ON STATISTICAL ANALYSIS OF SYNCHRONOUS STREAM CIPHERS

MELTEM SÖNMEZ TURAN

APRIL 2008

ON STATISTICAL ANALYSIS OF SYNCHRONOUS STREAM CIPHERS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

MELTEM SÖNMEZ TURAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN
THE DEPARTMENT OF CRYPTOGRAPHY

APRIL 2008

Approval of the Graduate School of Applied Mathematics

Prof. Dr. Ersan AKYILDIZ
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Doctor of Philosophy.

Prof. Dr. Ferruh ÖZBUDAK
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Doctor of Philosophy.

Assoc. Prof. Dr. Ali DOĞANAKSOY
Supervisor

Examining Committee Members

Prof. Dr. Ersan AKYILDIZ

Prof. Dr. Ferruh ÖZBUDAK

Prof. Dr. Semih Koray

Assoc. Prof. Dr. Ali DOĞANAKSOY

Dr. Orhun KARA

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name :

Signature :

Abstract

ON STATISTICAL ANALYSIS OF SYNCHRONOUS STREAM CIPHERS

Sönmez Turan, Meltem
Ph.D., Department of Cryptography
Supervisor: Assoc. Prof. Ali Doğanaksoy

April 2008, 146 pages

Synchronous stream ciphers constitute an important class of symmetric ciphers. After the call of the eSTREAM project in 2004, 34 stream ciphers with different design approaches were proposed. In this thesis, we aim to provide a general framework to analyze stream ciphers statistically. Firstly, we consider stream ciphers as pseudo random number generators and study the quality of their output. We propose three randomness tests based on one dimensional random walks. Moreover, we theoretically and experimentally analyze the relations of various randomness tests.

We focus on the ideas of algebraic, time memory tradeoff (TMTO) and correlation attacks and propose a number of chosen IV distinguishers. We experimentally observe statistical weaknesses in some of the stream ciphers that are believed to be secure.

Keywords: Synchronous stream ciphers, Statistical analysis, eSTREAM, Distinguishing attacks.

Öz

SENKRONİZE AKAN ŞİFRELERİN İSTATİSTİKSEL ANALİZİ ÜZERİNE

Sönmez Turan, Meltem
Doktora, Kriptografi Bölümü
Tez Yöneticisi: Doç. Dr. Ali Doğanaksoy

Nisan 2008, 146 sayfa

Senkronize akan şifreler simetrik anahtarlı kriptosistemlerin önemli bir parçasını oluşturur. 2004 yılında duyurulan eSTREAM projesi üzerine, farklı tasarımlara sahip 34 akan şifre önerilmiştir. Bu tezde, senkronize akan şifrelerin istatistiksel analizi üzerine genel bir çerçeve verilmesi hedeflenmiştir. İlk olarak, akan şifreler rastgele sayı üreteçleri olarak düşünülmüş ve çıktılarının kalitesi çalışılmıştır. Tek boyutlu rastgele yürüyüşlere dayanan üç test önerilmiştir. Ayrıca, teorik ve deneysel olarak testlerin birbirleri ile olan ilişkileri çalışılmıştır.

Cebirsel, zaman-hafıza özdünleşimi ve ilinti ataklarının fikirleri üzerinde durularak ayırt edici testler önerilmiştir. Güvenli olduğu düşünülen bazı şifrelerde deneysel zayıflıklar gözlemlenmiştir.

Anahtar Kelimeler: Senkronize akan şifreler, İstatistiksel analiz, eSTREAM, Ayırt edici ataklar.

To Fırat,

Acknowledgments

This thesis has been an inspiring, exciting and challenging experience. During the preparation of this thesis, I have been accompanied and supported by a great number of people whose contribution is worth to mention.

First, I would like to express my deep and sincere thanks to my supervisor, Assoc. Prof. Ali Doğanaksoy for his mentoring, expertise and guidance throughout the research.

Many thanks go to all members of the Institute of Applied Mathematics especially to Prof. Dr. Ersan Akyıldız for providing an excellent and inspiring working atmosphere.

Special thanks go to Dr. Orhun Kara, for his guidance and ideas.

I thank Çağdaş Çalık for his endless help in implementation of tests, proof reading of the thesis and most importantly for being a good friend. My sincere thanks go to Onur Özen and Kerem Varıcı for their motivation and encouragement. I thank Fatih Sulak for his help on probability calculations. I also thank Nurdan Buz Saran, Güçlü Dündar and Zülfükar Saygı for their support.

I gratefully thank Prof. Dr. Thomas Johansson for giving me the opportunity to work with him. Many thanks go to friends in Lund especially to Håkan, Martin and Koraljka.

I greatly appreciate the guidance of Prof. Dr. Nur Evin Özdemirel and Assoc. Prof. Dr. Haldun Süral during my MS studies, who taught me the importance of academic work.

Finally, I wish to express my love and gratitude to my family. I'd particularly like to thank my parents and brother Tunç. Last but not least, without the understanding of my husband Fırat, it would have been impossible for me to finish this work.

Table of Contents

Plagiarism iii

Abstract iv

Öz v

Acknowledgments vii

Table of Contents viii

List of Figures xi

List of Tables xiii

List of Abbreviations xvii

1 Introduction 1

1.1 Motivation 5

1.1.1 Outline of the Thesis 6

2 Brief Overview of Synchronous Stream Ciphers 8

2.1 Basic Building Blocks 9

2.1.1 Feedback Shift Registers 9

2.1.2 Boolean Functions and S-boxes 14

2.1.3 T-functions 15

2.2 Desired Properties of Keystream 16

2.2.1 Period 17

2.2.2 Randomness . . . 17
2.2.3 Complexity Measures . . . 20
2.3 Common Designs . . . 23
2.3.1 Designs Based on Feedback Shift Registers . . . 24
2.3.2 Designs Based on Block Ciphers . . . 28
2.3.3 Designs Based on Hash Functions . . . 30
2.3.4 Designs Based on NP-Hard Problems . . . 30
2.3.5 Other Designs . . . 31
2.4 Cryptanalysis of Stream Ciphers . . . 32
2.5 Distinguishing Attacks . . . 33

3 Analysis of Keystream . . . 38
3.1 Randomness Tests . . . 40
3.2 New Tests Based on Random Walks . . . 44
3.2.1 Random Walks . . . 45
3.2.2 Test Descriptions . . . 48
3.3 Independence of Tests . . . 50
3.3.1 Theoretical Results . . . 52
3.3.2 Experiments on Short Sequences . . . 53
3.4 Sensitivity of Tests . . . 57
3.5 Summary . . . 58

4 Tests Based on Algebraic Properties . . . 60
4.1 Basics of Algebraic Attacks . . . 61
4.2 Desired Properties of F_i's . . . 62
4.3 Analyzing Classical Designs . . . 63
4.4 A Case Study on Trivium . . . 67
4.4.1 Description of Trivium . . . 67
4.4.2 Linear Approximations . . . 69
4.4.3 Searching for Linear Approximations . . . 70

4.4.4 Linear Approximations for 2-round Trivium 71
4.4.5 Discussion 74
4.5 *d*-Monomial Approach 75
4.5.1 A Generalized Approach 77
4.5.2 Monomial Distribution Test 78
4.5.3 Maximal Degree Test 79
4.5.4 Experimental Results 81
4.5.5 Discussion 85
4.5.6 Improvement of Fischer *et al.*[1] 86
4.6 Linear Independence 86
4.6.1 Preliminaries 86
4.6.2 Linear Span Test 87
4.6.3 Experimental Results 88
4.7 Completeness Property 88
4.7.1 Diffusion Test 88
4.7.2 Experimental Results 90
4.8 Summary 90

5 Tests Based on Random Mappings 93
5.1 Preliminaries 94
5.2 Time Memory Tradeoff Attacks 95
5.3 Three New Distinguishers 98
5.3.1 Coverage Test 99
5.3.2 ρ-Test 100
5.3.3 DP-Coverage Test 102
5.4 Experimental Results 103
5.5 Summary 104

6 Tests Based on Correlations 105
6.1 Basics of Correlation Attacks 105

6.2 Tests Based on Correlation of Key, IV and Keystream 107
6.2.1 Key/Keystream Correlation Test 107
6.2.2 IV/Keystream Correlation Test 108
6.2.3 Frame Correlation Test 109
6.3 Experimental Results 110
6.4 Summary 112

7 Conclusion 114

References 117

Appendices 132

A Basics of Statistical Inference 132
A.1 Probability Theory 132
A.1.1 Some Special Distributions 133
A.2 Hypothesis Testing 135

B Other Attacks Against Stream Ciphers 139
B.1 Resynchronization Attacks 139
B.2 Guess and Determine Attacks 140
B.3 Side Channel Attacks 141

C NIST Test Results 143

D F_1 for 2-round Trivium 144

E Linear Regression Model for d-monomial Test of Grain 145

Vita 147

List of Figures

1.1 Encryption and decryption 1

1.2 Classification of cryptographic primitives 2

2.1 Generic structure of a synchronous stream cipher 9

2.2 Binary additive stream cipher 9

2.3 Block diagram of a FSR 10

2.4 Nonlinear combining generator 24

2.5 Nonlinear filtering generator 26

2.6 Distinguishing finite sequences 33

2.7 Distinguishing frames generated by different IVs 35

3.1 Independence and coverage of test suites 39

3.2 Multi level testing 44

3.3 Random walk examples 45

3.4 Height, excursion and expansion in one dimensional random walks . . . 46

3.5 Grid representation of an example random walk (1,2,1,2,1,0,-1,0,1,0,1,2,1,0,-1,-2,-1,-2,-1,-2,-3,-4,-3,-4) of length $w + m = 24$ with $\max\{y_i\}$ less than $r = 5$ and $\min\{y_i\}$ greater than $-s = -7$ 47

3.6 Distributions of test statistics for $n = 20$ 54

4.1 Linear approximations for n round stream ciphers 71

4.2 Linear approximations for 2-round Trivium 73

4.3 Number of clockings vs. $\min_i G(i, j)$ for original and proposed initialization of Trivium 75

5.1 Graphical representation of an iteration 95

5.2 Construction of the lookup table in the offline phase 96

5.3 The table generated in the coverage test 99

5.4 The rows generated in the ρ-test 100

5.5 Distinguished points 102

5.6 The number of p-values in intervals of length 0.1 versus expected values for Pomaranch 104

6.1 Use of BSC in fast correlation attacks 106

E.1 The linear regression model for d-monomial test of Grain 145

List of Tables

1.1 Timeline of the eSTREAM Project 5

2.1 Building blocks and proposed platforms of Phase III eSTREAM Candidates 25

3.1 A non-exhaustive list of statistical randomness tests with input parameters, test statistics and constraints (The sequence length is denoted as n.) 42

3.2 Interval and probability values of Random Walk Excursion Test for block lengths of 16,32, 64, 128 and 256 bits. 50

3.3 Interval and probability values of Random Walk Height Test for block lengths of 64,128, 256, 512 and 1024 bits. 50

3.4 Interval and probability values of Random Walk Expansion Test for block lengths of 32, 64 and 128 bits. 51

3.5 Lower Limits (LLs) and Upper Limits (ULs) of the test statistics for 20 bit sequences and corresponding type I error, α 55

3.6 Results of tests for all sequences of length $n = 20$ for $\alpha = 0.01$ 55

3.7 Results of tests for all sequences of length $n = 30$ for $\alpha = 0.01$ 56

3.8 Number of sequences that only fail the given test (but pass all other tests) 56

3.9 Sensitivity of randomness tests toward some transformations. 58

4.1 Details of the six small sized examples 66

4.2 Average cryptographic properties obtained using the first 40 Boolean functions 66

4.3 Comparison of d-monomial tests 76

4.4 Number of IV bits needed to attack the first keystream bit of Grain-128 for different number of rounds in the initialization (out of 256 rounds). . 81

4.5 Number of IV bits needed to attack the initial state variables Grain-128 for different number of rounds in the initialization (out of 256 rounds). . 82

4.6 Number of IV bits needed to attack the first keystream bit of Grain-128 with alternative key and IV loading for different number of rounds in the initialization (out of 256 rounds). 83

4.7 Number of IV bits needed to attack the initial state variables of Grain-128 with alternative key and IV loading for different number of rounds in the initialization (out of 256 rounds). 83

4.8 Number of IV bits needed to attack the first keystream bit of Trivium for different number of rounds in the initialization (out of 1152 rounds). 83

4.9 Number of IV bits needed to attack the initial state variables of Trivium for different number of rounds in the initialization (out of 1152 rounds). 84

4.10 Number of IV bits needed to attack the first keystream bit of Trivium with alternative key and IV loading for different number of rounds in the initialization (out of 1152 rounds). 84

4.11 Number of IV bits needed to attack the initial state variables of Trivium with alternative key and IV loading for different number of rounds in the initialization (out of 1152 rounds). 84

4.12 Number of IV bits needed to attack the first keystream bit of Decim-v2 for different number of rounds in the initialization (out of 768 rounds). . 85

4.13 Number of IV bits needed to attack the initial state variables of Decim-v2 for different number of rounds in the initialization (out of 768 rounds). . 85

4.14 The average of 100 p-values of Linear Span Test for Phase III eSTREAM Candidates 89

4.15 Interval and probability values of Diffusion Test using 1024 key and IV pairs. 89

4.16 The average of 100 p-values of Diffusion Test for Phase I eSTREAM Candidates 92

5.1 Interval and probability values of Coverage test using 12 and 14 IV bits 100

5.2 Interval and probability values of ρ-test using 15 and 20 IV bits 101

5.3 The average 100 p-values obtained from Coverage, ρ and DP-Coverage tests 103

6.1 Interval and probabilities values of Key/Keystream Correlation test for key size of 80 and 128 bits 109

6.2 Interval and probability values of IV/Keystream Correlation test for IV size of 64, 80 and 128. 110

6.3 The average of 100 p-values obtained using Key/Keystream Correlation, IV/Keystream Correlation and Frame Correlation Test against Phase I candidates of eSTREAM 113

C.1 The result of NIST tests that indicate weaknesses. There are total of 148 nonperiodic template test results for each cipher, and Decim fails 11 of them. 143

List of Abbreviations

AES	Advanced Encryption Standard
ANF	Algebraic Normal Form
BSC	Binary Symmetric Channel
CBC	Cipher Block Chaining
CFB	Cipher Feedback Mode
CTR	Counter Mode
DFT	Discrete Fourier Transform
DP	Distinguished Point
ECB	Electronic Codebook
ECRYPT	European Network of Excellence for Cryptology
EP	End Point
FCSR	Feedback Shift Registers with Carry
FSR	Feedback Shift Register
GF	Galois Field
IV	Initialization Vector
lcm	least common multiple
LEX	Leak Extraction
LFSR	Linear Feedback Shift Register
LL	Lower Limit
MAC	Message Authentication Code
MD	Message Digest
MOC	Maximum Order Complexity
NESSIE	New European Schemes for Signature, Integrity and Encryption
NFSR	Nonlinear Feedback Shift Register
NIST	National Institute of Standards and Technology
OFB	Output Feedback Mode
OTP	One Time Pad
PRNG	Pseudo Random Number Generator
S-box	Substitution-box

SHA	Secure Hash Algorithm
SP	Start Point
TMTO	Time Memory Tradeoff
XL	Extended Linearilization
XOR	Exclusive OR
XSL	Extended Sparse Linearilization
UL	Upper Limit

Chapter 1

Introduction

Cryptography is the branch of information security that has four major objectives. The first objective is *confidentiality* (or secrecy) that aims to keep the information secret from anyone except the intended receiver. The second objective is *integrity* that ensures that the message is not manipulated by unauthorized users during its transmission. *Authentication* aims to verify the identity of sender/receiver in a communication. Final objective is *non-repudiation* that aims to prevent users from denying their previous actions.

In cryptographic terminology, the process of encoding the *plaintext* (or message) so that the content is hidden from unintended parties is called *encryption*, whereas the process of converting the *ciphertext* (or encrypted message) back to the plaintext is called *decryption*. Encryption and decryption are done using keys K_e and K_d, respectively as given in Figure 1.1.

$$\text{Message } m \overset{\text{Encryption with } K_e}{\longrightarrow} \text{Ciphertext } c \overset{\text{Decryption with } K_d}{\longrightarrow} \text{Message } m$$

Figure 1.1: Encryption and decryption

Cryptanalysis is the study of cryptosystems with the aim of finding weaknesses that permit retrieval of the plaintext from the ciphertext without knowing the secret key. According to the famous principle of Kerckhoffs, *the security of ciphers should depend entirely on the secrecy of keys, not on the details of the encryption/decryption algorithms.*

Cryptographic primitives are algorithms that aim to provide cryptographic objectives and they are divided into three main groups; *unkeyed*, *asymmetric key* and *symmetric key primitives*. In Figure 1.2, the classification of cryptographic primitives are presented.

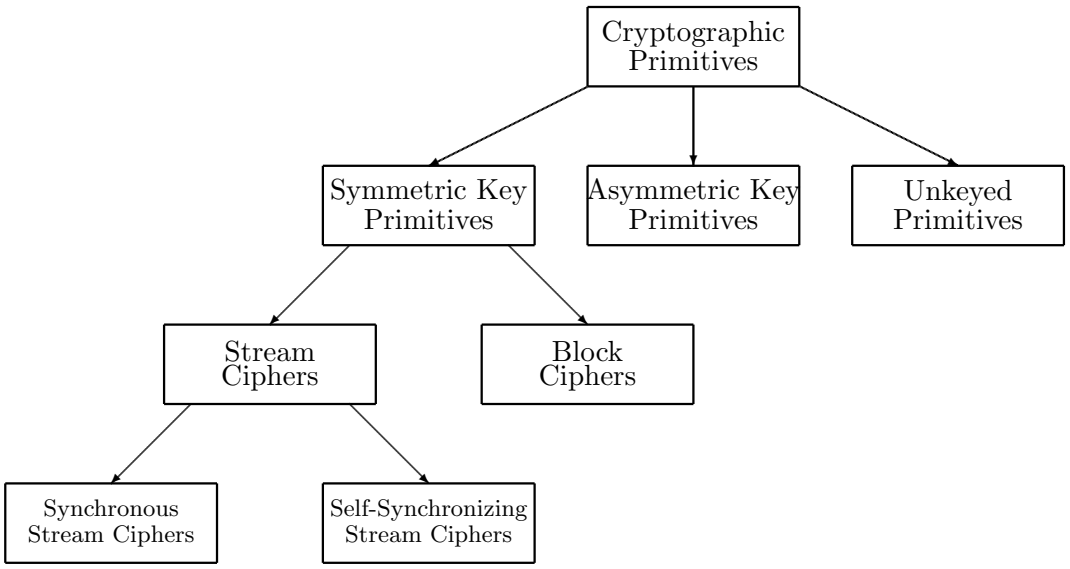


Figure 1.2: Classification of cryptographic primitives

Unkeyed Cryptography

The family of unkeyed primitives consists of the tools mainly used for message integrity and authentication. As the name implies, these primitives do not use secret keys, therefore no secrecy is involved in their algorithms. An important example of unkeyed cryptographic primitives is the *hash functions* that are fundamental components of many cryptographic applications such as digital signatures, random number generation, integrity protection, e-cash *etc.* MD5 (Message Digest) [2], SHA-1 (Secure Hash Algorithm) [3] and RIPEMD [4] are widely used hash functions especially in SSL, PGP, S/MIME, SSH and SFTP applications. Employing hash functions for these applications both increases the security and improves the efficiency of these systems.

The design of these functions are based on the hash function MD4 [5], as they iteratively use a compression function that inputs state variable and a fixed length block, and outputs another fixed length block. Recently, many attacks against hash functions having similar construction to MD4 are proposed [6, 7, 8, 9]. These recent studies motivated National Institute of Standards and Technology (NIST) to announce a public competition in 2007 to select a new cryptographic hash function to be used as the new standard [10].

Asymmetric Key Cryptography

Asymmetric key cryptography, also known as public key cryptography, is invented by Diffie and Hellman [11] in 1976. Here, each user has two different keys; a *public key*

K_e that is used for encryption and a *private key* K_d that is used for decryption. There is a mathematical relation between each public and private key, but private key cannot be practically derived from the public key.

To send a message m using asymmetric key cryptography, the public key of the intended receiver is used and the ciphertext c is obtained as $c = E_{K_e}(m)$. The receiver uses his private key and obtains the message as $m = D_{K_d}(c)$.

The security of asymmetric primitives rely on mathematical trapdoor functions that are easy to calculate in one direction, but hard to invert without knowing the secret trapdoor. As an example, RSA primitive is based on the famous *integer factorization* problem, that is, the problem of factorization of a given integer which is the product of two large primes.

Symmetric Key Cryptography

Symmetric key cryptography is a very important class of cryptographic primitives that is mainly used to provide confidentiality and integrity of transmitted data. Symmetric key ciphers are faster compared to asymmetric ciphers, therefore more suitable for applications requiring high data throughput.

In symmetric key cryptography, the sender and the receiver share the same secret key K (or in some cases different keys that can easily be generated from each other) that is used for encryption and decryption processes, in other words, $K_e = K_d = K$. Sender encrypts a message m and produces the ciphertext as $c = E_K(m)$. Then, the ciphertext is sent through an insecure channel. Receiver decrypts the ciphertext c and obtains the plaintext as $m = D_K(c)$.

Symmetric key primitives are mainly divided into parts; block and stream ciphers. *Block ciphers* are symmetric key encryption algorithms that transform a n bit block of plaintext into a block of ciphertext of the same length. For each secret key K, block ciphers define a permutation on the set of n bit binary blocks. Using the same key, decryption is performed by applying the inverse transformation to the ciphertext blocks.

Stream ciphers constitute another important class of symmetric key ciphers that are suitable for applications where the length of the plaintext is unknown in advance, such as secure wireless connections. Important examples of stream ciphers may be given as A5/1 used in GSM standard to provide security in the air link of voice and data communication, and E_0 [12], used to supply privacy in the radio network link, Bluetooth.

Self-synchronizing and *synchronous stream ciphers* are the two main types of stream ciphers. In self-synchronizing stream ciphers, the keystream is generated as a function of key and a fixed number of previous ciphertext bits. These ciphers have the advantage of automatically synchronizing after receiving a number of ciphertext digits.

Synchronous stream ciphers generate a *keystream* independent of plaintext and ciphertext and commonly the keystream is XORed (eXclusive ORed) with plaintext to produce ciphertext. Synchronous stream ciphers are widely adopted in many industrial, government and military applications compared to self-synchronizing stream ciphers. For correct decryption, the sender and receiver should be synchronized. In case of insertion/deletion of ciphertext bits, the synchronization is lost and should be restored using a pre-selected protocol. In case of bit flips, only the corresponding plaintext bits are affected and the error does not propagate to other parts of the message.

Recent Projects

In 2001, the block cipher Rijndael [13], developed by J. Daemen and V. Rijmen, was selected to be the Advanced Encryption Standard (AES), that became the new U.S. Federal Information Processing Standard to encrypt sensitive information. After selection of the AES, other similar projects have been announced. The first announcement is done by New European Schemes for Signature, Integrity and Encryption (NESSIE) [14] between 2000 and 2003 with the objective to generate strong cryptographic primitives in block ciphers, stream ciphers, hash functions, MAC algorithms, digital signatures schemes, and public key encryption schemes. For stream ciphers category, BMGL, Leviathan, LILI-128, SNOW, Sober-t16 and Sober-t32 were proposed, however no candidate in this category managed to satisfy the high security requirements, mainly due to the weaknesses against distinguishing attacks. This indicated that the study of stream ciphers is not as sound as the study of block ciphers.

Another project, CRYPTREC [15] was announced by Japanese government in 2000 to list cryptographic techniques to be used for electronic government applications. In 2003, CRYPTREC has recommended MUGI, Multi-S01, RC4 with 128 bit key length and some pseudo-random number generators using SHA-1 in the area of stream ciphers.

Also, in 2004, a new project eSTREAM [16] was announced by European Network of Excellence for Cryptology (ECRYPT), a 4-year network of excellence funded within the Information Societies Technology Program of the European Commission's Sixth Framework Program. eSTREAM received 34 stream ciphers suited to at least one of the profiles; (i) Profile I for software applications with high throughput requirements and (ii) Profile II for hardware applications with restricted resources. The timeline of

the project is given in Table 1.1

Table 1.1: Timeline of the eSTREAM Project

Date	Event
November 2004	eSTREAM Call for Primitives
April 2004	34 Submissions
February 2006	End of Phase I
July 2006	Beginning of Phase II
April 2007	Beginning of Phase III
May 2008	Final report of eSTREAM

1.1 Motivation

In the design proposals, apart from the details of the cipher, authors are expected to show that the proposal is resistant to all previously known attacks. After analyzing the specifications of eSTREAM candidates, we observed that limited theoretical security proofs were provided by most of the authors. This is not surprising. Proving security is extremely hard for most ad hoc designs, unlike the primitives of public key cryptography whose security is based on hard mathematical problems like integer factorization. For most designs, it is even hard to provide tight theoretical bounds for important properties of keystream such as period, linear complexity *etc.*

In this thesis, we focus on the *statistical analysis of synchronous stream ciphers* and give a framework for black box statistical testing. These tests play an important role in the design of cryptosystems, since in some cases, black box distinguishers are able to detect weaknesses that are hard to detect by theoretical analysis. Additionally, most of them have very low complexity and provide results in a very short time. Although black box distinguishers do not consider the inner structure of ciphers, to increase the success rate of the distinguishers, some properties of the cipher can be used as input to the distinguisher. For example, the word length in word oriented stream ciphers can be selected as block length of 2-level statistical tests. If a cipher fails any of the proposed tests, it is for certain that the design should be reevaluated and necessary changes should be applied. If a cipher is resistant to all the statistical tests, this increases the confidence on the cipher.

Firstly, considering stream ciphers as random number generators, we focus on testing the quality of the output keystream by statistical randomness tests. After defining new randomness tests based on random walks, we emphasize the importance of indepen-

dence of randomness tests in test suites and present some theoretical and experimental results on the relations of some commonly used tests. We also define the concept of *sensitivity*, where we analyze the effect of simple transformations on test results. We propose to add the composition of transformation and the test to randomness test suites to increase the coverage of the suite, if the transformation significantly changes the output p-values.

We also propose a number of statistical tests that are based on the classical attacks against stream ciphers. We focused on the ideas of algebraic, time-memory tradeoff (TMTO) and correlation attacks to design the tests. Most of the proposed tests can directly be used to distinguish output of the ciphers from ideal distributions. Distinguishers are important since they are suitable to detect the cipher used during communication, this has significant importance especially in military applications. Also, it is possible to extend some of the tests to recover the secret key.

Proposed tests mainly evaluate two security principles, *confusion* and *diffusion*, defined by Shannon in [17]. For synchronous stream ciphers, to satisfy diffusion each bit of keystream should depend on each bit of Initialization Vector (IV) and key and minor changes in the IV and key should result in random looking changes in the keystream. To satisfy confusion, the relation between IV, key and keystream should be too complex to be exploited by the attacker. Most attacks against cryptosystems are due to weaknesses in confusion and diffusion.

1.1.1 Outline of the Thesis

Chapter 2 (*Brief Overview of Stream Ciphers*) gives the basics of synchronous stream ciphers focusing on the desired properties of keystream and common building blocks. Moreover, widely used design approaches and different attack scenarios are presented.

Chapter 3 (*Analysis of Keystream*) focuses on the randomness properties of the output keystream and proposes three new randomness tests based on one dimensional random walks namely (i) excursion, (ii) height and (iii) expansion tests. Moreover, a classification of randomness tests is given as tests based on k-tuple pattern frequencies and tests based on ordering of k-tuples patterns. Some experimental results on independence of randomness tests are presented for short sequences. Also, the concept of *sensitivity* is defined for randomness tests to observe the effect of some transformations on the output p-values. The results of the chapter are published in [18, 19].

Chapter 4 (*Tests Based on Algebraic Properties*) aims to analyze stream ciphers based on the cryptographic properties of Boolean functions that input key and ini-

tial nonce and produce a particular keystream bit. These properties are also used in algebraic attacks against stream ciphers. First, we study some classical designs and experimentally observe the distribution of some cryptographic properties such as nonlinearity, number of linear terms, degree *etc.* for small sized ciphers. Then, we present a case study for eSTREAM finalist Trivium and obtain a linear approximation for F_1 with bias 2^{-31} using 288 initial clocking. To analyze the completeness properties of the ciphers, we apply a *diffusion test* that measures the effect of each key and IV bit on keystream bits. Moreover, we propose some new chosen IV distinguishing tests based on the d-monomial approach described by Filiol [20]. Also, we introduce another distinguisher *linear span* test that measures the inheritance of linear dependence of input IVs to keystream bits and apply the test to all Phase III candidates of the eSTREAM project. The results of the chapter are published in [21, 22, 23].

Chapter 5 (*Tests Based on Random Mappings*) aims to analyze stream ciphers based on some properties of random mappings. By focusing on different TMTO attacks, we try to find suitable test statistics. First, we consider the coverage of mappings generated using a subset of IV bits, then we analyze the index of the first repetition when the random mapping is iteratively applied. Finally, we consider the distinguished point method against stream ciphers and analyze the coverage properties of random mappings that are followed by a special keystream pattern. Using these test statistics, we propose three new distinguishers namely (i) coverage test, (ii) ρ-test and (iii) distinguished point coverage test and apply these tests to the Phase III Candidates of the eSTREAM project. The results of the chapter are published in [24].

Chapter 6 (*Tests Based on Correlations*) focuses on the correlations of key, input nonce and output keystream. We introduce three tests namely (i) *key/keystream correlation*, (ii) *IV/keystream correlation* and (iii) *frame correlation* and apply them to all Phase I candidates of eSTREAM project. The results of the chapter are published in [23].

Finally, Chapter 7 (*Conclusion*) summarizes the contributions of this study and suggests future research directions from the results.

Chapter 2

Brief Overview of Synchronous Stream Ciphers

In this chapter, we present a brief overview of synchronous stream ciphers, focusing on basic building blocks, the desired properties of keystream, common design approaches and distinguishing attacks.

A *synchronous stream cipher* is a finite state machine in which the output keystream z_t is produced independent of plaintext, using following equations;

$$\begin{aligned} \sigma_0 &= f_{init}(K, IV), \\ \sigma_{t+1} &= g(\sigma_t, K, IV), \\ z_t &= f(\sigma_t, K, IV), \\ c_t &= h(z_t, m_t), \end{aligned}$$

where σ_t is the *internal state* at time t, f_{init} is the *initialization function* that inputs k-bit secret key K and v-bit public IV and outputs the secret initial state of σ_0, g is the *next state function* that updates the internal state, f is the *keystream generation function* that produces keystream z_t and h is the *encryption function* that inputs *plaintext* m_t and keystream bits z_t to produces *ciphertext* c_t as given in Figure 2.1.

The synchronous stream ciphers using bitwise XOR as the encryption function h, are called *binary additive stream ciphers*. For binary additive stream ciphers, encryption is given as

$$c_t = z_t \oplus m_t, \tag{2.0.2}$$

and similarly decryption is given as

$$m_t = z_t \oplus c_t, \tag{2.0.3}$$

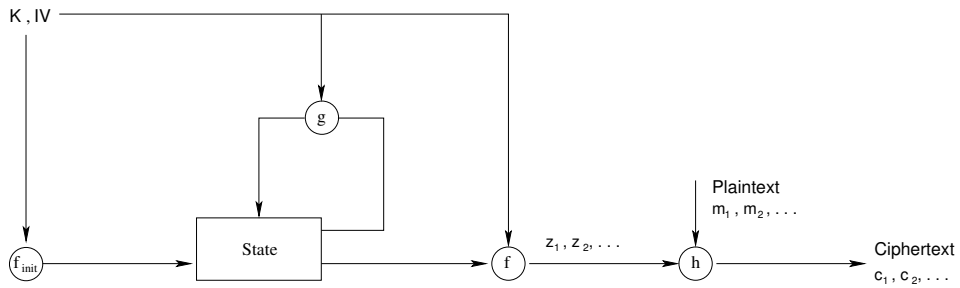


Figure 2.1: Generic structure of a synchronous stream cipher

as given in Figure 2.2. For correct decryption, sender and receiver must be synchronized, *i.e* must have the same internal state at the same time. Whenever the synchronization is lost, techniques for re-initialization should be employed.

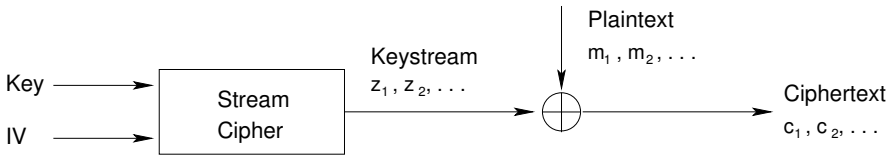


Figure 2.2: Binary additive stream cipher

2.1 Basic Building Blocks

The internal structure of stream ciphers varies extensively, however there are some very commonly used building blocks. In the following sections, brief background information about Feedback Shift Registers (FSRs), Boolean functions, S-boxes and t-functions are presented.

2.1.1 Feedback Shift Registers

FSRs are the most widely used building blocks of synchronous stream ciphers due to their large period, good statistical properties, and efficiency in hardware implementations.

A *FSR* is a device that shifts its contents into adjacent positions within the register and fills the position on the other end with a new value generated by *feedback polynomial*. The individual delay cells of the register are called the *stages* and the number of the stages n, is called the *length* of the FSR. The contents of the n stages are called the *state* of the FSR. The n bit vector $(x_0, x_1, \ldots, x_{n-1})$ initially loaded into FSR states specify the *initial conditions*. A block diagram of a FSR is given in Figure 2.3.

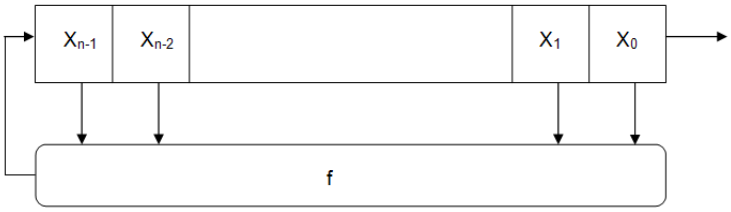


Figure 2.3: Block diagram of a FSR

The properties of a FSR are mainly based on the selection of its feedback polynomial

$$f(x_0, \ldots, x_{n-1}) = c_1 1 + c_{x_0} x_0 + c_{x_1} x_1 + c_{x_0 x_1} x_0 x_1 + \ldots + c_{x_0 x_1 \ldots x_n} x_0 x_1 \ldots x_n \quad (2.1.1)$$

where $c_i, x_i \in \mathbb{F}$. The output of the FSRs satisfy the following recursion,

$$x_{n+i} = f(x_i, \ldots, x_{n-1+i}), \ i \geq 0, \quad (2.1.2)$$

given $(x_0, \ldots, x_{n-1})$ as the initial condition.

There are total of 2^{2^n} different feedback polynomials of n variables. FSRs are divided into two groups depending of their feedback function. If the feedback function is linear, then the register is called *Linear Feedback Shift Register* (LFSR), otherwise it is called *Nonlinear Feedback Shift Register* (NFSR).

Linear Feedback Shift Registers

LFSRs are widely used as building blocks of stream ciphers due to their good statistical properties and long period of their output. LFSR based stream ciphers include A5, Snow [25] and E_0 [12]. A large amount of study on stream ciphers is related to the analysis of LFSR based stream ciphers [26, 27, 28] .

The output sequence **S**= $\{x_0, x_1, x_2, \ldots\}$ of a LFSR initialized by $(x_0, x_1, \ldots, x_{n-1})$ is uniquely determined by the following n^{th} order linear recursion equation

$$x_{n+j} = \sum_{i=1}^{n} c_i x_{n-i+j} \quad (2.1.3)$$

for $j \geq 0$ where c_i corresponds to the tap positions. Then, trivially following equation

is satisfied,

$$\sum_{i=0}^{n} c_i x_{n-i+j} = 0 \tag{2.1.4}$$

where $c_0 = -1$ and $j \geq 0$.

This recursion may be represented using either of the *feedback polynomial*

$$F(x) = \sum_{i=0}^{n} c_i x^i \tag{2.1.5}$$

and the *characteristic polynomial*,

$$C(x) = \sum_{i=0}^{n} c_i x^{n-i}. \tag{2.1.6}$$

Characteristic polynomial is the reciprocal of the feedback polynomial, *i.e.*

$$C(x) = x^n F(x^{-1}) \tag{2.1.7}$$

holds.

We can see that each state of a n-bit LFSR is a vector in the n-dimensional space. Then, a LFSR is a linear operator on n-dimensional vector space that changes current state into its successor vector according to the feedback function. The following $n \times n$ *state transition matrix* uniquely represents the LFSR,

$$\begin{pmatrix} 0 & 0 & 0 & \dots & 0 & c_0 \\ 1 & 0 & 0 & \dots & 0 & c_1 \\ 0 & 1 & 0 & \dots & 0 & c_2 \\ & & \vdots & & & \\ 0 & 0 & 0 & \dots & 1 & c_{n-1} \end{pmatrix}.$$

If $C(x)$ or $F(x)$ is a primitive polynomial with degree n, then each of the $2^n - 1$ non-zero initial states of the LFSR produce an output with maximum possible period $2^n - 1$. This sequence is called a *maximal length sequence* or *m-sequence*.

The maximal length sequences have good statistical properties and satisfy the randomness postulates of Golomb [27]. However, they are cryptographically insecure, since whenever $2n$ bits of the output of a n-bit LFSR is given, the sequence is totally deterministic using the Berlekamp-Massey algorithm. Therefore, many design attempts have been done to add nonlinearity to the systems based on LFSRs, such as combining outputs of several LFSRs by a nonlinear function, nonlinearly filtering the LFSR or

irregularly decimating the LFSR output.

For implementation purposes, choosing a LFSR with small number of nonzero coefficients is advantageous, but these LFSRs are vulnerable to algebraic and correlation attacks. Availability of a low-weight multiple of the feedback polynomial and bitwise linear approximations can be exploited [29]. Additionally, it is suggested to use a polynomial that satisfies full positive difference set to avoid guess and determine attacks.

Bit based LFSRs are very efficient in hardware, but inefficient in software. In order to increase the efficiency of software applications, the approach of using LFSRs on higher fields rather than $GF(2)$ is proposed in [25]. LFSRs over $GF(2^n)$ are mathematically equivalent to n parallel shift registers over $GF(2)$, each with same recurrence but different initial states.

Nonlinear Feedback Shift Registers

NFSRs are becoming more popular building blocks for stream ciphers. The eSTREAM candidates Grain [30], Achterbahn [31], Trivium [32], Vest [33] and Dragon [34] utilize NFSRs as building blocks. NFSRs are good tools to avoid algebraic attacks, since they do not have low linear complexity as LFSRs have.

