ldots, v_m$ is said to be basis of $\mathbb{V}$ iff each vector $v \in \mathbb{V}$ can be uniquely written as $v = a_1v_1 + \cdots + a_mv_m,\ a_i \in \mathbb{F}$.*

- *every finite dimensional vector space has a basis.*

- *any two bases of a finite dimensional vector space contain the same number of vectors.*

Definition 8. *The dimension of a finite dimensional vector space* $\mathbb{V}$ *is the cardinality of any basis of the vector space and is denoted by dim* $\mathbb{V}$.

Example 2. *dim* $\mathbb{P}_n(\mathbb{F}_p) = n + 1$.

2.1.3 System of linear equations

Definition 9. *An equation in* n *unknown variables* $x_1, \ldots, x_n$ *with* n *coefficients* $a_1, \ldots, a_n$ *in a field* $\mathbb{F}$ *is said to be* ***linear*** *if it is of the form* $a_1x_1 + \cdots + a_nx_n = b,\ b \in \mathbb{F}$. *An* n*-tuple* $(\alpha_1, \ldots, \alpha_n) \in \mathbb{F}^n$ *for which* $a_1\alpha_1 + \cdots + a_n\alpha_n = b$ *is said to be the* ***solution*** *of the above linear equation.*

Definition 10. *A collection of* m *linear equations in* n *unknown variables* $x_1, \ldots, x_n$ *with field coefficients is said to be a* ***system of*** m ***linear equations*** *in* n *unknowns. It is denoted by*

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \cdots \quad \cdots \quad \cdots \quad \cdots & \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m \end{aligned} \tag{2.1}$$

,where $a_{ij}, b_i \in \mathbb{F},\ 1 \le i \le m, 1 \le j \le n$.

A solution of the above system which simultaneously solves each linear equation, is an n-tuple $(\alpha_1, \ldots, \alpha_n) \in \mathbb{F}^n$. A system of linear equations is said to be solvable if there is a solution of it [27].

Definition 11. *An* $m \times n$ *matrix* M *in* $\mathbb{F}$ *with* m *rows and* n *columns is defined by a collection of elements* $a_{ij} \in \mathbb{F},\ 1 \le i \le m, 1 \le j \le n,\ m, n \in \mathbb{N}$ *such that*

$$M = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

The collection of all row vectors $R_i = (a_{i1}, \ldots, a_{in})$, $1 \leq i \leq m$ *of the matrix* M *forms a vector subspace, called the* $RowSpace(M)$ *of* $\mathbb{F}^n$. *Similarly, the collection of all column vectors* $C_j = (a_{1j}, \ldots, a_{mj})^T$, $1 \leq j \leq n$ *of the matrix* M *forms a vector subspace, called the* $ColSpace(M)$ *of* $\mathbb{F}^m$. *The set of all* $m \times n$ *matrices in* $\mathbb{F}$ *is denoted by* $\mathbb{F}^{m,n}$.

Lemma 1. *For every matrix* $M \in \mathbb{F}^{m,n}$, $dim(RowSpace(M)) = dim(ColSpace(M))$.

Definition 12. *The rank of a matrix* $M \in \mathbb{F}^{m,n}$ *is defined by*

$$rank(M) = dim(RowSpace(M)) = dim(ColSpace(M)).$$

Definition 13. *A matrix* $M \in \mathbb{F}^{n,n}$ *is said to be invertible if there exists a matrix* $N \in \mathbb{F}^{n,n}$ *such that* $MN = NM = I_n$, *the identity matrix.* N *is said to be the inverse of* M, *and it is denoted by* $M^{-1} = N$.

Definition 14. *The system of linear equations are associated with two matrices.*

1. *The matrix of the coefficients,* $A = [a_{ij}] \in \mathbb{F}^{m,n}$, $1 \leq i \leq m, 1 \leq j \leq n$.
2. *The matrix of the inhomogeneous terms,* $B = [b_1, \ldots, b_m]^T \in \mathbb{F}^{m,1}$.

The augmented matrix of the linear system is given by

$$[A : B] = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{array}\right]$$

So the system of linear equations can be represented as

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \text{ or } AX = B, \text{ where } X = [x_1, \ldots, x_n]^T.$$

Theorem 3. *The system of linear equations $AX = B$ is solvable iff $rank(A) = rank([A : B])$. Let $rank(A) = \rho$ and the system has n number of unknowns. Then,*

- *The total number of free unknowns is $n - \rho$.*
- *The selection of $n - \rho$ free unknowns is done in such a way that the remaining ρ unknowns correspond to linearly independent columns of A.*

2.1.4 Boolean function

Let $\mathbb{V}_n$ be a vector space over $\mathbb{F}_2$ of dimension n. The definitions regarding Boolean function [28] are given below:

Definition 15. *A Boolean function f in n variables is a function from $\mathbb{V}_n$ to $\mathbb{F}_2$ i.e. $f : \mathbb{V}_n \to \mathbb{F}_2$. The set of all Boolean functions in n variables is denoted by $\mathbb{B}_n$.*

Definition 16. *Let $f \in \mathbb{B}_n$ and the vectors of $\mathbb{V}_n$ are ordered in lexographically as $v_0 = (0, \ldots, 0), v_1 = (0, \ldots, 1), \ldots, v_{2^n-1} = (1, \ldots, 1)$. Then two sequences of f are defined as*

- *$(0, 1)$-sequence: $(f(v_0), \ldots, f(v_{2^n-1}))$.*
- *$(1, -1)$-sequence: $((-1)^{f(v_0)}, \ldots, (-1)^{f(v_{2^n-1})})$.*

Two important representations of a Boolean function $f \in \mathbb{B}_n$ are

1. **Truth table** representation: The $(0, 1)$-sequence of f.

2. **Algebraic normal form(ANF)** representation: The polynomial form as an element of $\frac{\mathbb{F}_2[x_1,\ldots,x_n]}{(x_1^2-x_1,\ldots,x_n^2-x_n)}$. That is, $f(x_1,\ldots,x_n) = \sum_{v\in\mathbb{V}_n} f_v x_1^{v_1}\cdots x_n^{v_n}$, where $f_v \in \mathbb{F}_2$. The value of f_v can be computed as $f_v = \sum_{x\leq v} f(x) \in \mathbb{F}_2$.

Definition 17. *An affine function $l_{a,c}$ is defined by $l_{a,c} = a.x \oplus c = a_1x_1 \oplus \cdots \oplus a_nx_n \oplus c$, where $a \in \mathbb{V}_n, c \in \mathbb{F}_2$. The set of all affine functions on n-variable is denoted by $\mathbb{A}_n$.*

Definition 18. *Let $f \in \mathbb{B}_n$, then*

- *The **Algebraic degree** of f is defined by the number of variables involved in the highest order monomial with the non-zero coefficient in the ANF of f. It is denoted by* $\deg(f)$.
- *The **complement** of f is defined by $\bar{f} = f \oplus 1$.*
- *The **support** of f is defined by $\Omega_f = \{x \in \mathbb{V}_n | f(x) = 1\}$.*
- *The **Hamming weight** of f is defined by $wt(f) = |\Omega_f|$.*
- *The **Hamming distance** between two Boolean functions f, g is defined by $d(f,g) = wt(f \oplus g)$.*
- *The **Non-linearity** of f is defined by $\mathbb{N}_f = min_{\phi\in\mathbb{A}_n} d(f,\phi)$.*

Definition 19. *A Boolean function f with n variables is called balanced function if $wt(f) = 2^{n-1}$.*

Definition 20. *A sign function or character form of a Boolean function f, $\hat{f} : \mathbb{V}_n \to \mathbb{C}^*$ is defined by $\hat{f}(x) = (-1)^{f(x)}$.*

Definition 21. *The Walsh transformation of a Boolean function f is a function $W(f) : \mathbb{V}_n \to \mathbb{F}$ such that $W(f)(w) = \sum_{x\in\mathbb{V}_n} (-1)^{w.x} f(x)$.*

2.1.5 Probability distributions and hypothesis testing

Definition 22. *Some definitions regarding basic probability theory [29] are given below:*

- ***Sample space:*** *The set of all possible outcomes of an experiment is called the sample space, denoted by* $\mathbb{S}$ *e.g., sample space of tossing a coin twice is* $\mathbb{S} = \{HH, HT, TH, TT\}$

- ***Event:*** *A subset of the sample space is called an event, denoted by* $\mathbb{E}$ *e.g., the event that the last toss is tail is* $\mathbb{E} = \{HT, TT\}$.

- ***Probability:*** *The likelihood of the occurrence of an event containing single sample point from a finite sample space is evaluted by mean of a set of real number, called probability, ranging from* 0 *to* 1.

- ***Probability of an event:*** *The probability of an event* $\mathbb{E}$ *is defined by the sum of the probabilities of all sample points in* $\mathbb{E}$ *and denoted by* $Pr(\mathbb{E})$. *Therefore,* $0 \leq Pr(\mathbb{E}) \leq 1, Pr(\phi) = 0$ *and* $Pr(\mathbb{S}) = 1$. *Furthermore if* A *and* B *are mutually exclusive events , then* $Pr(A \cup B) = Pr(A) + Pr(B)$. *In case of above example,* $Pr(\mathbb{E}) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$.

- ***Random variable:*** *It is a real valued function* $X : \mathbb{S} \to \mathbb{R}$ *that assigns a real number* $X(s)$ *to each outcome* s *in the sample space. A random varible which takes countably many values is called discrete random variable else it is called continuous random variable.*

- ***Probability mass function:*** *Let* X *be a discrete random variable. Then* P_i, $1 \leq i \leq n$, *where* $P_i = Pr[X = x_i]$ *is said to be probability mass function, if*

  1. $P_i \geq 0$, $\forall$ i.
  2. $\sum_{i=1}^{n} P_i = 1$.

- ***Probability density function:*** *Let X be a continuous random variable. Then $f(x)$ is said to be probability density function, if*

  1. $f(x) \geq 0$, $-\infty < x < \infty$.
  2. $\int_{-\infty}^{\infty} f(x)dx = 1$.

  Furthermore, the probability is calculated by $Pr(a \leq X \leq b) = \int_a^b f(x)dx$.

- ***Cumulative distribution function(CDF):*** *Let X be a random variable. The cumulative distribution function of X, $F : \mathbb{R} \to [0,1]$ is defined by*

$$F(x) = Pr[X \leq x] = \begin{cases} \sum_{x_i \leq x} P_i, & X \text{ is a discrete random variable} \\ \int_{-\infty}^{x} f(t)dt, & X \text{ is a continuous random variable} \end{cases}$$

- ***Mean of a random variable:*** *The mean or expected value of a random variable X is defined by*

$$\mu = E(X) = \begin{cases} \sum_{x_i} x_i P_i, & X \text{ is a discrete random variable} \\ \int x f(x)dx, & X \text{ is a continuous random variable} \end{cases}.$$

- ***Variance of a random variable:*** *The variance of a random variable X with mean μ is defined by $var(X) = \sigma^2 = E(X-\mu)^2$.*

- ***Standard deviation of a random variable:*** *The standard deviation of a random variable X with variance σ^2 is defined by $sd(X) = \sqrt{\sigma^2} = \sigma$.*

Example 3. *Some examples of important distributions of the random variables are given below [29]:*

- ***The Bernoulli distribution:*** *Let X be a discrete random variable for a binary coin flip. Let $Pr(X = 1) = p$, then $Pr(X = 0) = 1 - p$ for some $p \in [0,1]$. Then it is considered as Bernoulli distribution of X and denoted by $X \sim Bernoulli(p)$. The probability mass function is $P_i = p^{x_i}(1-p)^{1-x_i}$ for some $x_i \in \{0,1\}$.*

- ***The Binomial distribution:*** *Suppose there is a binary coin which falls heads up with probability p for some $0 \leq p \leq 1$. The coin is flipped for n times and let X be the number of heads. The probability mass function is* $P_i = \begin{cases} \binom{n}{x_i} p^{x_i}(1-p)^{n-x_i}, & x_i = 0, 1, \ldots, n \\ 0, & others \end{cases}$. *It is then considered the Binomial distribution of X and denoted by $X \sim Binomial(n, p)$.*

- ***Normal or Gaussian distribution:*** *A continuous random variable is said to have a Normal distribution with mean μ and standard deviation σ, denoted by $X \sim \mathcal{N}(\mu, \sigma^2)$, if $f(x) = \frac{1}{\sigma\sqrt{2\pi}} exp\{-\frac{(x-\mu)^2}{2\sigma^2}\}, \quad x \in \mathbb{R}$.*
  If $\mu = 0$ and $\sigma = 1$, then the above distribution is called standard normal distribution of X i.e. $X \sim \mathcal{N}(0, 1)$ and $f(x) = \frac{1}{\sqrt{2\pi}} exp\{-\frac{x^2}{2}\}, \quad x \in \mathbb{R}$.

There are two types of convergence of random variable X [29].

Definition 23. *Let $X_1, X_2, \cdots$ be a sequence of random variables and X be another random variable. Let F_n and F be the cumulative distribution functions of X_n and X respectively.*

1. *X_n converges to X in probability, denoted by $X_n \longrightarrow X$, if $Pr(|X_n - X| > \epsilon) \to 0$, for all $\epsilon > 0$ and $n \to \infty$.*

2. *X_n converges to X in distribution, denoted by $X_n \rightsquigarrow X$, if $lim_{n\to\infty} F_n(t) = F(t)$ at all t for which F is continuous.*

The relationship between these two types of convergence is given by a theorem.

Theorem 4. *The following relationships are hold:*

(a) $X_n \longrightarrow X$ implies that $X_n \rightsquigarrow X$.

(b) $X_n \rightsquigarrow X$ and $Pr(X = c) = 1$, $c \in \mathbb{R}$ implies that $X_n \longrightarrow X$.

Let $X = (X_1, \ldots, X_n)$,where X_i are the random variables, be a random vector whose probability density function is $f(x_1, \ldots, x_n)$. Also if $X_1, \ldots, X_n$ are independent, $Pr(X_1 \in \mathbb{E}_1, \ldots, X_n \in \mathbb{E}_n) = \prod_{i=1}^{n} Pr(X_i \in \mathbb{E}_i)$.

Definition 24. *If $X_1, \ldots, X_n$ are independent and have the same marginal distribution with cumulative distribution function F, we say that $X_1, \ldots, X_n$ are IID(independent and identically distributed) and we write $X_1, \ldots, X_n \sim F$. Also $X_1, \ldots, X_n$ are called a random sample of size n from F.*

Let $X_1, \ldots, X_n$ be an IID sample and $\mu = E(X_i)$ be the mean and σ^2 be the variance of X_i, for any $1 \leq i \leq n$. Note that the mean of each X_i is same as they are IID. The sample mean is defined by $\bar{X}_n = \frac{\sum_{i=1}^{n} X_i}{n}$. Finally we discuss about the two popular theorems regarding convergence of random variables. The first one is discussed below.

Theorem 5. *The sample mean $\bar{X}_n = \frac{\sum_{i=1}^{n} X_i}{n}$ converges in probability to the mean $\mu = E(X_i)$, for any i. It is known as law of large numbers.*

The another theorem is the most important theorem, where $\bar{X}_n$ has mean $E(\bar{X}_n) = \mu$ and variance $\sigma^2_{\bar{X}_n} = \frac{\sigma^2}{n}$.

Theorem 6. *Let $X_1, X_2, \ldots, X_n$ be the IID with mean μ and variance σ^2 and $\bar{X}_n = \frac{\sum_{i=1}^{n} X_i}{n}$. Then $Z_n = \frac{\bar{X}_n - \mu}{\sqrt{\sigma^2_{\bar{X}_n}}} = \frac{\sqrt{n}(\bar{X}_n - \mu)}{\sigma}$ converges in distribution to Z, where $Z \sim \mathcal{N}(0,1)$. In other words, $\lim_{n\to\infty} Pr(Z_n \leq z) = \int_{-\infty}^{z} \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}} dx$. It is known as the central limit theorem.*

The important part of a statistical distribution is hypothesis testing, a method to test a claim or hypothesis about a parameter in the population. Two types of hypothesis testing are there to test the hypothesis against each other. These are:

1. **Null hypothesis:** It is a statement about a population parameter that is assumed to be true. It is a starting point. Our goal is to test that whether the value stated in the null hypothesis is likely to be true. It is denoted by H_0.

2. **Alternative hypothesis:** It is a statement that directly contradicts a null hypothesis by stating that the actual value of the population parameter is less than or greater than or not equal to the value stated in the null hypothesis. It is denoted by H_1.

There are few terms related to the hypothesis. These are the following.

- **The level of significance** refers to a criterion of judgment upon which a decision is made regarding the value stated in H_0. The criterion is based on the probability of obtaining a statistic measured in a sample if the value stated in H_0 is true. Typically it is set at 5%.

- **Test statistic** is a mathematical formula that allows researchers to determine the likelihood of obtaining sample outcomes if the H_0 is true. The value of the test statistic is used to make a decision regarding H_0.

- The ***p*-value** is the probability of obtaining a sample outcome when the value stated in H_0 is true. The p-value for obtaining the sample outcome is compared to the level of significance.

There are two types of errors that occur at the time of deciding that whether H_0 is retained or rejected. These are the following.

1. **Type-I error** is the probability of rejecting a null hypothesis that is actually true.

2. **Type-II error** is the probability of retaining a null hypothesis that is actually false.

One more important term regarding rejecting H_0 is an **alpha(α) level** which is the level of significance for a hypothesis test. It is the largest probability of committing a Type-I error that we will allow and still decide to reject H_0.

2.2 Cryptographic Background

The main building block of stream cipher is the linear feedback shift register(LFSR). It is always associated with a connection polynomial $C(x) \in \mathbb{Z}_2[x]$ such that

$$C(x) = 1 + c_1x + c_2x^2 + \cdots + c_dx^d.$$

Definition 25. *The following definitions [2] regarding LFSR are:*

- *The LFSR with a connection polynomial $C(x)$ of degree d (i.e., $c_d = 1$) is said to be **non-singular**.*
- $[s_{d-1}, \cdots, s_1, s_0]$ *is said to be **initial state** of LFSR, where $s_i \in \{0, 1\},\ 0 \leq i < d$.*

There are also some facts regarding LFSR, which are listed below:

1. Every output sequence produced by all possible initial states of LFSR is periodic iff LFSR is non-singular.
2. If $C(x)$ is irreducible over $\mathbb{Z}_2$, then every output sequence produced by all possible initial states has a period less than or equal to p, where $p|2^d - 1$.
3. If $C(x)$ is primitive over $\mathbb{Z}_2$, then every output sequence produced by all possible nonzero initial states has a period equal to $2^d - 1$.

The LFSR is used to build a stream cipher for several reasons which are:

- It can be implemented in hardware very well.
- It produces the sequences which have a large period.
- It produces the sequences which have good statistical properties.
- It can be easily analyzed using algebraic techniques due to its structure.

Definition 26. *A (general)* ***feedback shift register****(FSR) of length d which can store one bit and has one input, one output and a clock to control the movement of data, consists of d contents* $s_0, ..., s_{d-1}$*. The following operations are performed during each unit of time:*

- *The output is* s_0 *which is a part of the output sequence.*
- $s_i = s_{i+1},\ 0 \leq i \leq d-2.$
- $s_{d-1} = f(s'_0, \ldots, s'_{d-1})$*, where the Boolean function* f *is the feedback function and* $[s'_{d-1}, \cdots, s'_0]$ *is the previous state.*

Based on the degree of the feedback function, it is classified by two parts [2]:

1. If the feedback function is linear, then the FSR is called **linear feedback shift register** (LFSR), where $f = c_1 s_{d-1} + c_2 s_{d-2} + \cdots + c_d s_0$ given that $[s_{d-1}, \cdots, s_0]$ is the previous state.
2. If the feedback function is non-linear i.e. the degree of the function is atleast 2, then the FSR is called **non-linear feedback shift register** (NFSR).

Note: For $[0, \cdots, 0]$ initial state of LFSR, the output sequence is the zero sequence.

Definition 27. *An LFSR is said to generate an infinite sequence* s *or a sequence whose first* n *terms are* s^n*, if it is generated from some initial state of the LFSR, where* s^n *is of length* n *whose terms are* $s_0, \ldots, s_{n-1}$*.*

Definition 28. *The* ***Linear complexities*** *[2] of infinite and finite sequences are discussed below:*

1. *The* ***Linear complexity*** $L(s)$ *of an inifinte binary sequence* s *is defined as follows:*
   - $L(s) = 0$ *for zero sequence.*
   - $L(s) = \infty$*, if no LFSR generates the sequence.*

- *Else, $L(s)$ is the length of the shortest LFSR which generates s.*

2. *The **Linear complexity** $L(s^n)$ of a finite binary sequence s^n is defined by the length of the shortest LFSR generating a sequence whose first n terms are s^n.*

Properties of linear complexity: For two binary sequences s_1 and s_2,

1. $0 \leq L(s^n) \leq n$, for any $n \geq 1$.
2. $L(s^n) = 0 \Leftrightarrow s^n$ is a zero sequence of length n.
3. $L(s^n) = n \Leftrightarrow s^n = 0, 0, \ldots, 0, 1$.
4. $L(s) \leq P$, if s is the periodic sequence of period P.
5. $L(s \oplus t) \leq L(s) \oplus L(t)$.

Berlekamp-Massey algorithm [2]: The linear complexity of a finite binary sequence s^n of length n is determined by the Berlekamp-Massey algorithm. This algorithm is based on the next discrepancy D_N, which is defined below:

Definition 29. *The next decrepancy D_N is the difference between $(N+1)^{th}$ term s_N of a finite binary sequence s^{N+1} and the $(N+1)^{th}$ term generated by a LFSR of length d with connection polynomial $C(x) = 1 + c_1 x + \cdots + c_d x^d$ which generates a subsequence s^N of the s^{N+1}.*

$$D_N = (s_N \oplus \sum_{i=1}^{d} c_i s_{N-i})$$

The algorithm follows some facts due to the next decrepancy D_N. Facts are given below:

1. A LFSR of length d with connection polynomial $C(x)$ which generates s^N also generates $s^{N+1} \Leftrightarrow D_N = 0$.
2. $L(s^{N+1}) = d$, if $D_N = 0$.

3. Let $D_N = 1$ and largest integer $m < N$ such that $L(s^m) < L(s^N)$. If a LFSR of length $L(s^m)$ with it's connection polynomial $B(x)$ generates the sequence s^m, then LFSR of smallest length $d^{'}$ with connection polynomial $C^{'}(x)$ generates s^{N+1} such that

$$d^{'} = \begin{cases} d, & if\ d > \frac{N}{2}, \\ N + 1 - d, & if\ d \leq \frac{N}{2}, \end{cases}$$

and $C^{'}(x) = C(x) + B(x)x^{N-m}$.

The Berlekamp-Massey algorithm is given below in Algorithm 1.

Algorithm 1: Berlekamp-Massey Algorithm

Input : A binary sequence $s^n = s_0, s_1, \cdots, s_{n-1}$ of length n.
Output: Linear complexity $L(s^n)$ of s^n, $0 \leq L(s^n) \leq n$.

```
Assign C(x) ← 1, d ← 0, m ← −1, B(x) ← 1, N ← 0;
while N < n do
    Compute the next discrepancy D_N where D_N = (s_N ⊕ Σ_{i=1}^{d} c_i s_{N−i});
    if D_N = 1 then
        T(x) ← C(x), C(x) ← C(x) + B(x).x^{N−m};
        if d ≤ N/2 then
            d ← N + 1 − d, m ← N, B(x) ← T(x);
        end
    end
    N ← N + 1;
end
return d;
```

Example 4. *Suppose* $s^n = 0, 0, 1, 1, 0, 1, 1, 1, 0$ *of length* $n = 9$. *Then* $L(s^n) = 5$.

There is one fact that there exists a unique LFSR of length d generating s^n, where $L(s^n) = d$ iff $d \leq \frac{n}{2}$. Based on the fact, one can easily modify the Algorithm 1 by returning both d and $C(x)$ for a subsequence of length atleast $2d$ of an infinite binary sequence s with linear complexity d. Due to that, the output sequence produced by an LFSR can be predicted easily, although many keystream generators use LFSR due to some good reasons. If any subsequence (of length at least $2d$) of the output

sequence s of an LFSR with linear complexity d is known, the Berlekamp-Massey algorithm can determine the sequence s efficiently. With its connection polynomial, the LFSR is initialized with any subsequence of length d and generates the reminder of s. An adversary can use known-plaintext attack to get the subsequence of s as $m_i \oplus c_i,\ 1 \leq i \leq 2d$, where m_i and c_i are the plaintext and corresponding ciphertext bits. To resist the known-plaintext attack on LFSR based system, one should not use only LFSR as a keystream sequence generator. So linearity properties are the biggest weakness of LFSR. Three methods are used to destroy the properties. Here is the following:

- The outputs of several LFSRs are used in a nonlinear combining function. It is known as **Nonlinear combination generators**. Geffe generator is an example of this generator whose nonlinear combining function is $f(x_1, x_2, x_3) = x_1x_2 \oplus (1 \oplus x_2)x_3$, where x_1, x_2, x_3 are the outputs of three LFSRs.

- The content of a single LFSR is used in a nonlinear filtering function. It is known as **Nonlinear filtering generators**. Knapsack generator is an example of it. Here the nonlinear function is $f(x) = \sum_{i=1}^{L} x_i a_i \ (mod\ 2^L)$, where $x = [x_L, \cdots, x_1]$ is a binary state of the LFSR at a time and $a_i,\ 1 \leq i \leq L$ are the knapsack integer weights of the secret key.

- To control the clock of one (or more) LFSRs by using the output of other(one or more) LFSRs. It is called **Clock-controlled generators**. The alternating step generator and shrinking generator are two examples of clock-controlled generators.

2.2.1 Design of cyptographic scheme

The state bits of the NFSR and LFSR are denoted by b_i and s_i for the grain family, respectively. There are two main algorithms for a Grain-like stream cipher. The algorithms are

1. key initialization round or key scheduling algorithm (KSA).

2. pseudo-random generation algorithm (PRGA).

We use the term key scheduling algorithm (KSA) everywhere in the thesis. The graphical view of these two phases of the stream cipher Grain-v1 is provided in Figure 2.1 and Figure 2.2. Note that no LFSR bit except the LFSR bits in h function is XORed at the time of calculating z_t in the case of Grain-v1.

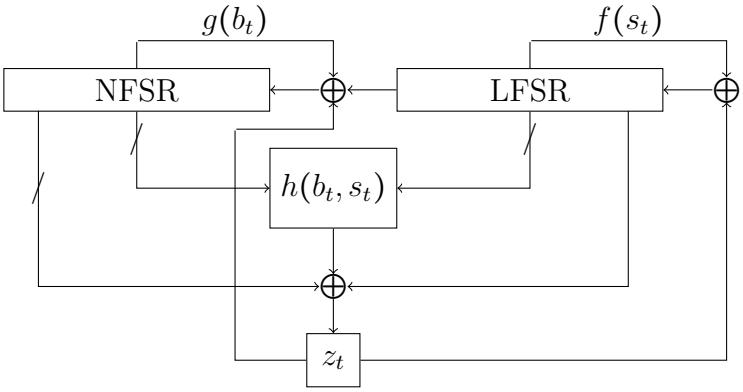

Figure 2.1: KSA of Grain Family

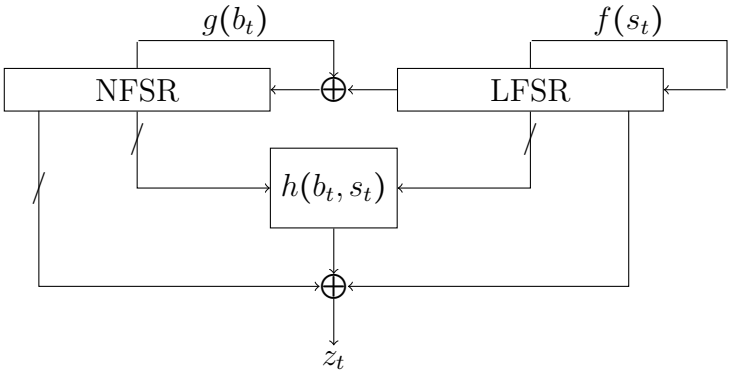

Figure 2.2: PRGA of Grain Family

2.2.2 Design specification of Grain-v1

Grain-v1 [4] is based on an 80-bit NFSR, an 80-bit LFSR and a nonlinear filter function. The feedback bit of LFSR (s_{t+80}) and NFSR (b_{t+80}) are computed using the following feedback functions.

$$s_{t+80} = s_{t+62} + s_{t+51} + s_{t+38} + s_{t+23} + s_{t+13} + s_t, \text{ for } t \geq 0. \tag{2.2}$$

$$\begin{aligned} b_{t+80} \;=\; & s_t + b_{t+62} + b_{t+60} + b_{t+52} + b_{t+45} + b_{t+37} + b_{t+33} + b_{t+28} + b_{t+21} + b_{t+14} \\ & + b_{t+9} + b_t + b_{t+63}b_{t+60} + b_{t+37}b_{t+33} + b_{t+15}b_{t+9} + b_{t+60}b_{t+52}b_{t+45} \\ & + b_{t+33}b_{t+28}b_{t+21} + b_{t+63}b_{t+45}b_{t+28}b_{t+9} + b_{t+60}b_{t+52}b_{t+37}b_{t+33} \\ & + b_{t+63}b_{t+60}b_{t+21}b_{t+15} + b_{t+63}b_{t+60}b_{t+52}b_{t+45}b_{t+37} \\ & + b_{t+33}b_{t+28}b_{t+21}b_{t+15}b_{t+9} + b_{t+52}b_{t+45}b_{t+37}b_{t+33}b_{t+28}b_{t+21}, \text{ for } t \geq 0. \end{aligned} \tag{2.3}$$

The algebraic normal form of the nonlinear filter generator (a Boolean function) h is

$$\begin{aligned} h(x_0, x_1, x_2, x_3, x_4) = {} & x_1 + x_4 + x_0x_3 + x_2x_3 + x_3x_4 + \\ & x_0x_1x_2 + x_0x_2x_3 + x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4. \end{aligned} \tag{2.4}$$

The variables x_0, x_1, x_2, x_3, x_4 correspond to the state bits s_{t+3}, s_{t+25}, s_{t+46}, s_{t+64}, b_{t+63} respectively at the t-th clock. In each round, the cipher computes one keystream bit z_t using some state bits from the NFSR and output of the nonlinear filter function. The algebraic expression of the keystream bit at t-th round is

$$z_t = \sum_{k \in A} b_{t+k} + h(s_{t+3}, s_{t+25}, s_{t+46}, s_{t+64}, b_{t+63}), \text{ for } t \geq 0, \tag{2.5}$$

where $A = \{1, 2, 4, 10, 31, 43, 56\}$.

KSA of Grain-v1: The KSA algorithm of Grain-v1 for r rounds is described in Algorithm 2. The value of r for the full round of Grain-v1 is 160.

Algorithm 2: KSA of Grain-v1

Input : $K = (k_0, k_1, \cdots, k_{79})$, $IV = (iv_0, iv_1, \cdots, iv_{63})$.
Output: State $S = (b_0, \cdots, b_{79}, s_0, \cdots, s_{79})$ of Grain-v1 after key scheduling process.

Assign $b_i = k_i$ for $i = 1, \cdots, 79$; $s_i = iv_i$ for $i = 0, \cdots, 63$; $s_i = 1$ for $i = 64, \cdots, 79$;
for *r rounds* **do**
  Compute $z = \sum_{k \in \mathcal{A}} b_k + h(s_3, s_{25}, s_{46}, s_{64}, b_{63})$, for $\mathcal{A} = \{1, 2, 4, 10, 31, 43, 56\}$;
  $t_1 = z + s_{62} + s_{51} + s_{38} + s_{23} + s_{13} + s_0$;
  $t_2 = z + b_{80}$ where b_{80} is computed as Equation (2.3) putting $t = 0$;
  $b_i = b_{i+1}$ and $s_i = s_{i+1}$ for $i = 0, 1, \cdots, 78$;
  $s_{79} = t_1$ and $b_{79} = t_2$;
end
return $S = (b_0, b_1, \cdots, b_{79}, s_0, s_1, \cdots, s_{79})$;

PRGA of Grain-v1: After the completion of the key scheduling phase, the cipher starts the pseudorandom bit generation phase, where instead of fed backing to LFSR and NFSR, the keystream bit is produced as output bit. These keystream bits are used for encryption/decryption of plaintext/ciphertext.

2.2.3 Design specification of Grain-128

Grain-128 [30] is based on a 128-bit NFSR, a 128-bit LFSR and a nonlinear filter function. The feedback bit of LFSR and NFSR are computed using the following feedback functions.

$$s_{t+128} = s_{t+96} + s_{t+81} + s_{t+70} + s_{t+38} + s_{t+7} + s_t, \text{ for } t \geq 0. \tag{2.6}$$

$$\begin{aligned} b_{t+128} &= s_t + b_{t+96} + b_{t+91} + b_{t+56} + b_{t+26} + b_t + b_{t+3}b_{t+67} + b_{t+11}b_{t+13} \\ &\quad + b_{t+17}b_{t+18} + b_{t+27}b_{t+59} + b_{t+40}b_{t+48} + b_{t+61}b_{t+65} + b_{t+68}b_{t+84} \\ &\quad , \text{ for } t \geq 0. \end{aligned} \tag{2.7}$$

The algebraic normal form of the nonlinear filter generator (a Boolean function) h is

$$h(x_0, \ldots, x_8) = x_0x_1 + x_2x_3 + x_4x_5 + x_6x_7 + x_0x_4x_8.$$

The variables x_0, x_1, x_2, x_3, x_4 correspond to the state bits $b_{12+t}, s_{8+t}, s_{13+t}, s_{20+t}, b_{95+t}$, $s_{42+t}, s_{60+t}, s_{79+t}, s_{94+t}$ respectively at the t-th clock. In each round, the cipher computes one keystream bit z_t using some state bits from the NFSR and output of the nonlinear filter function. The algebraic expression of the keystream bit at t-th round is

$$z_t = s_{t+93} + \sum_{k \in A} b_{t+k} + h(b_{12}, s_8, s_{13}, s_{20}, b_{95}, s_{42}, s_{60}, s_{79}, s_{94}), \text{ for } t \geq 0, \tag{2.8}$$

where $A = \{2, 15, 36, 45, 64, 73, 89\}$.

KSA of Grain-128: KSA algorithm of Grain-128 for r rounds is described in Algorithm 3. The value of r for the full round of Grain-128 is 256.

Algorithm 3: KSA of Grain-128

Input : $K = (k_0, k_1, \cdots, k_{127})$, $IV = (iv_0, iv_1, \cdots, iv_{95})$.
Output: State $S = (b_0, \cdots, b_{127}, s_0, \cdots, s_{127})$ of Grain-128 after key scheduling process.

Assign $b_i = k_i$ for $i = 1, \cdots, 127$; $s_i = iv_i$ for $i = 0, \cdots, 95$; $s_i = 1$ for $i = 96, \cdots, 127$;
for r *rounds* **do**
  Compute $z = \sum_{k \in \mathcal{A}} b_k + s_{t+93} + h(b_{12}, s_8, s_{13}, s_{20}, b_{95}, s_{42}, s_{60}, s_{79}, s_{94})$, for $\mathcal{A} = \{2, 15, 36, 45, 64, 73, 89\}$;
  $t_1 = z + s_{96} + s_{81} + s_{70} + s_{38} + s_7 + s_0$;
  $t_2 = z + b_{128}$ where b_{128} is computed as Equation (2.7) putting $t = 0$;
  $b_i = b_{i+1}$ and $s_i = s_{i+1}$ for $i = 0, 1, \cdots, 126$;
  $s_{127} = t_1$ and $b_{127} = t_2$;
end
return $S = (b_0, b_1, \cdots, b_{127}, s_0, s_1, \cdots, s_{127})$;

PRGA of Grain-128: It is same as Grain-v1.

2.2.4 Design specification of Grain-128a

Grain-128a [31] is based on an 128-bit NFSR, an 128-bit LFSR and a nonlinear filter function. The feedback bit of LFSR and NFSR are computed using the following feedback functions.

$$s_{t+128} = s_{t+96} + s_{t+81} + s_{t+70} + s_{t+38} + s_{t+7} + s_t, \text{ for } t \geq 0. \tag{2.9}$$

$$\begin{aligned} b_{t+128} = \; & s_t + b_{t+96} + b_{t+91} + b_{t+56} + b_{t+26} + b_t + b_{t+3}b_{t+67} + b_{t+11}b_{t+13} \\ & + b_{t+17}b_{t+18} + b_{t+27}b_{t+59} + b_{t+40}b_{t+48} + b_{t+61}b_{t+65} + b_{t+68}b_{t+84} \\ & + b_{t+22}b_{t+24}b_{t+25} + b_{t+70}b_{t+78}b_{t+82} + b_{t+88}b_{t+92}b_{t+93}b_{t+95} \\ & , \text{ for } t \geq 0. \end{aligned} \tag{2.10}$$

The algebraic normal form of the nonlinear filter generator (a Boolean function) h is

$$h(x_0, \ldots, x_8) = x_0x_1 + x_2x_3 + x_4x_5 + x_6x_7 + x_0x_4x_8.$$

The variables x_0, x_1, x_2, x_3, x_4 correspond to the state bits $b_{12+t}, s_{8+t}, s_{13+t}, s_{20+t}, b_{95+t}$, $s_{42+t}, s_{60+t}, s_{79+t}, s_{94+t}$ respectively at the t-th clock. In each round, the cipher computes one keystream bit z_t using some state bits from the NFSR and output of the nonlinear filter function. The algebraic expression of the keystream bit at t-th round is

$$z_t = s_{t+93} + \sum_{k \in A} b_{t+k} + h(b_{12}, s_8, s_{13}, s_{20}, b_{95}, s_{42}, s_{60}, s_{79}, s_{94}), \text{ for } t \geq 0, \tag{2.11}$$

where $A = \{2, 15, 36, 45, 64, 73, 89\}$.

KSA of Grain-128a: KSA algorithm of Grain-128a for r rounds is described in

Algorithm 4. The value of r for the full round of Grain-128a is 256.

Algorithm 4: KSA of Grain-128a

Input : $K = (k_0, k_1, \cdots, k_{127})$, $IV = (iv_0, iv_1, \cdots, iv_{95})$.
Output: State $S = (b_0, \cdots, b_{127}, s_0, \cdots, s_{127})$ of Grain-128a after key scheduling process.

Assign $b_i = k_i$ for $i = 1, \cdots, 127$; $s_i = iv_i$ for $i = 0, \cdots, 95$; $s_i = 1$ for $i = 96, \cdots, 126$, $s_{127} = 0$;
for r *rounds* **do**
  Compute $z = \sum_{k \epsilon \mathcal{A}} b_k + s_{t+93} + h(b_{12}, s_8, s_{13}, s_{20}, b_{95}, s_{42}, s_{60}, s_{79}, s_{94})$, for $\mathcal{A} = \{2, 15, 36, 45, 64, 73, 89\}$;
  $t_1 = z + s_{96} + s_{81} + s_{70} + s_{38} + s_7 + s_0$;
  $t_2 = z + b_{128}$ where b_{128} is computed as Equation (2.10) putting $t = 0$;
  $b_i = b_{i+1}$ and $s_i = s_{i+1}$ for $i = 0, 1, \cdots, 126$;
  $s_{127} = t_1$ and $b_{127} = t_2$;
end
return $S = (b_0, b_1, \cdots, b_{127}, s_0, s_1, \cdots, s_{127})$;

PRGA of Grain-128a: It is the similar as Grain-128. Only for authenticity part, the keystream is taken as z_{64+2t}, $t \geq 0$.

2.2.5 Design specification of Lizard

Lizard [13] is a lightweight stream cipher of state size 121. It consists of two NFSRs S and B of length $31, 90$ respectively and an output function z_t which contains four Boolean functions $\mathcal{L}_t, \mathcal{Q}_t, \mathcal{T}_t, \widehat{\mathcal{T}}_t$.

The update function f of NFSR1 of length 31 is

$$\begin{aligned} S_{31+t} = \quad & S_{0+t} + S_{2+t} + S_{5+t} + S_{6+t} + S_{15+t} + S_{17+t} + S_{18+t} + S_{20+t} + S_{25+t} \\ & + S_{8+t}S_{18+t} + S_{8+t}S_{20+t} + S_{12+t}S_{21+t} + S_{14+t}S_{19+t} + S_{17+t}S_{21+t} \\ & + S_{20+t}S_{22+t} + S_{4+t}S_{12+t}S_{22+t} + S_{4+t}S_{19+t}S_{22+t} + S_{7+t}S_{20+t}S_{21+t} \\ & + S_{8+t}S_{18+t}S_{22+t} + S_{20+t}S_{21+t}S_{22+t} + S_{4+t}S_{7+t}S_{12+t}S_{21+t} \\ & + S_{4+t}S_{7+t}S_{19+t}S_{21+t} + S_{4+t}S_{12+t}S_{21+t}S_{22+t} + S_{4+t}S_{19+t}S_{21+t}S_{22+t} \\ & + S_{7+t}S_{8+t}S_{18+t}S_{21+t} + S_{7+t}S_{8+t}S_{20+t}S_{21+t} + S_{7+t}S_{12+t}S_{19+t}S_{21+t} \\ & + S_{8+t}S_{18+t}S_{21+t}S_{22+t} + S_{8+t}S_{20+t}S_{21+t}S_{22+t} \\ & + S_{12+t}S_{19+t}S_{21+t}S_{22+t} \end{aligned} \tag{2.12}$$

The update function g of NFSR2 of size 90 is

$$\begin{aligned} B_{90+t} \quad = \quad & S_{0+t} + B_{0+t} + B_{24+t} + B_{49+t} + B_{79+t} + B_{84+t} + B_{3+t}B_{59+t} + B_{10+t}B_{12+t} \\ & + B_{15+t}B_{16+t} + B_{25+t}B_{53+t} + B_{35+t}B_{42+t} + B_{55+t}B_{58+t} + B_{60+t}B_{74+t} \\ & + B_{20+t}B_{22+t}B_{23+t} + B_{62+t}B_{68+t}B_{72+t} + B_{77+t}B_{80+t}B_{81+t}B_{83+t} \end{aligned} \tag{2.13}$$

The output function of 53 bits input is

$$z_t = \mathcal{L}_t + \mathcal{Q}_t + \mathcal{T}_t + \widehat{\mathcal{T}}_t \tag{2.14}$$

where

$$\mathcal{L}_t = B_{7+t} + B_{11+t} + B_{30+t} + B_{40+t} + B_{45+t} + B_{54+t} + B_{71+t} \tag{2.15}$$

$$\mathcal{Q}_t = B_{4+t}B_{21+t} + B_{9+t}B_{52+t} + B_{18+t}B_{37+t} + B_{44+t}B_{76+t} \tag{2.16}$$

$$\widehat{\mathcal{T}}_t = S_{23+t} + S_{3+t}S_{16+t} + S_{9+t}S_{13+t}B_{48+t} + S_{1+t}S_{24+t}B_{38+t}B_{63+t} \tag{2.17}$$

$$\begin{aligned}\mathcal{T}_t = {} & B_{5+t} + B_{8+t}B_{82+t} + B_{34+t}B_{67+t}B_{73+t} + B_{2+t}B_{28+t}B_{41+t}B_{65+t} \\ & + B_{13+t}B_{29+t}B_{50+t}B_{64+t}B_{75+t} + B_{6+t}B_{14+t}B_{26+t}B_{32+t}B_{47+t}B_{61+t} \\ & + B_{1+t}B_{19+t}B_{27+t}B_{43+t}B_{57+t}B_{66+t}B_{78+t}\end{aligned} \tag{2.18}$$

The state initialization of Lizard is performed in four phases as following.

Phase-1(Key and IV loading): Let $K = (K_0, \cdots, K_{119})$ and $IV = (IV_0, \cdots, IV_{63})$. Then

$$B_j^0 = \begin{cases} K_j + IV_j, & 0 \le j \le 63 \\ K_j & 64 \le j \le 89 \end{cases}$$

$$S_j^0 = \begin{cases} K_{j+90}, & 0 \le j \le 28 \\ K_{119} + 1, & j = 29 \\ 1, & j = 30. \end{cases}$$

Phase-2(Grain-like mixing): For $1 \le t \le 128$, the output bit z_t is fed back into both NFSRs by adding with the feedback bits of them.

Phase-3(Second key addition): Key bits are mixed with the states again as

$$B_j^{129} = B_j^{128} + K_j, \qquad 0 \le j \le 89$$

$$S_j^{129} = \begin{cases} S_j^{128} + K_{j+90}, & 0 \le j \le 29 \\ 1, & j = 30. \end{cases}$$

Phase-4(Final diffusion): This phase is similar to Phase-2 except that z_t is not fed back to both NFSRs. Both NFSRs are run up to 128 rounds. After that the keystream bit is produced.

2.3 Cryptanalysis techniques

In this section, we present some basic and vital cryptanalysis techniques used to analyse stream ciphers. We expect that the reader will gain basic knowledge on cryptanalysis and will not face problems following our proposals to investigate the stream ciphers. We use these techniques to analyse the stream ciphers.

2.3.1 Tools for cryptanalysis

2.3.1.1 Random polynomial function

The outputs of a stream cipher need to behave like the outputs of a random polynomial. Further, the random polynomial is required to study cube attack, differential attack, etc., on stream ciphers. We define the random polynomial [32] as below.

Definition 30. *A **random polynomial** of degree d in $n+m$ variables is a multivariate polynomial f such that each possible term of degree at most d is independently chosen to occur with probability* 0.5.
*Further, a d-**random polynomial** in $n+m$ variables is a multivariate polynomial such that each possible term of degree d (contains one secret and $d-1$ public variables) is independently chosen to occur with probability* 0.5 *and the remaining terms can be chosen arbitrarily.*

2.3.1.2 Birthday problem

This problem is exciting from a cryptanalytic perspective, and mainly it is used to find out the collision between two points. Assume that there are N independent and

identically distributed random variables $X_0, \ldots, X_{N-1}$. Each X_i is drawn uniformly from the alphabet $\chi = \{0, 1, \ldots, n-1\}$. The probability that at least two of the random variables have the same value, is $1 - \prod_{i=0}^{N-1}(1 - \frac{i}{n}) \approx 1 - e^{-\frac{N^2}{2n}}$. The name comes from the fact that if 23 people are present in a room, the chance that at least two persons share a birthday exceeds 50%. Further, a group of 70 people has a 99.9% chance of a shared birthday, while a group of 367 people is required to have a 100% chance of a shared birthday (pigeonhole principle).

2.3.1.3 Sample complexity for distinguishing attack

To distinguish the output of a Boolean function from a random sequence, we need to verify the deviation of a property of the Boolean function from a random function. Since the functions have a large domain, it is not computationally possible to verify for every input. Therefore, the cryptanalysts make use of statistics with a high confidence.

It is necessary to use a sufficient number of samples to distinguish the distribution Y (from cipher) from the distribution X (from random source) for a high confidence level. Let the observation be that an event E happens in X and Y with probabilities p and $p(1 + q)$ respectively. The number of required samples is $O(\frac{1}{pq^2})$ to confirm the distribution of Y with high confidence [33]. The confidence level (i.e., the success probability) can be increased by taking higher multiples of $\frac{1}{pq^2}$. The success probability is approximately 69% for $\frac{1}{pq^2}$ samples and it is increased to more than 99.9% for $\frac{39}{pq^2}$ samples [34]. For instant, $p = \frac{1}{2}$ and $q = \frac{1}{2^u}$, the component of complexity is $O(2^{2u+1})$. Further in case of cube tester, one needs 2^c data to cover all assignments of the cube variables, where c is the dimension of the cube. Hence, the sample complexity for this attack would be $O(2^{c+2u+1})$ for a cube distinguiher.

2.3.2 Differential Attack

The motivation of this attack [35] comes from the fact that the adversary imposes differences to some state bits of the output function at the initialization round such

that the derivative functions (Definition 31) behave nonrandomly at higher initialization rounds. In general, the cryptanalysts impose differences to some state bits such that the derivative functions are zero for the first few initialization rounds, and then the nonrandomness propagates to higher rounds. Then distinguishing the derivative at higher rounds from a random function results that the output function can be distinguished from a random one at that round.

Definition 31. *The derivatives of a Boolean function is defined below:*

- *The derivative of a Boolean function* $f : \mathbb{F}_2^n \to \mathbb{F}_2$ *with respect to* $a \in \mathbb{F}_2^n$ *is defined by* $\Delta_a f(x) = f(x \oplus a) + f(x)$. *It is also a Boolean function.*
- *If* $\sigma = \{a_1, ..., a_i\}$ *be a set of vectors in* $\mathbb{F}_2^n$ *and* $L(\sigma)$ *be the set of all* 2^i *linear combinations of elements in* σ. *The* i^{th} *derivative of* f *w.r.t* σ *is defined by* $\Delta_\sigma^{(i)} f(x) = \sum_{c \in L(\sigma)} f(x \oplus c)$. *So it can be evaluated by summing up* 2^i *evaluations of* f.

Note 1. *We always assume that* $a_1, ..., a_i$ *are linearly independent else* $\Delta_\sigma^{(i)} f(x) = 0$. *For keyed Boolean function, differences are applied to the IVs, not to the keys.*

Frequency test for random Boolean function : Let f be a random Boolean function on a set $D \subset \mathbb{F}_2^n$. Then central limit theorem (Theorem 6) says that for sufficiently many inputs $x_1, \ldots, x_s$, $t = \frac{\sum_{i=1}^{s} f(x_i) - \frac{s}{2}}{\sqrt{\frac{s}{4}}} \sim \mathcal{N}(0,1)$, where CDF is $\phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{\frac{-u^2}{2}} du$. Then f is said to pass the frequency test on $x_1, \ldots, x_s$ at a significance level α, if $\phi(t) < 1 - \frac{\alpha}{2}$. So f passes the frequency test with probability $1 - \alpha$. Let β be the probability to pass the frequency test for a random key k of a keyed Boolean function $f(k, .)$, then the advantage to distinguish the keyed Boolean function from a random Boolean function is $1 - \alpha - \beta$.

