St Clements University

Mat. No. 17694

Analysis of Shift Registers Stream Cipher Cryptosystems

A Dissertation
Submitted to St Clements University in a Partial Fulfillment of the Requirements for the degree of Doctor of Philosophy in Mathematics

By

Thair Sami Rasheed Al-Rubaie

Supervised By

Asist-prof. Dr. Hussin Niema Abbed Al-Hussieny

2012A.C Baghdad 1433A.H

Supervisor's Certification

I certify that this Dissertation entitled " ***Analysis of Shift Registers Stream Cipher Cryptosystems****" was prepared by "****Thair Sami Rasheed*** *" under my supervision at the Mathematics Science Department of the St Clements University in a partial fulfillment of the requirements for the Degree of Doctor in Mathematics Science.*

Signature :

Name : Asist-prof. Dr. Hussin Niema Abbed Al-Hussieny

Date : / / 2012

Linguistic Certification

*This is to certify that this thesis entitiled "**Analysis of Shift Registers Stream Cipher Cryptosystems"** by **"Thair Sami Rasheed"** was made under my linguistic supervision. Its language was amended to meet the English style.*

Signature:

Name:

Date: / / 2012

Examination Committee Certification

*We certify that we have read this thesis entitled "**Analysis of Shift Registers Stream Cipher Cryptosystems**" and as examining committee, examined the student "**Thair**" in its contents and what is related to it, and that in our opinion, it meets the standard of a thesis for the Degree of Master of Science in Computer Science.*

Signature :
Name :
Title :
Date : / /2012
(Chairman)

Signature:
Name :
Title :
Date : / /2012
(Member)

Signature :
Name :
Title :
Date : / /2012
(member)

Signature:
Name :
Title :
Date : / /2012
(Member)

Approved by .

Signature :
Name :
Title :
Date : / /2012

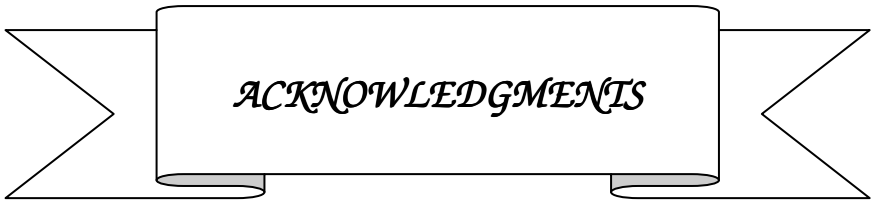
DEDICATION

TO Those Who Had Made Me What I'm ....
My Parents

The Cause of My Success ....

To My Brothers, Sisters and Every One Who Wishes the Best and Love.

Thair Sami Rasheed Al-Rubaie
December 2012



ACKNOWLEDGMENTS

Praise be to God Whenever I face any difficulty I used to pray to God to help me and He is always there protecting and saving me.

Then, I would like to express my warm thanks to

Asist-prof. Dr. Hussin Niema Abbed Al-Hussieny

for guidance, advice, encouragement, and constructive criticism throughout all stages of this work.