By using n-bit NFSRs, it is possible to generate cyclic sequences in which each n bit pattern appear exactly once. These sequences are called *de Bruijn* and they have very good statistical properties with period 2^n. The number of such sequences is $2^{2^{n-1}-n}$ [27]. However, there is no efficient method to find feedback polynomials producing de Bruijn sequences, moreover given a feedback polynomial, it is hard to predict its period. Also, optimal selection of tap numbers, optimal degree are not known, only average period values for large NFSRs are estimated in [27].

In [35], the necessary conditions for feedback polynomials $f(x_0, \ldots, x_{n-1}) = c_1 1 + c_{x_0} x_0 + c_{x_1} x_1 + c_{x_0 x_1} x_0 x_1 + \ldots + c_{x_0 x_1 \ldots x_n} x_0 x_1 \ldots x_n$ to generate a maximum length output are given as follows;

1. The coefficient c_1 should be 1 to ensure that the state $(0, 0, \ldots, 0)$ is followed by the state $(1, 0, \ldots, 0)$.
2. The number of terms in the feedback polynomial should be even, to ensure that the state $(1, 1, \ldots, 1)$ is followed by state $(0, 1, \ldots, 1)$.
3. The function f should be in the form $f(x_0, \ldots, x_{n-1}) = x_0 + f_1(x_1, \ldots, x_{n-1})$ to ensure that the function has one predecessor.
4. There should at least be one $c_{x_i} = 0$ for $i = 1, \ldots, n-1$.

Also, the feedback function f should contain at least 4 tap positions [27]. Two nonlinear functions with 4 tap positions are extensively studied in [27] and the following important theorem is stated.

Theorem 2.1.1. *All of the tap positions of a NFSR should be used in the feedback polynomial to generate a de Bruijn sequence.*

To obtain outputs with period $2^n - 1$, this condition is not necessary. Therefore, it is more advantageous to search for feedback polynomials with output period $2^n - 1$. It is possible to convert a feedback polynomial of an LFSR with period $2^n - 1$ to a feedback polynomial that generates a de Bruijn sequence by

$$F_{new} = F_{old} + c'_1 c'_2 \dots c'_{n-1}, \tag{2.1.8}$$

where c'_i is the complement of c_i.

Another theorem in [27] states that the distribution of cycle lengths, for fixed initial vector and variable feedback logic is uniform for all lengths between 1 and 2^n.

In [36], two different types of NFSRs are considered. In Type I, the feedback is added to arbitrary number of stages, whereas in Type II the feedback is added only to the first stage. Massey and Liu showed that for every Type-I register, there exists an equivalent Type-II register, that generates same keystream with the only difference in the labeling of the states.

The output of a NFSR cannot be directly used as a keystream generator. All LFSR models such as nonlinear combining, nonlinear filtering or decimation are applicable to NFSRs. In Achterbahn [31], maximum length NFSRs with sizes between 21 and 31 are combined using a nonlinear Boolean function. Due to their small size, the maximum length NFSRs are found by exhaustive search in the design of Achterbahn. As an alternative approach, it is possible to use larger NFSRs with unknown period. To avoid short cycles, the register may additionally fed by a maximum length LFSR as done in Grain [30].

Feedback Shift Registers with Carry or Memory

Feedback shift registers with carry (FCSR), another variant of FSRs, are introduced by Klapper and Goresky [37]. They can be considered as LFSRs with memory. The periodicity analysis, exponential representation, description of maximal period FCSR sequences, and parallel implementation architecture are presented in [38]. The eSTREAM candidate F-FCSR use FCSRs as building blocks.

Jump Registers

Irregular clocking of LFSRs removes most of the disadvantages of LFSRs such as linearity and therefore provides more security against many attacks such as correlation attacks. However, generating one bit of keystream after multiple clockings makes the cipher inefficient and also, resistance to side channel attacks such as timing and power attacks decreases. Jump registers are introduced as an alternative way to do irregular clocking in [39]. Jump registers allow transition from one step to another without going through all the intermediate states. The stream cipher Pomaranch [40] uses nine 14-bit jump registers as its main components.

2.1.2 Boolean Functions and S-boxes

Boolean functions play a crucial role in design of symmetric ciphers. A *Boolean function* f with n variables is a mapping from $\mathbb{F}_2^n$ into $\mathbb{F}_2$. Boolean functions can be represented using various forms such as *truth table, algebraic normal form (ANF) and numerical normal form* [41].

Let α_i be the n-bit vector corresponding to the binary representation of integers $i = 0, 1, 2, \ldots, 2^n - 1$. For a Boolean function with n variables, the sequence

$$(f_{(\alpha_0)}, f_{(\alpha_1)}, \ldots, f_{(\alpha_{2^n-1})}) \tag{2.1.9}$$

is called the *truth table* of f.

The *ANF* of a Boolean function is the polynomial

$$f(x_1, x_2, \ldots, x_n) = a_0 \oplus a_1 x_1 \oplus \ldots \oplus a_n x_n \oplus a_{12} x_1 x_2 \oplus \ldots \oplus a_{12\ldots n} x_1 x_2 \ldots x_n \tag{2.1.10}$$

with unique $a_{i_1 \ldots i_k}$'s in $\mathbb{F}_2$. The number of terms in the highest order product term with nonzero coefficient is called the *algebraic degree*, or simply the degree of f. The Boolean functions with degree 1 are called *affine* and in particular for $a_0 = 0$, the functions are called *linear*.

For a Boolean function to be cryptographically secure, it should satisfy various conflicting criteria such as being balanced, having high nonlinearity, high algebraic degree, high algebraic immunity and high correlation-immunity. For balancedness, the weight of the truth table, that is the number of ones in the truth table, should be 2^{n-1}. Nonlinearity of a Boolean function is the minimum distance of f to the set of all affine functions. For f to be m^{th} order correlation-immune, $f(x_1, \ldots, x_n)$ should be statistically independent of $(x_{i_1}, x_{i_2}, \ldots, x_{i_m})$ for every choice of $i_1, i_2, \ldots, i_m$. Balanced

m^{th} *order correlation immune* functions are called *m-resilient*. A Boolean function f with n variables satisfies the *Strict Avalanche Criteria*, if

$$f(x) \oplus f(x + \alpha) \tag{2.1.11}$$

is balanced for every $\alpha \in \mathbb{F}_2^n$ with weight 1. A statistical approach on the number of functions satisfying SAC is given in [42].

Theorem 2.1.2. *[43] For a n-bit function having m^{th} order correlation immunity, the degree of f is at most $n - m$, and for balanced functions the bound is $n - m + 1$. The maximum algebraic degree of a balanced function is $n - 1$.*

The number of n variable Boolean functions is 2^{2^n}, therefore exhaustive search for functions with good cryptographic properties is infeasible even for small values of $n(\geq 6)$.

A (n, m) Substitution-box (S-box) is a vectorial Boolean function F from $\mathbb{F}_2^n$ into $\mathbb{F}_2^m$ represented by m-tuple $(f_1, \ldots, f_m)$ Boolean functions f_i on $\mathbb{F}_2^n$. S-boxes are commonly used in block cipher designs, but also suitable for stream ciphers. Sober [44] uses 8×32 S-box which is a combination of the S-box of block cipher Skipjack and the S-box designed by the Information Security Research Centre. Snow [25] uses the S-box of Rijndael and Dragon [34] uses two 8×32 optimized S-box. To be more efficient, S-boxes are usually implemented as lookup tables.

2.1.3 T-functions

T (triangular)-functions are functions from n-bit input to n-bit output which do not propagate from left to right. Klimov and Shamir [45] showed that *t-functions* can be used as efficient building blocks for stream ciphers, especially as nonlinear maximum length state transition functions. Mir-1 [46], ABC [47], TSC-4 [48] use t-functions as building blocks. T-functions are fast in software oriented and dedicated hardware systems. They are more resistant to algebraic attacks, however their security is not well studied.

Some examples of t-functions are addition ($x + y \bmod 2^n$), subtraction ($x - y \bmod 2^n$), negation ($-x \bmod 2^n$), multiplication ($x \cdot y \bmod 2^n$), bit-wise complementation ($\overline{x}$), xor ($x \oplus y$), and ($\wedge$) and or ($\vee$) operations. Also, combinations of t-functions are also t-functions.

For a t-function to be used in a stream cipher design, it should be invertible and should have a single cycle property. An example of a single cycle t-function is given in

[45] as

$$f(x) = x + (x^2 \vee 5). \tag{2.1.12}$$

2.2 Desired Properties of Keystream

In most attacks against stream ciphers, the plaintext (so is the keystream) is assumed to be known. In this section, we first describe the perfectly secure One Time Pad (OTP), then present some of the desired properties of keystream like long period, randomness and high complexity measures.

One Time Pad

The basic design philosophy of synchronous stream ciphers is inspired by the OTP, invented by Vernam in 1917. OTP encrypts the plaintext with a random key using the XOR operation as follows;

$$c_i = m_i \oplus k_i \tag{2.2.1}$$

where m_i, k_i and c_i corresponds to the i^{th} bit of message, key and ciphertext, respectively.

Shannon [17] proved that OTP is *perfectly secure* provided that (i) key is truly random, (ii) the lengths of the key and plaintext are same, and (iii) the key is used only once. In perfectly secure ciphers, ciphertext provides no information about (even statistically weak) plaintext. Mathematically, conditional entropy of plaintext given ciphertext $H(M|C)$ is equal to the entropy of the plaintext $H(M)$.

The need for the key to be as long as the plaintext and distributing it securely in advance makes OTP impractical. As an alternative way of designing ciphers, *conditional security* concept is defined. A system is conditionally secure if breaking the system requires very high computing power compared to realistic attacks. As an approximation to the action of OTP, stream ciphers are developed with the motivation of generating a long pseudo-random keystream from a short random key to overcome the disadvantages of OTP. Although stream ciphers are practical, they are not able to provide the theoretical security of OTP.

Existence of any successful distinguisher of keystream from truly random sequences is a threat to the security of the ciphers. In this section, we present some desired cryptographic properties that should be satisfied by the keystream.

2.2.1 Period

The first of these properties is about the *period* of the keystream. For a sequence $\{s_i\}$, if there exists an integer $p > 0$ and $u > 0$ such that

$$s_{i+p} = s_i, \tag{2.2.2}$$

for all $i > u$, then the sequence is called *ultimately periodic*. The smallest p that satisfies the equation is called the *period* of the sequence.

Synchronous stream ciphers have finite internal states and it is clear that some of the internal states will occur twice after a number of clocking. Re-occurrences of the same internal state result in producing same keystream, *i.e.* encrypting different messages using identical keystream. This obviously violates one assumption of OTP that key is not used twice. In stream ciphers, to avoid keystream reusage, theoretical lower bounds of period for each key and IV pair should be given.

For a stream cipher with n bit internal state, the period of the keystream is bounded by 2^n. If the cipher is using a random next bit function, the keystream is expected to repeat itself after $2^{n/2}$ clockings. If this function is a permutation, number of expected clockings increase to 2^{n-1}. In the design of stream ciphers, estimating the period of the keystream is very important. It should be very unlikely to use the same portion of the keystream twice during encryption. The required period length depends on the applications and open to debate. For current applications, the period of a practical stream cipher should at least be 2^{64}.

2.2.2 Randomness

Randomness is another important criterion of keystream and existence of any statistical deviation from truly random sequences may be exploited to attack the cipher. For infinite sequences, basic properties of random numbers can be given as follows.

Unpredictability: Given the first t bits of the sequence $S = s_1, s_2, \ldots, s_t$, it should be infeasible to predict the next output bit s_{t+1} with probability statistically greater than $1/2$ without the knowledge of the secret seed. This property is called the *forward predictability*, similarly, *backward predictability* is also required unpredictability.

Uniformity: The distribution of 0's and 1's should be uniform throughout the sequence. For each randomly taken subsequence from the entire sequence, the probability of getting 1 or 0 should be equal.

Independence: For each i, j $i \neq j$, the probability that s_i is equal to s_j should be

equal to 1/2.

When we consider finite sequences, it is not easy to define randomness. This is mainly because the probability of generating a particular sequence of length n is equal to 2^{-n} using a binary random generator, therefore accepting or rejecting randomness of these finite sequences does not seem to be reasonable. Kolmogorov defined randomness of finite sequences as the length of the shortest description of a generation rule for the sequence [49]. According to this definition, a sequence is random if one of the shortest descriptions is the sequence itself. However, determining the shortest description for sequences is computationally infeasible, therefore cannot be used to test randomness.

Golomb's Postulates

Golomb [27] proposed three postulates for the appearance of periodic pseudo-random sequences as followings;

- Postulate 1 requires the sequence to be balanced. The difference between the number of 1's and the number of 0's must be at most 1.
- Postulate 2 is about the number of runs observed throughout the sequence. A run is defined as a subsequence that consists of consecutive 0's or 1's that is not preceded or succeeded by the same symbol. Half of the runs of the sequence must have length 1, one fourth of runs must have length 2, one eighth must have length 3, *etc.* Moreover, there must be equally many runs of 1's and of 0's for each of these lengths.
- Postulate 3 is related to the auto-correlation function of the sequence which is defined as follows.

  Definition 2.2.1. Let $S = s_0, s_1, \ldots$ be a sequence with period p. The auto-correlation function of S is an integer valued function $C(j)$ defined as

  $$C(j) = \frac{1}{p}\sum_{i=0}^{p-1}(2s_i - 1)(2s_{i+j} - 1) \tag{2.2.3}$$

  for $0 \leq j \leq p-1$.

  The auto-correlation function that gives the similarity of the sequence and its shifted version, and should be two-valued.

A sequence that satisfies these three postulates is called *G-random*. The output of LFSRs with primitive feedback function satisfies these postulates. Therefore, it is clear that these three postulates are not sufficient to describe random looking sequences.

Generating high quality random numbers is a very serious problem and also very difficult using deterministic methods. The best way to generate unpredictable random numbers is to use physical processes such as radioactive decay, thermal noise or sound samples from a noisy environment, since the output sequences of truly random generators are (i) not periodic, (ii) not based on an algorithm, (iii) not predictable and (iv) involve no inner correlation. However, generating random numbers using these physical methods is extremely inefficient. Therefore, most systems use Pseudo Random Number Generators (PRNGs) based on deterministic algorithms. The unpredictability of random sequence is established using a random *seed*, that is obtained by a physical source like timings of keystrokes. Without the seed, the attacker must not be able to make any predictions about the output bits, knowing the details of the design. Desired properties of PRNGs are (i) good randomness properties of the output sequence, (ii) reproducibility, (iii) speed or efficiency and (iv) large period. Reproducibility is especially necessary for simulation applications. Some of the PRNGs used in the literature are given below.

Linear Congruential Generators produce a pseudorandom sequence $x_1, x_2, \ldots$ according to the given linear recursion

$$x_n = ax_{n-1} + b \pmod{m} \tag{2.2.4}$$

where a,b and m are parameters and x_0 is the seed. The size of the modulus gives an upper bound on the period of the sequence. Since this generator is easily predictable, it is not suitable for cryptographic applications.

Shift Register Generators produce a sequence according to the given recursion

$$x_{n+k} = \sum_{i=0}^{k-1} a_i x_{n+i} \pmod{2} \tag{2.2.5}$$

where a_i values are parameters and x_i values are seed for $0 \leq i \leq k-1$. This generator is very effectively implemented in hardware and is able to generate sequences with good statistical properties. However, neither this generator is suitable for cryptographic applications.

Additive Lagged-Fibonacci Generators produce a pseudorandom sequence according to the given equation

$$x_n = x_{n-j} + x_{n-k} \pmod{2^k}. \tag{2.2.6}$$

The quality of the generator highly depends on the initial conditions, again due to predictability these generators are not suitable for cryptographic applications.

Some of the commonly used PRNGs for cryptographic applications are ANSI X9.17 generator, PGP 2.x generator, SSH generator and Applied Cryptography generator [50], generator based on elliptic curves [51] and biometric random number [52]. Stream ciphers can be considered as PRNGs and output keystream should behave randomly to anyone who does not know the secret key.

2.2.3 Complexity Measures

As mentioned in previous section, unpredictability of keystream is very essential for stream ciphers. There are various complexity measures to evaluate the unpredictability of output keystream such as *linear complexity*, *k-error linear complexity, maximum order complexity, quadratic complexity, 2-adic span, Lempel-Ziv complexity etc.* Keystream should have complexity measures approximately equal to that of a random sequence, otherwise it is possible to predict the keystream or even to recover the secret key.

Linear Complexity

The most important and widely used complexity measure is the *linear complexity*, which is determined by the length of the shortest LFSR (See Section 2.1.1) that produces the sequence. Mathematically, *linear complexity* of an infinite binary sequence $\Lambda(S)$, $S = s_1, s_2, \ldots$ is defined as the shortest length t recurrence

$$s_{n+t} = c_{t-1}s_{n+t-1} + c_{t-2}s_{n+t-2} + \ldots + c_0 s_n, \ n \geq 0. \tag{2.2.7}$$

satisfied by the sequence, where $c_i \in \mathbb{F}_2$ for $0 \leq i \leq t-1$. Using the Berlekamp-Massey algorithm [53], it is possible to calculate the linear recurrence satisfied by a sequence with linear complexity t after observing $2t$ bits of the sequence.

Linear complexity of a zero sequence is defined to be 0. If no LFSR generates S, then the linear complexity of S is defined to be ∞. As an example, the linear complexity of the sequence $10100100010000\ldots$ is ∞. Every periodic sequence can be generated using a cycling LFSR with length equal to the period of the keystream, thus the linear complexity of a periodic sequence is bounded by its period.

For a random sequence of length n, the expected value of the linear complexity [26] is

$$\mu = \frac{n}{2} + \frac{(9 + (-1)^{n+1})}{36} - \frac{\frac{n}{3} + \frac{2}{9}}{2^n}. \tag{2.2.8}$$

As the length of the sequence increases, linear complexity is approximately $n/2 + 2/9$

for even n, and $n/2 + 5/18$ for odd n and the variance of the linear complexity is approximately 86/81 [26]. For a sequence with period p, the expected linear complexity of the sequence is $p - 1 + 2^{-p}$. In [54], the distinction between periodic linear complexity -the length of the shortest LFSR that periodically generates the sequence- and aperiodic linear complexity -the length of the shortest LFSR that generates the sequence followed by arbitrary bits- is emphasized. Since it is not allowed to use stream cipher output longer than its period, it is claimed that aperiodic linear complexity is a better measure for predictability. Therefore, aperiodic linear complexity should be large for all subsequences of the output keystream. This has been also pointed out by Rueppel [26] who introduced the usage of the linear complexity profile for the analysis of stream ciphers and showed that the linear complexity of a random sequence follows the $y = x/2$ line.

Another complexity measure related to linear complexity is the *k-error Linear Complexity* denoted by $\Lambda_k(S)$. It is defined as the smallest linear complexity that can be obtained by changing k or less number of bits of the sequence. Trivially, when k is taken as zero, k-error complexity is equal to the linear complexity of the sequence, that is $\Lambda(S) = \Lambda_0(S)$. The definition is originally given for periodic sequences, however can also be defined for finite sequences. In [55], an efficient algorithm to compute k-error linear complexity of a periodic sequence having period a power of 2 is given.

Let $N_{n,k}(c)$ be the number of sequences of length n having k-error linear complexity c. It is an open question to find an algorithm that efficiently computes the k-error linear complexity spectrum of a binary sequence of an arbitrary period [56].

Theorem 2.2.2. *[57] For $k \geq 1$,*

$$N_{n,k}(0) = \sum_{t=0}^{k} \binom{n}{t} \text{ for } k \leq n. \tag{2.2.9a}$$

$$N_{n,k}(1) = \sum_{t=0}^{k} \binom{n}{t} + \binom{n-1}{k} \text{ for } k < (n-1)/4. \tag{2.2.9b}$$

$$N_{n,k}(n) = 0, k \leq n. \tag{2.2.9c}$$

Theorem 2.2.3. *[58] $N_{n,k}(c) = 0$, for $\lfloor \frac{n}{2} \rfloor \leq k \leq n$ and $2 \leq c \leq n$.*

Maximum Order Complexity

Maximum Order Complexity (MOC) is the length of the shortest FSR that generates the sequence, the feedback is polynomial no longer required to be linear as in linear complexity. MOC of a sequence can be calculated using a directed acyclic word graph

or suffix trees. The detailed analysis of MOC is given in [59]. In [60], an approximate probability distribution of MOC is given for random sequences. For a sequence of length n, the expected value of MOC is approximately $2log_2 n$, when n is large. One obvious relationship between linear complexity and MOC is that MOC is always less than or equal to the linear complexity of a sequence. It should be noted that sequences with very large linear complexity might have much smaller MOC values.

Quadratic Complexity

The *Quadratic Complexity* concept lies between linear and maximum order complexity, here the feedback polynomial is restricted to linear and quadratic terms. The expected value of quadratic complexity is approximately $\sqrt{2n}$, for sequences of length n [61]. Quadratic complexity of de Bruijn sequences ($n \geq 3$) is bounded above by $2^n - \binom{n}{2}$-1 and below by $n+1$ [62], also it is conjectured that the lower bound is $n+2$ for $n > 3$. The quadratic span distribution for de Bruijn sequences for small n (< 7) is listed in [62], for large n the distribution is unknown.

2-adic span

2-adic span of a sequence is length of the smallest FCSR (See Section 2.1.1) that generates the sequence. An efficient method to find 2-adic span is presented in [37]. $2M + 2lg(M)$ bits are needed to find the 2-adic span of a sequence with complexity M. The 2-adic span complexity profile of random sequences approximately follows $n/2$ line, similar to the linear complexity [37].

Lempel-Ziv Complexity

The *Lempel-Ziv Complexity* concentrates on the speed of occurrences of new patterns into the sequence. For instance Lempel-Ziv complexity of 010101001001011 is 7, since different patterns observed in the sequence are 0|1|01|010|0100|10|11. The Lempel-Ziv complexity LZ of a n bit sequence is bounded as;

$$\left\lceil \frac{-1+\sqrt{1+8n}}{2} \right\rceil \leq LZ \leq \left\lceil \frac{2^{t+2}+n-2t-4}{t+1} \right\rceil \tag{2.2.10}$$

where $t = \max\{i \in N : (i-1)2^{i+1}+2 \leq n\}$ [63]. However the expected value of Lempel-Ziv complexity for an arbitrary length of sequence is unknown. In [63], a recurrence relation for Lempel-Ziv complexity is given.

2.3 Common Designs

While designing a stream cipher, *security* and *performance* are the two most important goals. Security is satisfied if there does not exist an algorithm that gives the secret key with complexity less than the complexity of exhaustive key search, 2^k. Therefore, key size determines the level of security. Given the level of security, the secondary goal is to optimize the performance which can be measured by speed, chip area or power consumption. It is a challenge to design fast and secure stream ciphers.

Other design goals can be given as *flexibility*, *scalability* and *simplicity*. It should be noted that these design goals are satisfied differently for hardware and software oriented stream ciphers. In hardware oriented systems, usually the memory space is restricted and power consumption is more important whereas in software oriented systems usage of higher memory and security level is possible. To satisfy different requirements, hardware and software oriented ciphers use different building blocks, as an example, bit oriented building blocks are more commonly used in hardware oriented systems, whereas software oriented systems use word-oriented building blocks. Also, initial key/IV loading of hardware oriented stream ciphers are usually simpler compared to software oriented systems.

Initialization and Keystream Generation

After initialization of a cipher, each key and IV bit should be diffused to all internal state bits. Additionally, the relation between key, IV and the internal state variables should be as complex as possible. While designing the initialization phases, target applications should be considered. In some applications such as in wireless communication, the synchronization between sender and receiver may be lost, frequently. To regain synchronization, the cipher should be reinitialized. Usually, instead of choosing a new secret key, the IV is updated publicly according to a protocol. In those applications that require frequent reinitializations, initialization should be as efficient as possible. Also, if possible, key and IV loading phases may be separated.

Next state function that updates the internal state should not produce short cycles that lead to small period, and also should be $1 - 1$.

Selection of Building Blocks

In stream cipher design, the most crucial decision is probably the selection of building blocks. Cipher should be designed as simple and compact as possible and only

necessary components should be included to the system. Each building block and its parameters should be carefully selected to satisfy its purpose. Also, the advantages and disadvantages of each building block should be carefully analyzed and the disadvantages of a building block should be compensated with other components. Clearly, the relation between selected building blocks should be examined, since the interaction between secure building blocks may lead to a security flaw.

In Table 2.1, the building blocks of Phase III candidates of eSTREAM are listed. As seen from the table, more than half of the designs are based on FSRs. In the following sections, common design approaches based on FSRs, block ciphers, hash functions and NP hard problems are summarized.

2.3.1 Designs Based on Feedback Shift Registers

In most stream cipher designs, LFSR with maximum period are used as building blocks. However, necessary precautions must be taken against the linearity of LFSRs. To increase the nonlinearity of the system, there are three general methods, (i) nonlinear combining generator, (ii) filter generator and (iii) clock controlled generators. A huge amount of literature [77, 26] on these methods are available.

Nonlinear Combining Generator

The first method of designing a stream cipher using LFSRs is the *nonlinear combining generator*. This structure consists of n primitive LFSRs each of length L_i that are combined using a nonlinear Boolean function f as illustrated in Figure 2.4. The Geffe and the summation generator [77] are examples of nonlinear combiners.

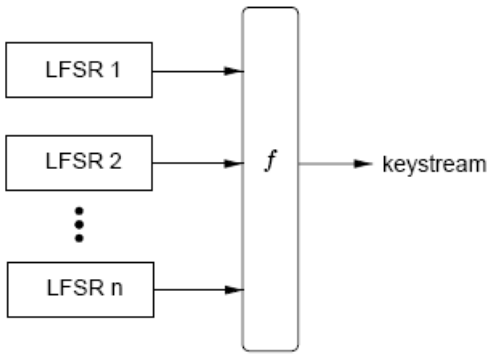


Figure 2.4: Nonlinear combining generator

The period of the system p is given as the least common multiple (lcm) of the

Table 2.1: Building blocks and proposed platforms of Phase III eSTREAM Candidates

Cipher	Platform	Building Blocks
Crypt-MT v.3 [64]	Software	Linear generator Filter with memory
Dragon [34]	Software	NFSR of length 1024 bits $F : \mathbb{F}_2^{192} \rightarrow \mathbb{F}_2^{192}$ 64 bit memory
Decim v2[65]	Hardware	LFSR of length 192 13 variable Boolean function ABSG decimation 32-bit buffer
Decim-128 [66]	Hardware	LFSR of length 288 14 variable Boolean function ABSG decimation 64-bit buffer
Lex [67]	Software	Block cipher AES
Grain [30]	Hardware	NFSR of length 128 LFSR of length 128 9 variable Boolean function
Trivium [32]	Hardware	3 NFSRs of lengths 93, 84, 111 Linear filter function
HC-256 [68]	Software	2 tables with 1024 32-bit integers Nonlinear Filter 32-bit-to-32 bit mapping Linear masking
NLSv2 [69]	Software	NFSR of length 544 Nonlinear Filter 16 bit Counter
Rabbit [70]	Software	8 32-bit state variables 8 32-bit counters Counter carry bit
Salsa20 [71]	Software	Hash function
Sosemanuk [72]	Software	Word oriented LFSR of length 10 Finite State Machine Serpent block cipher
Edon80 [73]	Hardware	80 e-transformers 4 Quasigroups
F-FCSR [74]	Hardware	FCSR Filtering
Mickey [75]	Hardware	2 Registers Irregular clocking
Moustique [76]	Hardware	Conditional complementing shift register Pipelined stages
Pomaranch [40]	Hardware	5 Jump registers

periods of each LFSR, that is.

$$p = lcm(2^{L_1} - 1, 2^{L_2} - 1, \ldots, 2^{L_n} - 1). \tag{2.3.1}$$

Upper bound on the linear complexity of the keystream is $f(L_1, \ldots, L_k)$ where f is evaluated over the integers. This bound is attained whenever all LFSRs are primitive with coprime degrees [77].

The choice of f is very critical. To produce unbiased keystream, f should be balanced. Also, to obtain high linear complexity, the algebraic degree of f should be as large as possible. Additionally, to avoid correlation and distinguishing attacks, the correlation-immunity and nonlinearity of f should be high. However, there is a trade off between algebraic degree and correlation immunity order of a Boolean function. For efficiency reasons, the gate complexity of f should be low. Moreover, to resist decimation attacks, the period of the LFSR should be prime [78].

Nonlinear Filter Generator

This structure consist of an LFSR and a nonlinear Boolean function f which is used as a filter function as given in Figure 2.5. Knapsack generator is an example of nonlinear filter generators.

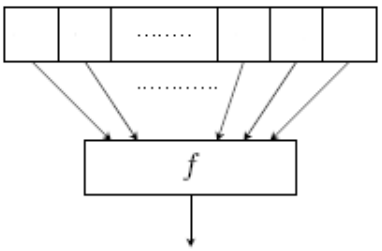


Figure 2.5: Nonlinear filtering generator

Period of the system is bounded by the period of the LFSR. Using a maximum length LFSR with length L and a filtering function of algebraic degree d, the linear complexity of the keystream is at most $\sum_{i=1}^{d} \binom{L}{i}$. Most of the Boolean functions attain the bound. According to [77], f should include many terms of each order up to the degree of f .

A complete study on the analysis of filter and combiner generators and their resis-

tance to most important cryptanalytic attacks is presented in [79].

For any filter generator, it is possible to find an equivalent combining generator such that the output keystreams are identical. Equivalent combining generator include n LFSRs used in the filter generator with shifted initial states and the combining function corresponds to the filtering function.

Clock Controlled Generators

Clock controlled generators introduce nonlinearity to the system by irregular clockings of the registers. Irregular clocking is good to resist correlation attacks. The main disadvantage of irregular clocking is decrease in efficiency and period. Also, these systems are vulnerable to timing and power attacks. To avoid these side channel attacks and to guarantee constant throughput, buffering mechanism is commonly used, as in Decim [80].

Alternative step generator is an example for clock controlled generators. In this design, three maximum length LFSRs R_1, R_2 and R_3 with lengths L_1, L_2 and L_3 are used. R_1 determines the clocking of R_2 and R_3 that generate the keystream. Suppose R_1 generates a de Bruijn sequence and L_2 and L_3 are relatively prime, then according to [77], the system has period

$$2^{L_1}(2^{L_2}-1)(2^{L_3}-1) \tag{2.3.2}$$

and the linear complexity of the output keystream L satisfies

$$(L_2+L_3)\cdot 2^{L_1-1} < L < (L_2+L_3)\cdot 2^{L_1}. \tag{2.3.3}$$

Shrinking generator [81], another important example for clock controlled generator, consists of two maximum length LFSRs R_1 and R_2 with lengths L_1 and L_2. The output of the first LFSR determines whether the output of the second LFSR is used as keystream. Thus, the keystream is irregular decimation output of the second LFSR. According to [77], provided that L_1 and L_2 are relatively prime, the keystream has period $(2^{L_2}-1)\cdot 2^{L_1-1}$ and the linear complexity L satisfies

$$L_2 \cdot 2^{L_1-2} < L < L_2 \cdot 2^{L_1-1}. \tag{2.3.4}$$

Self shrinking generator is a variant of the shrinking generator, in which only one LFSR is used. Any self-shrinking generator can be represented as a shrinking generator. The output of LFSR $(s_0, s_1, \ldots)$ is decimated so that s_{2i+1} is given as output if and

only if $s_{2i} = 1$. If a primitive LFSR of length l is used, the period of the shrinking generator is between $2^{\lfloor l/2 \rfloor}$ and 2^{l-1} [82]. A lower bound on the linear complexity of self-shrinking generator is given by half of the period and the upper bound is proven to be $2^{l-1} - (l-2)$ [83].

The stream cipher A5 is a real world example for stop and go clocking according to the tap bits of 3 LFSRs. Another example is the cipher LILI-128 [84], where the clocking of an LFSR is controlled by another LFSR that outputs the number of clockings. Various decimation mechanisms are available in the literature. The pseudocode of the decimation mechanism ABSG used in the stream cipher DECIM [65] is given in Algorithm 2.3.1.

Algorithm 2.3.1: ABSG(y_t)

$i \leftarrow 0$;
$j \leftarrow 0$;
Repeat
$\begin{cases} e = y_i,\ z_j = y_{i+1}; \\ i = i + 1; \\ \textit{while } (y_i = \bar{e})\ i = i + 1; \\ i = i + 1; \\ \textit{output } z_j; \\ j = j + 1; \end{cases}$

2.3.2 Designs Based on Block Ciphers

Block ciphers, as another class of symmetric-key encryption algorithms, transforms n-bit plaintext blocks to n-bit ciphertext blocks, with a user-provided secret key. Since they operate on fixed length blocks, there is a need for a technique on how to encrypt plaintext longer than n bits. Various modes of operations such as Electronic Codebook (ECB), Cipher Block Chaining (CBC), Output Feedback Mode (OFB), Cipher Feedback Mode (CFB), Counter Mode (CTR) are available in the literature [77].

The OFB and CTR modes enable the block cipher to be used as a synchronous stream cipher. These modes operate similarly, as they both encrypt sequence of blocks and generate keystreams by concatenating the ciphertexts. To encrypt the first block, a publicly known IV is chosen.

Main disadvantage of using block cipher mode of operations is the large number

of rounds and expensive operations of block cipher designs, this approach may not be suitable for applications that require high speed and low resource.

In OFB with r-bit feedback, r-bit of output keystream and a part of previous input sequence are used as input in the next encryption. When full feedback OFB is used, that is, $r = n$, the expected cycle length before repeating itself is around 2^{n-1}, assuming that the encryption is a random permutation among $(2^n)!$. For $r < n$, the expected cycle length reduces to $2^{n/2}$. Therefore, it is advised to use OFB with full n-bit feedback. In this mode of operation, it is not possible to parallelize the process, but it is possible to generate the output stream before the plaintext is available.

In CTR mode, the input sequences are generated incrementally using a counter. The period of the keystream can be controlled by the size of the counter, so this mode avoids short cycle problems. Also, it is possible parallelize the process.

Different from OFB and CTR, CFB mode enables the block cipher to be used as a self-synchronizing stream cipher. In this mode, the plaintext blocks also affect the next sequence to be encrypted.

The OFB and CTR modes of block cipher can be distinguished using $2^{n/2}$ blocks. According to the birthday paradox, a collision occurs in $2^{n/2}$ blocks with high probability. In the OFB mode, if a block occurs twice, then the remaining blocks should be repeated. Similarly, in the CTR mode, since it is not possible to observe a collision of blocks, it is easy to distinguish the output. However, no information about the key is revealed.

Alternatively, stream ciphers can be designed similar to block ciphers by including common building blocks of block ciphers such as S-boxes. The ciphers Phelix [85], Salsa20 [71] and Dragon [34] can be given as examples to such ciphers. The design of the stream cipher Sosemanuk [72] is combination of the design principles of the stream cipher Snow [25] and transformations derived from the block cipher Serpent [86].

eSTREAM candidate LEX (Leak EXtraction) [67], designed by Biryukov, is an AES-based stream cipher. The idea is to use some part of the internal state of the block cipher at certain rounds as keystream. In each AES round, 4 bytes of the internal state is used to generate keystream. This is similar to OFB mode, instead of using the ciphertext blocks are keystream, certain parts of the intermediate rounds are used. At each encryption, 320 bits (10 rounds $\times$ 4 bytes) are generated and this makes LEX 2.5 times faster than AES.

2.3.3 Designs Based on Hash Functions

Hash functions map an input with arbitrary length to an output of fixed length. For a hash function to be cryptographically secure, it should satisfy the following three properties;

- *one wayness:* It should be hard to find the input sequence x given the output $h(x)$.
- *second pre-image resistance:*Given x_1, it should be hard to find x_2 satisfying $h(x_1) = h(x_2)$.
- *collision resistance* It should be hard to find x_1 and x_2 with same hash value.

Hash functions can be used to design stream cipher. As one example, Salsa20 [71] that is composed of three main functions; *hash function*, *expansion function*, *encryption function* can be given. The hash function takes 64-byte input x and produces 64-byte output $Salsa20(x)$. The expansion function inputs 32 or 16 byte key K and 16-byte sequence n and produces 64-byte output $Salsa20_K(n)$. Finally, the encryption of an l-byte message m using key K, 8-byte nonce v is done as $Salsa20_K(v) \oplus m$. $Salsa20_K(v)$ produces a sequence of 2^{70} bytes as given below;

$$Salsa20_K(v, \underbrace{00\ldots0}_{64bits}), Salsa20_K(v, 00\ldots1), \ldots, Salsa20_K(v, \underbrace{11\ldots1}_{64bits}).$$

2.3.4 Designs Based on NP-Hard Problems

In public key cryptography, the general idea is to design ciphers based difficult problems that take very long time to solve. The systems utilize a private and a public key where the public key is used to encrypt messages and the private key is used for decryption. The security of these cryptosystems is based on the fact that the private key can be computed from the public key only by solving a NP-hard problem. As examples, RSA and Rabin cryptosystems based on integer factorization, and Diffie-Hellman, ElGamal systems based on discrete logarithm problem can be given.

The same idea may be used to design stream ciphers. However, generating fast stream ciphers using this approach is a challenge. The cipher QUAD, introduced by Berbain *et al.* [87], is based on the iteration of multivariate quadratic system of m equations and $n < m$ variables over a finite field $GF(q)$, typically $GF(2)$. The security of keystream generation of QUAD, is provably reduced to the conjectured intractability of solving multivariate system of quadratic equations. No efficient algorithm to solve

this problem with high success probability is known for large n (> 100) when the quadratic equations are random.

QUAD uses three fixed and publicly known multivariate system of quadratic equations, S, S_0 and S_1. $S = (Q_1, \ldots, Q_{kn})$ consists of kn randomly chosen equations with n variables over $GF(q)$. S_0 and S_1 are smaller systems with n equations and n variables. These two systems are only used during the initialization of the cipher where the internal state $(x_1, \ldots, x_n)$, n-tuple of $GF(q)$ values, is generated using key and IV.

Keystream generation phase of QUAD is iterative and in each iteration the following steps are performed;

- Compute $S(x) = (Q_1(x), \ldots, Q_{kn}(x))$ where x is the current internal state,
- Output $(Q_n(x), \ldots, Q_{kn}(x))$,
- Update the internal state as $x = (Q_1(x), \ldots, Q_n(x))$.

Analysis of different instances of QUAD(q, n, r) are given in [88], where q is the field size, n is the number of variables and r is the number of outputs per round. In particular, the instance of $QUAD(256, 20, 20)$ is broken using XL-Wiedemann in approximately 2^{66} cycles.

2.3.5 Other Designs

The internal structures of stream ciphers vary extensively and there are some ciphers that do not fit the previous categorizations. Some of these ciphers are given in this section.