The differential attack is done using following steps:

- First adversary finds a suitable set of difference vector σ on IV bits of a keyed Boolean function $f(k, .)$ such that the derivatives of the function are zero for first few (as much as possible) rounds (i.e., $\Delta_\sigma^i f(k, .) = 0$, for all/ some i).

- The derivatives $\Delta^i_\sigma f(k,.)$ are calculated for larger number of rounds.
- The statistical tests such as frequency test are used to distinguish $\Delta^i_\sigma f(k,.)$ from random Boolean function. Note that, there may exist some statistical tests where $\Delta^i_\sigma f(k,.)$ can't be distinguished but it may be distinguished by other statistical tests.

2.3.3 Cube attack

The primary motivation of this attack is to find out the values of the variables $k_1,\ldots,k_n$ of a multivariate polynomial $f(k_1,\ldots,k_n,v_1,\ldots,v_m)$ without knowing its ANF, where $K=(k_1,\ldots,k_n)$ is an n-bit secret key and $IV=(v_1,\ldots,v_m)$ is an m-bit IV. The adversary is provided a black box to query the unknown polynomial $f(k_1,\ldots,k_n,v_1,\ldots,v_m)$ over $\mathbb{F}_2$ for different secret keys and fixed IV. The black box returns a single output for each secret key. Some definitions and a theorem related to the attack are presented next before describing the attack.

Cube and its superpoly: Let $f\in\mathbb{B}_n$ for a positive integer n. For any $I=\{i_1,i_2,\ldots,i_c\}\subsetneq\{1,2,\ldots,n\}$, the monomial t_I and the subset of variables C_I are denoted by

$$t_I=x_{i_1}x_{i_2}\cdots x_{i_c} \text{ and } C_I=\{x_{i_1},x_{i_2},\ldots,x_{i_c}\}$$

respectively. Then f can be expressed as

$$f(x_1,x_2,\ldots,x_n)=t_I p_{s(I)}(y_1,y_2,\ldots,y_{n-c})+q(x_1,x_2,\ldots,x_n), \tag{2.19}$$

where $\{y_1,y_2,\ldots,y_{n-c}\}=\{x_1,x_2,\ldots,x_n\}\setminus C_I$ and no monomial of q is divisible by t_I. Here, $p_{s(I)}$ is a function in variables $y_1,y_2,\ldots,y_{n-c}$. Then adding the outputs of f at all possible 2^c assignments of the variables in the set C_I, we have

$$\bigoplus_{\{x_{i_1},\ldots,x_{i_c}\}\in\mathbb{F}_2^c} f(x_1,x_2,\ldots,x_n)=p_{s(I)}(y_1,y_2,\ldots,y_{n-c}). \tag{2.20}$$

Here C_I is called a `cube` of dimension c or a c-dimensional cube and $p_{s(I)}$ is called the `superpoly` of the cube. The polynomial $f(x_1, x_2, \ldots, x_n)$ is called the master polynomial.

Definition 32. *A maxterm of f is a term t_I such that $deg(p_{s(I)}) = 1$, i.e., $p_{s(I)}$ is a non-constant linear polynomial.*

Theorem 7. *Let t_I be a maxterm in a black box polynomial f [32]. Then*

1. *The **free term** in $p_{s(I)}$ can be determined by XORing the values of f over n variables which are zero except the i_c variables in C_I.*

2. *The coefficient of y_j in $p_{s(I)}$ is determined by XORing the values of f over n variables which are zero except the i_c variables in C_I and y_j, which is set to the value one.*

Dinur et. al. [32] first proposed a **cube attack**. The attack is implemented in two phases.

- **The preprocessing phase:** In this phase, sufficiently many maxterms are computed by using the probabilistic linearity test, which is used to check that the superpolies are linear or not. Then the ANF of the linear superpoly for each maxterm is calculated by finding the coefficients of the secret variables using Theorem 7. Next, the online phase is followed.

- **Online phase:** In this phase, the key is fixed and the IV variables which are not involved in the maxterms are set to constant values (the same values set in the preprocessing phase). Then the values of the superpolies of the corresponding maxterms are calculated by querying the black box. Finally, one gets a system of linear equations over secret variables. By solving the system of equations, one can recover the key bits.

Another popular form of the cube attack is **Cube tester**. Aumasson et al. [36] first introduced the cube tester technique to distinguish a family of functions from the random functions by testing some properties of their superpolies. The superpoly can be distinguished based on a property of random polynomial (See subsection 2.3.1.1), i.e., balancedness, non-constantness, high degree and uniformity of presence of a particular monomial, linear variables, and neutral variables, etc. A few of the popular tests are discussed below:

1. **Balancedness test**: The probability of the occurrence of the output value 1 of a random Boolean function is $\frac{1}{2}$. After choosing a cube from the variables representing IVs, the superpoly $p_{s(I)}$, is a Boolean function on key bits and remaining IV bits. Denote the variable representing the whole key by K and the remaining IV by $\overline{IV}$. Hence, for a random function f, the probability of $p_{s(I)} = 1$ is $\frac{1}{2}$ for every assignment of $(K, \overline{IV})$. If there is a cube for the output function used in a stream cipher such that the superpoly deviates from this probability, then the stream cipher can be distinguished from the random Boolean function.

2. **Constantness test**: The constantness test is a kind of balancedness test where $p_{s(I)}$ is checked for its constantness, i.e., the value of $p_{s(I)}$ is always 1 or 0 for every input. This attack is known as the one-sum or zero-sum distinguisher, respectively.

3. **Presence of linear variables:** In this case, one can check whether any variable y_j is linear in the $p_{s(I)}$. The test is done in the following way for the variable y_1:

   - Pick a random assignment of $(y_2, \ldots, y_{n-c})$.
   - If $p_{s(I)}(0, y_2, \ldots, y_{n-c}) = p_{s(I)}(1, y_2, \ldots, y_{n-c})$, then it returns y_1 as non-linear.
   - Repeat the above steps for N times.
   - Finally, it returns y_1 as linear in $p_{s(I)}$.

This test results correctly with a probability of about $1-2^{-N}$ in a time complexity $N.2^c$.

4. **Presence of neutral variables:** In this case, one can check whether any variable y_j is neutral in the $p_{s(I)}$. The test is done in the following way for y_1:

   - Pick a random assignment of $(y_2, \ldots, y_{n-c})$.
   - If $p_{s(I)}(0, y_2, \ldots, y_{n-c}) \neq p_{s(I)}(1, y_2, \ldots, y_{n-c})$, it returns y_1 as not neutral.
   - Repeat the above steps for N times.
   - Finally, it returns y_1 as neutral variable in $p_{s(I)}$.

   This test results correctly with a probability of about $1-2^{-N}$ in time complexity $N.2^c$.

To check the balancedness, constantness or any property of the superpoly $p_{s(I)}$, one has to evaluate $p_{s(I)}$ for all assignments of $(K, \overline{IV})$, i.e., the whole domain of the Boolean function $p_{s(I)}$. In general, the domain of $p_{s(I)}$ is so large that the evaluation for all assignments from the domain is not practical. In that case, a large number of random assignments from the domain, i.e., $(K, \overline{IV})$ samples, need to be evaluated for a probabilistic version of the test with a high confidence level (see paragraph 2.3.1.3).

2.3.4 Time memory data trade-off(TMDTO) attack

The motivation of this attack [37] comes from the two types of elemental attacks to find the encryption keys of a cipher. Consider that the keyspace contains N keys.

- **Exhaustive search:** It is a known-plaintext attack. Given a plaintext and corresponding ciphertext, the adversary tries to find the encryption key by encrypting the plaintext using each key from the keyspace until he gets the ciphertext. The process requires at most N encryption operations to find the encryption key. It may not be possible to search for all keys in the keyspace as N is very

large. If t number of searching operations is performed to find the encryption key, then the success probability is $\frac{t}{N}$.

- **Table look-up:** It is a chosen-plaintext attack. A preprocessing table is prepared to store the pairs of keys and the ciphertext of the chosen plaintext using the key. The pairs in the table are sorted as per the ciphertexts. During the online phase, given a ciphertext of the chosen plaintext, the adversary searches in the table to find the pair containing the ciphertext. Hence, he finds the encryption key. Hence, $O(log_2 N)$ operations are required to find the encryption key by binary search. If m pairs of key and ciphertext are stored in the table, then the success probability is $\frac{m}{N}$.

The TMDTO attack is a combination of exhaustive search and table look-up attacks. In an exhaustive search, the memory requirement is negligible as no preprocessing is required, and the time complexity is $O(N)$. In table look-up, the memory requirement is $O(N)$ in the preprocessing phase, and the time complexity of the online phase is $O(log_2 N)$. The TMDTO attack is used to invert a one-way function by trading off the time (T) and memory (M). The attack is processed in two phases as follows.

Preprocessing phase: In this phase, preprocessing tables are prepared to cover the whole space of state bits. The table can not cover the whole space as there is a large amount of repetition of states due to the collisions. To avoid collisions, t tables of size $m \times t$ (each one is generated from different functions) which satisfy $mt^2 = N$ are made to cover the whole state space except a few. Only the start and end points of each row in each table are stored to save memory. So the required memory is $M = mt$. Since N points are covered, the preprocessing time is $P = N$. The keystream bits are independent of the plaintext in the case of stream ciphers. Babbage [38] and Golić [39] took advantage of keystream bits (i.e., data D) to involve in the trade-off curve to reduce the time and memory costs. In this technique, a single $m \times t$ table (which satisfies $mt = N$) was proposed to cover the whole space of states. The table

can not cover the whole space as there is a large amount of repetition of states due to the collisions. The total M randomly chosen points x_i (internal states) are covered by calculating $y_i = f(x_i)$. In this case, the required memory is M. The attack is known as the BG-TMDTO attack. Since it covers M points in the table, the preprocessing time $P = M$. Later, Biryukov and Shamir [40] extended the BG-TMDTO attack to stream cipher by utilizing multiple data points, known as BS-TMDTO. Due to the involvement of the data D, the number of tables is reduced to $\frac{t}{D}$. As a result, the memory requirement is $M = \frac{mt}{D}$ and the preprocessing time is $P = \frac{N}{D}$.

Online phase: In this phase, the adversary tries to match the challenge ciphertext with an end point in the preprocessing table. If it matches with an end point, then he tries to verify that there is more than one pre-image or not. To verify the pre-image or the challenge ciphertext not matching with any end points, one tries to find the endpoints by enciphering the challenge ciphertext and again using the same procedure. For a given ciphertext y, it searches in t tables for a match. If a match is not found, then $f^i(y),\ i \leq t-1$ is calculated until the point is found in any t tables. Hence, the time complexity is $T = t^2$ and the success probability $P(S) = \frac{mt}{N}$. In case of BG TMDTO attack, for a given $D+s-1$ keystream bits $c_1, ..., c_{D+s-1}$, D many s-bit state values $y_1, ..., y_D$, are generated, where $y_i = c_i, ..., c_{s+i-1}$. Each y_i is searched in the table until a match is found. In this case, the time complexity $T = D$. If the point y_i is found, the corresponding x_i is the recovered plaintext, and the corresponding key is recovered.

Different TMDTO curves:

- **Hellman Tradeoff Curve:** $TM^2 = t^2.m^2t^2 = N^2$. Here $t \leq \sqrt{N}$, else $T > N$, as a result, the attack is slower than exhaustive search. So $1 \leq T \leq N$ and $\sqrt{N} \leq M \leq N$.

- **BG Tradeoff Curve:** $TP = DM = N$. So $1 \leq T \leq D$. By Birthday paradox (Subsection 2.3.1.2), if $DM = N$, then at least one of the y_i is found in the table with significant probability.

- **BS Tradeoff Curve:** $TM^2D^2 = t^2.(\frac{mt}{D})^2.D^2 = N^2$. This relation is true if $t \geq D$ i.e. $T \geq D^2$. So $D^2 \leq T \leq N$. Here, the number of table look-ups is $t.\frac{t}{D} = \frac{t^2}{D}$ for each of D points. So total number of table look ups is t^2 and the number of disk operations is t^2.

Chapter 3

Conditional Differential Attacks on Grain-v1 of Reduced KSA Rounds

3.1 Motivation

The differential attack [Subsection 2.3.2] plays a crucial role in the cryptanalysis of stream ciphers. This attack can be more effective if some conditions on state bits of the cipher are imposed to vanish the keystream bit difference for the first few rounds. Then this attack is considered as conditional differential cryptanalysis, which is discussed in Section 3.2. This attack is applied on Grain-v1 of reduced KSA rounds to distinguish the first keystream bit of the cipher from a random one. All the existing conditional differential cryptanalysis on Grain-v1 is based on the difference vector of weight one, known as the dynamic cube attack of dimension one. For the first time, we successfully used a difference vector of weight two on the initialization vector (IV) to improve the distinguishing round of Grain-v1.

3.1.1 Our Contributions

In the existing articles [17, 41–43], the authors considered a single bit difference in IV. In the case of a higher number of bit differences in IV is more complicated in handling several conditions. For the first time, we present certain distinguishers for the higher initialization rounds of Grain-v1 with two bits difference in the IV.

- The first distinguisher can distinguish Grain-v1 with 112 KSA rounds from a random source almost certainly (with a success rate approximately 99%).

- The second distinguisher can distinguish Grain-v1 with 114 KSA rounds from a random source with a success rate approximately 73% for 2^{78} weak keys, which is one-fourth of the full keyspace.

- Further, this distinguisher has been extended to 116 KSA rounds with the one-bit difference in the key and four-bit differences in the IV. Here we obtain a successful result in 62% cases for 2^{75} related keys.

The second and third distinguishers are designed by extending the idea of the first technique going backward direction from the initial state. As a result, the last two distinguishers fall in a weak/related key setup. The comparisons of our result with the previous attacks are presented in Table 3.1 and Table 3.2.

3.1.2 Organisation

The remaining part of this chapter is divided into six parts. Section 3.2 discusses the conditional differential attack on any NFSR based stream cipher. The previous such types of attacks on Grain-v1 are discussed in Section 3.3. We discuss our contributions in Section 3.4 and 3.5. The comparison of our attack with other similar kinds of attacks is discussed in Section 3.6. At last, we conclude this chapter in Section 3.7.

Table 3.1: Comparison Table (In single key model)

Reference	R	#Key	#Type I,II, III conditions	#Queries	$\lvert\mathcal{K}\rvert$	Success rate
Knellwolf et al. [17]	97	1024	$33, 5, 0$	2^{31}	2^{80}	83%
	104	1024	$25, 5, 0$	2^{39}	2^{80}	58%
Banik [41]	105	1000	$25, 6, 0$	$2^{39-n_1}*$	2^{80}	92%
Sarkar [42]	106	1000	$34, 6, 0$	2^{30}	2^{80}	63%
Ma et al. [43]	104	1024	$14, 15, 0$	2^{40}	2^{80}	97%
	107	64	$12, 12, 0$	2^{42}**	2^{80}	99%
	110	NA	$17, 15, 0$	2^{47}	2^{80}	NA
Our work	112	2048	$29, 7, 0$	2^{35}	2^{80}	99%

Table 3.2: Comparison Table (In weak, related key model)

Reference	R	#Key	#Type I,II, III conditions	#Queries	$\lvert\mathcal{K}\rvert$	Key model	Success rate
Watanabe et al. [44]	114	128	$23, 1, 39$	2^{32}	2^{40}	Weak	NA
Our work	114	2048	$30, 7, 2$	2^{34}	2^{78}	Weak	73%
	116	4096	$36, 7, 5$	2^{28}	2^{75}	Related†	62%

R: Number of KSA rounds.
#Key: Number of random keys used in the experiment. The higher number of keys confirms the success probability better.
#Queries: Number of queries used for each random key.
$\lvert\mathcal{K}\rvert$: Size of the key space where distinguisher gets success.
*: $b_1 \leq 5$ is the extra Type I conditions for faster implementation.
**: Only for the difference e_{63}.
†: Two related keys differ only at one place. Existing works are based on more number of key bit differences.

3.2 Conditional differential attack on NFSR based stream cipher

In this kind of cryptanalysis, a distinguisher is designed to distinguish the first keystream bit of the cipher from a uniform random source. The differential cryptanalysis technique works by placing certain conditions on the initialization vector (IV) and the key (K) bits. For the Grain family, such an idea was initiated by Knellwolf et al. [17].

The general framework of NFSR based stream ciphers contains a state of m bits such that $m > n + l$, where n, l are the lengths of the key and the initialization vector respectively. Let us denote the initial state as $S_0 = (s_0, s_1, \cdots, s_{m-1}) \in \mathbb{F}_2^m$. In every round, the state $S_i = (s_i, s_{i+1}, \cdots, s_{i+m-1}) \in \mathbb{F}_2^m, i \geq 0$, is updated by a nonlinear feedback shift register following a recursive formula as $S_{i+1} = (s_{i+1}, s_{i+2}, \cdots, s_{i+m-1}, s_{i+m})$, where $s_{i+m} = g(S_i)$ a nonlinear function on the state bits. After performing the nonlinear evolution for a certain number of rounds, the cipher generates its first keystream bit z. Therefore, the first output bit z of the cipher can be represented as the output of a keyed Boolean function $f : \mathbb{F}_2^n \times \mathbb{F}_2^l \mapsto \mathbb{F}_2$, where the first n bits correspond to the secret key K and the following l bits relate to the initialization vector, that is, $z = f(K, IV)$.

For a fixed secret key K, we define the Boolean function $f_K : \mathbb{F}_2^l \mapsto \mathbb{F}_2$ as $f_K(x) = f(K, x)$. Further, for a difference vector $a \in \mathbb{F}_2^l$ on the public parameter IV, we define the difference function $\Delta_a f_K(x) = f_K(x) + f_K(x + a)$. If one runs two instances of the cipher with the same key K and the IVs with difference $a \in \mathbb{F}_2^l$, then the nonlinear differences get added in the feedback bits in every round. That is, in every round, the difference starts affecting the state bits nonlinearly by adding the nonlinear differences in the feedback bits. It might be possible for the attacker to control the spread of differences by putting some conditions on the state bits involved in the nonlinear

function in a particular round. Then going back recursively, the conditions can be represented on the initial state bits. As per the involvement of the type of initial state bits, the conditions are classified as follows.

- Type I: Conditions involving only the bits of IV.
- Type II: Conditions involving the bits of both K and IV (but it may exploit without any information on the key bits).
- Type III: Conditions involving only the bits of K.

The Type I conditions put a restriction on the choice of the initialization vectors, which the attacker can easily achieve. On the other hand, the attacker can not do anything in the case of Type III conditions as the involved bits remain secret for the attacker. However, fixing certain secret key bits, the cryptanalyst can point out a subset of weak keys for which the attack can be implemented. In the case of Type II conditions, since some secret key bits are expressed as some bits of the IV bits; these conditions might be exploited without the knowledge of the secret key bits and consequently, that may help to expose some secret bits.

If the function f is truly random, then the Boolean function $\Delta_a f_K$ should behave like a random function in every sub-domain (i.e., subset) of the domain $\mathbb{F}_2^l$. To control the spreading of the difference by imposing the conditions on IV and K, the domain of the IV and secret key K is shrunk. Therefore, the spreading of difference is controlled in this sub-domain, and some bias is expected in the output of the difference function $\Delta_a f_K$ in this sub-domain. Since the statistical test needs to be performed to find the bias in $\Delta_a f_K$, the number of imposed conditions needs to be optimized such that the number of sample inputs should support the theoretical bound for the statistical test. Therefore, the cryptanalyst attempts to optimize the following parameters while presenting an improvement on such attack:

1. Maximization of the number of KSA rounds;

2. Maximization of the success rate;

3. Maximization of the space of initialization vectors (i.e., minimization of the Type I and Type II conditions);

4. Maximization of the effective keyspace (i.e., minimization of the Type III conditions);

5. Minimization of the number of queries to the oracle (stream cipher) (i.e., minimization of the Type II conditions); for x many IV-bits related to the Type II conditions, we need to query with 2^x many IVs, and one may like to minimize this.

3.2.1 A randomness test of the difference function $\Delta_a f_K$

In this section, we use a statistical method of testing the randomness of the difference function $\Delta_a f_K$. This concept was also followed in [17, 42]. If the Boolean function $\Delta_a f_K : \mathbb{F}_2^l \mapsto \mathbb{F}_2$ is a random Boolean function then the output of $\Delta_a f_K$ is randomly distributed for any non-empty subset S of $\mathbb{F}_2^l$. Hence, it follows from the central limit theorem that, for a sufficiently large many inputs $x_i \in S = \{x_1, x_2, \cdots, x_N\}$, the probability density function of the random variable,

$$X = \sum_{x_i \in S} \Delta_a f_K(x_i),$$

approximately follows the normal distribution $\mathcal{N}(\mu, \sigma)$, i.e.,

$$\phi(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

where the μ and σ are the mean and standard deviation of the distribution of X. Let $b \in \mathbb{F}_2$ be a fixed value. Then, for a random Boolean function $\Delta_a f_K$ with given

$Pr[\Delta_a f_K(x) = b] \geq \frac{1}{2}$, the expectation of $(Pr[\Delta_a f_K(x) = b] - \frac{1}{2})$ is

$$\frac{1}{\sqrt{2\pi}\sigma} \int_{\mu}^{N} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \Big(\frac{x}{N} - \frac{1}{2}\Big)\, dx.$$

In our case, the mean $\mu = \frac{N}{2}$ and standard deviation $\sigma = \frac{\sqrt{N}}{2}$. Hence, solving the integration, we have the required expectation

$$\mathcal{E} = \frac{1}{\sqrt{8\pi N}}. \tag{3.1}$$

If this experiment is done for m number of times then the sum

$$\sum_{Pr[\Delta_a f_K(x)=b] \geq \frac{1}{2}} \Big(Pr[\Delta_a f_K(x) = b] - \frac{1}{2}\Big)$$

is expected to be around $m\mathcal{E}$. For a specific stream cipher, if the value of the sum is deviated from $m\mathcal{E}$ (i.e., either bigger or smaller than $m\mathcal{E}$) for a large percentage of keys K, then it allows to distinguish the stream cipher from a uniform random source. Now we present the distinguishing technique on Grain-v1.

In the case of Grain-v1, the length of key is $n = 80$ and the length of IV is $l = 64$. Consider there are t_1, t_2 and t_3 many Type I, Type II and Type III conditions respectively. As per the involvement of state bits in Type I and Type III conditions, t_1 and t_3 many bits in the IV and the key K need to be fixed respectively. In the case of the Type II conditions, t_2 many IV bits are dependent on some key bits and IV bits. Each assignment of t_2 many IV bits provides each group where we need to check the bias. Let denote the assignments for t_2 many IV bits as $\mathcal{A}_i, 0 \leq i \leq 2^{t_2} - 1$ with some order. One of the 2^{t_2} many assignments for t_2 many IV bits must satisfy the Type II conditions. As the key bits are secret, the correct assignment is not known to the observer.

For each assignment $\mathcal{A}_i$, Grain-v1 has the domain of IV of size $2^{64-t_1-t_2}$. For all the assignments corresponding to Type II conditions, it is expected that the keystream bit

z will be produced with some bias. That is, for certain $b \in \mathbb{F}_2$, we expect a deviation on the $(Pr[\Delta z = b] - \frac{1}{2})$. Hence, for a random key and $b \in \mathbb{F}_2$, we calculate the probability $p_i = Pr[\Delta z = b]$ for each assignment $\mathcal{A}_i, 0 \leq i \leq 2^{t_2} - 1$. As each p_i $(0 \leq i \leq 2^{t_2} - 1)$ might be very close to 0.5, we calculate $\mathcal{W} = \sum_{0 \leq i \leq 2^{t_2}-1, p_i \geq \frac{1}{2}} \left(p_i - \frac{1}{2}\right)$.

If it is found that $\mathcal{W}$ is either greater than or lesser than $2^{t_2}\mathcal{E}$ for a large percentage out of the randomly chosen keys, then we can claim that the cipher output is not pseudorandom. In that event, it is possible to distinguish Grain-v1 from a random source. We summarize this in Algorithm 5. This technique has earlier been exploited in [42]. We too, follow this method to design the distinguishers on Grain-v1 in Sections 3.4 and 3.5.

3.3 Existing conditional differential attacks

In this subsection, we briefly present the existing works on the conditional differential attack on Grain-v1. All these works are based on a single bit difference (i.e., $wt(a) = 1$) on the initialization vector IV, also known as a one-dimensional cube attack. In the case of Grain-v1, the key length $n = 80$ and the IV length $l = 64$. Let denote $e_i \in \mathbb{F}_2^{64}, 0 \leq i \leq 63$ is the unit binary vector where the i-th position from left in e_i is 1 and other positions are 0.

In 2010, Knellwolf et al. [17] proposed the conditional differential attack with one bit difference in IV of Grain-v1 with 97 KSA rounds. The difference vector a was chosen as e_{37}, i.e., they selected two initialization vectors as $IV = (iv_0, \cdots, iv_{37}, \cdots, iv_{63})$ and $\widetilde{IV} = (iv_0, \cdots, 1+iv_{37}, \cdots, iv_{63})$ for the difference. Let z_i and $\widetilde{z}_i$ be the i-th keystream bits with initialization vectors IV and $\widetilde{IV}$ respectively and $\Delta z_i = z_i + \widetilde{z}_i$ be the difference between them. To control the initial spread of the differences in the state, they imposed certain conditions on the bits of IV to make $\Delta z_{12} = 0, \Delta z_{34} = 0$, $\Delta z_{40} = 0$ and $\Delta z_{46} = 0$. With such conditions, non-randomness has been observed at 97-th round. In the same paper, they extended the result to 104 KSA rounds with single

Algorithm 5: Distinguisher for the first keystream bit of Grain-v1 with r KSA rounds.

Input : Round r, Difference vector e_a, Type I (t_1 many), Type II (t_2 many) conditions.

Output: Pseudorandom or Not pseudorandom.

Initialize $count = 0$;
One random key K is generated first;
An attacker $\mathcal{A}$ is given access to the keystream bit z_r generated from Grain-v1 with r KSA rounds;
for *each possibilities of free IV bits* **do**
    Consider all possible 2^{t_2} many IVs corresponding to each 0/1 values for IV bits, which are involved in Type II conditions;
    For each of the above IVs consider a bucket $\mathcal{B}_i$, $i = 0, \ldots, 2^{t_2} - 1$;
    Construct IV and $\widetilde{IV} = IV + e_a$, satisfy Type I conditions;
    Oracle outputs z_r and $\widetilde{z}_r$ for each 2^{t_2} many IVs. Here z_r, $\widetilde{z}_r$ are generated by using IV and $\widetilde{IV}$ respectively;
    Put $z_r + \widetilde{z}_r$ in to their respective buckets $\mathcal{B}_i$.;
end
$\mathcal{A}$ computes the probability $p_i = Pr[z_r \neq \widetilde{z}_r]$, for each bucket $\mathcal{B}_i$;
$\mathcal{A}$ computes $\mathcal{W} = \sum_{0 \leq i \leq 2^{t_2}-1, p_i > \frac{1}{2}} \left(p_i - \frac{1}{2}\right)$;
if $\mathcal{W} > 2^{t_2} \cdot \mathcal{E}$ **then**
    $count++$;
end
Repeat the process from 2 to 15 for N many random keys K (N is large);
if $\frac{count}{N}$ *differs significantly from* $\frac{1}{2}$ **then**
    $D \leftarrow$ Not pseudorandom ;
end
else
    $D \leftarrow$ Pseudorandom ;
end
return D

bit difference in IV.

In 2014, Banik [41] chose the difference vector $a = e_{61}$ and improved the result till 105 KSA rounds. The author imposed some conditions on the bits of IV to make $\Delta z_{15} = \Delta z_{36} = \Delta z_{39} = \Delta z_{42} = 0$. Having the IV's with the imposed conditions, non-randomness in the first keystream bit of Grain-v1 with 105 KSA rounds could be observed.

In 2015, Sarkar [42] improved the number of rounds to 106 by taking the difference vector $a = e_{62}$. This could be achieved by finding the conditions on the IV bits by making $\Delta z_{16} = \Delta z_{34} = \Delta z_{37} = \Delta z_{40} = 0$. The distinguisher could distinguish Grain-v1 with 106 KSA rounds from a random source with a success rate approximately 63%.

In 2016, Ma et al. [43] proposed conditional differential attack on Grain-v1 with 107 KSA rounds. For 107 rounds, they chose three different difference vectors e_{34}, e_{60} and e_{63}. For the difference vector $a = e_{63}$ they imposed conditions on the IV bits to make $\Delta z_{17} = \Delta z_{35} = \Delta z_{38} = \Delta z_{41} = \Delta z_{46} = 0$. With these conditions, they observed bias at the first keystream bit of Grain-v1 with 107 KSA rounds. Similarly for e_{34}, e_{60} they imposed several conditions and observed the presence of bias in the first keystream bit of Grain-v1 with 107 KSA rounds. In the same paper, they extended the conditional differential attack on Grain-v1 with 110 KSA rounds by choosing the difference vector $a = e_{37}$. They imposed conditions on the IV bits to make $\Delta z_{12} = \Delta z_{34} = \Delta z_{40} = \Delta z_{46} = \Delta z_{48} = 0$. The authors have observed bias in the first keystream bit of Grain-v1 with 110 KSA rounds with these conditions. As these are experimental biases, the exact number of samples needs to be described. While this is clear in the case for 107 rounds [43, Table 4], the number of secret key used in the case of 110 rounds [43, Table 5] is not provided.

In the same year, Watanabe et al. [44] proposed a conditional differential attack on Grain-v1 with 114 KSA rounds. In this work, the authors imposed some conditions on IV bits as well as on secret key bits. Since conditions are applied on 39 secret key bits, the attack is restricted to a subset of keyspace of size 2^{40} whereas the size of keyspace

is 2^{80}. If the unknown secret key K is from the set of $(2^{80} - 2^{40})$ many keys, then their attacker will not be able to distinguish Grain-v1 from a random source. Further, the domain of weak keyspace (i.e., of size 2^{40}) is immediately prone to exhaustive key search attack, and thus this result does not look significant.

The comparison of the existing attacks and our present work is presented in the Table 3.1 and Table 3.2. However, one may immediately note that all the existing works in this direction are based on the difference vector of weight 1, i.e., the one-bit difference in the IV bits. In each case, the number of rounds is improved irrespective of the improvement distinguishing success chance, the dimension of IV space, keyspace, and query space. The dimension of IV space is equal to (64- the number of Type I and Type II conditions) and the dimension of keyspace is equal to (80 - the number of Type III conditions). Further, the dimension of the query space is proportional to (the number of Type II conditions+1) as in each case; we need to run the cipher with the same key and two different IVs.
Further, there are several other cryptanalytic results [44–54] available in the literature on Grain-v1 and other variants of Grain family.

3.4 Distinguisher for 112 KSA round of Grain-v1

The non-randomness in the first keystream bit of Grain-v1 with $97, 104, 105, 106$ and 110 KSA rounds have been observed in [17, 41–43]. Further, the same is done for 114 KSA rounds in the weak key set up of keyspace size 2^{40} in [44]. These works exploited the transmission of bias from a single bit difference in the initialization vector IV. In the current situation, improving result using a single bit difference seems exhaustive. Hence, working on a single bit difference in a similar direction seems difficult for higher rounds as it needs a powerful computer to study the equations generated at higher rounds.

Further working on multiple difference vectors, in general it spreads more differ-

ences into the state as there are multiple numbers of non zero positions in multiple bit difference vectors. Henceforth, it creates more situations when $\Delta z_t \neq 0$ and generates complicated equations at lower rounds. As a result, it becomes more difficult to analyze for the higher round. In contrast, there could be some difference vectors of multiple weight where the difference generated due to the different non-zero positions cancel each other to result in $\Delta z_t = 0$. Hence, if such a difference vector is chosen, then the attacker can make $\Delta z_t = 0$ for higher rounds and go for attacking for higher rounds.

In our work, we have improved the number of KSA rounds by imposing a two bit difference in the IV, i.e., by using a difference vector of weight two. Let denote the vector $e_{i,j} \in \mathbb{F}_2^{64}, 0 \leq i < j \leq 63$ such that $e_{i,j} = e_i + e_j$. For our work, we choose the difference vector $e_{20,45}$ of weight two from $\binom{64}{2} = 2016$ possibilities. The reason for choosing the specific difference vector $e_{20,45}$ is described in Remark 1.

Remark 1. *For each of* 2016 *possible difference vectors $e_{i,j}$ of weight two, we experimentally checked the probability of the difference $\Delta z_t = z_t + \widetilde{z}_t$ at the output of every round $t, 0 \leq t \leq 41$, for a large number of random key, IV pairs. Hence for each $e_{i,j}$, there are disjoint partitions S_1, S_2 and S_3 of the set $\{0, 1, \cdots, 41\}$ such that $z_t = 0$ for $t \in S_1$, $z_t = 1$ for $t \in S_2$ and z_t is a non-constant function of K, IV for $t \in S_3$. For a chosen $e_{i,j}$, we take action for these three different situations as following.*

- *When $t \in S_1$: Since $z_t = 0$, there is no addition of difference in the state from z_t. We need not to do anything in this situation.*
- *When $t \in S_2$: Since $z_t = 1$ (constant function), it is not possible to put any condition on state bits to stop the propagation of difference into the state. Hence, we prefer to choose such $e_{i,j}$ where $|S_2| = 0$.*
- *When $t \in S_3$: Since z_t is a non-constant function of IV and K, it is possible to put conditions on the bits of IV and/or K such that $z_t = 0$. As per the involvement of bits of IV and K, the conditions are classified as* Type I, Type II

and Type III. *Hence we need to choose such $e_{i,j}$'s such that the size of set S_3 is minimized, which possibly give a minimized set of conditions.*

From the initial rounds (i.e., $0 \le t \le 41$), we need to find the set $S_2 \cup S_3$ containing a few elements. For further refining, our aim is to choose $e_{i,j}$ such that $|S_2| = 0$ and S_3 is a minimized set. We experimentally checked for each possible $e_{i,j}$ for a large set of random K, IV pairs. The experimental result shows that the vectors $e_{20,45}, e_{23,61}, e_{38,62}$ are having minimized set $S_2 \cup S_3$. Further, our distinguisher gives best success rate for the difference vector $e_{20,45}$. Hence, we choose $e_{20,45}$ as the difference vector for the distinguisher.

For the difference vector $e_{20,45}$, the difference probabilities $Pr[\Delta z_t = 0]$ are nonzero for $t = 17, 20, 36, 37, 38$. Therefore, our aim is to find a set of conditions on IV and key bits such that the restriction $\Delta z_{17} = \Delta z_{20} = \Delta z_{36} = \Delta z_{37} = \Delta z_{38} = 0$ is satisfied. The reason towards choosing the restrictions on $\Delta z_{17}, \Delta z_{20}, \Delta z_{36}, \Delta z_{37}, \Delta z_{38}$ is as followed.

Let two instances of the cipher be initialized with IV and $\widetilde{IV} = IV + e_{20,45}$. The states at t-th round are S_t and $\widetilde{S}_t$ respectively. Denote $\Delta S_t = S_t + \widetilde{S}_t, t \ge 0$. The states S_0 and $\widetilde{S}_0$ at the zero-th round differ exactly at two places with probability 1. As the number of rounds increases in the KSA, the number of difference positions increases with a complicated probability distribution. Our goal is to minimize the differences for maximum possible KSA rounds by imposing specific conditions on the bit values in IV.

In the KSA, the keystream bit z_t involves the feedback bits from both the LFSR and the NFSR. The main reason for the transmission of difference into the state bits is the injection of the difference in z_t via these feedback bits. We denote the t-th keystream bits of the cipher with initial state S_0 and $\widetilde{S}_0$ by z_t and $\widetilde{z}_t$ respectively. Since there are differences in the two bits of the initial states, the keystream bits (z_t and $\widetilde{z}_t$) start differing after a certain number of rounds t. The difference of the keystream bits $\Delta z_t = z_t + \widetilde{z}_t$ is a function of the bits of key K and initialization vector IV. The algebraic expression of the function becomes more complicated as the number

of rounds increase. We use SAGE [55] to compute the algebraic expressions of the function Δz_t for $0 \le t \le 41$. The conditions on IV bits are generated by imposing the condition $\Delta z_t = 0$ as follows.

[C0.] Case ($0 \le t \le 41$ and $t \ne 17, 20, 36, 37, 38$): It is observed that $\Delta z_t = 0$ for $0 \le t \le 16, 18 \le t \le 19, 21 \le t \le 35, 39 \le t \le 41$. Hence, we have nothing to impose for these rounds.

[C1.] Case ($t = 17$): In 17-th round, $\Delta z_{17} = P_1(K, IV)$, where $P_1(K, IV)$ is a polynomial involving the bits of K and IV. The algebraic normal from of P_1 is provided in [56]. For a fixed key K, we need to find the set of IV's such that $P_1(K, IV) = 0$. Since finding this set is quite difficult, we choose a subset of IVs by imposing some conditions on the IV bits such that $P_1(K, IV) = \Delta z_{17} = 0$. We follow the method explained in Subsection 3.4.1 to make $\Delta z_{17} = 0$. We set $iv_{47} = iv_{63} = 0$ and $iv_1 = iv_4 + iv_{14} + iv_{24} + iv_{26} + iv_{39}$. With these conditions the equation becomes $\Delta z_{17} = iv_{52} + F_1(K)$, where F_1 is a function involving only the secret key bits. Further, fixing $iv_{52} = F_1(K)$, we get $\Delta z_{17} = 0$. Therefore, having three Type I conditions $iv_{47} = 0$; $iv_{63} = 0$; $iv_1 + iv_4 + iv_{14} + iv_{24} + iv_{26} + iv_{39} = 0$ and one Type II condition $iv_{52} = F_1(K)$, we have a smaller set of initialization vectors IVs where $\Delta z_{17} = 0$.

[C2.] Case ($t = 20$): At this round, $\Delta z_{20} = P_2(K, IV)$, where P_2 is a polynomial involving the bits of K and IV. The algebraic normal form of P_2 is provided in [56]. As similar to Item [C1], we set some conditions on the IV bits, so that $\Delta z_{20} = 0$. Setting $iv_{49} = 0$ and $iv_3 = iv_6 + iv_{23}$, we have the equation $\Delta z_{20} = iv_{28} + F_2(K)$, where F_2 is a function involving only the secret key bits. Further, imposing an extra condition $iv_{28} = F_2(K)$, we have $\Delta z_{20} = 0$. Therefore, we set two Type I conditions $iv_{49} = 0$; $iv_3 + iv_6 + iv_{23} = 0$ and one Type II condition $iv_{28} = F_2(K)$.

[C3.] Case ($t = 36$): In this case, we have $\Delta z_{36} = P_3(K, IV)$, where P_3 is a polynomial

involving the bits of K and IV. As the algebraic expression of P_3 is very large, the algebraic normal form of P_3 is placed at [56]. The same technique (as in Subsection 3.4.1) has been followed to make $P_3 = 0$. To set $\Delta z_{36} = 0$, we fix the conditions $iv_5 = iv_{14} = iv_{48} = 0; iv_2 = iv_{22} = iv_{44} = 1; iv_{15} = iv_{25}, iv_{40} = iv_{53}, iv_{16} = iv_{19} + iv_{23} + iv_{24} + iv_{26} + iv_{41}$. After setting these conditions, we have $\Delta z_{36} = iv_{54} + iv_{27}iv_{54} + iv_{54}F_3(K) + iv_{27}f_1(K) + f_2(K) = iv_{54}(1 + iv_{27} + F_3(K)) + iv_{27}f_1(K) + f_2(K)$.

Here, F_3, f_1 and f_2 are functions on key bits. Further, setting $iv_{27} = F_3(K)$ and $iv_{54} = F_3(K)f_1(K) + f_2(K) = F_4(K)$, we get $\Delta z_{36} = 0$. Therefore, setting nine Type I conditions $iv_5 = iv_{14} = iv_{48} = iv_{15} + iv_{25} = iv_{40} + iv_{53} = 0$; $iv_{16} + iv_{19} + iv_{23} + iv_{24} + iv_{26} + iv_{41} = 0$; $iv_2 = iv_{22} = iv_{44} = 1$; and two Type II conditions $iv_{27} = F_3(K)$; $iv_{54} = F_4(K)$, we get $\Delta z_{36} = 0$.

[C4.] Case ($t = 37$): In 37-th round, we have $\Delta z_{37} = P_4(K, IV)$, where P_4 is a polynomial involving the bits of K and IV. The algebraic normal form of P_4 is available at [56]. To make $\Delta z_{37} = 0$, we impose the conditions $iv_{24} = iv_{46} = iv_{50} = iv_{51} = iv_{62} = 0; iv_{19} = 1; iv_{34} = iv_{43} + iv_{53} + iv_{56}$; $iv_7 = iv_4 + iv_8 + iv_{18} + iv_{21} + iv_{29} + iv_{30} + iv_{59}$. After fixing these conditions, we have $\Delta z_{37} = iv_{53} + F_5(K)$. Now considering $iv_{53} = F_5(K)$, we have $\Delta z_{37} = 0$. Therefore, at the 37-th round, we set the following eight Type I and one Type II conditions.

Type I: $iv_{24} = iv_{46} = iv_{50} = iv_{51} = iv_{62} = 0$; $iv_{19} = 1$; $iv_{34} + iv_{43} + iv_{53} + iv_{56} = 0$; $iv_4 + iv_7 + iv_8 + iv_{18} + iv_{21} + iv_{29} + iv_{30} + iv_{59} = 0$

Type II: $iv_{53} = F_5(K)$.

[C5.] Case ($t = 38$): In this round, we have $\Delta z_{38} = P_5(K, IV)$, where P_5 is a polynomial involving the bits of K and IV. The algebraic normal form of P_5 is available at [56]. To have $\Delta z_{38} = 0$, we impose $iv_{17} = iv_{39} = iv_{42} = iv_{55} = 0$, $iv_8 = iv_{18}, iv_{21} = iv_{30}, iv_{56} = iv_{41} + iv_{43}$. So, we have equation $\Delta z_{38} = iv_{59} + iv_{23}iv_{59} + iv_{59}F_6(K) + iv_{23}f_3(K) + f_4(K) = iv_{59}(1 + iv_{23} + F_6(K)) + iv_{23}f_3(K) + f_4(K)$. Further, imposing

conditions $iv_{23} = F_6(K)$ and $iv_{59} = F_6(K)f_3(K) + f_4(K) = F_7(K)$, we have $\Delta z_{38} = 0$. Finally, for 38-th round, we set the following seven Type I conditions and two Type II conditions to have $\Delta z_{38} = 0$.

Type I: $iv_{17} = iv_{39} = iv_{42} = iv_{55} = iv_8 + iv_{18} = iv_{21} + iv_{30} = iv_{41} + iv_{43} + iv_{56} = 0$

Type II: $iv_{23} = F_6(K)$; $iv_{59} = F_7(K)$.

Table 3.3: Differential status of Grain-v1 for round 0 to round 41

Round (i)	Δz_i	Type-I conditions	Type-II conditions
0 to 16	0	No conditions	No conditions
17	$P_1(K, IV)$	$iv_{47} = iv_{63} = 0$; $iv_1 + iv_4 + iv_{14} + iv_{24} + iv_{26} + iv_{39} = 0$	$iv_{52} = F_1(K)$
18 to 19	0	No conditions	No conditions
20	$P_2(K, IV)$	$iv_{49} = 0; iv_3 + iv_6 + iv_{23} = 0$	$iv_{28} = F_2(K)$
21 to 35	0	No conditions	No conditions
36	$P_3(K, IV)$	$iv_5 = iv_{14} = iv_{48} = iv_{15} + iv_{25} = iv_{40} + iv_{53} = 0$; $iv_{16} + iv_{19} + iv_{23} + iv_{24} + iv_{26} + iv_{41} = 0$; $iv_2 = iv_{22} = iv_{44} = 1$;	$iv_{27} = F_3(K)$; $iv_{54} = F_4(K)$
37	$P_4(K, IV)$	$iv_{24} = iv_{46} = iv_{50} = iv_{51} = iv_{62} = 0$; $iv_{19} = 1; iv_{34} + iv_{43} + iv_{53} + iv_{56} = 0$; $iv_4 + iv_7 + iv_8 + iv_{18} + iv_{21} + iv_{29} + iv_{30} + iv_{59} = 0$	$iv_{53} = F_5(K)$
38	$P_5(K, IV)$	$iv_{17} = iv_{39} = iv_{42} = iv_{55} = iv_8 + iv_{18} = iv_{21} + iv_{30} = iv_{41} + iv_{43} + iv_{56} = 0$	$iv_{23} = F_6(K)$; $iv_{59} = F_7(K)$
39 to 41	0	No conditions	No conditions

Therefore, for a fixed key K, setting the conditions proposed in C1, C2, C3, C4, and C5 on the bits of IV, we will have $\Delta z_t = 0$, for $0 \le t \le 41$. We summarize the difference propagation and required Type I and Type II conditions in Table 3.3. It can be observed that unlike the results in [17, 41–43], in our case there is no t for which $\Delta z_t = 1$, where $0 \le t \le 41$. This provides an advantage to obtain an improved distinguisher by choosing the difference vector $e_{20,45}$.

For the 42-nd round, the algebraic expression of Δz_{42} is very large and complicated on the bits of K and IV. However, the Items [C1, C2, C3, C4, C5] contain 29 Type I conditions and 7 Type II conditions, which are listed below. Hence, with these conditions, we have $\Delta z_t = 0$ for $0 \le t \le 41$.

$$\left\{\begin{array}{c}\text{Type I:}\\ iv_5 = iv_{14} = iv_{17} = iv_{24} = iv_{39} = iv_{42} = iv_{46} = iv_{47} = iv_{48} = iv_{49} = iv_{50} = iv_{51} = iv_{55}\\ = iv_{62} = iv_{63} = 0;\ iv_2 = iv_{19} = iv_{22} = iv_{44} = 1;\ iv_8 = iv_{18};\ iv_{15} = iv_{25};\ iv_{21} = iv_{30};\\ iv_{40} = iv_{53};\ iv_3 = iv_6 + iv_{23};\ iv_{41} = iv_{43} + iv_{56};\ iv_{34} = iv_{43} + iv_{53} + iv_{56};\\ iv_1 = iv_4 + iv_{14} + iv_{24} + iv_{26} + iv_{39};\ iv_{16} = iv_{19} + iv_{23} + iv_{24} + iv_{26} + iv_{41};\\ iv_4 = iv_7 + iv_8 + iv_{18} + iv_{21} + iv_{29} + iv_{30} + iv_{59};\\ \text{Type II:}\\ iv_{52} = F_1(K); iv_{28} = F_2(K); iv_{27} = F_3(K); iv_{54} = F_4(K);\ iv_{53} = F_5(K);\\ iv_{23} = F_6(K); iv_{59} = F_7(K).\end{array}\right\}$$

This set of conditions can further be simplified as

$$\left\{\begin{array}{c}\text{Type I:}\\ iv_5 = iv_{14} = iv_{17} = iv_{24} = iv_{39} = iv_{42} = iv_{46} = iv_{47} = iv_{48} = iv_{49} = iv_{50} = iv_{51} = iv_{55}\\ = iv_{62} = iv_{63} = 0;\ iv_2 = iv_{19} = iv_{22} = iv_{44} = 1;\ iv_1 = iv_4 + iv_{26};\ iv_3 = iv_6 + iv_{23};\\ iv_4 = iv_7 + iv_{29} + iv_{59};\ iv_8 = iv_{18};\ iv_{15} = iv_{25};\ iv_{16} = 1 + iv_{23} + iv_{26} + iv_{41};\\ iv_{21} = iv_{30};\ iv_{34} = iv_{41} + iv_{53};\ iv_{40} = iv_{53};\ iv_{41} = iv_{43} + iv_{56};\\ \text{Type II:}\\ iv_{52} = F_1(K);\ iv_{28} = F_2(K); iv_{27} = F_3(K);\ iv_{54} = F_4(K);\ iv_{53} = F_5(K);\\ iv_{23} = F_6(K);\ iv_{59} = F_7(K).\end{array}\right\} \quad (3.2)$$

The Type II conditions are imposed on seven known IV bits iv_{23}, iv_{27}, iv_{28}, iv_{52}, iv_{53},

iv_{54}, iv_{59} and a set of unknown key bits. For an unknown fixed key K and a chosen IV, if values of $F_i(K)$, $i = 1, 2, \cdots, 7$ match with the values of above mentioned IV bits respectively, then all Type II conditions are satisfied and hence, $\Delta z_t = 0$; $0 \leq t \leq 41$. Consider an unknown random key K and IV satisfying all Type I conditions. Then, there is one possibility out of 2^7 possible assignments of 7 bits iv_{23}, iv_{27}, iv_{28}, iv_{52}, iv_{53}, iv_{54}, iv_{59}, that satisfies the Type II conditions. Since the secret key K is unknown, we have to try for all 2^7 possible assignments of state bits iv_{23}, iv_{27}, iv_{28}, iv_{52}, iv_{53}, iv_{54}, iv_{59} and there is a case for which $\Delta z_t = 0, 0 \leq t \leq 41$. That is, there must be a case for which $Pr[\Delta z_t = 0] = 1$ and for other 127 cases $Pr[\Delta z_t = 0] \leq 1$ for $0 \leq t \leq 41$. Note that there might be many assignments from these non-satisfying 127 assignments where $Pr[\Delta z_t = 0] = 1$ for $0 \leq t \leq 41 - l$ for some small integers l.

For the high values of t, the probability $Pr[\Delta z_t = 0]$ is expected to be $\frac{1}{2}$. But the existence of a case where $Pr[\Delta z_t = 0] = 1$ for $0 \leq t \leq 41$ (or, for $0 \leq t \leq 41 - l$ for some integers l) motivates us to search for non-randomness at higher rounds and to construct a distinguisher. For a random key K, the probability $p_i = Pr[\Delta z_r = 1]$ is calculated for each assignment $A_i, 0 \leq i \leq 127$ of the IV bits $iv_{23}, iv_{27}, iv_{28}, iv_{52}, iv_{53}, iv_{54}, iv_{59}$ for higher value of r. We have observed small non-randomness in some assignments for $r = 112$. As each assignment corresponds to a very small bias, we use the fact to present a distinguisher as discussed in Subsection 3.2.1.