Thair 2012

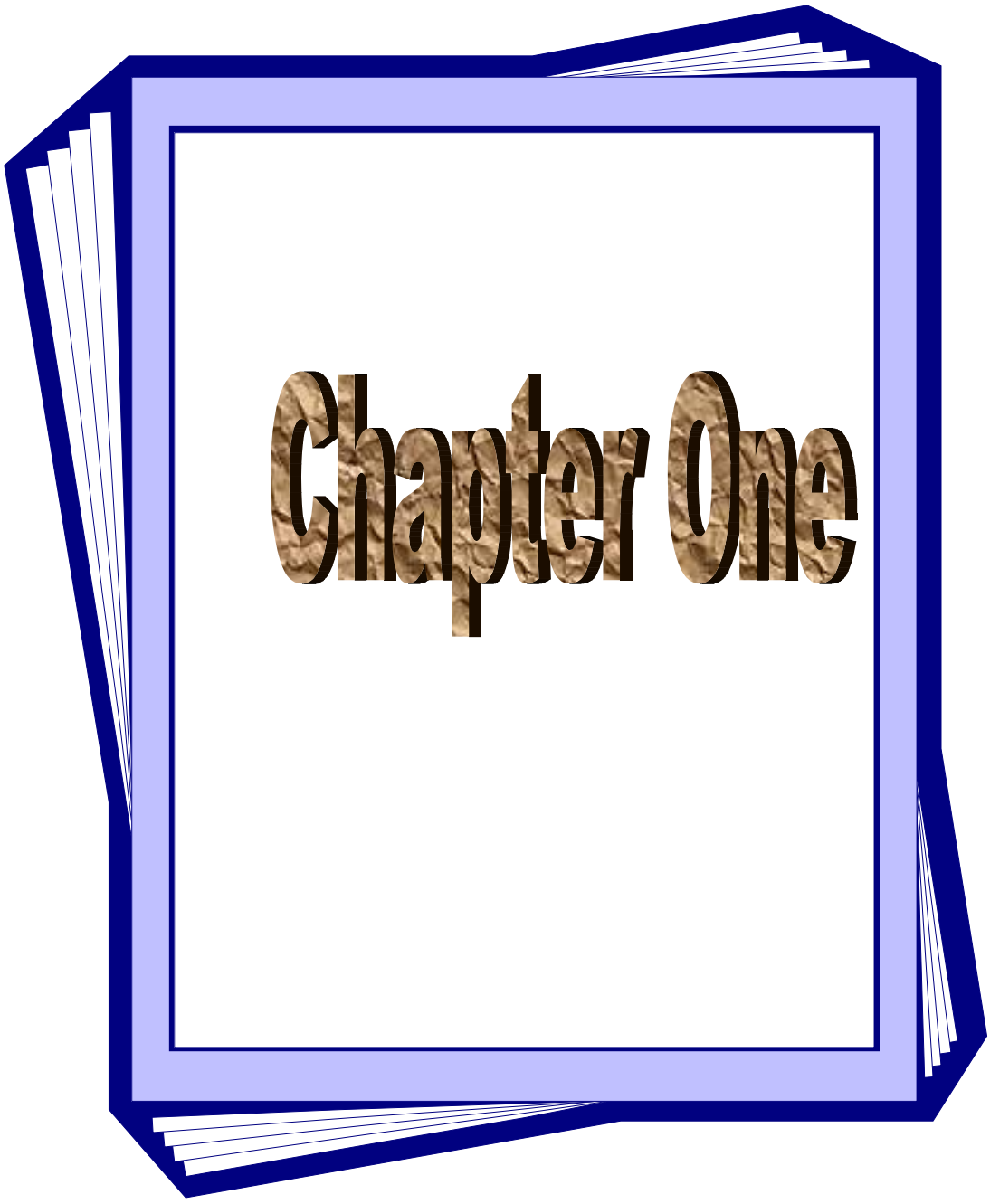
TABLE OF CONTENTS

Chapter Five Design & Analysis of New Stream Cipher Generators

Chapter Six Discussion of Thesis Results

List of Figures

List of Tables



Abstract

CHAPTER ONE

ABSTRACT

Stream Ciphers are an important class of encryption algorithms. They encrypt individual characters (usually binary digits) of a plaintext message one at a time, using an encryption transformation which varies with time. **Shift register sequences**, which are generated from **Linear Feedback Shift Register** (LFSR), are used in both cryptography and coding theory.

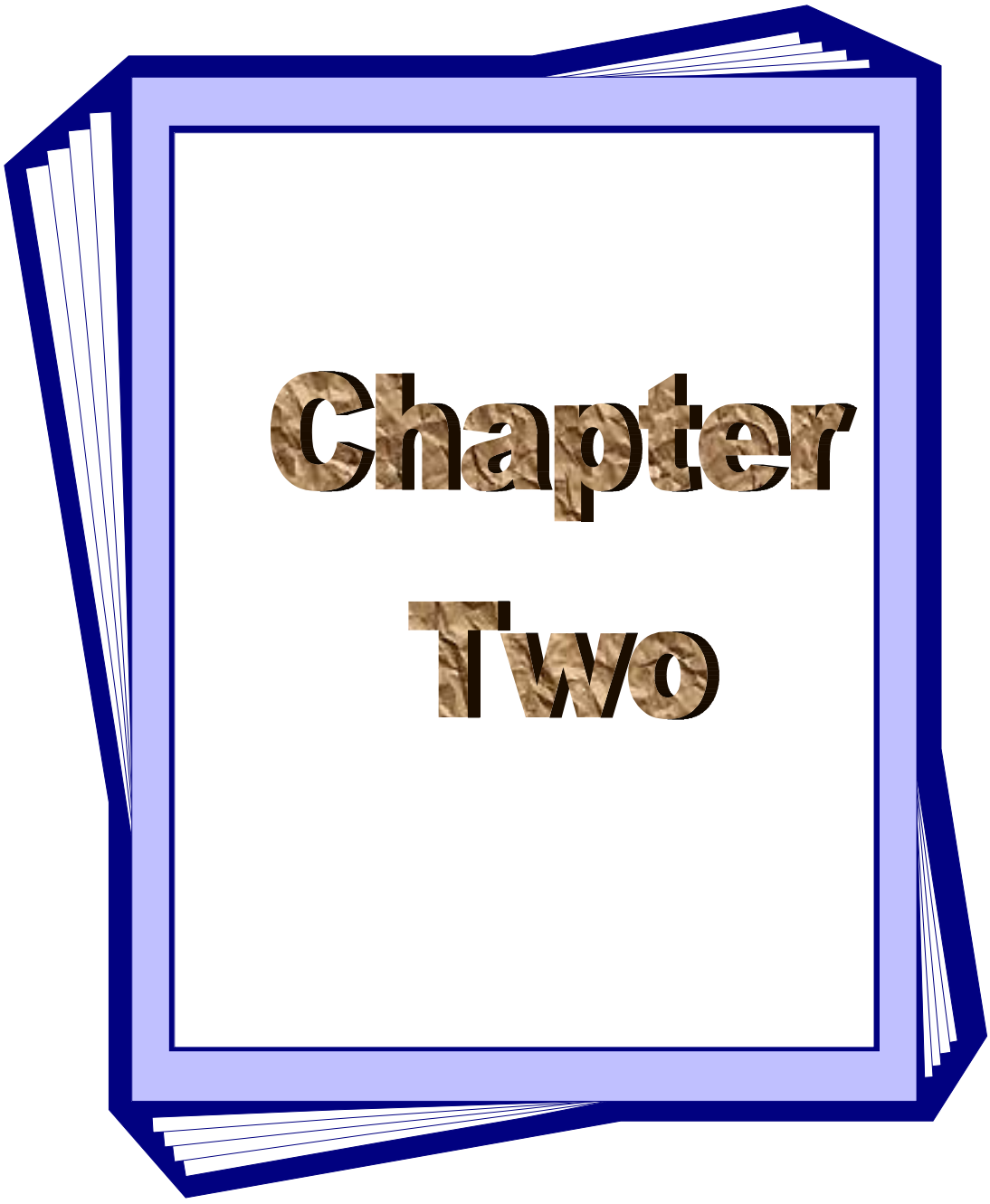
The **Periodicity**, **Linear Complexity**, **Randomness** and **Correlation Immunity** used as basic criterions to measure **Key Generator Efficiency**. The key generator depends basically on LFSR which is considered as one of the basic units, beside the combining function, of stream cipher systems.