- *RC4*, designed by Rivest [89], is commonly used in SSL and WEP applications. The internal state of the cipher is a permutation of values from 0 to 255 byte.
- *Rabbit*, designed by Boesgaard *et al.* [90] in 2003, utilizes eight 32-bit state variables, eight 32-bit counters and one counter carry bit with internal state size of 513 bits. The state variables are updated by eight nonlinear functions. The cipher is word oriented cipher and outputs 128 bit per clocking.
- *SEAL* (Software-Optimized Encryption Algorithm) [77] is a length-increasing pseudorandom function that transforms a 32-bit sequence number n to an L-bit keystream using a 160-bit secret key. The cipher requires large amount of precomputation in initialization that use the hash function SHA.

2.4 Cryptanalysis of Stream Ciphers

Cryptanalysis is the study of deciphering the encrypted message without knowing the secret key. A trivial attack to any cryptosystem is to try every possible key until the correct one is recovered. If the key size is k, attacker has to try 2^k keys in the worst case, and on the average 2^{k-1} keys. To detect the correct key, a plaintext/ciphertext pair is needed. However, if the plaintext contain redundancies, it may still be possible to recover the key using only the ciphertexts. One advantage of the attack is that it can be implemented in parallel. As the ability to compute increases, larger key sizes are needed. The size of the secret key should be large enough to avoid these exhaustive or brute force search attacks. In todays technology, 80-bit keys seem to offer acceptable level of security.

All other attacks applied to stream ciphers are compared to exhaustive key search in terms of *time, data and memory complexity*. *Time complexity* is related to the required number of primitive operations and can be separated into two parts, offline and online. The offline part is done once independent of the secret key and the results obtained in this part is used to attack the system in the online part. *Data complexity* considers the amount of data required for the attack, lastly, *memory complexity* is determined by the required amount of memory. The attack is considered to be successful if the maximum of time, data and memory complexity is less than the complexity of exhaustive search. Attacks with higher complexity are not be considered to be successful.

Different Attack Scenarios

- *Ciphertext Only Attacks:* In this type, the attacker aims to obtain information about the secret key or the plaintext by only observing the ciphertext. This kind of attack is successful if some information about the plaintext such as the language is available. Also, if plaintext is written using ASCII characters, it is very likely that the most significant bit of each character is equal to 0. In that case, ciphertext only attacks can be considered as known plaintext attacks.

- *Known Plaintext Attacks:* Most of the attacks available in the literature use this scenario. Since keystream is generated independent of plaintext, this type is equivalent to *known keystream attacks*. Well known chosen plaintext or chosen ciphertext attacks are not suitable to analyze synchronous stream ciphers since chosen plaintext attack is also equivalent to the known plaintext attack.

- *Known IV or Chosen IV Attacks:* In most of the protocols the value of IV is assumed to be public. In chosen IV attacks, the attacker is allowed to choose the

set of IVs and resynchronize the cipher using this set.

The attack models against symmetric cryptosystems are divided into two groups depending on the intention of the attacker. *Key recovery attacks* aim to recover the secret key whereas *distinguishing attacks* aim to find deviations of cipher output from ideal distribution. It is clear that distinguishing attacks are weaker compared to key recovery attacks that enable to decrypt any given encrypted message. However, distinguishing attacks are suitable to detect the cipher used during communication, this has significant importance especially in military applications. Moreover, in many cases, the idea of distinguishers may be improved to recover the key. It should also be noted that most of the candidates of the NESSIE project [14] were broken by distinguishing attacks and were not included in the final report of the project. In the next section, more information on distinguishing attacks is presented.

2.5 Distinguishing Attacks

Given a part of the ciphertext or keystream, *distinguishing attacks* make a decision; *the sequence is random* or *the sequence is generated using a specific cipher* via statistical hypothesis testing (See Appendix A.2). Existence of any successful distinguisher of keystream from truly random sequences violates the first requirement of OTP and is a thread to the security of stream ciphers. In some situations, it is possible to make incorrect conclusions such as claiming a truly random sequence as cipher output, or conversely claiming cipher output as a truly random sequence. A distinguisher is considered to be successful, if the probability of making correct decisions is significantly better than pure guessing.

Distinguishing attacks against stream ciphers usually assume a known plaintext scenario and aim to determine whether the given keystream $(z_1, z_2, \ldots)$ has been generated by a particular generator or by a truly random number generator as presented in Figure 2.6.

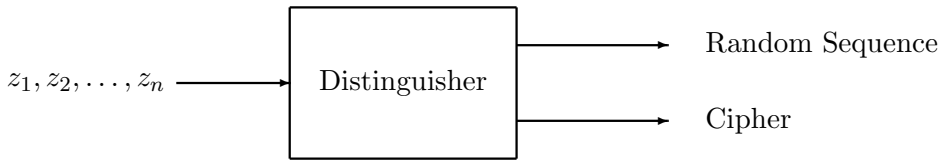


Figure 2.6: Distinguishing finite sequences

In cases where the message space $M_1, \ldots, M_l$ is small, distinguishing attacks can be used to recover encrypted message. Each message M_i is XORed with ciphertext C,

corresponding keystream Z is generated and if Z is distinguished as the cipher output, it is likely that M_i is the encrypted message (See Algorithm 2.5.1).

Distinguishing attacks generally use very long keystream and do not leak any information about the key and the internal state of the cipher. In [91], authors argue that weak distinguishing attacks that use large known plaintext do not represent a security problem in practice. However, it is usually possible to improve the ideas of these attacks to recover a part of secret key or initial state. One open discussion about distinguishing attacks is that there is no commonly accepted keystream length in terms of key size to accept the cipher as broken using the distinguishing attack. However, it is pointed out that the today's largest application use 2^{50} bytes and encrypting this amount of data using a single key and IV pair does not seem to be reasonable. According to [92], a synchronous stream cipher should not encrypt more than $2^{\frac{k}{2}}$ message blocks using the same secret key. To avoid distinguishing attacks that require long keystream, in some of the cipher specifications the size of the longest allowable keystream generated using a key and IV pair is specified, in Lex it is limited by $320 \times 500 = 160,000$ bits [67] wheres in Trivium the bound is 2^{64} bits [32].

Algorithm 2.5.1: DISTINGUISHER$(C; M_1, M_2, \ldots, M_l)$

$message$ = NONE;
for $i = 1$ **to** l
{
  $Z = C \oplus M_l$;
  Apply distinguisher to Z;
  if Z is distinguished as cipher output
  {
    $message = M_l$;
    $break$;
  }
return $(message)$

Chosen IV Distinguishers

Alternative to using very long keystream, it is also possible to distinguish the output of stream ciphers using a number of shorter keystream portions (called *frame*) generated using different IVs as given in Figure 2.7. Stream ciphers should also be resistant to distinguishers using chosen IVs, because in many wireless applications, instead of generating long keystream, the ciphers are frequently synchronized and short keystreams are generated. More information on resynchronization process is available

in Section B.1.

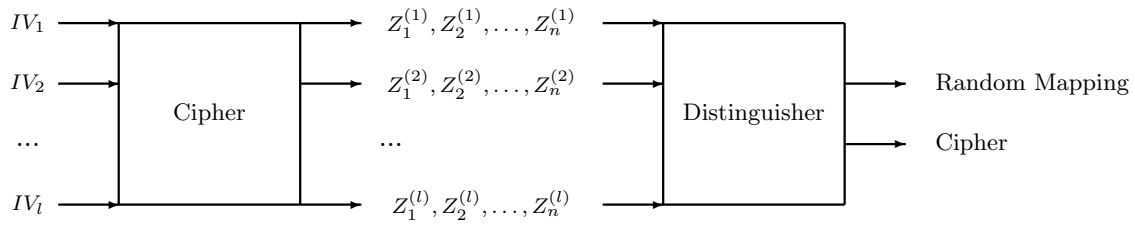


Figure 2.7: Distinguishing frames generated by different IVs

Black box and Cipher Specific Distinguishers

To attack a specific stream cipher using distinguishing attacks, the inner structure of the cipher should be analyzed extensively. In many cases, the distinguishers are successful due to the wrong selection of some parameters such as tab points of registers. As example, due to the choice of the weak feedback functions, a distinguishing attack against Grain is presented in [93].

In [94, 95], it is shown that for a binary keystream generator with M bit memory, there exists a linear function of at most $M + 1$ consecutive output bits which is an unbalanced function of the initial state variables. Also, an efficient method for finding distinguishers based on linear circuit approximation is presented. The linear function of these consecutive bits can be used to distinguish the keystream and the required keystream length is inversely proportional to the bias of the liner approximation. Some examples for the ciphers analyzed using this approach are Sober [96], Snow [97], Grain and Trivium [98].

Englund *et al.* [99] proposed an extension to the generic distinguisher against OFB mode of block ciphers. The attack is successful if the part of the state that depends on the both the key and the IV is smaller than twice the key size. It is shown that LEX is vulnerable to this distinguisher using approximately $2^{65.7}$ resynchronizations. Other examples of distinguishing attacks against stream ciphers are presented in [100, 101, 102].

Alternative to cipher specific distinguishers, some black box distinguishers that are independent of the inner structure of ciphers, can be used. Traditional statistical randomness tests such as frequency, runs, serial *etc.* [103] can be given as examples of distinguishers presented in Figure 2.6.

Approach of eSTREAM Candidates

Anashin *et al.* [47] applied the randomness tests in the NIST and DIEHARD test suite and observed no statistical deviation of keystream and they also proved that the distribution of 32-bit words in the keystream of ABC is uniform. In [104], Fubuki and AES have been tested according to their bit diffusion property with small number of rounds and it is claimed that there exist no diffusion bias for 4-round AES and 2-round Fubuki. In [34], the keystream of Dragon is tested by the statistical randomness tests given in Crypt-X suite. Authors applied the frequency, binary derivative, change point, subblock and runs tests to 30 keystreams of length 8 megabits. Additionally, the sequence and linear complexity tests were applied to 30 streams with 200 kilobits each. Dragon showed no deviation from randomness according to these results. Also, the output of the F-FCSR generator is tested using the NIST suite [74].

Theoretical validation for diffusion criteria in the initialization state has been done for Grain to defeat statistical chosen-IV attacks [30]. Wu [68] analyzed the keystream of HC-256 with no linear masking and weakened feedback function and concluded that distinguishing 2^{128} bits of keystream from a truly random sequence is computationally infeasible.

In [105], it is reported that the output of Hermes8 is tested using the FIBS 140-2 and DIEHARD Battery of Tests and no deviation from randomness is observed. Vuckovac [106] reported that the output of Mag has been tested for patterns in every stage of development by using statistical randomness tests available in ENT, DIEHARD and Crypt-X test suites and observed no statistical deviation. The cipher Py had also been tested using statistical randomness tests [107]. It is claimed that the output keystream is uncorrelated and statistical tests should not succeed even when more extensive tests are made.

For the cipher Rabbit, the statistical tests from NIST, DIEHARD and ENT suites was applied. The tests were done for both the internal state and the keystream [70]. Also, various statistical tests were applied to the key setup function and also to the reduced version of Rabbit where each state variable has been given in 8 bits. Authors did not find any statistical weakness in any of these cases. Hong *et al.* [108] reported that they had applied statistical randomness tests similar to the ones in NIST suite and had not found any weaknesses.

Bigeard *et al.* [33] tested the output of each component of Vest and claimed that individual streams of any of the outputs of Vest accumulators, combined Vest counters and complete Vest ciphers were indistinguishable from truly random sequences.

The randomness property of WG is analyzed in terms of high period, balance, two-level autocorrelation, t-tuple distribution and linear complexity [109]. The keystream generated using ZK-Crypt passed from the statistical randomness tests of NIST and DIEHARD [110].

As seen the different approaches, there is no general framework on how to test ciphers using block box distinguishers. Moreover, no statistical analysis is reported for the ciphers Achterbahn, Decim, Dicing, Edon80, Lex, Mickey, Mickey-128, Mir-1, NLS, Phelix, Polar Bear, Pomaranch, Salsa20, Sfinks, Sosemanuk, Trivium and Yamb in algorithm specification documents. In this thesis, we aim to provide a general framework to statistically analyze stream ciphers. First, we test all Phase I candidates of eSTREAM using the NIST test suite and the results are presented in Appendix C. Statistical weaknesses are observed in the keystream of Decim and Frogbit.

Chapter 3

Analysis of Keystream

Random numbers are widely used in many applications such as statistical sampling, Monte Carlo simulation, numerical analysis, game theory, cryptography *etc.* In cryptography, the need for random numbers arises in key generation, authentication protocols, digital signature schemes, zero-knowledge protocols, *etc.* Using weak random numbers in such applications may result in an adversary ability to break the whole cryptosystem. However, for many simulation applications strong random numbers are not required, in other words, different applications may require different levels of randomness.

Generating high quality random numbers is a very serious problem–and is very difficult if deterministic methods are used. The best way to generate unpredictable random numbers is to use physical processes such as radioactive decay, thermal noise or sound samples from a noisy environment. However, generating random numbers using these physical processes is extremely inefficient. Therefore, most systems use PRNGs based on deterministic algorithms. Some of the commonly used PRNGs for cryptographic applications are ANSI X9.17 generator, PGP 2.x generator, SSH generator and Applied Cryptography generator [50], generator based on elliptic curves [51] and biometric random number [52].

Desired properties of PRNGs are (i) good randomness properties of the output sequence, (ii) reproducibility (especially for simulation applications), (iii) speed or efficiency and (iv) large period.

Unpredictability of random sequences is established using a random *seed* that is obtained by a physical source like timings of keystrokes. Without the seed, the attacker must not be able to make any predictions about the output bits, even when all details of the generator is known.

Since stream ciphers produce random looking keystream using the key as the seed of the generator, stream ciphers are also PRNGs. All testing mechanisms applied for

PRNGs can also be applied to stream ciphers to analyze their randomness properties.

A theoretical proof for the randomness of a generator is impossible to give, therefore statistical inference based on observed sample sequences produced by the generator seems to be the best option. Considering the properties of binary random sequences, various statistical tests can be designed to evaluate the assertion that the sequence is generated by a perfectly random source. Test suites [103, 111, 112, 113, 114] define a collection of tests to evaluate randomness of generators extensively.

One important attribute of test suites is the variety or coverage of tests that they include. A generator may behave randomly based on a number of tests, and fail when it is evaluated by another test. So, to have more confidence in the randomness of generators, coverage of test suite should be as high as possible. We defined the coverage of a test suite as the sequences that fail any of the tests in the suite for a pre-specified type I error.

Test suites produce many p-values while evaluating a random number generator. Interpretation of these p-values is sometimes complicated and requires careful attention. According to the NIST test suite, these p-values are evaluated based on (i) their uniformity and (ii) the proportion of p-values that are less than a predefined threshold value which is typically 0.01 [103]. As Soto stated in [115] to achieve reliable results using these measures, the statistical tests in a suite should be independent, in other words the results of the tests should be uncorrelated. In [116], the relation between *approximate entropy, overlapping serial and universal test* is analyze and highly correlated results are obtain using defective sources. In Figure 3.1, the relation of coverage and independence of tests are illustrated. The proportion of highlighted parts is equal to the coverage of the suite designed by six randomness tests. According to the figure, the tests T_5 and T_6 are not independent, due to their large intersection.

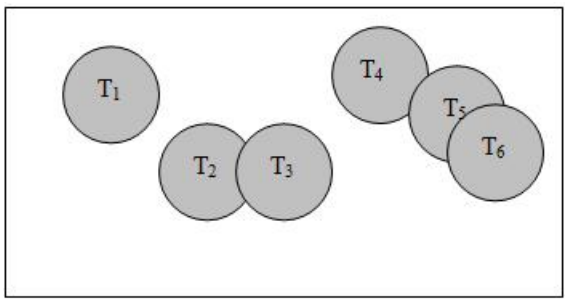


Figure 3.1: Independence and coverage of test suites

There is a strong relation between the coverage of the suite and independence of tests. In this chapter, we study the independence of some commonly used randomness

tests and present some theoretical and experimental results. Moreover, we define the concept of sensitivity, the effect of some transformations to sequences on test results. We propose to add the composition of the transformation and the test to the suite, whenever the effect of transformation is significant.

The organization of the chapter is as follows. In Section 3.1, basic background information about randomness tests and test suites, and a rough classification based on the test statistics, are presented. In Section 3.2, three new randomness tests based on random walks are proposed. In Section 3.3, theoretical and experimental results on the independence of randomness tests are presented for a subset of commonly used tests. In Section 3.4, the concept of *sensitivity* is defined and effect of some basic transformations is presented. In the last section, we give some concluding remarks and possible future work directions.

3.1 Randomness Tests

Let L be the set of n bit binary sequences, trivially $|L| = 2^n$. A statistical test T is a deterministic algorithm that takes a sample sequence and produces a decision regarding the randomness of a sequence, as $T : L \rightarrow \{\text{accept}, \text{reject}\}$. Therefore, T divides the set L of binary sequences $S_n = (s_1, s_2, \ldots, s_n)$ of length n into two sets;

$$\begin{aligned} L_A &= \{S_n \in L : T(S_n) = \text{accept}\} \subseteq L \\ L_R &= \{S_n \in L : T(S_n) = \text{reject}\} \subseteq L \end{aligned}$$

The size of these two sets is determined by the type I error α, *i.e.* $|L_R|/|L| = \alpha$.

The most commonly used test is probably the *frequency test* [103] that evaluates the randomness of a given sequence based on the number of ones (that is, the weight w) of the sequence. A trivial extension of the frequency test can be given as the *equidistribution test* that focuses on the frequencies of k-bit tuples throughout the sequence. *Overlapping* and *non-overlapping template matching tests* [103] only consider the number of occurrences of pre-specified templates in the sequence.

Another commonly used statistics in randomness testing is about the changes from 1 to 0 or visa versa, which is called a *run*. Runs test [103] focuses on the number of runs in the sequence which is expected to be around $\frac{(n+1)}{2}$ for a random sequence of length n. Alternatively, there are tests that concentrate on length of runs, particularly *length of the longest run of ones* [103] in the sequence.

Especially in cryptographic applications, the complexity of sequences -the ability to

reproduce the sequence- is of interest. Sequences with low complexity measures can be easily generated with alternative machines. Some examples of randomness tests based on complexity measures can be given as *linear complexity* [103], *Lempel-Ziv complexity* [63] and *maximum order complexity* [117]. A non-exhaustive list of randomness tests is given in Table 3.1.

Classification of Tests

While selecting a subset of randomness tests to be included in a test suite, it is necessary to understand what exactly the test measures. Tests that focus on similar properties should not be included in the test suite, but one of such a class should be chosen.

In [120], two important properties of random sequences are emphasized. The first is about the appearance of the sequence which is related to the *ability of the attacker to guess the next bit better than random guessing*. The second property is about the *ability to reproduce the sequence*, *i.e.*, the complexity of the sequence. Considering these properties, we give a rough classification of randomness tests into two categories:

- tests based on k-tuple pattern frequencies and
- tests based on ordering of k-tuples patterns.

Tests based on k-tuple pattern frequencies aim to detect the biases in the appearance of the sequences. Weaknesses in the appearance of sequences can be considered as the deviations in the frequencies of k-tuples ($1 \leq k \leq log_2 n$) which are expected to occur nearly equally many times throughout the sequence. An example for this category with $k = 1$ is the most commonly used frequency test and its extension equidistribution test with $k = 2, 3, \ldots$. Other examples can be given as runs, overlapping template, serial, coupon collectors, *etc.*

Tests based on ordering of k-tuples aim to detect weaknesses based on the reproducibility of sequences. Most of the test statistics in this category are related to the size of simpler systems that generate the sequence. If the size of these machines are less than expected, it is possible to use these machines to regenerate the sequence. Tests based on complexity measures like linear complexity, maximum order complexity can be given as examples.

These categories are closely related and obviously not disjoint. It is possible to find randomness tests that can be included in more than one category. As an example, consider the runs test which can be included to the category of *tests based on ordering*

Table 3.1: A non-exhaustive list of statistical randomness tests with input parameters, test statistics and constraints (The sequence length is denoted as n.)

Test	*Input Parameters*	*Test Statistics*	*Constraints*
Frequency [103]	-	Weight	$n > 100$
Frequency within a Block [103]	Block length, M	Block weights	$n > 100$ $M \geq 20$, $M > 0.01n$
Overlapping Templates Matching [103]	Template length, m Template T	Frequency of T	$n \geq 10^6$ $m = 9$ or $m = 10$
Non-overlapping Templates Matching [103]	Template length, m Template T	Frequency of T	$n \geq 10^6$ $m = 9$ or $m = 10$
Runs [103]	-	Number of runs	$n > 100$
Longest Run of ones [103]	Block length, M	Length of the longest run	$M = 8$, $n > 128$ $M = 128$, $n > 6272$ $M = 10^4$, $n > 750000$
Linear Complexity [103]	Block length, M	Linear complexity	$n > 10^6$ $500 \leq M \leq 5000$
Maximum Order Complexity [117]	-	Maximum order complexity	$n > 10^6$
Lempel-Ziv [103, 63]	Block length, M	Lempel-Ziv complexity	$M = 1024$, $n > 100000$
Matrix Rank [103]	Number of rows, M Number of columns, Q	Matrix ranks	$M = Q = 32$, $n > 38912$
Universal [118]	Block length, L Number of blocks, Q	Distance between matching patterns	$6 \leq L \leq 16$ $n > 387840$
Discrete Fourier Transform (DFT) [103]	-	Peak heights in DFT	$n > 1000$
Serial [103]	Block length, M	Frequency of M-bit patterns	$M < \lfloor log_2 n \rfloor - 2$
Approximate Entropy [103]	Block length, M	Frequency of M and $M + 1$-bit patterns	$M < \lfloor log_2 n \rfloor - 2$
Random Walk Excursion [18]	Block length, M	Number of excursions	$M = 16$, $n > 500$ $M = 128$, $n > 3600$ $M = 256$,$n > 6800$
Random Walk Height [18]	Block length, M	Maximum Height	$M = 64$, $n > 2000$ $M = 512$, $n > 14000$ $M = 1024$, $n > 26000$
Random Walk Expansion [18]	Block length, M	Maximum Expansion	$M = 32$, $n > 1000$ $M = 64$, $n > 2000$ $M = 128$, $n > 4000$
Coupon Collectors [119]	Pattern size, t	First index where all t-bit patterns are observed	$n > 2^t$
Maximum-of-t	Block length, M Pattern size, t	Maximum of t-bit patterns	$n > 100M$ $M > 10t$
Minimum-of-t	Block length, M Pattern size, t	Minimum of t-bit patterns	$n > 100M$ $M > 10t$

of $k = 1$-tuples and also to the category of *tests based on $k = 2$-tuple pattern frequency* since it actually counts the number of 01 and 10 patterns throughout the sequence.

Even for applications that do not require strong random numbers, generators should produce uniform outputs, in other words, should pass the tests in the first category. The tests in the second category usually take more time compared to the first category, but they are more important for applications that require strong randomness such as cryptographic applications where randomness is used in key, nonce and initial vector generation.

Test Suites

A randomness test suite is a collection of randomness tests that are selected to analyze the randomness properties of generators. Here, we provide some information about widely used test suites.

- *Knuth Test Suite* [113], developed in 1998, presents several empirical tests including frequency, serial, gap, poker, coupon collector's, permutation, run, maximum-of-t, collision, birthday spacings and serial correlation.
- *DIEHARD Test Suite* [111] consists of 18 different, independent statistical tests including; birthday spacings, overlapping 5-permutations, binary rank, bitstream test, monkey tests on 20-bit Words, monkey tests OPSO, OQSO, DNA, count the 1's in a stream of bytes, count the 1's in specific bytes, parking lot, minimum distance, 3D spheres, squeeze, overlapping sums, runs and craps. Since the sequence sizes are fixed for most of the tests, suite is not suitable to test sequences with variable sizes.
- *Crypt-XS* [112] suite which was developed in the Information Security Research Centre at Queensland University of Technology consists of frequency, binary derivative, change point, runs, sequence complexity and linear complexity tests.
- *NIST Test Suite* [103] consists of 16 tests namely frequency, block frequency, cumulative sums, runs, long runs, rank, spectral, nonoverlapping template matchings, overlapping template matchings, Maurer's universal statistical, approximate entropy, random excursions, Lempel-Ziv complexity, linear complexity, and serial. During the evaluation of block ciphers presented for AES, Soto [115] proposed nine different ways to generate large number of data streams from a block cipher and tested these streams using the NIST test suite to evaluate the security of AES candidates.

- *TestU01 Suite* [114] is another test suite for empirical testing of random number generators. This suite consists of many randomness tests and it is also suitable to test sequences that take real values.

Multi Level Testing

In [121], to increase the power of tests, it is proposed to apply the test N times to disjoint parts of the sequence, yielding N different test statistics, $t_1, t_2, \ldots, t_N$. Then, using standard goodness of fit tests, the empirical distribution of t_i values is compared to the theoretical distribution of the test statistic under H_0. This is called level-2 testing (See Figure 3.2). To increase the power, the level of tests may be increased further. The level-2 version of frequency test is the *frequency test within a block* [103] that focuses on the weight of disjoint parts of a given sequences.

$$\text{One Level: } \underbrace{(s_1, \ldots, s_n)}_{t} \rightarrow \text{Evaluate } t$$

$$\text{Two Level: } \underbrace{(s_1, \ldots, s_m)}_{t_1}, \ldots \underbrace{(s_{n-m+1}, \ldots, s_n)}_{t_N} \rightarrow \text{Evaluate } t_1, \ldots, t_N$$

Figure 3.2: Multi level testing

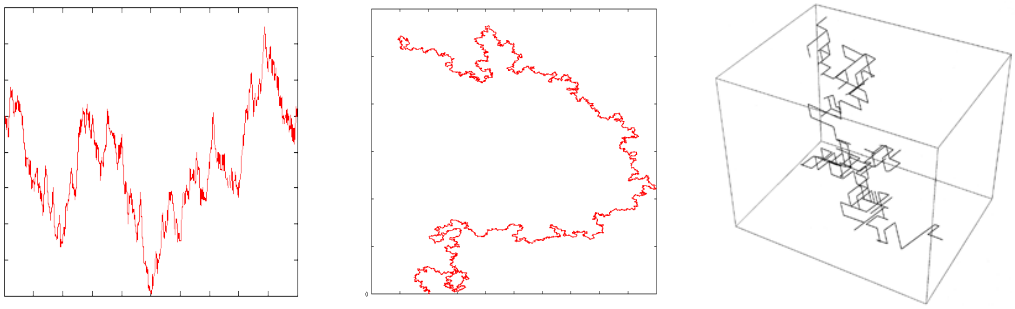
3.2 New Tests Based on Random Walks

Random walks are stochastic processes based on a sequence of changes, where the magnitude or direction of change is determined by chance. They are widely used in applications of physics, economics and population genetics. They can also be used to analyze randomness properties of binary sequences. In the test suite of NIST [103], there are three tests concerning random walks namely, *Cumulative Sums*, *Random Excursions* and *Random Excursion Variants* test. The last two tests can only be applied to sequences with length $n > 10^6$, due to approximations used for the distribution of test statistic.

In this section, we present three new randomness tests based on random walks. Proposed tests observe properties associated with the sequence of partial sums, such as number of zero sums, maximum value and the expansion of random walks. The basic idea of tests are very similar to the tests presented in the NIST suite, but they are also suitable to test shorter sequences, since exact probability distributions are used during evaluation.

3.2.1 Random Walks

Consider a particle on the real line located at a point $k \in Z$. At each step, this particle moves either to $k+1$ or $k-1$ by equal probabilities. The motion described by this particle is known as the one dimensional random walk. Random walks can be generalized by various ways; a motion in 2, 3 or higher dimension, a motion concerning different probabilities to different directions *etc.* Examples of one, two and three dimensional random walks are given in Figure 3.3



(a) One dimensional random walk.

(b) Two dimensional random walk

(c) Three dimensional random walk

Figure 3.3: Random walk examples

In this work, we only consider one dimensional random walks with equal probabilities, since they are very suitable for analyzing the randomness properties of finite binary sequences.

Let $s_1, s_2, \ldots$ be independent random variables with the following distribution

$$P(s_i = 1) = P(s_i = -1) = \frac{1}{2}. \tag{3.2.1}$$

The partial sum S_i is defined by $S_i = s_1 + s_2 + \ldots + s_i$, $i = 1, 2, \ldots$ and $S_0 = 0$. Then, the sequence $\{S_i\}_{i=0}^{\infty}$ defines a one dimensional random walk.

A random walk can be represented in the plane by the points (x_i, y_i) where $x_i = i$, $y_i = S_i$, for $i \geq 1$ and $x_0 = y_0 = 0$. The points at which the path intersects the x-axis correspond to $S_n = 0$. The part of the sequence between two such successive intersection points is referred to an *excursion*. Length of an excursion starting at (x_i, y_i) and ending at (x_i, y_i) is $j - i$. Note that in such a case, $y_i = y_j = 0$ and $y_k \neq 0$ for $i < k < j$. The maximum value of $|y_n|$ is defined as the *height* of a random walk and the *expansion* of a random walk is defined as the difference between the maximum and the minimum value of S_n, i.e. $\max\{y\} - \min\{y\}$ (See Figure 3.4).

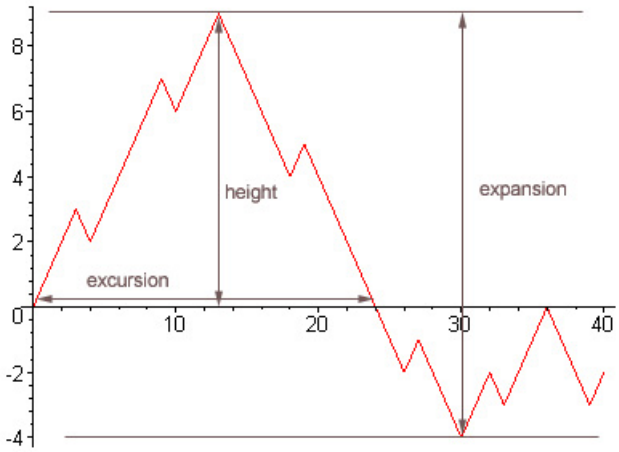


Figure 3.4: Height, excursion and expansion in one dimensional random walks

Let l be the length of an excursion. The probability distribution of l is given by

$$P(l = x) = \begin{cases} \frac{C_{\frac{x}{2}-1}}{2^{x-1}}, & \text{even } x \\ 0, & \text{odd } x \end{cases} \tag{3.2.2}$$

where C_i is the Catalan number $\frac{1}{i+1}\binom{2i}{i}$. It is obvious that the length of an excursion is always even.

Let P_k be the probability that the length of a random excursion is greater than k. Obviously,

$$P_{2k} = 1 - \sum_{i=1}^{k-1} P(2i) \tag{3.2.3}$$

and $P_{2k} = P_{2k+1}$.

For our tests, we are interested in the probability $P(N, k)$ that having exactly k excursions in a given sequence of length N. We calculate the probability distribution separately for balanced and unbalanced sequences.

- *Balanced sequences with k complete excursions.* To calculate the number of balanced sequences with k complete excursions, we first utilize the number of different partitions of $N/2$ with k positive integers $p_1, \ldots, p_k$, such that

$$N = 2p_1 + 2p_2 + \ldots + 2p_k. \tag{3.2.4}$$

  Then, the probability of having k successive excursions of respective lengths $2p_1, \ldots, 2p_k$ is $P(2p_1).P(2p_2) \ldots P(2p_k)$. Consequently, the desired probability is given by the sum $\sum P(2p_1) \ldots P(2p_k)$ where the summation ranges over all

ordered partitions $(p_1, \ldots, p_k)$ of $N/2$ into k positive integers.

- *Unbalanced sequences with $k-1$ complete and 1 incomplete excursions.* Let the length of the first $k-1$ complete excursions be $N-2m$ and the last incomplete excursion be of length $2m$ ($m > 0$). In this case, the required probability is $\sum_{m=1}^{N/2} P_{N-m} \sum P(2p_1) \ldots P(2p_{k-1})$ where the inner summation runs over all ordered partitions $p_1, \ldots, p_{k-1}$ of $N/2 - m$ into $k-1$ positive integers.

Given a random walk of length B. Let w be the weight of the sequence and $m = B - w$. It is possible to represent this walk as a path starting from the bottom left corner in a $m \times w$ grid, (stepping one unit up for a $+1$ term and stepping one unit right for a -1 in the sequence) and ending at the top right corner (See Figure 3.5). The paths that intersects the line l_1 corresponds to random walks with height greater than r. The number of such paths is known to be equal to $\binom{m+w}{m-r}$, provided $r > |m-w|$.

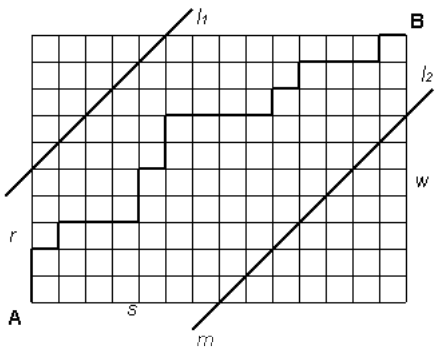


Figure 3.5: Grid representation of an example random walk (1,2,1,2,1,0,-1,0,1,0,1,2,1,0,-1,-2,-1,-2,-1,-2,-3,-4,-3,-4) of length $w + m = 24$ with $\max\{y_i\}$ less than $r = 5$ and $\min\{y_i\}$ greater than $-s = -7$

Let us investigate a more general case. To calculate the number of paths starting from A, finishing at B and not cutting the two given lines l_1 and l_2 (that corresponds to random walks with m ones and w zeros and taking values between $[-s, r]$), the principle of inclusion-exclusion is employed. The required number can be found by excluding the paths cutting the above and below lines from all paths and including the paths cutting both above and below lines and continuing in this way. Thus, the probability, $Pr(S_{max} < r, S_{min} > -s|m, w)$ is

$$\frac{\binom{m+w}{m} - \sum_{i=1}^{\infty} \left(\binom{m+w}{m-ir-(i-1)s} + \binom{m+w}{w-(i-1)r-is} - \binom{m+w}{m-ir-is} - \binom{m+w}{w-ir-is} \right)}{2^{m+w}} \tag{3.2.5}$$

for $r, s > |m - w|$ and 0, otherwise.

If $r = s$, then

$$Pr(S_{max} < r|m, w) = \begin{cases} \frac{\binom{m+w}{m} - \sum_{i=1}^{\infty} \left\{ \left[\binom{m+w}{m-ir} + \binom{m+w}{w-ir} \right] (-1)^{i+1} \right\}}{2^{m+w}} & \text{if } r > |m - w|, \\ 0 & \text{otherwise.} \end{cases} \quad (3.2.6)$$

3.2.2 Test Descriptions

We define three new tests to decide the randomness of a sequence considering statistical distributions based on certain properties of the random walk; (i) *Random Walk Excursion Test*, (ii) *Random Walk Height Test*, (iii) *Random Walk Expansion Test*. As their names suggest, tests consider the number of excursions, the height and the expansion of the random walk generated by the input binary sequence, respectively. All of these tests are level-2 tests, *i.e.* tests partition the input sequence into non-overlapping blocks with certain lengths ($B = 16, 32, 64, \ldots$) and calculate test statistics e_j, $(j = 1, 2, \ldots, \lfloor \frac{N}{B} \rfloor)$ for each subblock and apply a χ^2-goodness of fit test.

To test a binary sequence $\{a_i\}_{i=1}^n = \{a_1, a_2, \ldots, a_n\}$ for randomness, we first convert $\{a_i\}$ into the sequence $\{b_i\}_{i=1}^n$ by setting $b_i = 1 - 2i$ so that $b_i \in \{-1, 1\}$, $i = 1, 2, \ldots, n$. The pseudocode of the tests are given in Algorithm 3.2.1, 3.2.2 and 3.2.3.

Algorithm 3.2.1: RANDOM WALK EXCURSION TEST(B, b_n)

---

for $j \leftarrow 1$ **to** $\lfloor \frac{n}{B} \rfloor$
{
  $e_j = 0$;
  for $i \leftarrow 1$ **to** B
  { $S_i = \sum_{k=1}^{i} b_{(j-1)B+k}$;
  for $i \leftarrow 1$ **to** B
  {
    if $S_i = 0$
      then $e_j + +$;
Apply χ^2-Goodness of Fit test to e_j values;
return $(p - value)$

Algorithm 3.2.2: RANDOM WALK HEIGHT TEST(B, b_n)

---

for $j \leftarrow 1$ **to** $\lfloor \frac{n}{B} \rfloor$

$\begin{cases} e_j = 0; \\ \textbf{for } i \leftarrow 1 \textbf{ to } B \\ \begin{cases} S_i = \sum_{k=1}^{i} b_{(j-1)B+k}; \end{cases} \\ \textbf{for } i \leftarrow 1 \textbf{ to } B \\ \begin{cases} \textbf{if } |S_i| > e_j \\ \quad \textbf{then } e_j = |s_i|; \end{cases} \end{cases}$

Apply χ^2-Goodness of Fit test to e_j values;

return $(p - value)$

Algorithm 3.2.3: RANDOM WALK EXPANSION TEST(B, b_n)

---

for $j \leftarrow 1$ **to** $\lfloor \frac{n}{B} \rfloor$

$\begin{cases} e_j = 0,\ max = 0,\ min = 0; \\ \textbf{for } i \leftarrow 1 \textbf{ to } B \\ \begin{cases} S_i = \sum_{k=1}^{i} b_{(j-1)B+k}; \end{cases} \\ \textbf{for } i \leftarrow 1 \textbf{ to } B \\ \begin{cases} \textbf{if } S_i > max \\ \quad \textbf{then } max = S_i; \\ \textbf{if } -S_i > min \\ \quad \textbf{then } min = -S_i; \end{cases} \\ e_j = max + min; \end{cases}$

Apply χ^2-Goodness of Fit test to e_j values;

return $(p - value)$

In the following tables, intervals and corresponding probabilities are presented. These intervals are determined in such a manner that each interval has approximately equal probability. Firstly, we calculate the exact probability distributions for length of excursion, heights and expansion of random walks. Then, to apply χ^2-Goodness of fit test, we generate boxes with approximately equal probabilities for block lengths of $16, 32, 64, 128$ and 256 bits. These probabilities are given in Tables 3.2, 3.3 and 3.4, respectively.

Table 3.2: Interval and probability values of Random Walk Excursion Test for block lengths of 16,32, 64, 128 and 256 bits.

	B=16, $n > 509$		B=32, $n > 761$		B=64, $n > 1849$		B=128, $n > 3615$		B=256, $n > 6772$	
	Interval	Pr.	Interval	Pr.	Interval	Pr.	Interval	Pr.	Interval	Pr.
Box 1	0	0.196	0-1	0.279	0-1	0.198	0-2	0.210	0-3	0.198
Box 2	1	0.196	2-3	0.261	2-3	0.192	3-5	0.200	4-7	0.189
Box 3	2	0.183	4-5	0.210	4-5	0.173	6-8	0.177	8-12	0.206
Box 4	3	0.157	6-16	0.247	6-8	0.206	9-12	0.184	13-19	0.211
Box 5	4-8	0.266	-	-	9-32	0.228	13-64	0.227	20-128	0.193

Table 3.3: Interval and probability values of Random Walk Height Test for block lengths of 64,128, 256, 512 and 1024 bits.