Consider $\mathcal{W} = \sum_{0 \leq i \leq 127, p_i > \frac{1}{2}} \left(p_i - \frac{1}{2}\right)$. If Grain-v1 with 112 KSA rounds generates pseudorandom bits then the value of the $\mathcal{W}$ would be expected as $\mathcal{W}_\phi = 2^7\mathcal{E}$, where $\mathcal{E} = \frac{1}{\sqrt{8\pi N}}$ and N is the size of the sample space. The number of IV bits fixed by the Type I and Type II conditions are $29 + 7 = 36$. Further the two IV bits iv_{20}, iv_{45} can take the complementary values, either $(0,0), (1,1)$ or $(0,1), (1,0)$ to create the differences. There will be $64 - 38 = 26$ free IV bits that can be used to generate 2^{26} samples. Considering the pair of complementary values, we finally obtain the sample size $N = 2^{27}$. Putting $N = 2^{27}$ in Equation (3.1), we have $\mathcal{W}_\phi \approx 0.0022$.

We performed experiments on 2048 many random keys, and it is observed that for

approximately 99% of keys $\mathcal{W} > 0.0022$. This took around 5 days in a machine having 120 processors of 2.8 GHz clock in a multi-user environment. With this experiment, we can distinguish Grain-v1 with 112 KSA rounds from a random source with the success rate of 99% (almost certainty). For cross-checking, we too perform the same experiment for the round 113 and achieved the success rate of 45%, which is close to 50%. To claim the success rate of 45%, we need to run the same experiment for a large number of random keys. This is not possible at this point with the present computational power we are having.

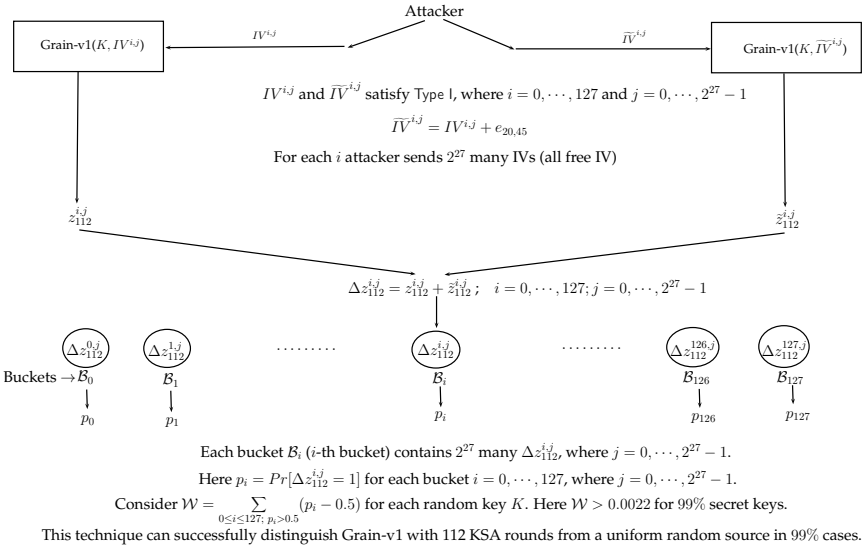

Figure 3.1: Distinguisher for Grain-v1 with 112 KSA rounds

Hence, the distinguisher, designed by selecting the IV differential set of dimension 2 can distinguish the stream cipher Grain-v1 with 112 KSA rounds from a uniform random source with significantly high success rate (approximately 99%).

The proposed distinguisher for Grain-v1 with 112 KSA rounds is presented as following. The pictorial view of our distinguisher is provided in Figure 3.1.

- **A distinguisher for the first keystream bit of Grain-v1 with** 112 **KSA round:**

1. A random key K of 80 bit is generated.
2. An attacker $\mathcal{A}$ is given oracle access to the pusedo-random bit generator, which generates keystream bit by using K.
3. $\mathcal{A}$ selects 64 bits IV (IV), which satisfies Type I, Type II conditions in Equation (3.2).
4. $\mathcal{A}$ constructs another IV, $\widetilde{IV} = IV + e_{20,45}$.
5. $\mathcal{A}$ considers all possible 0/1 values to 7 IV bits which satisfies Type II conditions.
6. For each 0/1 possible values of 7 IV bits (involved in Type II), $\mathcal{A}$ queries 2^{27} many IV and $\widetilde{IV}$ to the oracle.
7. For each $2^7.2^{27} = 2^{34}$ many IV bits, the oracle returns z and $\widetilde{z}$ corresponding to IV and $\widetilde{IV}$.
8. For each 0/1 possible values of those 7 IV bits (involved in Type II) $\mathcal{A}$ segregates the keystream bits into 2^7 many buckets ($\mathcal{B}_i$, $i = 0, \ldots, 127$). Here, each bucket contains 2^{27} many z and $\widetilde{z}$.
9. $\mathcal{A}$ computes the probability $p_i = Pr[z \neq \widetilde{z}]$ for each bucket $\mathcal{B}_i$, $i = 0, \ldots, 127$.
10. $\mathcal{A}$ computes $\mathcal{W} = \sum_{0 \leq i \leq 127, p_i > \frac{1}{2}} \left(p_i - \frac{1}{2}\right)$.
11. If $\mathcal{W} > 0.0022$, then $\mathcal{A}$ claims that the oracle was is Grain-v1 with 112 KSA round to generate the keystream bits. Otherwise $\mathcal{A}$ claims that the oracle is generating the random bits.

It has been experimentally observed that the success rate of the attacker $\mathcal{A}$ is approximately 99%.

3.4.1 Function reduction method

It can be observed from [56], that the ANF of the functions $P_i(K, IV)$, $1 \leq i \leq 5$ are quite complicated. In Section 3.4 we have imposed some Type I and Type II conditions

to get $P_i(K, IV) = 0$ for $1 \leq i \leq 5$. These Type I and Type II conditions are obtained by carefully analyzing the functions. We follow the following steps to get these Type I and Type II conditions.

1. Firstly, we save the complete algebraic expression of the function into a file.

2. We assign 0 or 1 values to some IV bits to simplify the function.

3. Then we replace some IV bits in terms of the linear combination of some other IV bits to get more simplified form of the function.

4. Finally, some IV bits are substituted in terms of the secret key bits to get $P_i(K, IV) = 0$.

Most of these things are done manually. Let us discuss the scenario for $P_2(K, IV)$, as the other functions can be tackled in the same technique. From the algebraic normal form of the function $P_2(K, IV)$ (can be found in [56]), one can observe that the bit iv_{49} is involved in many monomials in the algebraic normal form. Hence, the algebraic normal form is made simpler by substituting $iv_{49} = 0$. Further, substituting $iv_3 + iv_6 + iv_{23} = 0$, the algebraic normal form of the function $P_2(K, IV)$ becomes as simple as $iv_{28} + F_2(K)$. Finally, the Type II condition $iv_{28} = F_2(K)$ helps us to achieve $P_2(K, IV) = 0$. We have followed a similar method for the other complicated functions, and naturally, for the functions P_3, P_4, P_5, it took quite a bit of effort. Software to handle these issues may provide even better results that may be explored in the future.

3.5 Distinguisher for 114 KSA round of Grain-v1

In this section, we design a distinguisher on Grain-v1 with 114 KSA rounds. The idea is to increase the number of round follows in two steps.

- In the first step, we put conditions on $iv_{63} = iv_{62} = \cdots = iv_{63-j} = 1$ for some $j \geq 0$ and generate conditions as discussed in Section 3.4 to obtain a distinguisher at the r-th round.

- Since we are able to run the inverse of KSA for $j+1$ rounds as $iv_{63} = \cdots = iv_{63-j} = 1$, with some more conditions on key bits, i.e., Type III conditions, we can design a distinguisher for $(r + j + 1)$ KSA rounds with some conditions on key space.

For our work, we first put three Type I conditions $iv_{63} = iv_{62} = iv_{61} = 1$ on last three IV bits. Then following a similar technique as in Section 3.4, we choose the same difference vector $a = e_{20,45}$ and generate the conditions as follows. We have followed the same technique (as in Section 3.4.1) to construct the following Type I and Type II conditions.

[C0.] Case ($0 \leq t \leq 41$ and $t \neq 17, 20, 36, 37, 38$): It is observed that $\Delta z_t = 0$ for $0 \leq t \leq 16, 18 \leq t \leq 19, 21 \leq t \leq 35, 39 \leq t \leq 41$. Hence, we need not require any additional condition for these rounds.

[C1.] Case ($t = 17$): In this round, $\Delta z_{17} = Q_1(K, IV)$, where Q_1 is a polynomial involving the bits of K and IV. From now on we will use the term Q_i in general for this. Imposing the conditions on the IV bits as

Type I: $iv_{46} = iv_0 + iv_3 + iv_{25} = 0$;

Type II: $iv_{42} = G_1(K)$,

we have $\Delta z_{17} = Q_1(K, IV) = 0$.

[C2.] Case ($t = 20$): In this round, $\Delta z_{20} = Q_2(K, IV)$. Similarly, imposing the following conditions on IV bits

Type I: $iv_{49} = iv_3 + iv_6 + iv_{23} = 0$;

Type II: $iv_{28} = G_2(K)$,

we get $\Delta z_{20} = Q_2(K, IV) = 0$.

[C3.] Case ($t = 36$): Here, $\Delta z_{36} = Q_3(K, IV)$. Here, we set the following conditions on IV bits to make $\Delta z_{36} = Q_3(K, IV) = 0$.

Type I: $iv_5 = iv_{48} = iv_{47} = 0, iv_2 = 1,\ iv_{25} = iv_{15}, iv_{40} = iv_{53}, iv_{22} = iv_{44}, iv_4 = iv_{26},\ iv_1 = iv_{16} + iv_{19} + iv_{23} + iv_{26} + iv_{39} + iv_{41}$;

Type II: $iv_{27} = G_3(K), iv_{54} = G_4(K)$.

[C4.] Case ($t = 37$): Here, $\Delta z_{37} = Q_4(K, IV)$. We set the following conditions on IV bits to make $\Delta z_{37} = Q_4(K, IV) = 0$.

Type I: $iv_{24} = iv_{50} = iv_{51} = 0,\ iv_{16} = iv_{23} + iv_{26} + iv_{41},\ iv_7 = iv_8 + iv_{16} + iv_{18} + iv_{21} + iv_{23} + iv_{29} + iv_{30} + iv_{34} + iv_{41} + iv_{43} + iv_{44} + iv_{53} + iv_{56} + iv_{59}$;

Type II: $iv_{53} = G_5(K)$.

[C5.] Case ($t = 38$): In this round, $\Delta z_{38} = Q_5(K, IV)$. We set the following conditions to make $\Delta z_{38} = Q_5(K, IV) = 0$.

Type I: $iv_{34} = 0,\ iv_8 = iv_{17} + iv_{18} + iv_{21} + iv_{30} + iv_{34} + iv_{43} + iv_{44} + iv_{55},\ iv_{17} = iv_{55}, iv_{19} = iv_{39}, iv_{23} = iv_{41} + iv_{44}$;

Type II: $iv_{56} = G_6(K), iv_{59} = G_7(k)$.

Hence for a fixed key K, if the IV bits satisfy the conditions [C1, C2, C3, C4, C5] and the initial conditions $iv_{63} = iv_{62} = iv_{61} = 1$ then $\Delta z_t = 0$ for $0 \leq t \leq 41$. Complete set of

Type I and Type II conditions on IV bits is given below.

$$\left\{\begin{array}{c} \text{Type I:} \\ iv_1 = iv_5 = iv_{24} = iv_{34} = iv_{46} = iv_{47} = iv_{48} = iv_{49} = iv_{50} = iv_{51} = 0; \\ iv_2 = iv_{61} = iv_{62} = iv_{63} = 1;\ iv_0 = iv_3 + iv_{25}; iv_3 = iv_6 + iv_{23}; iv_4 = iv_{26}; \\ iv_7 = iv_{26} + iv_{29} + iv_{53} + iv_{56} + iv_{59};\ iv_8 = iv_{18} + iv_{21} + iv_{30} + iv_{43} + iv_{44}; \\ iv_{15} = iv_{25};\ iv_{16} = iv_{26} + iv_{44};\ iv_{17} = iv_{55};\ iv_{19} = iv_{39}; iv_{22} = iv_{44} \\ iv_{23} = iv_{41} + iv_{44}; iv_{40} = iv_{53}; \\ \text{Type II:} \\ iv_{42} = G_1(K);\ iv_{28} = G_2(K); iv_{27} = G_3(K);\ iv_{54} = G_4(K);\ iv_{53} = G_5(K); \\ iv_{56} = G_6(K);\ iv_{59} = G_7(k). \end{array}\right\} \tag{3.3}$$

Since the last 3 bits of the IV are 1 (i.e., $iv_{63} = iv_{62} = iv_{61} = 1$), there is a possibility to go for $t, (t \leq 3)$ inverse KSA rounds to have another valid initial state keeping the last 16 bits of the initial state S_0 (i.e., the padding bits) as 1. In this manner, one can increase the round number to $(112+t)$ rounds with few more conditions on IV and K bits. We discuss the possibility of such improvement of round numbers for $t = 1, 2, 3$ as follows.

For the first step of inversion in KSA (i.e., $t = 1$), we need to make $z + s_0 = 1$ where z is the output bit and s_0 is the feedback bit of the LFSR for inverse KSA. For this, we need to set the following conditions on key and IV bits to have $z = 1, s_0 = 0$.

$$k_0 = k_1 + k_3 + k_9 + k_{30} + k_{42} + k_{55}, \tag{3.4}$$

$$iv_{12} = iv_{37} + iv_{44} + 1. \tag{3.5}$$

Following the same process for the second inverse round of the KSA, we set the

conditions as

$$iv_{44} = 0;\ iv_{11} = iv_{21} + iv_{36} + iv_{41} + iv_{60};\ iv_{41} = 0. \quad (3.6)$$

$$\begin{aligned} k_{79} = k_1 + k_2 + k_3 + k_9 + k_{13} + k_{20} + k_{27} + k_{29} + k_{30} + k_{32} + k_{36} + k_{41} + k_{42} + k_{44} + \\ k_{51} + k_{54} + k_{55} + k_{59} + k_{61} + k_8k_{14} + k_{32}k_{36} + k_{59}k_{62} + k_{20}k_{27}k_{32} + k_{44}k_{51}k_{59} + \\ k_8k_{27}k_{44}k_{62} + k_{14}k_{20}k_{59}k_{62} + k_{32}k_{36}k_{51}k_{59} + k_8k_{14}k_{20}k_{27}k_{32} + \\ k_{36}k_{44}k_{51}k_{59}k_{62} + k_{20}k_{27}k_{32}k_{36}k_{44}k_{51}. \end{aligned} \quad (3.7)$$

If we run one more inverse KSA round, then a difference vector for the secret key K is formed. Since the difference is not allowed for the secret key K, we cannot proceed with the third KSA inverse (i.e., $t = 3$). This scenario has been discussed in Section 3.5.1. The inclusion of two Type III conditions (Equation (3.4) and Equation (3.7)) reduces the keyspace by a dimension of two (i.e., one-fourth of the original keyspace). Including this four Type I and two Type III conditions with the constraints in Equation (3.3) on the key and IV bits and further imposing two inverse KSA rounds on the state we have the initial state S_0 for the Grain-v1. Then starting from the initial state S_0, we have a non-randomness in the first keystream bit of Grain-v1 with 114 KSA rounds.

In this case, the total number of free IV variables is 26 and Type II conditions is 7. Hence, the value of $\mathcal{W}_\phi = 2^7 \times \frac{1}{\sqrt{8\pi N}} \approx 0.00312$ where $N = 2^{26}$ is the size of the sample space (see Section 3.2.1 for the calculation of $\mathcal{W}_\phi$).

Further, to show a non-randomness in 114-th round of KSA, as in Section 3.4, we compute the sum $\mathcal{W} = \sum_{0 \le i \le 127, p_i > 0.5} (p_i - 0.5)$ where $p_i = Pr[\Delta z_{114} = 1]$ corresponds with the i-th assignment $\mathcal{A}_i$ $(0 \le i \le 127)$. Each assignment $\mathcal{A}_i$ is an assignment of binary value to the 7 IV bits involved in the Type II conditions.

From the experiment with 2048 random keys, it is observed that for approximately 73% cases $\mathcal{W} < \mathcal{W}_\phi = 0.00312$. Therefore, like 112 KSA rounds, we can design a distinguisher for the first keystream bit of the Grain-v1 with 114 KSA rounds with a

success rate of approximately 73% in weak key setup as the key relation provided in Type III conditions. For further cross-checking, we performed the same experiment for 115 rounds and noted that the experimental success rate of distinguishing became approximately 56%. That is, for 2^{78} keys, the first keystream bit of Grain-v1 with 114 KSA rounds can be distinguished from a random bit with a success rate of approximately 73%.

The success rate for the case of Grain-v1 with 115 KSA rounds is approximately 56%. To claim this small success rate to use for designing a distinguisher, we need to run the same experiment for a large number of random keys, which is quite impossible for us with our present computational power.

3.5.1 Non-randomness of Grain-v1 with 116 KSA round with one bit difference in keys

This section extends the distinguisher from 114 KSA round to 116 round. We first introduce one extra Type I condition $iv_{60} = 1$, then start the inverse KSA of Grain-v1 with the same setup presented in Section 3.5. As we have already performed 2 inverse KSA rounds, the IV difference bits moved to 22-nd and 47-th positions of the LFSR. After one more round of inverse KSA, these difference bits will move to 23-rd and 48-th positions of the LFSR. During the inverse KSA, the bit at the 23-rd position of LFSR is involved in the computation of the linear feedback bit of the LFSR. Hence, it will flip the feedback bit of the LFSR in this inverse KSA round. Further, this feedback bit of LFSR is involved linearly in the feedback bit computation of NFSR. So it will also flip the feedback bit of the NFSR. After this inverse round the state bits $\{s_0, s_{23}, s_{48}\}$ and $\{b_0\}$ of the present state of the cipher are flipped. As we have set $iv_{61} = 1$ (in Section 3.5), the last 16 bits of the LFSR remain valid (i.e., all 1). For this one round of inverse KSA we set some conditions on IV bits and secret key bits to make $z + s_0 = 1$ with $z = 1$ and $s_0 = 0$ (as in Section 3.5). The following conditions

are introduced here,

$$iv_{43} = 0;\ iv_6 = iv_{15} + 1;\ iv_{10} = iv_{20} + iv_{35} + iv_{59}; \quad (3.8)$$

$$\begin{aligned} k_{78} = {} & k_2 + k_3 + k_8 + k_9 + k_{12} + k_{19} + k_{26} + k_{28} + k_{29} + k_{30} + k_{31} + k_{35} + k_{40} + k_{41} + \\ & k_{42} + k_{43} + k_{50} + k_{53} + k_{54} + k_{55} + k_{58} + k_{60} + k_7k_{13} + k_{31}k_{35} + k_{58}k_{61} + \\ & k_{19}k_{26}k_{31} + k_{43}k_{50}k_{58} + k_7k_{26}k_{43}k_{61} + k_{13}k_{19}k_{58}k_{61} + k_{31}k_{35}k_{50}k_{58} + \\ & k_7k_{13}k_{19}k_{26}k_{31} + k_{35}k_{43}k_{50}k_{58}k_{61} + k_{19}k_{26}k_{31}k_{35}k_{43}k_{50}. \end{aligned} \quad (3.9)$$

Further, we run the inverse KSA for one more round. It can be observed that iv_{60} was free for the distinguisher on 114-th round (presented in Section 3.5), but here we have set $iv_{60} = 1$. For the second inverse KSA round, the flipped bit at NFSR will move to $\{b_1\}$ of the present state of NFSR. As the NFSR bit $\{b_1\}$ is involved in the computation of keystream bit, the keystream bit z will be flipped (i.e., $\Delta z = 1$) in this round. As a result, the linear feedback bit of the LFSR of this inverse KSA round will also be flipped. Due to the linear involvement of both the keystream bit and the linear feedback bit in the computation of the nonlinear feedback bit of the NFSR, the nonlinear feedback bit remains unaffected in this inverse KSA round. After this inverse KSA round, state bits $\{s_0, s_1, s_{24}, s_{49}\}$ of current state of LFSR and state bit $\{b_1\}$ of current state of NFSR are flipped. The last 16 bits of LFSR remain valid (i.e., all 1) as we have set $iv_{60} = 1$. In this inverse KSA round we also set following conditions on IV bits and secret key bits to make $z + s_0 = 1$, where $z = 1$ and $s_0 = 0$

(as in Section 3.5).

$$iv_{21} = iv_{42} + 1; iv_9 = iv_{39} + iv_{58}; \quad (3.10)$$

$$k_{59} = 0; \quad (3.11)$$

$$\begin{aligned} k_{77} = {} & k_1 + k_2 + k_7 + k_8 + k_{11} + k_{18} + k_{25} + k_{27} + k_{28} + k_{29} + k_{30} + k_{34} + k_{39} + \\ & k_{40} + k_{41} + k_{42} + k_{49} + k_{52} + k_{53} + k_{54} + k_{57} + k_6k_{12} + k_{30}k_{34} + k_{57}k_{60} + \\ & k_{18}k_{25}k_{30} + k_{42}k_{49}k_{57} + k_6k_{25}k_{42}k_{60} + k_{12}k_{18}k_{57}k_{60} + k_{30}k_{34}k_{49}k_{57} + \\ & k_6k_{12}k_{18}k_{25}k_{30} + k_{34}k_{42}k_{49}k_{57}k_{60} + k_{18}k_{25}k_{30}k_{34}k_{42}k_{49}. \end{aligned} \quad (3.12)$$

Now to go back further, we need to set $iv_{59} = 1$, which is not possible as this bit is involved in Type II conditions (see Equation (3.3)). Here we have allowed 1 bit difference in the state of the NFSR, i.e., we have allowed 1 bit difference in the secret key bits for 116 rounds. For extra 2 inverse KSA rounds, 3 extra Type III and 6 extra Type I conditions (including $iv_{60} = 1$) are introduced. Here, we consider two states, which differ only at $\{b_1, s_0, s_1, s_{24}, s_{49}\}$ as the initial state of two ciphers. After that we perform 116 KSA rounds on both the ciphers. With all these Type I, Type II and Type III conditions we have observed the following non-randomness in Δz_{116} after 116 KSA rounds.

It can notice that the sample size is reduced to 2^{20}. With this we calculate $\mathcal{W}_\phi$, which is approximately 0.02493. Now we compute $\mathcal{W} = \sum_{0 \le i \le 127, p_i > 0.5} (p_i - 0.5)$ where $p_i = Pr[\Delta z_{116} = 1]$ corresponds with i-th assignment $\mathcal{A}_i$ $(0 \le i \le 127)$.

For each key we compute $\mathcal{W}$ and compare with $\mathcal{W}_\phi$ (≈ 0.02493). We perform this experiment for 4096 many random keys and it has been observed that for 62% cases $\mathcal{W} < \mathcal{W}_\phi (\approx 0.02493)$. Hence Grain-v1 with 116 KSA rounds can be distinguished in weak key setup with the 1 bit difference in the secret key and 4 bits difference in the IV. Further to cross-check our distinguisher, we perform the same experiment for 117 round, but the success rate is 52% (which is very close to 50%). To claim this success chance ($\approx 52\%$), we need to repeat this experiment for a large number of random keys,

which is an impossible task to verify with our present computation power.

3.6 Comparison

The comparison of the existing attacks and our present work is presented in the Table 3.1 and Table 3.2. However, one may immediately note that all the existing works in this direction are based on the difference vector of weight 1, i.e., the one-bit difference in the IV bits. In each case, the number of rounds is improved irrespective of the improvement distinguishing success chance, the dimension of IV space, keyspace, and query space. The dimension of IV space is equal to (64 – the number of Type I and Type II conditions) and the dimension of keyspace is equal to (80 – the number of Type III conditions). Further, the query space dimension is proportional to (the number of Type II conditions+1) as in each case; we need to run the cipher with the same key and two different IVs.

3.7 Conclusion

In this chapter, we have introduced distinguishers for Grain-v1 with 112 and 114 KSA rounds. The first one can distinguish Grain-v1 with 112 KSA rounds from a random source with a 99% success rate. The second one can distinguish Grain-v1 with 114 KSA rounds from a random source with a 73% success rate in a weak key setup for one-fourth of all the keys. Here, we have used the difference vector of weight 2 to improve the number of rounds for the first time. The analysis in certain cases is indeed complicated and required manual intervention rather than writing computer programs. Finally, the distinguisher for 114 rounds could be extended to 116 rounds with 1 bit and 4 bits differences in key and IV, respectively. The success rate of this distinguisher is 62%.

Chapter 4

Conditional Cube Testers for Grain-128a of Reduced KSA Rounds

4.1 Motivation

In recent days, cube attacks and cube testers [Subsection 2.3.3] are extensively used for cryptanalysis of the stream ciphers. These attacks exploit the ANF of the output function of the cipher to find its weaknesses. There is no proper method to find a suitable cube for a cipher. Several heuristic methods to find suitable cubes are available in the literature [34, 57, 58]. These methods are used to find a suitable cube such that the corresponding superpoly is highly nonrandom at lower rounds. The nonrandomness is further transmitted to higher rounds by introducing few modifications in the cube set and imposing some conditions on other state variables. The imposed conditions are set up by a careful study of the algebraic properties of the superpoly at different rounds. In this process, some key bits may be required for conditioning in some cases. If at least one key bit is used for conditioning, then the tester is in

weak key setup. Otherwise, it is in *single key setup*. We observed that if the Boolean function of a cipher is sparse, then finding a suitable cube is easy. For instance, the Boolean function in Grain-128a is sparser than the Boolean function in Grain-v1. Hence, we expect that the cube attack is more effective on Grain-128a than Grain-v1. This chapter has analyzed Grain-128a to find a cube tester for implementing a distinguishing attack on it.

4.1.1 Our Contributions

The adversary increases the cube dimension by adding the IV variables using different methods to increase the reduced KSA round in cube attacks. This chapter presents a heuristic method to select a suitable cube for any nonlinear filter-based stream cipher by combining the previous techniques. The previous methods are based on the maximum initial zero and maximum last zero strategies. In the maximum initial zero strategy, the state variables for which the number of rounds from the initial round such that the Pr(superpoly = 1) = 0 for each round is maximum are chosen. Whereas the maximum last zero strategy selects the state variables for which the last round such that Pr(superpoly = 1) = 0 is maximum. We introduce another strategy based on a maximum α round, where α is a small number close to 0. This strategy selects the state variables for which the last round such that $0 < \Pr(\text{superpoly} = 1) < \alpha$ is maximum. Details of these techniques are discussed in Section 4.2.1. Further, the non-constant superpolies for the first few rounds are made as zero superpoly by imposing conditions on appropriate state bits. Our heuristic method uses the three strategies mentioned above and the conditions on state variables to find a suitable cube. Using the heuristic, we design a cube tester that can distinguish the Grain-128a of 191 and 201 KSA rounds in the single key and the weak key setup, respectively, which are the highest round till now and using a small dimensional cube. The paper [59] presents a nonrandomness detector on Grain-128a of 203 KSA round using the cubes on key and IV variables. This nonrandomness detector is a weaker attack than the distinguishing

attack in the weak key setup as the cube variables in the cube attacks are taken from the IV variables only. The comparisons of the number of rounds in Grain-128a of our result with the previous attacks are presented in Table 4.1.

Table 4.1: Comparison of the number of rounds in different attacks

Attack Type	Attack Name	Round	Dimension	Time complexity
Single key	Conditional differential attack [50]	169	1*	$2^{46.25}$
	Chosen IV statistical attack [60]	169	22	2^{26}
	Chosen IV statistical attack [61]	171	25	$2^{36.64}$
	Cube tester [58]	177	33	2^{44}
	Fast correlation attack [62]	256	NA	$2^{115.4}$
	Cube tester[our result]	191	5	$2^{33.86}$
Weak key	Conditional differential attack [50]	195	1*	$2^{31.25}$
	Cube tester [58]	189	6	-
	Nonrandomness detectors [59]	203	38†	-
	Cube tester[our result]	201	5	$2^{33.86}$

∗: Dimension of difference vector in differential attack.
†: Dimension of the cube containing key and IV bits.

We also tested our technique over Grain-128 and achieved good results by using small dimensional cubes. In the single key setup, the previous attacks show bias at $207, 237$ and 250 KSA rounds using the cubes of dimensions $12, 40$ and 50, respectively. Our attack shows a bias at 207 and 235 KSA rounds using a cube of dimension 7 for the single and the weak key setup, respectively. Table 4.2 presents the comparison with the previous attacks. From this experiment and comparison on Grain-128, we further assure that our technique (i.e., Algorithm 6) to find a cube works well on Grain-like ciphers.

Table 4.2: Comparison with previous attack on Grain-128

Attack Type	Attack Name	Round	Dimension	Time complexity
Single key	Cube tester [58]	207	12	2^{23}
	Cube tester [58]	236	33	2^{44}
	Cube tester [63]	237	40	-
	Cube tester [64]	256	50	2^{90}
	Dynamic cube attack [65]	207	19	2^{31}
	Dynamic cube attack [65]	250	37	2^{101}
	Conditional differential attack [66]	215	13^{*}	2^{25}
	Cube tester[our result]	205	6	$2^{28.22}$
	Cube tester[our result]	207	7	$2^{35.86}$
Weak key	Nonrandomness detectors[59]	256	$25^{\dagger}$	2^{28}
	Cube tester[our result]	229	6	$2^{34.22}$
	Cube tester[our result]	235	7	$2^{34.69}$

$*$: Dimension of difference vector in differential attack.
†: Dimension of the cube containing key and IV bits.

4.1.2 Organisation

Section 4.2 explains a cube tester method by using a new cube searching strategy. This section splits into three subsections. Subsection 4.2.1 explains the technique to search for a suitable cube for any cipher. Subsection 4.2.2 discusses the conditional cube tester and two different setups of analysis. Our heuristic for cube searching is presented in Subsection 4.2.3. Section 4.3 presents the cube tester for Grain-128a. Subsection 4.3.1 presents a structural analysis of Grain-128a to find some conditional bits for reducing the nonlinear degrees of involved Boolean functions. The above method is used to find suitable cubes for Grain-128a in Subsection 4.3.2. The outcomes of the distinguishing attack on Grain-128a by using those cubes for the cube testers are presented in Subsection 4.3.3. Section 4.4 presents the cube tester for Grain-128.

Subsection 4.4.1 presents the study of our technique over Grain-128. The comparison of the obtained results with the previous results is presented in Section 4.5. In the end, Section 4.6 concludes the chapter.

4.2 Cube tester by using a cube searching method

The keystream bits of a stream cipher are a function in terms of key bits and IV bits. The IV bits are considered public variables, i.e., an adversary can choose bits from IV and get the corresponding output for an unknown key. The adversary's prime goal in the distinguishing attack model is to distinguish the cipher from a random source. In this model, the adversary is allowed to choose some IV bits as cube variables. Then corresponding to an unknown key, the adversary can get the output bits for each assignment of cube variables. From these output bits, the adversary can perform cube sum (as in Equation (2.20)) to check the randomness of the corresponding superpoly. If the superpoly is distinguishable from a random function, then the cipher can be distinguished from the random source.

4.2.1 Cube searching techniques

Finding a proper cube is the most crucial task in cube attacks. The nonrandomness properties of superpoly at lower rounds are exploited to get a proper cube. In [34, 57], the cubes whose superpoly is 0 (zero function) are used to form a cube tester at a higher round. Stankovski [57] studied the cubes at the round, which is the last round of the first continuous zero superpolies (i.e., the maximum initial round of zero superpolies). Sarkar et al. [34] studied the cubes at the round, which has the last zero superpoly. The presence of some kind of nonrandomness among the state variables in the cube results in a zero superpoly. We expect that these state variables appear in some later rounds, which form a bias. Further adding to this idea, we have studied cubes that form superpolies of very low weight. Such cubes with highly

biased superpoly (i.e., having very low weight) are expected to lead us to get cubes with biased superpoly at higher rounds. Further extending the cube's dimension may not always give the best results as the attack complexities increase. So the challenge is to find the small dimensional cubes which give the bias in higher rounds. We have included three ideas with the experiments in our study to find cubes that show bias in superpoly at higher rounds. Three techniques are used for searching cubes are given below.

1. **Maximum initial zero**: In general, the superpolies formed by the cubes are zero Boolean functions for some initial rounds. In this technique, the cubes are chosen, which form zero superpoly for the highest time continuously. The bias is further extended for higher rounds by choosing new cubes from the variables of selected cubes. Stankovski [57] used this idea to choose a cube of one dimension which gives maximum continuous zero superpoly. By pairing the cube variable with other variables, the author found a cube of dimension 2, which gives continuous zero superpoly. Iteratively, one can attack at higher rounds by adding one dimension each time until there is no further possibility to increase the round. This technique is known as the GreedyAddOneBit heuristic, and the cube tester is known as a zero-sum distinguisher. Using this technique, Stankovski [57] constructed a cube tester for several stream ciphers and block ciphers. The same idea can be used for the GreedyAddTwoBits heuristic as well.

2. **Maximum last zero**: By extending the maximum initial zero ideas, Sarkar et al. [34] studied cubes that form zero superpoly at the highest round. They studied cubes with the maximum number of zero superpolies. Using this technique, they could construct cube testers for Trivium and Trivia-SC.

3. **Maximum last** α: In addition to the above two techniques, we introduce the maximum last α technique to find a cube that shows the presence of bias at a

higher round. It is found that the superpolies of some cubes at higher rounds output 1 with very less probability. We expect that such types of cubes can forward the bias to a higher round for the following reason.

A superpoly is a constant (i.e., 0 or 1) implies that the variables in the superpoly are not present in the ANF of the polynomial at that round. Once the variables are involved (suddenly) in the ANF of the superpoly at a higher round, the superpoly function may suddenly become a balanced function. Therefore, in some cases, the maximum initial zero and maximum last zero techniques may not carry the bias for higher rounds. If a superpoly $p_{s(I)}$ is highly unbalanced (i.e., $\Pr(p_{s(I)} = 1)$ is nonzero but a very small number α) at some round, then some variables are present in the ANF of the polynomial. In this case, the superpoly shows a high bias (i.e., $(0.5 - \alpha)$) with the involvement of some variables in the ANF. At higher rounds, although the algebraic manipulations occur in the ANF of superpoly, the existence of variables in the ANF of the superpoly does not happen suddenly. Hence we expect that the bias gets forward to the higher round and the superpoly slowly becomes a balanced polynomial.

Now having a bound α (very small), the cubes producing superpoly $p_{s(I)}$ such that $0 < \Pr(p_{s(I)} = 1) < \alpha$ can be interesting to study. Hence, those cubes whose superpoly $p_{s(I)}$ satisfying $0 < \Pr(p_{s(I)} = 1) < \alpha$ for the highest round are chosen. Further, the attack can be improved to the higher rounds by adding new variables with the selected cubes. Here we used to make non-cube IV variables as zero while searching for the maximum last α round by varying keys.

It can not be said that which one of the above three techniques works best to find cube testers. Exploiting our experimental result, we have properly used the combination of three techniques to find cubes for higher rounds. We further use conditions on variables along with the cube searching techniques to attack for a higher round.

4.2.2 Conditional cube tester

We impose some conditions on state variables to increase the distinguishing rounds in the cube tester. Such a distinguishing attack is called a conditional cube tester. Imposing conditions on some variables, some initial non-constant superpolies can be converted to zero superpoly. By doing this, it is expected that the bias can be extended for some more rounds. Imposing conditions is a tough task as the number of monomials in the ANF of polynomials increases highly as round increases. Based on conditions on key and IV bits, we have two different setups as following.

1 **Single key setup:** Only IV bits are fixed (or, termed as conditioned) and $(K, \overline{IV})$ pairs are taken as random, where $\overline{IV}$ represents the remaining IV bits.

2 **Weak key setup:** At least one key bit is fixed and $(\overline{K}, \overline{IV})$ pairs are taken as random, where $\overline{K}$ and $\overline{IV}$ represent the remaining key and IV bits respectively. As conditioning on key bits may not be practical, the nonrandomness of the stream cipher is tested in this setup.

4.2.3 Our method

In our approach, we use all three techniques and conditional techniques to search for a cube that shows bias at a higher round of Grain-128a. As described in Section 4.2.2, we enlist the conditions for first some rounds (as high as we can compute using SAGE [55]) for every 1-dimensional cube. When we add a variable for increasing the cube's dimension by one, we impose all the conditions associated with the cube variable. Then we extend the cube's dimension by combining three techniques such that distinguishing round is as high as possible and choose the cube at that round. The chosen criteria are discussed in Section 4.3.2 for Grain-128a. The searching process of the cubes is described next.

Step-1: A nonzero threshold α (very small) is fixed, which is chosen from the

prior experiments. All 1-dimensional cubes C_I whose superpoly $p_{s(I)}$ satisfying $0 < \Pr(p_{s(I)} = 1) < \alpha$ in at least one round are collected. Denote the set of variables of these 1-dimensional cubes as B_α. The further cubes of a higher dimension will be searched from the variables in B_α. Hence, the domain of cube variables is restricted to a smaller set of variables B_α.

Then for each cube $C_I = \{s_i\}, s_i \in B_\alpha$, we find the conditions on key and IV bits such that $p_{s(I)}$ becomes zero function for first l rounds (except at those rounds where $p_{s(I)} = 1$). The number l is the maximum round such that we could compute the polynomials at the round using SAGE software. Let CND_I be the set of conditions for the cube C_I.

Step-2: The 1-dimensional cubes C_I from B_α which satisfy maximum last α round (i.e., the last round satisfying $0 < \Pr(p_{s(I)} = 1) < \alpha$ is maximum among all state variables in B_α) are selected. These 1-dimensional cubes are expanded to bigger cubes by adding some variables iteratively from B_α as described in the following step.

Step-3: An extra variable from the remaining variables in B_α is added with each selected cube in the previous step and its behavior is studied. In this way, the cubes with one more dimension can be searched. The stored conditions corresponding to the cube variables are imposed. Note that we can not consider those cubes where two conditions conflict with each other. We observe the maximum initial zero round, maximum last zero round, and maximum last α round for the superpoly of each new cube. The cubes satisfying at least two cases are chosen. If no cube satisfies two cases, then the cubes satisfying the maximum last α round are considered. This technique is repeated until there is no improvement in the round.

The selected cubes are used with different scenarios (e.g., single key, weak key) to find bias at higher rounds. The explanation for different scenarios in the case of Grain-128a

is presented in Section 4.3.3.

The algorithmic form of the above strategy is presented in Algorithm 6. The following notations are used in the algorithm.

- $R^{\texttt{MIZ}}_{\texttt{CUBE}}$ is the maximum initial zero round for the cube CUBE after imposing conditions CND_I for each 1-dimensional cube $C_I \in$ CUBE.

- $R^{\texttt{MLZ}}_{\texttt{CUBE}}$ is the maximum last zero round for the cube CUBE after imposing conditions CND_I for each 1-dimensional cube $C_I \in$ CUBE.

- $R^{\texttt{ML}\alpha}_{\texttt{CUBE}}$ is the maximum last α round for the cube CUBE after imposing conditions CND_I for each 1-dimensional cube $C_I \in$ CUBE.

$\texttt{CUBE}_j$ is defined by adding an extra variable s_j from B_α with the previous cube CUBE such that no condition for the cube $\{s_j\}$ fixes a variable in CUBE and conflicts with earlier conditions. Then the conditions for each cube variable $s_i \in \texttt{CUBE}_j$ is applied and the rounds $R^{\texttt{MIZ}}_{\texttt{CUBE}_j}$,$R^{\texttt{MLZ}}_{\texttt{CUBE}_j}$ and $R^{\texttt{ML}\alpha}_{\texttt{CUBE}_j}$ are stored for choosing cubes of one dimension more.

The value of α and l can be chosen as described in Step-1. The statements from 2 to 17 in Algorithm 6 select the possible state variables satisfying α condition (i.e., those satisfy $\Pr(p_{s(I)} = 1) < \alpha$) and store those variables in B_α. Since the value of α is very small (i.e., the bias $(0.5 - \alpha)$ is high), the experiment for finding probability in the statement number 7 does not need a very large number of samples for a high confidence level. Now the domain of cube variables is restricted to B_α. The variables max_round and max_state_index store the maximum round where α test is satisfied and the index of the state variable in the cube respectively. The constant KSA_round is the number of rounds in the KSA of the cipher. Since in these steps, we are searching the state variables satisfying α condition, the required time complexity is $T_1 = O(\frac{1}{pq^2} \times \text{No. of IV bits} \times KSA\_round)$ where $p(1-q) = \alpha$ as in Section 2.3.1.3. Since α is very small, the value of $\frac{1}{pq^2}$ is too small.

Algorithm 6: Algorithm to find best cube.

Input : α, l, and m = number of IV bits
Output: A cube CUBE and $dim(\texttt{CUBE})$.

1 Set $B_\alpha = \emptyset, max\_round = 0, max\_state\_index = -1$;
2 **for** i *from* 0 *to* $m-1$ **do**
3 Select the cube $C_I = \{s_i\}$; $R_{C_I}^{\texttt{ML}\alpha} = -1$;
4 **for** j *from* 0 *to* KSA_round **do**
5 Consider the superpoly $p_{s(I)}$ at the j-th round with cube C_I;
6 **if** $(0 < \Pr(p_{s(I)} = 1) < \alpha)$ **then**
7 $R_{C_I}^{\texttt{ML}\alpha} = j$
8 **end**
9 **end**
10 **if** $R_{C_I}^{\texttt{ML}\alpha} > -1$ **then**
11 $B_\alpha = B_\alpha \cup \{s_i\}$;
12 **if** $R_{C_I}^{\texttt{ML}\alpha} > max\_round$ **then**
13 $max\_round = R_{C_I}^{\texttt{ML}\alpha}$ and $max\_state\_index = i$
14 **end**
15 **end**
16 **end**
17 **for** $s_j \in B_\alpha$ **do**
18 Consider the cube $C_I = \{s_j\}$;
19 Store CND_I as the set of conditions on key and IV bits such that $p_{s(I)} = 0$ for first l rounds (except those rounds where $p_{s(I)} = 1$).
20 **end**
21 $temp = \max\_round$; $\texttt{CUBE} = \{s_{max\_state\_index}\}$; $B_\alpha = B_\alpha \setminus \texttt{CUBE}$;
22 **while** *TRUE* **do**
23 **for** $s_j \in B_\alpha$ **do**
24 $\texttt{CUBE}_j = \texttt{CUBE} \cup \{s_j\}$;
25 If there is a round such that $\Pr(p_{s(I)} = 1) < \alpha$ for the cube $\texttt{CUBE}_j$ after imposing the required conditions from CND_I for each $C_I = \{s_i\} \subset \texttt{CUBE}_j$ then store $R_{\texttt{CUBE}_j}^{\texttt{MIZ}}$,$R_{\texttt{CUBE}_j}^{\texttt{MLZ}}$ and $R_{\texttt{CUBE}_j}^{\texttt{ML}\alpha}$;
26 **end**
27 $max_1 = \max\{R_{\texttt{CUBE}_j}^{\texttt{MIZ}}\}$; $max_2 = \max\{R_{\texttt{CUBE}_j}^{\texttt{MLZ}}\}$; $max_3 = \max\{R_{\texttt{CUBE}_j}^{\texttt{ML}\alpha}\}$;
28 $max = \max\{max_1, max_2, max_3\}$;
29 For each $s_i \in B_\alpha$ such that at least two of $R_{\texttt{CUBE}_i}^{\texttt{MIZ}}, R_{\texttt{CUBE}_i}^{\texttt{MLZ}}, R_{\texttt{CUBE}_i}^{\texttt{ML}\alpha}$ satisfies the max_1, max_2, max_3 respectively. $\texttt{CUBE} = \texttt{CUBE} \cup \{s_i\}$ and $B_\alpha = B_\alpha \setminus \{s_i\}$;
30 **if** $\max > temp$ **then**
31 $temp = \max$;
32 **end**
33 **else**
34 Break;
35 **end**
36 **end**
37 Return CUBE and $dim(\texttt{CUBE})$.

The statements from 18 to 21 in Algorithm 6 store the conditions for each variable in B_α for converting initial non-zero functions to zero function. The conditions are

stored up to first l rounds for each 1-dimensional cube $C_I \subset B_\alpha$ as CND_I. In this step, the conditions are pre-searched. The time complexity T_2 depends on the adversary's available time to evaluate the conditions at as high as possible rounds. In our case, we could go up to 69 KSA rounds for Grain-128a using the SAGE software.

Then the remaining part of the algorithm selects possible cubes by combining three techniques. In this step, the adversary improves the number of KSA rounds by increasing cube variables from the set B_α till there is no improvement of rounds. Hence the time complexity T_3 in this step, is $O(\frac{1}{pq^2} \times \sum_{d=1}^{cube\_dimension} \binom{|B_\alpha|}{d})$. Hence the total time complexity for cube searching is $T = T_1 + T_2 + T_3$.

4.3 Cube tester for Grain-128a

4.3.1 Structure observation of Grain-128a

To improve the attack, imposing conditions on some variables (state bits) may help to find a cube where the superpoly $p_{s(I)}$ satisfies a nonrandomness criterion. The structure of the cipher needs to be observed carefully to find those conditional variables. This observation can be checked algebraically and experimentally. We observe the algebraic structure of Grain-128a to find conditional bits for reducing the nonlinear degree of z_t and g. These conditions are helpful to extend the round (from 198 round to 201 round) for nonrandomness (see Subsection 4.3.3).

There are some bits in Grain-128a which are involved in the highest degree monomial of z_t as well as NFSR update function g. The degree of z_t is reduced for some rounds by fixing those bits as zero. b_{95+t} is involved in the highest degree monomial (i.e., $b_{t+12}b_{t+95}s_{t+94}$ as $x_0x_4x_8$) of z_t (i.e., 3 degree monomial) and NFSR update function g (i.e., 4 degree monomial) of Grain-128a (See Equation 2.11 and Equation 2.10). Assigning $b_{95+t} = 0$ for first 32 rounds, the degree of the output function z_t is reduced to 2 from 3. Further, at $t = 32$, $b_{95+t} = b_{128}$ is substituted by g. As the highest degree

term of g (i.e., $b_{t+88}b_{t+92}b_{t+93}b_{t+95}$) contains b_{t+95} and $b_{95+t} = 0$ for 32 rounds, the degree of g is reduced by 1 for next 32 rounds (i.e., from 32 to 63 rounds). As a result, some higher degree monomials of the function z_t vanish for next 32 rounds.

Further b_{12} is involved in the highest degree monomial of z_t. After 64 rounds, the state bit b_{12} becomes b_{76}. Now assigning $b_{76+t} = 0$ for next 19 rounds, the highest degree monomial in z_t vanishes for these rounds. As b_{95+t} is already assigned to 0 for 32 rounds and the bits $b_{76}, b_{77}, \ldots, b_{127}$ are fixed as zero, the highest degree monomial in z_t vanishes for 51 rounds. Hence, by fixing 51 key bits, the degree of z_t is reduced, or some high degree monomials vanish for the first 84 rounds.

4.3.2 Cube searching for Grain-128a

We followed the method described in Section 4.2.3 to find a cube for a higher KSA round of Grain-128a. The process of searching the cubes of Grain-128a is presented next.

Step 1 : First we checked $\Pr(p_{s(I)} = 1)$ for each 1-dimensional cube with large number of samples. We observed that there are several cases such that $0.06 < \Pr(p_{s(I)} = 1) < 0.07$. Hence, we choose the parameter $\alpha = 0.07$. Let B_α be the set of variables in the cube C_I such that $0 < \Pr(p_{s(I)} = 1) < \alpha$. In the case of Grain-128a, $B_\alpha = \{s_{42}, s_{43}, \ldots, s_{69}, s_{95}\}$. For finding the probabilities of superpolies (i.e., $\Pr(p_{s(I)} = 1)$), we fix all non-cube IV bits as zero and vary key bits for a large number of samples.

Further for each 1-dimensional cube C_I whose variable is from B_α, we find the set of conditions on key and IV bits as CND_I such that $p_{s(I)} = 0$ for first 69 rounds except those rounds where the superpoly is a constant Boolean function (i.e., zero or one function). While searching those conditions, no IV bit was pre-fixed as zero. We choose the conditions up to 69 rounds because we could compute polynomials up to 69 rounds in SAGE software. The conditions for each 1-dimensional cube from B_α is listed in Table 4.3. For $I = \{42\}$, the conditions in $CND_I = \{s_{35} = 0, s_{49} = 0, b_{46} = 0, b_{50} = 0, b_{95} = 0\}$ are required to make the superpoly $p_{s(I)} = 0$ for first 69 rounds

expect the rounds where $p_{s(I)} = 1$.

Table 4.3: Conditions for 1-dimensional cubes from B_α

Cube	Conditions	Cube	Conditions
$\{s_{42}\}$	$s_{35} = s_{49} = b_{46} = b_{50} = b_{95} = 0$	$\{s_{57}\}$	$s_{50} = s_{64} = b_{61} = b_{65} = b_{110} = 0$
$\{s_{43}\}$	$s_{36} = s_{50} = b_{47} = b_{51} = b_{96} = 0$	$\{s_{58}\}$	$s_{51} = s_{65} = b_{62} = b_{66} = b_{111} = 0$
$\{s_{44}\}$	$s_{37} = s_{51} = b_{48} = b_{52} = b_{97} = 0$	$\{s_{59}\}$	$s_{52} = s_{66} = b_{63} = b_{67} = b_{112} = 0$
$\{s_{45}\}$	$s_{38} = s_{52} = b_{49} = b_{53} = b_{98} = 0$	$\{s_{60}\}$	$s_{53} = s_{67} = s_{79} = b_{64} = b_{68} = b_{113} = 0$
$\{s_{46}\}$	$s_{39} = s_{53} = b_{50} = b_{54} = b_{99} = 0$	$\{s_{61}\}$	$s_{54} = s_{68} = s_{80} = b_{65} = b_{69} = b_{114} = 0$
$\{s_{47}\}$	$s_{40} = s_{54} = b_{51} = b_{55} = b_{100} = 0$	$\{s_{62}\}$	$s_{55} = s_{69} = s_{81} = b_{66} = b_{70} = b_{115} = 0$
$\{s_{48}\}$	$s_{41} = s_{55} = b_{52} = b_{56} = b_{101} = 0$	$\{s_{63}\}$	$s_{56} = s_{70} = s_{82} = b_{67} = b_{71} = b_{80} = b_{116} = 0$
$\{s_{49}\}$	$s_{42} = s_{56} = b_{53} = b_{57} = b_{102} = 0$	$\{s_{64}\}$	$s_{57} = s_{71} = s_{83} = b_{68} = b_{72} = b_{117} = 0$
$\{s_{50}\}$	$s_{43} = s_{57} = b_{54} = b_{58} = b_{103} = 0$	$\{s_{65}\}$	$s_{58} = s_{72} = s_{84} = b_{69} = b_{73} = b_{118} = 0$
$\{s_{51}\}$	$s_{44} = s_{58} = b_{55} = b_{59} = b_{104} = 0$	$\{s_{66}\}$	$s_{59} = s_{73} = s_{85} = b_{70} = b_{74} = b_{119} = 0$
$\{s_{52}\}$	$s_{45} = s_{59} = b_{56} = b_{60} = b_{105} = 0$	$\{s_{67}\}$	$s_{60} = s_{74} = s_{86} = b_{71} = b_{75} = b_{120} = 0$
$\{s_{53}\}$	$s_{46} = s_{60} = b_{57} = b_{61} = b_{106} = 0$	$\{s_{68}\}$	$s_{61} = s_{75} = s_{87} = b_{72} = b_{76} = b_{121} = 0$
$\{s_{54}\}$	$s_{47} = s_{61} = b_{58} = b_{62} = b_{107} = 0$	$\{s_{69}\}$	$s_{62} = s_{76} = s_{88} = b_{73} = b_{77} = b_{122} = 0$
$\{s_{55}\}$	$s_{48} = s_{62} = b_{59} = b_{63} = b_{108} = 0$	$\{s_{95}\}$	29 NFSR and 4 LFSR bits are 0
$\{s_{56}\}$	$s_{49} = s_{63} = b_{60} = b_{64} = b_{109} = 0$		1 NFSR and 1 LFSR bits are 1.

Step 2: In this step, we select a 1-dimensional cube whose variable is from B_α such that the cube satisfies the maximum last α round. Here, the cubes $\{s_{42+i}\}, 0 \leq i \leq 27$ satisfy α test at maximum round $95+i$ respectively and the cube $\{s_{95}\}$ satisfies α test at maximum round 35. Hence the superpoly corresponds to the cube $\{s_{69}\}$ satisfies the α test at maximum round 122. It is too found that the cube $\{s_{69}\}$ satisfies maximum initial zero round and maximum last zero round. The details of the cube are presented in Table 4.4.

Step-3: In this step, the dimension of cubes is extended by adding one variable in each iteration.

1. For all pair $\{s_{69}, c_I\}, c_I \in B_\alpha \setminus \{s_{69}\}$, the conditions for $\{s_{69}\}$ and c_I from Table 4.3 are imposed and the experiment for superpoly to each cube is performed

for several random key and IV pairs. Note that the experiment for the cube $\{s_{62}, s_{69}\}$ can not be performed as s_{62} is already conditioned for the variable s_{69} and vice versa. We choose the cube $\{s_{66}, s_{69}\}$ of dimension 2 which satisfies maximum last α round and maximum last zero round. The cubes satisfying the tests are listed in Table 4.5.

Table 4.4: The 1-dimensional cube(s) satisfying the conditions

Maximum last α		Maximum last zero		Maximum initial zero	
Cube	Round	Cube	Round	Cube	Round
$\{s_{69}\}$	122	$\{s_{69}\}$	115	$\{s_{69}\}$	107

Table 4.5: The 2-dimensional cube(s) satisfying the conditions

Maximum last α		Maximum last zero		Maximum initial zero	
Cube	Round	Cube	Round	Cube	Round
$\{\mathbf{s_{66}}, \mathbf{s_{69}}\}, \{s_{67}, s_{69}\}$	153	$\{\mathbf{s_{66}}, \mathbf{s_{69}}\}$	155	$\{s_{61}, s_{69}\}$	122

2. To find the 3-dimensional cubes from the cube $\{s_{66}, s_{69}\}$, we consider the triplets $\{s_{66}, s_{69}, c_I\}$ where $c_I \in B_\alpha \setminus \{s_{66}, s_{69}\}$ and set the conditions from Table 4.3 associated with these three variables. As s_{59}, s_{62} are conditioned for variables s_{66}, s_{69} respectively, c_I is taken from $B_\alpha \setminus \{s_{59}, s_{62}, s_{66}, s_{69}\}$. Two 3-dimensional cubes $\{s_{66}, s_{67}, s_{69}\}, \{s_{66}, s_{68}, s_{69}\}$ are found with satisfying all three tests and chosen for further processing. The cubes satisfying the tests are listed in Table 4.6.

Table 4.6: The 3-dimensional cube(s) satisfying the conditions

Maximum last α		Maximum last zero		Maximum initial zero	
Cube	Round	Cube	Round	Cube	Round
$\{\mathbf{s_{66}}, \mathbf{s_{67}}, \mathbf{s_{69}}\}$ $\{\mathbf{s_{66}}, \mathbf{s_{68}}, \mathbf{s_{69}}\}$	159	$\{\mathbf{s_{66}}, \mathbf{s_{67}}, \mathbf{s_{69}}\}$ $\{\mathbf{s_{66}}, \mathbf{s_{68}}, \mathbf{s_{69}}\}$	158	All Triplets	128

3. We search for cubes of dimension four from these two 3-dimensional cubes. The cubes $\{s_{66}, s_{67}, s_{69}, c_{I_1}\}$ and $\{s_{66}, s_{68}, s_{69}, c_{I_2}\}$ where $c_{I_1} \in B_\alpha \setminus \{s_{66}, s_{67}, s_{69}\}$ and $c_{I_2} \in B_\alpha \setminus \{s_{66}, s_{68}, s_{69}\}$ are considered for experiment. The conditions associated with the variables for each case are imposed. Here, $s_{59}, s_{60}, s_{61}, s_{62}$ are fixed for $s_{66}, s_{67}, s_{68}, s_{69}$ respectively. Hence the new variables c_{I_1} and c_{I_2} are taken from $B_\alpha \setminus \{s_{59}, s_{60}, s_{62}, s_{66}, s_{67}, s_{69}\}$ and $B_\alpha \setminus \{s_{59}, s_{61}, s_{62}, s_{66}, s_{68}, s_{69}\}$ respectively. We choose the 4-dimensional cube $\{s_{63}, s_{66}, s_{68}, s_{69}\}$ which satisfies the most number of tests. The cubes satisfying the tests are listed in Table 4.7.

Table 4.7: The 4-dimensional cube(s) satisfying the conditions

Maximum last α		Maximum last zero		Maximum initial zero	
Cube	Round	Cube	Round	Cube	Round
$\{s_{55}, s_{66}, s_{67}, s_{69}\}$	160	$\{s_{66}, s_{67}, s_{68}, s_{69}\}$	160	$\{s_{45}, s_{66}, s_{67}, s_{69}\}$	158
$\{s_{55}, s_{66}, s_{67}, s_{69}\}$		$\{\mathbf{s_{63}, s_{66}, s_{68}, s_{69}}\}$		$\{\mathbf{s_{63}, s_{66}, s_{68}, s_{69}}\}$	
$\{s_{56}, s_{66}, s_{67}, s_{69}\}$		$\{s_{66}, s_{67}, s_{68}, s_{69}\}$			
$\{s_{57}, s_{66}, s_{67}, s_{69}\}$		$\{s_{45}, s_{67}, s_{68}, s_{69}\}$			
$\{s_{60}, s_{66}, s_{67}, s_{69}\}$					

4. The cubes of dimension five are searched as the previous technique. All cubes $\{s_{63}, s_{66}, s_{68}, s_{69}, c_I\}$ where $c_I \in B_\alpha \setminus \{s_{56}, s_{59}, s_{61}, s_{62}, s_{63}, s_{66}, s_{68}, s_{69}\}$ are studied. We got three such cubes which satisfy all three tests. In this case, the maximum initial zero round and maximum last zero round are the same, i.e., 160 and they satisfy the maximum last α test. The results are listed in Table 4.8.

5. The same extension process of a dimension of the cube to 6 does not improve rounds. Therefore, we stop the extension process. Further, we perform some experiments on the cubes from the variables present in the selected 5-dimensional cubes to observe their biases. From the variables of two cubes $\{s_{63}, s_{64}, s_{66}, s_{68}, s_{69}\}$ and $\{s_{63}, s_{66}, s_{67}, s_{68}, s_{69}\}$, we found a new 5-dimensional cube $\{s_{61}, s_{63}, s_{64}, s_{67}, s_{69}\}$ and two 6-dimensional cubes $\{s_{63}, s_{64}, s_{66}, s_{67}, s_{68}, s_{69}\}$

Table 4.8: The 5-dimensional cube(s) satisfying the conditions

Maximum last α		Maximum last zero		Maximum initial zero	
Cube	Round	Cube	Round	Cube	Round
$\{s_{43}, s_{63}, s_{66}, s_{68}, s_{69}\}$	161	All tuples	160	$\{s_{51}, s_{66}, s_{67}, s_{68}, s_{69}\}$	160
$\{s_{44}, s_{64}, s_{66}, s_{68}, s_{69}\}$				$\{\mathbf{s_{60}}, \mathbf{s_{63}}, \mathbf{s_{66}}, \mathbf{s_{68}}, \mathbf{s_{69}}\}$	
$\{s_{46}, s_{66}, s_{67}, s_{68}, s_{69}\}$				$\{\mathbf{s_{63}}, \mathbf{s_{64}}, \mathbf{s_{66}}, \mathbf{s_{68}}, \mathbf{s_{69}}\}$	
$\{s_{47}, s_{66}, s_{67}, s_{68}, s_{69}\}$				$\{\mathbf{s_{63}}, \mathbf{s_{66}}, \mathbf{s_{67}}, \mathbf{s_{68}}, \mathbf{s_{69}}\}$	
$\{s_{57}, s_{66}, s_{67}, s_{68}, s_{69}\}$					
$\{s_{58}, s_{66}, s_{67}, s_{68}, s_{69}\}$					
$\{\mathbf{s_{60}}, \mathbf{s_{63}}, \mathbf{s_{66}}, \mathbf{s_{68}}, \mathbf{s_{69}}\}$					
$\{\mathbf{s_{63}}, \mathbf{s_{64}}, \mathbf{s_{66}}, \mathbf{s_{68}}, \mathbf{s_{69}}\}$					
$\{\mathbf{s_{63}}, \mathbf{s_{66}}, \mathbf{s_{67}}, \mathbf{s_{68}}, \mathbf{s_{69}}\}$					

and $\{s_{61}, s_{63}, s_{64}, s_{66}, s_{67}, s_{69}\}$. Note that, in the 5-dimensional cube and the second 6-dimensional cube, s_{68} is replaced by s_{61} as both variables are conditioned to each other. These cubes do not extend the rounds but show higher bias at some particular rounds (see Table 4.11).

Since the dimension of cubes are small and the bias (unbalancedness) of superpoly, i.e., $(0.5 - \alpha)$, is high, we need a small number of samples to verify the bias. Our experiments are performed in a normal PC (core-i5 processor with four cores and 8 GB RAM), and it took less than 3 hours for searching all cubes (excluding the required time for finding the conditions using SAGE).

4.3.3 Results for Grain-128a

After having the cubes (in Section 4.3.2) which are expected to give bias in higher rounds, we perform the experiments for higher rounds to find bias. The experiment is performed in the following way:

- First, the cube C_I along with its fixed IVs (and fixed keys in the weak key setup) is taken.

- Then the values of the superpoly $p_{s(I)}$ of the cube are calculated for random key-IV pairs.
- The probability $\Pr(p_{s(I)} = 1)$ is estimated. If it is different from 0.5, then the superpoly has a bias on the balancedness property.
- Finally, a sufficient number of random key-IV pairs is taken to verify the correctness of the bias with high confidence.

The complexity of the experiment depends on the number of sample key-IV pairs and the dimension of the cube $dim(C_I)$. As discussed in Section 2.3.1.3, we have taken at least $\frac{39}{pq^2}$ random key-IV pairs where $p(1-q) = \Pr(p_{s(I)} = 1)$ for each experiment. Hence, the total complexity of the experiment is $\frac{39}{pq^2} \times dim(C_I)$ to distinguish the cipher in reduced rounds from a random source. The experiments are performed in a normal PC (core-i5 processor and 8 GB RAM) running eight threads parallelly. Every thread was further running 2^6 samples parallelly. It took less than $2, 3$ and 6 hours to distinguish Grain-128a using our cubes of dimension $4, 5$ and 6 respectively.

Our experiments are performed on the chosen cubes of dimension $4, 5$ and 6. The 4-dimensional and 5-dimensional cubes $\{s_{63}, s_{66}, s_{68}, s_{69}\}$, $\{s_{60}, s_{63}, s_{66}, s_{68}, s_{69}\}$, $\{s_{63}, s_{64}, s_{66}, s_{68}, s_{69}\}$ and $\{s_{63}, s_{66}, s_{67}, s_{68}, s_{69}\}$ improve the results on the number of rounds in the single key setup as well as weak key setup. Note that in the single key setup, the cube variables are considered with the conditions on IV bits and ignoring the conditions on key bits from CND_I. In a weak key setup, the cubes, along with all conditions from CND_I are considered. We performed experiments on the 5-dimensional new cubes $\{s_{61}, s_{63}, s_{64}, s_{67}, s_{69}\}$, $\{s_{63}, s_{64}, s_{66}, s_{67}, s_{68}, s_{69}\}$ and 6-dimensional new cube $\{s_{61}, s_{63}, s_{64}, s_{66}, s_{67}, s_{69}\}$ to improve the bias.

Single key setup: In this setup, we apply the conditions on IV bits listed in Table 4.9 and ignore the conditions on key bits. Since the conditions are applied on some IV bits only, all key bits and remaining IV bits (except the cube variables) are taken randomly. We found a distinguisher for the cubes which can distinguish Grain-

Table 4.9: Conditions for different cube

Cube	Conditional IV Bits	Conditional key Bits
$\{s_{63}, s_{66}, s_{68}, s_{69}\}$	$s_{56}, s_{59}, s_{61}, s_{62},$ $s_{70}, s_{73}, s_{75}, s_{76},$ $s_{82}, s_{85}, s_{87}, s_{88}$	$b_{67}, b_{70}, b_{71}, b_{72}, b_{73},$ $b_{74}, b_{76}, b_{77}, b_{80},$ $b_{116}, b_{119}, b_{121}, b_{122}$
$\{s_{60}, s_{63}, s_{66}, s_{68}, s_{69}\}$	$s_{53}, s_{56}, s_{59}, s_{61}, s_{62},$ $s_{67}, s_{70}, s_{73}, s_{75}, s_{76},$ $s_{79}, s_{82}, s_{85}, s_{87}, s_{88}$	$b_{64}, b_{67}, b_{68}, b_{70}, b_{71}, b_{72},$ $b_{73}, b_{74}, b_{76}, b_{77}, b_{80},$ $b_{113}, b_{116}, b_{119}, b_{121}, b_{122}$
$\{s_{63}, s_{64}, s_{66}, s_{68}, s_{69}\}$	$s_{56}, s_{57}, s_{59}, s_{61}, s_{62},$ $s_{70}, s_{71}, s_{73}, s_{75}, s_{76},$ $s_{82}, s_{83}, s_{85}, s_{87}, s_{88}$	$b_{67}, b_{68}, b_{70}, b_{71}, b_{72},$ $b_{73}, b_{74}, b_{76}, b_{77}, b_{80},$ $b_{116}, b_{117}, b_{119}, b_{121}, b_{122}$
$\{s_{63}, s_{66}, s_{67}, s_{68}, s_{69}\}$	$s_{56}, s_{59}, s_{60}, s_{61}, s_{62},$ $s_{70}, s_{73}, s_{74}, s_{75}, s_{76},$ $s_{82}, s_{85}, s_{86}, s_{87}, s_{88}$	$b_{67}, b_{70}, b_{71}, b_{72}, b_{73},$ $b_{74}, b_{75}, b_{76}, b_{77}, b_{80},$ $b_{116}, b_{119}, b_{120}, b_{121}, b_{122}$
$\{s_{61}, s_{63}, s_{64}, s_{66}, s_{67}, s_{69}\}$	$s_{54}, s_{56}, s_{57}, s_{59}, s_{60}, s_{62},$ $s_{68}, s_{70}, s_{71}, s_{73}, s_{74}, s_{76},$ $s_{80}, s_{82}, s_{83}, s_{85}, s_{86}, s_{88}$	$b_{65}, b_{67}, b_{68}, b_{69}, b_{70}, b_{71},$ $b_{72}, b_{73}, b_{75}, b_{76}, b_{77}, b_{80},$ $b_{114}, b_{116}, b_{117}, b_{119}, b_{120}, b_{122}$

128a from a random source at 186, 187 and 191 rounds. We have taken 2^{29} samples (i.e., key-IV pairs which are not fixed and cube variables) for each experiment which provides at least 99% confidence level for the experiment (see Section 2.3.1.3). The result is tabulated in Table 4.10. From a practical attack perspective, we too have experimented using our best cube $\{s_{60}, s_{63}, s_{66}, s_{68}, s_{69}\}$ on a random key with 2^{29} $\overline{IV}$ samples. We performed this experiment for 200 random keys and found bias for 39 keys at 191 KSA round of Grain-128a. This result indicates that there are a significant number of keys which show the bias at 191 KSA round of Grain-128a.

Weak key setup: In this setup, the conditions on both key and IV bits as listed in Table 4.9 are imposed. In this weak key setup, we observed the nonrandomness in Grain-128a for higher rounds. 2^{29} random samples of key-IV pairs (which are not fixed and cube variables) are used for the experiment. We have two different scenarios for imposing conditions on key bits.

Table 4.10: Result of cube testers on Grain-128a in the single key setup

Cube	Round	$\Pr(p_{s(I)} = 1)$	No. of fixed IVs	Time complexity
$\{s_{63}, s_{66}, s_{68}, s_{69}\}$	186	0.4990	12	$2^{28.22}$
$\{s_{63}, s_{66}, s_{67}, s_{68}, s_{69}\}$	186	0.4995	15	$2^{31.22}$
$\{s_{63}, s_{64}, s_{66}, s_{68}, s_{69}\}$	186	0.4993	15	$2^{30.24}$
$\{s_{60}, s_{63}, s_{66}, s_{68}, s_{69}\}$	191	0.4998	15	$2^{33.86}$
$\{s_{61}, s_{63}, s_{64}, s_{66}, s_{67}, s_{69}\}$	186	0.498	18	$2^{28.22}$

Scenario I: Here, we fix the key and IV bits to make a non-zero superpoly as zero superpoly except the constant superpolies for the first 69 rounds. The conditions are listed in Table 4.9. From the experiment, we found the cubes which are presented in Table 4.11.

Table 4.11: Result of cube testers on Grain-128a in the weak key setup (Scenario I)

Cube	Round	$\Pr(p_{s(I)} = 1)$	No. of fixed IV bits	No. of fixed key bits	Time complexity
$\{s_{61}, s_{63}, s_{64}, s_{67}, s_{69}\}$	192	0.487	15	15	$2^{21.82}$
	194	0.498	15	15	$2^{27.22}$
$\{s_{63}, s_{64}, s_{66}, s_{68}, s_{69}\}$	196	0.4998	15	15	$2^{33.86}$
$\{s_{63}, s_{66}, s_{67}, s_{68}, s_{69}\}$	197	0.4998	15	15	$2^{33.86}$
$\{s_{60}, s_{63}, s_{66}, s_{68}, s_{69}\}$	198	0.4991	15	16	$2^{29.52}$
$\{s_{61}, s_{63}, s_{64}, s_{66}, s_{67}, s_{69}\}$	186	0.355	18	19	$2^{15.86}$
	189	0.455	18	19	$2^{19.23}$
	192	0.491	18	19	$2^{23.88}$
	193	0.495	18	19	$2^{25.57}$
	194	0.498	18	19	$2^{28.22}$
	196	0.4998	18	19	$2^{34.86}$

Scenario II: To have a bias at a higher round, more conditions on key bits are imposed by observing the structure of Grain-128a as presented in Section 4.3.1. From the structure observation, the additional conditional key bits for the cube $\{s_{60}, s_{63}, s_{66}, s_{68}, s_{69}\}$ are $b_{64}, b_{67}, b_{70}, b_{71}, b_{72}, b_{73}, b_{74}, b_{76}, b_{77}, b_{78}, b_{79}, b_{80}, b_{81}, b_{82}, b_{83}, b_{84}, b_{85}, b_{86}, b_{87}, b_{91}, b_{94}, b_{95}, b_{102}, b_{104}, b_{105}, b_{108}, b_{110}, b_{112}, b_{113}, b_{114}, b_{116}, b_{118}, b_{119}, b_{121}, b_{122}, b_{125}$.

We achieved a bias at 201 round from the experiment that is listed in Table 4.12. In a practical attack perspective, we have experimented on a random key with 2^{29} $\overline{IV}$ samples using our best cube $\{s_{60}, s_{63}, s_{66}, s_{68}, s_{69}\}$. We performed this experiment for 200 random keys and found bias for 195 keys at 201 KSA round of Grain-128a. This result hints that there may exist a bias in Grain-128a of 191 KSA round for a large percentage of keys.

Table 4.12: Result of cube testers on Grain-128a in the weak key setup (Scenario II)

Cube	Round	$\Pr(p_{s(I)} = 1)$	No. of fixed IV bits	No. of fixed key bits	Time complexity
$\{s_{60}, s_{63}, s_{66}, s_{68}, s_{69}\}$	201	0.4998	15	36	$2^{33.86}$

4.4 Cube tester for Grain-128

4.4.1 Study for Grain-128

The design of Grain-128 [30] is similar to Grain-128a. We have implemented our technique on this cipher. From the experiment on Grain-128, we set α as 0.13. Then the 1-dimensional cube variables are stored in B_α along with the conditions in CND_I as described in Algorithm 6. The 1-dimensional cube $\{s_{62}\}$ is selected as it goes for the maximum last α round. As per the process of expanding the cube, we gradually get 2-dimensional, 3-dimensional and 4-dimensional cubes $\{s_{60}, s_{62}\}$, $\{s_{60}, s_{62}, s_{65}\}$

and $\{s_{60}, s_{62}, s_{65}, s_{66}\}$ respectively, from the 1-dimensional cube. Then we get two 5-dimensional cubes $\{s_{60}, s_{62}, s_{63}, s_{65}, s_{66}\}$ and $\{s_{60}, s_{62}, s_{64}, s_{65}, s_{66}\}$. Further, the same 6-dimensional cube $\{s_{60}, s_{62}, s_{63}, s_{64}, s_{65}, s_{66}\}$ is obtained by extending both the 5-dimensional cubes. Then by extending the 6-dimensional cube, a 7-dimensional cube $\{s_{34}, s_{60}, s_{62}, s_{63}, s_{64}, s_{65}, s_{66}\}$ is obtained. The selection of cube with satisfying rounds are presented in Table 4.13.

Table 4.13: Cube selection for Grain-128

Cube	Round		
	Maximum last α	Maximum last zero	Maximum initial zero
$\{s_{60}, s_{62}\}$	152	149	103
$\{s_{60}, s_{62}, s_{65}\}$	182	161	152
$\{s_{60}, s_{62}, s_{65}, s_{66}\}$	180	177	157
$\{s_{60}, s_{62}, s_{64}, s_{65}, s_{66}\}$	188	184	168
$\{s_{60}, s_{62}, s_{63}, s_{64}, s_{65}, s_{66}\}$	199	184	184
$\{s_{34}, s_{60}, s_{62}, s_{63}, s_{64}, s_{65}, s_{66}\}$	201	187	197

For the 7-dimensional cube, 20 IV bits and 19 key bits are fixed. Then performing the experiment for 2^{29} samples (which is a sufficient number of samples) provides a significant bias in 207 and 235 KSA rounds of Grain-128 in the single key and the weak key setup, respectively. The details of the attack are presented in Table 4.14. From a practical attack perspective, we have experimented using our best cube $\{s_{34}, s_{60}, s_{62}, s_{63}, s_{64}, s_{65}, s_{66}\}$ on a random key with 2^{29} $\overline{IV}$ samples. We performed this experiment for 200 random keys in both setups. We found bias for 4 keys at 207 KSA round of Grain-128 in single key setup and for 197 keys at 235 KSA round of Grain-128 in weak key setup. This result hints that there may exist a bias in Grain-128 of 207 KSA round for a significant number of keys in both setups.

Table 4.14: Attacks on Grain-128

Attack Type	Attack Name	Cube dimension	Round	$\Pr(p_{s(I)} = 1)$	Time complexity
Single key	Cube tester	6	205	0.498	$2^{28.22}$
	Cube tester	7	207	0.4998	$2^{35.86}$
Weak key	Cube tester	6	229	0.4998	$2^{34.22}$
	Cube tester	7	235	0.4997	$2^{34.69}$

4.5 Comparison

In this section, the comparisons of our results with the previous attacks on Grain-128a are presented.

Single key setup: In this setup, we got distinguishers for 186 and 191 rounds using 4-dimensional and 5-dimensional cubes respectively (see Table 4.10). The comparisons of our attack with other existing attacks on Grain-128a in the single key setup are presented below.

- Lehmann et al. [58] used 33-dimensional cubes to get a distinguisher at 177 KSA round.

- The papers [50] (conditional differential attack), [60, 61] (chosen IV statistical attacks) have presented attacks on Grain-128a of KSA rounds $169, 169, 171$ using $1, 22, 25$-dimensional difference vector or cube respectively.

- Recently a fast correlation attack [62] is proposed to cryptanalyse Grain-128a of the full KSA round (i.e., 256). The time complexity of this attack is $2^{115.4}$, which is yet to be practical. The time complexity of our distinguishing attack (at reduced KSA round 191) is $2^{33.86}$, which is practical. Therefore, our attack is on the reduced KSA round (which is highest so far) with practical time complexity,

whereas the attack in [62] is on the full KSA round with non-practical time complexity.

Weak key setup: In this setup, we could find a bias at 201 KSA round using a 5-dimensional cube. The comparisons of our attack with other existing attacks on Grain-128a in the weak key setup are presented below.

- Lehmann et al. [58] used 6-dimensional cubes to get a distinguisher at 189 KSA round.
- Ma et al. [50] proposed a conditional differential attack on Grain-128a of KSA round 195 using a 1-dimensional difference vector.
- Karlsson et al. [59] used a maximum degree monomial signature technique to propose a nonrandomness detector on Grain-128a of 203 KSA round. This attack uses a 38-dimensional cube on both key and IV variables. Since the nonrandomness detector involves key variables in the cube, it is a weaker attack than the distinguishing attack in the weak key setup.

The numeric comparisons in the single key and the weak key setups are presented in Table 4.1.

Similarly, in the **single key setup**, we could find a bias for Grain-128 of 205 and 207 KSA rounds by using the cubes of dimension 6 and 7 respectively. In the earlier attacks, we have the following.

- The cube testers presented in [58, 63, 64] can distinguish Grain-128 of KSA rounds $207, 236, 237$ and 256 using cubes of dimension $12, 33, 40$ and 50 respectively.
- A dynamic cube attack [65] leads to attack Grain-128 of 207 and 250 KSA rounds using 19-dimensional and 37-dimensional cubes respectively.
- In a conditional differential attack [66], Grain-128 is being attacked for 215 KSA round using 13 difference vector bits.

Although we could attack till 207 KSA round of Grain-128 using our technique, the cube's dimension is 7, which is very small compared to the earlier results.

In the **weak key setup**, we could distinguish Grain-128 of 235 KSA round by using 7-dimensional cube. The nonrandomness detector presented in [59] can find nonrandomness in Grain-128 of full KSA round by using a 25-dimensional cube of key and IV bits. However, the nonrandomness detector attack is weaker as its cube involves key bits. The detailed comparisons for both setups are presented in Table 4.2.

4.6 Conclusion

In this chapter, we used the maximum last α technique combining with maximum last zero and maximum initial zero techniques to find a good cube for a cube tester on the Grain-like stream ciphers. Using the technique, we presented some distinguishers on Grain-128a at higher KSA rounds for both single and weak key setups. We get a distinguisher for Grain-128a of 191 KSA round using a 6-dimensional cube in a single key setup. In weak key setup, we get distinguishers for Grain-128a of $189, 198$ and 201 KSA rounds by using 6-dimensional, 5-dimensional, and another 5- dimensional cubes, respectively. The dimension of the cubes is lesser than the previous results. As Grain-128AEAD inherits KSA (without the authentication initialization) and encryption modules of Grain-128a, our attack is too applicable to Grain-128AEAD with no authentication initialization. We also studied our technique over the cipher Grain-128 to check its validity, and it returns good results.

Chapter 5

Time-Memory-Data Trade-off (TMDTO) Attack by using state bit recovery attack

5.1 Motivation

The state bit recovery of a cipher plays a crucial role in cryptanalysis. From the recovered state, the following keystream bits can be generated without knowing the key of the cipher. Moreover, since some ciphers' state update function is invertible, the initial state of such ciphers can be recovered from any of its later internal states. In this case, the state recovery leads to a key recovery attack. Recent ciphers are designed in such a way that it is challenging to recover a large number of its internal state bits without fixing a large number of state bits using a known-plaintext attack. The challenge is to recover many state bits of a cipher from some known keystream bits by fixing fewer state bits. Some papers are available to recover the state bits of a particular cipher by carefully observing its internal structure [20, 67–70]. The proposal of an algorithm to recover state bits of a class of ciphers by fixing an optimal

number of state bits is an essential problem in the study of cryptanalysis.

If a state of n bits of a cipher can be recovered by guessing $n-l$ bits with the knowledge of some initial keystream bits, the sampling resistance of the cipher is defined to be $R = 2^{-l}$. For example, the sampling resistance of Grain-v1 is at most 2^{-18} [71]. The sampling resistance can be used for TMDTO attack as a BSW-sampling technique which is used to cryptanalyse many stream ciphers [20, 67–74]. Having some pattern of data, i.e., keystream, some state bits of the stream cipher can be recovered. Further, imposing conditions on some state bits increases the number of recovering state bits. Therefore, having these conditions on some state bits, the BSW-sampling attack can further be improved. This attack is known as conditional BSW-sampling TMDTO attack or simply conditional TMDTO attack. Several papers have used this technique to analyze stream ciphers. In the first paragraph, we discussed that some state bits are recovered from some known keystream bits and fixing some state bits in a state bit recovery attack. Therefore, the state bit recovery attack promotes to implementation of conditional TMDTO attack. As the number of fixing bits increases, the online time complexity (T) and data complexity (D) (see Equation 5.9) are increased, the number of fixing bits needs to be decreased.

5.1.1 Our Contribution

Conditional BSW-sampling is being used to analyze FSR-based stream ciphers like Grain-v1 [4] and Lizard [13]. For this attack, it is possible to recover some state bits by fixing some other state bits from known specific keystream bits. A higher number of recovering bits and a lower number of fixing bits can improve the attack complexities. There is no algorithm available to find the state bits which need to be recovered and fixed. In this chapter, for the first time, we present a deterministic algorithm for this purpose. The algorithm recovers the same number of state bits of any FSR-based cipher with possibly an optimal number of fixing bits from some known consecutive keystream bits. We implement the algorithm on two famous FSR-

based ciphers, Lizard and Grain-128a (without authentication mode). Also, we have used the classical method to recover the state bits of Grain-v1. We get the following results.

- Grain-v1: 33 state bits are recovered from the same number of keystream bits by fixing 45 state bits.

- Lizard: $10, 11, \ldots, 24$ state bits are recovered from the same number of keystream bits by fixing $10, 12, 14, 16, 18, 20, 22, 24, 38, 40, 42, 44, 46, 48, 50$ state bits respectively.

- Grain-128a: $35, 48$ state bits are recovered from $35, 48$ keystream bits by fixing $34, 54$ state bits respectively. This is the first result on this cipher in this direction.

These recovering and fixing state bits from specific keystream bits are used to implement a conditional BSW-sampling TMDTO attack. We have followed the strategy implemented by Jiao et al. [68], and Mihalijevic et al. [69] for Grain-v1. Then we presented the TMDTO curve using the number of fixing and recovering bits in Theorem 8. Looking into the practicality, we analyse our results on three different conditions on TMDTO parameters T, M, D as follows.

1. $D < T = M$, which is considered by looking into the practicality of the availability of data D.

2. $T = 2^{-f-2r}D^2, M = D$, which satisfies the lower bound for T in terms of D, i.e., to have the least time complexity in terms of the data complexity.

3. $D = T = M$, minimizes $\max(D, T, M)$. Maitra et al. [70] first used this criterion on Lizard with an additional condition, i.e., $T' = D'^2$. Here D' and T' are time and data in reduced space.

The previous result by Maitra et al. [70] achieved the complexity $T = M = D = 2^{54}$. We improved this result concerning all parameters, which are presented in Table 5.16. However, the best results chosen from our cases are as follows.

1. $T = M = 2^{54}, D = 2^{48}$ where the data is reduced by 64 times than Maitra et al. [70].

2. $T = 2^{52}, M = D = 2^{53}$ or, $T = M = 2^{53}, D = 2^{52}$ where the result on minimization of $\max\{T, M, D\}$ is improved.

3. $T = 2^{50}, M = D = 2^{54}$, where the time complexity is reduced by 16 times than Maitra et al. [70].

4. $T = 2^{42}, M = D = 2^{60}$ which reduces time complexity by 2^{18} times than the overall complexity claimed by Hamann et al. [13].

Using the criteria, we can recover Lizard's whole state bits with the best complexities until now. Our TMDTO attack on Grain-v1 and Grain-128a provides better complexities than previously known results.

5.1.2 Organisation

The remaining part of this chapter is divided into four sections. Section 5.2 is divided into two subsections where Subsection 5.2.1 discusses the analysis of h function of Grain-v1, and the guess and determine strategy is discussed in Subsection 5.2.2. Section 5.3 presents the state recovery algorithm in two different subsections. Subsection 5.3.1 presents the basic idea of the recovery technique of any FSR-based stream cipher, and Subsection 5.3.2 presents the algorithm to recover state bits. The algorithm is implemented on stream ciphers Lizard and Grain-128a along with results in Subsection 5.3.3.1 and 5.3.3.2, respectively. Section 5.4 discusses TMDTO attack using state recovery and is divided by five subsections. The conditional TMDTO attack

is discussed in Subsection 5.4.1. Our Contribution regarding TMDTO curve is explained in Subsection 5.4.2. The conditional TMDTO attacks on Lizard, Grain-128a and Grain-v1 are presented in Subsection 5.4.3, Subsection 5.4.4 and Subection 5.4.5, respectively. Finally, in Section 5.5, we conclude our work.

5.2 Bit recovery of Grain-v1 by observing algebraic structure

In this section, we are discussing the recovery of the internal state of Grain-v1 at a particular clock. A study on the subfunctions (i.e., fixing some variables of the function) of the nonlinear function h is presented in the following subsection. The study is useful for the state recovery of Grain-v1.

5.2.1 Analysis of the Non-linear Filter Function of Grain-v1

The nonlinear function h (Equation 2.4) is a 3 degree polynomial on 5 variables. The algebraic normal form (ANF) of h contains only 8 nonlinear terms. The sparseness of nonlinear terms in the ANF helps to find the affine or constant subfunctions by fixing a few variables. It is observed that all 3-degree monomials and one 2-degree monomial contain the variable bit s_{t+46}. Therefore, fixing $s_{t+46} = 0$, h can be made a quadratic function with 2 nonlinear terms. In addition to this fixing, if we fix $s_{t+64} = 0$ or, 1, we will have a linear function independent of the variable s_{t+3} or, b_{t+63} respectively. Moreover, exploiting the normality order of h (i.e., 2), we can have a constant function by fixing 3 variables. We listed the observations on the ANF of h as follows.

Observation 1. *The observations on the ANF of h are:*

1. $h(s_{t+3}, s_{t+25}, 0, s_{t+64}, b_{t+63}) = s_{t+25} + b_{t+63} + s_{t+3}s_{t+64} + s_{t+64}b_{t+63}$;

   1.1. $h(s_{t+3}, s_{t+25}, 0, 1, b_{t+63}) = s_{t+3} + s_{t+25}$;

1.1.1. $h(s_{t+3}, 1, 0, 1, b_{t+63}) = 1 + s_{t+3}$;

1.2. $h(s_{t+3}, s_{t+25}, 0, 0, b_{t+63}) = s_{t+25} + b_{t+63}$;

2. $h(1, 0, s_{t+46}, 1, b_{t+63}) = 1$;

3. $h(s_{t+3}, 0, s_{t+46}, 0, 0) = 0$;

4. $h(s_{t+3}, 0, 1, s_{t+64}, b_{t+63}) = s_{t+64} + b_{t+63} + s_{t+3}b_{t+63}$.

4.1. $h(1, 0, 1, s_{t+64}, b_{t+63}) = s_{t+64}$.

By fixing some state bits, we use these observations to extract relations (mostly linear) among the state bits. These relations help to recover some state bits by guessing the rest of the state bits. We use the relations in observation in Item 1 for $17 - 20$ rounds, in Item 1.1. for $3, 4, 11$ rounds, in Item 1.1.1. for $5 - 10$ rounds, in Item 1.2. for $0 - 2, 16$ rounds, in Item 2 for $12 - 14$ rounds, in Item 3 for 15 round, in Item 4 for $21 - 26$ rounds and in Item 4.1. for $27 - 32$ rounds of Grain-v1. Table 5.1 lists the observations of the relations of state bits in terms of the subfunctions of h in the order of round. The state bits in brackets are previously fixed with the same value, and the bits in bold letters are having a position greater than 79, which can be expressed in terms of recurrence as defined in Equation 2.2 and Equation 2.3.

5.2.2 Guess and Determine Strategy

Exploiting the relations among the state bits presented in Table 5.1, the guess and determine strategy is used to recover 33 state bits from the first 33 keystream bits of Grain-v1. Having 33 known keystream bits ($z_t, 0 \leq t \leq 32$), appropriately replacing the h function in Equation 2.5 by the equation presented in the Table 5.1, we will have a system of 33 equations. For this process, 45 state bits (presented in the 3rd column in Table 5.1) need to be fixed. If the system of equations is linearly independent, it is possible to recover 33 state bits by guessing the rest (i.e., $160 - (45 + 33) = 82$) of the state bits.

Table 5.1: Relations of state bits in Grain-v1 as the subfunctions of h

Round(t)	Observation	Fixing Bit	h function
0 – 2	1.2.	$s_{t+46} = 0, s_{t+64} = 0$	$s_{t+25} + b_{t+63}$
3 – 4	1.1.	$s_{t+46} = 0, s_{t+64} = 1$	$s_{t+3} + s_{t+25}$
5 – 10	1.1.1.	$s_{t+25} = 1, s_{t+46} = 0, s_{t+64} = 1$	$1 + s_{t+3}$
11	1.1.	$s_{t+46} = 0, s_{t+64} = 1$	$s_{t+3} + s_{t+25}$
12 – 14	2	$s_{t+3} = 1, s_{t+25} = 0, s_{t+64} = 1$	1
15	3	$s_{t+25} = 0, s_{t+64} = 0, b_{t+63} = 0$	0
16	1.2.	$s_{t+46} = 0, \mathbf{s_{t+64}} = 0$	$s_{t+25} + b_{t+63}$
17	1	$s_{t+46} = 0$	$s_{t+25} + \mathbf{b_{t+63}} + s_{t+3}\mathbf{s_{t+64}}$ $+\mathbf{s_{t+64}b_{t+63}}$
18 – 20	1	$(s_{t+46} = 0)$	$s_{t+25} + \mathbf{b_{t+63}} + s_{t+3}\mathbf{s_{t+64}}$ $+\mathbf{s_{t+64}b_{t+63}}$
21 – 26	4	$(s_{t+25} = 0, s_{t+46} = 1)$	$\mathbf{s_{t+64}} + \mathbf{b_{t+63}} + s_{t+3}\mathbf{b_{t+63}}$
27 – 32	4.1.	$(s_{t+3} = 1, s_{t+25} = 0, s_{t+46} = 1)$	$\mathbf{s_{t+64}}$

Since some equations are nonlinear and nonlinear recurrence relations represent some state bits, choosing appropriate recovery bits and guessing bits is not obvious. As the gap between the terms b_{t+10} and b_{t+31} in Equation 2.5 is maximum and the terms are involved linearly, we consider those bits as recovery bits. The recovery process of the state bits is presented below. The detailed order of evaluation and evaluation process is presented in Table 5.4 and Table 5.5.

R1. For $0 \le t \le 2$, (i.e., for first 3 rounds), we use the observation(Item 1.2.) for h to get a linear equation on state bits for z_t by fixing two state bits as mentioned in Table 5.1. Here, each linear equation contains 9 terms as $b_{t+1} + b_{t+2} + b_{t+4} + b_{t+10} + b_{t+31} + b_{t+43} + b_{t+56} + s_{t+25} + b_{t+63} = z_t$ for $0 \le t \le 2$. Now, we can recover three state bits $b_{t+10}, 0 \le t \le 2$ by guessing remaining state bits in the equations.

R2. For $t = 3, 4, 11$, the observation(Item 1.1.) for h is used to get a linear equation on state bits for z_t by fixing two state bits as mentioned in Table 5.1. Here, each linear equation contains 9 terms as $b_{t+1} + b_{t+2} + b_{t+4} + b_{t+10} + b_{t+31} + b_{t+43} +$

$b_{t+56} + s_{t+3} + s_{t+25} = z_t$ for $t = 3, 4, 11$. Similarly, three state bits $b_{t+10}, t = 3, 4, 11$ are recovered by guessing remaining state bits in the equations.

R3. For $5 \leq t \leq 10$, the observation(Item 1.1.1.) for h is used to get a linear equation on state bits for z_t by fixing three state bits as mentioned in Table 5.1. Here, each linear equation contains 8 terms as $b_{t+1} + b_{t+2} + b_{t+4} + b_{t+10} + b_{t+31} + b_{t+43} + b_{t+56} + s_{t+3} + 1 = z_t$ for $5 \leq t \leq 10$. Here, Six state bits $b_{t+10}, 5 \leq t \leq 10$ are recovered by guessing remaining state bits in the equations.

R4. For $12 \leq t \leq 14$, the observation(Item 2) for h is used to get a linear equation on state bits for z_t by fixing three state bits as mentioned in Table 5.1. Here, each linear equation contains 7 terms as $b_{t+1} + b_{t+2} + b_{t+4} + b_{t+10} + b_{t+31} + b_{t+43} + b_{t+56} + 1 = z_t$ for $12 \leq t \leq 14$. Here, three state bits $b_{t+10}, 12 \leq t \leq 14$ are recovered by guessing remaining state bits in the equations.

R5. For $t = 15$, the observation(Item 3) for h is used to get a linear equation on state bits for z_t by fixing three state bits as mentioned in Table 5.1. The linear equation contains 7 terms as $b_{t+1} + b_{t+2} + b_{t+4} + b_{t+10} + b_{t+31} + b_{t+43} + b_{t+56} = z_t$ for $t = 15$. Here, the state bit $b_{t+10}, t = 15$ is recovered by guessing remaining state bits in the equation.

It can be observed that for $t \geq 16$, at least one term in fixing bits or in h function (written in the bold letter in Table 5.1) which is expressed as a linear or nonlinear combination of other state bits. For example, at $t = 16$, we need to fix $s_{t+16} = s_{80} = s_0 + s_{13} + s_{23} + s_{38} + s_{51} + s_{62} = 0$(see Equation 2.2). Therefore, the involved bits need to be considered for fixing or guessing bits.

Table 5.2 and Table 5.3 contain the involved bits in the linear update state relations and the nonlinear update state relations respectively.

R6. For $t = 16$, the observation(Item 1.2.) for h is used to get a linear equation on state bits for z_t by fixing two state bits $s_{62} = 0$ and $s_0 = s_{13} + s_{23} + s_{38} + s_{51} + s_{62}$

Table 5.2: The state bits involved to calculate the linear feedback bits

Feedback Bits	State bits used	Feedback Bits	State bits used
s_{80}	$s_0, s_{13}, s_{23}, s_{38}, s_{51}, s_{62}$	s_{81}	$s_1, s_{14}, s_{24}, s_{39}, s_{52}, s_{63}$
s_{82}	$s_2, s_{15}, s_{25}, s_{40}, s_{53}, s_{64}$	s_{83}	$s_3, s_{16}, s_{26}, s_{41}, s_{54}, s_{65}$
s_{84}	$s_4, s_{17}, s_{27}, s_{42}, s_{55}, s_{66}$	s_{85}	$s_5, s_{18}, s_{28}, s_{43}, s_{56}, s_{67}$
s_{86}	$s_6, s_{19}, s_{29}, s_{44}, s_{57}, s_{68}$	s_{87}	$s_7, s_{20}, s_{30}, s_{45}, s_{58}, s_{69}$
s_{88}	$s_8, s_{21}, s_{31}, s_{46}, s_{59}, s_{70}$	s_{89}	$s_9, s_{22}, s_{32}, s_{47}, s_{60}, s_{71}$
s_{90}	$s_{10}, s_{23}, s_{33}, s_{48}, s_{61}, s_{72}$	s_{91}	$s_{11}, s_{24}, s_{34}, s_{49}, s_{62}, s_{73}$
s_{92}	$s_{12}, s_{25}, s_{35}, s_{50}, s_{63}, s_{74}$	s_{93}	$s_{13}, s_{26}, s_{36}, s_{51}, s_{64}, s_{75}$
s_{94}	$s_{14}, s_{27}, s_{37}, s_{52}, s_{65}, s_{76}$	s_{95}	$s_{15}, s_{28}, s_{38}, s_{53}, s_{66}, s_{77}$
s_{96}	$s_{16}, s_{29}, s_{39}, s_{54}, s_{67}, s_{78}$	s_{97}	$s_{17}, s_{30}, s_{40}, s_{55}, s_{68}, s_{79}$

as mentioned in Table 5.1. The linear equation contains 9 terms as $b_{t+1} + b_{t+2} + b_{t+4} + b_{t+10} + b_{t+31} + b_{t+43} + b_{t+56} + s_{t+25} + b_{t+63} = z_t$ for $t = 16$. Here, the state bit $b_{t+63}, t = 16$ is recovered by guessing remaining state bits in the equation.

The non-linear state update relations are involved for some terms available in the h function from this step onward.

Table 5.3: The state bits involved to calculate the non-linear feedback bits

Feedback Bits	State bits used	Feedback Bits	State bits used
b_{80}	$b_0, b_9, b_{14}, b_{15}, b_{21}, b_{28}, b_{33}$ $b_{37}, b_{45}, b_{52}, b_{60}, b_{62}, b_{63}, s_0$	b_{81}	$b_1, b_{10}, b_{15}, b_{16}, b_{22}, b_{29}, b_{34}$ $b_{38}, b_{46}, b_{53}, b_{61}, b_{63}, b_{64}, s_1$
b_{82}	$b_2, b_{11}, b_{16}, b_{17}, b_{23}, b_{30}, b_{35}$ $b_{39}, b_{47}, b_{54}, b_{62}, b_{64}, b_{65}, s_2$	b_{83}	$b_3, b_{12}, b_{17}, b_{18}, b_{24}, b_{31}, b_{36}$ $b_{40}, b_{48}, b_{55}, b_{63}, b_{65}, b_{66}, s_3$
b_{84}	$b_4, b_{13}, b_{18}, b_{19}, b_{25}, b_{32}, b_{38}$ $b_{41}, b_{49}, b_{56}, b_{64}, b_{66}, b_{67}, s_4$	b_{85}	$b_5, b_{14}, b_{19}, b_{20}, b_{26}, b_{33}, b_{39}$ $b_{42}, b_{50}, b_{57}, b_{65}, b_{67}, b_{68}, s_5$
b_{86}	$b_6, b_{15}, b_{20}, b_{21}, b_{27}, b_{34}, b_{40}$ $b_{43}, b_{51}, b_{58}, b_{66}, b_{68}, b_{69}, s_6$	b_{87}	$b_7, b_{16}, b_{21}, b_{22}, b_{28}, b_{35}, b_{41}$ $b_{44}, b_{52}, b_{59}, b_{67}, b_{69}, b_{70}, s_7$
b_{88}	$b_8, b_{17}, b_{22}, b_{23}, b_{29}, b_{36}, b_{42}$ $b_{45}, b_{53}, b_{60}, b_{68}, b_{70}, b_{71}, s_8$	b_{89}	$b_9, b_{18}, b_{23}, b_{24}, b_{30}, b_{37}, b_{43}$ $b_{46}, b_{54}, b_{61}, b_{69}, b_{71}, b_{72}, s_9$

R7. For $27 \leq t \leq 32$, the observation(Item 4.1.) for h is used to get equations on state bits for z_t by fixing three state bits as mentioned in Table 5.1 which are already fixed in previous steps. Most of the state bits involved in the equations are already guessed or fixed in earlier steps. In this step, six state bits $b_{28}, b_{29}, b_{30}, b_{73}, b_{74}, b_{75}$ can be recovered by guessing two bits s_{24} for $t = 27$ and b_{27} for $t = 30$. We brought these rounds before some previous rounds (for $17 \leq t \leq 26$), because the recovering bits b_{73}, b_{74}, b_{75} are used for the equations in the rounds $t = 17, 18, 19$ respectively.