	B=64, $n > 1693$		B=128, $n > 3555$		B=256, $n > 6597$		B=512, $n > 14222$		B=1024, $n > 26256$	
	Interval	Pr.	Interval	Pr.	Interval	Pr.	Interval	Pr.	Interval	Pr.
Box 1	0-6	0.252	0-8	0.180	0-12	0.196	0-17	0.180	0-25	0.196
Box 2	7-8	0.227	9-11	0.241	13-16	0.230	18-22	0.205	26-32	0.203
Box 3	9-10	0.182	12-14	0.209	17-20	0.194	23-28	0.214	33-40	0.200
Box 4	11-13	0.173	15-18	0.183	21-26	0.195	29-36	0.195	41-52	0.204
Box 5	14-64	0.163	19-128	0.185	27-256	0.232	37-512	0.203	53-1024	0.195

3.3 Independence of Tests

There are extensive number of randomness tests in the literature and to design a test suite a careful selection should be done. Many generators may appear to be random according to a number of tests, but may be non-random when subjected to another test, therefore the variety or the coverage of the tests used in the test suite should be high enough. However, including dependent tests may result in wrong conclusions about the generators.

Two tests T_1 and T_2 are considered to be independent if the distribution of their test statistics t_1 and t_2 (and corresponding p-values) are independent, that is

$$Pr(t_1|t_2) = Pr(t_1), \tag{3.3.1}$$

and visa versa.

In this section, we analyze the relation between some of the commonly used tests and try to observe if there exists any statistically significant correlation. We consider ten level-1 tests, which are also suitable for testing to short sequences. Given a sequence $(s_1, s_2, \ldots, s_n)$ of length n, each test defines a test statistic as described below.

- *Frequency Test:* The test statistic is the weight of the sequence, that is $t = s_1 + \ldots + s_n$, taking values between 0 and n.

Table 3.4: Interval and probability values of Random Walk Expansion Test for block lengths of 32, 64 and 128 bits.

	B=32, $n > 963$		B=64, $n > 2025$		B=128, $n > 3978$	
	Interval	Pr.	Interval	Pr.	Interval	Pr.
Box 1	0-6	0.307	0-8	0.188	0-12	0.196
Box 2	7	0.166	9-10	0.234	13-14	0.165
Box 3	8-9	0.266	11-12	0.212	15-17	0.235
Box 4	10-32	0.259	13-15	0.206	18-21	0.214
Box 5	-	-	16-64	0.158	22-128	0.187

- *Overlapping Template Test:* Test statistic is the number of occurrences of a m bit pattern $\mathbf{a}$ throughout the sequence, that is

$$t = |\{i|(s_i, \ldots, s_{i+m-1}) = \mathbf{a}, 1 \leq i \leq n - m + 1\}|. \tag{3.3.2}$$

  For our experiments, $\mathbf{a}$ is chosen to be 111.

- *Longest Run of Ones Test:* Test statistic is the length of the longest run of ones, that is

$$t = \max\{m|s_i = s_{i+1} \ldots = s_{i+m} = 1, 1 \leq i \leq n\}, \tag{3.3.3}$$

  taking values between 0 and n.

- *Runs Test:* Test statistic is the number runs throughout the sequence, taking values between 1 and n.

- *Random Walk Height Test:* Test statistic is the height of random walk, that is

$$t = \max_{i=1,\ldots,n} |\sum_{j=1}^{i}(2s_j - 1)| \tag{3.3.4}$$

  taking values between 1 and n.

- *Random Walk Excursion Test:* Test statistic is the number of excursions in the random walk, that is

$$t = |\{i|\sum_{j=1}^{i}(2s_j - 1) = 0, 1 \leq i \leq n\}|, \tag{3.3.5}$$

  taking values between 0 and $n/2$

- *Linear Complexity Test:* Test statistic is the linear complexity of the sequence, that is, the length of the shortest LFSR that generates the sequence, taking values between 0 and n. Linear complexity of a sequence can efficiently be calculated using the Berlekamp-Massey algorithm.

- *k-error Linear Complexity Test:* Test statistic is the k-error linear complexity of the sequence that is the length of the shortest LFSR that generate the sequence with at most k bit difference. In our experiments, we focused on the $k = 1$ case, in which the test statistic takes values between 0 and $n/2$.
- *Maximum Order Complexity Test:* Test statistic is the maximum order complexity of the sequence that is the length of the shortest feedback shift register that generates the sequence, taking values between 0 and $n - 1$.
- *Lempel-Ziv Test:* Test statistic is the Lempel-Ziv complexity of the sequence that takes its maximum value as $n/2$. For instance Lempel-Ziv complexity of s = 010101001001011 is 7, since different patterns observed are 0|1|01|010|0100|10|11.

3.3.1 Theoretical Results

In this section, to analyze the relation of two tests T_1 and T_1, we present some theoretical bounds on the maximum and minimum values of t_1 as a function of t_2.

Frequency versus Runs Test Given a sequence of length n and weight w, the maximum possible number of runs R is

$$\max\{R\} = \begin{cases} n & \text{if } w = \frac{n}{2} \\ \min\{2w+1, 2(n-w)+1\} & \text{if } w \neq \frac{n}{2} \end{cases}$$

whereas the minimum number of runs R is

$$\min\{R\} = \begin{cases} 2 & \text{if } 1 \leq w < n \\ 1 & w = 0 \text{ or } w = n \end{cases} .$$

For balanced sequences, the number of runs takes values between 1 and n, but as the weight of the sequence deviates from $n/2$, the maximum possible number of runs decreases. Sequences with weight less than $n/4$, it is not possible to achieve expected number of runs. Conversely, given the number of runs R, $w \geq \lfloor \frac{R}{2} \rfloor$.

Frequency versus Random Walk Height Test Given a sequence of length n and weight w, the random walk height test statistic H attains the maximum value as $\max\{w, n-w\}$ and minimum value

$$\min\{H\} = \begin{cases} 1 & \text{if } w = \lfloor \frac{n}{2} \rfloor \\ |n-2w| & \text{otherwise} \end{cases} .$$

Given H, the weight of the sequence is at least $\min\{H, n-H\}$. From this property, if a sequence fails random walk height test, it is very likely that it also fails the frequency

test. The relation is more significant for short sequences.

Frequency versus Longest Run of Ones Given a sequence of length n and weight w, the longest run of ones test statistic L takes its maximum value as w and minimum value as $\left\lceil \frac{w}{n-w-1} \right\rceil$.

Frequency versus Random Walk Excursion Test Given the weight w of a n-bit sequence, the number of random walk excursion test statistic E takes its maximum value as $\min\{w, n-w\}$ and minimum value as

$$\min\{E\} = \begin{cases} 1 & \text{if } w = \frac{n}{2} \\ 0 & \text{otherwise} \end{cases} .$$

Runs versus Random Walk Excursion Both tests are related to the speed of changes from 0 to 1 (or from 1 to 0). Sequences with large number of excursions are expected to have large number of runs, similarly sequences with small number of runs, are expected to have less number of excursions. Each excursion consists of at least two runs, therefore number of runs is at least twice the number of excursions.

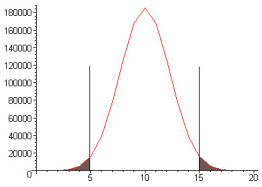
Frequency versus Linear Complexity There is no direct relation between the weight and the linear complexity of the sequence. Even with very low weight, it is possible to achieve high linear complexity values. As an example, consider the sequences with $w = 1$, location of the bit 1 determines the linear complexity. There is however a strong relation between the weight of the Discrete Fourier Transform of the sequence and its linear complexity, so-called Blahut's theorem [122].

3.3.2 Experiments on Short Sequences

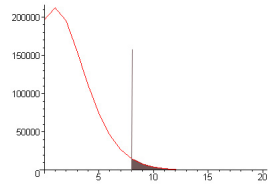
As an alternative definition of independence of randomness tests, tests T_1 and T_2 can be considered independent, if their rejection regions are independent for all selection of α. In this part of the study, we analyze the relations of 10 tests given in Section 3.3, focusing the rejection regions with size approximately 0.01.

Considering all binary sequences of length $n = 20$ and 30, we formed the rejection regions $R_n^{T_i}$ of each test, that is, the set of all n bit sequences that fail T_i. The upper and lower acceptable limits for test statistics are calculated so that $\alpha \approx |R_n^1| \approx \ldots \approx |R_n^{10}| \approx 0.01$. If the test statistic is more extreme than the lower or upper limit given in Table 3.5, the sequence is assumed to be non-random. Empirical distributions of two test statistics for $n = 20$ is given in Figure 3.6.

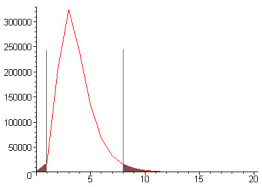
The $(i,j)^{th}$ entry of Tables 3.6 and 3.7 represents the proportion of sequences that fail T_i and T_j to the sequences that fail T_i, that is, $\frac{|R_n^i \cap R_n^j|}{|R_n^i|}$, for $n = 20, 30$. The expected



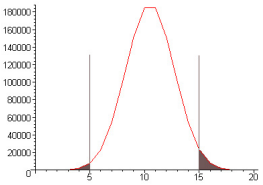
(a) Frequency



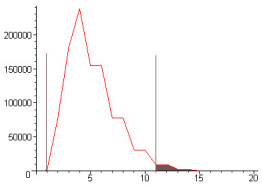
(b) Overlapping Template



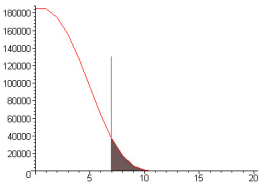
(c) Longest Run of Ones



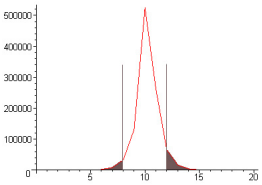
(d) Runs



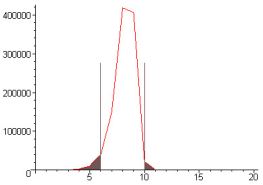
(e) Random Walk Height



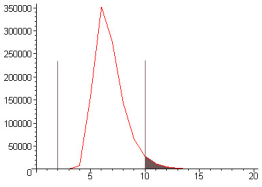
(f) Random Walk Excursion



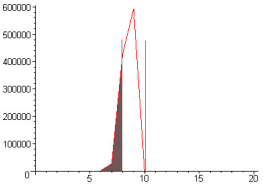
(g) Linear Complexity



(h) k-error Linear Complexity



(i) Maximum Order Complexity



(j) Lempel-Ziv

Figure 3.6: Distributions of test statistics for $n = 20$

Table 3.5: Lower Limits (LLs) and Upper Limits (ULs) of the test statistics for 20 bit sequences and corresponding type I error, α

Test	$n = 20$			$n = 30$		
	LL	UL	α	LL	UL	α
Frequency	5	15	0.011818	9	21	0.016125
Overlapping Template	0	8	0.014478	0	10	0.017750
Longest Run of Ones	1	8	0.012691	1	9	0.010727
Runs	5	15	0.011818	9	21	0.016125
RW Height	1	11	0.014395	1	14	0.010446
RW Excursion	0	7	0.022461	0	10	0.011818
LC	8	12	0.031250	13	17	0.031250
1-error LC	6	10	0.012996	11	15	0.019407
Maximum Order Complexity	2	10	0.017889	4	12	0.016291
Lempel-Ziv	8	10	0.026367	11	13	0.031378

value of this proportion is 0.01 and tables are expected to be symmetric for larger n values. In the tables, the percentages that significantly deviate (> 0.10) from expected values are highlighted.

Table 3.6: Results of tests for all sequences of length $n = 20$ for $\alpha = 0.01$

Test	Frequency	Overlapping Template	Longest Run of Ones	Runs	RW Height	RW Excursion	LC	1-error LC	MOC	Lempel Ziv
Frequency	-	0.4334	0.2012	0.04341	1	0	0.1011	0.0657	0.1785	0.3689
Overlapping Template	0.3538	-	0.4391	0.0516	0.4171	0	0.0491	0.0234	0.1699	0.1632
Longest Run of Ones	0.1874	0.5009	-	0.0634	0.2370	0	0.0385	0.0225	0.1740	0.1040
Runs	0.04341	0.0632	0.0681	-	0.0618	0.1933	0.0933	0.0531	0.1635	0.1228
RW Height	0.8210	0.4195	0.2089	0.0507	-	0	0.1050	0.0542	0.1972	0.3108
RW Excursion	0	0	0	0.1017	0	-	0.0351	0.0160	0.0369	0.0348
LC	0.0383	0.0227	0.0156	0.0353	0.0484	0.0252	-	0.1339	0.1737	0.0424
1-error LC	0.0598	0.0260	0.0220	0.0483	0.0601	0.0277	0.3220	-	0.0961	0.0643
MOC	0.1179	0.1375	0.1235	0.1080	0.1586	0.0464	0.3035	0.0698	-	0.0951
Lempel-Ziv	0.1654	0.0896	0.05006	0.0550	0.1697	0.0296	0.05031	0.0317	0.0645	-

According to Table 3.6 and 3.7, frequency, overlapping template (with input template 111), longest run of ones, random walk height tests and maximum order complexity tests are closely related. Also, there is a correlation between the results of linear complexity and 1-error linear complexity tests. Moreover, a significant relation is observed between Lempel-Ziv and frequency test. As also mentioned in the theoretical results part in Section 3.3.1, no correlation between the weight and the linear complexity is observed.

Another interesting result is that none of the sequences that fail the random walk excursion test, fail any of the (i) frequency; (ii) overlapping template; (iii) longest run of ones or (iv) random walk height tests for sequences of length $n = 20$ and 30. This means that including the random walk excursion test increases the coverage of test suites significantly. To measure the effect of each test to the coverage, we present the number of 20 bit sequences that only fail the given test in Table 3.8. The *tests based*

Table 3.7: Results of tests for all sequences of length $n = 30$ for $\alpha = 0.01$

Test	Frequency	Overlapping Template	Longest Run of Ones	Runs	RW Height	RW Excursion	LC	k-error LC	MOC	Lempel Ziv
Frequency	-	0.3853	0.1231	0.0409	0.5591	0	0.0339	0.0207	0.1035	0.3399
Overlapping Template	0.3500	-	0.3096	0.0733	0.2490	0	0.0309	0.0185	0.0875	0.1631
Longest Run of Ones	0.1851	0.5124	-	0.0869	0.1602	0	0.0303	0.0172	0.1026	0.0991
Runs	0.0409	0.0807	0.0578	-	0.0441	0.0876	0.0343	0.0212	0.1016	0.1308
RW Height	0.8630	0.4231	0.1645	0.0681	-	0	0.0355	0.0204	0.1498	0.3520
RW Excursion	0	0	0	0.1195	0	-	0.0320	0.0192	0.0328	0.0395
LC	0.0175	0.0175	0.0104	0.0177	0.0119	0.0121	-	0.1398	0.0250	0.0328
1-error LC	0.0172	0.0170	0.0095	0.0176	0.0110	0.0117	0.2251	-	0.0184	0.0330
MOC	0.1025	0.0953	0.0676	0.1005	0.0960	0.0238	0.0480	0.0219	-	0.0917
Lempel-Ziv	0.1747	0.0922	0.0339	0.0672	0.1172	0.0149	0.0327	0.0204	0.0476	-

Table 3.8: Number of sequences that only fail the given test (but pass all other tests)

Tests	Number of Sequences
Linear Complexity	22436
Lempel-Ziv Complexity	19680
Random Walk Excursion	19428
Maximum Order Complexity	8419
$k = 1$-error Linear Complexity	7895
Runs	6454
Longest Run of Ones	6196
Overlapping Template	4765
Random Walk Height	1163
Frequency	0

on ordering of k-tuples seem to increase the coverage of the selection more compared to *tests based on k-tuple pattern frequency* and this is mainly due to the correlation of these tests presented in above tables. Also, it is observed that all sequences that fail frequency test also fail any of the other tests in our scheme. So, there is no contribution of frequency tests to the coverage of selected tests, for sequences of length 20.

We also calculated the coverage, that is $|\cup_{i=1}^{10} R_n^i|$, for $n = 20$ and 30 as 0.122948 and 0.134930, respectively. Whenever $R_n^{T_i}$ sets are disjoint, coverage takes its maximum value as $\sum \alpha_i$ which is 0.176163 for 20 and 0.181317 for 30 bit sequences. Due to the correlations in this suite, coverage reduces around 30%.

Testing short sequences, these correlations should be considered. As the length of the sequences increase, it is possible to observe weaker correlation between tests. For instance, in Table 3.6 for $n = 20$, the number of highlighted cells (proportion > 0.1) is 37, whereas this number decreases to 27 for $n = 30$ given in Table 3.7. It should be noted that in case of testing longer sequences by level-2 version of these tests, correlations still exist whenever the input block size is small.

3.4 Sensitivity of Tests

To have more confidence in a random number generator, it is advantageous to use as many randomness tests as possible. In this part of the study, we propose to apply simple transformations to input sequences that significantly change the output p-value of a randomness test as an alternative to developing more tests.

Definition 3.4.1. Consider a randomness test T and a transformation $\sigma : L \rightarrow L$ where L is the set of all n bit binary sequences. T is said to be invariant under σ if for any S in L, $T(S) = T(\sigma(S))$.

Here, we define a new concept of *sensitivity* to measure the effect of a transformation to output p-values. If a test T is invariant under σ, sensitivity of T to σ is represented by 0. If the transformation has small effect on the test results, that is, there is a significant correlation between $T(S)$ and $T(\sigma(S))$, sensitivity is represented by 1. Whenever $T(S)$ and $T(\sigma(S))$ are statistically independent, sensitivity is represented by 2, in those cases $T(\sigma(.))$ can be added to the test suite as a new test.

The transformation σ can be chosen in various ways, in this section, we consider a few of them as examples.

- *Complementation* a binary sequence is applying the unary bitwise NOT operation, that is $\sigma_c(s_1, \ldots, s_n) = (s_1 \oplus 1, \ldots, s_n \oplus 1)$.
- *l-Rotation* is a circular shift operation commonly used in cryptography, that is $\sigma_{l-rot}(s_1, \ldots, s_n) = (s_{l+1}, \ldots, s_n, s_1, \ldots, s_l)$. Most of the level-2 tests are invariant to l-rotation, when l is equal to the block length.
- i^{th} *Bit flip* is simply flipping i^{th} bit of the sequence, that is $\sigma_{f_i}(s_1, \ldots, s_n) = (s_1, \ldots, s_i \oplus 1, \ldots, s_n)$.
- *Reversing* is simply considering the sequence backwards, that is $\sigma_{rvs}(s_1, \ldots, s_n) = (s_n, \ldots, s_1)$.
- l^{th} *Derivative* of a sequence is the summation of the sequence and its l-bit rotation, that is $\sigma_{d_l}(s_1, \ldots, s_n) = (s_1 \oplus s_{l+1}, \ldots, s_{n-l} \oplus s_n, \ldots, s_n \oplus s_l)$.

In Table 3.9, sensitivity of the tests selected in previous section are given according to $\sigma_c, \sigma_{l-rot}, \sigma_{f_i}, \sigma_{rvs}$ and σ_{d_l}.

As observed from the Table 3.9, some of the transformations do not have any effect on the test results. Complementing the input sequences only affect the results of overlapping template and longest run of ones tests. However, for the overlapping template

Table 3.9: Sensitivity of randomness tests toward some transformations.

Tests	σ_c	σ_{l-rot}	σ_{f_i}	σ_{rvs}	σ_d
Frequency	0	0	1	0	2
Overlapping template	2	1	1	0	2
Longest run of ones	2	1	1	0	2
Runs	0	1	1	0	2
Random walk height	0	1	1	2	2
Random walk excursion	0	1	1	2	2
Linear complexity	1	2	2	2	2
1-error linear complexity	1	2	0	2	2
Maximum order complexity	0	1	2	2	2
Lempel-Ziv	0	1	1	2	2

test, instead of complementing the whole sequence, it is also possible to complement the input template which would result in the same p-value. l-rotation affects the results of linear complexity and 1-error linear complexity tests, whereas flipping i^{th} bit affects the results of linear complexity and maximum order complexity tests. Reversing the sequences significantly changes the outputs of tests based on random walks and complexity measures. However, for balanced sequences reversing the output of random excursion test does not affect the output. Taking the l^{th} derivative of sequences significantly affect all test results available in our set. So, taking the l^{th} derivative seems to be a good choice of transformation to design new tests.

It is obvious that the independence of $T(\sigma(S))$ and $T(S)$ is not enough to justify adding $T(\sigma(\cdot))$ to the suite. It should also be independent of other tests in the suite. As an example, applying the frequency tests to the first derivative of the sequence is equivalent to applying the runs test.

3.5 Summary

In this chapter, we focus on the statistical testing of PRNGs, it is our interest since stream ciphers can also be considered as PRNGs. We first consider the sequences as random walks, and proposed three tests. The apparent benefit of using the proposed tests rises from the fact that no approximations have been used for test statistics. Therefore, they can be used to test short sequences. As a future work, it is also possible to extend these tests using two-dimensional graphs.

We also emphasize the importance of independence of randomness tests in test suites and present some theoretical and experimental results. We experimentally observe that frequency, overlapping template (with input template 111), longest run of ones, random

walk height tests and maximum order complexity tests produce correlated results for short sequences. These correlations should be considered while analyzing generators using short sequences. The strength of these correlations is likely to decrease as the input lengths increase, but in the case of testing longer sequences by level-2 version of these tests, correlations still exist whenever the input block size is small.

We also define the concept of sensitivity, where we analyze the effect of simple transformations on test results. If the transformation significantly changes the output p-values, then the composition of transformation and the test may be included in the suite to increase the coverage. Ideally, we would like to have each test applied to a transformed sequence $\sigma(S)$ to be independent of all different tests applied to the original sequence. Clearly, as the set of allowable transformations grows, this becomes harder to achieve. By choosing a good set of allowable transformations, one can use a given set of tests in a more powerful fashion. For example, one should not introduce unnatural transformations of the data, but stick to a set of transformations which are generated by a small set of basic transformations, such as the ones given here as examples. It is of interest to investigate this problem further in future work.

Chapter 4

Tests Based on Algebraic Properties

According to the famous quote of Shannon, *breaking a secure cipher should require as much work as solving a system of simultaneous equations in a large number of unknowns of a complex type.* Any stream cipher can be modeled as a system of algebraic equations, where the solution of the binary nonlinear system gives the secret key.

Let

$$F_i : \mathbb{F}_2^k \times \mathbb{F}_2^v \to \mathbb{F}_2,\ i = 1, 2, \ldots \tag{4.0.1a}$$

$$(K, IV) \to z_i \tag{4.0.1b}$$

where each F_i produces the i^{th} keystream bit z_i using k bit key and v bit IV.

In this chapter, we aim to analyze the security of stream ciphers based on the cryptographic properties of the Boolean functions, F_i's. These properties are also utilized in algebraic attacks against stream ciphers (See Section 4.1). First, we study some classical designs and experimentally observe the distribution of some cryptographic properties such as nonlinearity, number of linear terms, degree *etc.* for small sized ciphers. In Section 4.4, we present a case study for eSTREAM candidate Trivium and obtain a linear approximation for F_1 with bias 2^{-31} which is valid for a subset of keys and IVs with 288 initial clocking. In Section 4.5, we consider the number of monomials of degree d and propose two new tests that are variants of the tests described by Filiol [20] and analyze the security of Grain, Trivium and Decim. We also propose the *linear span* test that measures the inheritance of linear dependence of input IVs to keystream bits. Finally, to analyze the completeness property of the ciphers, we apply the *diffusion test* that measures the effect of each key and IV bit on keystream bits.

4.1 Basics of Algebraic Attacks

The idea of algebraic attacks is to find the equations of unknown key bits and to solve these equations. The steps of an algebraic attack are (i) finding a system of equations in terms of keystream bits and the key bits, (ii) reducing the degree of equations, (iii) obtaining keystream bits, and (iv) recovering the secret key by solving the system of equations.

The problem of solving equations is NP-hard, to solve the problem many algorithms based on linearization such as XL (eXtended Linearilization), XSL (eXtended Sparse Linearilization) [123] or using Gröbner bases are proposed. It should be noted that having a large size system and high number of variables does not mean that the system is hard to solve. The difference between the number of equations and the number of monomials is what makes the system harder. In some situations it is seen that big systems with multivariate equations are solved efficiently. Some systems with over defined or sparse can be solved easier than expected by the XL method [124] and XSL method [123]. Therefore, the systems of equations of a cryptosystem should behave randomly in terms of number of monomials and other cryptographic properties.

Analysis of filter generators using algebraic attacks is given in [125]. The formulation of the attack is as follows.

$$\begin{aligned} z_0 &= f(K) \\ z_1 &= f(L(K)) \\ z_2 &= f(L^2(K)) \\ &\vdots \end{aligned} \tag{4.1.1a}$$

where L is the linear next state function of the LFSR and f is the nonlinear filter function. In this formulation, the aim is to recover the secret key using the equations.

In practical filter generators, the degree of f is chosen to be so high that it is not possible to solve the systems of equations efficiently. In [125], a new idea of multiplying f with a properly chosen polynomial g such that $f \cdot g$ has low degree is proposed. This gives a new system of equations in terms of state bits, that can be solved efficiently if sufficiently many keystreams exist where the system of equations is rewritten as

$$z_t \cdot g(L^t(K)) = f(L^t(K)) \cdot g(L^t(K)). \tag{4.1.2}$$

The idea of algebraic attacks is also extended to attack combining generators in [126]. Let x_t^i be the output of LFSR i at time t. Then, the system of equations can be

written as;

$$\begin{aligned} z_0 &= f(x_0^1, \ldots, x_0^l) = F_1(K) \\ z_1 &= f(x_1^1, \ldots, x_1^l) = F_2(K) \\ &\vdots \end{aligned}$$

Since in combining generators the internal state variables are updated linearly, the degree d of F_i's are equal to the degree of f. After the linearization process, a system of linear equations with around $\binom{n}{d}$ unknowns are obtained. The required keystream is around n^d bits, whereas the required number of operations is approximately n^{dw} where $w < 3$. In [127], a more general theorem is given and it is shown that fast algebraic attacks exist for any cipher that outputs several bits at a time, this method is applied on modified versions of Snow, E0, LILI-128 and Turing.

Other examples for successful algebraic attacks can be given as (i) Toyocrypt using 2^{49} CPU clocks and 20 Kbytes of keystream [125], (ii) LILI-128 using 2^{57} CPU clocks [125], (iii) Sfinks with 2^{70} computation and 2^{43} keystream bits [128].

4.2 Desired Properties of F_i's

Each cipher design can be treated as a subset of Boolean functions F_i's of $k + v$ variables where k and v are the key and IV size, respectively. There are $2^{2^{k+v}}$ many Boolean functions of $k + v$ input. Ideally, the subset that simulate the cipher should be randomly selected among this set.

The cryptographic properties of F_i's are closely related to the security of the stream cipher and any cryptographic weaknesses of these Boolean functions may be exploited to distinguish the keystream or to recover a part of secret key. If for any F_i,

$$Pr(F_i(K, IV) = 1) = Pr(z_i = 1) = 1/2 + \epsilon \tag{4.2.1}$$

where $\epsilon \neq 0$, then F_i is *not balanced*. The value of z_i can be predicted with success probability depending on the amount of bias. Therefore, any linear combination of F_i's must be balanced for a secure stream cipher.

Nonlinearity of F_i's is also very important for the security of the stream ciphers. Existence of a linear F_i has catastrophic results, using the related keystream bit, one bit key information can be recovered by solving this linear equation. For functions that have very low nonlinearity, similar results can be obtained. If there exists a linear

function L_i such that

$$Pr(F_i = L_i) = 1/2 + \epsilon, \tag{4.2.2}$$

then the nonlinearity of F_i, N_{F_i} is at most $2^n(\frac{1}{2} - \epsilon)$ for a n variable Boolean function.

Similarly, if there exists *non-complete* F_i's, in other words not all input bits affect output, then it may be possible to mount guess and determine or algebraic attacks. If a subset of key bits is not used, this may be used to reduce the search space.

Correlation between any two F_i's

$$Pr(F_i(K) = F_{i+\delta}(K)) = \frac{1}{2} + \epsilon \tag{4.2.3}$$

where $\delta \neq 0$, means that corresponding z_i's are also correlated, *i.e.*

$$Pr(z_i = z_{i+\delta},) = \frac{1}{2} + \epsilon. \tag{4.2.4}$$

By converting the sequences to $Z_i^* = z_i \oplus z_{i+\delta}$, the output keystream may be distinguished from random streams depending on the magnitude of ϵ. Therefore, another condition for F_i's is that they should be *independent* of each other.

4.3 Analyzing Classical Designs

In most of the classical stream ciphers, linear next state functions are used to guarantee high period and good statistical properties. Especially, maximum length LFSRs are used as building blocks in such designs.

Here, we present some results on the balancedness, degree and nonlinearity of F_i's for some classical stream cipher designs such as nonlinear combiner, nonlinear filter and clock controlled stream ciphers. Since there is no generic IV loading schemes for these classical designs, we consider state bits as input variables.

Consider a stream cipher of state size n. Let $F_i : \mathbb{F}_2^n \to \mathbb{F}_2$ and $f : \mathbb{F}_2^l \to \mathbb{F}_2$ be the filter function that generates keystream using the current state variables. Let $\varphi : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be the next state function of the stream cipher.

Proposition 4.3.1. *The nonlinearity of F_1 is determined by the nonlinearity of the keystream generation (filter) function f. For a $l-$bit filter function with nonlinearity N_f, the nonlinearity of F_1 is equal to $2^{n-l}N_f$. If the next state mapping is linear, then $N_{F_i} = N_{F_1}$ for all i.*

Proof. The filter function f is from $\mathbb{F}_2^l$ to $\mathbb{F}_2$. Without loss of generality,

$$\begin{aligned} F_1 : \mathbb{F}_2^l \times \mathbb{F}_2^{n-l} &\to F_2 \\ (x, y) &\to f(x) \end{aligned}$$

The Walsh spectrum of F_1 can be written as;

$$\begin{aligned} W_{F_1}(\gamma) &= W_{F_1}(\alpha, \beta), \\ &= \sum_{(x,y) \in F_2^l \times F_2^{n-l}} (-1)^{F_1(x,y) + <(\alpha,\beta),(x,y)>}, \\ &= \sum_{(x,y) \in F_2^l \times F_2^{n-l}} (-1)^{F_1(x,y) + <(\alpha,x)> + <(\beta,y)>}, \\ &= \sum_{(x,y) \in F_2^l \times F_2^{n-l}} (-1)^{f(x) + <(\alpha,x)> + <(\beta,y)>}, \\ &= \sum_{x \in F_2^l} (-1)^{f(x) + <(\alpha,x)>} \cdot \sum_{y \in F_2^{n-l}} (-1)^{<(\beta,y)>}, \\ &= W_f(\alpha) \cdot \delta_0^\beta 2^{n-l} \end{aligned}$$

where $\gamma = (\alpha, \beta) \in F_2^l \times F_2^{n-l}$ and the Kronecker delta function $\delta(\beta)$ is equal to 1 only when $\beta = 0$ and zero otherwise. Then, we obtain

$$W_{F_1}(\gamma) = \begin{cases} 2^{n-l} W_f(\alpha) & \text{if } \beta = 0 \\ 0 & \text{if } \beta \neq 0. \end{cases}$$

Here it follows that,

$$N_{F_1} = 2^{n-l} N_f. \tag{4.3.3}$$

For any Boolean function F and a nonsingular linear transformation $\varphi : \mathbb{F}_2^n \to \mathbb{F}_2^n$, we have $N_{F \circ \varphi} = N_F$. By definition,

$$F_2 = F_1 \circ \varphi, F_3 = F_1 \circ \varphi^2, \ldots, F_i = F_1 \circ \varphi^{i-1}. \tag{4.3.4}$$

Since φ^i is a linear mapping, it follows that

$$N_{F_i} = N_{F_1} = 2^{n-l} N_f \text{ for } i = 2, 3, \ldots \tag{4.3.5}$$

□

Proposition 4.3.2. *Weight(F_1) is equal to $2^{n-l} \cdot weight(f)$. If the next state mapping is one to one, weights of all F_i are same, in particular, F_i's are all balanced if and only if f is balanced.*

Proof. From the previous proof, we have $W_{F_1}(0) = 2^{n-l} \cdot W_f(0)$ and it is known that for a Boolean function g, $weight(g) = (2^n - W_g(0))/2$ holds. Hence, $weight(F_1) = 2^{n-k} weight(f)$ holds.

If the next state mapping $\varphi : F_2^n \to F_2^n$ of a stream cipher is one to one, then it defines a permutation on the set F_2^n. It is known that for any permutation $\varphi : F_2^n \to F_2^n$ and a Boolean function g, we have $weight(g) = weight(g \circ \varphi)$. Hence,

$$weight(F_i) = weight(F_1) = 2^{n-k} weight(f). \tag{4.3.6}$$

Then, it is obvious that F_i's are balanced if and only if f is balanced. □

Proposition 4.3.3. *If the next state function of a stream cipher is linear, then the degree of F_i's are equal to the degree of keystream generation function.*

Proof. It is obvious by definition that the degree of F_1 is equal to the degree of f. If the next state function is linear, then the next state mapping φ is also linear. It is known that $deg(F \circ \varphi) = deg(F)$. □

Some Small Sized Examples

In this part of the section, we present six small sized stream cipher examples where we can evaluate the ANFs of F_i's, efficiently. First, we start with classical LFSR based ciphers; nonlinear filter and nonlinear combiner, then repeat the same experiments using NFSRs with period $2^n - 1$ instead of maximum length LFSRs. Finally, we give two clock controlled examples; shrinking and self-shrinking generators. The details of the example ciphers are presented in Table 4.1.

All of the given examples have state size of 20, therefore it is possible to calculate the cryptographic properties of F_i's efficiently. Considering the first 40 Boolean functions, we compared the degree, weight, nonlinearity and number of linear and nonlinear terms. These figures are presented in Table 4.2. It is observed that for clock controlled ciphers (Example 5 and 6), the balancedness of F_i's is not achieved. In Example 5, the weight of the F_i's are fixed to 524032, in other words,

$$Pr(F_i = 1) = \frac{1}{2} - \frac{256}{1048576} = 1/2 - 0.00245 \tag{4.3.7}$$

holds. In Example 6, different weights are obtained with the average 524274.7 and the standard deviation 693.2267. Degree of n bit Boolean functions is expected to be around $n - 1$ and this is only achieved by self-shrinking generator considering the first 40 Boolean functions, all other examples have smaller degree. When we compare

Table 4.1: Details of the six small sized examples

Examples	*Parameters*
1. Nonlinearly Filtering of a primitive LFSR of size 20	Recursion: $x_{i+20} = x_{i+19} + x_{i+15} + x_{i+4} + x_i$ Filter function: $f(x_1, \ldots, x_{20}) = x_1 + x_3 + x_7 + x_{12} + x_{15} + x_2.x_4 + x_5.x_8 + x_6.x_9.x_{14} + x_{10}.x_{11}.x_{13}.x_{16}$
2. Nonlinear Combining of 3 LFSRs of size 5, 7 and 8	Recursions: $x_{i+5} = x_i + x_{i+3}$ $x_{i+7} = x_i + x_{i+4} + x_{i+5} + x_{i+6}$ $x_{i+8} = x_i + x_{i+3} + x_{i+5} + x_{i+7}$ Combining function: $f(x_1, x_2, x_3) = x_1 + x_2 + x_1.x_2 + x_2.x_3$
3.Nonlinearly Filtering of an NFSR of size 20	Recursion: $g(x_1, \ldots, x_{20}) = x_1 + x_{18} + x_{19} + x_{20} + x_7x_9 + x_8x_{16}$ Filter function: $f(x_1, \ldots, x_{20}) = x_1 + x_3 + x_7 + x_{13} + x_{19} + x_2.x_4 + x_5.x_8 + x_6.x_9.x_{15} + x_{11}.x_{12}.x_{14}.x_{20}$
4.Nonlinear Combining of 3 NFSRs of size 5, 7 and 8	Recursions: $x_{i+5} = x_i + x_{i+1} + x_{i+2} + x_{i+4} + x_{i+1}x_{i+2} + x_{i+1}x_{i+4}$ $x_{i+7} = x_i + x_{i+6} + x_{i+1}x_{i+2} + x_{i+2}x_{i+3}x_{i+6}$ $x_{i+8} = x_i + x_{i+7} + x_{i+4}x_{i+6} + x_{i+5}x_{i+6}$ Combining function: $f(x_1, x_2, x_3) = x_1 + x_2 + x_1.x_2 + x_2.x_3$
5. Shrinking Generator using two LFSRs of size 11 and 9	Recursions: $x_{i+11} = x_i + x_{i+3} + x_{i+5} + x_{i+9}$ $x_{i+9} = x_i + x_{i+5}$
6. Self-shrinking Generator using a LFSR of size 20	Recursion: $x_{i+20} = x_{i+17} + x_{i+15} + x_{i+11} + x_i$

the average nonlinearity values, it is seen that filtering generators (Examples 1 and 3) achieve higher nonlinearity compared to combining generators (Examples 2 and 4). It is observed that NFSR based systems have higher degree and nonlinearity compared to LFSR based ciphers.

The number of linear terms which is expected to be around $n/2 = 10$, is only achieved by filter generators (Examples 1 and 3). It is interesting to observe that shrinking generator in Example 5 does not include any linear terms. Finally, when we consider the number of all terms which is expected to be $\frac{1}{2}\binom{n}{2} = 522248$. It is only achieved using the self-shrinking generator in Example 6.

Table 4.2: Average cryptographic properties obtained using the first 40 Boolean functions

Example	*Balancedness*	*Average Degree*	*Average Nonlinearity*	*Average # of Linear Terms*	*Average # of Terms*
1. Nonlinear Filter	Yes	4 (fixed)	438272 (fixed)	9.8	1556.9
2. Nonlinear Combiner	Yes	2 (fixed)	262144 (fixed)	5.4	22.9
3. Nonlinear Filter	Yes	13.0	505842.2	11.7	238139.9
4. Nonlinear Combiner	Yes	10.3	351436.8	5.4	5711.8
5. Shrinking	No, 524032 (fixed)	12 (fixed)	475161.6	0	9086.5
6. Self-shrinking	No, 524274.7	19.4	477556.9	7.7	524164.9
Expected values	Yes	19	523776 (maximum)	10	524288

Irregular clocking significantly improves the total number of monomials. However, it should be used carefully in designs, since it is hard to guarantee balancedness of the keystream which is very important criteria. Additionally, as observed from the examples, using linear next state functions should be avoided to obtain functions with high degree.

4.4 A Case Study on Trivium

Trivium [32], one of the focus algorithms in eSTREAM project, is a synchronous binary additive stream cipher. Previously, two attacks have been presented for the analysis of Trivium and none of them has complexity less than exhaustive search. In [98], the linear sequential circuit approximations are used to evaluate the strength of Trivium against distinguishing attacks. The correlation coefficient is calculated as 2^{-72}, and the complexity to distinguish the output is $O(2^{144})$. According to the paper, it is not possible to find a linear function with correlation coefficient larger than 2^{-40} using linear sequential circuit approximations. Another study [129] tries to find 288 unknown internal state bits by solving systems of equations. Solving this system has complexity $O(2^{164})$, which is more than exhaustive search complexity. Also, in terms of randomness properties, no statistical weaknesses are observed [23].

We concentrate on the initialization of Trivium which consists of 1152 clockings. Although Trivium is one of the fastest ciphers proposed in eSTREAM project [130], its initialization with 1152-clockings may hinder the speed in platforms where resynchronization is performed very often. For instance, frequent initializations of Trivium may slow it down more than five times in a frame based encryption like GSM over-the-air privacy standard since length of each frame is 228 bits. We propose a new input to the initialization function of Trivium which provides a faster diffusion of key bits and IV bits into the register. Moreover, we introduce some open problems on security margins of Trivium.

4.4.1 Description of Trivium

Trivium supports the usage of 80 bit key and 80 bit IV with internal state size of 288 bits. It is claimed to be suitable to generate up to 2^{64} bits of keystream from a pair of key and IV.

Initialization

80 bit key K $(k_1, k_2, \ldots, k_{80})$, and IV $(iv_1, iv_2, \ldots, iv_{80})$, is directly assigned to the internal state of the cipher $(s_1, s_2, \ldots, s_{288})$ and the remaining bits (except the last three) are set to zero. Then, the cipher is clocked over 4 full cycles without producing any output. The pseudocode of the initialization phase is given in Algorithm 4.4.1.

Algorithm 4.4.1: LOADING KEY AND IV(K, IV)

$(s_1, s_2, \ldots\ s_{93}) \leftarrow (k_1, k_2, \ldots, k_{80}, 0, \ldots, 0)$;
$(s_{94}, s_{95}, \ldots\ s_{177}) \leftarrow (iv_1, iv_2, \ldots, iv_{80}, 0, \ldots, 0)$;
$(s_{178}, s_{179}, \ldots\ s_{288}) \leftarrow (0, \ldots, 0, 0, 1, 1, 1)$;
for $i \leftarrow 1$ **to** $4 \cdot 288$

$t_1 \leftarrow s_{66} + s_{91}.s_{92} + s_{93} + s_{171}$;
$t_2 \leftarrow s_{162} + s_{175}.s_{176} + s_{177} + s_{264}$;
$t_3 \leftarrow s_{243} + s_{286}.s_{287} + s_{288} + s_{69}$;
$(s_1, s_2, \ldots\ s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$;
$(s_{94}, s_{95}, \ldots\ s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$;
$(s_{178}, s_{179}, \ldots\ s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$;

Keystream Generation

Keystream generation function is very similar to key and IV loading. The only difference is the filter function that generates the keystream z_i, $i = 1, 2, \ldots$. The pseudocode of keystream generation is given in Algorithm 4.4.2.

Algorithm 4.4.2: KEYSTREAM GENERATION(N)

for $i \leftarrow 1$ **to** N

$t_1 \leftarrow s_{66} + s_{93}$;
$t_2 \leftarrow s_{162} + s_{177}$;
$t_3 \leftarrow s_{243} + s_{288}$;
$z_i \leftarrow t_1 + t_2 + t_3$;
$t_1 \leftarrow t_1 + s_{91}.s_{92} + s_{171}$;
$t_2 \leftarrow t_2 + s_{175}.s_{176} + s_{264}$;
$t_3 \leftarrow t_3 + s_{286}.s_{287} + s_{69}$;
$(s_1, s_2, \ldots\ s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$;
$(s_{94}, s_{95}, \ldots\ s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$;
$(s_{178}, s_{179}, \ldots\ s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$;

4.4.2 Linear Approximations

Linear cryptanalysis, introduced by Matsui [131], is an effective known plaintext attack against block ciphers. It exploits some statistical correlations between input and output bits. For a block cipher with k-bit key $(k_1, \ldots, k_k)$, n-bit plaintext $(p_1, \ldots, p_n)$ and ciphertext $(c_1, \ldots, c_n)$, the aim of the attack is to find the index sets I, J, L such that

$$\sum_{i \in I} k_i + \sum_{j \in J} p_j = \sum_{l \in L} c_l \tag{4.4.1}$$

holds with probability $p = 1/2 + \epsilon$, $\epsilon \neq 0$.

Some variants of linear cryptanalysis are applied to stream ciphers. The most famous example may be the correlation attacks mounted on LFSR based stream ciphers [132, 133, 134] where some linear approximations between key and keystream bits are utilized. Another example is proposed by Golić [135]. In [136], a linear approximation for $t-$functions is used to attack the TSC stream ciphers. In [137], a new method to find biased linear approximations without searching all possible linear relations individually is presented and it is used to distinguish the output of the stream cipher Pomaranch.

Here, we introduce a new version of linear cryptanalysis on stream ciphers. The analysis is a kind of resynchronization attack. We consider the initialization phase (key and IV loading) of a stream cipher as an iterated function. Then, we apply Matsui's linear cryptanalysis to the initialization by finding approximations for each iteration

and combining them by piling-up lemma. As an example, we consider the initialization phase of Trivium as 8-round function and find a linear approximation for 2-round Trivium with a bias of 2^{-31} for a subset of key and IVs.

In the following section, we give a framework for finding linear approximations and present a linear approximation for 2-round Trivium.

4.4.3 Searching for Linear Approximations

Searching for linear approximations is composed of three steps:

(i) **Selecting a subset of z_i.** In this step, our aim is to find the right hand side of our linear approximation. Selection of the subset of z_i's or equivalently the subset of F_i's is done such that $\sum z_i$ or $\sum F_i$ is affected from minimum number of internal state variables.

(ii) **Partitioning the initialization phase.** Initialization phase of stream ciphers consists of iteratively applying the same next state function to the internal state variables. To find a linear approximation for F_i efficiently, the initialization is partitioned into rounds with t_i clockings so that for each round, it is possible to find approximations efficiently.

As given in Figure 4.1, the initialization phase of the cipher can be represented as n rounds, where t_i is the number of clockings in each round. The sum of t_i's should be equal to the total number of clockings, T, in initialization.

In the extreme case, each round is composed of one clockings, then for each round, linear approximations with high biases can be found easily. However, as T gets large, the approximation for the whole cipher is likely to have a very small bias. On the other hand, if the number of clockings in each round is chosen to be very high, then finding linear approximations for each round becomes infeasible. This gives a trade-off between number of rounds and the selection of t_i values. Optimal selection of t_i values is an open question.

(iii) **Combining linear approximations.** The approximations found for each round are combined to find an approximation for the whole cipher. Approximations are selected so that all internal state bits are canceled out. Finally, the linear approximation based on key, IV and keystream is obtained. Bias of the approximation is found by using the piling-up lemma.

Lemma 4.4.1. *For n independent random variables $X_1, \ldots, X_n$ that take values from $\{0,1\}$, the summation given by $X = X_1 + X_2 + \ldots X_n$ has bias $\epsilon = p - 1/2$ is given by*

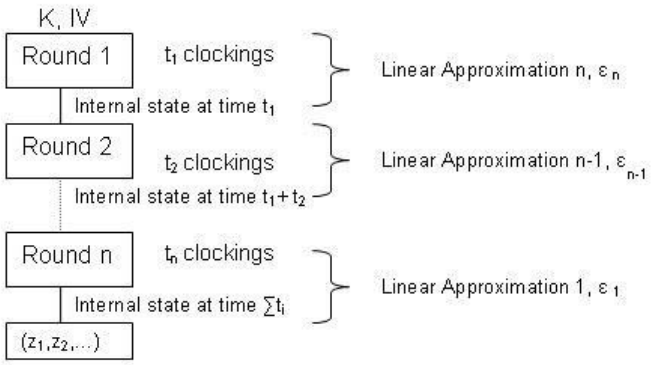


Figure 4.1: Linear approximations for n round stream ciphers

$$\epsilon = 2^{n-1} \prod_{j=1}^{n} \epsilon_j \tag{4.4.2}$$

where ϵ_i's are biases of the terms X_i's.

To attack the cipher, the found bias should be greater than $2^{-k/2}$. It is also possible to fix some of the IV bits to increase the bias, however this puts a restriction on the number of resynchronizations. In such cases, the attack requires chosen IVs. There may be other approximations valid for a subset of keys, with better biases.

4.4.4 Linear Approximations for 2-round Trivium

The initialization of Trivium can be modeled as n disjoint rounds. As a result of making a trade-off between number of rounds and number of clockings in each round, the number of clockings in each round is chosen to be 144. The more number of clockings in each round, the better approximations are found, but number of clockings should not be too large to prevent finding linear approximations exhaustively.

For 2-round Trivium, the followings

$$\begin{aligned} K &= (s_0(1), s_0(2), \ldots, s_0(80)), \\ IV &= (s_0(94), s_0(95), \ldots, s_0(173)), \\ z_1 &= s_{288}(66) + s_{288}(93) + s_{288}(162) + s_{288}(177) + s_{288}(243) + s_{288}(288) \end{aligned}$$

hold where $s_t(i)$ is the i^{th} internal state bit at time t.

To check for possible trivial weaknesses of 2-round Trivium, diffusion of IV and key to internal state bits are examined using 1000 random key and IV pairs. All key and

IV bits are diffused to all internal state bits, except the last seven internal state bits. However, this does not lead to any trivial attack.

While selecting the subset of F_i's to approximate, the only restriction is the total number of internal state variables that affect the keystream bits. Each z_i is generated using the modulo 2 summation of six internal state bits. A subset of z_i's such that their summation includes less than six internal state bits after cancellations is not found. Therefore, the function to be approximated is chosen to be F_1 that generates the first output bit, z_1, in terms of key and IV bits.

For 2-round Trivium, finding the equation F_1 in terms of initial state variables is not efficient, therefore an approximation is found for each round and then they are combined to find an approximation as given in Figure 4.2. The bias of the obtained approximation is found by the piling-up lemma.

The output bit z_1 is the modulo sum of bits $s_{288}(66)$, $s_{288}(93)$, $s_{288}(162)$, $s_{288}(177)$, $s_{288}(243)$ and $s_{288}(288)$. The algebraic normal form of F_1 is found exhaustively in terms of the internal state bit of $t = 144$ as

$$\begin{aligned} z_1 \;=\; & s_{144}(6) + s_{144}(16).s_{144}(117) + s_{144}(31)s_{144}(32) + s_{144}(33) + s_{144}(57) + \\ & s_{144}(82).s_{144}(83) + s_{144}(84) + s_{144}(96) + s_{144}(97).s_{144}(98) + s_{144}(99) + \\ & s_{144}(111) + s_{144}(129) + s_{144}(142).s_{144}(143) + s_{144}(144) + s_{144}(150) + \\ & s_{144}(162) + s_{144}(163).s_{144}(164) + s_{144}(165) + s_{144}(186) + s_{144}(192) + \\ & s_{144}(208).s_{144}(209) + s_{144}(210) + s_{144}(231) + s_{144}(235).s_{144}(236) + \\ & s_{144}(237) + s_{144}(252) \end{aligned}$$

and its closest linear approximation is

$$\begin{aligned} z_1 \;=\; & s_{144}(6) + s_{144}(33) + s_{144}(57) + s_{144}(84) + s_{144}(96) + s_{144}(99) + \\ & s_{144}(111) + s_{144}(129) + s_{144}(144) + s_{144}(150) + s_{144}(162) + \\ & s_{144}(165) + s_{144}(186) + s_{144}(192) + s_{144}(210) + s_{144}(231) + \\ & s_{144}(237) + s_{144}(252) \end{aligned} \tag{4.4.3}$$

with bias $1/2 + 2^{-9}$.

Since the aim is to obtain an approximation based on key, IV and output bits, the linear approximation given above is rewritten in terms of $s_0(i)$, $i = 1, 2, \ldots, 80$ and $i = 94, \ldots, 173$ values, the remaining terms are omitted, since they are assigned to

constants during initialization. Then, the equation given in Appendix D is obtained. The equation has 24 linear, 59 quadratic terms, 20 terms with degree 3. The linear approximation for the function is found as

$$\begin{aligned} z_1 \;=\; & 1 + s_0(3) + s_0(6) + s_0(15) + s_0(21) + s_0(27) + s_0(30) + s_0(39) + \\ & s_0(54) + s_0(57) + s_0(67) + s_0(68) + s_0(69) + s_0(72) + s_0(96) + \\ & s_0(99) + s_0(114) + s_0(117) + s_0(123) + s_0(126) + s_0(132) + s_0(138) + \\ & s_0(144) + s_0(165) + s_0(171) \end{aligned} \tag{4.4.4}$$

with bias $2^{78}\cdot(0.25)^{59}\cdot(0.375)^{20} = 2^{-68.30}$, assuming all nonlinear terms are independent. We increase the amount of the bias by assigning zero string to certain IV and key bits.

Chosen IVs

For IVs in the form $iv_{25} = iv_{26} = iv_{31} = iv_{32} = iv_{49} = iv_{50} = iv_{54} = iv_{55} = iv_{70} = iv_{71} = 0$, the bias of the equation increases to 2^{-44}. This bias is still very low and cannot be used to break 2-round Trivium. To improve bias further, also some of the key bits are fixed. Then, the bias of the second linear approximation increases to 2^{-23} for keys satisfying $k_{14} = k_{19} = k_{20} = k_{38} = k_{39} = k_{45} = k_{63} = k_{64} = k_{65} = k_{77} = 0$.

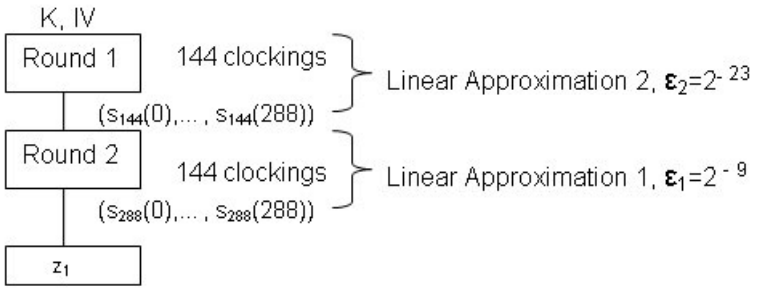


Figure 4.2: Linear approximations for 2-round Trivium

Combining two linear approximations (4.4.3) and (4.4.4), the total bias of the following approximation,

$$\begin{aligned} z_1 \;=\; & 1 + k_3 + k_6 + k_{15} + k_{21} + k_{27} + k_{30} + k_{39} + k_{57} + k_{67} + k_{68} + \\ & k_{69} + k_{72} + iv_3 + iv_6 + iv_{21} + iv_{24} + iv_{30} + iv_{33} + iv_{39} + iv_{45} + \\ & iv_{51} + iv_{72} + iv_{78}, \end{aligned}$$

is obtained as $2 \cdot 2^{-9} \cdot 2^{-23} = 2^{-31}$ by piling-up lemma. The upper bound on the number of resynchronizations is 2^{70}, since 10 bits of IV bits are fixed to zero. To identify a key with specified bits, we need 2^{62} chosen IV.

Proposal for Initialization

In this section, we propose a new method for initialization which is very similar to the original. The only difference is related to the initial assignment of state bits. Only 22 of the internal state variables are set to constants and this change does not increase the cost significantly. This obviously increases the number of variables while searching for linear approximations and therefore it gets harder to find linear approximations. As a result, it may be possible to decrease the number of initial clockings.

The proposed initial assignment is

$$\begin{aligned}
(s_1, \ldots, s_{93}) &\leftarrow (iv_1, \ldots, iv_{13}, iv_{14} + k_1, \ldots, iv_{80} + k_{67}, k_{68}, \ldots k_{80}), \\
(s_{94}, \ldots, s_{177}) &\leftarrow (iv_1 + k_1, \ldots, , iv_{80} + k_{80}, 0, 0, 0, 0), \\
(s_{178}, \ldots, s_{288}) &\leftarrow (k_1, \ldots, k_{13}, k_{14} + iv_1, \ldots, k_{80} + iv_{67}, iv_{68}, \ldots iv_{80}, 0, \ldots, 0, 1, 1, 1).
\end{aligned}$$

Let us note that we propose 13 shifts while loading key bits into the first register and 13 shifts while loading IV bits into the third register. The number of shifts may be chosen something else. However, same key bits or same IV bits should not be XORed in the feedback functions of the registers during the first 80-90 clockings.

A comparison of the proposed and original method is done in terms of the completeness property. Let $G(i, j)$ be the number of key and IV bits that affect the state bit i after j clockings. The comparison of both methods is done based on $\min_i G(i, j)$ and as seen from Figure 4.3 the diffusion of key and IV bits are better in the proposed method. In the original method, completeness is satisfied after 525 clockings, whereas in the proposed method, 484 clockings are enough.

4.4.5 Discussion

In this part of the chapter, we mainly concentrate on the initialization of Trivium which is one of the focus ciphers of eSTREAM project. We model the initialization phase of Trivium as an iterated cipher with 8 rounds. For frame based applications requiring frequent resynchronizations, we question the efficiency of the initialization phase and try to attack initialization with smaller rounds. For 2-round Trivium, we obtain a linear approximation of z_1, which is valid for a subset of key and IV's.

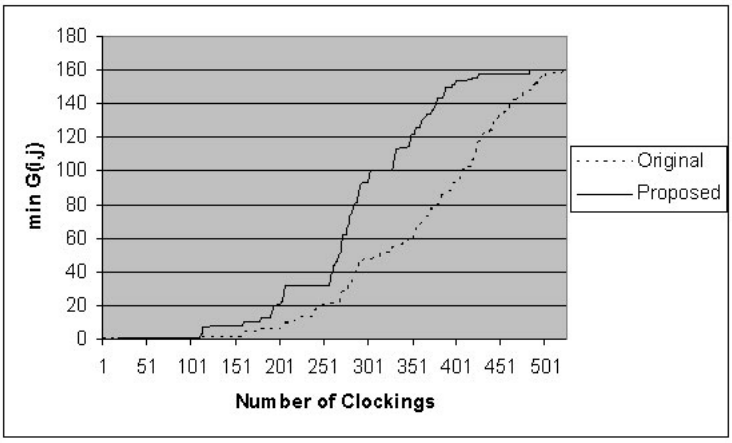


Figure 4.3: Number of clockings vs. $\min_i G(i,j)$ for original and proposed initialization of Trivium

As a list of future studies, the followings can be given:

- *Use of multiple approximations.* As an extension of linear cryptanalysis, in [138], the use of multiple linear approximations is proposed. As a future work, different approximations for the same output bits can be found and combined to make better approximations.
- *Use of nonlinear approximations.* As an alternative to linear approximations, nonlinear approximations may be applied to the initialization phase to find better approximations.
- *Different modelings of initialization.* The initialization phase can be remodeled differently, using different number of clockings in each round and better approximations may be found.

4.5 d-Monomial Approach

In the d-monomial approach, the aim is to analyze ciphers using the number of monomials of degree d in F_i's. In an ANF of a randomly chosen F_i, each monomial should occur with probability $\frac{1}{2}$ which is equivalent to the Bernoulli process and the total number of all monomials M follow the Binomial distribution with parameters 2^n and $p = 1/2$ for a n bit Boolean function F_i. Let the number of monomials of degree d be denoted by M_d, *i.e.*, $M = \sum_{d=0}^{n} M_d$. The distribution of M_d is $\text{Bin}(\binom{n}{d}, \frac{1}{2})$ with expected value of $E(M_d) = \frac{1}{2}\binom{n}{d}$.

Analyzing cryptosystems based on the number of monomials is first proposed by

Filiol [20]. Filiol defined *affine constant* and *d-monomial tests*. The affine constant test focuses on the number of F_i's $i = 1, 2, \ldots, N$ that include the affine constant a_0. For large N, the distribution of the number of F_i's with a_0 is Normal with parameters $N/2$ and $N/4$. The d-monomial tests only consider the number of terms with degree d. Two different variation of the tests T_1^d and T_2^d are proposed, T_1^d considers first N F_i's with input d, computes the number of terms with degree d in N ANFs, then performs a χ^2 test with $N-1$ degrees of freedom, whereas T_2^d considers first N F_i's and categorizes the number of monomials with degree d to v groups. Then, again χ^2 test with v degrees of freedom is applied.

Instead of working on key bits, Saarienen [139] proposed d-monomial IV distinguishers based on the work of [20]. The behavior of the keystream is analyzed using a function of n IV bits, *i.e.*, $z = f(iv_0, \ldots, iv_{n-1})$. All other IV and key bits are considered to be constants. In Saarinen's scheme, the number of terms with degree d is counted and a χ^2-test with 1 degree of freedom is applied. In the bit flip test, given a vector $\mathbf{b}$ with weight 1, the number of occurrences where $f(x) = f(x + b)$ is counted and a χ^2-test with 1 degrees of freedom is applied. The monomial tests are summarized in Table 4.3. A similar approach called *Algebraic Structure Defectoscopy* is given in [140].

Table 4.3: Comparison of d-monomial tests

Test Name	Input	Number of Keystream	Distribution	Description
Filiol's Affine Constant Test	-	N	Normal(N/2,N/4)	Counts the number of affine terms in N ANFs
Filiol's d-monomial T_1^d	d	N	χ^2 with $N-1$ degrees of freedom	Counts the number of terms with degree d in N ANFs.
Filiol's d-monomial T_2^d	d	N	χ^2 with v degrees of freedom, $2 \leq v \leq 9$	Categorizes the number of monomials with degree d into v groups.
Saarinen's d-monomial test	d	1	χ^2 with 1	Counts the number of terms with degree d
Saarinen's bit flip test	d	1	χ^2 with 1	Counts the number of occurrences where $f(x) = f(x + b)$

Here, we give a classical example of d-monomial IV distinguisher. First, we find the number of monomials for each d and evaluate the result using $n+1$ degrees of freedom. The algorithm for the d-Monomial test is summarized in Algorithm 4.5.1.

Algorithm 4.5.1: d-MONOMIAL TEST(d)

for $iv \leftarrow 1$ **to** $2^n - 1$
$\begin{cases} \text{Initialize cipher with IV;} \\ v[iv] = \text{First keystream bit after initialization;} \end{cases}$
Compute ANF of vector v and store result in v;
for $i \leftarrow 1$ **to** $2^n - 1$
$\begin{cases} \textbf{if } v[i] = 1 \\ \quad \begin{cases} weight = \text{weight of monomial } i; \\ distr[weight] + +; \end{cases} \end{cases}$
for $d \leftarrow 1$ **to** n
$\begin{cases} \chi^2 += \frac{(\text{distr}[d] - \frac{1}{2}\binom{n}{d})^2}{\frac{1}{2}\binom{n}{d}}; \end{cases}$
if $\chi^2 > \chi^2(1-\alpha; n+1)$
  then *return* cipher;
  else *return* random;

The complexity of this attack is $O(n \log n)$ operations and it needs memory $O(n \log n)$. The downside of this method is that statistical deviations for lower and higher degree monomials are hard to detect since their numbers are few. So even if the maximal degree monomial never occurs, the test does not detect this anomaly. In the next section, we present alternative attacks that solves this problem.

4.5.1 A Generalized Approach

As a general approach, instead of analyzing just one function in ANF form, we can study the behavior of more polynomials so that monomials that are more (or less) probable than others can be detected.

Let us select n IV values, denoted $iv_0, \ldots, iv_{n-1}$, as our *variables*. The remaining IV values as well as key bits are kept constant. Using the first output symbol, $z_1 = f_1(iv_0, \ldots, iv_{n-1})$, for each choice of $iv_0, \ldots, iv_{n-1}$, the ANF of f_1 can be constructed.

The new approach is to use some other choice on IV values outside the IV variables $iv_0, \ldots, iv_{n-1}$. Running through each choice of IV variables in this case gives us a new function f_2. Continuing in this way, we derive the ANFs of P different Boolean functions $f_1, f_2, \ldots, f_P$. In some situations, it might also be possible to obtain polynomials from different keys, where the same IVs have been used.

Having P different polynomials in our possession we can now design any test that

looks promising, taken over all polynomials. The d-monomial test would appear for the special case $P = 1$ and the test being counting the number of weight d monomials. We now propose in detail two different tests.

4.5.2 Monomial Distribution Test

The attack scenario is similar to the d-monomial test, but instead of counting the number of monomials of a certain degree, we use P polynomials and calculate in how many of the polynomials each monomial is present. That is, we generate P polynomials of the form

$$f = a_0 + a_1x_1 + \ldots + a_{n+1}x_1x_2 + \ldots + a_{2^n-1}x_1x_2 \ldots x_{n-1}x_n \tag{4.5.1}$$

and count the number of occurrences of $a_i = 1,\ 0 \leq i \leq 2^n - 1$.

Let us denote the number of occurrences of coefficient a_i by M_{a_i}. Since each monomial should be included in a function with probability 1/2, *i.e.*, $P(a_i = 1) = 0.5,\ 0 \leq i \leq 2^n - 1$. Therefore, the expected number of occurrences is binomially distributed with expected value $E(M_{a_i}) = P/2$ for each monomial. As previously, we perform a χ^2-goodness of fit test with 2^n degrees of freedom, as described by the following equation (4.5.2).

$$\chi^2 = \sum_{i=0}^{2^n-1} \frac{(M_{a_i} - \frac{P}{2})^2}{\frac{P}{2}}. \tag{4.5.2}$$

If the observed amount is larger than some tabulated limit $\chi^2(1 - \alpha; 2^n)$, for some significance level α, we can distinguish the cipher from a random one. The pseudocode of the Monomial Distribution Test is given in Algorithm 4.5.2.

Algorithm 4.5.2: MONOMIAL DISTRIBUTION TEST(P)

```
for j ← 1 to P
{ for iv ← 1 to 2^n − 1
  { Initialize cipher with IV;
    v[iv] = First keystream bit after initialization;
  Compute ANF of vector v and store result in v;
  for i ← 1 to 2^n − 1
  { if v[i] = 1
    { M_{a_i} + +;
  for d ← 1 to 2^n − 1
  { χ^2 + = ((M_{a_d} − P/2)^2) / (P/2);
if χ^2 > χ^2(1 − α; 2^n)
  then return cipher;
  else return random;
```

This algorithm has a higher computational complexity than the d-Monomial attack, $O(Pn \log n)$, and needs the same amount of memory, $O(n \log n)$. On the other hand, if for a cipher some certain monomials are highly non-randomly distributed, the attack may be successful with less number of IV bits, in other words, smaller n, compared to the d-monomial test. Additionally, although this attack is originally proposed for chosen IV scenario of a fixed unknown key, it is also possible to apply the test for different key values, if the same IV bits are considered.

4.5.3 Maximal Degree Test

A completely different and very simple test is to see if the maximal degree monomial can be produced by the keystream generator. The maximal degree monomial is the product of all IV bits and can hence only occur if all the IV bits have been properly mixed. In hardware oriented stream ciphers the IV loading is usually as simple as possible to save gates, usually the IV bits are loaded into different memory cells. The update function is then performed a number of steps to produce proper diffusion of the bits, intuitively it will take many clockings before all IV bits meet in the same memory cell and even more clocking before they spread to all the memory cells and are mixed nonlinearly. The aim of the maximal degree monomial is to check in a simple way whether the number of initial clockings are sufficient. Since the maximal degree monomial is unlikely to exist if lower degree monomials do not exist, this is our best

candidate to study. Hence, the existence of the maximal degree term in ANFs is a good indication to the satisfaction of diffusion criteria, especially completeness.

According to the Reed-Muller transform the maximal degree monomial can be calculated as the XOR of all entries in the truth table. So the test is similar to the previous tests performed by initializing the cipher with all possible combinations for n IV bits, $z^{iv_0,\ldots,iv_{n-1}} = f(iv_0,\ldots,iv_{n-1})$, all other bits are considered to be constants. The existence of the maximal degree monomial can be checked by XORing the first keystream bit from each initialization this is equivalent to determining a_{2^n-1}.

$$a_{2^n-1} = \bigoplus_{iv_0,\ldots,iv_{n-1}} z^{iv_0,\ldots,iv_{n-1}}.$$

By changing some other IV bits, we receive a new polynomial, perform the same procedure again, and we repeat it to obtain P polynomials. If the maximal degree polynomial never occurs in any of the polynomials or if it occurs in all of the polynomials we successfully distinguish the cipher. Hence we can, with low complexity, and more importantly, almost no memory, check whether the maximal degree monomial exists in the output from the cipher. It is possible, with the same complexity, to consider other weak monomials, the coefficient can be calculated according to the Reed-Muller transform. The complexity of the Maximal Degree Attack is $O(P2^n)$ and it only requires $O(1)$ memory. The description of the test is given in Algorithm 4.5.3.

Algorithm 4.5.3: MAXIMAL DEGREE MONOMIAL TEST(P)

---

for $j \leftarrow 1$ **to** P
  do
  $a_{2^n-1} = 0$;
  for $IV \leftarrow 1$ **to** $2^n - 1$
    do
    Initialize cipher withIV;
    $z =$ first keystream bit after initialization;
    $a_{2^n-1} = a_{2^n-1} \oplus z$;
  if $a_{2^n-1} = 1$
    then $ones++$;
if ones=0 or ones=P
  then *return* cipher;
  else *return* random;

4.5.4 Experimental Results

We applied the proposed tests described above on some of the Phase III eSTREAM candidates to evaluate the efficiency of their initializations. We evaluated their security margin by testing reduced round versions of the ciphers. We also presented some results on the statistical properties of the internal state variables.

The significance level of the hypothesis tests is chosen to be approximately $1-\alpha = 1-2^{-10}$. The tabulated results have a success rate of at least 90%. The required number of IVs, polynomials and the amount of memory needed to attack the ciphers are given in tables. Also, the results for initial state variables are presented with the percentage of weak initial state variables.

Hardware oriented stream ciphers use simple initial key and IV loading compared to software oriented ciphers. Generally, key and IV bits affect one initial state variable. Therefore, they require a large number of clockings to satisfy the diffusion of each input bit on each state bit. We repeated some of our simulations using alternative key and IV loadings in which each IV bit is assigned to more than one internal state bit and then we compared the results to the original settings. In the alternative loadings, the hardware complexity is slightly higher, however on the other hand the cipher has more resistance to chosen IV attacks.

Grain-128

Grain-128 [141] is a hardware oriented stream cipher using a LFSR and a NFSR together with a nonlinear filter function. In the initialization of Grain, a 128 bit key is loaded into the NFSR and a 96 bit IV is loaded into the first 96 positions of the LFSR, the rest of the LFSR is filled with ones. The cipher is then clocked 256 times and for each clock the output bit is fed back into both the LFSR and the NFSR.

In Table 4.4, the results obtained for reduced version of Grain are presented. The highest number of rounds, we succeeded to break is 192 out of the original 256 which corresponds to the 75% of the initialization.

Table 4.4: Number of IV bits needed to attack the first keystream bit of Grain-128 for different number of rounds in the initialization (out of 256 rounds).

Rounds	d-Monomial test			Monomial distr. test			Max. degree monomial		
	P	IVs	Memory	P	IVs	Memory	P	IVs	Memory
160	1	14	2^{14}	2^6	7	2^7	2^5	11	1
192	1	25	2^{25}	2^6	22	2^{22}	2^5	22	1

In Table 4.5, the results of the experiments for initial state variables are presented. The number of weak initial state variables are three times better in the maximum degree test compared to the d-monomial test. The statistical deviations in state bits remain even after full initialization. These weak state bits are located in the left most positions of the feedback shift registers. To remove the statistical deviations in state variables, at least 320 initial clockings are needed.

It is possible that if we use larger number of IV bits, the weaknesses in state variables may also be observed from the keystream bits. In Appendix E, we applied a linear regression model and predicted that using around 50 IV bits, it is possible to attack Grain with d-monomial tests.

Table 4.5: Number of IV bits needed to attack the initial state variables Grain-128 for different number of rounds in the initialization (out of 256 rounds).

Rounds	d-Monomial test				Monomial distr. test				Max. degree monomial			
	P	IVs	Memory	Fraction	P	IVs	Memory	Fraction	P	IVs	Memory	Fraction
256	1	14	2^{14}	33/256	2^6	8	2^8	20/256	2^5	14	1	108/256
256	1	16	2^{16}	40/256	2^6	10	2^{10}	35/256	2^5	16	1	120/256
256	1	20	2^{20}	56/256	2^6	15	2^{15}	44/256	2^5	20	1	138/256
288	1	20	2^{20}	0/256	2^6	20	2^{20}	0/256	2^5	20	1	73/256

Alternative Key and IV Loading for Grain-128

Here, we propose an alternative key and IV loading in which only the loading of the first 96 bits of the NFSR is different from the original. Instead of directly assigning the key, we assign the modulo 2 summation of IV and the first 96 bits of the key. The proposed loading is very similar to the original and the increase in number of gates required is approximately 10-15%. In an environment where many resynchronizations are expected, one can reduce the number of initial clockings by using some more gates in the hardware implementation. In the new loading, each IV bit affects two internal state variables. We repeated our experiments using the new loading and the results are given in Table 4.6 and Table 4.7. Using alternative loading, Grain shows more resistance to the presented attacks, but still the statistical deviations in the state bits remain after full initialization. However, the number of weak state bits decreases to 47 from 73.

Table 4.6: Number of IV bits needed to attack the first keystream bit of Grain-128 with alternative key and IV loading for different number of rounds in the initialization (out of 256 rounds).

Rounds	d-Monomial test			Monomial distr. test			Max. degree monomial		
	P	IVs	Memory	P	IVs	Memory	P	IVs	Memory
160	1	19	2^{19}	2^6	20	2^{20}	2^5	21	1

Table 4.7: Number of IV bits needed to attack the initial state variables of Grain-128 with alternative key and IV loading for different number of rounds in the initialization (out of 256 rounds).

Rounds	d-Monomial test				Monomial distr. test				Max. degree monomial			
	P	IVs	Memory	Fraction	P	IVs	Memory	Fraction	P	IVs	Memory	Fraction
256	1	14	2^{14}	1/256	2^6	8	2^8	4/256	2^5	14	1	100/256
256	1	16	2^{16}	5/256	2^6	10	2^{10}	10/256	2^5	20	1	108/256
288	1	20	2^{20}	0/256	2^6	20	2^{20}	0/256	2^5	20	1	47/256

Trivium

Trivium [32] is another hardware oriented stream cipher based on NFSRs (See Section 4.4.1 for details of the algorithm.). The results for Trivium are given in Table 4.8 and Table 4.9. The distinguishers are successful to attack 704 rounds of Trivium. The percentage of weak initial state variables for Trivium are approximately same using d-monomial and maximal degree tests.

Table 4.8: Number of IV bits needed to attack the first keystream bit of Trivium for different number of rounds in the initialization (out of 1152 rounds).

Rounds	d-Monomial test			Monomial distr. test			Max. degree monomial		
	P	IVs	Memory	P	IVs	Memory	P	IVs	Memory
608	1	12	2^{12}	2^5	9	2^9	2^5	9	1
640	1	15	2^{15}	2^6	13	2^{13}	2^5	13	1
672	1	20	2^{20}	2^8	18	2^{17}	2^5	18	1
704	1	27	2^{27}	2^6	23	2^{23}	2^5	24	1

Alternative Key and IV Loading for Trivium

In the original key and IV loading, 128 bits of the initial state are assigned to constants and the key and IV bits affect only one state bit. Here, we propose an alternative initial key and IV loading in which the first register is filled with the modulo 2 summation of key and IV, the second register is filled with IV and the last register is

Table 4.9: Number of IV bits needed to attack the initial state variables of Trivium for different number of rounds in the initialization (out of 1152 rounds).

Rounds	d-Monomial test				Monomial distr. test				Max. degree monomial			
	P	IVs	Memory	Fraction	P	IVs	Memory	Fraction	P	IVs	Memory	Fraction
608	1	12	2^{12}	144/288	2^5	12	2^{12}	105/288	2^5	12	1	169/288
640	1	12	2^{12}	57/288	2^5	12	2^{12}	29/288	2^5	12	1	86/288
672	1	15	2^{15}	87/288	2^5	15	2^{15}	0/288	2^5	15	1	108/288
704	1	20	2^{20}	74/288	2^5	20	2^{20}	12/288	2^5	20	1	76/288

filled with the complement of key plus IV. In this setting, each IV bit affects 3 internal state bits, therefore the diffusion of IV bits to the state bits is satisfied in less number of clockings. We repeated the tests using the alternative loading and obtained the results given in Table 4.10 and Table 4.11. In the alternative loading, the required number of IV bits and memory needed to attack Trivium are approximately 50 percent more compared to the original loading.

Table 4.10: Number of IV bits needed to attack the first keystream bit of Trivium with alternative key and IV loading for different number of rounds in the initialization (out of 1152 rounds).

Rounds	d-Monomial test			Monomial distr. test			Max. degree monomial		
	P	IVs	Memory	P	IVs	Memory	P	IVs	Memory
608	1	18	2^{18}	2^5	22	2^{22}	2^5	17	1
640	1	23	2^{23}	–	–	–	2^5	21	1

Table 4.11: Number of IV bits needed to attack the initial state variables of Trivium with alternative key and IV loading for different number of rounds in the initialization (out of 1152 rounds).

Rounds	d-Monomial test				Monomial distr. test				Max. degree monomial			
	P	IVs	Memory	Fraction	P	IVs	Memory	Fraction	P	IVs	Memory	Fraction
608	1	12	2^{12}	4/288	2^5	12	2^{12}	2/288	2^5	12	1	21/288
640	1	18	2^{18}	17/288	2^5	18	2^{18}	19/288	2^5	18	1	24/288
672	1	20	2^{20}	0/288	2^5	20	2^{20}	0/288	2^5	20	1	0/288

Decim

Decim-v2 [65] is also a hardware oriented stream cipher based on a nonlinearly filtered LFSR and the irregularly decimation mechanism, ABSG. The internal state

size of Decim-v2 is 192 bit and it is loaded with 80 bit Key and 64 bit IV. The first 80 bits of the LFSR are filled with the key, the bits between 81 and 160 are filled with linear functions of key and IV and the last 32 bits are filled with a linear function of IV bits.

The results we obtained for Decim-v2 are given in Table 4.12 and Table 4.13. The security margin for Decim against chosen IV attacks is very large, the cipher can only be broken when not more than about 3% of the initialization is used. This is mainly because of the initial loading of key and IV in which each IV bits affect 3 state variables and the high number of quadratic terms in the filter function. The weakness in initial state variables can be observed for higher number of clockings. The number of weak initial state variables are approximately same for all attacks.