Further, in this step, we recover bits from the equations of the rounds in the order of $t = 29, 28, 27, 30, 31, 32$ (see Table 5.4 and Table 5.5). b_{30}, b_{29}, b_{28} are recovered in this order, because b_{30} and b_{29} are required for the recovery of b_{29} and b_{28} respectively and b_{28} is required for the recovery of b_{74}.

R8. For $17 \leq t \leq 20$, the observation(Item 1) for h is used to get non-linear equations on state bits for z_t by fixing a state bit $s_{t+46} = 0$ as mentioned in Table 5.1. However, $s_{t+46} = 0$ for $18 \leq t \leq 20$ is already fixed in step R1. From the equation and update relations, the state bit $s_{t+25}, 17 \leq t \leq 20$ can be recovered by guessing the remaining state bits in the equation.

R9. For $21 \leq t \leq 26$, from the non-linear equation of h observations(Item 4), six state bits $s_{18} - s_{19}$ and $s_{58} - s_{61}$ are recovered as the linear bits in the respective equations. In this step the b_{77}, b_{78} bits are guessed for $t = 21, 22$ respectively.

From the above process, we recover 33 state bits from known 33 consecutive keystream bits. To recover the whole internal state, we need to fix 45 state bits (out of which 44 bits as a constant value and one bit as a linear equation of state bits) and guessing the rest 82 bits. For this purpose, we exploited 33 equations. The detailed recovery process of bits with the fixing, recovering and guessing bits are presented in Table 5.4 and Table 5.5. The rows of the Table are shown in order of recovery. The terms in brackets are previously assigned (i.e., fixing or, guessing or, recovering).

Table 5.4: Recovery of state bits

Round (t)	Constrains	Key Bits(z_t)	Recovery equation	Guessing bits	Recovering bit
0	$s_{46} = 0, s_{64} = 0$	z_0	$b_{10} = z_0 + b_1 + b_2 + b_4$ $b_{31} + b_{43} + b_{56} + s_{25} + b_{63}$	$b_1, b_2, b_4, b_{31}, b_{43}$, b_{56}, s_{25}, b_{63}	b_{10}
1	$s_{47} = 0, s_{65} = 0$	z_1	$b_{11} = z_1 + b_2 + b_3 + b_5$ $+b_{32} + b_{44} + b_{57} + s_{26} + b_{64}$	$(b_2), b_3, b_5, b_{32}$, $b_{44}, b_{57}, s_{26}, b_{64}$	b_{11}
2	$s_{48} = 0, s_{66} = 0$	z_2	$b_{12} = z_2 + b_3 + b_4 + b_6$ $+b_{33} + b_{45} + b_{58} + s_{27} + b_{65}$	$(b_3, b_4), b_6, b_{33}$, $b_{45}, b_{58}, s_{27}, b_{65}$	b_{12}
3	$s_{49} = 0, s_{67} = 1$	z_3	$b_{13} = z_3 + b_4 + b_5 + b_7$ $+b_{34} + b_{46} + b_{59} + s_6 + s_{28}$	$(b_4, b_5), b_7, b_{34}$, $b_{46}, b_{59}, s_6, s_{28}$	b_{13}
4	$s_{50} = 0, s_{68} = 1$	z_4	$b_{14} = z_4 + b_5 + b_6 + b_8$ $+b_{35} + b_{47} + b_{60} + s_7 + s_{29}$	$(b_5, b_6), b_8, b_{35}$, $b_{47}, b_{60}, s_7, s_{29}$	b_{14}
5	$s_{51} = 0, s_{69} = 1$ $s_{30} = 1$	z_5	$b_{15} = z_5 + b_6 + b_7 + b_9$ $+b_{36} + b_{48} + b_{61} + s_8 + 1$	$(b_6, b_7), b_9, b_{36}$, b_{48}, b_{61}, s_8	b_{15}
6	$s_{52} = 0, s_{70} = 1$ $s_{31} = 1$	z_6	$b_{16} = z_6 + b_7 + b_8 + b_{10}$ $+b_{37} + b_{49} + b_{62} + s_9 + 1$	$(b_7, b_8, b_{10}), b_{37}$, b_{49}, b_{62}, s_9	b_{16}
7	$s_{53} = 0, s_{71} = 1$ $s_{32} = 1$	z_7	$b_{17} = z_7 + b_8 + b_9 + b_{11} +$ $b_{38} + b_{50} + b_{63} + s_{10} + 1$	$(b_8, b_9, b_{11}), b_{38}$, $b_{50}, (b_{63}), s_{10}$	b_{17}
8	$s_{54} = 0, s_{72} = 1$ $s_{33} = 1$	z_8	$b_{18} = z_8 + b_9 + b_{10} + b_{12} +$ $b_{39} + b_{51} + b_{64} + s_{11} + 1$	$(b_9, b_{10}, b_{12}), b_{39}$, $b_{51}, (b_{64}), s_{11}$	b_{18}
9	$s_{55} = 0, s_{73} = 1$ $s_{34} = 1$	z_9	$b_{19} = z_9 + b_{10} + b_{11} + b_{13} +$ $b_{40} + b_{52} + b_{65} + s_{12} + 1$	$(b_{10}, b_{11}, b_{13}), b_{40}$, $b_{52}, (b_{65}), s_{12}$	b_{19}
10	$s_{56} = 0, s_{74} = 1$ $s_{35} = 1$	z_{10}	$b_{20} = z_{10} + b_{11} + b_{12} + b_{14} +$ $b_{41} + b_{53} + b_{66} + s_{13} + 1$	$(b_{11}, b_{12}, b_{14}), b_{41}$, b_{53}, b_{66}, s_{13}	b_{20}
11	$s_{57} = 0, s_{75} = 1$	z_{11}	$b_{21} = z_{11} + b_{12} + b_{13} + b_{15} +$ $b_{42} + b_{54} + b_{67} + s_{14} + s_{36}$	$(b_{12}, b_{13}, b_{15}), b_{42}$, $b_{54}, b_{67}, s_{14}, s_{36}$	b_{21}
12	$s_{15} = 1, s_{76} = 1$ $s_{37} = 0$	z_{12}	$b_{22} = z_{12} + b_{13} + b_{14} +$ $b_{16} + b_{43} + b_{55} + b_{68} + 1$	$(b_{13}, b_{14}, b_{16}, b_{43})$, b_{55}, b_{68}	b_{22}
13	$s_{16} = 1, s_{77} = 1$ $s_{38} = 0$	z_{13}	$b_{23} = z_{13} + b_{14} + b_{15} +$ $b_{17} + b_{44} + b_{56} + b_{69} + 1$	$(b_{14}, b_{15}, b_{17}, b_{44}$, $b_{56}), b_{69}$	b_{23}
14	$s_{17} = 1, s_{78} = 1$ $s_{39} = 0$	z_{14}	$b_{24} = z_{14} + b_{15} + b_{16} +$ $b_{18} + b_{45} + b_{57} + b_{70} + 1$	$(b_{15}, b_{16}, b_{18}, b_{45}$, $b_{57}), b_{70}$	b_{24}
15	$b_{78} = 0, s_{79} = 0$ $s_{40} = 0$	z_{15}	$b_{25} = z_{15} + b_{16} + b_{17} +$ $b_{19} + b_{46} + b_{58} + b_{71}$	$(b_{16}, b_{17}, b_{19}, b_{46}$, $b_{58}), b_{71}$	b_{25}
16	$s_{62} = 0$ $s_0 = s_{13} + s_{38}$ $+s_{23} + s_{51}$	z_{16}	$b_{79} = z_{16} + b_{17} + b_{18} +$ $b_{20} + b_{26} + b_{47} + b_{59}$ $+b_{72} + s_{41}$	$(b_{17}, b_{18}, b_{20}, b_{47}$, $b_{59}, s_{13}, s_{38}.s_{51})$, $b_{26}, b_{72}, s_{23}, s_{41}$	b_{79}
29	$(s_{75} = 1, s_{54} = 0$ $s_{32} = 1)$	z_{29}	$b_{30} = z_{29} + b_{31} + b_{33} +$ $b_{39} + b_{60} + b_{72} + b_{85} + s_{93}$	$(b_{31}, b_{33}, b_{60}, b_{72}$ $b_{39})$	b_{30}

Compared with the previous works, we can recover 33 state bits by fixing 45 state bits and guessing the rest 82 bits. In earlier results, it is possible to recover $18, 28, 31, 32$ bits fixing $54, 51, 32, 0$ state bits and guessing the rest of state bits, respectively. Hence, our result improves the conditional sampling resistance to 2^{-33}. The comparison is presented in Table 5.6.

Table 5.5: Recovery of state bits continued

Round (t)	Constrains	Key Bits(z_t)	Recovery equation	Guessing bits	Recovering bit
28	$(s_{74}=1, s_{53}=0$ $s_{31}=1), s_{63}=0$	z_{28}	$b_{29}=z_{28}+b_{30}+b_{32}+b_{38}+$ $b_{59}+b_{71}+b_{84}+s_{92}$	$(b_{29}, b_{30}, b_{32}, b_{59},$ $b_{71}, b_{38})$	b_{29}
27	$(s_{73}=1$ $s_{52}=0, s_{30}=1)$	z_{27}	$b_{28}=z_{27}+b_{29}+b_{31}+b_{37}+$ $b_{58}+b_{70}+b_{83}+s_{91}$	$(b_{28}, b_{29}, b_{31}, b_{58},$ $b_{70}, b_{37}), s_{24}$	b_{28}
30	$(s_{76}=1$ $s_{55}=0, s_{33}=1)$	z_{30}	$b_{73}=z_{30}+b_{31}+b_{32}+b_{34}+$ $b_{40}+b_{61}+b_{86}+s_{94}$	$(b_{31}, b_{32}, b_{34}, b_{61},$ $b_{40}), b_{27}$	b_{73}
17	$(s_{63}=0)$	z_{17}	$s_{42}=z_{17}+b_{18}+b_{19}+$ $b_{21}+b_{48}+b_{60}+b_{27}+$ $+b_{73}+h(s_{20}, s_{81}, b_{80})$	$(b_{18}, b_{19}, b_{21}, b_{48},$ $b_{60}, b_{73})$ $s_1, s_{20}, (b_{28})$	s_{42}
31	$(s_{77}=1$ $s_{56}=0, s_{34}=1)$	z_{31}	$b_{74}=z_{31}+b_{32}+b_{33}+$ $b_{35}+b_{41}+b_{62}+b_{87}+s_{95}$	$(b_{32}, b_{33}, b_{35}, b_{62},$ $b_{41}, b_{28})$	b_{74}
18	$(s_{64}=0)$	z_{18}	$s_{43}=z_{18}+b_{19}+b_{20}+$ $b_{22}+b_{28}+b_{49}+b_{61}+$ $b_{74}+h(s_{21}, s_{82}, b_{81})$	$(b_{19}, b_{20}, b_{22}, b_{49},$ $b_{61}, b_{74})$ $s_2, s_{21}, (b_{29})$	s_{43}
32	$(s_{78}=1$ $s_{57}=0, s_{35}=1)$	z_{32}	$b_{75}=z_{32}+b_{33}+b_{34}+$ $b_{36}+b_{42}+b_{63}+b_{88}+s_{96}$	$(b_{33}, b_{34}, b_{36}, b_{63},$ $b_{42}, b_{29})$	b_{75}
19	$(s_{65}=0)$	z_{19}	$s_{44}=z_{19}+b_{20}+b_{21}+$ $b_{23}+b_{29}+b_{50}+b_{62}+$ $b_{75}+h(s_{22}, s_{83}, b_{82})$	$(b_{20}, b_{21}, b_{23}, b_{50},$ $b_{62}, b_{75})$ $s_3, s_{22}, (b_{30})$	s_{44}
20	$(s_{66}=0)$	z_{20}	$s_{45}=z_{20}+b_{21}+b_{22}+$ $b_{24}+b_{30}+b_{51}+b_{63}+$ $b_{76}+h(s_{23}, s_{84}, b_{83})$	$(b_{21}, b_{22}, b_{24}, b_{51},$ $b_{63}), b_{76}, s_4, (s_{42})$	s_{45}
21	$(s_{67}=1, s_{46}=0)$	z_{21}	$s_{56}=z_{21}+b_{22}+b_{23}+b_{25}+b_{31}$ $+b_{52}+b_{64}+b_{77}+h(s_{85}, b_{84})$	$(b_{22}, b_{23}, b_{25}, b_{52}, b_{64}$ $, b_{31}), b_{77}, s_5, (s_{43})$	s_{18}
22	$(s_{68}=1, s_{47}=0)$	z_{22}	$s_{57}=z_{22}+b_{23}+b_{24}+b_{26}+b_{32}$ $+b_{53}+b_{65}+b_{78}+h(s_{86}, b_{85})$	$(b_{23}, b_{24}, b_{26}, b_{53},$ $b_{65}, b_{32}, s_{44}), b_{78}$	s_{19}
23	$(s_{69}=1, s_{48}=0)$	z_{23}	$s_{58}=z_{23}+b_{24}+b_{25}+b_{27}+b_{33}$ $+b_{54}+b_{66}+b_{79}+h(s_{87}, b_{86})$	$(b_{24}, b_{25}, b_{27}, b_{54},$ $b_{66}, b_{33}, b_{79}, s_{45})$	s_{58}
24	$(s_{70}=1, s_{49}=0)$	z_{24}	$s_{59}=z_{24}+b_{25}+b_{26}+b_{28}+b_{34}$ $+b_{55}+b_{67}+b_{80}+h(s_{88}, b_{87})$	$(b_{25}, b_{26}, b_{28}, b_{55},$ $b_{67}, b_{34}, b_{28})$	s_{59}
25	$(s_{71}=1, s_{50}=0)$	z_{25}	$s_{60}=z_{25}+b_{26}+b_{27}+b_{29}+b_{35}$ $+b_{56}+b_{68}+b_{81}+h(s_{89}, b_{88})$	$(b_{26}, b_{27}, b_{29}, b_{56},$ $b_{68}, b_{35}, b_{29})$	s_{60}
26	$(s_{72}=1, s_{51}=0)$	z_{26}	$s_{61}=z_{26}+b_{27}+b_{28}+b_{30}+b_{36}$ $+b_{57}+b_{69}+b_{82}+h(s_{90}, b_{89})$	$(b_{27}, b_{28}, b_{30}, b_{57},$ $b_{69}, b_{36}, b_{30})$	s_{61}

Table 5.6: Comparison of our result with previous results

References	Fixing bits	Recovering bits	Required Keystream Bits	Guessing bits
Bjørstad [71]	0	21	21	139
Mihaljević et al. [75]	54	18	18	88
Jiao et al. [68]	51	28	28	81
Mihaljević et al. [69]	32	31	31	97
Siddhanti et al. [76]	0	32	36	96
Our work	45	33	33	82

5.3 State bit recovery of FSR based stream ciphers

We present a method to recover some state bits of an FSR-based stream cipher by fixing and guessing the remaining state bits. The technique works for the stream cipher Lizard and Grain-128a.

5.3.1 General idea of state bit recovery

This subsection presents a general idea of state bit recovery of an NFSR based stream cipher. In this process, we recover some consecutive state bits of one FSR (NFSR or LFSR). Then we possibly increase the number of recovering state bits by recovering another set of consecutive state bits. We consider the following notations on the primitives of an FSR based stream cipher.

- l : the length of the state of the FSR;
- $A_0, A_1, \ldots, A_{l-1}$: the state bits at 0-th clock;
- g : the feedback function of the FSR;
- z : the output/filter function in the stream cipher;
- z_t : the output bit at the t-th clock.

For the explanation purpose, we use the NFSR2 of Lizard cipher in this subsection. The state update formula of the NFSR and the output function of Lizard are presented in Equation 2.13 and Equations 2.14 respectively. In this case, $l = 90$.

The output z_t is dependent on a subset of state bits. Some of these state bits are linear, i.e., they are not involved in any nonlinear terms. In the example, B_{5+t}, B_{7+t}, $B_{11+t}, B_{30+t}, B_{40+t}, B_{45+t}, B_{54+t}, B_{71+t}$ are linear state bits of NFSR2 in the output function z_t(see Equations 2.15-2.18). Therefore, knowing z_t and guessing some state bits involved in nonlinear terms, a linear equation on these linear bits can be obtained. As

a result, knowing k bits $z_0, z_1, \ldots, z_{k-1}$, one can have a system of equations over the state bits where a subset of state bits (say, m many) are linear and the rest $l-m$ state bits are not linear. Therefore, by guessing the nonlinear state bits and state bits from other FSRs in the system of equations, one can have a system of k linear equations on m variables. Then it is possible to recover "the rank of the system" many state bits. Hence, the aim is to

- generate a system of linear equations having rank as high as possible by guessing and fixing as few as possible state bits;

If we continue increasing k, the number of linear state bits (m) gets decreased after a stage. This happens because some nonlinear terms can arise involving some of the linear state bits occurring for a lower value of k. In the example, B_{45} is a linear term for $k = 0$. Then at $k = 1$, z_1 involves B_{45} in the nonlinear term $B_{45}B_{77}$(see Equation 2.16). As a result, B_{45} is no more a linear term in the system for $k > 0$. However, the linearity of B_{45} can be restored by fixing B_{77} as 0 or 1. Hence, if there is a nonlinear term containing the state variable, we fix one of the other state bit(s) as 0 to restore the linearity of the state variable in the system. Further, if there are some variables (x) involved in a quadratic term (like xy), then one can impose a condition $y = 1$ to get a new linear state bit x.

Therefore, our main purpose is to find out a set of fixing state bits such that we can have more linear state bits. This can be achieved by the above discussed method. As a result, three sets of state bits are generated from the set of equations $z_t, 0 \leq t < k$. That is,

- a set of fixing state bits C,

- a set of guessing state bits G and

- a set of recovering state bits R from the set of linear bits.

As a summary, we need to partition the state bits into C, G and R such that after fixing the state bits from C, we would have a set of equations of the form

$$MR^T + F(G,R) = Z^T \tag{5.1}$$

where M is a $k \times k$ nonsingular lower triangular matrix (i.e., no zero entry in diagonal);
$R^T = (x_{i_0}, x_{i_1}, \ldots, x_{i_{k-1}})^T$ is the transpose of the ordered state variables in R in a vector form;
$F(G,R) = (f_0, f_1, \ldots, f_{k-1})^T$ where f_j is a Boolean function on the state variables $G \cup \{x_{i_0}, x_{i_1}, \ldots, x_{i_{j-1}}\}$ for $0 \le j < k$; and
$Z^T = (z_0, z_1, \ldots, z_{k-1})^T$.
The Equation 5.1 can further be written as

$$R^T = NR^T + F(G,R) + Z^T \tag{5.2}$$

where N is a $k \times k$ lower triangular matrix with diagonal entries 0, i.e., $M = N + I$. In the case of Lizard, the Equation 5.1 is as follows:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} B_{40} \\ B_{41} \\ B_{42} \\ B_{43} \\ B_{44} \end{pmatrix} + \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix} = \begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \\ z_4 \end{pmatrix}$$

where $f_t = \mathcal{L}_t + \mathcal{Q}_t + \mathcal{T}_t + \widehat{\mathcal{T}}_t + B_{40+t} + B_{2+t}B_{28+t}B_{41+t}B_{65+t}$
$+ B_{1+t}B_{19+t}B_{27+t}B_{43+t}B_{57+t}B_{66+t}B_{78+t}$, $0 \le t \le 4$ (see Equation 2.14).

5.3.2 An algorithm for state bit recovery

In this subsection, we present a general algorithm to recover a number of state bits of an FSR-based stream cipher. Let denote the ordered (ascending on the index of

state bits) set of linear bits in the output function of the FSR-based stream cipher by $\mathcal{L}$. In the case of NFSR2 of Lizard, $\mathcal{L} = \{B_5, B_7, B_{11}, B_{30}, B_{40}, B_{45}, B_{54}, B_{71}\}$ (see Equations 2.15-2.18). Knowing z_t, one of these linear bits can be expressed as the other state bits present in the equation. In case of NFSR2 of Lizard, we can write

$$\begin{aligned} B_{11+t} &= z_t + B_{5+t} + B_{7+t} + B_{30+t} + B_{40+t} + B_{45+t} + B_{54+t} + B_{71+t} \\ &+ B_{4+t}B_{21+t} + B_{9+t}B_{52+t} + B_{18+t}B_{37+t} + B_{44+t}B_{76+t} + \mathcal{T}_t + \widehat{\mathcal{T}}_t \end{aligned} \tag{5.3}$$

For every $t \geq 0$, the state bit B_{11+t} in the left-hand side (LHS) of the equation is expressed in terms of some state bits which are present in the right-hand side (RHS) of the equation. The LHS of the equation can be any other state bit in $\mathcal{L}$, but we choose the one which is farthest from the next linear bit. The state bit B_{11+t} can be recovered by fixing some state bits and guessing the remaining state bits present in the RHS of the equation.

The number of recovering state bits can be increased by considering a system of more equations. A system of k equations on the state bits can be generated by varying t from 0 to $k-1$ for a $k \geq 1$. This system of equations can be solved easily if for every $t, 1 \leq t < k$ the state bit in LHS of the equation (e.g., B_{11+t} in our example) is not involved in the RHS of any earlier equations, i.e., generated from z_j for $0 \leq j < t$. Such system of equations is of the form Equation 5.2. Such a system of equations can be formed by

1. taking state bits in the LHS of equations between two consecutive linear state bits in the list $\mathcal{L}$ (to avoid the presence of any LHS state bit as a linear bit in the RHS of an equation) and

2. fixing a state bit (other than recovering bits) involved in nonlinear terms (to avoid the presence of any LHS state bit in a nonlinear term in the RHS of an equation).

In our example, B_{11} and B_{30} are two consecutive state bits in the ordered set $\mathcal{L}$.

We can express the bits B_{11+t} for $0 \leq t \leq 18$ to get a system of 19 equations. In this case, no LHS state bits, i.e., B_{11+t} for $0 \leq t \leq 18$ is present as a linear bit in RHS of the system. If we go for $t = 19$, the bit $B_{11+t} = B_{30}$ is present in the equation generated for $t = 0$. Hence, we can not go for $t > 18$ for B_{11+t}.

Using our technique, we can recover state bits between two linear state bits present in the output function. Hence the interval between two linear state bits plays a crucial role in our state recovery method. The linear intervals are made from the indices of two consecutive linear bits in the output function of a cipher. In our example, it may possible to recover the state bits $B_{11}, B_{12}, \ldots, B_{29}$ as B_{11} and B_{30} are two consecutive linear bits. In this case, the interval of indices of state bits between two consecutive linear state bits B_{11} and B_{30}, i.e., $(11, 29)$, is a linear interval. If linear bits are not available in the output function of a cipher, then one tries to make some nonlinear monomials as linear to follow our recovery process. We define now sets of linear intervals of a stream cipher with the output function z and m FSRs of length $l_1, l_2, \ldots, l_m$. Consider the state bits of a cipher with m FSRs are A_i^j, $0 \leq i < l_j$, $1 \leq j \leq m$. Also, $A_i^j, i \geq l_j$ is considered as state update bit. Let $i_1^j, i_2^j, \ldots, i_{p_j}^j$ (in ascending order) be indices of the linear state bits of j^{th} FSR in z, where $1 \leq j \leq m$. The set of linear intervals of j^{th} FSR of the stream cipher is defined by

$$\mathcal{I}_j \quad = \quad \{(i_1^j, i_2^j - 1), (i_2^j, i_3^j - 1), \ldots, (i_{p_j-1}^j, i_{p_j}^j - 1), (i_{p_j}^j, l_j + i_1^j - 1)\}; \tag{5.4}$$

The last interval in $\mathcal{I}_j$ is basically $(i_{p_j}^j, l_j - 1) \cup (l_j, l_j + i_1^j - 1)$. We try to recover the state bits with indices from each interval. We expect to recover a larger number of state bits from the long intervals. The length of an interval (a, b) is defined by $\text{len}(a, b) = b - a + 1$. If we have a system of equations of the form Equation 5.2 as we discussed earlier, then the size of triangular matrix $k \leq \max_{1 \leq j \leq m} \max_{(a,b) \in \mathcal{I}_j} \text{len}(a, b)$. If we have a system having more than $\max_{1 \leq j \leq m} \max_{(a,b) \in \mathcal{I}_j} \text{len}(a, b)$ equations, then

as per our method the matrix M can not be a lower triangular matrix.

Lemma 2. *The maximum number of state bits that can be recovered in an FSR based stream cipher with m FSRs using our technique is* $\max_{1\leq j\leq m}\max_{(a,b)\in\mathcal{I}_j} len(a,b)$.

For the example purpose, let rename the state bits of Lizard as $A_i^1 = S_i, 0 \leq i < 31$ for NFSR1 and $A_i^2 = B_i, 0 \leq i < 90$ for NFSR2.
Therefore, $\mathcal{L} = \{A_{23}^1, A_5^2, A_7^2, A_{11}^2, A_{30}^2, A_{40}^2, A_{45}^2, A_{54}^2, A_{71}^2\}$. The sets of linear intervals in Lizard are $\mathcal{I}_1$ and $\mathcal{I}_2$, where

- $\mathcal{I}_1 = \{(23, 31 + 22)\}$.
- $\mathcal{I}_2 = \{(5,6), (7,10), (11,29), (30,39), (40,44), (45,53), (54,70), (71,90 + 4)\}$

The length of above intervals are calculated as: $\text{len}(23, 31 + 22) = 31$,$\text{len}(5,6) = 2, \text{len}(7,10) = 4, \text{len}(11,29) = 19, \text{len}(30,39) = 10, \text{len}(40,44) = 5, \text{len}(45,53) = 9,$ $\text{len}(54,70) = 17, \text{len}(71,90 + 4) = 24$. Using our technique, it is possible to recover at most 31 state bits in Lizard.

Some state bits in between two linear bits in an interval may be involved in some nonlinear terms of the output function z and the state update function g. As a result, some LHS bits of the system are present in the nonlinear terms of RHS of equations. In our example, B_{21+t} is present in the output function z_t as a nonlinear term $B_{4+t}B_{21+t}$. The LHS bit at $t = 10$, i.e., B_{21}, is present in a nonlinear term B_4B_{21} in the equation at $t = 0$. To eliminate B_{21} from the nonlinear term, the state bit B_4 (which is not an LHS state bit) can be fixed as 0. This way, the state bit B_{21} is restored as a linear bit in the system. Hence, B_{21+t} for $0 \leq t \leq 8$ are present in the nonlinear terms in the system of equation. To remove these terms from the system of equations, the state bits $B_{4+t}, 0 \leq t \leq 8$ need to be fixed as 0.

It is needed to store all these state bits which are to be fixed (e.g., $B_{4+t}, 0 \leq t \leq 8$). In our example, instead of collecting all 9 bits, it is enough to store the initial state bit, i.e., B_4 and the fixing length, i.e., 9. Hence, for every such state in the LHS of the system which is involved in a nonlinear term, we need to collect

1. the initial state bit of all such possible state bits to be fixed and
2. the fixing length of the initial state bit (which is denoted by $fl_z^i(a,b)$ later).

In this way, it is possible to get a system of equations of the form Equation 5.2. We aim to form such a system of equations of k as high as possible with a smaller value of $|C|$. Now we define a few notations.

- R be the set of all LHS state bits of the system which will be recovered. Note that we already defined R as the set of all recovering bits. In our example, $R = \{B_{11+t} | 0 \leq t \leq 18\}$.
- a, b : a and b are the indices of the first and last bit of the continuous state bits to be recovered. In our recovery technique, (a, b) should be inside of a linear interval. We can choose $a = 11, b = 25$ which is the part of the linear interval $(11, 29)$.

The state bits in R, which are involved in a nonlinear term of the system of equations, can be recognized from the ANF of the output function z and update function g. Such state bits are recognized from two situations.

- **Situation I**: Some state bits in R are involved in nonlinear terms of the output function z. As discussed in the example, B_{21} is such a state bit, and B_4 is the corresponding state bit to be fixed as 0.
- **Situation II**: Some state bits in R appear by the update function g when a state bit A_p is updated during $0 \leq t \leq b - a$ clocks. In this case, $p \geq l - (b - a)$ and A_p needs to be present in the ANF of z. In our example of Lizard, B_{78} is present in the nonlinear term $B_1B_{19}B_{27}B_{43}B_{57}B_{66}B_{78}$ of z. The state bit B_{78+t} is replaced by the g when $t = 12$. As a result, some more nonlinear terms containing state bits from R appear. In this situation removing all the nonlinear terms containing state bits from R may require to fix lots of state bits. Therefore, we remove the term containing $B_{78+t}, t \geq 12$ in z by fixing a state bit in the nonlinear term,

i.e., a state bit from $\{B_{1+t}, B_{19+t}, B_{27+t}, B_{43+t}, B_{57+t}, B_{66+t}\}$. Therefore, we need to collect B_{78} and the possible state bits $\{B_1, B_{19}, B_{27}, B_{43}, B_{57}, B_{66}\}$ to be fixed.

We collect the state bits whose involvement in nonlinear terms of z creates a problem. In the case of situation I, it is those $A_d \in R$ which are part of a nonlinear term in z. In situation II, it is those $A_p, p \geq l-(b-a)$ which are part of a nonlinear term in z. Further, let $b \geq l-(b-a)$ and there is a state bit A_q in a nonlinear term in z such that $l-(b-a) \leq q \leq b$. Such A_q is part of both Situation I and Situation II. To avoid repetition, such state bits are considered in Situation I and not in Situation II. The state bits $A_p, \max\{l-(b-a), b+1\} \leq p < l$ which are present in a nonlinear term of z are considered in Situation II. Now we have the following definition to collect such state bits A_d and A_p.

Definition 33. *Given a function z on the state bits $A_i, 0 \leq i < l$ and two integers a, b $(0 \leq a \leq b < l)$.*

- **Nonlinear index list** *in between the indices a, b for z is defined by* $NL_z(a,b) = \{d : a \leq d \leq b$ *and A_d is present in a nonlinear term of z}.

- **Updating nonlinear index list** *of two state bits with indices a, b for z is defined by* $UL_z(a,b) =$ $\{p : \max\{l-(b-a), b+1\} \leq p < l$, *$A_p$ is present in a nonlinear term of z}.

Now, we present a toy example of a NFSR based cipher in Example 5 to explain our technique.

Example 5. *Consider a stream cipher having one NFSR of length $l = 20$ with states $A_0, A_1, \ldots, A_{19}$ and state update function g follows the recursion $A_{20} = A_0 + A_3A_{13} + A_5A_{16}$. The output function of the cipher is $z = A_3 + A_7 + A_1A_5 + A_4A_{15} + A_4A_9A_{11} + A_{13}A_{18}$. Here the set of linear bits in z is $\mathcal{L} = \{A_3, A_7\}$ and set of intervals $\mathcal{I} = \{(3,6), (7, 20+2)\}$. Consider $a = 3$ and $b = 6$. Here, the state bits in $R = \{A_{a+t}, 0 \leq t \leq$*

$b-a=3\}$ *are intended to recover. Therefore, the equations are derived as*

$$A_{3+t} = z_t + A_{7+t} + A_{1+t}A_{5+t} + A_{4+t}A_{15+t} + A_{4+t}A_{9+t}A_{11+t} + A_{13+t}A_{18+t}, \quad \text{for } 0 \le t \le b-a. \tag{5.5}$$

1. *Here,* $NL_z(a,b) = \{4,5\}$ *as* A_4, A_5 *are the only state bits whose indices are in between* $a = 3$ *and* $b = 6$ *and present in some nonlinear monomials of* z. $NL_z(a,b)$ *informs that the state bits* A_{4+t} *and* $A_{5+\bar{t}}$ *for some* $t, \bar{t} \ge 0$, *are involved in respective monomials in the system of equations. The respective monomials need to be eliminated.*

2. *Further,* $UL_z(a,b) = \{18\}$ *as* A_{18} *is the only state bit with index in between* $\max\{l-(b-a), b+1\} = \max\{20-(6-3), 6+1\} = 17$ *and* $l-1 = 19$ *and present in some nonlinear monomials of* z. *The state update function* g *appears in the system of equations (Equation 5.5) for* $t \ge l-18 = 2$. *As a result, some of the state bits from* R *may appear in the system. Instead of eliminating monomials from* g, *we aim to eliminate the term containing* $A_{18+t}, t \ge 2$. *For that reason, we need to fix* $A_{13+t}, t \ge 2$ *as* 0.

Now we will deal with the nonlinear monomials containing a state bit having index from $NL_z(a,b) \cup UL_z(a,b)$. We need to remove the monomials by fixing a state bit from these monomials. Fixing one state bit as 0 removes the monomial. Hence, for each index in $NL_z(a,b) \cup UL_z(a,b)$, we collect the starting state bits of all possible state bits to be fixed in a set and its fixing length.

Notation 1. *Let* $M = A_{i_1}A_{i_2}\cdots A_{i_p}$ *be a monomial on some state bits from* $A_j, 0 \le j < l$. *Then*

- $S(M) = \{A_{i_1}, A_{i_2}, \ldots, A_{i_p}\}$ *denotes the set of state bits involved in the monomial* M.

- *For a positive integer r, $(M >> r) = A_{i_1+r}A_{i_2+r}\cdots A_{i_p+r}$ denotes the monomial by right shifting the indices of state bits in M by r places.*

Definition 34. *Let z be the output function on the state bits $A_j, 0 \le j < l$ and a, b be two integers such that $0 \le a \le b < l$.*

1. *If $p \in NL_z(a,b)$, then the set of fixing state bits for the state bit A_p is defined as $FS_z^p = \{S(M) \setminus R : M$ is a nonlinear monomial containing A_p in $z\}$.*
2. *If $p \in UL_z(a,b)$, then the set of fixing state bits for the state bit A_p is defined as $FS_z^p = \{(S(M >> (l-p)) \setminus (R \cup \{A_l\}) : M$ is a nonlinear monomial containing A_p in $z\}$.*

From the definition, we have that FS_z^p stores the possible starting state bits which are to be fixed as 0. In the case of Situation I, FS_z^p is quite clear. In the case of Situation II, we need to remove the nonlinear terms after $l-p$ clocks. Hence, FS_z^p needs to store the state bits of the monomials after $(l-p)$ right shifts. Since the state bits in R can not be fixed, the state bits in R are removed from the set FS_z^p in both the situations.

For example, if $FS_z^p = \{\{A_i\}, \{A_{j_1}, A_{j_2}\}\}$ then the state bits A_{i+t} and A_{j_1+t} or A_{j_2+t} need to be fixed as 0 for $0 \le t < k$ where k is fixing length for p. We define the fixing length of given $p \in NL_z(a,b) \cup UL_z(a,b)$ by the maximum length we need to fix the variables in FS_z^p as 0. For given output function z and two indices of state bits a, b, the fixing length for every $p \in NL_z(a,b) \cup UL_z(a,b)$ is denoted as $fl_z^p(a,b)$. Now we have the following lemma on fixing length.

Lemma 3. *Let z be the output function on the state bits $A_j, 0 \le j < l$ and a, b be two integers such that $0 \le a \le b$.*

1. *For $p \in NL_z(a,b)$, the fixing length $fl_z^p(a,b) = b - p + 1$.*
2. *For $p \in UL_z(a,b)$, the fixing length*

$$fl_z^p(a,b) = \begin{cases} 0 & \text{if } NL_g(a+l-p,b) = \varnothing; \\ b - \min NL_g(a+l-p,b) + 1 & \text{otherwise.} \end{cases}$$

Proof. If $p \in NL_z(a,b)$ then there is a nonlinear term in z which contains A_p where $a \le p \le b$. As a result A_{p+t} is present in the equation for every $0 \le t \le b-a$. The state bits A_{p+t} for $0 \le t \le b-p$ belongs to $R = \{A_j : a \le j \le b\}$. For $t > b-p$, the state bits A_{p+t} does not belong to R and such state bits A_{p+t} are not problematic. Hence, the fixing length in this case is $fl_z^p(a,b) = b-p+1$.

If $p \in UL_z(a,b)$ then there is a nonlinear term in z which contains A_p where $\max\{l-b-a, b+1\} \le p < l$. Then the state bit A_p gets updated by the function g at $(l-p)$-th clock. If g does not contain any state bit from $\{a+l-p, a+l-p+1, \ldots, b\}$ (i.e., $NL_g(a+l-p,b) = \varnothing$), then we need not bother anything, i.e., the fixing length $fl_z^p(a,b) = 0$. Otherwise the term containing A_{p+t} needs to be made 0 for $l-p \le t \le (l-p)+(b-\min NL_g(a+l-p,b))$. Hence, the fixing length $fl_z^p(a,b) = b-\min NL_g(a+l-p,b)+1$. □

Note 2. *Since $p \in UL_z(a,b)$ with $fl_z^p(a,b) = 0$ does not force to fix any state bits, the set $UL_z(a,b)$ can be updated by removing those p from itself. That is, $UL_z(a,b) = UL_z(a,b) \setminus \{p \in UL_z(a,b) | fl_z^p(a,b) = 0\}$.*

Example 5. ***(continued)*** *We continue the example of toy stream cipher presented in Example 5. Here $R = \{A_3, A_4, A_5, A_6\}$ for $a = 3$ and $b = 6$.*

- *Here $NL_z(a,b) = \{4,5\}$. Since A_4 is involved with the monomials A_4A_{15} and $A_4A_9A_{11}$, $FS_z^4 = \{\{A_{15}\}, \{A_9, A_{11}\}\}$. Similarly, $FS_z^5 = \{\{A_1\}\}$ as A_5 is in the monomial A_1A_5. Further, $fl_z^4(a,b) = b-4+1 = 3$ and $fl_z^5(a,b) = b-5+1 = 2$. Here, $FS_z^4 = \{\{A_{15}\}, \{A_9, A_{11}\}\}$ indicates that the state bit A_{15} and one of the state bits from A_{9+t} and A_{11+t} need to be fixed as 0 for $0 \le t < fl_z^4(a,b) = 3$. $FS_z^5(a,b) = \{\{A_1\}\}$ implies that $A_{1+\bar{t}}$ needs to be fixed as 0 for $0 \le \bar{t} < fl_z^5(a,b) = 2$.*

- *Here, $UL_z(a,b) = \{18\}$. As the monomial $A_{13}A_{18}$ contains A_{18}, therefore $FS_z^{18} = S(A_{13}A_{18} >> (l-18)) \setminus \{\{A_{20}\}\} = S(A_{15}A_{20}) \setminus \{\{A_{20}\}\} = \{\{A_{15}\}\}$. Further, $NL_g(a+l-18,b) = NL_g(5,6) = \{5\}$ as A_5A_{16} is a monomial in g. Hence,*

$fl_z^{18}(a,b) = b - \min NL_g(5,6) + 1 = 6 - 5 + 1 = 2$. *Here,* A_{5+t} *is contained in* A_{20+t} *which is a part of monomial* $A_{15+t}A_{20+t}$ *for* $0 \le t < fl_z^{18}(a,b) = 2$. *Hence, the monomials* $A_{15+t}A_{20+t}$ *can be eliminated by fixing* $A_{15+t} = 0$ *for* $0 \le t \le 1$.

- *Now we have the following information on the cipher in Example 5 with* $a = 3, b = 6$.

  . $NL_z(a,b) \cup UL_z(a,b) = \{4,5,18\}$;

  . $fl_z^4(a,b) = 3, fl_z^5(a,b) = 2, fl_z^{18}(a,b) = 2$;

  . $FS_z^4 = \{\{A_{15}\},\{A_9,A_{11}\}\}, FS_z^5 = \{\{A_1\}\}, FS_z^{18} = \{\{A_{15}\}\}$.

  From this collection of information, we can enumerate possible sets of state variables that can be fixed as 0. *The index of every state bit needs to be extended till its fixing length. Here,* $\{A_9, A_{11}\} \in FS_4^z$ *implies that one of the* A_9 *and* A_{11} *with its corresponding state bits of indices up to fixing length needs to be* 0. *Hence, the possible sets are*

  1. $\{A_{15},A_{16},A_{17}\} \cup \{A_9,A_{10},A_{11}\} \cup \{A_1,A_2\} \cup \{A_{15},A_{16}\} = \{A_1,A_2,A_9,A_{10},A_{11},A_{15},A_{16},A_{17}\}$ *and*
  2. $\{A_{15},A_{16},A_{17}\} \cup \{A_{11},A_{12},A_{13}\} \cup \{A_1,A_2\} \cup \{A_{15},A_{16}\} = \{A_1,A_2,A_{11},A_{12},A_{13},A_{15},A_{16},A_{17}\}$

  In both cases the size of set is 8, *i.e., we need to fix* 8 *state bits as* 0 *to choose any set.*

From the sets $NL_z(a,b) \cup UL_z(a,b)$ and corresponding FS_z^p and $fl_z^p(a,b)$, we can enumerate possible sets of fixing state bits. Then we choose the set with minimum size for our purpose. To find all possible such sets of state bits, we have the following notation and an easily derivable lemma.

Notation 2. • *Let* $T = \{A_{j_1}, A_{j_2}, \ldots, A_{j_p}\}$ *be a set of some state bits and* t *be a positive integer then* (T,t) *is defined by*

$(T,t) = \{\{A_{j_1}, A_{j_1+1}, \ldots, A_{j_1+t-1}\}, \{A_{j_2}, A_{j_2+1}, \ldots, A_{j_2+t-1}\}, \ldots,$
$\{A_{j_p}, A_{j_p+1}, \ldots, A_{j_p+t-1}\}\}$.

- *Let U, V be two sets containing some sets of state bits. For example, $U = (T_1, t_1), V = (T_2, t_2)$ where T_1, T_2 are two sets of state bits and t_1, t_2 are two positive integers. Then we define*

$$U \otimes V = \{X \cup Y : X \in U, Y \in V\}.$$

It is obvious that $(T, 0) = \varnothing$. As per this notation, in the above example, it is observed that
$(\{A_{15}\}, 3) = \{A_{15}, A_{16}, A_{17}\}$ and $(\{A_9, A_{11}\}, 3) = \{\{A_9, A_{10}, A_{11}\}, \{A_{11}, A_{12}, A_{13}\}\}$. So,
$(\{A_{15}\}, 3) \otimes (\{A_9, A_{11}\}, 3) = \{\{A_9, A_{10}, A_{11}, A_{15}, A_{16}, A_{17}\}, \{A_{11}, A_{12}, A_{13}, A_{15}, A_{16}, A_{17}\}\}$.
Hence the two sets in the above example are $(\{A_{15}\}, 3) \otimes (\{A_9, A_{11}\}, 3) \otimes (\{A_1\}, 2) \otimes$
$(\{A_{15}\}, 2) = \{\{A_1, A_2, A_9, A_{10}, A_{11}, A_{15}, A_{16}, A_{17}\}, \{A_1, A_2, A_{11}, A_{12}, A_{13}, A_{15}, A_{16}, A_{17}\}\}$.

Lemma 4. *The set of all possible sets of fixing state bits is*

$$ASF_z(a,b) = \bigotimes_{p \in NL_z(a,b) \cup UL_z(a,b)} \bigotimes_{T \in FS_z^p} (T, fl_z^p(a,b)).$$

Since each element $T \in FS_z^p$ spreads into $|T|$ many branches,
$\prod_{p \in NL_z(a,b) \cup UL_z(a,b)} \prod_{T \in FS_z^p} |T|$ many possible sets arose and the cardinality of each set is at most $\sum_{p \in NL_z(a,b) \cup UL_z(a,b)} \sum_{T \in FS_z^p} fl_z^p(a,b)$.
To minimize the number of fixing bits, we need to choose a set with the least cardinality.

Note 3. *For a linear interval (a,b) where $a \le b$, it may not always be possible to have a system $b - a + 1$ equations of the form Equation 5.2. In such situations we go for shorter interval $(a,d), d \le b$. Such situations are as following.*

1. *Let A_q be the last linear bit in the ordered set $\mathcal{L}$. If $b - a + 1 > l - q$, then the*

update function g substitutes A_l at $l-q$ clocks. If $NL_g(a+l-q,b) = \varnothing$, then there is no problem. If $NL_g(a+l-q,b) \neq \varnothing$ then we can have the desired system of equations for the interval (a,d) where $d = \min\{p : A_p \in NL_g(a+l-q,b)\} - 1$.

2. *Let the state bits $A_p, \ldots, A_{p+fl_z^p(a,b)}$ (which are generated from the starting state bit A_p) are needed to be fixed. If, for a $0 \leq j \leq fl_z^p(a,b)$, $A_{p+j} \in R$ or $p+j \geq l$, then the last index b needs to be reduced. In this case, we can choose the shorter interval (a,d) where $p + fl_z^p(a,d) < a$ or $b < p + fl_z^p(a,d) < l$.*

Now we present an algorithm to find the maximum possible recovering bits with fixing bits in a given linear interval (a,b). The Algorithm 7 is presented when $a \leq b$. Here, the parameters g, z, l, q are the update function, the output function, the length of the state, and the index of the last linear bit in $\mathcal{L}$, respectively.

Here we will explain Algorithm 7. Given an FSR based stream cipher, at first, we will check that which linear interval produces the most recovering bits with as less as possible number of fixing bits. For a given linear interval $(a,b), a \leq b$, we will have the corresponding FSR with its length l and the index (q) of the last linear bit in $\mathcal{L}$. The steps 1-6 in Algorithm 7 deal with the fact in Note 3[Item 1]. We search for the maximum number of bits that can be recovered in the for loop from Step 8 by increasing the length of the interval 1 each time. For each $p \in NL_z(a,b) \cup UL_z(a,b)$, we check the possible fixing bits by taking the variables from the monomials involving A_p in the for loop in step 11. Steps 15-19 deal with the fact in Note 3[Item 2] to remove the possible fixing bits from the list. Steps 20-22 check that if it is possible to fix at least one bit to remove the monomial containing A_p. If it is not possible, then we exit from the algorithm, and the set of bits R and C for the previous interval are the desired output.

The Algorithm 7 is discussed for the interval (a,b) such that $a \leq b < l$. For the interval (a,b) such that $b > l$, the recovery process is presented in Note 4.

Complexity of Algorithm 7: The algorithm runs four nested for loops for e, i, j and A_p at the steps 8, 11, 14 and 15 respectively. The loop e can run for at most

Algorithm 7: State bit recovery in a linear interval

Input : A linear interval $(a, b) \in \mathcal{I}$ such that $a < b$
Output: Set of recovery bits R and set of fixing bits C

if $NL_g(a+l-q, b) = \varnothing$ **then**
  $b' = b$;
end
else
  $b' = \min\{p : A_p \in NL_g(a+l-q, b)\} - 1$;
end
$R = C = \varnothing$;
for *e from a to* b' **do**
  Compute $NL_z(a,e) \cup UL_z(a,e) = \{p_1, \ldots, p_{|NL_z(a,e) \cup UL_z a,e|}\}$;
  $P = \{\varnothing\}$;
  for *i from 1 to* $|NL_z(a,e) \cup UL_z(a,e)|$ **do**
    Compute $FS_z^{p_i} = \{T_1^{p_i}, \ldots, T_{|FS_z^{p_i}|}^{p_i}\}$;
    $t_i = fl_z^{p_i}(a,e)$;
    for *j from* 1 *to* $|FS_z^{p_i}|$ **do**
      for $A_p \in T_j^{p_i}$ **do**
        if $(p + t_i) \geq l$ *or* $a \leq (p + t_i) \leq e$ **then**
          Remove A_p from $T_j^{p_i}$;
        end
      end
      if $T_j^{p_i} = \varnothing$ **then**
        exit;
      end
      Calculate $P = P \otimes (T_j^{p_i}, t_i)$;
    end
  end
  $C = \{S \in P : |S| \text{ is minimum}\}$;
  $R = \{A_a, \ldots, A_e\}$.;
end

$\max\{len(I) : I \in \mathcal{I}\}$, i.e., the length of the largest linear interval. The loop i runs for $|NL_z(a,e) \cup UL_z(a,e)|$. If the output function z is dense then the value of i will be large. Further, the value of i increases as e increases. Let consider p_0 be the probability that a state bit is in a nonlinear monomial. Then the number of state bits from $A_a, \ldots, A_e$ and $A_{l-e+a-1}, \ldots, A_{l-1}$ in $NL_z(a,e) \cup UL_z(a,e)$ is expected to be $2p_0(e-a+1)$. Then further for each $p_i \in NL_z(a,e) \cup UL_z(a,e)$, the loop j runs for the number of nonlinear monomials in z containing the state bit A_{p_i}. Consider the expected number

of nonlinear monomials in z containing a state bit is q. Further, the number of loops at step 15 or the number of union operations in the computation of $P \otimes (T_j^{p_i}, t_i)$ in Step 23 is the number of state bits in the monomial which is bounded by the degree of z. Hence the expected time complexities is $l_m \times (2p_0 l_m) \times q \times \deg(z) = 2p_0 q l_m^2 \deg(z)$ where $l_m = \max\{len(I) : I \in \mathcal{I}\}$.

For efficiency purpose the output function of stream cipher is generally very sparse. As a result p_0, q and $\deg(z)$ are low. Considering the parameters $p_0, q, \deg(z)$ as constants, the time complexity is $O(l_m^2)$.

Note 4. *If A_a is the last linear bit and A_b is the first linear bit in z, we can also attempt to recover state bits $A_a, A_{a+1}, \ldots, A_{l-1}, A_i, \ldots, A_{i+b-1}$ where $0 \le i < a - b + 1$. In this situation, the LHS linear bit passes through A_l which is updated by the update function g to have A_i. In Example 5, we can attempt to recover state bits $A_7, A_8, \ldots, A_{19}, A_0, A_1, A_2$. In this situation, we can use Algorithm 7 separately using the output function z and z_g for recovering the states $A_a, \ldots A_{l-1}$ and $A_i, \ldots, A_{i+b-1}$ respectively. Here, z_g is the output function by shifting the involved states $l - b$ clocks and using the update function g appropriately. Such a situation is used to recover the state bits of Grain-128a in Subsection 5.3.3.2.*

5.3.2.1 Combining different intervals to recover more bits with fewer fixing bits

Using Algorithm 7, one can recover $t = e - a + 1$ state bits $A_a, A_{a+1}, \ldots, A_e$ where $e \le b$ from an input of a linear interval (a, b). Further, it is possible to add another interval to increase the number of recovering bits and/or decrease the number of fixing bits. Algorithm 9 is similar as Algorithm 7 with a little difference for computing the sets $NL_z(a, b)$ and $UL_z(a, b)$, which is described below.

Consider that the initial t state bits recovered from the linear interval (a, b) using Algorithm 7. In this case, we recover the state bits $A_a, A_{a+1}, \ldots, A_e$ from first t equations where $e = a + t - 1$. Combining another linear interval (c, d), we will have

more equations including the state variables $A_c, \ldots, A_d$. Since the first t clocks are used to recover t state bits $A_a, \ldots, A_e$, the state variables $A_c, \ldots, A_{c+t-1}$ are too present as linear bits in the equations respectively. As a result, the state bits $A_c, \ldots, A_{c+t-1}$ can not be recovered and we attempt to recover some state bits from $A_{c+t}, \ldots, A_d$. Hence, we have a condition to choose the second interval such that $d - c \geq t$. Therefore, we need to eliminate the nonlinear monomials containing the state bits $A_{c+t}, \ldots, A_d$. The sets $NL_z(c+t, d)$ and FS_z^p for $p \in NL_z(c+t, d)$ are required for the elimination of monomials. Further, the state variables in $NL_z(c, c+t-1)$ become some variables from $A_{c+t}, \ldots, A_d$ and the monomials containing these variables need to be eliminated. If $q \in NL_z(c, c+t-1)$, then after $c+t-q+j$ clocks the state bit A_q is shifted to A_{c+t+j} for $j \geq 0$ and the state bits A_{c+t+j} are present in respective monomials. Since we need to recover the state bits $A_{c+t}, \ldots, A_d$, these monomials have to be eliminated. In this case, we need to include $c+t$ in $NL_z(c+t, d)$ and the set of variables $\{(S(M >> (c+t-q)) \setminus R : M$ is a nonlinear monomial containing A_q in $z\}$ need to be added with FS_z^{c+t} for every $q \in NL_z(c, c+t-1)$. Now we define $\overline{NL}_z(c+t, d)$ and $\overline{FS}_z^p$ as discussed above.

Definition 35. *Given a function z on the state bits $A_i, 0 \leq i < l$ and integers c, d, t such that $(0 \leq c < d < l, 0 < t \leq d - c - 1)$.*

- *The nonlinear index list in between the indices $c+t, d$ for z is defined by $\overline{NL}_z(c+t, d) = NL_z(c+t, d) \cup \{c+t : \text{ if } NL_z(c, c+t-1) \neq \varnothing\}$.*
- *For $p \in \overline{NL}_z(c+t, d) \cup UL_z(c, d)$, the set of fixing state bits for the state bit A_p is defined as*
  - *if $p = c+t, \overline{FS}_z^p = \cup_{q \in NL_z(c, c+t-1)} \{(S(M >> (c+t-q)) \setminus R : M$ is a nonlinear monomial containing S_q in $z\} \cup FS_z^p$.*
  - *if $p \neq c+t$, $\overline{FS}_z^p = FS_z^p$.*

Note 5. *If $c+t > \max NL_g(c+l-p, d)$, $p \in UL_z(c, d)$, then $fl_z^p(c, d) = d - (c+t) + 1$, else it is same as Lemma 3.*

Example 6. *Consider the output function* $z = A_2 + A_6 + A_{12} + A_1A_5 + A_4A_{14} + A_4A_9A_{13} + A_{13}A_{18}$ *with length of FSR* $l = 20$. *Consider the the linear interval* $(2,5)$, *we have* $NL_z(2,5) \cup UL_z(2,5) = \{4,5,18\}$ *and* $FS_z^4 = \{\{A_{14}\},\{A_9,A_{13}\}\}, FS_z^5 = \{\{A_1\}\}, FS_z^{18} = \{\{A_{15}\}\}$. *Considering the second interval* $(6,11)$, *we have* $NL_z(6,11) \cup UL_z(6,11) = \{9,18\}$ *and* $FS_z^9 = \{\{A_4,A_{13}\}\}, FS_z^{18} = \{\{A_{15}\}\}$. *If Algorithm 7 is used to recover* A_2 *to* A_5, *then from* $t = 5-2+1 = 4$ *clocks onwards the state bits from* A_{10} *to* A_{11} *can possibly be recovered using the second linear interval* $(6,11)$. *In that scenario, the state bit* A_6 *of recovering bit for second case is shifted by* $t = 4$. *Here,* $NL_z(6,9) = \{A_9\} \neq \varnothing$. *Hence,* $\overline{NL}_z(10,11) \cup UL_z(6,11) = NL_z(10,11) \cup \{10\} \cup UL_z(6,11) = \{10,18\}$. *Also,* $\overline{FS}_z^{10} = \{\{A_5,A_{14}\}\}$ *and* $FS_z^{18} = \{\{A_{15}\}\}$. *Then we can use Algorithm 7 with* $\overline{NL}_z(10,11) \cup UL_z(6,11)$ *and* $\overline{FS}_z^p$, $p \in \overline{NL}_z(10,11) \cup UL_z(6,11)$ *instead of using* $NL_z(10,11) \cup UL_z(6,11)$ *and* FS_z^p, $p \in NL_z(10,11) \cup UL_z(6,11)$.

Hence recovering state bits by combining two different intervals (from same FSRs or from different FSRs), we propose Algorithm 9 by extending Algorithm 7. In the first part of Algorithm 9(i.e., from the line no 8 to the line number 27), Algorithm 7 is run on the input of interval (a,e) with an extra condition at line number 15 (i.e., $c+t \leq (p+t_i) \leq d'$). As we priorly know that $t(=e-a+1)$ state bits can be recovered from the linear interval (a,b), supplying (a,e) as the input for first interval is justified. Algorithm 9 is presented in Appendix A.

Similarly, it is possible to add three or more intervals to get more recovering bits with fewer fixing bits. However, the maximum number of state bits that can be recovered is the length of the largest linear interval. Even if we achieved the maximum number of recovering state bits, we try to combine the possible linear intervals to get fewer fixing bits. In this process, we first try with single linear intervals. Let consider that from linear interval I_j we could recover $e_j - a_j + 1$ state bits by fixing c_j state bits. Then we search by combining the interval I_j with other intervals of length at least $e_j - a_j + 2$. Then from the combination we hope to recover (i) more than $e_j - a_j + 1$ state bits or (ii) $e_j - a_j + 1$ state bits by fixing lesser than c_j state bits. We implement

this method on Lizard and Grain-128a (in Section 5.3.3) and get the best results.

5.3.3 Implementation of Algorithm on ciphers

In this section we explain the implementation of our technique on Lizard and Grain-128a to recover state bits and the fixing bits.

5.3.3.1 State bit recovery of Lizard

The set of linear bits in z function of Lizard is

$$\mathcal{L} = \{S_{23}, B_5, B_7, B_{11}, B_{30}, B_{40}, B_{45}, B_{54}, B_{71}\}.$$

It is observed from the Algorithm 7 that we can recover 9 state bits S_{23} to S_{31} of NFSR1 by fixing only 8 bits of NFSR2. Further, we present following three best possible cases in NFSR2 using Algorithm 7.

Case-1(using the linear interval $(11, 29)$): It is possible to recover 8 state bits from B_{11} to B_{18} by fixing 11 state bits. The state bit B_{19} or further state bits can not be recovered using our technique because the term $B_{8+t}B_{82+t}, t \geq 8$ can not be vanished as B_{8+t} is already recovered and B_{82+t} is updated by g.

Case-2 (using the linear interval $(54, 70)$): In this case, 14 state bits from B_{54} to B_{67} can be recovered by fixing 22 state bits. It is not possible to recover more than 14 bits as the term $B_{44+t}B_{76+t}, t \geq 14$ can not be vanished.

Case-3 (using the linear interval $(71, 90 + 4)$): Here the output functions z and z_g are used for recovery as described in Note 4. In this case, it is possible to recover maximum 18 state bits by fixing 42 bits.

Further, we aim to increase the number of recovering bits with as less possible number of fixing bits. For that purpose, we use Algorithm 9 (see Note 6) combining two linear

intervals. Combining the linear intervals $(23, 31)$ of NFSR1 and $(54, 70)$ of NFSR2, we get the best possible result, i.e., it is possible to recover

- $10, 11, 12, 13, 14, 15, 16, 17$ state bits by fixing
- $10, 12, 14, 16, 18, 20, 22, 24$ state bits respectively.

By combining the intervals $(23, 31)$ of NFSR1 and $(71, 90 + 4)$ of NFSR2, we can recover

- $18, 19, 20, 21, 22, 23, 24$ state bits by fixing
- $38, 40, 42, 44, 46, 48, 50$ state bits respectively.

Note 6. *Using Algorithm 9, we could recover* $10, 11, 12, 13, 14, 15, 16, 17$ *state bits by fixing* $10, 12, 14, 16, 19, 24, 26, 29$ *state bits respectively. Observing the following fact in Lizard, we tweaked Algorithm 9 to improve the result. As a result, the same number of state bits can be recovered by fixing* $10, 12, 14, 16, 18, 20, 22, 24$ *state bits respectively. We recover state bits of Lizard by combining two linear intervals* $(23, 31)$ *of NFSR1 and* $(54, 70)$ *of NFSR2. Since the first* 9 *state bits are recovered from NFSR1 using the interval* $(23, 31)$*, the remaining state bits will be recovered from the interval* $(63, 70)$ *of NFSR2. Here,* $UL_z(54, 70) = \{76, 78, 82\}$ *as per Definition 33 and Note 2. Therefore, the set of fixing state bits are*

- $FS^{76} = S(B_{44} >> 14) = \{B_{58}\}$,
- $FS^{78} = S(B_1 B_{19} B_{27} B_{43} B_{57} B_{66} >> 12) = \{B_{13}, B_{31}, B_{39}, B_{55}, B_{78}\}$,
- $FS^{82} = S(B_8 >> 8) = \{B_{16}\}$.

We observed that the only monomial $B_{62}B_{68}B_{72}$ *of the state update function* g *in Lizard (Equation 2.13) contains a state variable from* B_{63} *to* B_{70}*. Hence, it is enough to vanish only the monomial* $B_{62}B_{68}B_{72}$ *instead of vanishing the whole* g *as we do in Algorithm 9. Further, vanishing the monomial* $B_{62}B_{68}B_{72}$*, we need to vanish only*

one of the state variables B_{62}, B_{68} *and* B_{72}*. Since* B_{62}, B_{68} *can not be fixed due to recovering state bits, we can fix* B_{72}*. Therefore we tweaked Algorithm 9 by adding* B_{72} *with the original sets as*

- $FS^{76} = \{B_{58}, B_{72}\}$,
- $FS^{78} = \{B_{13}, B_{31}, B_{39}, B_{55}, B_{72}, B_{78}\}$,
- $FS^{82} = \{B_{16}, B_{72}\}$.

The results on the number of recovering and fixing bits are presented in Table 5.8 and the comparison with the previous result by Maitra et al. [70] is shown in Table 5.9. The recovery process of 17 state bits is discussed below.

Recovering 17 state bits of Lizard: It is possible to recover 17 state bits (9 state bits from NFSR1 and 8 state bits from NFSR2) by fixing 24 state bits. To discuss the recovery process, we consider the two linear intervals (a, e) and (c, d), where

1. $S_a = S_{23}$ and $S_e = S_{31}$
2. $B_c = B_{54}$ and $B_d = B_{70}$.

The Algorithm 9 is implemented in SAGE and the intermediate steps are explained in Table 5.7. In Table 5.7, $p \in NL_z(23, 31) \cup UL_z(23, 31)$ and $T_i^p \in FS_z^p$ for NFSR1. Then $R = \{S_{23}, \ldots, S_{30}, S_2\}$. Since 9 state bits of NFSR1 are recovered from first linear interval, the starting bit B_{54} of NFSR2 is shifted by 9 bits. Hence we can recover the bits of NFSR2 from B_{63}. $NL_z(54, 70) \cup UL_z(54, 70) = \{57, 61, 63, 64, 65, 66, 67, 68, 76, 78, 82\}$, we have $\overline{NL}_z(63, 70) \cup UL_z(54, 70) = \{63, 64, 65, 66, 67, 68, 76, 78, 82\}$. Therefore $p \in \overline{NL}_z(63, 70) \cup UL_z(54, 70)$ and $T_i^p \in \overline{FS}_z^p$ for NFSR2. Since 9 state bits are already recovered, $R = \{S_{23}, \ldots, S_{30}, S_2, B_{63}, \ldots, B_{70}\}$. The other symbols in Table 5.7 have their usual meaning.

In total we get $|ASF_z(54, 70)| = |T_1^{24}| \times |T_1^{63}| \times |T_1^{64}| \times |T_1^{65}| \times |T_1^{66}| \times |T_1^{67}| \times |T_1^{76}| \times |T_1^{78}| \times |T_1^{82}| = 2 \times 6 \times 5 \times 3 \times 4 \times 3 \times 6 \times 2 \times 2 \times 6 \times 2 = 6,22,080$ combinations of set of fixing bits. Finally, $R = \{S_{23}, \ldots, S_{30}, S_2, B_{63}, \ldots, B_{70}\}$ and $C = \{S \in ASF_z(54, 70) :$

Table 5.7: Intermediate steps of Algorithm 9 (using Note 6) to recover state bits of Lizard

(a,e)	p	$T_i^p \in FS_z^p$	$fl_z^p(a,e)$	$ASF_z(a,e) = P$
$(23,31)$	24	$\{S_1, B_{38}\} = T_1^{24}$	8	$P = (T_1^{24}, 8)$
(c,d)	p	$T_i^p \in \overline{FS}_z^p$	$fl_z^p(c,d)$	$ASF_z(c,d) = P'$
$(54,70)$	63	$S(B_1B_{19}B_{27}B_{43}B_{66}B_{78} >> 6) = \{B_7, B_{25}, B_{33}, B_{49}, B_{72}, B_{84}\} = T_1^{63}$ $S(B_6B_{14}B_{26}B_{32}B_{47} >> 2) = \{B_8, B_{16}, B_{28}, B_{34}, B_{49}\} = T_2^{63}$ $S(B_{38}S_1S_{24}) = \{B_{38}, S_1, S_{24}\} = T_3^{63}$	8	$P' = P \otimes (T_1^{63}, 8)$ $P' = P' \otimes (T_2^{63}, 8)$ $P' = P' \otimes (T_3^{63}, 8)$
	64	$\{B_{13}, B_{29}, B_{50}, B_{75}\} = T_1^{64}$	7	$P' = P' \otimes (T_1^{64}, 7)$
	65	$\{B_2, B_{28}, B_{41}\} = T_1^{65}$	6	$P' = P' \otimes (T_1^{65}, 6)$
	66	$\{B_1, B_{19}, B_{27}, B_{43}, B_{57}, B_{78}\} = T_1^{66}$	5	$P = P \otimes (T_1^{66}, 5)$
	67	$\{B_{34}, B_{73}\} = T_1^{67}$	4	$P' = P' \otimes (T_1^{67}, 4)$
	76	$\{B_{58}, B_{72}\} = T_1^{76}$	3	$P' = P' \otimes (T_1^{76}, 3)$
	78	$\{B_{13}, B_{31}, B_{39}, B_{55}, B_{72}, B_{78}\} = T_1^{78}$	3	$P' = P' \otimes (T_1^{78}, 3)$
	82	$\{B_{16}, B_{72}\} = T_1^{82}$	9	$P' = P' \otimes (T_1^{82}, 9)$

$|S|$ is minimum}. The fixing bits are presented in Table 5.8. The obtained system of equations (in the form of Equation 5.2) is presented in Appendix B.

5.3.3.2 Bit recovery of Grain-128a

Here we discuss the recovery process of 35 and 48 state bits in Grain-128a. The set of linear bits in Grain-128a is

$$\mathcal{L} = \{b_2, b_{15}, b_{36}, b_{45}, b_{64}, b_{73}, b_{89}, s_{93}\}.$$

Two largest linear intervals are $(89, 128+1)$ for NFSR and $(93, 128+92)$ for LFSR. We could recover more bits from the latter interval.

To recover state bits from the interval $(93, 128+92)$, first we use Algorithm 7 to recover the state bits from s_{93} to s_{127}. Further, we observed that the largest linear intervals in LFSR update function s_{128} (see Equation 2.9) are $(7, 37)$ and $(38, 69)$. We

Table 5.8: Recovering of 7 to 24 state bits of Lizard

No. of recovering bits	Recovering bits	No. of fixing bits	Fixing Bits	Keystream
7	$S_{23}, \ldots, S_{29}$	6	$B_{38} = \cdots = B_{43} = 0$	$z_0, \ldots, z_6$
8	$S_{23}, \ldots, S_{30}$	7	$B_{38} = \cdots = B_{44} = 0$	$z_0, \ldots, z_7$
9	$S_{23}, \ldots, S_{30}, S_2$	8	$B_{38} = \cdots = B_{45} = 0$	$z_0, \ldots, z_8$
10	$S_{23}, \ldots, S_{30}, S_2$ B_{63}	10	$B_{38} = \cdots = B_{45} = 0$ $B_{49} = 0, B_{73} = 0$	$z_0, \ldots, z_9$
11	$S_{23}, \ldots, S_{30}, S_2$ B_{63}, B_{64}	12	$B_{38} = \cdots = B_{45} = 0$ $B_{49} = B_{50} = 0$ $B_{73} = B_{74} = 0$	$z_0, \ldots, z_{10}$
12	$S_{23}, \ldots, S_{30}, S_2$ B_{63}, B_{64}, B_{65}	14	$B_{38} = \cdots = B_{45} = 0$ $B_{49} = \cdots = B_{51} = 0$ $B_{73} = \cdots = B_{75} = 0$	$z_0, \ldots, z_{11}$
13	$S_{23}, \ldots, S_{30}, S_2$ $B_{63}, \ldots, B_{66}$	16	$B_{38} = \cdots = B_{45} = 0$ $B_{49} = \cdots = B_{52} = 0$ $B_{73} = \cdots = B_{76} = 0$	$z_0, \ldots, z_{12}$
14	$S_{23}, \ldots, S_{30}, S_2$ $B_{63}, \ldots, B_{67}$	18	$B_{38} = \cdots = B_{45} = 0$ $B_{49} = \cdots = B_{53} = 0$ $B_{73} = \cdots = B_{77} = 0$	$z_0, \ldots, z_{13}$
15	$S_{23}, \ldots, S_{30}, S_2$ $B_{63}, \ldots, B_{68}$	20	$B_{34} = \cdots = B_{45} = 0$ $B_{72} = \cdots = B_{79} = 0$	$z_0, \ldots, z_{14}$
16	$S_{23}, \ldots, S_{30}, S_2$ $B_{63}, \ldots, B_{69}$	22	$B_{34} = \cdots = B_{45} = 0$ $B_{72} = \cdots = B_{81} = 0$	$z_0, \ldots, z_{15}$
17	$S_{23}, \ldots, S_{30}, S_2$ $B_{63}, \ldots, B_{70}$	24	$B_{34} = \cdots = B_{46} = 0$ $B_{72} = \cdots = B_{82} = 0$	$z_0, \ldots, z_{16}$
18	$S_{23}, \ldots, S_{30}, S_2$ $B_{80}, \ldots, B_{88}$	38	$B_8 = \cdots = B_{26} = 0$ $B_{38} = \cdots = B_{56} = 0$	$z_0, \ldots, z_{17}$
19	$S_{23}, \ldots, S_{30}, S_2$ $B_{80}, \ldots, B_{89}$	40	$B_8 = \cdots = B_{27} = 0$ $B_{38} = \cdots = B_{57} = 0$	$z_0, \ldots, z_{18}$
20	$S_{23}, \ldots, S_{30}, S_2$ $B_{80}, \ldots, B_{89}, B_0$	42	$B_8 = \cdots = B_{28} = 0$ $B_{38} = \cdots = B_{58} = 0$	$z_0, \ldots, z_{19}$
21	$S_{23}, \ldots, S_{30}, S_2$ $B_{80}, \ldots, B_{89},$ B_0, B_1	44	$B_8 = \cdots = B_{29} = 0$ $B_{38} = \cdots = B_{59} = 0$	$z_0, \ldots, z_{20}$
22	$S_{23}, \ldots, S_{30}, S_2$ $B_{80}, \ldots, B_{89},$ B_0, B_1, B_2	46	$B_8 = \cdots = B_{30} = 0$ $B_{38} = \cdots = B_{60} = 0$	$z_0, \ldots, z_{21}$
23	$S_{23}, \ldots, S_{30}, S_2$ $B_{80}, \ldots, B_{89},$ B_0, B_1, B_2, B_3	48	$B_8 = \cdots = B_{31} = 0$ $B_{38} = \cdots = B_{61} = 0$	$z_0, \ldots, z_{22}$
24	$S_{23}, \ldots, S_{30}, S_2$ $B_{80}, \ldots, B_{89},$ B_0, B_1, B_2, B_3, B_4	50	$B_8 = \cdots = B_{32} = 0$ $B_{38} = \cdots = B_{62} = 0$	$z_0, \ldots, z_{23}$

Table 5.9: Comparison of number of fixing state bits of Lizard

Instances	No. of recovering bits (r)	No. of fixing bits(f)	No. of guessing bits($n-f-r$)	No. of fixing in [70]
1	7	6	108	–
2	8	7	106	–
3	9	8	104	12
4	10	10	101	16
5	11	12	98	20
6	12	14	95	24
7	13	16	92	27
8	14	18	89	30
9	15	20	86	–
10	16	22	83	–
11	17	24	80	–
12	18	38	65	–
13	19	40	62	–
14	20	42	59	–
15	21	44	56	–
16	22	46	53	–
17	23	48	50	–
18	24	50	47	–

can not recover many bits from the latter interval. So we recover state bits from the former interval $(7, 37)$ as we need to handle only one monomial $s_{13}s_{20}$ in z function after recovering s_{93} to s_{127}. Hence the state bit recovery is done from two intervals.

1. **Recovery of s_{93} to s_{127}**: Here $NL_z(93, 127) \cup UL_z(93, 127) = \{94\}$ and $FS_z^{94} = \{\{b_{12}, b_{95}\}\}$. $(T_1^{94}, fl_z^{94}(93, 127)) = (\{b_{12}, b_{95}\}, 34)$. Hence $R = \{S_{93}, \ldots, S_{127}\}$.

2. **Recovery of s_7 to s_{19}**: In this case the update function g is used to replace S_{128} in the output function z. Hence, we need to use the updated output function z_g for computing the nonlinear index list and updating nonlinear index list. Here, $NL_{z_g}(7, 19) \cup UL_{z_g}(7, 19) = \{8, 13\}$. Then $FS_{z_g}^{8} = \{\{b_{12}\}\}$, $FS_{z_g}^{13} = \{\{s_{20}\}\}$. Now we calculate $(T_1^8, fl_z^8(7, 19)) = (\{b_{12}\}, 12)$ and $(T_1^{13}, fl_z^{13}(7, 19)) = (\{s_{20}\}, 7)$. Here we could recover $\{S_7, \ldots, S_{19}\}$. The final $R = \{S_{93}, \ldots, S_{127}, S_7, \ldots, S_{19}\}$.

Since we recover 13 bits of LFSR from second case and the recovery process of second case uses the update function g, $(T_1^{94}, 34)$ will be $(T_1^{94}, 34 + 13)$. The recovery process

of 48 state bits of Grain-128a is given in Table 5.10. The symbols have their usual meaning. Here $(a, b) = (93, 128 + 12) = (93, 140)$ and $(c, d) = (7, 19)$.

Table 5.10: Intermediate steps of Algorithm 7 (using Note 4) to recover state bits of Grain-128a

(a, b)	p	$T_i^p \in FS_z^p$	$fl_z^p(a, b)$	$ASF_z(a, b) = P$
$(93, 140)$	94	$T_1^{94} = \{b_{12}, b_{95}\}$	47	$P = (T_1^{94}, 47)$
(c, d)	p	$T_i^p \in FS_{z_g}^p$	$fl_{z_g}^p(c, d)$	$ASF_{z_g}(c, d) = P$
$(7, 19)$	8	$T_1^8 = \{b_{12}\}$	12	$P = P \otimes (T_1^8, 12)$
	13	$T_1^{13} = \{s_{20}\}$	7	$P = P \otimes (T_1^{13}, 7)$

Then $C = \{S \in ASF_{z_g}(93, 140) : |S| \text{ is minimum})\} = \{b_{12}, \ldots, b_{58}, s_{20}, \ldots, s_{26}\}$.

Finally, we recover 48 state bits of Grain-128a with a 54 number of fixing bits. The results on state bits in the recovery process of Grain-128a are presented in Table 5.11.

Table 5.11: Recovering of (up to) 48 state bits of Grain-128a

Step (i)	Constrains	Key Bits	Removed Bits	Feedback Bits	Bit Recover
$0 - 32$	$b_{12+i} = 0$	z_i	s_{94+i}		s_{93+i}
$33 - 34$	$b_{12+i} = 0$	z_i	s_{94+i}	b_{95+i}	s_{93+i}
$35 - 40$	$b_{12+i} = 0$, $s_{20+j} = 0, 0 \le j \le 5$	z_i	s_{94+i}, s_{13+j}	b_{95+i}	s_{7+j}
41	$b_{12+i} = 0, s_{26} = 0$	z_i	s_{94+i}, s_{19}	b_{95+i}	s_{13}
$42 - 46$	$b_{12+i} = 0$	z_i	s_{94+i}	b_{95+i}	$s_{14}, \ldots, s_{18}$
47		z_i		s_{94+i}, b_{95+i}	s_{19}

5.4 State Recovery by TMDTO Attack Approach

To exploit BSW sampling with the BS-TMDTO attack, it is needed to fix the first k bits of the keystream. Hence, the available data having first k bits with a pattern is

reduced to $D' = \frac{D}{2^k}$ for the TMDTO. This generic technique was proposed in [77] and the attack was deployed on the stream cipher A5/1. Suppose the sampling resistance of a stream cipher is R $(0 < R < 1)$, then both ultimate state space size N and data D are reduced by RN and $RD(> 1)$ respectively for the attack. Hence, the trade-off curve is the same as $TM^2D^2 = N^2$, but the range of $T > D^2$ is wider by $T > (RD)^2$ and the number of disk operations is reduced from t to Rt.

5.4.1 Conditional TMDTO attack

As the output function of FSR stream ciphers is nonlinear, with given specific first k keystream bits, r state bits can be recovered by fixing f state bits for some r and f. For example, Table 5.9 and Table 5.11 present our results on k, r, f for ciphers Lizard and Grain-128a respectively. With this pattern of keystream bits and recovering bits, the BSW-sampling technique can further be improved. Therefore, this technique of cryptanalysis is called as *conditional BSW-sampling TMDTO attack* or simply, *conditional TMDTO attack*. Recently this is a popular technique used to analyze FSR-based stream ciphers [20, 69, 70]. Section 4.2 presents a method to find a set of conditional state bits C and a set of recovering state bits R. Here, we denote

- n be the size of the state;
- $f = |C|$ be the number of fixing/conditioned state bits;
- $r = |R|$ be the number of recovering state bits;
- $k = r$ be the number of specific keystream bits.

Like other TMDTO attacks, the conditional TMDTO attack is implemented in two phases (i) offline phase (lookup table construction) and (ii) online phase (search for matching). The attack complexities of this attack is too based on five parameters: pre-processing time complexity (P), memory required to store states in a table (M), online time complexity (T), data required (D) and total search space $(N = 2^n)$. The

primary goal of the TMDTO attack is to minimize the parameters T, M, D and P. Since the parameter P is associated during the offline phase, we do not bother much about it. Therefore, the general goal is to minimize T, M and D.

The general TMDTO curve over BSW-sampling is $TM^2D^2 = N^2$ such that $T \geq D^2$. But in the case of conditional TMDTO, the original search space is reduced due to fixing and recovering some bits in the search space. In that case, the parameters P, M, T, D, and N are related differently. Consider the parameters for conditional TMDTO are

- N': Reduced search space;
- P': Preprocessing time complexity over reduced space;
- M': Memory required for the preprocessing over reduced space;
- T': Online time complexity over reduced space.
- D': Data over reduced space.

We explain the conditional TMDTO attack in the following subsections. One can too follow [70].

5.4.1.1 Offline phase

In this preprocessing phase of this attack, a table of randomly chosen states is stored. Consider that the total number of states is $N = 2^n$. If f state bits are fixed and r state bits are recovered, the number of states reduces to $N' = 2^{n-f-r}$. Now the task is to have a table of N' random states with the least possible collisions. To reduce collision, Hellman [37] proposed to have t different tables which together store N' states. If D' data is available, then the number of tables can be reduced to $\frac{t}{D'}$.

Given a specific pattern ζ of first r bits of data, we now demonstrate to prepare the look up tables. Each table has m rows and t columns such that $\frac{mt^2}{D'} = N'$. The entries of the tables are of $\tau = n - f - r$ bits. For every row of each table, a random string of τ

bits is chosen to be stored as the row's first element. The f fixing bits are joined with the chosen bits to have a string of $n-r$ bits. Then from the r bits of fixing pattern ζ of the keystream, another r bits are recovered (using Equation 5.2). Now we have a state (S) of n bits by joining these new r bits with the $n-r$ bit string. Using an one way function $\mathcal{F}: \{0,1\}^n \mapsto \{0,1\}^\tau$, a string of τ bit is generated. The function $\mathcal{F}$ is chosen as $\mathcal{F}(S) = (z_r, z_{r+1}, \ldots, z_{n-f-1})$ where $S \in \{0,1\}^n$ is the input to the stream cipher as an initial state or, $\mathcal{F}(S)$ is a slight modification of the $(z_r, z_{r+1}, \ldots, z_{n-f-1})$ such that it can be obtained by few operations. Slight modification is required to have different functions for different tables to reduce the number of collisions. This new string is the second entry of the row. This process is repeated t times to have a complete row. Further, all m rows are generated by choosing a random start entry of τ bits for each row to complete a table. Then, the same process is repeated for each table by selecting a different one-way function $\mathcal{F}$ to minimize the collision. Only two entries, i.e., the start entry and the end entry of each row, are stored, and intermediate entries are not stored. Using hash coding, the rows are stored in such a way that searching for the end entry can be performed in a constant number of operations. Hence, the pre-processing time and memory requirement are respectively,

$$P' = P = m \times t \times \frac{t}{D'} = \frac{mt^2}{D'}; \qquad M' = M = m \times \frac{t}{D'} = \frac{mt}{D'}. \tag{5.6}$$

5.4.1.2 Online phase

Consider a data of $n-f$ bits with initial r bits contains the pattern ζ. The remaining $\tau = n-f-r$ bits of the data (say, γ) is searched for a match among the end point entries (i.e., t-th entry) of the offline table. If a match is found, then the $(t-1)$-th entry of row possibly stores the state. This $(t-1)$-th entry can be obtained by performing the operation from the starting point entry of the row till $(t-1)$-th entry. Upon getting the $(t-1)$-th entry, the same row operation (as described in Section 5.4.1.1) is performed to check whether it is matching with the string γ. If it matches, then the state of the

cipher can be obtained. If there is no match, then the adversary performs the row operation on γ, and the search process is repeated till to get a match. Hence, the adversary may need to repeat the process t times for the whole row in the worst-case scenario.

Such a data of $n-f$ bits can be linearly searched from the known keystream of D bits. Upon a hit a pattern ζ of r bits, the following τ bits are collected as γ and the search process is performed. Further, once there is a match in the offline table, the recovered state containing the actual fixing pattern of f bits is having a probability of 2^{-f}. Hence, the requirement of data and time are, respectively

$$D = 2^{r+f} \cdot D'; \qquad T = 2^{f} \cdot T'. \tag{5.7}$$

5.4.2 Results based on our TMDTO curve equation and criteria

The TMDTO curve as per the literature is

$$T'M'^2D'^2 = N'^2, \quad T' \geq D'^2 \tag{5.8}$$

The parameters of the curve over full space are defined by (from Equations 5.6, 5.7)

$$D = 2^{f+r}D', \quad T = 2^{f}T', \quad M = M' \text{ and } P = P' = \frac{N'}{D'} = \frac{N}{D}. \tag{5.9}$$

Hence, the TMDTO curve in whole space is obtained from the Equations 5.8 and 5.9 is presented in the following theorem.

Theorem 8. *If r state bits can be recovered by fixing f state bits in a stream cipher having n-bits state then the TMDTO curve for the stream cipher is*

$$TM^2D^2 = 2^{f}N^2, \quad T \geq 2^{-f-2r}D^2, \quad PD = N \tag{5.10}$$

Observation 2. *We have the following observations from Theorem 8.*

1. *It is clear from the first equality in Equation 5.10 that the increase in the number of fixing bits (f) increases data, time, and memory complexities.*

2. *The number of recovering bits (r) has not a very significant role in the TMDTO curve except reducing the lower bound of the ratio $\frac{T}{D^2}$ (from the second inequality in Equation 5.10).*

3. *The third equality in Equation 5.10 notifies that the preprocessing complexity and data complexity are inversely proportional to each other. The reduction of one complexity increases the other.*

In general, one random access to a hard disk takes several million times longer in real-time than a simple computational step such as addition or multiplication in the current processor. The advantage of BSW sampling is reducing hard disk operations (i.e., matching in the table). In the conditional BSW-sampling attack, the disk access is reduced by 2^{f+r} factor, and this is a huge advantage for real-time consumption. The increase in the number of recovering state bits r reduces the complexity of disk access. Although the increase in the number of fixing state bits f reduces the complexity of disk access, it increases the data, time, and memory complexities. Further, the actual memory requirement for the offline table is too reduced by $\frac{n}{n-f-r}$ factor as the table stores the entries of size $\tau = n - f - r$ instead of entries of size n.

Therefore, one needs to be careful not to have a large number of fixing bits f to recover r state bits. Further, the availability of more data(D) reduces the preprocessing time(P), online time(T) and memory(M). Unfortunately, having a huge amount of data is not practical. The challenge is to find a lower bound of available data. Further, from Item 2 of Observation 2, we have that r plays the role to minimize the bound of $\frac{T}{D^2}$. We consider three different cases, i.e.,

1. when $D < T = M$ (to have low data);

2. when $T = 2^{-f-2r}D^2$ (to have low T by satisfying the lower bound for $\frac{T}{D^2}$).

3. when $D = T = M$ (to minimize the $\max\{T, M, D\}$ from [70]);

We have analyzed TMDTO over Lizard and Grain-128a as per the state bit recoveries, which are presented in Table 5.8 and 5.11 respectively.

5.4.3 TMDTO attack on Lizard

Case 1($D < T = M$): This case is considered keeping in mind that having a huge amount of data is impractical. Therefore, we want to minimize D. When $T = M$ we have $T^3D^2 = 2^fN^2$ from Equation 5.10. Here, we consider the following two sub cases to have smaller D than T and M.

1. **Sub case $D^2 = T$:** We have $D^8 = 2^fN^2$. That implies, $D = 2^{\frac{2n+f}{8}}$. Hence, $T = M = D^2 = 2^{\frac{2n+f}{4}}$, $T' = 2^{-f}T = 2^{\frac{2n-3f}{4}}$, $D' = 2^{-f-r}D = 2^{\frac{2n-7f-8r}{8}}$.

   We have presented two instances in r and f of Lizard in Table 5.12, where the parameters satisfy the condition $D^2 = T$ as whole numbers. Although the parameters M and T are more than 2^{60}, the data D is closer to the practical value.

   Table 5.12: The parameters for Lizard satisfying $D^2 = T = M$

r	f	N'	T'	D'	T	$M = M'$	D	$P = P'$
7	6	2^{108}	2^{56}	2^{18}	2^{62}	2^{62}	2^{31}	2^{90}
12	14	2^{95}	2^{50}	2^{6}	2^{64}	2^{64}	2^{32}	2^{89}

2. **Sub case $D'^2 = T'$:** Here $T = 2^fT' = 2^fD'^2 = 2^f(2^{-2f-2r}D^2) = 2^{-f-2r}D^2$. Putting the values of T in $T^3D^2 = 2^fN^2$, we have $2^{-3f-6r}D^8 = 2^{2n+f}$, i.e., $D = 2^{\frac{n+2f+3r}{4}}$. Hence, $M = T = 2^{-f-2r}D^2 = 2^{\frac{n-r}{2}}$, $D' = 2^{-f-r}D = 2^{\frac{n-2f-r}{4}}$. In Table 5.13[Instances 1-5], the parameters satisfy the condition $D'^2 = T'$ exactly as whole numbers.

In the case of other instances in Table 5.13, the exponents in parameters are in fractions to satisfy the condition. Therefore, the parameters presented for these instances are close to the condition $D'^2 = T'$ as whole numbers. In Table 5.13[Instance 7], the number of fixing bit is increased by one to get complexity in a whole number.

Table 5.13: The parameters for Lizard satisfying $D'^2 = T'$

Instance	r	f	N'	T'	D'	T	$M = M'$	D	$P = P'$
1	9	8	2^{104}	2^{48}	2^{24}	2^{56}	2^{56}	2^{41}	2^{80}
2	10	10	2^{101}	2^{46}	2^{23}	2^{56}	2^{55}	2^{43}	2^{78}
3	11	12	2^{98}	2^{42}	2^{21}	2^{54}	2^{56}	2^{44}	2^{77}
4	12	14	2^{95}	2^{40}	2^{20}	2^{54}	2^{55}	2^{46}	2^{75}
5	13	16	2^{92}	2^{38}	2^{19}	2^{54}	2^{54}	2^{48}	2^{73}
6	14	18	2^{89}	2^{36}	2^{18}	2^{54}	2^{53}	2^{50}	2^{71}
7	15	21	2^{85}	2^{32}	2^{16}	2^{53}	2^{53}	2^{52}	2^{69}
8	17	24	2^{80}	2^{26}	2^{13}	2^{50}	2^{54}	2^{54}	2^{67}

Case 2($T = 2^{-f-2r}D^2$): With this condition we want to reduce the time complexity by satisfying the second equality in Equation 5.10. Hence we will have the curve equation (by putting $T = 2^{-f-2r}D^2$ in the first equation in Equation 5.10) $MD^2 = 2^{f+r}N = 2^{f+r+n}$. Considering equal complexity for memory and data, i.e., $M = D$, we have $M = D = 2^{\frac{f+r+n}{3}}$. That implies, $T = 2^{\frac{-f-4r+2n}{3}}$. In this case, it is possible to reduce time complexity by considering higher values of r, but the memory and data complexities increase. We have presented few best possible instances by changing the pair r and f of Lizard in Table 5.14, where the parameters satisfy the condition $D = M$.

Case 3($D = T = M$): The case $D = T = M$ arises to minimize $\max(T, M, D)$. Here, $T = D$ implies that $T' = 2^r D'$ and $M = D$ implies that $M' = 2^{f+r}D'$. Substituting these values with $N' = 2^{n-f-r}$ in the TMDTO equation 5.8 we have $2^r D' \cdot 2^{2f+2r} D'^2 \cdot D'^2 =$

Table 5.14: The parameters for Lizard satisfying $T = 2^{-f-2r}D^2, D = M$

r	f	T	M	D	P
16	22	2^{52}	2^{53}	2^{53}	2^{68}
17	24	2^{50}	2^{54}	2^{54}	2^{67}
18	38	2^{44}	2^{59}	2^{59}	2^{62}
19	40	2^{42}	2^{60}	2^{60}	2^{61}

$2^{2n-2f-2r}$, i.e., $D' = 2^{\frac{2n-4f}{5}-r}$. This equation suggests that the value of D' needs to be decreased if f and r increases. Further the inequality $T' \geq D'^2$, i.e., $2^r D' \geq D'^2$, i.e., $2^r \geq D'$ implies that $r \geq \frac{n-2f}{5}$. Hence, this restriction on r is a necessary condition for happening $T = M = D$.

The instances presented in Table 5.15 satisfy the necessary bound $r \geq \frac{n-2f}{5}$ for $T = M = D$. The parameters in instance 1 in Table 5.15 are $n = 121, r = 15$ and $f = 23$ by adding extra five fixing bits (to satisfy the bound). Here we get $r = \frac{n-2f}{5} = 15$. Hence, $D' = 2^{\frac{2\cdot121-4\cdot23}{5}-15} = 2^{15}$, $T' = 2^{15} \cdot 2^{15} = 2^{30}$ and finally $T = M = D = 2^{23+15} \cdot 2^{15} = 2^{53}$. In other two instances in Table 5.15, we have parameters $n = 121, f = 23, 28$ and $r = 16, 17$ by adding extra one and three fixing bits respectively. Here we get $r > \frac{n-2f}{5}$. Hence, $D' = 2^{\frac{2\cdot121-4\cdot23}{5}-16} = 2^{14}$, $T' = 2^{16} \cdot 2^{14} = 2^{30}$ and so $T = M = D = 2^{53}$ for second instance. For the third instance, $D' = 2^{\frac{2\cdot121-4\cdot28}{5}-17} = 2^9$, $T' = 2^{17} \cdot 2^9 = 2^{26}$ and $T = M = D = 2^{54}$. In the last two instances, we added few extra fixing bits to have the exponent in T, M, D as whole numbers.

Table 5.15: The parameters for Lizard satisfying $D = T = M$

Instance	r	f	N'	T'	D'	T	$M = M'$	D	$P = P'$
1	15	23	2^{83}	2^{30}	2^{15}	2^{53}	2^{53}	2^{53}	2^{68}
2	16	23	2^{82}	2^{30}	2^{14}	2^{53}	2^{53}	2^{53}	2^{68}
3	17	28	2^{76}	2^{26}	2^{9}	2^{54}	2^{54}	2^{54}	2^{67}

We do not apply the TMDTO attack using the last five data reported in Table 5.9 (i.e., for $r = 20, 21, 22, 23, 24$) due to the large number of fixing bits for which the value of D becomes very high.

Table 5.16 presents a comparison with the previous TMDTO attack on Lizard by Maitra et al. [70]. We recover more bits by having much fewer fixing bits than the Maitra et al.'s attack. The fewer fixing bits help us to reduce the complexities. The TMDTO attack has four parameters to minimize and balance as per the available resource. We have considered different cases, and our attack complexities on different parameters supersede the complexities presented by Maitra et al. [70]. We have results from Case 1 to outperform at least in data complexity (D), from case 2 to outperform in $T = M = D$, and from case 3 to outperform at least in T. The outperforming complexities compared to Maitra et al. results are presented in bold text in Table 5.16. The pictorial presentation of trade-off curve by using our cases such that

Table 5.16: Comparison with previous result for Lizard

Citation	r	f	T	M	D	P	Case
Maitra et al. [70]	13	28	2^{54}	2^{54}	2^{54}	2^{67}	Case 3, i.e., $T = M = D$
Our attack	13	16	2^{54}	2^{54}	$\mathbf{2^{48}}$	2^{73}	Case 1, i.e., $D < T \approx M$
	14	18	2^{54}	$\mathbf{2^{53}}$	$\mathbf{2^{50}}$	2^{71}	
	15	21	$\mathbf{2^{53}}$	$\mathbf{2^{53}}$	$\mathbf{2^{52}}$	2^{69}	
	15	23	$\mathbf{2^{53}}$	$\mathbf{2^{53}}$	$\mathbf{2^{53}}$	2^{68}	Case 3, i.e., $T = M = D$
	16	22	$\mathbf{2^{52}}$	$\mathbf{2^{53}}$	$\mathbf{2^{53}}$	2^{68}	Case 2, i.e., $M = D > T$
	17	24	$\mathbf{2^{50}}$	2^{54}	2^{54}	2^{67}	
	18	38	$\mathbf{2^{44}}$	2^{59}	2^{59}	$\mathbf{2^{62}}$	
	19	40	$\mathbf{2^{42}}$	2^{60}	2^{60}	$\mathbf{2^{61}}$	

$2^{40} \leq T, M, D \leq 2^{60}$ is given in Figure 5.1.

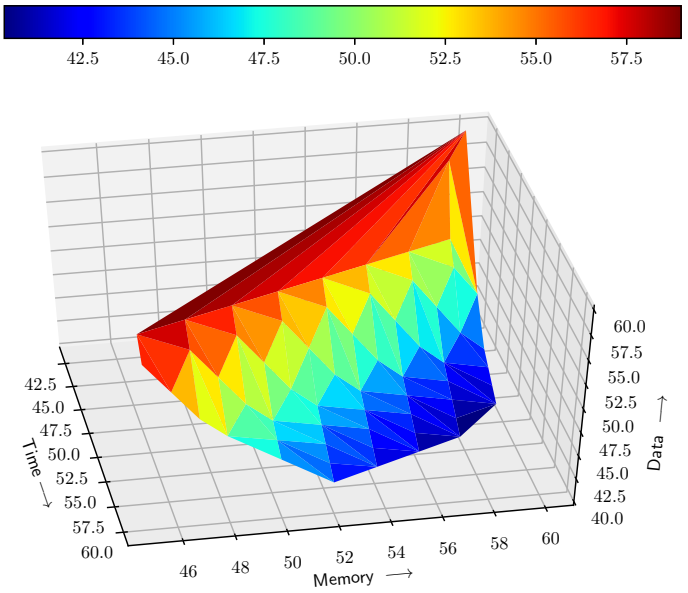

Figure 5.1: Pictorial presentation of TMDTO curve of Lizard (Parameters are in $\log_2(L)$ forms, $L = T, M, D$)

5.4.4 TMDTO attack on Grain-128a

We have implemented TMDTO attack on Grain-128a with state recovery parameters $n = 256, f = 54$ and $r = 48$. For two suitable choices of D', the attack complexities are summarized in Table 5.17. The previous attack (fast correlation attack) on Grain-128a is available in [62] having time and data complexities $2^{115.4}$ and $2^{113.8}$. Our conditional TMDTO attack has better complexity than it.

Table 5.17: TMDTO attack parameters on Grain-128a

Instances	r	f	N'	T'	D'	T	M	D	P
1	35	34	2^{187}	2^{76}	2^{38}	2^{110}	2^{111}	2^{107}	2^{149}
2	48	54	2^{154}	2^{60}	2^{11}	2^{114}	2^{113}	2^{113}	2^{143}

5.4.5 TMDTO Attack on Grain-v1

In this section, we have exploited the TMDTO attack technique to recover the state bits using the above results to recover state bits. Here we consider two TMDTO attacks, by Jiao et al. [68] and by Mihalijevic et al. [69] as the later TMDTO curve is the modified version of the TMDTO curve followed by Bjørstad [71] and Mihaljević et al. [75].

The TMDTO parameters by the attack by Jiao et al. [68] are as following.

- Data requirement (D) : 2^{f+r};
- Pre-processing time complexity (P) : 2^{n-f-r};
- Online time complexity (T): $t2^{f}$;
- Memory requirement: (M) : $m = \frac{2^{n-f-r}}{t}$;

where m is the number of rows in the state storing matrix and t is the number of columns of the matrix such that $m \times t = 2^{n-f-r}$.

Now we will fit our case $f = 45, r = 33$ and $n = 160$ with this curve as following.

- Data requirement (D) : $2^{45+33} = 2^{78}$;
- Pre-processing time complexity (P) : $2^{160-45-33^{*}} = 2^{82}$;
- Online time complexity (T): $t2^{45}$;
- Memory requirement: (M) : $m = \frac{2^{82}}{t}$;

If we take the number of columns in the stored matrix $t = 2^{16}$, then the required memory is $M = 2^{82-16} = 2^{66}$ and the online time complexity is $T = 2^{45+16} = 2^{61}$. The TMDTO curve is $TMD = 2^{61}.2^{66}.2^{78} = 2^{45}.2^{160} = 2^{s}N$. So, we have seen that the required data is reduced by half and the memory requirement is reduced by 2^{-5}, compared to their attack [67].

Then Mihaljević et al. [69] used the BS-TMDTO attack as follows. In this method, the number of fixing bits f and the number of recovering bits r reduce the total space $N = 2^n$ to $N' = 2^{n-f-r}$. The number of stored $m \times t$ table is t such that $mt^2 = N'$. The used BS trade-off curve is $T'M'^2D'^2 = N'^2$ such that $T' \geq D'^2$, where $T' = t^2$ and $M' = mt$. The probability of n bits having f fixing bits and r recovering bits is $p = 2^{-(f+r)}$ and the probability of occurring the given keystream with first f bits are fixed is $p' = 2^{-f}$. The total time complexity is $T = p'^{-1}T'$ and required total data is $D = p^{-1}D'$.

In our case, we have $f = 45, r = 33$ and then $N' = 2^{82}$. If we take $t = 2^{12}$, then $T' = 2^{24}$. Now by considering $D' = 2^0$, we need memory $M' = 2^{70}$ by TMDTO curve. The pre-processing complexity $P' = \frac{N'}{D'} = 2^{82}$. The probabilities $p = 2^{-78}$ and $p' = 2^{-45}$. Then total data is $D = p^{-1}D' = 2^{78}$ and the total time is $T = p'^{-1}T' = 2^{69}$.