Table 4.12: Number of IV bits needed to attack the first keystream bit of Decim-v2 for different number of rounds in the initialization (out of 768 rounds).

Rounds	*d*-Monomial test			Monomial distr. test			Max. degree monomial		
	P	IVs	Memory	P	IVs	Memory	P	IVs	Memory
20	1	16	2^{16}	2^5	13	2^{13}	2^5	19	1

Table 4.13: Number of IV bits needed to attack the initial state variables of Decim-v2 for different number of rounds in the initialization (out of 768 rounds).

Rounds	*d*-Monomial test				Monomial distr. test				Max. degree monomial			
	P	IVs	Memory	Fraction	P	IVs	Memory	Fraction	P	IVs	Memory	Fraction
160	1	12	2^{12}	47/192	2^5	17	2^{17}	47/192	2^5	12	1	44/192
192	1	20	2^{20}	18/192	2^5	20	2^{20}	13/192	2^5	20	1	17/192

4.5.5 Discussion

In this part of the study, we generalize the idea of *d*-monomial attacks and propose a framework for chosen IV statistical analysis. The proposed framework can be used as an instrument for designing good initialization procedures. It can be used to verify the effectiveness of the initialization, but also to help designing a well-balanced initialization, *e.g.*, prevent an unnecessary large number of initial clockings or even reduce the number of gates used in an hardware implementation by being able to use a simpler loading procedure.

Also, we propose a few new statistical attacks, apply them on some existing stream cipher proposals, and give some conclusions regarding the strength of their IV initialization. In particular, we experimentally detected statistical weaknesses in the keystream

of Trivium using an initialization reduced to 704 rounds as well as in some state bits of Grain-128 with full IV initialization.

For ciphers Grain and Trivium, we also propose alternative initialization schemes with slightly higher hardware complexity. In the proposed loadings, each IV and key bit affects more than one state bit and the resistance of the ciphers to the proposed attacks increases about 50%. Decim seems to have a high security margin and it is an interesting question whether a simpler loading procedure could be used in Decim which could mean a smaller footprint in hardware, fewer intial clockings could also be used for a faster intialization procedure.

4.5.6 Improvement of Fischer *et al.*[1]

Fischer *et al.* [1] proposed an attack using the concept of probabilistic neutral key bits that is, the key bits having no influence on the value of coefficient with some high probability. Using this approach, it is possible to improve our distinguishers to recover a part of the secret key. In [1], this method is applied to reduced round Grain and Trivium. In this proposed approach, weak IV bits are selected randomly, how to select weak IV bits intelligently is an open question.

4.6 Linear Independence

The focus of this test is to analyze the effect of linear relations of IVs on keystream bits. For a secure cipher, the linear relation of IVs should completely be destroyed by the encryption function.

4.6.1 Preliminaries

Here, we first present some necessary definitions.

Definition 4.6.1. Let $\alpha_1, \alpha_2, \ldots, \alpha_k$ be distinct elements of the vector space V. If the vector equation

$$c_1\alpha_1 + c_2\alpha_2 + \ldots + c_k\alpha_k = 0, \tag{4.6.1}$$

has only the trivial solution $c_1 = c_2 = \ldots = c_k = 0$, then α_i's are said to be *linearly independent*. If there exist scalars c_i not all zero, the vectors are *linearly dependent*.

Definition 4.6.2. The vector γ is said to be *linear combination* of $\alpha_1, \alpha_2, \ldots, \alpha_k$ if

there exists scalars $c_1, c_2, \ldots, c_k$ such that

$$\gamma = c_1\alpha_1 + c_2\alpha_2 + \ldots + c_k\alpha_k. \tag{4.6.2}$$

The collection of all linear combinations of the vectors $\alpha_1, \alpha_2, \ldots, \alpha_k$ is called the *linear span* of these vectors.

Rank of a matrix is defined to be the number of independent rows (or columns) and it is calculated using the Gaussian elimination. For a $M \times M$ square matrix, the probability distribution of the rank value is given as

$$Pr(Rank = r) = 2^{r(2M-r)-M^2} \prod_{i=0}^{r-1} \frac{(1-2^{i-M})^2}{1-2^{i-r}}, \tag{4.6.3}$$

where $r = 0, 1, 2, \ldots, M$ [142].

4.6.2 Linear Span Test

Well known *Binary Matrix Rank Test* [103] evaluates the randomness properties of a given sequence based on the ranks of binary non-overlapping matrices generated from the sequence. It can directly be applied to evaluate stream ciphers given a keystream (of length $> 40,000$ bits) generated by the cipher.

Here, we present a new application of the binary matrix rank test to stream ciphers. Instead of generating a long keystream, we use a chosen IV approach and input linearly dependent IVs and observe the linear dependence of keystream portions.

In our approach, we consider all linear combinations of m independent IVs $\{$ IV_1, $IV_2, \ldots,$ $IV_m\}$ that are generated randomly. Then, we initialize the cipher using (2^m) IVs and for each IV we produce 2^m bit keystream and generate a matrix of size $(2^m) \times (2^m)$. We repeat this process N times and obtain N rank values, then evaluate the distribution of ranks as given in the Matrix Rank Test. The pseudocode of the Linear Span Test is given in Algorithm 4.6.1. The total number of resynchronization required during the test is $(2^m)N$.

The rank of a matrix is expected to be high for random matrices. The probability of full rank is approximately 0.2888 and the rank distribution takes its maximum value 0.5776 for one less of full rank. Obtaining low rank matrices is an indication that the encryption function is not able to destroy linear relations of the input IVs.

Algorithm 4.6.1: LINEAR SPAN TEST(N, m)

for $i \leftarrow 1$ **to** N

$$\begin{cases} \text{Randomly select } m \text{ independent vectors } \alpha_1, \ldots, \alpha_m; \\ \text{Generate } 2^m \text{ linear combinations of } \alpha_i\text{'s and produce} IV_1, \ldots, IV_{2^m}; \\ \textbf{for } j \leftarrow 1 \textbf{ to } (2^m) \\ \quad \begin{cases} \text{Initialize the cipher with } IV_j; \\ Z_j = \text{First } 2^m \text{ bits of keystream;} \end{cases} \\ M = \begin{bmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_{2^m} \end{bmatrix} \\ R_i = \text{Rank of } M; \end{cases}$$

Evaluate $R_1, \ldots, R_N$ using Matrix Rank Test;

return $(p - value)$

4.6.3 Experimental Results

For our experiments, we fixed the input m as 5 and choose $N = 100$. So, the size of the matrices are fixed to 32×32 as given in the NIST test suite. We applied the tests to the Phase III Candidates of eSTREAM project and obtained the results presented in Table 4.14. As seen from the table, none of the results significantly deviates from 0.5, therefore all Phase III candidates are strong against linear span test.

4.7 Completeness Property

Completeness is a very important security measure and a Boolean function is said to be *complete* if its output depends on all input bits. In this part of the study, we propose a diffusion test that measures the effect of each key and IV bit on keystream.

4.7.1 Diffusion Test

In the *Diffusion Test*, firstly, a random vector $(u_1, \ldots, u_k, u_{k+1}, \ldots, u_{k+v})$ is chosen, where the first k bits represent the key, and the remaining v bits represent the IV. Using this key and IV, a keystream of length L is generated. Then, $k + v$ new vectors are generated by the operation $(u_1, \ldots, u_{k+v}) \oplus e_i$, where e_i is the vector having 1 in

Table 4.14: The average of 100 p-values of Linear Span Test for Phase III eSTREAM Candidates

Cipher	Average p-value
Crypt-MT	0.517496
Decim	0.525619
Dragon	0.487644
Edon80	0.481070
FFCSR	0.507010
Grain	0.441762
HC	0.559911
Lex	0.491445
Mickey	0.484703
NLSv2	0.485840
Pomaranch	0.516326
Rabbit	0.543500
Salsa20	0.535868
Sosemanuk	0.482181
Trivium	0.496277

the entry i and zero elsewhere. For each vector, keystream of length L is generated. Then, these keystreams are XORed with the original keystream. Using these vectors, a matrix of size $(k+v) \times L$ is obtained. This procedure is repeated N times and obtained matrices are added. Matrix entries are evaluated using χ^2 goodness of fit test using the probabilities given in Table 4.15. The pseudocode of the test is given in Algorithm 4.7.1.

The entries of the matrix are expected to follow a normal distribution with mean $N/2$ and variance $N/4$, when N is large. For this test to be applicable, the value of N should be greater than 100. Entries with high/low value indicate poor diffusion properties of corresponding cells. The χ^2 goodness of fit test is applied to the entries of the matrix to evaluate diffusion property. If the cipher fails this test, initialization phase of the algorithm should be revised.

Table 4.15: Interval and probability values of Diffusion Test using 1024 key and IV pairs.

Category Limits	*Probability*
0 - 498	0.199405
499 - 507	0.189855
508 - 516	0.221481
517 - 525	0.189855
526 - 1024	0.199405

Algorithm 4.7.1: DIFFUSION(N)

Let M be a zero matrix of size $(k+v) \times l$;
for $i \leftarrow 1$ **to** N
{
Randomly choose K_i and V_i;
$M_i = S(K_i, V_i, l)$ and let $M_i = (M_{i_1}, \ldots, M_{i_l})$;
for $j \leftarrow 1$ **to** k
{
$K_j = K_i \oplus e_j$
$P_j = S(k'_j, Vj, l)$;
}
for $j \leftarrow k$ **to** $k+v$
{
$V_j = V_i \oplus e_j$
$P_j = S(k'_i, V'_j, l)$;
}
$M^* = \begin{bmatrix} P_1 \\ \vdots \\ P_{k+v} \end{bmatrix}$
$M = M + M^*$;
}
Evaluate entries of M using χ^2 test;
return $(p - value)$

4.7.2 Experimental Results

For the *diffusion test*, a matrix of size $(k+v) \times 256$ is generated using 2^{10} random key and IV pairs. Table 4.16 presents the average of 100 p-values of Phase I candidates of eSTREAM from the diffusion test. F-FCSR-8 fails this test mainly due to the lack of diffusion property of IV bits between 66 and 101. Also, the cipher Frogbit and Mag do not satisfy the necessary diffusion property. For Zk-Crypt, the 29^{th} and 30^{th} bits of IV and key do not satisfy the desired diffusion. These p-values are expected to distributed normally with mean 0.5 and standard deviation 0.0289. Therefore, it is unlikely to obtain an average greater than 0.6 and less than 0.4. Therefore, diffusion property of the ciphers ABC, CryptMT, Decim, Dicing, Mir-1, Sfinks and TSC-3 are suspicious.

4.8 Summary

To summarize the chapter, analysis of stream ciphers based on cryptographic properties of Boolean functions are studied. We obtained a linear approximation for F_1 with bias 2^{-31} valid for a subset of keys and IVs for Trivium with 288 initial clocking.

Using this approximation, it is possible to attack the cipher using 2^{62} resynchronizations, given that one of the weak keys are used. Using the d-monomial approach, we experimentally detected statistical weaknesses in the keystream of Trivium using an initialization reduced to 704 rounds as well as in some state bits of Grain-128 with full IV initialization. Also, we experimentally show that F-FCSR-8, Frogbit, Mag and Zk-Crypt do not satisfy the required diffusion of input bits.

Table 4.16: The average of 100 p-values of Diffusion Test for Phase I eSTREAM Candidates

Cipher	*Key Size*	*IV Size*	*Diffusion*
ABC v.2	128	128	0.356969
Achterbahn	80	64	0.462577
CryptMT	128	128	0.340794
Decim	80	64	0.379434
Dicing	128	64	0.279894
Dragon	128	128	0.471513
Edon80	80	64	0.535973
F-FCSR-8	128	128	0.000000
Frogbit	128	128	0.000000
Fubuki	128	128	0.557280
Grain	80	64	0.478812
HC-256	128	64	0.574439
Hermes8	128	128	0.535768
LEX	128	128	0.460434
Mag	128	32	0.000000
Mickey	80	64	0.528977
Mickey-128	128	128	0.517717
Mir-1	128	64	0.658961
NLS	128	128	0.511876
Phelix	128	128	0.469131
Polar Bear	128	128	0.499867
Pomaranch	128	64	0.569247
Py	128	64	0.489706
Rabbit	128	64	0.537224
Salsa20	128	64	0.480442
SFINKS	80	80	0.676244
Sosemanuk	128	64	0.447910
Trivium	80	64	0.533170
TSC-3	128	64	0.637575
Vest-4	128	64	0.427980
WG	128	128	0.527545
Yamb	128	64	0.407011
Zk-Crypt	128	128	0.000000

Chapter 5

Tests Based on Random Mappings

Random mappings are functions from a finite set of n elements onto itself. They are widely used in many combinatorial problems such as (i) proportion of empty urns after throwing n balls into n urns, (ii) probability that two persons have the same birthday among a group of n people, (iii) the required number of random selections among n coupons to obtain a full collection, *etc.* Using a subset of key and IV bits as input and a subset of keystream bits as output, it is possible to form random mappings by using stream ciphers.

The problem of inverting (finding the pre-image of a given point) random mappings has a great practical importance. For mappings generated by stream ciphers, this problem corresponds to obtaining secret state bits using the output keystream sequence, *i.e.*, breaking the cipher. However, there is no known efficient algorithm to find the pre-image of a given point for a random function. Trivially, using exhaustive search, it is possible to check all possible inputs until desired output is obtained, leading excessive time requirement. Alternatively, a lookup table that contains the pre-images of all points can be constructed, resulting in large memory requirement. To balance between solution time and required memory, TMTO attacks are proposed as generic methods to invert random mappings [143].

It is possible to apply TMTO attacks to stream ciphers by various approaches [144, 145, 146, 147]. The success probabilities of the attacks are calculated under the assumption that the cipher behaves like a random mapping or a random permutation depending on the setting. To avoid the attack, some conditions for stream ciphers are given as (i) IV size should be at least equal to key size, (ii) state size should be at least the size of key plus IV and (iii) key size should be at least 80 bits [147].

In this chapter, we aim to analyze the security of stream ciphers based on some properties of random mappings. By focusing on different TMTO attacks, we try to find suitable test statistics. First, we consider the coverage of mappings generated using a subset of IV bits, then we analyze the index of the first repetition when the random mapping is iteratively applied. Finally, we consider the distinguished point method against stream ciphers and analyze the coverage properties of random mappings that are followed by a special keystream pattern. Using these test statistics, we propose three new distinguishers namely (i) coverage test, (ii) ρ-test and (iii) distinguished point (DP) coverage test and apply these tests to all Phase III Candidates of eSTREAM project.

The organization of the chapter is as follows. In the following section, some preliminary information about random mappings is presented and then background knowledge on TMTO attacks focusing on stream ciphers are presented. In Section 5.3, the new distinguishers are presented and in Section 5.4, experimental results on eSTREAM candidates are given. Finally, we give a summary of the chapter.

5.1 Preliminaries

Let $f(x)$ be a random mapping $X \rightarrow X$ where X is a finite set of n elements. Random mappings independently assign one of the image points $y \in X$ for each input $x \in X$. The sample space consists of n^n random mappings. For each mapping f, we can associate the random variable $C = |\text{image of } f|$, that is the coverage of f.

Proposition 5.1.1. *Let us have n independent and identically distributed random variables $X_1, X_2, \ldots X_n$, each uniformly selected from the set $\{1, 2, \ldots, n\}$. Let A_k be the number of selections that contain k different elements. The recursive formulation of A_k is given as*

$$A_k = \binom{n}{k} \left[k^n - \sum_{i=1}^{k-1} \binom{k}{i} \frac{A_{k-i}}{\binom{n}{k-i}} \right] \tag{5.1.1}$$

where $A_1 = \binom{n}{1}$.

Example 5.1.2. *Let $n = 3$. The total number of selections is $3^3 = 27$. The selections with $k = 1$ distinct elements are $\{111, 222, 333\}$, with $k = 2$ distinct elements are* {112, 121, 211, 113, 131, 311, 221, 212, 122, 223, 232, 322, 331, 313, 133, 332, 323, 233} *and with $k = 3$ distinct elements are $\{123, 132, 213, 231, 312, 321\}$. Then, $A_1 = 3$, $A_2 = 18$, $A_3 = 6$, with a total of 27.*

Following the above proposition, the probability distribution of C is obtained as

$$Pr(C = k) = \frac{A_k}{n^n} \tag{5.1.2}$$

for $k = 1, \ldots, n$.

The expected value of C is $n - n(1 - \frac{1}{n})^n$ which is approximately $n(1 - e^{-1})$ and for large n, hence $C \approx 0.632^n$.

Iteratively application of f to $x_0 \in X$ yields the sequence

$$\{x_0, x_1 = f(x_0), x_2 = f(x_1), \ldots, x_n = f(x_{n-1})\}. \tag{5.1.3}$$

In Figure 5.1, the typical behavior of an iteration operation is given. Since the set X is finite, after some iterations, we will encounter a point that has occurred before. Starting with a point x_0, let x_m be the point that the iteration enters a loop to form a cycle. The path between x_0 and x_m is called the *tail length*. The sum of the tail length and *cycle length* is defined as the *ρ-length*.

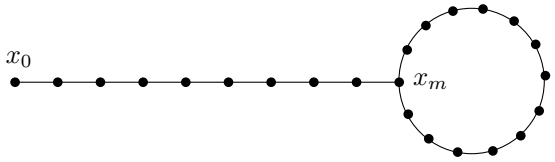


Figure 5.1: Graphical representation of an iteration

Proposition 5.1.3. *The probability distribution of the ρ-length for a random mapping of n elements is given as*

$$Pr(\rho - length = k) = (\frac{k-1}{n}) \prod_{i=1}^{k-2} (\frac{n-i}{n}), \ for\ k \geq 2. \tag{5.1.4}$$

5.2 Time Memory Tradeoff Attacks

TMTO attacks aim to speed up the exhaustive key search at the expense of memory usage and the success rate depends on the time and memory allocated for cryptanalysis. The attacks consist of *offline* (or pre-computation) and *online* phases. In the offline phase, a look up table based on the cipher is constructed. In the online phase, a given target is searched using the previously constructed table. In the attack,

- N is the size of the search space,
- P is the complexity reserved for pre-calculations,
- M is the amount of memory available,
- T is the online time complexity and

- D is the amount of data available.

Complexity of the TMTO attack is usually assumed to be the maximum of T and M and the attack is considered to be successful if the complexity is less than N. Generally, the pre-computation complexity P is not included to the attack complexity.

In 1980, Hellman presented a probabilistic tradeoff attack that recovers the secret key in $N^{2/3}$ operations using $N^{2/3}$ words of memory after N operations of precomputation, for an arbitrary block cipher with N keys [143]. The success rate of the attack depends on the assigned memory and time. Different from exhaustive key search that uses an arbitrary known plaintext and ciphertext pair, tradeoff attacks against block ciphers require a chosen plaintext block. Given a plaintext and ciphertext pair (P_0, C_0), the aim is to find the secret key k that satisfies $C_0 = E_k(P_0)$.

In the offline phase, chains of length t are generated using the function f which is the composition of the *reduction function* R and the encryption function E. Reduction functions are necessary for cases where the input and output size of f are different. Successive keys are generated using the following equation

$$\begin{aligned} x_{j\ i+1} &= f(x_{j\ i}), \\ &= R(E_{x_{j\ i}}(P_0)),\ 0 \leq i \leq t. \end{aligned}$$

where $x_{j\ 0}$ $(1 \leq j \leq m)$ is selected randomly. Starting from m random points x_{j0}, the target P_0 is encrypted using the successive keys and start points (SPs) and end points (EPs) of m chains are stored in a table. The construction of the table is summarized in Figure 5.2.

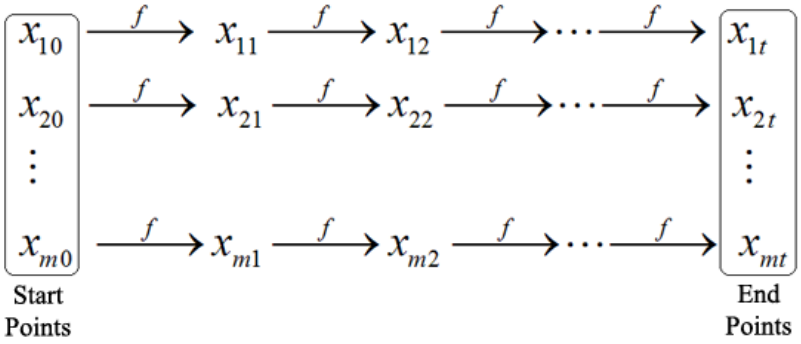


Figure 5.2: Construction of the lookup table in the offline phase

In the online phase, the aim is to find the key that generates (P_0, C_0) pair, assuming it is one of the keys used in the offline phase. Since only SPs and EPs of the table are stored, a similar chain for C_0 is generated and after each encryption, the obtained

value is compared to the EPs of the table. If a match is obtained, then the whole chain corresponding to the EP is regenerated and the key just before reduction of C_0 is obtained as the secret key. Therefore, the success probability of the attack is closely related to the percentage of the keys that are covered in the offline phase. It should be noted that sometimes the desired key is a part of a chain that is merged with another chain of the table which causes a false alarm.

The efficiency of TMTO attacks can be improved using different approaches. Using distinguished points approach, instead of generating fixed length chains, the chains are terminated whenever a distinguished k_i (e.g. having last m bits as zero) is obtained. This improvement reduces the memory access in the online phase, since the value k_i is compared to the end points only if it is a distinguished point. In [148], the idea of distinguished points is used to attack 40-bit DES and the key is recovered in 10 seconds with a success rate of 72%. Another improvement is to use Rainbow tables [149] where different reduction functions are utilized. In this approach, instead of having t different tables, only one table of size $mt \times t$ is generated.

TMTO Attacks Against Stream Ciphers:

TMTO attacks against stream ciphers are firstly proposed by Babbage [144] and Golić [145] independently. In the pre-computation phase, the function $f : \mathbb{F}_2^{logN} \rightarrow \mathbb{F}_2^{logN}$ that inputs the $logN$ bit state and outputs $logN$ keystream bits is considered and the following table is generated;

$$\begin{aligned} &(S_1, f(S_1)) \\ &(S_2, f(S_2)) \\ &\vdots \\ &(S_M, f(S_M)) \end{aligned}$$

using M randomly selected initial states S_i. Then, the table is sorted with complexity $O(MlogM)$ based on the output keystream bits. Given the output keystream of length $D+logN-1$, D overlapping $logN$ bit subsequences are generated and compared to the table generated in the offline phase. It should also be noted that different from block ciphers, the TMTO attacks against stream ciphers do not require chosen plaintext.

In [146], Biryukov and Shamir combined the tradeoff attacks presented by Hellman and Babbage-Golić. Using the same f function, chains of length t are generated by assigning the output keystream as the new internal state as presented in Figure 5.2. Instead of using output size of n bits (internal state size), for stream ciphers using k

bit key and v bit IVs the output size is taken as $k + v$ bits, this function is also close to a permutation. Using this approach, it is possible to recover the key with time and memory complexity $T = M = 2^{\frac{1}{2}(k+v)}$ if $D = 2^{\frac{1}{4}(k+v)}$ frames are available [147]. For IVs shorter than key ($v < k$), the stream cipher is vulnerable to the TMTO attack. Therefore, to avoid the attack, IV size should be at least equal to the key size and IVs should not be used in a predictable way and the state size should at least be the size of key plus IV.

According to [146], it is also possible to use distinguished points idea against stream ciphers. Similar to Babbage-Golić scheme, the states and the corresponding keystream with distinguished property are stored. This is especially important in online phase where the different keystream portions are compared to the table. Since there is no need to check keystream portions that do not satisfy the distinguishing property, this reduces the complexity of checking memory significantly.

In [147], the following remarks are given; (i) putting restriction on the number of resynchronizations using a fixed key does not increase the security of the cipher against TMTO, (ii) the complexity of the initialization phase has no effect on the efficiency of the attack, and (iii) the attack may be applied to any keystream positions as long as they are known.

In [150], a tradeoff attack against LILI-128 is proposed using 2^{64} bits of keystream, a lookup table of 2^{45} 89-bit words and 2^{48} operations. In [145], a TMTO attack against A5 is presented with $T.M \geq 2^{63.32}$ and precomputation time about $T/102$. An improvement of this attack, presented by Biryukov *et al.* [151], requires about 2 minutes of GSM conversation and finds the key in a few seconds with 2^{42} preprocessing and memory complexity of four 73 GB hard-drives.

5.3 Three New Distinguishers

In the following part of the chapter, we present three new distinguishers against synchronous stream ciphers, but they can also be used to test the security of other cryptosystems such as block ciphers and hash functions. For a synchronous stream cipher with k bit key $K = (k_1, k_2, \ldots, k_k)$ and v bit $IV = (iv_1, iv_2, \ldots, iv_v)$, let $Z = z_0, z_1, \ldots$ denote the keystream sequence. In this part, we consider the chosen IV approach where the attacker has access to a number of different keystream sequences generated using different (possibly chosen) IV values and same random secret key.

$$
\begin{aligned}
IV^{(1)} &= (\overbrace{***\ldots*}^{v-l \text{ bits}} | \overbrace{000\ldots0}^{l \text{ bits}}) \rightarrow Z^{(1)} = (z_1^{(1)}, z_2^{(1)}, \ldots, z_l^{(1)}) \\
IV^{(2)} &= (***\ldots* | 000\ldots1) \rightarrow Z^{(2)} = (z_1^{(2)}, z_2^{(2)}, \ldots, z_l^{(2)}) \\
&\vdots \\
IV^{(2^l)} &= (***\ldots* | 111\ldots1) \rightarrow Z^{(2^l)} = (z_1^{(2^l)}, z_2^{(2^l)}, \ldots, z_l^{(2^l)})
\end{aligned}
$$

Figure 5.3: The table generated in the coverage test

5.3.1 Coverage Test

The coverage test is a new probabilistic distinguisher against stream ciphers where a table similar to the approach of Babbage-Golić is used. In the original approach, output keystreams of length n (state size) are generated. For our statistical test, it is impractical to use keystream bits of size as large as n, so we focus on a subset of IV bits (l out of v) and generate l bit keystreams.

First, we select l random (active) positions from IV and fix the rest (inactive) bits to a random value. Then, we synchronize the cipher for all possible 2^l IVs and generate l bit keystream $(z_1^{(i)}, z_2^{(i)}, \ldots, z_l^{(i)})$ for each IV as given in Figure 5.3. Then, calculate the number of distinct $Z^{(i)}$'s and denote it as C_1 which is expected to be around 0.63×2^l. We repeat the experiment for a number of times with different assignments of the inactive IV bits and obtain a coverage variable for each trial, then evaluate the randomness of the cipher based on the distribution of C_i's. The pseudocode of the Coverage test is given in Algorithm 5.3.1.

Using the recursive formula (5.1.2), the probability distributions of C_i for 12 and 14 bits are calculated and categorized into 5 groups with approximately equal probability. The limit of the groups and corresponding probabilities are given in Table 5.1.

If the coverage test returns low p-value (< 0.01), it means that the coverage of the corresponding mapping is statistically different than the expected values. Obtaining a low coverage value means that the first keystream bits that are generated using different IVs are similar, it is obviously a threat for frequently resynchronized ciphers. This is also an indication of low diffusion properties. Obtaining a high coverage value means that the mapping is close to a permutation. This may be interpreted as follows; whenever a subpart of the secret bits is recovered and the rest of the bits form a permutation, to identify unknown state bits, the required number of keystream bits is equal to the number of unknown state bits. For mappings close to permutation, cipher is more vulnerable to TMTO attacks.

$$Z^{(0)} \xrightarrow{IV^{(0)}=(***|Z^{(0)})} Z^{(1)} \xrightarrow{IV^{(1)}=(***|Z^{(1)})} \cdots \xrightarrow{IV^{(R)}=(***|Z^{(R)})} Z^{(R)}$$

Figure 5.4: The rows generated in the ρ-test

Algorithm 5.3.1: COVERAGE TEST(R, l)

Randomly select l positions $p_1, p_2, \ldots, p_l$ from v bits of IV;
for $i \leftarrow 0$ **to** R
{
  Randomly select $IV = (iv_1, iv_2, \ldots, iv_v)$;
  for $j \leftarrow 0$ **to** $2^l - 1$
  {
    $J = (j_1, j_2, \ldots, j_l)$ binary representation of j;
    $(iv_{p_1}, iv_{p_2}, \ldots, iv_{p_l}) = J$;
    $Z^{(j)}$ = First l keystream bits using K and IV;
  }
  $Coverage_i$ = Number of distinct $Z^{(1)}, \ldots, Z^{(2^j)}$;
}
Evaluate $(Coverage_1, \ldots, Coverage_R)$ using χ^2 test;
return $(p - value)$

Table 5.1: Interval and probability values of Coverage test using 12 and 14 IV bits

12 IV Bits		*14 IV Bits*	
Category Limits	*Probability*	*Category Limits*	*Probability*
0-2572	0.199139	0-10323	0.201591
2573-2584	0.204674	10324-10346	0.195966
2585-2594	0.197856	10347-10367	0.207519
2595-2606	0.203225	10367-10390	0.195253
2607-4096	0.195106	10392-16384	0.199671

5.3.2 ρ-Test

ρ-test is another probabilistic distinguisher against stream ciphers where the encryption function is iteratively applied and a sequence of l bits keystreams $Z^{(1)}, Z^{(2)}, \ldots$ are generated until one of the entries is repeated (See Figure 5.4). The index of the last entry, ρ-length, is used to evaluate the randomness of the cipher.

First, we select l random positions from IV and fix the rest (inactive) bits to a random value, then initialize the cipher with this IV and secret key K and generate l bit keystream, $Z^{(1)}$. Then, $Z^{(1)}$ is assigned to the variable part of IV and iteratively l bit keystreams are generated until one of the $Z^{(i)}$ is repeated and the index of the last entry is stored as R_1. This is repeated for different assignments of inactive IV bits,

then R_i values are compared to their theoretical distribution using χ^2 goodness of fit tests. The pseudocode of ρ test is given in Algorithm 5.3.2.

Using the recursive formula (5.1.4), the probability distribution of R_i for 15 and 20 bits is calculated and categorized into 5 groups with approximately equal probability. The limit of the groups and corresponding probabilities are given in Table 5.2.

Algorithm 5.3.2: ρ-TEST(R, l)

Randomly select $K = (k_1, k_2, \ldots, k_k)$;
Randomly select l positions $p_1, p_2, \ldots, p_l$ from v bits of IV;
for $i \leftarrow 0$ **to** R
do
{
Randomly select $IV = (iv_1, iv_2, \ldots, iv_v)$;
repeat
$Z =$ First l keystream bits using K and IV;
$(iv_{p_1}, iv_{p_2}, \ldots, iv_{p_l}) = (z_1, z_2, \ldots, z_l)$;
$X_i = Z$;
$Index_i++$;
until *a* X_i *value is repeated*
}
Evaluate $(Index_1, \ldots, Index_R)$ using χ^2 test;
return $(p - value)$

Table 5.2: Interval and probability values of ρ-test using 15 and 20 IV bits

15 IV Bits		*20 IV Bits*	
Category Limits	*Probability*	*Category Limits*	*Probability*
2- 122	0.201906	2-685	0.200258
123 - 184	0.200448	686-1036	0.200124
185 - 246	0.199904	1037-1386	0.199400
247 - 325	0.198270	1387-1838	0.200518
326 - 32768	0.199472	1839 -1048576	0.199700

Rows of Hellman tables are generated by applying encryption and reduction functions iteratively. This test generates rows similar to Hellman's table and calculates their ρ-length. Obtaining a low p-value from the ρ test means that length of iterations is statistically longer or shorter than the expected values. Having short cycles results low coverage, which motives us to use smaller number of iterations t.

5.3.3 DP-Coverage Test

The last distinguisher is very similar to the Coverage test described in Section 5.3.1. The only difference is instead of considering the coverage of first l keystream bits, we find the coverage of l bit keystream after the first k bit DP, as given in Figure 5.5.

$$(\underbrace{0,0,\ldots,0}_{k\ bits},\underbrace{*,*,\ldots,*}_{l\ bits})$$

Figure 5.5: Distinguished points

First, we select l random positions (active bits) from IV and fix the rest (inactive) bits to a random value. Then, we synchronize the cipher for all possible 2^l IVs and generate l bit distinguished keystreams. Then, calculate the number of different l-bit keystreams and denote it as C_1. We repeat the experiment for a number of times with different assignments of the inactive IV bits and obtain a coverage variable for each trial, then evaluate the randomness of the cipher based on the distribution of C_i's. The pseudocode of the DP Coverage test is given in Algorithm 5.3.3. To evaluate the output coverage values, the theoretical distribution given in the Coverage test is used.

Distinguished points in TMTO attacks are used to reduce the number of memory checks, since only distinguished keystream portions with a special property are checked. These distinguished portions are assumed to be uniformly distributed throughout the keystream, otherwise it is possible to distinguish the cipher using the Overlapping Template Matching Test from the randomness test suite of NIST [103].

An important criterion against stream ciphers is that the initial states (states that are generated after key/IV initialization phase) should be uniformly distributed throughout the keystream. If for any key, there exist IV_1 and IV_2 that identify close starting points, most important assumption of stream ciphers *keystream must only be used once* may be violated. A similar observation against Grain is pointed out by Küçük [152]. The ciphers having close starting points are expected to reach the same distinguished keystream portions resulting in low coverage.

Algorithm 5.3.3: DP-Coverage Test(R, l, k)

Randomly select K;
Randomly select l positions $p_1, p_2, \ldots, p_l$ from v bits of IV;
for $i \leftarrow 0$ **to** R
{
  Randomly select $IV = (iv_1, iv_2, \ldots, iv_v)$;
  for $j \leftarrow 0$ **to** $2^l - 1$
  {
    $J = (j_1, j_2, \ldots, j_l)$ binary representation of j;
    $(iv_{p_1}, iv_{p_2}, \ldots, iv_{p_l}) = J$;
    $Z^{(j)} = l$ bit keystream after k bit distinguisher using K and IV;
  }
  $Coverage_i$ = Number of distinct $Z^{(1)}, \ldots, Z^{(2^j)}$;
}
Evaluate $(Coverage_1, \ldots, Coverage_R)$ using χ^2 test;
return $(p - value)$

5.4 Experimental Results

We applied the three distinguishers to the Phase III candidates of eSTREAM project with following parameters; Coverage(100,12), Coverage(100,14), ρ(100,15), ρ(100,20), DP-Coverage(100,12,10), DP-Coverage(100,14,10). Each test is repeated 100 times using random keys and the average values are tabulated in Table 5.3. Since the p-values are expected to distribute uniformly between 0 and 1, the average of 100 p-values are expected to be distributed normally with mean 0.5 and standard deviation 0.0289.

Table 5.3: The average 100 p-values obtained from Coverage, ρ and DP-Coverage tests

Cipher	*Coverage Test*		*ρ Test*		*DP Coverage Test*	
	12	14	15	20	12	14
Crypt.MT	0.421172	0.502259	0.438767	0.434522	0.491562	0.496520
Decim	0.483673	0.498243	0.515213	0.561964	0.507519	0.434853
Dragon	0.467800	0.524956	0.531447	0.508126	0.475863	0.490167
Edon80	0.503102	0.504606	0.496585	0.506221	0.458080	0.511870
FFCSR	0.501281	0.536393	0.522143	0.505929	0.487775	0.501770
Grain128	0.507743	0.546400	0.521265	0.473777	0.514002	0.481063
HC-128	0.453212	0.502525	0.489393	0.475678	0.516889	0.513914
Lex	0.472844	0.497004	0.500221	0.475852	0.473196	0.497984
Mickey-128	0.490894	0.499849	0.510405	0.479021	0.434051	0.544828
NLS	0.508358	0.483571	0.474961	0.477997	0.516875	0.468336
Pomaranch	0.433858	0.320433	0.506190	0.520325	0.483106	0.513709
Rabbit	0.512423	0.473658	0.522667	0.518543	0.470558	0.470504
Salsa20	0.485817	0.527911	0.533091	0.501501	0.430587	0.498490
Sosemanuk	0.439461	0.487562	0.497158	0.527524	0.555262	0.531222
Trivium	0.413683	0.500991	0.491455	0.495252	0.458730	0.508625

Most significant deviation from 0.5 is obtained from the cipher Pomaranch [153]

using the coverage test with 14 variable IV bits. For a secure cipher, although the probability that the average is less than 0.320433 is negligible, we repeated the experiment 450 times and obtained the following histogram. As seen from the figure, the distribution of p-values significantly deviates from uniform distribution.

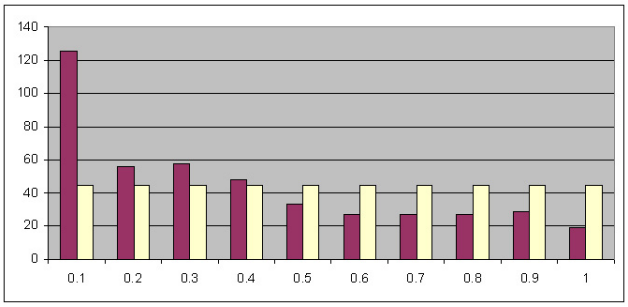


Figure 5.6: The number of p-values in intervals of length 0.1 versus expected values for Pomaranch

5.5 Summary

In this chapter, we propose a new framework of randomness testing based on some properties of random mappings, focusing on TMTO attacks against stream ciphers. We present three chosen IV distinguishers namely; (i) coverage test, (ii) ρ-test and (iii) DP-coverage test. Unlike most of the randomness tests available in the literature, we try to give clues to convert the results to attack the cipher, whenever cipher fails from the tests and we experimentally observed some statistical deviations in the distribution of p-values in Pomaranch.

Chapter 6

Tests Based on Correlations

Correlation is a very commonly used statistics to describe the degree of relationship between two variables. In cryptographic applications, any significant correlation between public and secret variables can be exploited to attack the cipher.

In this chapter, we focus on correlations between input (key and IV) and output (keystream) of ciphers. First, we give the basics of correlation attacks that divide the cipher into several components and exploit the correlations between weak components and known keystream. Then, we present three new statistical tests to analyze the correlation between key, IV and keystream. Finally, we present some experimental results on Phase I candidates of eSTREAM Project.

6.1 Basics of Correlation Attacks

Correlation attacks constitute a type of divide and conquer attacks, where the attacker divides the keystream generator into several weak components and exploits the correlation between the keystream and the output of weak components.

The first correlation attack, proposed by Siegenthaler [132], attempts to analyze the nonlinear combining generator only using the ciphertext information. The statistical relations between output of a subset of LFSRs and the keystream are analyzed to identify internal states of the LFSRs independently. For a nonlinear combiner with n LFSRs each having length l_i, if a significant correlation for each LFSR output and keystream is available, the complexity of recovering the secret internal state reduces to $\sum_{i=1}^{n}(2^{l_i}-1)$ from $\prod_{i=1}^{n}(2^{l_i}-1)$. This attack is very practical especially if the $max\{l_i\}$ is less than 50. To avoid attacks against combining generators, the nonlinear combiner should have high correlation immunity. However, it should be noted that there is a tradeoff between the correlation immunity and the nonlinearity of the combining

function.