For our case, the TMDTO curve followed by Jiao et al. [68] gives the best result. For the first time, in 2008, Bjørstad [71] mounted a TMDTO attack on Grain-v1, using BSW sampling to recover the state. Showing that Grain-v1 has low sampling resistance, he could reduce the time T and memory M by increasing data D. In 2012, Mihaljević et al. [75] used the normality of order two of the nonlinear functions of Grain-v1 for BSW sampling to recover 18 state bits where 18 consecutive key-stream bits are set as zero. It needed to fix 54 state bits and to guess 88 state bits and then BS TMDTO [40] with a single table lookup in pre-processing phase is used to recover the state bits. In 2015, Jiao et al. [68] recovered 28 state bits from 28 consecutive key-stream bits by using the normality order of the nonlinear function of Grain-v1. They fixed 51 state bits and guessed 81 state bits to reduce the sampling resistance from 2^{-18} to 2^{-28}. Then they used the TMDTO attack to recover state bits. The TMDTO curve was $TMD = 2^f N$. In 2017, Mihaljević et al. [69] used the same strategy to recover 24 and 31 state bits from the same number of consecutive key-stream bits in two different instants, respectively. It is needed to fix 6 and 31 state bits and guess 130 and 97 state bits, respectively. Then BS TMDTO is implemented over the

Table 5.18: Comparison of our result with previous results

References	Time (T)	Memory (M)	Keystream (D)	Pre-processing (P)
Bjørstad [71]	2^{70}	2^{69}	2^{56}	2^{104}
Mihaljević et al. [75]	2^{54}	2^{88}	2^{61}	2^{88}
Jiao et al. [68]	2^{61}	2^{71}	2^{79}	2^{81}
Mihaljević et al. [69]	2^{58} 2^{70}	2^{71} 2^{71}	2^{76} 2^{70}	2^{84} 2^{90}
Siddhanti et al. [76]	$2^{68.06}$	2^{64}	2^{64}	2^{96}
Our work	2^{61}	2^{66}	2^{78}	2^{82}

reduced space to recover state bits. The latest, this year, Siddhanti et al. [76] proposed TMDTO attack by recovering 32 state bits, fixing none state bits, and guessing 96 state bits with 36 known consecutive keystream bits. Now we compare our result with the previous results in the following Table 5.18.

5.5 Conclusion

In this chapter, we have used the classical method to recover the state bits of Grain-v1 by observing the algebraic structure of the cipher and then presented an algorithm to recover state bits of a cipher by fixing as small as possible number of state bits from some given keystream bits. Then we implemented the algorithm on Lizard and Grain-128a, where we recover state bits by fixing the minimum number of state bits as of now. Then we use the result in a conditional TMDTO attack by presenting a TMDTO curve using the number of recovering and fixing bits, given by Jiao et al. for Grain-v1 and given by us for Lizard and Grain-128a. We studied the attack on Lizard in three different cases $D < T = M$, $T < M = D$ and $D = T = M$. We supersede the previous best result with respect to the different parameters. As lightweight stream ciphers' state size is small compared to standard stream ciphers, the TMDTO attack is an exciting tool to analyze them. Currently, NIST has initiated standardizing

lightweight stream cipher. We hope that our algorithm can be used to analyze NIST second-round candidates under the TMDTO attack.

Chapter 6

Degree Evaluation of Grain-v1

6.1 Motivation

Most cryptographic primitives, specially NFSR based ciphers, consist of Boolean functions, which take private (i.e., key) and public (i.e., IV) bits as inputs. By exploiting the degree of the Boolean function, one can find out the weaknesses of the primitives. The correct estimation of the degree of an NFSR based stream cipher is a challenging job. There are few tools, such as statistical analysis, symbolic computation, etc., to estimate the algebraic degree. Some popular cryptanalysis techniques like cube attacks, integral attacks, algebraic attacks, higher-order differential attacks can be executed by estimating the algebraic degree of the Boolean function.

The theories of estimating degree are based on the two ideas, the use of the Walsh spectrum of Boolean function [78–80] and the use of the simple facts. Our work follows the latter. At CRYPTO 2017, Liu [81] has described an algorithm to find out the upper bounds on the degrees of NFSR based cipher by using a new concept, called "Numeric mapping". A degree evaluation technique for Trivium-like cipher was designed and shown that the estimated bound is close to its original value for maximum cases. They have used their degree estimation technique to identify the number of rounds where the key and IV bits are mixed properly or not, and further,

it is used for the distinguishing purpose as a cube tester. Our work is similar to this work. After that, Ye et al. [82] have presented an algorithm to find the exact super polynomial of a cube and proved that it is not a zero-sum distinguisher for 838 rounds of Trivium given by Liu. To find the super polynomial, they first compute the ANF of the output function by a backward method iteratively. The ANF of the involved bits in the output function z up to some manageable rounds are computed, and finally, the super polynomial is calculated.

At CRYPTO 2018, Fu et al. [83] considered the output function z as $P_1P_2 + P_3$, where P_1 should be selected by (a) frequency of P_1 is high in higher degree term (b) P_1 has low degree (c) minimum number of key guessing in P_1. The right key guessing of P_1 gives a simple polynomial as $(1+P_1)z = (1+P_1)P_3$ and wrong one gives $(1+P_1')z = (1+P_1')(P_1P_2+P_3)$. Finally they calculated the degree of $(1+P_1)z = (1+P_1)P_3$ as d by using their proposed algorithm and used d+1 dimensional cube as distinguisher. There is another literature [84], where degrees of NFSR based cryptosystem are discussed.

This chapter has simplified the h function of Grain-v1 by using some static variables and then calculated the maximum degree over initial state bits up to some round of Grain-v1. We aim to find out the round where the feedback bit and output can achieve the highest degree and claim that the key bits and IV bits mix properly at this round.

6.1.1 Our Contribution

We aim to evaluate the degree of NFSR and LFSR feedback bits and the output bit over the initial state bits (including the bits of key, IV, and nonce bits). This chapter has initiated work to evaluate the degree of the said bits using a difference of involved tap points in the NFSR and the output functions. The degree of NFSR update terms in Grain-v1 is higher than the degree of its output function. To control the degree of output function, we have put some conditions on IV bits. Therefore, our degree evaluation is subjected to some conditions on IV bits. We have followed the following

steps to evaluate the degree.

1. The differences between the state bits involved in the NFSR update function.
2. The common bits with shifted bits according to the differences.
3. Calculating the degree of the quadratic terms.
4. Common bits according to the difference trail.
5. The possibility of degrees of NFSR updated terms.

We could calculate the degrees of the NFSR, LFSR update bits, and the output bits up to 54 rounds, i.e., the degree of feedback bits b_{80+t}, l_{80+t} and output bit z_{80+t} for $0 \leq t \leq 54$. We have verified the degree up to 42nd round by SAGE.

6.1.2 Organization

The remaining parts of this chapter are divided into three sections. The main work is presented in Section 6.2, where we present all steps and tools for the degree evaluation. In Section 6.3, we apply the technique to evaluate the degrees of the functions in Grain-v1. In the last section, we conclude our work.

6.2 Degree Evaluation of Grain-v1

We see that during KSA, both the LFSR and NFSR feedback have a term z_t. Hence the degree of LFSR bits is dominated by the degree of z_t. As the NFSR has nonlinear terms, the degree of NFSR bits may not be dominated by the degree of z_t in all rounds. From Equation 2.5, it is clear that the highest degree term of z_t (i.e., 3-degree terms) comes from h. So we vanish those 3-degree terms of h by imposing some restrictions on IV bits for few initial rounds as following.

It can be seen from Equation 2.4 that s_{t+46} is present in all 3-degree terms (and one 2-degree term) in h. If the value of state bits s_{46} to s_{63} are equal to 0 (i.e., the

last 18 bits of IV), then all 3-degree terms of h are vanished for first 18 rounds. The equation of h for $0 \le t \le 17$ becomes

$$h(s_{t+3}, s_{t+25}, 0, s_{t+64}, b_{t+63}) = s_{t+25} + b_{t+63} + s_{t+3}s_{t+64} + s_{t+64}b_{t+63}.$$

As during the key loading the LFSR bits from s_{64} to s_{79} are set 1 as padding bits, h contains only two linear bits for first 16 rounds as the equation of h is

$$h(s_{t+3}, s_{t+25}, 0, 1, b_{t+63}) = s_{t+25} + s_{t+3} \text{ for } 0 \le t \le 15.$$

As round increases the conditions set (i.e., $s_{46} = \cdots = s_{63} = 0$ and $s_{64} = \cdots = s_{79} = 1$), the conditions shifted to other state bits in the LFSR. Therefore, the equation of h is simplified for higher rounds (i.e., $t \ge 16$). The equation of h is presented in Table 6.1 for first 77 rounds (i.e., $0 \le t \le 76$).

Grain-v1 is prone to be attacked by different way. There are several literature [35, 54, 67, 69, 71, 75, 76, 85] on this. Interested reader can go through this.

6.2.1 Calculating Repeated Bits

In Grain-v1, the set of state bits $B = \{b_9, b_{15}, b_{21}, b_{28}, b_{33}, b_{37}, b_{45}, b_{52}, b_{60}, b_{63}\}$ are involved in the non-linear terms of the state update relation of NFSR (see Equation 2.3). We enumerate the differences between each pair of these state bits. The differences can be computed using the following recursion.

Definition 36. *Let there is a ordered set of n integers $S = \{a_1, a_2, \cdots, a_n\}$. For $1 \le k \le n-1$, the kth order difference of S is defined as $\Delta^k S = \{a_{k+1} - a_1, a_{k+2} - a_2, \cdots, a_n - a_{n-k}\}$ and $\Delta^0 S = S$. The i-th element of $\Delta^k S$ (i.e., $a_{k+i} - a_i$) is denoted as $\Delta^k S_i$ for $1 \le i \le n-k$.*

It can be easily checked that $\Delta^k S$ can be recursively computed as the following.

Lemma 5. *For $2 \le k \le n-1$, $\Delta^k S_i = \Delta^{k-1} S_i + \Delta^1 S_{k+i-1}$ for $1 \le i \le n-k$.*

Table 6.1: ANF of h function of Grain-v1 at different rounds

Static Bit	h function	Rounds
$s_{t+46} = 0, s_{t+64} = 1$	$h = s_{t+3} + s_{t+25}$	0 – 15
$s_{t+46} = 0$	$h = s_{t+25} + b_{t+63} + s_{t+3}\mathbf{s}_{t+64} + \mathbf{s}_{t+64}b_{t+63}$	16
$s_{t+46} = 0$	$h = s_{t+25} + \mathbf{b}_{t+63} + s_{t+3}\mathbf{s}_{t+64} + \mathbf{s}_{t+64}\mathbf{b}_{t+63}$	17
$s_{t+46} = 1$	$h = s_{t+25} + \mathbf{b}_{t+63} + \mathbf{s}_{t+64} + s_{t+3}s_{t+25} +$ $s_{t+3}\mathbf{b}_{t+63} + s_{t+25}\mathbf{b}_{t+63}$	18 – 20
$s_{t+25} = 0, s_{t+46} = 1$	$h = \mathbf{b}_{t+63} + \mathbf{s}_{t+64} + s_{t+3}\mathbf{b}_{t+63}$	21 – 32
$s_{t+25} = 0$	$h = \mathbf{b}_{t+63} + s_{t+3}\mathbf{s}_{t+64} + \mathbf{s}_{t+46}\mathbf{s}_{t+64} + \mathbf{s}_{t+64}\mathbf{b}_{t+63} +$ $s_{t+3}\mathbf{s}_{t+46}s_{t+64} + s_{t+3}\mathbf{s}_{t+46}\mathbf{b}_{t+63} + \mathbf{s}_{t+46}\mathbf{s}_{t+64}\mathbf{b}_{t+63}$	33 – 38
$s_{t+25} = 1$	$h = 1 + \mathbf{b}_{t+63} + s_{t+3}\mathbf{s}_{t+64} + \mathbf{s}_{t+46}\mathbf{s}_{t+64} + \mathbf{s}_{t+64}\mathbf{b}_{t+63}$ $+s_{t+3}\mathbf{s}_{t+46} + s_{t+3}\mathbf{s}_{t+46}s_{t+64} + s_{t+3}\mathbf{s}_{t+46}\mathbf{b}_{t+63} +$ $\mathbf{s}_{t+46}\mathbf{b}_{t+63} + \mathbf{s}_{t+46}\mathbf{s}_{t+64}\mathbf{b}_{t+63}$	39 – 42
$s_{t+3} = 0,$ $s_{t+25} = 1$	$h = 1 + \mathbf{b}_{t+63} + \mathbf{s}_{t+46}\mathbf{s}_{t+64} + \mathbf{s}_{t+64}\mathbf{b}_{t+63} +$ $\mathbf{s}_{t+46}\mathbf{b}_{t+63} + \mathbf{s}_{t+46}\mathbf{s}_{t+64}\mathbf{b}_{t+63}$	43 – 54
$s_{t+3} = 0$	$h = \mathbf{s}_{t+25} + \mathbf{b}_{t+63} + \mathbf{s}_{t+46}\mathbf{s}_{t+64} + \mathbf{s}_{t+64}\mathbf{b}_{t+63} +$ $\mathbf{s}_{t+25}\mathbf{s}_{t+46}\mathbf{b}_{t+63} + \mathbf{s}_{t+46}\mathbf{s}_{t+64}\mathbf{b}_{t+63}$	54 – 60
$s_{t+3} = 1$	$h = \mathbf{s}_{t+25} + \mathbf{b}_{t+63} + \mathbf{s}_{t+64} + \mathbf{s}_{t+64}\mathbf{b}_{t+63} + \mathbf{s}_{t+25}\mathbf{s}_{t+46}$ $+\mathbf{s}_{t+46}\mathbf{b}_{t+63} + \mathbf{s}_{t+25}\mathbf{s}_{t+46}\mathbf{b}_{t+63} + \mathbf{s}_{t+46}\mathbf{s}_{t+64}\mathbf{b}_{t+63}$	61 – 76

For the Grain-v1 case, let take the set S as the index of the bits involved in the nonlinear terms of the state update relation of NFSR, i.e., in set B. Then Table 6.2 presents the order difference of the indices of these state bits where the differences and repeated indices can be identified. The value of the term $\Delta^k S_i$ represents that in Grain-v1 the state value of $(k+i)$-th nonlinear bit in ordered set S is shifted to i-th state after $\Delta^k S_i$ rounds. For example, $\Delta^2 S_3 = 12$ in Table 6.2 represents that the state value of the 33-th state (i.e., b_{33}) is shifted to 21-th state (i.e., b_{21}) in the NFSR after 12 rounds.

The sorted list of differences (from Table 6.2) and the pair of bits where the difference occurs are presented in the Table 6.3.

When two terms are multiplied, the final term's degree is lesser than the sum of the individual terms if there are some common variables between the terms. The following theorem presents the degree of the multiplication of two nonlinear terms in

Table 6.2: Order difference Δ^k of the indices

Bits	b_9	b_{15}	b_{21}	b_{28}	b_{33}	b_{37}	b_{45}	b_{52}	b_{60}	b_{63}
Order (k)										
0	9	15	21	28	33	37	45	52	60	63
1		6	6	7	5	4	8	7	8	3
2			12	13	12	9	12	15	15	11
3				19	18	16	17	19	23	18
4					24	22	24	24	27	26
5						28	30	31	32	30
6							36	37	39	35
7								43	45	42
8									51	48
9										54

Table 6.3: Table of Common bits according to differences

Difference	Between pair of bits	Difference	Between pair of bits	Difference	Between pair of bits
3	(b_{60}, b_{63})	16	(b_{21}, b_{37})	31	(b_{21}, b_{52})
4	(b_{33}, b_{37})	17	(b_{28}, b_{45})	32	(b_{28}, b_{60})
5	(b_{28}, b_{33})	18	$(b_{15}, b_{33}), (b_{45}, b_{63})$	35	(b_{28}, b_{63})
6	$(b_9, b_{15}), (b_{15}, b_{21})$	19	$(b_9, b_{28}), (b_{33}, b_{52})$	36	(b_9, b_{45})
7	$(b_{21}, b_{28}), (b_{45}, b_{52})$	23	(b_{37}, b_{60})	37	(b_{15}, b_{52})
8	$(b_{37}, b_{45}), (b_{52}, b_{60})$	22	(b_{15}, b_{37})	39	(b_{21}, b_{60})
9	(b_{28}, b_{37})	24	$(b_9, b_{33}), (b_{21}, b_{45})$, (b_{28}, b_{52})	42	(b_{21}, b_{63})
11	(b_{52}, b_{63})	26	(b_{37}, b_{63})	43	(b_9, b_{52})
12	$(b_9, b_{21}), (b_{21}, b_{33})$, (b_{33}, b_{45})	27	(b_{33}, b_{60})	45	(b_{15}, b_{60})
13	(b_{15}, b_{28})	28	(b_9, b_{37})	48	(b_{15}, b_{63})
15	$(b_{37}, b_{52}), (b_{45}, b_{60})$	30	$(b_{15}, b_{45}), (b_{33}, b_{63})$	51	(b_9, b_{60})
				54	(b_9, b_{63})

the case of feedback shift registers.

Theorem 9. *Let denote B_k and C_k be two nonlinear terms at the k-th round (i.e.,*

multiplication of some state bits in a feedback shift register at k -th round). Further denote that b_i is i-th state bit in k-th round. If b_i is a variable in B_k and b_{i-j} is a variable in C_k for some $0 \leq j \leq i$ then $\deg(B_k C_{k+j}) < \deg(B_k) + \deg(C_{k+j})$. *If there are exactly m such pairs of b_i and b_{i-j} in B_k and C_k respectively, then* $\deg(B_k C_{k+j}) = \deg(B_k) + \deg(C_{k+j}) - m$.

For an example, consider B_k and C_k as the highest degree term of the NFSR in Grain-v1, i.e., $B_k = C_k = b_{52}b_{45}b_{37}b_{33}b_{28}b_{21}$. Since b_{37} is in B_k and $b_{33} = b_{37-4}$ is in C_k, the $\deg(B_k C_{k+4}) = \deg(B_k)\deg(C_k) - 1 = 11$. For a demonstration, Table 6.4 represents the degrees of $B_k C_{k+j}, k \geq 80$ for those shifts j for the terms $B_k = C_k = b_{52}b_{45}b_{37}b_{33}b_{28}b_{21}$.

Table 6.4: Table of degrees of $B_k C_{k+j}, i \geq 80$

Shift(j)	$Deg(B_k C_{k+j})$	Bits in B_k, C_k	Shifts(j)	$Deg(B_k C_{k+j})$	Bits in B_k, C_k
4	11	b_{37}, b_{33}	15	11	b_{52}, b_{37}
5	11	b_{33}, b_{28}	16	11	b_{37}, b_{21}
7	10	$b_{28}, b_{21}; b_{52}, b_{45}$	17	11	b_{45}, b_{28}
8	11	b_{45}, b_{37}	19	11	b_{52}, b_{33}
9	11	b_{37}, b_{28}	24	10	$b_{45}, b_{21}; b_{52}, b_{28}$
12	10	$b_{33}, b_{21}; b_{45}, b_{33}$	31	11	b_{52}, b_{21}

Further, if there is a common bit in m terms, then the degree of the multiplication of m terms reduced by $m-1$ from the sum of the degrees of m terms because of the $m-1$ times of repetition of the bit. Since we are checking common terms between a pair of terms at a time, the repetition is subtracted $\binom{m}{2}$ times (instead of $m-1$) from the sum of degree as in Theorem 9. This can be settled by adding $\binom{m-1}{2}$ common repetition again for correct degree calculation.

Observation 3. *Let a bit b be present in m terms of a multiplication of n terms where $n \geq m$. The common bit b comes in $\binom{m}{2}$ pairs of terms, and each time the degree is subtracted by 1 as in Theorem 9. As a result, the actual degree of the multiplication is reduced by $\binom{m-1}{2}$, which is the number of repeated counting of the common bit b.*

Hence the number $\binom{m-1}{2}$ *needs to be added to the final degree for the correct degree calculation.*

By exploiting Observation 3, it is possible to make a list of trails of three differences (i.e., $i-j-(i+j)$) with respect to a set (say B) such that a bit is common in three terms. That is, if a bit b_{k+i+j}, b_{k+i} and b_k are present in the terms X_k, Y_k and Z_k respectively, then the terms X_k, Y_{k+j}, Z_{k+i+j} contains a common term b_{k+i+j}. As we aim to find out the maximum degree, we need to find the terms when the degree of their multiplication does not reduce. Hence, it will be easier to find out such possible terms looking from the list of trails. Table 6.5 presents the list of trails with respect to the set B (i.e., the bits involved in the nonlinear terms of NFSR of Grain-v1).

Table 6.5: Table of Repeated Common bit with difference Trail

Difference Trail	Common bit-Via bit-Shift bit	Difference Trail	Common bit-Via bit-Shift bit	Difference Trail	Common bit-Via bit-Shift bit
6 – 6 – 12	$b_9 - b_{15} - b_{21}$	27 – 3 – 30	$b_{33} - b_{60} - b_{63}$	6 – 18 – 24	$b_9 - b_{15} - b_{33}$
7 – 5 – 12	$b_{21} - b_{28} - b_{33}$	28 – 8 – 36	$b_9 - b_{37} - b_{45}$	6 – 16 – 22	$b_{15} - b_{21} - b_{37}$
4 – 8 – 12	$b_{33} - b_{37} - b_{45}$	30 – 7 – 37	$b_{15} - b_{45} - b_{52}$	7 – 17 – 24	$b_{21} - b_{28} - b_{45}$
7 – 8 – 15	$b_{45} - b_{52} - b_{60}$	31 – 8 – 39	$b_{21} - b_{52} - b_{60}$	19 – 5 – 22	$b_9 - b_{28} - b_{33}$
12 – 7 – 19	$b_9 - b_{21} - b_{28}$	32 – 3 – 35	$b_{28} - b_{60} - b_{63}$	4 – 23 – 27	$b_{33} - b_{37} - b_{60}$
13 – 5 – 18	$b_{15} - b_{28} - b_{33}$	36 – 7 – 43	$b_9 - b_{45} - b_{52}$	8 – 18 – 26	$b_{37} - b_{45} - b_{63}$
12 – 4 – 16	$b_{21} - b_{33} - b_{37}$	37 – 8 – 45	$b_{15} - b_{52} - b_{60}$	6 – 22 – 28	$b_9 - b_{15} - b_{37}$
9 – 8 – 17	$b_{28} - b_{37} - b_{45}$	39 – 3 – 42	$b_{21} - b_{60} - b_{63}$	6 – 24 – 30	$b_{15} - b_{21} - b_{45}$
12 – 7 – 19	$b_9 - b_{21} - b_{28}$	43 – 8 – 51	$b_9 - b_{52} - b_{60}$	7 – 24 – 31	$b_{21} - b_{28} - b_{52}$
15 – 8 – 23	$b_{37} - b_{52} - b_{60}$	45 – 3 – 48	$b_{15} - b_{60} - b_{63}$	5 – 27 – 32	$b_{28} - b_{33} - b_{60}$
15 – 3 – 18	$b_{45} - b_{60} - b_{63}$	6 – 7 – 13	$b_{15} - b_{21} - b_{28}$	4 – 26 – 30	$b_{33} - b_{37} - b_{63}$
19 – 5 – 24	$b_9 - b_{28} - b_{33}$	5 – 4 – 9	$b_{28} - b_{33} - b_{37}$	6 – 30 – 36	$b_9 - b_{15} - b_{45}$
18 – 4 – 22	$b_{15} - b_{33} - b_{37}$	8 – 7 – 15	$b_{37} - b_{45} - b_{52}$	6 – 31 – 37	$b_{15} - b_{21} - b_{52}$
16 – 8 – 24	$b_{21} - b_{37} - b_{45}$	8 – 3 – 11	$b_{52} - b_{60} - b_{63}$	7 – 32 – 39	$b_{21} - b_{28} - b_{60}$
17 – 7 – 24	$b_{28} - b_{45} - b_{52}$	6 – 13 – 19	$b_9 - b_{15} - b_{28}$	5 – 30 – 35	$b_{28} - b_{33} - b_{63}$
19 – 8 – 27	$b_{33} - b_{52} - b_{60}$	6 – 12 – 18	$b_{15} - b_{21} - b_{33}$	6 – 37 – 43	$b_9 - b_{15} - b_{52}$
23 – 3 – 26	$b_{37} - b_{60} - b_{63}$	7 – 9 – 16	$b_{21} - b_{28} - b_{37}$	6 – 39 – 45	$b_{15} - b_{21} - b_{60}$
24 – 4 – 28	$b_9 - b_{33} - b_{37}$	5 – 12 – 17	$b_{28} - b_{33} - b_{45}$	7 – 35 – 42	$b_{21} - b_{28} - b_{63}$
22 – 8 – 30	$b_{15} - b_{37} - b_{45}$	4 – 15 – 19	$b_{33} - b_{37} - b_{52}$	6 – 45 – 51	$b_9 - b_{15} - b_{60}$
24 – 7 – 31	$b_{21} - b_{45} - b_{52}$	8 – 15 – 23	$b_{37} - b_{45} - b_{60}$	6 – 42 – 48	$b_{15} - b_{21} - b_{63}$
24 – 8 – 32	$b_{28} - b_{52} - b_{60}$	7 – 11 – 18	$b_{45} - b_{52} - b_{63}$		

Example 7. *If the bits* b_{37}, b_{33} *and* b_{28} *are present in the terms* X_{28}, Y_{28} *and* Z_{28} *respectively, then there is difference trail* $5-4-9$ *between* b_{28} *and* b_{37}*. So the terms* $X_{28}, Y_{28+4}, Z_{28+5+4}$ *contains a common term* $b_{28+5+4} = b_{37}$.

The NFSR update function of Grain-v1 contains the terms of degree 6 and less. Expecting the higher degree terms contribute for the highest degree term in NFSR, we have taken terms of degree $6, 5$ or 4. The terms are $b_{21}b_{28}b_{33}b_{37}b_{45}b_{52}$, $b_{37}b_{45}b_{52}b_{60}b_{63}$, $b_9b_{15}b_{21}b_{28}b_{33}$, $b_9b_{28}b_{45}b_{63}$, $b_{33}b_{37}b_{52}b_{60}$ and $b_{15}b_{21}b_{60}b_{63}$. Hence, there are 36 possibilities of pair wise multiplications among them self. Each possibility will show the number of common bits and the differences. Based on those differences we can estimate that which pair of terms give the maximum degree of NFSR update bits. Table 6.6 contains all 36 possibilities, where one can find the number of common bits and the differences of bits between each pair. To explain the contents of the table, let consider the entry between the row containing the term $n_k = b_{15}b_{21}b_{60}b_{63}$ and the column containing the term $m_k = b_{33}b_{37}b_{52}b_{60}$. $b_{33} : 27$ implies that the multiplication $m_{k+27}n_k$ contains a common bit $b_{33+27} = b_{60}$. Similarly the multiplication $m_{k+30}n_k$ contains a common bit $b_{33+30} = b_{63}$ and other 5 cases.

In the following section, we present the algorithm to find the degree of NFSR function (b_{80+t}), LFSR function (s_{80+t}) and output function (z_t).

6.3 Calculating the Degree of Feedback and Output Bits

In the key scheduling phase, the output bit z_t is added with the NFSR feedback function. Here, the NFSR update function's degree is 6, and the degree of output function is 3. As the round increases, the degree of output bit z_t increases. If the degree of output bit z_t can be resisted to increase it (using the conditional equations in Table 6.1), then the degree of NFSR update bits are dominated by NFSR update functions for some more rounds. We present a correct degree estimation technique of NFSR update bit (b_{80+t}) and output bit (z_t) of Grain-like cipher in Algorithm 8.

Example 8. *We evaluate the degree of* $b_{117} = b_{80+37}$ *at round* $t = 37$.

Table 6.6: Table of differences (i) between terms m_{k+i} and n_k contain a common bit

$m_k \rightarrow$ $n_k \downarrow$	$b_{15}b_{21}$ $b_{60}b_{63}$	$b_{33}b_{37}$ $b_{52}b_{60}$	b_9b_{28} $b_{45}b_{63}$	b_9b_{15} $b_{21}b_{28}b_{33}$	$b_{37}b_{45}$ b_{52} $b_{60}b_{63}$	$b_{21}b_{28}b_{33}$ $b_{37}b_{45}b_{52}$
$b_{15}b_{21}$ $b_{60}b_{63}$	$b_{15}:6,45$ 48 $b_{21}:39,42$ $b_{60}:3$	$b_{33}:27,30$ $b_{37}:23,26$ $b_{52}:8,11$ $b_{60}:3$	$b_9:6,12,$ $51,54$ $b_{28}:32,35$ $b_{45}:15,18$	$b_9:6,12,51,54$ $b_{15}:6,45,48;\ b_{21}$ $:39,42;\ b_{28}:32$ $,35;\ b_{33}:27,30$	$b_{37}:23,26$ $b_{45}:15,18$ $b_{52}:8,11$ $b_{60}:3$	$b_{21}:39,42;\ b_{28}:$ $32,35;b_{33}:27,30$ $b_{37}:23,26;\ b_{45}:$ $15,18;\ b_{52}:8,11$
$b_{33}b_{37}$ $b_{52}b_{60}$	$b_{15}:18,22,$ $37,45$ $b_{21}:12,16,$ $31,39$	$b_{33}:4,19,$ 27 $b_{37}:15,23$ $b_{52}:8$	$b_9:24,28$ $43,51$ $b_{28}:5,9,$ $24,32$ $b_{45}:7,15$	$b_9:24,28,43,51$ $b_{15}:18,22,37,45$ $b_{21}:12,16,31,39$ $b_{28}:5,9,24,32$ $b_{33}:4,19,27$	$b_{37}:15,23$ $b_{45}:7,15$ $b_{52}:8$	$b_{21}:12,16,31,39$ $b_{28}:5,9,24,32$ $b_{33}:4,19,27$ $b_{37}:15,23$ $b_{45}:7,15;\ b_{52}:8$
b_9b_{28} $b_{45}b_{63}$	$b_{15}:13,30,$ 48 $b_{21}:7,24,$ $42;\ b_{60}:3$	$b_{33}:12,30$ $b_{37}:8,26$ $b_{52}:11$ $b_{60}:3$	$b_9:19,36,$ 54 $b_{28}:17,35$ $b_{45}:18$	$b_9:19,36,54;b_{15}:$ $13,30,48;\ b_{21}:7,$ $24,42;\ b_{28}:17,35$ $b_{33}:12,30$	$b_{37}:8,26$ $b_{45}:18$ $b_{52}:11$ $b_{60}:3$	$b_{21}:7,24,42;\ b_{28}$ $:17,35;\ b_{33}:12,$ $30;\ b_{37}:8,26$ $b_{45}:18;\ b_{52}:11$
b_9b_{15} b_{21} $b_{28}b_{33}$	$b_{15}:6,30,$ 18 $b_{21}:7,12$		$b_9:6,12,$ $19,24$ $b_{28}:5$	$b_9:6,12,19,24$ $b_{15}:6,30,18;\ b_{21}$ $:28,12;\ b_{28}:5$		$b_{21}:7,12$ $b_{28}:5$
$b_{37}b_{45}$ b_{52} $b_{60}b_{63}$	$b_{15}:22,30,$ $37,45,48$ $b_{21}:16,24,$ $31,39,42$ $b_{60}:3)$	$b_{33}:4,12,$ $19,27,30$ $b_{37}:8,15$ $23,26$ $b_{52}:8,11$ $b_{60}:3)$	$b_9:28,36,$ $43,51,54$ $b_{28}:9,17,$ $24,32,35$ $b_{45}:7,15,$ 18	$b_9:28,36,43,51,$ $54;b_{15}:22,30,37$ $,45,48;b_{21}:16,24$ $,31,39,42;b_{28}:9,$ $17,24,32,35;b_{33}:$ $4,12,19,27,30$	$b_{37}:8,15,$ $23,26$ $b_{45}:7,15,$ 18 $b_{52}:8,11$ $b_{60}:3$	$b_{21}:16,24,31,39,$ $42;b_{28}:9,17,24,$ $32,35;b_{33}:4,12,$ $19,27,30;b_{37}:8,$ $15,23,26;b_{45}:7,$ $15,18;b_{52}:8,11$
$b_{21}b_{28}$ $b_{33}b_{37}$ $b_{45}b_{52}$	$b_{15}:6,13,$ $18,22,30$ $,37$ $b_{21}:7,12,$ $16,24,31$	$b_{33}:4,12,$ 19 $b_{37}:8,15$	$b_9:12,19,$ $24,28,36$ 43 $b_{28}:5,9,$ $17,24$ $b_{45}:7$	$b_9:12,19,24,,28,$ $36,43;b_{15}:6,13,$ $18,22,30,37;b_{21}:$ $7,12,16,24,31$ $b_{28}:5,9,17,24$ $b_{33}:4,12,19$	$b_{37}:8,15$ $b_{45}:7$	$b_{21}:7,12,16,24$ $,31;b_{28}:5,9,17$ $24;b_{33}:4,12,19$ $b_{37}:8,15;b_{45}:7$

1. *There are six terms of* $\boldsymbol{b}_{117}$ *of degree* $4, 5$ *or* 6. *For the example we considered the term, say* $T_5 = b_{100}b_{97}b_{89}b_{82}b_{74}$ *which gives the maximum degree.*

2. *There are differences of the indices of bits (which are greater than or equal to* 80*) are* $3, 8, 7, 11, 15, 18$ *using Table 6.2. Further* b_{100} *contain some terms. For this example, we replace* b_{100} *by* $b_{83}b_{80}b_{72}b_{65}b_{57}$ *and for other high degree terms it can be done similar way.*

   The term can be rewritten as $T = b_{97}b_{89}b_{83}b_{82}b_{80}b_{74}b_{72}b_{65}b_{57}$. *Now we will find out the degree of* T. *Since last four bits are of degree one, for sake of simplicity, we consider* T *as* $b_{97}b_{89}b_{83}b_{82}b_{80}$. *Now we apply step 4 of the algorithm.*

3. *As per the step 4 in Algorithm 8, the rows of the difference table as Table 6.2*

Algorithm 8: Algorithm for max degree of NFSR bits of Grain-v1.

Input : Round t
Output: A highest degree term of b_{80+t}

List the high degree terms $T_1, T_2, \cdots, T_n$ in b_{80+t};
Set $DEG_i = 0, 1 \le i \le n$ and let $T_i = b_{t+i_1} b_{t+i_2} \cdots b_{t+i_k}$;
for *i from 1 to n* **do**
    Construct a difference table as Table 6.2 using the bits in T_i.;
    Find out differences of the pair from Table 6.3.;
    Find the terms involved in each bit b_{t+i_j} in T_i, which gives highest degree using Table 6.6. Let the terms are $Q_{t+i_1}, Q_{t+i_2}, \cdots, Q_{t+i_k}$;
    Set $DEG_i = Deg(Q_{t+i_1}) + \cdots + Deg(Q_{t+i_k}))$.;
    Find out pairwise common key bits according to the difference of the bits in the terms Q_{t+i_j} using Table 6.3 and count the number, say l.;
    Set $DEG_i = DEG_i - l$.;
    Count repeated common bits from Table 6.5, say m.;
    Set $DEG_i = DEG_i + m$.;
end
Set $DEG = MAX(DEG_i)$.;
return DEG.;

are $8, 6, 1, 2; 14, 7, 3; 15, 9; 17$.

4. *The differences of the pair of bits are found from Table 6.3:*
   $(b_9, b_{15}), (b_{21}, b_{28}), (b_{28}, b_{45}), (b_{37}, b_{52}), (b_{45}, b_{52}), (b_{45}, b_{60}), (b_{15}, b_{21}),$
   $(b_{28}, b_{37}), (b_{37}, b_{45}), (b_{45}, b_{52}), (b_{52}, b_{60}), (b_{60}, b_{63})$.

5. *Consider the pair* (b_9, b_{15}). *This pair indicates that* b_9 *is in* b_{89} *and* b_{15} *in* b_{83}, *which is a common bit of both. We write such information of all pairs as* $b_{89} : b_9$ *and* $b_{83} : (b_{15})$ *in step 1 in Table 6.7.*

6. *Now we select the suitable terms for each bit of* $b_{80}, b_{82}, b_{83}, b_{89}, b_{97}, b_{100}$ *step by step (Step 2 to Step 5 in Table 6.7 respectively). As example, for* $b_{80} : (b_{37}, b_{45}, b_{63})$, *we will try to find the term from Table 6.6 where* b_{37}, b_{45}, b_{63} *are not involved or are least involved. So we set* $b_{80} = b_{33} b_{28} b_{21} b_{15} b_9$ *and corresponding pairs of* b_{80} *with others are deleted. Here,* b_{28} *is removed from* b_{97}, b_{89} *and* b_{60} *is removed from* b_{83} *in step 2 in Table 6.7.*

7. *The last step of Table 6.7 performs step 7 to step 9 of Algorithm 8, where the degrees of all terms are added in step 7. Then we see from the step 8 that* b_{28} *is a common bit between* b_{89} *and* b_{82}*, so* $l = 1$*. Hence* $DEG(b_{117}) = Degb_{97} + Degb_{89} + Degb_{83} + Degb_{82} + Degb_{80} + Degb_{74} + Degb_{72} + Degb_{65} + Degb_{57} = 5 + 6 + 6 + 5 + 5 + 1 + 1 + 1 + 1 - 1 = 30$.

Table 6.7: Table of Calculating Degrees of NFSR bit b_{117}

Step 1:$b_{100} : b_{45}, b_{52}, b_{60}$. $b_{97} : b_{28}, b_{37}, b_{45}, b_{52}(b_{63})$. $b_{89} : b_9, b_{15}, b_{21}, b_{28}, b_{45}$. (b_{45}, b_{60}, b_{63}). $b_{83} : b_{60}(b_{15}, b_{21})$. $b_{82} : (b_{28}, b_{52}, b_{60}, b_{63})$. $b_{80} : (b_{37}, b_{45}, b_{63})$.	**Step 2**:$b_{100} : b_{45}, b_{52}, b_{60}$. $b_{97} : b_{37}, b_{45}, b_{52}(b_{63})$. $b_{89} : b_9, b_{15}, b_{21}, b_{45}$. (b_{45}, b_{60}, b_{63}). $b_{83} : (b_{15}, b_{21})$. $b_{82} : (b_{28}, b_{52}, b_{60}, b_{63})$. $b_{80} = b_{33}b_{28}b_{21}b_{15}b_9$.
Step 4:$b_{100} : b_{60}$. $b_{97} : \mathbf{b}_{37}, (b_{63})$. $b_{89} = b_{52}b_{45}b_{37}b_{33}b_{28}b_{21}(\mathbf{b}_{21})$. (b_{45}). $b_{83} :$. $b_{82} = b_{33}b_{28}b_{21}b_{15}b_9(\mathbf{b}_{28})$. $b_{80} = b_{33}b_{28}b_{21}b_{15}b_9$.	**Step 3**:$b_{100} : b_{60}$. $b_{97} : b_{37}, b_{52}(b_{63})$. $b_{89} : b_9, b_{15}, \mathbf{b}_{21}$. (b_{45}, b_{60}, b_{63}). $b_{83} : (b_{15}, b_{21})$. $b_{82} = b_{33}b_{28}b_{21}b_{15}b_9(\mathbf{b}_{28})$. $b_{80} = b_{33}b_{28}b_{21}b_{15}b_9$.
Step 5:$b_{100} :$. $b_{97} = b_{33}b_{28}b_{21}b_{15}b_9$. $b_{89} = b_{52}b_{45}b_{37}b_{33}b_{28}b_{21}(\mathbf{b}_{21})$. $b_{83} = b_{52}b_{45}b_{37}b_{33}b_{28}b_{21}$. $b_{82} = b_{33}b_{28}b_{21}b_{15}b_9(\mathbf{b}_{28})$. $b_{80} = b_{33}b_{28}b_{21}b_{15}b_9$.	$Deg(b_{117}) = Deg(b_{100}b_{97}b_{89}b_{82}) + 1$. $= Deg(b_{97}b_{89}b_{83}b_{82}b_{80}) + 3 + 1$. $= Deg(b_{97}b_{89}b_{83}b_{82}) + 5 + 3 + 1$. $= Deg(b_{97}b_{89}b_{83}) + 5 + 5 + 3 + 1$. $= Deg(b_{97}b_{83}) + 6 + 5 + 5 + 4$. $= 5 + 6 + 6 + 5 + 5 + 4 - 1 = 30$.

We used Algorithm 8 to calculate the degrees of the NFSR, LFSR update bits and the output bit up to some rounds. Table 6.8 presents the degrees of these bits. The degree of terms up to first 42 rounds in Table 6.8 are verified using the software SAGE.

Table 6.8: Table of Degrees of different non-linear functions

Rounds (i)	Degree of b_{80+i}	Degree of z_{80+i}	Degree of s_{80+i}	Rounds (i)	Degree of b_{80+i}	Degree of z_{80+i}	Degree of s_{80+i}
0 – 15	6	1	1	42	33	22	22
16	6	2	2	43	34	22	22
17 – 19	10	7	7	44	34	22	22
20 – 27	15	7	7	45	38	26	26
28 – 33	19	7	7	46	38	26	26
34	23	18	18	47	38	26	26
35	26	18	18	48	38	26	26
36	26	18	18	49	38	26	26
37	30	22	22	50	38	34	34
38	31	22	22	51	45	39	39
39	31	22	22	52	47	41	41
40	32	22	22	53	47	45	45
41	32	22	22	54	51	48	48

6.4 Conclusion

This chapter aims to develop a degree evaluation technique for feedback and output bits of the NFSR based stream cipher. Using our technique, we can calculate the degree of the said bits during the key scheduling algorithm of Grain-v1 of the reduced round.

Chapter 7

Conclusions

This dissertation presents the cryptanalysis of contemporary famous FSR-based stream ciphers. The first chapter presents the introduction and motivation towards the cryptanalysis of stream ciphers. Then, the required mathematical and cryptographical terms and definitions are presented in the next chapter. Moreover, the design of some stream ciphers and some essential cryptanalysis techniques are too presented in the same chapter. Finally, we have discussed our contributions in Chapter 3 to Chapter 6.

We explored conditional differential attack on Grain-v1 in Chapter 3. For the first time, certain distinguishers using two bits differences in the IV are proposed for higher initialization rounds of Grain-v1.

- The first distinguisher can distinguish Grain-v1 with 112 initialization rounds from a random source with a success rate of approximately 99%.
- The second distinguisher can distinguish Grain-v1 with 114 initialization rounds from a random source with a success rate of approximately 73% for 2^{78} weak keys, one-fourth of the whole keyspace.
- Further, this distinguisher has been extended to 116 initialization rounds with 1 bit difference in the key and 4 bit differences in the IV. Here the success rate is 62% for 2^{75} related keys.

Chapter 4 discusses the conditional cube attack on Grain-128a. This chapter introduces a new heuristic by combining maximum initial zero, maximum last zero, and maximum last α strategies for searching a suitable cube. This heuristic method also imposes conditions on state variables to find a suitable cube. A cube tester using the heuristic is designed with a small dimensional cube to distinguish the Grain-128a of 191 KSA rounds in the single key setup and 201 KSA rounds in the weak key setup. The distinguishing rounds are the highest round till now. Further, the attack is implemented on Grain-128.

We discuss two different state recovery attacks in Chapter 5. The first one uses the classical method to recover the state bits by observing the ANF of the output and non-linear update functions. The second one is a deterministic algorithm to recover a maximum number of state bits by fixing fewer state bits. We implement the algorithm on two famous FSR-based ciphers, Lizard and Grain-128a (without authentication mode). Also, we have used the classical method to recover the state bits of Grain-v1. We get the following results.

- Grain-v1: 33 state bits are recovered from same number of keystream bits by fixing 45 state bits.

- Lizard: $10, 11, 12, 13, 14, 15, 16, 17$ state bits are recovered from same number of keystream bits by fixing $9, 10, 11, 12, 14, 18, 22, 25$ state bits respectively.

- Grain-128a: $35, 48$ state bits are recovered from $35, 48$ keystream bits by fixing $34, 54$ state bits respectively, which is the first result on this cipher in this direction.

These recovering and fixing state bits from specific keystream bits are exploited to implement a conditional BSW-sampling TMDTO attack. Looking into the practicality, we analyze our results on three different conditions on TMDTO parameters T, M, D as follows.

1. $D < T = M$ which is considered by looking into the practicality of the availability of data D.

2. $T = 2^{-f-2r}D^2, M = D$, which satisfies the lower bound for T in terms of D.

3. $D = T = M$ which is to minimize $\max(D, T, M)$.

By using these criteria, we can recover the whole state bits of Lizard with the best complexities till now. Furthermore, our TMDTO attacks on Grain-v1 and Grain-128a provide better complexities than previously known results.

A degree evaluation method to compute the algebraic degrees of output and state update functions of Grain-v1 is discussed in Chapter 6.

There are many scopes to extend our works. Here a list of the proposal is given.

- The stream ciphers can be analyzed by applying conditional differential attack with a suitable difference vector of weight two or more. Therefore, automated handling of such scenarios will be of considerable interest.

- From the practical point of view of the cube attack, small dimensional cubes of any cipher are of great interest. Hence, the techniques for searching suitable small cubes for different stream ciphers can be explored.

- The approximations of the output function of any cipher can be utilized for a better state recovery attack. An improved TMDTO curve can also be designed for cryptanalysis purposes.

- A deterministic algorithm can be proposed to evaluate the algebraic degree of output and state update functions for as high as possible rounds. As a result, it could be possible to analyze the proper mixing of NFSR and LFSR bits of Grain-like cipher till that round. Further, one can find out the degrees of the IV bits (where key bits are taken as a constant) to mount a cube attack on a Grain-like cipher.

Appendix A

Algorithm to recover state bits from two different intervals of a cipher

Algorithm 9: Algorithm for adding two different intervals.

Input : Intervals (a, e) and (c, d).
Output: Set R and C of recovering bits and fixing bits respectively.