A disadvantage of the attack proposed by Siegenthaler is that the initial state of each LFSR has to be found using exhaustive search. Meier and Staffelbach [133] improved the idea and proposed fast correlation attacks based on decoding algorithms.

Let L be the set of binary sequences. For a LFSR with length l, there are 2^l possible different LFSR sequences. The set of truncated sequences of L is a linear $[N, l]$ block code C, for a fixed N. Then, N bit output of a LFSR is a codeword from the code C. Assumed that the codeword is transferred though a Binary Symmetric Channel (BSC) where probability of error is p as given in Figure 6.1. Then, the keystream z_i can be regarded as the received channel output. By using error correcting techniques, the initial state of the target LFSR can recovered, so the requirement to search for all possible initializations is eliminated.

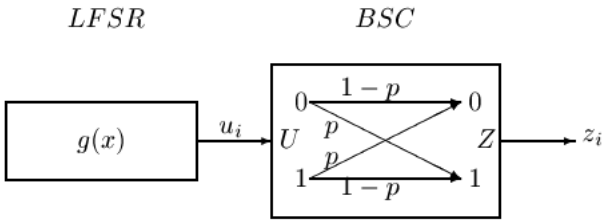


Figure 6.1: Use of BSC in fast correlation attacks

Fast correlation attacks have two different phases; (i) finding a set of suitable parity check equations, and (ii) using these equations to recover the initial state of the LFSR. For a LFSR with feedback polynomial $g(x) = 1 + c_1x + c_2x^2 + \ldots + c_lx^l$, the output sequence $\{u_i\}$ satisfies the following recursion;

$$u_i = c_1u_{i-1} + c_2u_{i-2} + \ldots + c_lu_{i-l}, i > l. \tag{6.1.1}$$

Let t be the number of tab points (number of nonnegative c_i's). Using the recursion given in (6.1.1), $t + 1$ different parity check equations for u_i can be found. Also, using the fact that $g(x)^j = g(x^j)$ for $j = 2^k$, more equations can be generated by repeatedly squaring the polynomial $g(x)$. Using these methods, the total number of parity check equations [133] is approximately

$$m = (t + 1)log(\frac{N}{2l}). \tag{6.1.2}$$

In the second phase, these parity check equations are used to decode the keystream.

Two different decoding methods, Algorithm A and Algorithm B are proposed in [133]. Success of these algorithms depend on the number of tabs t and for the attack to be successful t should be small. As t gets large more equations are needed to succeed. Some improvements for finding more parity check equations are proposed in [154].

As an example, in [155], an irregularly decimated filter generator LILI-128 is attacked using fast correlation attacks with complexity around 2^{71} operations, precomputation complexity of 2^{79} table lookups and a keystream of length 128 MByte.

6.2 Tests Based on Correlation of Key, IV and Keystream

Considering the importance of correlation of input and output in analysis of stream ciphers, we proposed three new statistical tests.

- *Key/Keystream Correlation Test* considers the correlation between key and the corresponding keystream using a fixed IV.
- *IV/Keystream Correlation Test* considers the correlation between IV and the corresponding keystream using a fixed key.
- *Frame Correlation Test* considers the correlation between keystreams using different IV values.

The first test uses different keys, whereas the other two tests use a chosen IV approach. If a cipher fails any of these tests, it can be concluded that the correlations between key, IV and keystream are significant, therefore designers should revise their cipher to remove this correlation.

The following notation is used in the next sections. For a stream cipher S with k-bit key K, v-bit initialization vector V, let $S(K, V, l)$ be the first l bits of the keystream generated by the cipher S, key K and initialization vector V.

6.2.1 Key/Keystream Correlation Test

The purpose of this test is to evaluate the bitwise correlation between the key and the first k bits of keystream. Significant correlation between key and keystream may enable the cryptanalyst to recover the secret key using the keystream or may reduce the exhaustive search space for key significantly. If the cipher fails this test, key loading part of the initialization phase should be revised. For ciphers with iterative initialization function, the number of rounds may be increased to remove correlation.

In this test, firstly IV is fixed to a random value and m key values are generated randomly. Next, keystream $z_1, \ldots, z_k$ of length k, is produced for each key. To evaluate the correlation between key and its corresponding keystream, they are XORed and weight of the resulting sequence is calculated. The pseudocode of the test is given in Algorithm 6.2.1.

Small weights indicate that the key and its corresponding keystream are similar, meaning they are positively correlated. High weights point to a negative correlation between i^{th} bit of key and i^{th} bit of keystream for $i = 1, \ldots, k$.

Algorithm 6.2.1: KEY/KEYSTREAM CORRELATION(m)

Fix V;
for $i \leftarrow 1$ **to** m
$\begin{cases} \text{Choose } K_i \text{ randomly;} \\ w_i = \text{weight of } S(K_i, V, k) \oplus K_i; \end{cases}$
Evaluate w_i's using χ^2-goodness of fit tests;
return $(p - value)$

For a secure cipher, distribution of the weights is Binomial with parameters k and $1/2$ with given probability distribution

$$P(weight = x) = \binom{k}{x}(1/2)^k,$$

for weights between 0 and k.

Using χ^2-goodness of fit tests, obtained m weights are compared to the Binomial distribution by categorizing weights into five intervals with approximately equal probabilities. The intervals and corresponding probabilities for $k = 80$ and $k = 128$ are given in Table 6.1. For the χ^2 test to be applicable, the value of m should at least be 100.

6.2.2 IV/Keystream Correlation Test

The purpose of this test is to evaluate the bitwise correlation between IV and the first v bits of keystream. Significant correlation between IV and keystream may lead to generation of keystream without knowing the value of secret key. If the cipher fails this test, IV loading part of the initialization phase should be revised. Similar to the previous test, the number of rounds may be increased for ciphers with iterative initialization function.

Table 6.1: Interval and probabilities values of Key/Keystream Correlation test for key size of 80 and 128 bits

Key Size = 80		*Key Size = 128*	
Interval	*Probability*	*Interval*	*Probability*
0 - 35	0.157153	0 - 58	0.165468
36 - 38	0.211624	59 - 62	0.230035
39 - 41	0.262446	63 - 65	0.208994
42 - 44	0.211624	66 - 69	0.230035
45 - 80	0.157153	70 - 128	0.165468

Firstly, the value of key is fixed to a random value and m random IVs are selected. Then, a keystream of length v is produced using each IV value and the fixed key. To evaluate the correlation, IV and its corresponding keystream are XORed and its weight is calculated. The pseudocode of the test is given in Algorithm 6.2.2.

Algorithm 6.2.2: IV/KEYSTREAM CORRELATION(m)

Fix K;
for $i \leftarrow 1$ **to** m
$\begin{cases} \text{Select } V_i \text{ randomly;} \\ w_i = \text{weight of } S(K, V_i, v) \oplus V_i; \end{cases}$
Evaluate w_i's using χ^2-goodness of fit tests;
return $(p - value)$

Similar to the previous test, small weights indicate that the IV and its corresponding keystream are similar, meaning they are positively correlated. High weights point to a negative correlation between i^{th} bit of IV and i^{th} bit of keystream for $i = 1, \ldots, v$. The smallest IV size is chosen to be 64.

Using the χ^2-goodness of fit tests, obtained weights are compared to the Binomial distribution by categorizing into five intervals with approximately equal probabilities. The intervals and corresponding probabilities for $v = 64, 80$ and 128 are given in Table 6.2. For χ^2 test to be applicable, the value of m should at least be 100.

6.2.3 Frame Correlation Test

In synchronous stream ciphers, after generating a fixed length keystream called *frame*, IV values are updated. Since counters are commonly used as IVs, two consecutive IV values are similar. The purpose of this test is to analyze the correlation between

Table 6.2: Interval and probability values of IV/Keystream Correlation test for IV size of 64, 80 and 128.

IV Size = 64		*IV Size = 80*		*IV Size = 128*	
Interval	*Probability*	*Interval*	*Probability*	*Interval*	*Probability*
0 - 28	0.190866	0 - 35	0.157153	0 - 58	0.165468
29 - 30	0.163124	36 - 38	0.211624	59 - 62	0.230035
31 - 33	0.292019	39 - 41	0.262446	63 - 65	0.208994
34 - 35	0.163124	42 - 44	0.211624	66 - 69	0.230035
36 - 64	0.190866	45 - 80	0.157153	70 - 128	0.165468

frames generated using similar IVs. In this test, first a random key and an IV value are selected, then a keystream of length l is produced. This procedure is repeated m times with incremented values of IV. Using these keystreams, a matrix of size $m \times l$ is generated and the column weights of the matrix are calculated. Distribution of the weights is approximately $N(m/2, m/4)$, when m is large. Columns with very high/low weight indicate weaknesses due to insecure resynchronization. The χ^2-goodness of fit test is applied to evaluate the correlation between frames. If the cipher fails this test, IV loading part of initialization phase should be revised.

Algorithm 6.2.3: FRAME CORRELATION(l, m)

Randomly choose K and V;
for $i \leftarrow 1$ **to** m
$\begin{cases} (Z_{i_1}, \ldots, Z_{i_l}) = S(K, V, l); \\ \text{Increment } V; \end{cases}$
$w_i = \sum_j Z_{i_j}$ for $i = 1, \ldots l$;
Evaluate w_i's using χ^2-goodness of fit tests;
return $(p - value)$

6.3 Experimental Results

The proposed tests based on correlation of key, IV and keystream are applied to the candidates of Phase I candidates eSTREAM with the following settings.

- *Key/Keystream Correlation Test*: $m = 2^{20}$ keys are generated randomly and for each key, keystream of length k (80 or 128 bits) is generated using a zero vector as IV.

- *IV/Keystream Correlation Test*: $m = 2^{20}$ IVs and a fixed key are generated randomly and for each IV, keystream of length v (64, 80 or 128 bits) is generated.
- *Frame Correlation Test*: Starting with the IV 0x00000001 and incrementing until the IV 0x00100000, $m = 2^{20}$ keystreams of length $l = 256$ bits are generated with a fixed random key.

Table 6.3 lists the average results of three tests for each Phase I candidate of eSTREAM. Each test is repeated 100 times, and the p-values significantly different from 0.5 are assumed to be weak. Following ciphers are detected to be weak as results of the proposed tests.

Decim

Decim [80] is hardware oriented cipher based on both a nonlinear filter LFSR and an irregular decimation mechanism called ABSG. Decim fails all three tests meaning there exist a significant correlation between (i) key and keystream bits, (ii) IV and keystream and (iii) frames generated using similar IVs.

F-FCSR-8

F-FCSR-8 [74] is a software oriented cipher in which the output is obtained by filtering internal states of an FCSR automaton using linear Boolean functions. The cipher fails the frame correlation test meaning there exists a significant correlation between keystreams generated by similar IVs. However, no weaknesses are observed using the other two tests.

Frogbit

Frogbit [156] is a software oriented patented stream cipher including a message authentication code. According the our test results, Frogbit fails the frame correlation test meaning the frames generated using different IVs are correlated.

Mag

Mag [106] is proposed for both software and hardware applications. Due to the small IV size of Mag (32 bit), the *IV/Key Correlation Test* is not applied. According the our test results, the cipher fails the frame correlation test meaning the frames generated using different IVs are correlated.

Zk-Crypt

The ZK-Crypt [110] is a hardware oriented stream cipher utilizing permutations and non-linear correlation immunizing parallel and serially operating hardware functions. According the our tests, the 29^{th} and 30^{th} bits of IV do not satisfy the desired diffusion in Zk-Crypt, therefore the cipher fails the frame correlation test. However, no weaknesses are observed using the other two tests.

For a secure cipher, it is very unlikely to obtain an average of 100 p-values less than 0.4 and greater than 0.6. The p-values outside the interval $(0.4 - 0.6)$ are also assumed to be suspicious and highlighted in the table. Suspicious results are obtained for Edon80, Fubuki, Mickey, Rabbit, TSC-3, Vest-4, WG and Yamb.

6.4 Summary

In this chapter, considering the important class of divide and conquer attacks, we focused on the correlation attacks and the correlations between key, IV and keystream bits. Different from correlation attacks, we presented tests that do not consider the inner structure of ciphers, but focus on the input and output values. Key/Keystream and IV/Keystream correlation tests do not analyze the correlations between i^{th} bit of keystream to j^{th} bit of key and IV ($i \neq j$), respectively. As trivial extension of these tests, the correlations of key and keystream bits with different indices can be studied by applying a permutation to keystream bits. Depending on the correlation of key and keystream, first test can be used to recover key.

Table 6.3: The average of 100 p-values obtained using Key/Keystream Correlation, IV/Keystream Correlation and Frame Correlation Test against Phase I candidates of eSTREAM

Cipher	*Key Size*	*IV Size*	*Key/Keystream Correlation*	*IV/Keystream Correlation*	*Frame Correlation*
ABC v.2	128	128	0.594847	0.581965	0.499155
Achterbahn	80	64	0.447456	0.509330	0.445827
CryptMT	128	128	0.467099	0.565633	0.504488
Decim	80	64	0.000000	0.000000	0.000000
Dicing	128	64	0.510794	0.518198	0.469110
Dragon	128	128	0.515234	0.509020	0.493730
Edon80	80	64	0.395271	0.490394	0.450119
F-FCSR-8	128	128	0.465567	0.497677	0.000000
Frogbit	128	128	0.469433	0.605391	0.000000
Fubuki	128	128	0.428599	0.660642	0.449259
Grain	80	64	0.424768	0.454764	0.498692
HC-256	128	64	0.453745	0.546300	0.513012
Hermes8	128	128	0.528543	0.504307	0.496917
LEX	128	128	0.569770	0.412822	0.507724
Mag	128	32	0.534794	-	0.000000
Mickey	80	64	0.576635	0.475921	0.608017
Mickey-128	128	128	0.525417	0.657516	0.528486
Mir-1	128	64	0.518211	0.490121	0.415622
NLS	128	128	0.526565	0.472976	0.525124
Phelix	128	128	0.506611	0.506333	0.521635
Polar Bear	128	128	0.447812	0.482900	0.487471
Pomaranch	128	64	0.462254	0.502730	0.591795
Py	128	64	0.496108	0.443042	0.522252
Rabbit	128	64	0.547722	0.503607	0.608813
Salsa20	128	64	0.504370	0.535973	0.510886
SFINKS	80	80	0.540354	0.497505	0.529415
Sosemanuk	128	64	0.471755	0.507739	0.516827
Trivium	80	64	0.479925	0.504355	0.518177
TSC-3	128	64	0.521729	0.351436	0.485147
Vest-4	128	64	0.526154	0.657904	0.512552
WG	128	128	0.532289	0.363903	0.518250
Yamb	128	64	0.326160	0.427037	0.497137
Zk-Crypt	128	128	0.593001	0.468749	0.000000

Chapter 7

Conclusion

In this thesis, we have studied *statistical analysis of synchronous stream ciphers* which is an important class of symmetric ciphers. Since proving the security of stream ciphers are extremely hard for most ad hoc designs, these tests play an important role in the design of cryptosystems, since in some cases, black box distinguishers are able to detect weaknesses that are hard to detect by theoretical analysis. Additionally, most of them have very low complexity and provide results in a very short time. Although black box distinguishers do not consider the inner structure of ciphers, to increase the success rate of the distinguishers, some properties of the cipher can be used as input to the distinguisher. If a cipher fails any of the proposed tests, it is for certain that the design should be reevaluated and necessary changes should be applied.

The contributions of the thesis and some future research suggestions are given below.

- Considering stream ciphers as PRNGs, we have focused on the statistical test suites that are used to evaluate PRNGs. We have emphasized the importance of independence of randomness tests in test suites. We have presented some theoretical and experimental results on relations of ten commonly used randomness tests. We have experimentally observed that frequency, overlapping template (with input template 111), longest run of ones, random walk height tests and maximum order complexity tests produce correlated results for short sequences. It is suggested that these correlations are considered while analyzing generators using short sequences. Moreover, for testing longer sequences with level-2 version of these tests, these correlations should still be taken into account.

- We have defined the concept of *sensitivity*, the effect of simple transformations to input sequences on the test results. If a transformation significantly changes the output p-values, then we have proposed to include the composition of transformation and the test to the suite to increase the coverage. Ideally, we would like

to have each test applied to a transformed sequence $\sigma(S)$ to be independent of all different tests applied to the original sequence. Clearly, as the set of allowable transformations grows, this becomes harder to achieve. By choosing a good set of allowable transformations, one can use a given set of tests in a more powerful fashion. For example, one should not introduce unnatural transformations of the data, but stick to a set of transformations which are generated by a small set of basic transformations, such as the ones presented in the thesis. It is of interest to investigate this problem further in future work.

- We have proposed three randomness tests based on one dimensional random walks namely; (i) *random walk excursion*, (ii) *random walk height* and (iii) *random walk expansion*. The exact distributions of the test statistics are provided, therefore tests are suitable to analyze short sequences, unlike the random walk tests based in NIST suite. Moreover, we have experimentally showed that for short sequences, the random walk expansion test is independent of some of the commonly used randomness tests such as frequency, overlapping templates, longest run of ones and random walk height tests, therefore significantly increases the coverage of test suites containing given tests. Proposed random walk tests can be extended using two or more dimensional random walks.

- We have analyzed the stream ciphers based on the properties of Boolean functions F_i's that generate the i^{th} keystream bit from key and IV. We have generalized the idea of Filiol [20] and Saarinen [157], and have proposed a framework for chosen IV statistical attacks using a polynomial description and we have experimentally detected statistical weaknesses in the keystream of Trivium using an initialization reduced to 736 rounds as well as in some state bits of Grain-128 with full IV initialization. In our tests, weak IV bits are selected randomly and how to select weak IV bits intelligently is an open question.

- We have focused on the initialization of Trivium which is one of the focus ciphers of eSTREAM. We have obtained a linear approximation for the Boolean function F_1 with bias 2^{-31} valid for a subset of keys and IVs for Trivium with 288 initial clocking. Using this approximation, it is possible to attack the cipher using 2^{62} resynchronizations, given that one of the weak keys are used. Using multiple and nonlinear approximations, better biases may be obtained. Also, different modeling of initializations may lead to better approximations.

- We have proposed the *linear span test* which is an alternative application of Binary Matrix Rank test given in NIST suite [103]. Linear span test analyzes the effect of linear relation of IVs on the relations of keystream blocks. We have

applied the test on Phase III candidates of eSTREAM, and none of the candidates show statistical weaknesses based on the test. It is of interest to improve the linear span test so that it is possible to recover the key whenever a weaknesses is observed.

- We have also tested the Phase I candidates of eSTREAM using the *diffusion* test that considers the effect of each bit of key and IV on the keystream. Severe statistical weaknesses are observed in Phase I versions of Frogbit, Mag and Zk-Crypt. Also, the results obtained from the ciphers ABC, CryptMT, Decim, Dicing, Mir-1, Sfinks and TSC-3 seem to be suspicious.

- Considering the generic TMTO attacks against stream ciphers, we have proposed three chosen IV distinguishers namely; (i) *coverage*, (ii) ρ and (iii) *DP-coverage tests*. Using the coverage test with 14 variable IV bits, we have experimentally observed positive skewness in the distribution of p-values in Pomaranch, due to this deviation the average of 100 p-values which is expected to be 0.5 is obtained as 0.320433. Improvement of these tests so that it is possible to recover the key is of interest.

- We have focused on correlations between input (key and IV) and output (keystream) of ciphers, since any significant correlation between public and secret variables can be exploited to attack the cipher. To test these correlations, we have proposed three distinguishers namely; (i) *key/keystream correlation*, (ii) *IV/keystream correlation* and (iii) *frame correlation tests*. We have experimentally observed significant correlations in Decim, F-FCSR-8, Frogbit, Mag and Zk-crypt. Key/Keystream and IV/Keystream correlation tests do not consider the correlations between i^{th} bit of keystream to j^{th} bit of key and IV ($i \neq j$), respectively. As trivial extension of these tests, the correlations of key and keystream bits with different indices can be analyzed by applying a permutation to keystream bits.

References

[1] S. Fischer, S. Khazaei, and W. Meier. Chosen IV Statistical Analysis for Key Recovery on Stream Ciphers. SASC 2008 - The State of the Art of Stream Ciphers, 2008.

[2] R. L. Rivest. The MD5 message digest algorithm, 1992. `http://theory.lcs.mit.edu/rivest/rfc1321.txt`. Request For Comments 1321.

[3] *Secure Hash Standard*. National Institute of Standards and Technology, Washington, 2002. `http://csrc.nist.gov/publications/fips/`. Note: Federal Information Processing Standard 180-2.

[4] H. Dobbertin, A. Bosselaers, and B. Preneel. RIPEMD-160: A strengthened version of RIPEMD. In D. Gollmann, editor, *Fast Software Encryption*, volume 1039 of *Lecture Notes in Computer Science*, pages 71–82. Springer, 1996.

[5] R. L. Rivest. The MD4 message digest algorithm. In *CRYPTO '90: Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology*, pages 303–311, London, UK, 1991. Springer-Verlag.

[6] X. Wang, D. Feng, X. Lai, and H. Yu. Collisions for hash functions MD4, MD5, HAVAL–128 and RIPEMD, 2004. URL: `http://eprint.iacr.org/2004/199/`.

[7] F. Chabaud and A. Joux. Differential collisions in SHA-0. In *CRYPTO '98: Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology*, pages 56–71, London, UK, 1998. Springer-Verlag.

[8] X. Wang and H. Yu. How to break MD5 and other hash functions. In Cramer [158], pages 19–35.

[9] E. Biham, R. Chen, A. Joux, P. Carribault, C. Lemuet, and W. Jalby. Collisions of SHA-0 and reduced SHA-1. In Cramer [158], pages 36–57.

[10] Announcing the Development of New Hash Algorithm(s) for the Revision of Federal Information Processing Standard (FIPS) 180-2, Secure Hash Standard.

http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Jan07.pdf, 2007.

[11] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[12] SIG Bluetooth. Bluetooth specification. Available at http://www.bluetooth.com, Accessed May 1, 2007, 2003.

[13] J. Daemen and V. Rijmen. *The Design of Rijndael*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[14] B. Preneel. New European Schemes for Signature, Integrity and Encryption (NESSIE): A Status Report. In *PKC '02: Proceedings of the 5th International Workshop on Practice and Theory in Public Key Cryptosystems*, pages 297–309, London, UK, 2002. Springer-Verlag.

[15] H. Imai and A. Yamagishi. CRYPTREC Project - Cryptographic Evaluation Project for the Japanese Electronic Government. In *ASIACRYPT '00: Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security*, pages 399–400, London, UK, 2000. Springer-Verlag.

[16] eSTREAM, the ECRYPT Stream Cipher Project. http://www.ecrypt.eu.org/stream, 2004.

[17] C.E. Shannon. Communication Theory of Secrecy Systems. *Bell System Technical Journal*, 28:656–715, 1949.

[18] A. Doğanaksoy, Ç. Çalık, F. Sulak, and M. Sönmez Turan. New randomness tests using random walk. In *II. Ulusal Kriptoloji Sempozyumu*, 2006.

[19] M. Sönmez Turan, A. Doğanaksoy, and S. Boztaş. On Independence and Sensitivity of Statistical Randomness Tests. In *International Conference on Sequences and Their Applications (SETA)*, Lecture Notes in Computer Science. Springer, 2008.

[20] E. Filiol. A new statistical testing for symmetric ciphers and hash functions. In V. Varadharajan and Y. Mu, editors, *International Conference on Information, Communications and Signal Processing*, volume 2119, pages 21–35, 2001.

[21] Håkan Englund, Thomas Johansson, and Meltem Sönmez Turan. A framework for chosen iv statistical analysis of stream ciphers. In K. Srinathan, C. Pandu

Rangan, and Moti Yung, editors, *INDOCRYPT*, volume 4859 of *Lecture Notes in Computer Science*, pages 268–281. Springer, 2007.

[22] M. Sönmez Turan and O. Kara. Linear approximations for 2-round trivium. In *Proc. First International Conference on Security of Information and Networks (SIN 2007)*, pages 96–105. Trafford Publishing, 2007.

[23] M. Sönmez Turan, A. Doğanaksoy, and Ç. Çalık. Statistical analysis of synchronous stream ciphers. *SASC 2006: Stream Ciphers Revisited*, 2006.

[24] M. Sönmez Turan, Ç. Çalık, N. Buz Saran, and A. Doğanaksoy. New distinguishers based on random mappings against stream ciphers. In *International Conference on Sequences and Their Applications (SETA)*, Lecture Notes in Computer Science. Springer, 2008.

[25] P. Ekdahl and T. Johansson. A new version of the stream cipher SNOW. In *SAC '02: Revised Papers from the 9th Annual International Workshop on Selected Areas in Cryptography*, pages 47–61, London, UK, 2003. Springer-Verlag.

[26] R. A. Rueppel. *Analysis and Design of Stream Ciphers*. Springer-Verlag New York, Inc., New York, NY, USA, 1986.

[27] S. W. Golomb. *Shift Register Sequences*. Aegean Park Press, Laguna Hills, CA, USA, 1981.

[28] S. W. Golomb and G. Gong. *Signal Design for Good Correlation: For Wireless Communication, Cryptography, and Radar*. Cambridge University Press, New York, NY, USA, 2004.

[29] P. Hawkes and G. G. Rose. Exploiting Multiples of the Connection Polynomial in Word-Oriented Stream Ciphers. In *ASIACRYPT '00: Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security*, pages 303–316, London, UK, 2000. Springer-Verlag.

[30] M. Hell, T. Johansson, and W. Meier. Grain - A Stream Cipher for Constrained Environments. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/010, 2005. http://www.ecrypt.eu.org/stream.

[31] B. Gammel, R. Göttfert, and O. Kniffler. The Achterbahn Stream Cipher. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/002, 2005. http://www.ecrypt.eu.org/stream.

[32] C. De Cannière and B. Preneel. Trivium - a stream cipher construction inspired by block cipher design principles. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/030, 2005. http://www.ecrypt.eu.org/stream.

[33] C. Bigeard, S. O'Neil, B. Gittins, and H. Landman. VEST hardware dedicated stream ciphers. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/032, 2005. http://www.ecrypt.eu.org/stream.

[34] E. Dawson, K. Chen, M. Henricksen, W. Millan, L. Simpson, and S. Moon H. Lee. Dragon: A Fast Word Based Stream Cipher. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/006, 2005. http://www.ecrypt.eu.org/stream.

[35] D. Ferrero, R. Gonzalo, and M. Soriano. Some properties of non linear feedback shift registers with maximum period. *Proc. Sixth Int. Conf. Telecommunications Systems*, 1998.

[36] J. L. Massey and R. W. Liu. Equivalence of Nonlinear Shift-Registers. *Information Theory, IEEE Transactions*, 10, 1964.

[37] A. Klapper and M. Goresky. Feedback shift registers, 2-adic span, and combiners with memory. *Journal of Cryptology*, 10(2):111–147, 1997.

[38] A. Klapper. A survey of feedback with carry shift registers. In Tor Helleseth, Dilip V. Sarwate, Hong-Yeop Song, and Kyeongcheol Yang, editors, *SETA*, volume 3486 of *Lecture Notes in Computer Science*, pages 56–71. Springer, 2004.

[39] C.J.A. Jansen. Stream cipher design based on jumping finite state machines. Cryptology ePrint Archive, Report 2005/267, 2005.

[40] C. Jansen and A. Kolosha. Cascade Jump Controlled Sequence Generator. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/022, 2005.

[41] C. Carlet and P. Guillot. A new representation of Boolean functions. In *AAECC-13: Proceedings of the 13th International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, pages 94–103, London, UK, 1999. Springer-Verlag.

[42] Elif Yıldırım, Zülfükar Saygı, Meltem Sönmez Turan, and Ali Doğanaksoy. A statistical approach on the number of functions satisfying Strict Avalanche Criteria. In Jean-Francis Michon, Pierre Valarcher, and Jean-Baptiste Yunès, editors, *Proceedings of BFCA'05 Conference, March 7–8, 2005 Rouen, France*, pages 39–48, 2005.

[43] T. Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Transactions on Information Theory*, 30(5):776–, 1984.

[44] G. G. Rose. A stream cipher based on linear feedback over GF(2^8). In *ACISP '98: Proceedings of the Third Australasian Conference on Information Security and Privacy*, pages 135–146, London, UK, 1998. Springer-Verlag.

[45] A. Klimov and A. Shamir. Cryptographic applications of t-functions. In Matsui and Zuccherato [159], pages 248–261.

[46] A. Maximov. A New Stream Cipher Mir-1. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/017, 2005. http://www.ecrypt.eu.org/stream.

[47] V. Anashin, Bogdanov A., Kizhvatov I., and Kumar S. ABC: A New Fast Flexible Stream Cipher. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/001, 2005.

[48] J. Hong, D. H. Lee, Y. Yeom, D. Han, and S. Chee. T-function based stream cipher TSC-4. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/024, 2005. http://www.ecrypt.eu.org/stream.

[49] A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1:1–11, 1965.

[50] B. Schneier. *Applied cryptography (2nd ed.): protocols, algorithms, and source code in C*. John Wiley & Sons, Inc., New York, NY, USA, 1995.

[51] L. Lee and K. Wong. An elliptic curve random number generator. In R. Steinmetz, J. Dittmann, and M. Steinebach, editors, *Communications and Multimedia Security*, volume 192 of *IFIP Conference Proceedings*. Kluwer, 2001.

[52] J. Szczepanski, E. Wajnryb, J. Amigo, Maria V. Sanchez-Vives, and M. Slater. Biometric random number generators. *Computers & Security*, 23(1):77–84, 2004.

[53] J. Massey. Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, 15:122–127, 1969.

[54] R. T. C. Kwok and M. Beale. Aperiodic linear complexities of de bruijn sequences. In *CRYPTO '88: Proceedings on Advances in cryptology*, pages 479–482, New York, NY, USA, 1990. Springer-Verlag New York, Inc.

[55] M. Stamp and C. F. Martin. An algorithm for the k-error linear complexity of binary sequences with period 2^n. *IEEE Transactions on Information Theory*, 39(4):1398–, 1993.

[56] A. G. B. Lauder and K. G. Paterson. Computing the error linear complexity spectrum of a binary sequence of period 2^n. *IEEE Transactions on Information Theory*, 49(1):273–280, 2003.

[57] W. Meidl and H. Niederreiter. Counting functions and expected values for the k-error linear complexity. *Finite Fields Appl*, 8:142–154, 2002.

[58] W. Meidl and H. Niederreiter. Counting functions and expected values for the k-error linear complexity. *Finite Fields Appl*, 8:142–154, 2002.

[59] C.J.A Jansen. *Investigations on nonlinear streamcipher systems: Construction and evaluation methods*. Ph.d. thesis, Technical University of Delft, 1989.

[60] D. Erdmann and S. Murphy. An approximate distribution for the maximum order complexity. *Designs Codes Cryptography*, 10(3):325–339, 1997.

[61] A.M. Youssef and G. Gong. On the quadratic span of binary sequences. http://citeseer.ist.psu.edu/309367.html.

[62] A. H. Chan and R. A. Games. On the quadratic spans of periodic sequences. In *CRYPTO '89: Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology*, pages 82–89, London, UK, 1990. Springer-Verlag.

[63] A. Doğanaksoy and F. Göloğlu. On Lempel-Ziv complexity of sequences. In G. Gong, T. Helleseth, H. Song, and K. Yang, editors, *SETA*, volume 4086 of *Lecture Notes in Computer Science*, pages 180–189. Springer, 2006.

[64] M. Matsumoto, M. Saito, T. Nishimura, and M. Hagita. CryptMT Stream Cipher Version 3. eSTREAM, ECRYPT Stream Cipher Project, Report 2007/028, 2007. http://www.ecrypt.eu.org/stream.

[65] C. Berbain, O. Billet, A. Canteaut, N. Courtois, B. Debraize, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin, and H. Sibert. Decimv2. eSTREAM, ECRYPT Stream Cipher Project, 2007. http://www.ecrypt.eu.org/stream.

[66] C. Berbain, O. Billet, A. Canteaut, N. Courtois, B. Debraize, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin, and H. Sibert. Decim-128. eSTREAM, ECRYPT Stream Cipher Project, 2007. http://www.ecrypt.eu.org/stream.

[67] A. Biryukov. A New 128-bit Key Stream Cipher LEX. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/012, 2005. http://www.ecrypt.eu.org/stream.

[68] H. Wu. Stream Cipher HC-256. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/011, 2005. http://www.ecrypt.eu.org/stream.

[69] G. Rose, P. Hawkes, M. Paddon, and M. W. de Vries. Primitive Specification for NLS. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/019, 2005. http://www.ecrypt.eu.org/stream.

[70] M. Boesgaard, M. Vesterager, T. Christensen, and E. Zenner. The Stream Cipher Rabbit. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/024, 2005. http://www.ecrypt.eu.org/stream.

[71] D. J. Bernstein. Salsa20 Design. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/025, 2005. http://www.ecrypt.eu.org/stream.

[72] C. Berbain, O. Billet, A. Canteaut, N. Courtois, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M Minier, T. Pornin, and H. Sibert. Sosemanuk, a Fast Software-oriented Stream Cipher. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/027, 2005. http://www.ecrypt.eu.org/stream.

[73] D. Gligoroski, S. Markovski, L. Kocarev, and M. Gusev. Edon80. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/007, 2005. http://www.ecrypt.eu.org/stream.

[74] T. Berger, F. Arnault, and C. Lauradoux. F-FCSR. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/008, 2005. http://www.ecrypt.eu.org/stream.

[75] S. Babbage and M. Dodd. The Stream Cipher MICKEY (version 1). eSTREAM, ECRYPT Stream Cipher Project, Report 2005/015, 2005. http://www.ecrypt.eu.org/stream.

[76] J. Daemen and P. Kitsos. The self-synchronizing stream cipher moustique. eSTREAM, ECRYPT Stream Cipher Project, 2006.

[77] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography.* CRC Press, Boca Raton, Florida, 1996. URL: `http://cacr.math.uwaterloo.ca/hac`.

[78] E. Filiol. Decimation attack of stream ciphers. In B. K. Roy and E. Okamoto, editors, *INDOCRYPT*, volume 1977 of *Lecture Notes in Computer Science*, pages 31–42. Springer, 2000.

[79] A. Braeken and J. Lano. On the (im)possibility of practical and secure nonlinear filters and combiners. In *Selected Areas in Cryptography*, pages 159–174, 2005.

[80] C. Berbain, O. Billet, A. Canteaut, N. Courtois, B. Debraize, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin, and H. Sibert. Decim, A New Stream Cipher for Hardware Applications. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/004, 2005. http://www.ecrypt.eu.org/stream.

[81] D. Coppersmith, H. Krawczyk, and Y. Mansour. The shrinking generator. In Douglas R. Stinson, editor, *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 22–39. Springer, 1993.

[82] W. Meier and O. Staffelbach. The self-shrinking generator. In *EUROCRYPT*, pages 205–214, 1994.

[83] S. R. Blackburn. The linear complexity of the self-shrinking generator. *IEEE Transactions on Information Theory*, 45(6):2073–2077, 1999.

[84] L. Simpson, E. Dawson, J. Dj. Golic, and W. Millan. LILI keystream generator. In *SAC '00: Proceedings of the 7th Annual International Workshop on Selected Areas in Cryptography*, pages 248–261, London, UK, 2001. Springer-Verlag.

[85] D. Whiting, B. Schneier, S. Lucks, and F. Muller. Phelix, Fast Encryption and Authentication in a Single Cryptographic Primitive. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/020, 2005. http://www.ecrypt.eu.org/stream.

[86] E. Biham, R. J. Anderson, and L. R. Knudsen. Serpent: A new block cipher proposal. In S. Vaudenay, editor, *Fast Software Encryption*, volume 1372 of *Lecture Notes in Computer Science*, pages 222–238. Springer, 1998.

[87] C. Berbain, H. Gilbert, and J. Patarin. Quad: A practical stream cipher with provable security. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 109–128. Springer, 2006.

[88] B. Yang, O. C. Chen, D. J. Bernstein, and J. Chen. Analysis of QUAD. In A. Biryukov, editor, *FSE*, volume 4593 of *Lecture Notes in Computer Science*, pages 290–308. Springer, 2007.

[89] R.L. Rivest. The RC4 encryption algorithm. RSA Data Security, Inc, 1992.

[90] M. Boesgaard, M. Vesterager, T. Pedersen, J. Christiansen, and O. Scavenius. Rabbit: A new high-performance stream cipher. In Thomas Johansson, editor, *FSE*, volume 2887 of *Lecture Notes in Computer Science*, pages 307–329. Springer, 2003.

[91] P. Hawkes and G. Rose. On the Applicability of Distinguishing Attacks Against Stream Ciphers. 2002.

[92] E. Zenner and M. Boesgaard. How Secure is Secure? On Message and IV Lengths for Synchronous Stream Ciphers. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/039, 2005.

[93] A. Maximov. Cryptanalysis of the "Grain" family of stream ciphers. In *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 283–288, New York, NY, USA, 2006. ACM.

[94] J. Dj. Golić. Linear models for keystream generators. *IEEE Trans. Comput.*, 45(1):41–49, 1996.

[95] J. Dj. Golić. Intrinsic statistical weakness of keystream generators. In *ASIACRYPT*, pages 91–103, 1994.

[96] P. Ekdahl and T. Johansson. Distinguishing attacks on sober-t16 and t32. In J. Daemen and V. Rijmen, editors, *FSE*, volume 2365 of *Lecture Notes in Computer Science*, pages 210–224. Springer, 2002.

[97] D. Watanabe, A. Biryukov, and C. De Cannière. A distinguishing attack of snow 2.0 with linear masking method. In Matsui and Zuccherato [159], pages 222–233.

[98] S. Khazaei, M. M. Hasanzadeh, and M. S. Kiaei. Linear Sequential Circuit Approximation of Grain and Trivium Stream Ciphers. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/063, 2005.

[99] H. Englund, M. Hell, and T. Johansson. A note on distinguishing attacks. In *Information Theory for Wireless Networks, 2007 IEEE Information Theory Workshop*, pages 1–4, 2007.

[100] C. De Cannire, J. Lano, B. Preneel, and J. Vandewalle. Distinguishing Attacks on SOBER-t32. In *Proceedings of the 3rd NESSIE Workshop*, page 14, Munich,B,D, 2002.

[101] J. Daemen and G. Van Assche. Distinguishing stream ciphers with convolutional filters. Cryptology ePrint Archive, Report 2005/039, 2005. http://eprint.iacr.org/.

[102] S. Fluhrer, I. Mantin, and A. Shamir. Weaknesses in the key scheduling algorithm of RC4. *Lecture Notes in Computer Science*, 2259:1–24, 2001.

[103] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo. A statistical test suite for random and pseudorandom number generators for cryptographic applications. 2001. http://www.nist.gov.