$P = \{\varnothing\}$ and $R = C = \varnothing$;
if $NL_g(c + l_2 - q, d) = \varnothing$ **then**
  $d^{'} = d$;
end
else
  $d^{'} = \min\{p : A_p \in NL_g(c + l_2 - q, d), p > c + t\} - 1$;
end
for $e^{'}$ *from* e *to* $a + 1$ **do**
  Calculate $NL_z(a, e^{'}) \cup UL_z(a, e^{'}) = \{p_1, \ldots, p_{|NL_z(a,e') \cup UL_z(a,e')|}\}$;
  for i *from 1 to* $|NL_z(a, e^{'}) \cup UL_z(a, e^{'})|$ **do**
    Compute $FS_z^{p_i} = \{T_1^{p_i}, \ldots, T_{|FS_z^{p_i}|}^{p_i}\}$;
    $t_i = fl_z^{p_i}(a, e^{'})$;
    for j *from* 1 *to* $|FS_z^{p_i}|$ **do**
      for $A_p \in T_j^{p_i}$ **do**
        if $(p + t_i) \geq l_1$ *or* $a \leq (p + t_i) \leq e^{'}$ *or* $c + t \leq (p + t_i) \leq d^{'}$ **then**
          Remove A_p from $T_j^{p_i}$
        end
      end
      if $T_j^{p_i} = \varnothing$ **then**
        goto 9;
      end
      Calculate $P = P \otimes (T_j^{p_i}, t_i)$;
    end
  end
  $R^{'} = \{A_a, A_{a+1}, \ldots, A_{e'}\}$;
  exit;
end

for *f from* $c+t$ *to* $d^{'}$ **do**
  $\overline{NL}_z(c+t,f) \cup UL_z(c,f) = \{p_1^*, \ldots, p^*_{|\overline{NL}_z(c+t,f) \cup UL_z(c,f)|}\}$;
  $P^{'} = \{\varnothing\}$;
  for *i from 1 to* $|\overline{NL}_z(c+t,f) \cup UL_z(c,f)|$ **do**
    Compute $\overline{FS}_z^{p_i^*} = \{T_1^{p_i^*}, \ldots, T^{p_i^*}_{|FS_z^{p_i^*}|}\}$;
    $t_i^* = fl_z^{p_i^*}(c+t,f)$;
    for *j from* 1 *to* $|\overline{FS}_z^{p_i^*}|$ **do**
      for $A_p \in T_j^{p_i^*}$ **do**
        if $(p+t_i^*) \geq l_2$ *or* $a \leq (p+t_i^*) \leq e^{'}+1$ *or* $c+t \leq (p+t_i^*) \leq d^{'}$ **then**
          Remove A_p from $T_j^{p_i^*}$
        end
      end
      if $T_j^{p_i^*} = \varnothing$ **then**
        exit;
      end
      Calculate $P^{'} = P \otimes P^{'} \otimes (T_j^{p_i^*}, t_i^*)$;
    end
  end
  $C = \{S \in P^{'} : |S| \text{ is minimum}\}$;
  $R = \{A_{c+t}, \ldots, A_f\} \cup R^{'}$;
end

Appendix B

The system of equations obtained to recover 17 bits of Lizard

The following system of equations (as Equation 5.2) is obtained to recover 17 bits of Lizard using the conditions in Table 5.8. Here,

- the set of fixing state bits $C = \{B_{34}, \ldots, B_{46}, B_{72}, \ldots, B_{82}\}$,
- the set of recovering state bits $R = \{S_{23}, \ldots, S_{30}, S_2, B_{63}, \ldots, B_{70}\}$,
- the set of guessing state bits
  $G = \{S_0, S_1, S_3 \ldots, S_{22}, B_0, \ldots, B_{33}, B_{47}, \ldots, B_{62}, B_{71}, B_{83}, \ldots, B_{89}\}$.

$$
\begin{aligned}
S_{23} &= z_0 + f_0(G) \\
S_{24} &= z_1 + f_1(G, S_{23}) \\
S_{25} &= z_2 + f_2(G, S_{23}, S_{24}) \\
S_{26} &= z_3 + f_3(G, S_{23}, S_{24}, S_{25}) \\
S_{27} &= z_4 + f_4(G, S_{23}, S_{24}, S_{25}, S_{26}) \\
S_{28} &= z_5 + f_5(G, S_{23}, \ldots, S_{26}, S_{27}) \\
S_{29} &= z_6 + f_6(G, S_{23}, \ldots, S_{27}, S_{28}) \\
S_{30} &= z_7 + f_7(G, S_{23}, \ldots, S_{28}, S_{29}) \\
S_2 &= z_8 + f_8(G, S_{23}, \ldots, S_{29}, S_{30}) \\
B_{63} &= z_9 + f_9(G, S_{23}, \ldots, S_{30}, S_2) \\
B_{64} &= z_{10} + f_{10}(G, S_{23}, \ldots, S_{30}, S_2, B_{63}) \\
B_{65} &= z_{11} + f_{11}(G, S_{23}, \ldots, S_{30}, S_2, B_{63}, B_{64}) \\
B_{66} &= z_{12} + f_{12}(G, S_{23}, \ldots, S_{30}, S_2, B_{63}, B_{64}, B_{65}) \\
B_{67} &= z_{13} + f_{13}(G, S_{23}, \ldots, S_{30}, S_2, B_{63}, \ldots, B_{65}, B_{66}) \\
B_{68} &= z_{14} + f_{14}(G, S_{23}, \ldots, S_{30}, S_2, B_{63}, \ldots, B_{66}, B_{67}) \\
B_{69} &= z_{15} + f_{15}(G, S_{23}, \ldots, S_{30}, S_2, B_{63}, \ldots, B_{67}, B_{68}) \\
B_{70} &= z_{16} + f_{16}(G, S_{23}, \ldots, S_{30}, S_2, B_{63}, \ldots, B_{68}, B_{69}).
\end{aligned}
$$

As per the Equation 13, we have the lower triangular matrix N is the zero matrix and $F(G,R) = (f_0, f_1, \ldots, f_{16})^T$ where

- $f_0(G) = B_5 + B_7 + B_{11} + B_{30} + B_{54} + B_{71} + B_4 * B_{21} + B_6 * B_{14} * B_{26} * B_{32} * B_{47} * B_{61} + B_9 * B_{52} + B_{48} * S_9 * S_{13} + S_3 * S_{16}$;
- $f_1(G, S_{23}) = B_6 + B_8 + B_{12} + B_{31} + B_{55} + B_5 * B_{22} + B_7 * B_{15} * B_{27} * B_{33} * B_{48} * B_{62} + B_9 * B_{83} + B_{10} * B_{53} + B_{49} * S_{10} * S_{14} + S_4 * S_{17}$;
- $f_2(G, S_{23}, S_{24}) = S_{25} = z_2 + B_7 + B_9 + B_{13} + B_{32} + B_{47} + B_{56} + B_6 * B_{23} + B_{10} * B_{84} + B_{11} * B_{54} + B_{50} * S_{11} * S_{15} + S_5 * S_{18}$;
- $f_3(G, S_{23}, \ldots, S_{25}) = B_8 + B_{10} + B_{14} + B_{33} + B_{48} + B_{57} + B_7 * B_{24} + B_{11} * B_{85} + B_{12} * B_{55} + B_{51} * S_{12} * S_{16} + S_6 * S_{19}$;
- $f_4(G, S_{23}, \ldots, S_{26}) = B_9 + B_{11} + B_{15} + B_{49} + B_{58} + B_8 * B_{25} + B_{12} * B_{86} + B_{13} * B_{56} + B_{52} * S_{13} * S_{17} + S_7 * S_{20}$;
- $f_5(G, S_{23}, \ldots, S_{27}) = B_{10} + B_{12} + B_{16} + B_{50} + B_{59} + B_6 * B_{24} * B_{32} * B_{48} * B_{62} * B_{71} * B_{83} + B_9 * B_{26} + B_{13} * B_{87} + B_{14} * B_{57} + B_{53} * S_{14} * S_{18} + S_8 * S_{21}$;
- $f_6(G, S_{23}, \ldots, S_{28}) = B_{11} + B_{13} + B_{17} + B_{51} + B_{60} + B_{10} * B_{27} + B_{14} * B_{88} + B_{15} * B_{58} + B_{54} * S_{15} * S_{19} + S_9 * S_{22}$;
- $f_7(G, S_{23}, \ldots, S_{29}) = B_{12} + B_{14} + B_{18} + B_{47} + B_{52} + B_{11} * B_{28} + B_{15} * B_{89} + B_{16} * B_{59} + B_{51} * B_{83} + B_{55} * S_{16} * S_{20} + B_{61} + S_{10} * S_{23}$;
- $f_8(G, S_{23}, \ldots, S_{30}) = B_0 * B_{16} + B_3 * B_{16} * B_{59} + B_{10} * B_{12} * B_{16} + B_{12} * B_{29} + B_{13} + B_{15} * B_{16} + B_{15} + B_{16} * B_{20} * B_{22} * B_{23} + B_{16} * B_{24} + B_{16} * B_{25} * B_{53} + B_{16} * B_{49} + B_{16} * B_{55} * B_{58} + B_{16} * B_{84} + B_{16} * S_0 + B_{17} * B_{60} + B_{19} + B_{48} + B_{52} * B_{84} + B_{53} + B_{56} * S_{17} * S_{21} + B_{62} + S_0 + S_4 * S_7 * S_{12} * S_{21} + S_4 * S_7 * S_{19} * S_{21} + S_4 * S_{12} * S_{21} * S_{22} + S_4 * S_{12} * S_{22} + S_4 * S_{19} * S_{21} * S_{22} + S_4 * S_{19} * S_{22} + S_5 + S_6 + S_7 * S_8 * S_{18} * S_{21} + S_7 * S_8 * S_{20} * S_{21} + S_7 * S_{12} * S_{19} * S_{21} + S_7 * S_{20} * S_{21} + S_8 * S_{18} *$

$S_{21} * S_{22} + S_8 * S_{18} * S_{22} + S_8 * S_{18} + S_8 * S_{20} * S_{21} * S_{22} + S_8 * S_{20} + S_{12} * S_{19} * S_{21} * S_{22} + S_{12} * S_{21} + S_{14} * S_{19} + S_{15} + S_{17} * S_{21} + S_{17} + S_{18} + S_{20} * S_{21} * S_{22} + S_{20} * S_{22} + S_{20} + S_{25} + S_{11} * S_{24}$;

- $f_9(G, S_{23}, \ldots, S_{30}, S_2) = B_1 * B_{17} + B_4 * B_{17} * B_{60} + B_{11} * B_{13} * B_{17} + B_{13} * B_{30} + B_{14} + B_{16} * B_{17} + B_{16} + B_{17} * B_{21} * B_{23} * B_{24} + B_{17} * B_{25} + B_{17} * B_{26} * B_{54} + B_{17} * B_{50} + B_{17} * B_{56} * B_{59} + B_{17} * B_{85} + B_{17} * S_1 + B_{18} * B_{61} + B_{20} + B_{49} + B_{53} * B_{85} + B_{54} + B_{57} * S_{18} * S_{22} + S_1 + S_3 + S_5 * S_8 * S_{13} * S_{22} + S_5 * S_8 * S_{20} * S_{22} + S_6 + S_7 + S_8 * S_9 * S_{19} * S_{22} + S_8 * S_9 * S_{21} * S_{22} + S_8 * S_{13} * S_{20} * S_{22} + S_8 * S_{21} * S_{22} + S_9 * S_{19} + S_9 * S_{21} + S_{13} * S_{22} + S_{15} * S_{20} + S_{16} + S_{18} * S_{22} + S_{18} + S_{19} + S_{21} + S_5 * S_{13} * S_{22} * S_{23} + S_5 * S_{13} * S_{23} + S_5 * S_{20} * S_{22} * S_{23} + S_5 * S_{20} * S_{23} + S_9 * S_{19} * S_{22} * S_{23} + S_9 * S_{19} * S_{23} + S_9 * S_{21} * S_{22} * S_{23} + S_{13} * S_{20} * S_{22} * S_{23} + S_{21} * S_{22} * S_{23} + S_{21} * S_{23} + S_{12} * S_{25} + S_{26}$;

- $f_{10}(G, S_{23}, \ldots, S_{30}, S_2, B_{63}) = B_2 * B_{18} + B_5 * B_{18} * B_{61} + B_{12} * B_{14} * B_{18} + B_{14} * B_{31} + B_{15} + B_{17} * B_{18} + B_{17} + B_{18} * B_{22} * B_{24} * B_{25} + B_{18} * B_{26} + B_{18} * B_{27} * B_{55} + B_{18} * B_{51} + B_{18} * B_{57} * B_{60} + B_{18} * B_{86} + B_{19} * B_{62} + B_{21} + B_{28} * B_{47} + B_{50} + B_{54} * B_{86} + B_{55} + S_4 + S_7 + S_8 + S_{10} * S_{20} + S_{10} * S_{22} + S_{16} * S_{21} + S_{17} + S_{19} + S_{20} + S_{22} + B_{58} * S_{19} * S_{23} + S_6 * S_9 * S_{14} * S_{23} + S_6 * S_9 * S_{21} * S_{23} + S_6 * S_{14} * S_{23} * S_{24} + S_6 * S_{21} * S_{23} * S_{24} + S_9 * S_{10} * S_{20} * S_{23} + S_9 * S_{10} * S_{22} * S_{23} + S_9 * S_{14} * S_{21} * S_{23} + S_9 * S_{22} * S_{23} + S_{10} * S_{20} * S_{23} * S_{24} + S_{10} * S_{22} * S_{23} * S_{24} + S_{14} * S_{21} * S_{23} * S_{24} + S_{14} * S_{23} + S_{19} * S_{23} + S_{22} * S_{23} * S_{24} + S_6 * S_{14} * S_{24} + S_6 * S_{21} * S_{24} + S_{10} * S_{20} * S_{24} + S_{22} * S_{24} + S_{13} * S_{26} + S_{27} + B_{18} * S_2 + S_2$;

- $f_{11}(G, S_{23}, \ldots, S_{30}, S_2, B_{63}, B_{64}) = B_3 * B_{19} + B_6 * B_{19} * B_{62} + B_{13} * B_{15} * B_{19} + B_{15} * B_{32} + B_{16} + B_{18} * B_{19} + B_{18} + B_{19} * B_{23} * B_{25} * B_{26} + B_{19} * B_{27} + B_{19} * B_{28} * B_{56} + B_{19} * B_{52} + B_{19} * B_{58} * B_{61} + B_{19} * B_{87} + B_{19} * S_3 + B_{22} + B_{29} * B_{48} + B_{51} + B_{55} * B_{87} + B_{56} + S_3 + S_5 + S_8 + S_9 + S_{11} * S_{21} + S_{17} * S_{22} + S_{18} + S_{20} + S_{21} + S_{10} * S_{11} * S_{23} * S_{24} + S_{10} * S_{23} * S_{24} + S_{11} * S_{23} * S_{24} * S_{25} + S_{11} * S_{23} + S_{23} * S_{24} * S_{25} + B_{59} * S_{20} * S_{24} + S_{23} * S_{25} + S_{23} + S_7 * S_{10} * S_{15} * S_{24} + S_7 * S_{10} * S_{22} * S_{24} + S_{10} * S_{11} * S_{21} * S_{24} + S_{10} * S_{15} * S_{22} * S_{24} + S_{11} * S_{21} * S_{24} * S_{25} + S_{15} * S_{24} + S_{15} * S_{22} * S_{24} * S_{25} + S_{20} * S_{24} + S_7 * S_{15} * S_{24} * S_{25} + S_7 * S_{15} * S_{25} + S_7 * S_{22} * S_{24} * S_{25} + S_7 * S_{22} * S_{25} + S_{11} * S_{21} * S_{25} + S_{14} * S_{27} + S_{28} + B_{20} * B_{63}$;

- $f_{12}(G, S_{23}, \ldots, S_{30}, S_2, B_{63}, \ldots, B_{65}) = B_4 * B_{20} + B_{14} * B_{16} * B_{20} + B_{16} * B_{33} + B_{17} + B_{19} * B_{20} + B_{19} + B_{20} * B_{24} * B_{26} * B_{27} + B_{20} * B_{28} + B_{20} * B_{29} * B_{57} + B_{20} * B_{53} + B_{20} * B_{59} *$

$B_{62} + B_{20} * B_{83} + B_{20} * B_{88} + B_{20} * S_4 + B_{23} + B_{30} * B_{49} + B_{52} + B_{56} * B_{88} + B_{57} + B_{83} + S_4 + S_6 + S_9 + S_{10} + S_{12} * S_{22} + S_{19} + S_{21} + S_{22} + S_8 * S_{11} * S_{23} * S_{25} + S_8 * S_{23} * S_{25} * S_{26} + S_8 * S_{23} * S_{26} + S_{16} * S_{23} * S_{25} * S_{26} + S_{11} * S_{16} * S_{23} * S_{25} + S_{18} * S_{23} + S_{11} * S_{12} * S_{24} * S_{25} + S_{11} * S_{24} * S_{25} + S_{12} * S_{24} + S_{12} * S_{24} * S_{25} * S_{26} + S_{24} * S_{26} + S_{24} + B_{60} * S_{21} * S_{25} + S_8 * S_{11} * S_{16} * S_{25} + S_8 * S_{16} * S_{25} * S_{26} + S_{11} * S_{12} * S_{22} * S_{25} + S_{12} * S_{22} * S_{25} * S_{26} + S_{16} * S_{25} + S_{21} * S_{25} + S_{24} * S_{25} * S_{26} + S_8 * S_{16} * S_{26} + S_{12} * S_{22} * S_{26} + S_{15} * S_{28} + S_{29} + B_7 * B_{20} * B_{63} + B_{21} * B_{64};$

- $f_{13}(G, S_{23}, \ldots, S_{30}, S_2, B_{63}, \ldots, B_{66}) = B_5 * B_{21} + B_{15} * B_{17} * B_{21} + B_{18} + B_{20} * B_{21} + B_{20} + B_{21} * B_{25} * B_{27} * B_{28} + B_{21} * B_{29} + B_{21} * B_{30} * B_{58} + B_{21} * B_{54} + B_{21} * B_{84} + B_{21} * B_{89} + B_{21} * S_5 + B_{24} + B_{31} * B_{50} + B_{53} + B_{57} * B_{89} + B_{58} + B_{84} + S_5 + S_7 + S_{10} + S_{11} + S_{20} + S_{22} + S_{12} * S_{13} * S_{23} * S_{26} + S_{13} * S_{23} * S_{26} * S_{27} + S_{13} * S_{23} * S_{27} + S_{13} * S_{23} + S_{23} + S_9 * S_{12} * S_{24} * S_{26} + S_9 * S_{24} * S_{26} * S_{27} + S_9 * S_{24} * S_{27} + S_{17} * S_{24} * S_{26} * S_{27} + S_{19} * S_{24} + S_{12} * S_{17} * S_{24} * S_{26} + S_{12} * S_{13} * S_{25} * S_{26} + S_{12} * S_{25} * S_{26} + S_{13} * S_{25} * S_{26} * S_{27} + S_{13} * S_{25} + S_{25} * S_{27} + S_{25} * S_{26} * S_{27} + S_{25} + S_9 * S_{12} * S_{17} * S_{26} + S_9 * S_{17} * S_{26} * S_{27} + S_9 * S_{17} * S_{27} + S_{17} * S_{26} + S_{22} * S_{26} + B_{61} * S_{22} * S_{26} + S_{16} * S_{29} + S_{30} + B_{21} * B_{60} * B_{63} + B_8 * B_{21} * B_{64} + B_{22} * B_{65};$

- $f_{14}(G, S_{23}, \ldots, S_{30}, S_2, B_{63}, \ldots, B_{67}) = B_0 * B_{22} + B_0 * B_{58} + B_3 * B_{22} * B_{59} + B_3 * B_{58} * B_{59} + B_6 * B_{22} + B_{10} * B_{12} * B_{22} + B_{10} * B_{12} * B_{58} + B_{15} * B_{16} * B_{22} + B_{15} * B_{16} * B_{58} + B_{16} * B_{18} * B_{22} + B_{19} + B_{20} * B_{22} * B_{23} * B_{58} + B_{20} * B_{22} * B_{23} + B_{21} * B_{22} + B_{21} + B_{22} * B_{24} + B_{22} * B_{25} * B_{53} + B_{22} * B_{26} * B_{28} * B_{29} + B_{22} * B_{30} + B_{22} * B_{31} * B_{59} + B_{22} * B_{49} + B_{22} * B_{55} * B_{58} + B_{22} * B_{55} + B_{22} * B_{83} * B_{86} * B_{87} * B_{89} + B_{22} * B_{84} + B_{22} * B_{85} + B_{22} * S_0 + B_{22} * S_6 + B_{24} * B_{58} + B_{25} * B_{53} * B_{58} + B_{25} + B_{32} * B_{51} + B_{49} * B_{58} + B_{54} + B_{55} * B_{58} + B_{58} * B_{84} + B_{58} * S_0 + B_{59} + B_{85} + S_0 + S_4 * S_7 * S_{12} * S_{21} + S_4 * S_7 * S_{19} * S_{21} + S_4 * S_{12} * S_{21} * S_{22} + S_4 * S_{12} * S_{22} + S_4 * S_{19} * S_{21} * S_{22} + S_4 * S_{19} * S_{22} + S_5 + S_7 * S_8 * S_{18} * S_{21} + S_7 * S_8 * S_{20} * S_{21} + S_7 * S_{12} * S_{19} * S_{21} + S_7 * S_{20} * S_{21} + S_8 * S_{18} * S_{21} * S_{22} + S_8 * S_{18} * S_{22} + S_8 * S_{18} + S_8 * S_{20} * S_{21} * S_{22} + S_8 * S_{20} + S_8 + S_{11} + S_{12} * S_{19} * S_{21} * S_{22} + S_{12} * S_{21} + S_{12} + S_{14} * S_{19} + S_{15} + S_{17} * S_{21} + S_{17} + S_{18} + S_{20} * S_{21} * S_{22} + S_{20} * S_{22} + S_{20} + S_{21} + B_{62} * S_{23} * S_{27} + S_{23} * S_{27} + S_{23} + S_{13} * S_{14} * S_{24} * S_{27} + S_{14} * S_{24} * S_{27} * S_{28} + S_{14} * S_{24} * S_{28} + S_{14} * S_{24} + S_{24} + S_{10} * S_{25} * S_{27} * S_{28} + S_{10} * S_{25} * S_{28} + S_{13} * S_{18} * S_{25} * S_{27} + S_{18} * S_{25} * S_{27} * S_{28} + S_{20} * S_{25} + S_{25} + S_{13} * S_{14} * S_{26} * S_{27} + S_{13} * S_{26} * S_{27} + S_{14} * S_{26} * S_{27} * S_{28} + S_{14} * S_{26} + S_{26} * S_{27} * S_{28} + S_{26} * S_{28} + S_{26} + S_{10} * S_{13} * S_{18} * S_{27} + S_{10} * S_{13} * S_{25} * S_{27} + S_{10} * S_{18} * S_{27} *$

$S_{28}+S_{18}*S_{27}+S_{10}*S_{18}*S_{28}+S_{17}*S_{30}+S_2+B_{22}*B_{61}*B_{64}+B_9*B_{22}*B_{65}+B_{23}*B_{66}$;

- $f_{15}(G,S_{23},\ldots,S_{30},S_2,B_{63},\ldots,B_{68}) = B_0*B_{23}*B_{84}*B_{87}*B_{88}+B_1*B_{23}+B_1*B_{59}+B_3*B_{23}*B_{59}*B_{84}*B_{87}*B_{88}+B_4*B_{23}*B_{60}+B_4*B_{59}*B_{60}+B_7*B_{23}+B_{10}*B_{12}*B_{23}*B_{84}*B_{87}*B_{88}+B_{11}*B_{13}*B_{23}+B_{11}*B_{13}*B_{59}+B_{15}*B_{16}*B_{23}*B_{84}*B_{87}*B_{88}+B_{16}*B_{17}*B_{23}+B_{16}*B_{17}*B_{59}+B_{17}*B_{19}*B_{23}+B_{20}*B_{22}*B_{23}*B_{84}*B_{87}*B_{88}+B_{20}+B_{21}*B_{23}*B_{24}*B_{59}+B_{21}*B_{23}*B_{24}+B_{22}*B_{23}+B_{22}+B_{23}*B_{24}*B_{84}*B_{87}*B_{88}+B_{23}*B_{25}*B_{53}*B_{84}*B_{87}*B_{88}+B_{23}*B_{25}+B_{23}*B_{26}*B_{54}+B_{23}*B_{27}*B_{29}*B_{30}+B_{23}*B_{31}+B_{23}*B_{32}*B_{60}+B_{23}*B_{49}*B_{84}*B_{87}*B_{88}+B_{23}*B_{50}+B_{23}*B_{55}*B_{58}*B_{84}*B_{87}*B_{88}+B_{23}*B_{56}*B_{59}+B_{23}*B_{56}+B_{23}*B_{84}*B_{87}*B_{88}*S_0+B_{23}*B_{84}*B_{87}*B_{88}+B_{23}*B_{85}+B_{23}*B_{86}+B_{23}*S_1+B_{23}*S_7+B_{25}*B_{59}+B_{26}*B_{54}*B_{59}+B_{26}+B_{33}*B_{52}+B_{50}*B_{59}+B_{55}+B_{56}*B_{59}+B_{59}*B_{85}+B_{59}*S_1+B_{60}+B_{86}+S_0*S_{18}+S_1+S_3+S_4*S_7*S_{12}*S_{18}*S_{21}+S_4*S_7*S_{18}*S_{19}*S_{21}+S_4*S_{12}*S_{18}*S_{21}*S_{22}+S_4*S_{12}*S_{18}*S_{22}+S_4*S_{18}*S_{19}*S_{21}*S_{22}+S_4*S_{18}*S_{19}*S_{22}+S_5*S_8*S_{13}*S_{22}+S_5*S_8*S_{20}*S_{22}+S_5*S_{18}+S_6*S_{18}+S_6+S_7*S_8*S_{18}*S_{20}*S_{21}+S_7*S_8*S_{18}*S_{21}+S_7*S_{12}*S_{18}*S_{19}*S_{21}+S_7*S_{18}*S_{20}*S_{21}+S_8*S_9*S_{19}*S_{22}+S_8*S_9*S_{21}*S_{22}+S_8*S_{13}*S_{20}*S_{22}+S_8*S_{18}*S_{20}*S_{21}*S_{22}+S_8*S_{18}*S_{20}+S_8*S_{18}*S_{21}*S_{22}+S_8*S_{18}*S_{22}+S_8*S_{18}+S_8*S_{21}*S_{22}+S_9*S_{19}+S_9*S_{21}+S_9+S_{12}*S_{18}*S_{19}*S_{21}*S_{22}+S_{12}*S_{18}*S_{21}+S_{12}+S_{13}*S_{22}+S_{13}+S_{14}*S_{18}*S_{19}+S_{15}*S_{18}+S_{15}*S_{20}+S_{16}+S_{17}*S_{18}*S_{21}+S_{17}*S_{18}+S_{18}*S_{20}*S_{21}*S_{22}+S_{18}*S_{20}*S_{22}+S_{18}*S_{20}+S_{18}*S_{22}+S_{19}+S_{21}+S_{22}+S_5*S_{13}*S_{22}*S_{23}+S_5*S_{13}*S_{23}+S_5*S_{20}*S_{22}*S_{23}+S_5*S_{20}*S_{23}+S_9*S_{19}*S_{22}*S_{23}+S_9*S_{19}*S_{23}+S_9*S_{21}*S_{22}*S_{23}+S_{13}*S_{20}*S_{22}*S_{23}+S_{21}*S_{22}*S_{23}+S_{21}*S_{23}+S_{24}*S_{28}+S_{24}+S_{14}*S_{15}*S_{25}*S_{28}+S_{15}*S_{25}*S_{29}+S_{15}*S_{25}+S_{15}*S_{25}*S_{28}*S_{29}+S_{18}*S_{25}+S_{25}+S_{11}*S_{14}*S_{26}*S_{28}+S_{11}*S_{26}*S_{28}*S_{29}+S_{11}*S_{26}*S_{29}+S_{14}*S_{19}*S_{26}*S_{28}+S_{21}*S_{26}+S_{19}*S_{26}*S_{28}*S_{29}+S_{26}+S_{14}*S_{27}*S_{28}+S_{11}*S_{14}*S_{19}*S_{28}+S_{11}*S_{19}*S_{28}*S_{29}+S_{14}*S_{15}*S_{27}*S_{28}+S_{15}*S_{27}*S_{28}*S_{29}+S_{15}*S_{27}+S_{27}*S_{28}*S_{29}+S_{27}*S_{29}+S_{27}+S_{19}*S_{28}+S_{11}*S_{19}*S_{29}+S_2*S_{18}+B_{63}*S_{24}*S_{28}+B_{23}*B_{62}*B_{65}+B_{10}*B_{23}*B_{66}+B_{24}*B_{67}$;

- $f_{16}(G,S_{23},\ldots,S_{30},S_2,B_{63},\ldots,B_{69}) = B_1*B_{24}*B_{85}*B_{88}*B_{89}+B_2*B_{24}+B_2*B_{60}+B_4*B_{24}*B_{60}*B_{85}*B_{88}*B_{89}+B_5*B_{24}*B_{61}+B_5*B_{60}*B_{61}+B_8*B_{24}+B_{11}*B_{13}*B_{24}*B_{85}*B_{88}*B_{89}+B_{12}*B_{14}*B_{24}+B_{12}*B_{14}*B_{60}+B_{16}*B_{17}*B_{24}*B_{85}*B_{88}*B_{89}+B_{17}*B_{18}*B_{24}+B_{17}*B_{18}*B_{60}+B_{18}*B_{20}*B_{24}+B_{21}*B_{23}*B_{24}*B_{85}*B_{88}*B_{89}+B_{21}+B_{22}*$

$$B_{24} * B_{25} * B_{60} + B_{22} * B_{24} * B_{25} + B_{23} * B_{24} + B_{23} + B_{24} * B_{25} * B_{85} * B_{88} * B_{89} + B_{24} * B_{26} * B_{54} * B_{85} * B_{88} * B_{89} + B_{24} * B_{26} + B_{24} * B_{27} * B_{55} + B_{24} * B_{28} * B_{30} * B_{31} + B_{24} * B_{32} + B_{24} * B_{33} * B_{61} + B_{24} * B_{50} * B_{85} * B_{88} * B_{89} + B_{24} * B_{51} + B_{24} * B_{56} * B_{59} * B_{85} * B_{88} * B_{89} + B_{24} * B_{57} * B_{60} + B_{24} * B_{57} + B_{24} * B_{85} * B_{88} * B_{89} * S_1 + B_{24} * B_{85} * B_{88} * B_{89} + B_{24} * B_{86} + B_{24} * B_{87} + B_{24} * S_2 + B_{24} * S_8 + B_{26} * B_{60} + B_{27} * B_{55} * B_{60} + B_{27} + B_{50} * B_{83} * B_{89} + B_{51} * B_{60} + B_{56} + B_{57} * B_{60} + B_{60} * B_{86} + B_{60} * S_2 + B_{61} + B_{87} + S_1 * S_{19} + S_3 * S_{19} + S_4 + S_5 * S_8 * S_{13} * S_{19} * S_{22} + S_5 * S_8 * S_{19} * S_{20} * S_{22} + S_6 * S_{19} + S_7 * S_{19} + S_7 + S_8 * S_9 * S_{19} * S_{21} * S_{22} + S_8 * S_9 * S_{19} * S_{22} + S_8 * S_{13} * S_{19} * S_{20} * S_{22} + S_8 * S_{19} * S_{21} * S_{22} + S_9 * S_{19} * S_{21} + S_9 * S_{19} + S_{10} * S_{20} + S_{10} * S_{22} + S_{10} + S_{13} * S_{19} * S_{22} + S_{13} + S_{14} + S_{15} * S_{19} * S_{20} + S_{16} * S_{19} + S_{16} * S_{21} + S_{17} + S_{18} * S_{19} * S_{22} + S_{18} * S_{19} + S_{19} * S_{21} + S_{20} + S_{22} + S_5 * S_{13} * S_{19} * S_{22} * S_{23} + S_5 * S_{13} * S_{19} * S_{23} + S_5 * S_{19} * S_{20} * S_{22} * S_{23} + S_5 * S_{19} * S_{20} * S_{23} + S_6 * S_9 * S_{14} * S_{23} + S_6 * S_9 * S_{21} * S_{23} + S_6 * S_{14} * S_{23} * S_{24} + S_6 * S_{21} * S_{23} * S_{24} + S_9 * S_{10} * S_{20} * S_{23} + S_9 * S_{10} * S_{22} * S_{23} + S_9 * S_{14} * S_{21} * S_{23} + S_9 * S_{19} * S_{21} * S_{22} * S_{23} + S_9 * S_{19} * S_{22} * S_{23} + S_9 * S_{19} * S_{23} + S_9 * S_{22} * S_{23} + S_{10} * S_{20} * S_{23} * S_{24} + S_{10} * S_{22} * S_{23} * S_{24} + S_{13} * S_{19} * S_{20} * S_{22} * S_{23} + S_{14} * S_{21} * S_{23} * S_{24} + S_{14} * S_{23} + S_{19} * S_{21} * S_{22} * S_{23} + S_{19} * S_{21} * S_{23} + S_{19} * S_{23} + S_{22} * S_{23} * S_{24} + S_{23} + S_6 * S_{14} * S_{24} + S_6 * S_{21} * S_{24} + S_{10} * S_{20} * S_{24} + S_{22} * S_{24} + S_{25} * S_{29} + S_{25} + S_{15} * S_{16} * S_{26} * S_{29} + S_{19} * S_{26} + S_{16} * S_{26} * S_{29} * S_{30} + S_{16} * S_{26} * S_{30} + S_{16} * S_{26} + S_{26} + S_{12} * S_{27} * S_{29} * S_{30} + S_{12} * S_{27} * S_{30} + S_{12} * S_{15} * S_{20} * S_{29} + S_{12} * S_{15} * S_{27} * S_{29} + S_{20} * S_{27} * S_{29} * S_{30} + S_{22} * S_{27} + S_{15} * S_{20} * S_{27} * S_{29} + S_{27} + S_{15} * S_{28} * S_{29} + S_{16} * S_{28} * S_{29} * S_{30} + S_{16} * S_{28} + S_{28} * S_{29} * S_{30} + S_{28} * S_{30} + S_{15} * S_{16} * S_{28} * S_{29} + S_{28} + S_{12} * S_{20} * S_{29} * S_{30} + S_{20} * S_{29} + S_{12} * S_{20} * S_{30} + S_2 + B_{24} * B_{63} * B_{66} + B_{64} * S_{25} * S_{29} + B_{11} * B_{24} * B_{67} + B_{25} * B_{68}.$$

Reference

[1] Claude E. Shannon. "Communication theory of secrecy systems". *Bell System Technical Journal* 28.4 (1949), pp. 656–715.

[2] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. 1996.

[3] *eSTREAM : Stream cipher project for ECrypt.* http://www.ecrypt.eu.org/stream/. 2005.

[4] Martin Hell, Thomas Johansson, and Willi Meier. "Grain: A Stream Cipher for Constrained Environments". *International Journal of Wireless and Mobile Computing* 2.1 (2007), pp. 86–93.

[5] Steve Babbage and Matthew Dodd. "The MICKEY Stream Ciphers". In: *New Stream Cipher Designs - The eSTREAM Finalists*. Vol. 4986. Lecture Notes in Computer Science. Springer, 2008, pp. 191–209.

[6] Christophe De Cannière and Bart Preneel. "Trivium". In: *New Stream Cipher Designs - The eSTREAM Finalists*. Vol. 4986. Lecture Notes in Computer Science. Springer, 2008, pp. 244–266.

[7] Daniel J. Bernstein. "The Salsa20 Family of Stream Ciphers". In: *New Stream Cipher Designs - The eSTREAM Finalists*. Vol. 4986. Lecture Notes in Computer Science. Springer, 2008, pp. 84–97.

[8] Côme Berbain, Olivier Billet, Anne Canteaut, et al. "Sosemanuk, a Fast Software-Oriented Stream Cipher". In: *New Stream Cipher Designs - The eSTREAM Finalists*. Vol. 4986. Lecture Notes in Computer Science. Springer, 2008, pp. 98–118.

[9] Hongjun Wu. "The Stream Cipher HC-128". In: *New Stream Cipher Designs - The eSTREAM Finalists*. Vol. 4986. Lecture Notes in Computer Science. Springer, 2008, pp. 39–47.

[10] Martin Boesgaard, Mette Vesterager, and Erik Zenner. "The Rabbit Stream Cipher". In: *New Stream Cipher Designs - The eSTREAM Finalists*. Vol. 4986. Lecture Notes in Computer Science. Springer, 2008, pp. 69–83.

[11] *National institute of standards and technology: Lightweight Cryptography (LWC) Standardization project.* https://csrc.nist.gov/Projects/lightweight-cryptography/round-2-candidates. 2019.

[12] Martin Hell, Thomas Johansson, Willi Meier, et al. "An AEAD Variant of the Grain Stream Cipher". In: *Codes, Cryptology and Information Security,C2SI 2019, Rabat, Morocco*. Vol. 11445. Lecture Notes in Computer Science. Springer, 2019, pp. 55–71.

[13] Matthias Hamann, Matthias Krause, and Willi Meier. "LIZARD - A Lightweight Stream Cipher for Power-constrained Devices". *IACR Transactions on Symmetric Cryptology* 2017.1 (2017), pp. 45–79.

[14] Vahid Amin Ghafari and Honggang Hu. "Fruit-80: A Secure Ultra-Lightweight Stream Cipher for Constrained Environments". *Entropy* 20.3 (2018), p. 180.

[15] Vasily Mikhalev, Frederik Armknecht, and Christian Müller. "On Ciphers that Continuously Access the Non-Volatile Key". *IACR Transactions on Symmetric Cryptology* 2016.2 (2016), pp. 52–79.

[16] August Kerckhoffs. *Kerckhoffs's Principle*. http://www.petitcolas.net/fabien/kerckhoffs/. 1983.

[17] Simon Knellwolf, Willi Meier, and María Naya-Plasencia. "Conditional Differential Cryptanalysis of NLFSR-Based Cryptosystems". In: *Advances in Cryptology - ASIACRYPT 2010*. Vol. 6477. Lecture Notes in Computer Science. Springer, 2010, pp. 130–145.

[18] Deepak Kumar Dalai, Subhamoy Maitra, Santu Pal, et al. "Distinguisher and non-randomness of Grain-v1 for 112, 114 and 116 initialisation rounds with multiple-bit difference in IVs". *IET Information Security* 13.6 (2019), pp. 603–613.

[19] Deepak Kumar Dalai, Santu Pal, and Santanu Sarkar. "Some Conditional Cube Testers for Grain-128a of Reduced Rounds". *IEEE Transactions on Computers, DOI:10.1109/TC.2021.3085144* (2021).

[20] Deepak Kumar Dalai and Santu Pal. "Recovering Internal States of Grain-v1". In: *Information Security Practice and Experience - ISPEC 2019, Kuala Lumpur, Malaysia*. Vol. 11879. Lecture Notes in Computer Science. Springer, 2019, pp. 325–337.

[21] Deepak Kumar Dalai and Santu Pal. "Wip: Degree Evaluation of Grain-v1". In: *Information Systems Security, ICISS 2019, Hyderabad, India*. Vol. 11952. Lecture Notes in Computer Science. Springer, 2019, pp. 239–251.

[22] Rudolf Lidl and Harald Niederreiter. *Finite Fields, 2nd ed.* 1997.

[23] P.B. Bhattacharya, S.K. Jain, and S.R. Nagpal. *Basic Abstruct Algebra, 2nd ed.* 1995.

[24] Joseph A. Gallian. *Contemporary Abstract Algebra, 4th ed.* 1999.

[25] Alfred J. Menezes(Editor). *Applications of Finite Fields*. 1993.

[26] Sheldon Axler. *Linear Algebra Done Right*. 2015.

[27] Giovanni Landi and Alessandro Zampini. *Linear Algebra and Analytic Geometry for Physical Sciences*. 2018.

[28] Thomas W. Cusick and Pantelimon Stanica. *Cryptographic Boolean Functions and Applications: Second edition*. 2017.

[29] Larry Wasserman. *All of Statistics, A Concise Course in Statistical Inference*. 2004.

[30] Martin Hell, Thomas Johansson, Alexander Maximov, et al. "A Stream Cipher Proposal: Grain-128". In: *IEEE International Symposium on Information Theory, ISIT 2006, The Westin Seattle, Seattle, Washington, USA*. IEEE, 2006, pp. 1614–1618.

[31] Martin Ågren, Martin Hell, Thomas Johansson, et al. "Grain-128a: a new version of Grain-128 with optional authentication". *IJWMC* 5.1 (2011), pp. 48–59.

[32] Itai Dinur and Adi Shamir. "Cube Attacks on Tweakable Black Box Polynomials". In: *Advances in Cryptology - EUROCRYPT 2009, Cologne, Germany*. Vol. 5479. Lecture Notes in Computer Science. Springer, 2009, pp. 278–299.

[33] Itsik Mantin and Adi Shamir. "A Practical Attack on Broadcast RC4". In: *Fast Software Encryption, FSE 2001 Yokohama,Japan*. Vol. 2355. Lecture Notes in Computer Science. Springer, 2001, pp. 152–164.

[34] Santanu Sarkar, Subhamoy Maitra, and Anubhab Baksi. "Observing biases in the state: case studies with Trivium and Trivia-SC". *Designs, Codes and Cryptography* 82.1-2 (2017), pp. 351–375.

[35] Simon Knellwolf, Willi Meier, and María Naya-Plasencia. "Conditional Differential Cryptanalysis of NLFSR-Based Cryptosystems". In: *Advances in Cryptology - ASIACRYPT 2010*. Vol. 6477. Lecture Notes in Computer Science. Springer, 2010, pp. 130–145.

[36] Jean-Philippe Aumasson, Itai Dinur, Willi Meier, et al. "Cube Testers and Key Recovery Attacks on Reduced-Round MD6 and Trivium". In: *Fast Software Encryption, FSE 2009, Leuven, Belgium*. Vol. 5665. Lecture Notes in Computer Science. Springer, 2009, pp. 1–22.

[37] Martin Hellman. "A cryptanalytic time-memory trade-off". *IEEE Transactions on Information Theory* 26.4 (1980), pp. 401–406.

[38] Steve Babbage. "A Space/Time Tradeoff in Exhaustive Search Attacks on Stream Ciphers." In: *European Convention on Security and Detection*. Vol. 408. IEE Conference Publication, 1995.

[39] Jovan Dj. Golić. "Cryptanalysis of Alleged A5 Stream Cipher". In: *Advances in Cryptology – EUROCRYPT 1997*. Vol. 1233. Lecture Notes in Computer Science. Springer-Verlag, 1997, pp. 239–255.

[40] Alex Biryukov and Adi Shamir. "Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers". In: *Advances in Cryptology – ASIACRYPT 2000*. Vol. 1976. Lecture Notes in Computer Science. Springer-Verlag, 2000, pp. 1–13.

[41] Subhadeep Banik. "Conditional differential cryptanalysis of 105 round Grain v1". *Cryptography and Communications* 8.1 (2016), pp. 113–137.

[42] Santanu Sarkar. "A New Distinguisher on Grain v1 for 106 Rounds". In: *Information Systems Security, ICISS 2015*. Vol. 9478. Lecture Notes in Computer Science. Springer, 2015, pp. 334–344.

[43] Zhen Ma, Tian Tian, and Wen-Feng Qi. "Improved conditional differential attacks on Grain v1". *IET Information Security* 11.1 (2017), pp. 46–53.

[44] Yuhei Watanabe, Yosuke Todo, and Masakatu Morii. "New Conditional Differential Cryptanalysis for NLFSR-based Stream Ciphers and Application to Grain v1". In: *11th Asia Joint Conference on Information Security - AsiaJCIS 2016*. IEEE Computer Society, 2016, pp. 115–123.

[45] Subhadeep Banik, Subhamoy Maitra, and Santanu Sarkar. "A Differential Fault Attack on the Grain Family of Stream Ciphers". In: *Cryptographic Hardware and Embedded Systems - CHES 2012*. Vol. 7428. Lecture Notes in Computer Science. Springer, 2012, pp. 122–139.

[46] Itai Dinur and Adi Shamir. "Breaking Grain-128 with Dynamic Cube Attacks". In: *Fast Software Encryption - FSE 2011*. Vol. 6733. Lecture Notes in Computer Science. Springer, 2011, pp. 167–187.

[47] Simon Fischer, Shahram Khazaei, and Willi Meier. "Chosen IV Statistical Analysis for Key Recovery Attacks on Stream Ciphers". In: *Progress in Cryptology - AFRICACRYPT 2008*. Vol. 5023. Lecture Notes in Computer Science. Springer, 2008, pp. 236–245.

[48] Subhadeep Banik, Subhamoy Maitra, and Santanu Sarkar. "A Differential Fault Attack on the Grain Family under Reasonable Assumptions". In: *Progress in Cryptology - INDOCRYPT 2012*. Vol. 7668. Lecture Notes in Computer Science. Springer, 2012, pp. 191–208.

[49] Subhadeep Banik. "Some Insights into Differential Cryptanalysis of Grain v1". In: *Information Security and Privacy - ACISP 2014*. Vol. 8544. Lecture Notes in Computer Science. Springer, 2014, pp. 34–49.

[50] Zhen Ma, Tian Tian, and Wen-Feng Qi. "Conditional differential attacks on Grain-128a stream cipher". *IET Information Security* 11.3 (2017), pp. 139–145.

[51] Simon Knellwolf. "Cryptanalysis of hardware-oriented ciphers the Knapsack generator, and SHA-1". PhD dissertation, Zurich (2012).

[52] Yuseop Lee, Kitae Jeong, Jaechul Sung, et al. "Related-Key Chosen IV Attacks on Grain-v1 and Grain-128". In: *Information Security and Privacy- ACISP 2008*. Vol. 5107. Lecture Notes in Computer Science. Springer, 2008, pp. 321–335.

[53] Christophe De Cannière, Özgül Küçük, and Bart Preneel. "Analysis of Grain's Initialization Algorithm". In: *Progress in Cryptology - AFRICACRYPT 2008*. Vol. 5023. Lecture Notes in Computer Science. Springer, 2008, pp. 276–289.

[54] Bin Zhang, Chao Xu, and Willi Meier. "Fast Near Collision Attack on the Grain v1 Stream Cipher". In: *Advances in Cryptology - EUROCRYPT 2018*. Vol. 10821. Lecture Notes in Computer Science. Springer, 2018, pp. 771–802.

[55] William Stein et al. "SAGE: Open source mathematical software" (2008).

[56] *Supporting Materials for Grain-v1.* https://drive.google.com/drive/folders/1FukMCaCeCRVidgQVMLpf_L5yFoVC60-9?usp=sharing.

[57] Paul Stankovski. "Greedy Distinguishers and Nonrandomness Detectors". In: *Progress in Cryptology - INDOCRYPT 2010, Hyderabad, India*. Vol. 6498. Lecture Notes in Computer Science. Springer, 2010, pp. 210–226.

[58] Michael Lehmann and Willi Meier. "Conditional Differential Cryptanalysis of Grain-128a". In: *Cryptology and Network Security, CANS 2012, Darmstadt, Germany*. Vol. 7712. Springer, 2012, pp. 1–11.

[59] Linus Karlsson, Martin Hell, and Paul Stankovski. "Not So Greedy: Enhanced Subset Exploration for Nonrandomness Detectors". In: *Information Systems Security and Privacy - Third International Conference, ICISSP 2017, Porto, Portugal*. Vol. 867. Communications in Computer and Information Science. Springer, 2017, pp. 273–294.

[60] Vahid Amin Ghafari and Honggang Hu. "A New Chosen IV Statistical Attack on Grain-128a Cipher". In: *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2017, Nanjing, China*. IEEE, 2017, pp. 58–62.

[61] Vahid Amin Ghafari and Honggang Hu. "A new chosen IV statistical distinguishing framework to attack symmetric ciphers, and its application to ACORN-v3 and Grain-128a". *J. Ambient Intelligence and Humanized Computing* 10.6 (2019), pp. 2393–2400.

[62] Yosuke Todo, Takanori Isobe, Willi Meier, et al. "Fast Correlation Attack Revisited - Cryptanalysis on Full Grain-128a,Grain-128, and Grain-v1". In: *Advances in Cryptology - CRYPTO 2018*. Vol. 10992. Lecture Notes in Computer Science. Springer, 2018, pp. 129–159.

[63] Jean-Philippe Aumasson, Itai Dinur, Luca Henzen, et al. "Efficient FPGA Implementations of High-Dimensional Cube Testers on the Stream Cipher Grain-128". *IACR Cryptology ePrint Archive* 2009 (2009), p. 218.

[64] Itai Dinur, Tim Güneysu, Christof Paar, et al. "An Experimentally Verified Attack on Full Grain-128 Using Dedicated Reconfigurable Hardware". In: *Advances in Cryptology - ASIACRYPT 2011, Seoul, South Korea*. Vol. 7073. Lecture Notes in Computer Science. Springer, 2011, pp. 327–343.

[65] Itai Dinur and Adi Shamir. "Breaking Grain-128 with Dynamic Cube Attacks". In: *Fast Software Encryption, FSE 2011, Lyngby, Denmark*. Vol. 6733. Lecture Notes in Computer Science. Springer, 2011, pp. 167–187.

[66] Simon Knellwolf, Willi Meier, and María Naya-Plasencia. "Conditional Differential Cryptanalysis of NLFSR-Based Cryptosystems". In: *Advances in Cryptology - ASIACRYPT 2010,Singapore*. Vol. 6477. Lecture Notes in Computer Science. Springer, 2010, pp. 130–145.

[67] Lin Ding, Chenhui Jin, Jie Guan, et al. "New state recovery attacks on the Grain v1 stream cipher". *China Communications* 13.11 (2016), pp. 180–188.

[68] Lin Jiao, Bin Zhang, and Mingsheng Wang. "Two Generic Methods of Analyzing Stream Ciphers". In: *International Conference on Information Security - ISC 2015*. Vol. 9290. Lecture Notes in Computer Science. Springer-Verlag, 2015, pp. 379–396.

[69] Miodrag Mihaljević, Nishant Sinha, Sugata Gangopadhyay, et al. "An Improved Cryptanalysis of Lightweight Stream Cipher Grain-v1". In: *Cryptacus: Workshop and MC meeting*. 2017.

[70] Subhamoy Maitra, Nishant Sinha, Akhilesh Siddhanti, et al. "A TMDTO Attack Against Lizard". *IEEE Transactions on Computers* 67.5 (2018), pp. 733–739.

[71] Tor E. Bjørstad. *Cryptanalysis of Grain using Time / Memory / Data Tradeoffs*. http://www.ecrypt.eu.org/stream. 2008.

[72] Fabian van den Broek and Erik Poll. "A Comparison of Time-Memory Trade-Off Attacks on Stream Ciphers". In: *Progress in Cryptology - AFRICACRYPT 2013*. Vol. 7918. Lecture Notes in Computer Science. Springer-Verlag, 2013, pp. 406–423.

[73] Lin Ding, Chenhui Jin, Jie Guan, et al. "New Treatment of the BSW Sampling and Its Applications to Stream Ciphers". In: *Progress in Cryptology - AFRICACRYPT 2014*. Vol. 8469. Lecture Notes in Computer Science. Springer-Verlag, 2014, pp. 136–146.

[74] Orr Dunkelman and Keller Nathan. "Treatment of the initial value in Time-Memory-Data Tradeoff attacks on stream ciphers". *Information Processing Letters* 107.5 (2008), pp. 133–137.

[75] Miodrag J. Mihaljević, Sugata Gangopadhyay, Goutam Paul, et al. "Internal state recovery of grain-v1 employing normality order of the filter function". *IET Information Security* 6.2 (2012), pp. 55–64.

[76] Akhilesh Siddhanti, Subhamoy Maitra, and Nishant Sinha. "Certain Observations on ACORN v3 and Grain v1—Implications Towards TMDTO Attacks". *Journal of Hardware and Systems Security* 3.1 (2019), pp. 64–77.

[77] Alex Biryukov, Adi Shamir, and David Wagner. "Real Time Cryptanalysis of A5/1 on a PC". In: *Fast Software Encryption–FSE 2000*. Vol. 1978. Lecture Notes in Computer Science. Springer-Verlag, 2001, pp. 1–18.

[78] Anne Canteaut and Marion Videau. "Degree of Composition of Highly Nonlinear Functions and Applications to Higher Order Differential Cryptanalysis". In: *Advances in Cryptology - EUROCRYPT 2002*. Vol. 2332. Lecture Notes in Computer Science. Springer, 2002, pp. 518–533.

[79] Christina Boura, Anne Canteaut, and Christophe De Cannière. "Higher-Order Differential Properties of Keccak and *Luffa*". In: *Fast Software Encryption - FSE 2011*. Vol. 6733. Lecture Notes in Computer Science. Springer, 2011, pp. 252–269.

[80] Christina Boura and Anne Canteaut. “On the Influence of the Algebraic Degree of F^{-1} on the Algebraic Degree of $G \circ F$”. *IEEE Transactions on Information Theory* 59.1 (2013), pp. 691–702.

[81] Meicheng Liu. “Degree Evaluation of NFSR-Based Cryptosystems”. In: *Advances in Cryptology - CRYPTO 2017*. Vol. 10403. Lecture Notes in Computer Science. Springer, 2017, pp. 227–249.

[82] Chen-Dong Ye and Tian Tian. “Deterministic Cube Attacks: A New Method to Recover Superpolies in Practice”. *IACR Cryptology ePrint Archive* 2018 (2018), p. 1082.

[83] Ximing Fu, Xiaoyun Wang, Xiaoyang Dong, et al. “A Key-Recovery Attack on 855-round Trivium”. In: *Advances in Cryptology - CRYPTO 2018*. Vol. 10992. Lecture Notes in Computer Science. Springer, 2018, pp. 160–184.

[84] Chen-Dong Ye and Tian Tian. “A New Framework for Finding Nonlinear Superpolies in Cube Attacks Against Trivium-Like Ciphers”. In: *Information Security and Privacy - ACISP 2018*. Vol. 10946. Lecture Notes in Computer Science. Springer, 2018, pp. 172–187.

[85] Bin Zhang, Zhenqi Li, Dengguo Feng, et al. “Near Collision Attack on the Grain v1 Stream Cipher”. In: *Fast Software Encryption - FSE 2013*. Vol. 8424. Lecture Notes in Computer Science. Springer, 2013, pp. 518–538.