[104] M. Matsumoto, H. Mariko, T. Nishimura, and M. Saito. Cryptographic Mersenne Twister and Fubuki Stream/Block Cipher. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/003, 2005. http://www.ecrypt.eu.org/stream.

[105] U. Kaiser. Hermes Stream Cipher. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/011, 2005. http://www.ecrypt.eu.org/stream.

[106] R. Vuckovac. MAG My Array Generator (A New Strategy for Random Number Generation). eSTREAM, ECRYPT Stream Cipher Project, Report 2005/014, 2005. http://www.ecrypt.eu.org/stream.

[107] E. Biham and J. Seberry. Py: A Fast Secure Stream Cipher using Rolling Arrays. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/023, 2005. http://www.ecrypt.eu.org/stream.

[108] J. Hong, D. H. Lee, Y. Yeom, D. Han, and S. Chee. T-function based stream cipher TSC-3. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/031, 2005. http://www.ecrypt.eu.org/stream.

[109] G. Gong and Y. Nawaz. The WG stream cipher. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/033, 2005. http://www.ecrypt.eu.org/stream.

[110] C. Gressel, R. Granot, and G. Vago. Zk-crypt - a compact stream cipher and more. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/035, 2005. http://www.ecrypt.eu.org/stream.

[111] G. Marsaglia. The Marsaglia random number CDROM including the DIEHARD battery of tests of randomness. 1996.

[112] W. Caelli, E. Dawson, L. Nielsen, and H. Gustafson. CRYPT–X statistical package manual, measuring the strength of stream and block ciphers, 1992.

[113] D. E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, 1981.

[114] P. L'Ecuyer and R. Simard. TestU01: A C library for empirical testing of random number generators. *ACM Transactions on Mathematical Software*, 2006. to appear.

[115] J. Soto. Randomness testing of the AES candidate algorithms, 1999.

[116] P. Hellekalek and S. Wegenkittl. Empirical evidence concerning AES. *ACM Trans. Model. Comput. Simul.*, 13(4):322–333, 2003.

[117] P.R. Kasselman. A statistical test for stream ciphers based on the maximum order complexity. In *South African Symposium On Communication and Signal Processing*, pages 213–218, 1998.

[118] U. M. Maurer. A universal statistical test for random bit generators. *J. Cryptol.*, 5(2):89–105, 1992.

[119] R. E. Greenwood. Coupon collector's test for random digits. *Math. Tables and other Aids to Computation*, 9:1–5, 224, 229, 1955.

[120] M.J.B. Robshaw. Stream ciphers. Technical Report TR - 701, July 1994.

[121] P. L'Ecuyer. Testing random number generators. In *Proceedings of the 1992 Winter Simulation Conference*, pages 305–313. IEEE Press, Dec 1992.

[122] J. L. Massey and S. Serconek. A Fourier Transform approach to the linear complexity of nonlinearly filtered sequences. In *CRYPTO '94: Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology*, pages 332–340, London, UK, 1994. Springer-Verlag.

[123] N. Courtois and J. Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In *ASIACRYPT*, pages 267–287, 2002.

[124] N. Courtois, A. Klimov, J. Patarin, and A. Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *EUROCRYPT*, pages 392–407, 2000.

[125] N. Courtois and W. Meier. Algebraic attacks on stream ciphers with linear feedback. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359. Springer, 2003.

[126] F. Armknecht and M. Krause. Algebraic attacks on combiners with memory. In Boneh [160], pages 162–175.

[127] N. Courtois. Algebraic attacks on combiners with memory and several outputs. In Choonsik Park and Seongtaek Chee, editors, *ICISC*, volume 3506 of *Lecture Notes in Computer Science*, pages 3–20. Springer, 2004.

[128] N. T. Courtois. Cryptanalysis of Sfinks. Cryptology ePrint Archive, Report 2005/243, 2005.

[129] H. Raddum. Cryptanalytic results on Trivium. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/039, 2006.

[130] F. K. Gürkaynak, P. Luethi, N. Bernold, R. Blattmann, V. Goode, M. Marghitola, H. Kaeslin, N. Felber, and W. Fichtner. Hardware Evaluation of eSTREAM Candidates: Achterbahn, Grain, Mickey, Mosquito, Sfinks, Trivium, Vest, ZK-crypt. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/015, 2006.

[131] M. Matsui. Linear cryptanalysis method for DES cipher. In *EUROCRYPT '93: Workshop on the theory and application of cryptographic techniques on Advances in cryptology*, pages 386–397, Secaucus, NJ, USA, 1994. Springer-Verlag New York, Inc.

[132] T. Siegenthaler. Decrypting a class of stream ciphers using ciphertext only. *IEEE Trans. Computers*, 34(1):81–85, 1985.

[133] W. Meier and O. Staffelbach. Fast correlation attacks on certain stream ciphers. *Journal of Cryptology*, 1(3):159–176, 1989.

[134] V. V. Chepyzhov, T. Johansson, and B. Smeets. A simple algorithm for fast correlation attacks on stream ciphers. In *Fast Software Encryption*, pages 181–195, London, UK, 2001. Springer-Verlag.

[135] J. Dj. Golić. Linear cryptanalysis of stream ciphers. In *Fast Software Encryption*, pages 154–169, 1994.

[136] F. Muller and T. Peyrin. Linear cryptanalysis of the TSC family of stream ciphers. In *ASIACRYPT*, pages 373–394, 2005.

[137] M. Hell and T. Johansson. On the Problem of Finding Linear Approximations and Cryptanalysis of Pomaranch Version 2. In *SAC*, 2006.

[138] Jr. B. S. Kaliski and M. J. B. Robshaw. Linear cryptanalysis using multiple approximations. In *CRYPTO*, pages 26–39, London, UK, 1994. Springer-Verlag.

[139] M.J. O. Saarinen. Chosen-IV Statistical Attacks on eSTREAM Stream Ciphers. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/013, 2006. http://www.ecrypt.eu.org/stream.

[140] S. O'Neil. Algebraic structure defectoscopy. Cryptology ePrint Archive, Report 2005/378, 2007.

[141] M. Hell, T. Johansson, A. Maximov, and W. Meier. A stream cipher proposal: Grain-128. ISIT, Seattle, USA, 2006.

[142] I. N. Kovalenko. Distribution of the linear rank of a random matrix. *Theory of Probability and Its Applications*, (17):342–346, 1972.

[143] M. E. Hellman. A cryptanalytic time-memory trade off. *IEEE Trans. Inform. Theory*, IT-26:401–406, 1980.

[144] H. Babbage. Improved exhaustive search attacks on stream ciphers. *European Convention on Security and Detection, IEE Conference publication*, (408):161–166, 1995.

[145] J. D. Golić. Cryptanalysis of alleged A5 stream cipher. In Walter Fumy, editor, *Advances in Cryptology - EuroCrypt '97*, pages 239–255, Berlin, 1997. Springer-Verlag. Lecture Notes in Computer Science Volume 1233.

[146] A. Biryukov and A. Shamir. Cryptanalytic time/memory/data tradeoffs for stream ciphers. In Okamoto [161], pages 1–13.

[147] J. Hong and P. Sarkar. Rediscovery of time memory tradeoffs. Cryptology ePrint Archive, Report 2005/090, 2005. http://eprint.iacr.org/.

[148] J. Quisquater, F. Standaert, G. Rouvroy, J. David, and J. Legat. A cryptanalytic time-memory tradeoff: First FPGA implementation. In *FPL '02: Proceedings of the Reconfigurable Computing Is Going Mainstream, 12th International Conference on Field-Programmable Logic and Applications*, pages 780–789, London, UK, 2002. Springer-Verlag.

[149] P. Oechslin. Making a Faster Cryptanalytic Time-Memory Trade-off. In Boneh [160], pages 617–630.

[150] M. O. Saarinen. A time-memory tradeoff attack against LILI-128. In *FSE '02: Revised Papers from the 9th International Workshop on Fast Software Encryption*, pages 231–236, London, UK, 2002. Springer-Verlag.

[151] A. Biryukov, A. Shamir, and D. Wagner. Real time cryptanalysis of A5/1 on a PC. In *FSE*, pages 1–18, 2000.

[152] Ö. Küçük. Slide resynchronization attack on the initialization of Grain 1.0. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/044, 2006.

[153] T. Helleseth, C. J.A. Jansen, and A. Kholosha. Pomaranch - design and analysis of a family of stream ciphers. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/008, 2006.

[154] M. J. Mihaljevic and J. Dj. Golić. A fast iterative algorithm for a shift register initial state reconstruction given the nosiy output sequence. In Jennifer Seberry and Josef Pieprzyk, editors, *AUSCRYPT*, volume 453 of *Lecture Notes in Computer Science*, pages 165–175. Springer, 1990.

[155] F. Jönsson and T. Johansson. A fast correlation attack on LILI-128. *Inf. Process. Lett.*, 81(3):127–132, 2002.

[156] T. Moreau. The Frogbit cipher, A Data Integrity Algorithm. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/009, 2005. http://www.ecrypt.eu.org/stream.

[157] M. O. Saarinen. d-monomial Tests are Effective against Stream Ciphers. SASC 2006-The State of the Art of Stream Ciphers http://www.ecrypt.eu.org/stream 2006,.

[158] R. Cramer, editor. *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*. Springer, 2005.

[159] Mitsuru Matsui and Robert J. Zuccherato, editors. *Selected Areas in Cryptography, 10th Annual International Workshop, SAC 2003, Ottawa, Canada, August 14-15, 2003, Revised Papers*, volume 3006 of *Lecture Notes in Computer Science*. Springer, 2004.

[160] D. Boneh, editor. *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*. Springer, 2003.

[161] T. Okamoto, editor. *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings*, volume 1976 of *Lecture Notes in Computer Science*. Springer, 2000.

[162] G. Boero, J. Smith, and K. F. Wallis. The sensitivity of chi-squared goodness-of-fit tests to the partitioning of data. *Econometric Reviews*, 23(4):341–370, January 2005. available at http://ideas.repec.org/a/taf/emetrv/v23y2005i4p341-370.html.

[163] J. Daemen, R. Govaerts, and J. Vandewalle. Resynchronization weaknesses in synchronous stream ciphers. In *EUROCRYPT '93: Workshop on the theory and*

application of cryptographic techniques on Advances in cryptology, pages 159–167, Secaucus, NJ, USA, 1994. Springer-Verlag New York, Inc.

[164] F. Armknecht, J. Lano, and B. Preneel. Extending the resynchronization attack. In *Selected Areas in Cryptography*, pages 19–38, 2004.

[165] S. Kiyomoto, T. Tanaka, and K. Sakurai. Experimental Analysis of Guess-and-Determine Attacks on Clock-Controlled Stream Ciphers. *IEICE Transactions*, 88-A(10):2778–2791, 2005.

[166] J. Mattsson. A Guess and Determine Attack on The Stream Cipher Polar Bear. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/017, 2006. http://www.ecrypt.eu.org/stream.

[167] P. Hawkes and G. G. Rose. Guess-and-Determine Attacks on SNOW. In *Selected Areas in Cryptography*, pages 37–46, 2002.

[168] D. Bleichenbacher and S. Patel. Sober crytanalysis. In Lars R. Knudsen, editor, *Fast Software Encryption*, volume 1636 of *Lecture Notes in Computer Science*, pages 305–316. Springer, 1999.

[169] P. Hawkes and G. G. Rose. Exploiting multiples of the connection polynomial in word-oriented stream ciphers. In Okamoto [161], pages 303–316.

[170] Y. Zhou and D. Feng. Side-Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing. Cryptology ePrint Archive, Report 2005/388, 2005.

[171] C. Rechberger and E. Oswald. Stream Ciphers and Side-Channel Analysis. SASC 2004-The State of the Art of Stream Ciphers, 2004.

Appendix A

Basics of Statistical Inference

A.1 Probability Theory

Probability theory is based on the paradigm of a random experiment. *Sample space* of an experiment is the set of all possible outcomes. An *event* is a subset of a sample space. If the observed outcome of an experiment is an element of the set E, then the event E is said to occur.

Let Ω be a sample space and P be a function that associates a number for each event. Then, P is called a *probability measure* provided that (i) for any event E, $0 \leq P(E) \leq 1$, (ii) $P(\Omega) = 1$, (iii) for any sequence $E_1, E_2, \ldots$ of disjoint events,

$$P(\cup_{i=1}^{\infty} E_i) = \sum_{i=1}^{\infty} P(E_i). \tag{A.1.1}$$

A *random variable* is a function defined on the points of the sample space. Random variables that take only distinct values on a scale are called *discrete random variable*, whereas random variables having values on a continuum are called *continuous random variables*.

The *probability distribution* of a discrete random variable X associates a probability $f(x) = Pr(X = x)$ with each distinct outcome x. For continuous random variables, the distribution corresponds to the mathematical function for which the area under the curve corresponding to any interval is equal to the probability that X with take on a value in the interval, that is

$$P(a \leq X \leq b) = \int_a^b f(x)dx \tag{A.1.2}$$

for any real constants a and b with $a \leq b$.

The *cumulative distribution function* F of X is defined as $F(a) = P(X \leq a)$ for $-\infty < a < \infty$. For discrete random variables, it corresponds to the summation $F(a) = \sum_{x \leq a} f(x)$, and for continuous random variables, it corresponds to the integral $F(a) = \int_{-\infty}^{a} f(x)dx$.

Let X be a continuous random variable with probability distribution $f(x)$, and $u(x)$ be a function of X such that $\int_{-\infty}^{\infty} u(x)f(x)dx$ ($\sum_x u(x)f(x)$ for discrete random variables) exists, the integral (or the sum) is called the *mathematical expectation* of $u(X)$ and denoted by $E[u(x)]$. The mean value μ of a random variable X is defined as $\mu = E[X]$, if exists. The variance of X, denoted by σ^2, is equal to $E[(X - \mu)^2]$.

Linear relation between two random variables X and Y can be measured by two measures *covariance* and *correlation of correlation*. The *covariance* of two random variables $cov(X, Y)$ is defined as

$$cov(X, Y) = E[(X - \mu_x)(Y - \mu_y)] = E[XY] - E(X)E[Y] \tag{A.1.3}$$

The correlation of coefficient $\rho_{X,Y}$ between two random variables X and Y is

$$\rho_{X,Y} = \frac{cov(X, Y)}{\sigma_x \sigma_y} \tag{A.1.4}$$

where σ_x and σ_y are the standard deviations of X and Y, respectively. The correlation coefficient $\rho\{X, Y\}$ takes values between -1 and 1. The coefficient 1 is obtained when $Y = a + bX$ with a positive b value, similarly the coefficient -1 is obtained when $Y = a + bX$ with a negative b value. As the coefficient gets close to -1 or 1, the correlation between variables gets stronger. A coefficient of correlation of 0 indicate that there is no linear relation between X and Y.

The discrete random variables $X_1, \ldots, X_n$ are *independent* if

$$P[X_1 = a_1, \ldots, X_n = a_n] = P(X_1 = a_1) \cdots P(X_n = a_n) \tag{A.1.5}$$

for all $a_i \in E$.

A.1.1 Some Special Distributions

The probability distribution of *Discrete Uniform Distribution* is

$$p(x) = \begin{cases} \frac{1}{k}, & \text{if } x = 1, 2, \ldots, k \\ 0, & \text{otherwise} \end{cases}$$

with mean $\frac{(k+1)}{2}$ and variance $\frac{(k^2-1)}{12}$.

The probability distribution of *Bernoulli Distribution* is

$$p(x) = \begin{cases} p & \text{if } x = 1 \\ 1-p & \text{if } x = 0 \\ 0 & \text{otherwise} \end{cases}$$

where p is the probability of success. The mean and the variance of the Bernoulli distribution is $\mu = p$ and $\sigma^2 = p(1-p)$, respectively.

The sum of n independent and identically distributed (i.i.d.) Bernoulli random variables is called the *binomial random variable*. The distribution of Binomial distribution is

$$p(x) = \binom{n}{x} p^x (1-p)^{n-x} \tag{A.1.6}$$

with mean $\mu = np$ and $\sigma^2 = np(1-p)$.

The probability distribution of *continuous uniform distribution* is $f(x) = \frac{1}{(b-a)}$ where $a \leq x \leq b$. The mean and variance of a continuous uniform probability distribution are $\mu = \frac{(b+a)}{2}$ and $\sigma^2 = \frac{(b-a)^2}{12}$.

The *Normal distribution* (also called Gaussian distribution) is widely used and has great importance in many fields. The distribution is determined by two parameters, mean μ and variance σ^2 and represented by $N(\mu, \sigma^2)$. The probability distribution with mean μ and standard deviation σ is given as

$$f(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{A.1.7}$$

where $-\infty < x, \mu < \infty$ and $\sigma > 0$. The cumulative distribution function of normal distribution is

$$\begin{aligned} F(x; \mu, \sigma) = Pr(X < x) &= \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{x} e^{-\frac{(u-\mu)^2}{2\sigma^2}} du \\ &= \Phi(\frac{x-\mu}{\sigma}) \end{aligned}$$

where Φ is the cumulative distribution function of the *standard normal distribution* which is a special case of Normal distribution with $\mu = 0$ and $\sigma = 1$. If the random variable X is normally distributed with mean μ and variance σ^2, $\sigma^2 > 0$, then the random variable $Y = \frac{(X-\mu)}{\sigma}$ is distributed standard normally.

For a random variable $X \sim N(0, 1)$, the probability that X is between $-a$ and a

where $a > 0$ is

$$Pr(-a < X < a) = \Phi(a) - \Phi(-a) = 2\Phi(a) - 1 \tag{A.1.8}$$

for any μ and σ.

Importance of the Normal distribution lies on the following *Central Limit Theorem* which is the foundation of many statistical procedures.

Theorem A.1.1. *Let $X_1, \ldots, X_n$ is a random sample from a distribution with mean μ and variance $\sigma^2 > 0$. Then, the random variable $Y = \frac{(\sum_{i=1}^{n} X_i - n\mu)}{\sqrt{n}\sigma}$ has a standard normal limiting distribution.*

For large n, Binomial approximation can be approximated by the Normal distribution, given that the distribution is not highly skewed. As a rule of thumb, np and $n(1-p)$ should be greater than 5.

The probability distribution of the *Gamma distribution* is

$$f(x|\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x} \tag{A.1.9}$$

for $x \geq 0$ where α and β are input parameters and $\Gamma[\alpha] = \int_0^\infty x^{\alpha-1} e^{-x} dx$. Gamma distributions with parameters α and β are denoted as $\Gamma(\alpha, \beta)$.

χ_n^2 is the distribution of $X_1^2 + \cdots + X_n^2$, where $X_1, \ldots, X_n$ are i.i.d standard normal. Gamma distribution with parameters $\frac{n}{2}$ and $\frac{1}{2}$, $\Gamma(\frac{n}{2}, \frac{1}{2})$, coincides with the χ_n^2 distribution.

A.2 Hypothesis Testing

In most of the statistical inferences, hypothesis testing is employed in which the null hypothesis H_0 is tested against the alternative hypothesis H_1 (or H_a). A hypothesis is a statement or a conjecture about the distribution of one or more random variables. The null hypothesis claims a theory that is believed to be true or is used as a basis for argument, but has not been proved. In randomness testing, an example null hypothesis is *The data is generated by a truly random generator.*

A hypothesis test is defined by a test statistic T whose distribution under H_0 is either well approximated or known. The aim is to find an empirical evidence against H_0, if the evidence is found in a reasonable time, then H_0 is rejected. There are two possible decisions *reject* H_0 or *do not reject* H_0. Concluding *do not reject* H_0, does not mean that H_0 is true, the only suggestion is that there is not sufficient evidence against H_0 in favor of the alternative hypothesis.

While testing the hypothesis $\theta \in \delta$, the sample is partitioned into subsets $\{S_0, S_1\}$ and if $T \in S_0$ then the null hypothesis can not be rejected and otherwise the alternative hypothesis is accepted. Here, S_0 is known as the *acceptance region* and S_1 as the *critical region* of the test. There are two similar decision ways; to check the test statistic T and if T is in S_0 conclude H_0, otherwise conclude H_1. The second way to decide is to use a probability value that corresponds to the probability of observing a test statistic as extreme or more extreme than the observed test statistic given that the null hypothesis H_0 is true. This probability value is called p-value. Obtaining small p-values is considered as evidence against the null hypothesis. If the output p-value is less than the significance level α, then H_0 is rejected, otherwise H_1 is accepted.

The probability of rejecting null hypothesis H_0 when H_0 is in fact true is called the *significance level* or *size of the test*, α and $1 - \alpha$ is called the *confidence level*.

$$P(Type\ I\ error) = P(T \in S_1 | \theta \in \delta) = \alpha \tag{A.2.1}$$

Another type of error, *Type II error* or β is done if H_0 is accepted when in fact H_1 is true.

$$P(Type\ II\ error) = P(T \in S_0 | \theta \notin \delta) = \beta \tag{A.2.2}$$

The *power of test*, $1 - \beta$, is defined to be the probability that type II error is not made, equivalently it is the ability of a test to detect a defect, if it exists. As expected, these two errors probabilities, α and β are inversely related.

The procedure of hypothesis testing is given in the following algorithm.

Algorithm A.2.1: HYPOTHESIS TESTING$(-)$

Formulate H_0 and H_1 ;
Choose level of significance, α;
Collect a sample;
Calculate test statistic, T;
$p - value = Pr(x > T | H_0)$;
if $p - value < \alpha$
 decision = Reject H_0;
 else decision = Not reject H_0;
return $(decision)$

Goodness of Fit Tests

One classical problem in statistics is the degree of correspondence between observed values and the expected values based on a hypothesized distribution. There are several nonparametric approaches to test goodness of fit.

Pearson's Chi Square (χ^2) Test involves grouping data into classes and comparing observed outcomes to the expected figures under the null distribution. The tests is applied to both continuous and discrete data, however the data should be categorized into groups. The structure of the test is as follows; firstly the data is categorized into k mutually exclusive groups and the test statistic is defined as

$$\chi^2 = \sum_{i=1}^{k} (o_i - e_i)^2 / e_i \tag{A.2.3}$$

where o_i and e_i is the observed and expected frequency for group i, respectively. If H_0 is true, then χ^2 is distributed according to $\chi^2(k-1-p)$ where k is the number of groups and p is the number of parameters estimated from the data.

For the χ^2 approximation to be valid, the expected frequency of each group should be at least 5. Therefore, enough sample size should be available. One assumption of the model is that the sample is independent and identically distributed.

The test statistic highly depend on the the categorization of data. In [162], some points are emphasized about the usage of χ^2 tests

- The number of classes should be in the range 8-12 to maximize the power of χ^2.
- The choice of non-equiprobable classes can increase the power of the χ^2 test significantly.

Kolmogorov Smirnov Test is another approach to test goodness of fit and it utilizes the empirical cumulative distribution function.

Definition A.2.1. Let $x_1, x_2, \ldots, x_n$ be a random sample. The *empirical distribution* $S(x)$ is a function of x, that equals to the proportion of x_i's less than or equal to x, for each x, that is,

$$S(x) = \frac{n(i)}{N} \tag{A.2.4}$$

where $n(i)$ is the number of points less than x_i

The Kolmogorov-Smirnov test statistic is the maximum distance between the em-

pirical and the hypothesized distribution $F^*(x)$, that is

$$T = max_x|S(x) - F^*(x)|. \tag{A.2.5}$$

The test statistics does not depend on the underlying distribution, however it can only be applied to continuous distributions. The population mean and variance should be known to apply this test.

Appendix B

Other Attacks Against Stream Ciphers

The details of the distinguishing, correlation, algebraic and TMTO attacks against stream ciphers are presented in the thesis. Basics of resynchronization, guess and determine and side channel attacks are presented in this part of the study.

B.1 Resynchronization Attacks

The transmission of ciphertext is assumed to be done using a noisy channel. Due to the noise, some modifications such as flip/flop or insertion/deletion errors may occur. If the value of a ciphertext bit is changed, decryption of the corresponding bit is done incorrectly and remaining bits are not affected which means that the flip/flop errors in synchronous stream ciphers are not propagated. However, if a ciphertext bit is deleted or inserted during transmission, synchronization between sender and receiver is lost. If sender and receiver are not synchronized in other words do not have exact same internal state variables, it is not possible to decrypt the ciphertext correctly until synchronization is reestablished.

Different resynchronization mechanisms can be used depending on the selected protocol. The first method is the *fixed resynch* in which the message is divided into frames of equal length and each frame is encrypted using a different IV. This approach is commonly used in wireless communication applications such as Bluetooth and GSM systems. The frame length is selected depending on the probability of error, in GSM applications, the frame length is selected as 228 bits. Second method can be given as *requested resync* in which resynchronization is done only when it is lost and detected by the receiver. As another approach, some special markers may be placed in predefined

intervals to avoid synchronization lost.

The first resynchronization attack is described by Daemen *et al.* [163]. According to the attack scenario, the attacker has access to R frames $(z_0^i, z_1^i, \ldots, z_T^i)$, $i = 0, 2, \ldots, R-1$ and their corresponding IVs. This attack is applicable to ciphers having following structure;

$$\begin{aligned} S_0^i &= A.K + B.IV^i, \\ z_t^i &= f(\Pi S_t^i), \\ S_{t+1}^i &= L.S_t^i \end{aligned}$$

where $A \in Z_2^{n\times k}$, $B \in Z_2^{n\times v}$, $L \in Z_2^{n\times n}$ and $\Pi \in Z_2^{\varphi\times n}$ are known. Since A and B are linear, this attack only works for stream ciphers with linear resynchronizations. According to the matrix Π, only φ of n bits are used in keystream generation. To obtain the secret key K, the following equations are utilized;

$$K_t = \Pi \cdot L^t \cdot A \cdot K, \tag{B.1.2a}$$

$$IV_t^i = \Pi \cdot L^t \cdot A \cdot IV^i. \tag{B.1.2b}$$

The system of equations may be given as

$$z_i^t = f(K_t \oplus IV_t)\ , \tag{B.1.3a}$$

for $0 \leq i \leq R-1$ and $0 \leq t \leq T-1$. For small values of φ, exhaustive key search may be done on K_t for each t using R equations. When R is larger than φ, unique solution of K_t can be found. This procedure may be repeated until the entire key is found. The complexity of the attack is $\lceil k/\varphi \rceil \cdot 2^\varphi$ evaluations of f, at least φ resynchronizations, and k bits from the kesytream in total. The main drawback of the attack is that it requires the cipher to have a linear initialization. Also, for large φ, the complexity of the attack is very high. In [164], the idea of resynchronization attacks is extended and the attack is combined with algebraic attacks and linear cryptanalysis.

B.2 Guess and Determine Attacks

In Guess and Determine attacks, as the name implies, the general approach is to guess some of the secret variables and then to determine the value of other variables based on the observed keystream. To remove nonlinearity of the system, some assumptions are made and new linear recurrences that reduces the number of variables to be guessed are obtained.

The procedure can be summarized as follows;

- Guessing some parts of key or the internal state bits,
- Determining the rest of the unknown bits, using an assumption,
- Comparing generated keystream with the available keystream, to check whether the guess is correct. If the guess is correct, it can be confirmed by using the output keystream. If the guess is not correct, a new guess is made and the whole procedure is repeated.

For a stream cipher with k bit key, the attack is successful if $2^g.(1/p).w \leq 2^k$ where g is the number of guessed variables, p is the probability that the assumption holds and w is the work done to check whether guessed bits are correct when the assumption holds. The expected required number of trials is given by $1/p$. A problem regarding the probability of the assumptions is pointed out by [165] and it is claimed that the internal states that permit a guess and determine attack are non-uniformly distributed in the period of the cipher. Thus it is claimed that the real average of complexity of a guess and determine attack is approximately twice the theoretical complexity.

Some examples of the attack may be listed as follows. A guess and determine attack is presented in [166] against Polar Bear that requires the internal state with complexity of $O(2^{79})$ and 24 bytes of keystream. In [167], two attacks are presented against SNOW, the first of them has data complexity $O(2^{64})$ and process complexity $O(2^{256})$ and the second has process complexity of $O(2^{224})$ and a data complexity of $O(2^{95})$. Another attack is against SOBER by Bleichenbacher and Patel [168].

Using irregular clocking increases resistance to guess and determine attacks, but the ciphers excluding the irregular clocking part should be designed secure against guess and determine attacks. Also, in [169], it is shown that guess and determine attacks are more effective in word-oriented stream ciphers.

B.3 Side Channel Attacks

Side channel attacks utilize implementation-specific characteristics such as time delays, power consumptions or electromagnetic radiation. Since these attacks are implementation specific, physical implementation of the cipher is very critical, even tiny changes may result in big differences in security. Timing attacks and power analysis are important types of side channel attacks. In timing attacks, the time taken to execute various steps in algorithms is analyzed. In power analysis, the attacker uses the varying

power consumption of a cryptographic hardware device during computation and tries to find information about the state of the device. Other types of side channel attacks; fault, electromagnetic, acoustic, visible light, error message, cache based, frequency based, scan based attacks are summarized in [170].

To avoid side channel attacks in stream ciphers, apart from implementation issues, in [171], it is advised not to use building blocks such as stuttering phase in Sober-t32 and repeated manipulations of same bytes as in key schedule of RC4.

Appendix C

NIST Test Results

We test the Phase I candidates of eSTREAM using the test suite of NIST [103]. First, we generated 100 keystreams each having length 2^{20}, by randomly chosen key and IV pairs then, tested the outputs of all candidates using 15 tests with variable parameters. We obtained 100×188 many p-values for each cipher. The results that indicate weaknesses are summarized in Table C.1. Significant deviations are observed for Decim and Frogbit.

Table C.1: The result of NIST tests that indicate weaknesses. There are total of 148 nonperiodic template test results for each cipher, and Decim fails 11 of them.

Cipher	*Test*	*p-value*	*proportion*
Decim	Block Frequency	0.000000	0.0000
	Runs	0.000000	0.0200
	Longest Run	0.000000	0.9300
	Nonperiodic Templates	0.000000	0.7200
	Overlapping Templates	0.000000	0.7500
	Approximate Entropy	0.000000	0.8900
Frogbit	Frequency	0.000000	1.0000
	Block Frequency	0.000000	0.9200
	Cumulative Sum	0.000000	1.0000

Appendix D

F_1 for 2-round Trivium

$z_1 = 1 + s_0(3) + s_0(6) + s_0(15) + s_0(21) + s_0(27) + s_0(30) + s_0(39) + s_0(54) + s_0(57) +$
$s_0(67) + s_0(68) + s_0(69) + s_0(72) + s_0(96) + s_0(99) + s_0(114) + s_0(117) + s_0(123) +$
$s_0(126) + s_0(132) + s_0(138) + s_0(144) + s_0(165) + s_0(171) + s_0(4).s_0(5) + s_0(13).s_0(14) +$
$s_0(13).s_0(41) + s_0(13).s_0(119) + s_0(14).s_0(40) + s_0(14).s_0(118) + s_0(16).s_0(17) + s_0(19).s_0(20) +$
$s_0(19).s_0(47) + s_0(19).s_0(125) + s_0(20).s_0(46) + s_0(20).s_0(124) + s_0(22).s_0(23) + s_0(25).s_0(26) +$
$s_0(28).s_0(39) + s_0(34).s_0(35) + s_0(37).s_0(38) + s_0(37).s_0(65) + s_0(37).s_0(143) + s_0(38).s_0(64) +$
$s_0(39).s_0(40) + s_0(38).s_0(142) + s_0(40).s_0(119) + s_0(41).s_0(118) + s_0(43).s_0(44) + s_0(45).s_0(46) +$
$s_0(46).s_0(125) + s_0(47).s_0(124) + s_0(49).s_0(50) + s_0(52).s_0(53) + s_0(58).s_0(59) + s_0(58).s_0(164) +$
$s_0(59).s_0(163) + s_0(61).s_0(62) + s_0(63).s_0(64) + s_0(64).s_0(65) + s_0(64).s_0(143) + s_0(64).s_0(170) +$
$s_0(65).s_0(169) + s_0(65).s_0(142) + s_0(67).s_0(68) + s_0(70).s_0(71) + s_0(76).s_0(77) + s_0(79).s_0(77) +$
$s_0(103).s_0(104) + s_0(106).s_0(107) + s_0(118).s_0(119) + s_0(124).s_0(125) + s_0(127).s_0(128) +$
$s_0(130).s_0(131) +$
$s_0(133).s_0(149) + s_0(134).s_0(148) + s_0(142).s_0(143) + s_0(147).s_0(148) +$
$s_0(151).s_0(152) + s_0(154).s_0(155) + s_0(160).s_0(161) + s_0(163).s_0(164) +$
$s_0(166).s_0(167) + s_0(13).s_0(39).s_0(40) + s_0(14).s_0(38).s_0(39) +$
$s_0(19).s_0(45).s_0(46) + s_0(20).s_0(44).s_0(45) + s_0(37).s_0(63).s_0(64) +$
$s_0(38).s_0(39).s_0(40) + s_0(38).s_0(39).s_0(41) + s_0(38).s_0(39).s_0(119) +$
$s_0(38).s_0(62).s_0(63) + s_0(39).s_0(40).s_0(118) + s_0(44).s_0(45).s_0(46) +$
$s_0(44).s_0(45).s_0(47) + s_0(44).s_0(45).s_0(125) + s_0(45).s_0(46).s_0(124) +$
$s_0(62).s_0(63).s_0(64) + s_0(62).s_0(63).s_0(65) + s_0(62).s_0(63).s_0(143) +$
$s_0(63).s_0(64).s_0(142) + s_0(133).s_0(147).s_0(148) + s_0(134).s_0(146).s_0(147)$
with bias 2^{-9}.

Appendix E

Linear Regression Model for d-monomial Test of Grain

Linear regression is a statistical tool to analyze the relationship between two variables; X and Y. One of the variables is considered to be explanatory and the other is considered to be dependent variable. Linear regression models the relation by fitting a linear equation to the observed data. The linear equation has the form $Y = a + bX$, where X is the explanatory variable and Y is the dependent variable. The association between X and Y are measured by the correlation coefficient that takes values between -1 and 1.

In this part, we model the relationship between the number of IVs and number of rounds using linear regression. We fit a linear equation to the observed data of d-monomial test of Grain as given in Figure E.1.

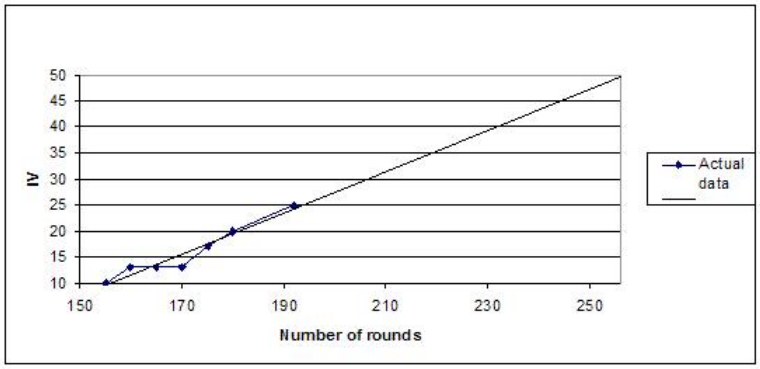


Figure E.1: The linear regression model for d-monomial test of Grain

The trend equation is obtained as $y = 0.3981092437x - 52.21953782$ where y represents the number of IVs and x represents the number of rounds in initialization. The correlation coefficient of the model is 0.96850. Using this model, the

prediction of required number of IVs to attack Grain with d-monomial test is $y = 0.3981092437(256) - 52.21953782 = 49.69643 \approx 50$.

VITA

Meltem Sönmez Turan was born in Samsun on May 27, 1977. She received her BSc degree in Department of Statistics from Middle East Technical University (METU), in June 1999 and her MS degree in Industrial Engineering Department in January 2003, specializing in genetic algorithms. During her BS studies, she also received a minor degree in Production Planning and Control from from Industrial Engineering Department, METU. She had worked as a teaching assistant in Informatics Institute, METU between 1999 and 2006.

In April 2008, she received her PhD degree in Cryptography Department of METU. Her research interests are analysis of symmetric cryptosystems, hash functions and genetic algorithms.

Meltem is married to Fehmi Fırat Turan.

List of Publications

- *Sönmez Turan M.*, Doğanaksoy A., Boztaş S., *On Independence and Sensitivity of Statistical Randomness Tests*, International Conference on Sequences and Their Applications (SETA), Lecture Notes of Computer Science, Springer-Verlag, 2008.
- *Sönmez Turan M.*, Çalık Ç., Buz Saran N., Doğanaksoy A., *New Distinguishers Based on Random Mappings Against Stream Ciphers*, International Conference on Sequences and Their Applications (SETA) Lecture Notes of Computer Science, Springer-Verlag, 2008.
- *Sönmez Turan M.*, Kara O. *Linear Approximations for 2-round Trivium*, SASC07 Stream Ciphers Revisited, Bochum, Germany 2007, also in Proc. First International Conference on Security of Information and Networks (SIN 2007), Gazimagusa, TRNC, May 2007, Trafford Publishing, 96-105, ISBN: 978-1-4251-4109-7.

- Englund H., Johansson T., *Sönmez Turan M.*, *A Framework for Chosen IV Statistical Analysis of Stream Ciphers*, Tools for Cryptanalysis, Poland, 2007 also in Progress in Cryptology - INDOCRYPT 2007, volume 4859 of Lecture Notes of Computer Science, 268-281, Springer-Verlag, 2007.

- *Sönmez Turan M.*, Doğanaksoy A., Çalık Ç., *Statistical Analysis of Synchronous Stream Ciphers*, SASC06 Stream Ciphers Revisited, Leuven, Belgium 2006

- Doğanaksoy A., Çalık Ç., Sulak F., *Sönmez Turan M.*, *New Randomness Tests Using Random Walk*, 2. Ulusal Kriptoloji Sempozyumu, Ankara, 2006.

- Saygı E., Saygı Z., *Sönmez Turan M.*, Doğanaksoy A., *Statistical approach on the number of functions satisfying SAC*, BFCA'05, Boolean Functions: Cryptography and Applications, Editors: J-F. Michon, P. Valarcher, J-B. Yuns, pp. 39-48, 7-9 Mart 2005, Rouen, France.

- Süral H., *Sönmez Turan M*, Özdemirel, Nur E., *Nearest Neighborhood and Greedy Crossovers in an Evolutionary Algorithm for Solving Traveling Salesman Problem*, MIC'05, The 6th Metaheuristics International Conference, Vienna, Austria., 2005

- Toz D., Doğanaksoy A., *Sönmez Turan M.*, *Statistical Analysis of Block Ciphers*, First National Cryptology Symposium, Ankara, Turkey 2005

- *Sönmez M.*, Özdemirel N.E., Süral H., *An Evolutionary Approach to Travelling Salesman Problem*, YA/EM'02, Operational Research and Industrial Engineering XXIII. National Congress, İstanbul, Turkey, 2002