This Dissertation introduces two important approaches; design and analysis of two new proposed key generators. These two cryptosystem are developed from two known generators. The two generators are called Modified Geffe and Threshold generators.

Through this Dissertation, the basic criteria can be calculated theoretically for the two proposed key generators before it be implemented or constructed (software or hardware). The two generators are passing these criteria successfully.

Lastly, the two proposed modified generator are analyzed by constructing, testing uniqueness then solving the system of linear equations system (SLE's) of the output sequence of each generator although the high complexity they own. The solving of SLE's means finding the initial values of the combined LFSR's in the generator.

The main conclusion made by this work that any generator necessarily passes the basic efficient criteria but it's not sufficient to be immune against the attacks.



Literature Survey

CHAPTER TWO

LITERATURE SURVEY

In 1989, Staffelbach and Meier [Sta.89] presented two new so-called fast correlation attacks which are more efficient than Siegenthaler's attack in the case where the component LFSRs have sparse feedback polynomials, or if they have low-weight polynomial multiples (e.g., each having fewer than 10 non-zero terms) of not too large a degree.

In 1990, Jansen and Boekee [Jan.90] defined the maximum order complexity of a sequence to be the length of the shortest (not necessarily linear) feedback shift register (FSR) that can generate the sequence. An efficient linear-time algorithm for computing this complexity measure was also presented.

Gustafson et al. in 1994 [Gus.94] describe a computer package which implements various statistical tests for assessing the strength of a pseudorandom bit generator.

In 1996, Gustafson [Gus.96-1] considered alternative statistics for the runs test and the autocorrelation test. Gustafson, Dawson, and Golić [Gus.96-2] proposed a new repetition test which measures the number of repetitions of l-bit blocks. The test requires a count of the number of patterns repeated, but does not require the frequency of each pattern.

The security of GSM conversation is based on usage of the A5 family of stream ciphers. Many hundred million customers in Europe are protected from over-the-air piracy by the stronger version in this family, the A5/1 stream cipher. Other customers on other markets use the weaker version, A5/2. The approximate design of A5/1 leaked in 1994, and in 1999 the

exact design of both A5/1 and A5/2 was discovered by Briceno [Bri.99]. A lot of investigations of the A5 stream ciphers followed.

In 1987, Ron Rivest from RSA Data Security, Inc. made a design of a byte oriented stream cipher called RC4 [Sma.03]. This cipher found its application in many Internet and security protocols. The design was kept secret up to 1994, when the alleged specification of RC4 was leaked for the first time [Sch.96]. Since that time many cryptanalysis attempts have been done on RC4 [Flu.00,Man.01,Pau.03].

At FSE 2004, a new stream cipher called VMPC [Zol.04] was proposed by Bartosz Zoltak, which appeared to be a modification of the RC4 stream cipher. In cryptanalysis, a linear distinguishing attack is one of the most common attacks on stream ciphers. In the paper [Pau.04] it was claimed that VMPC is designed especially to resist distinguishing attacks.

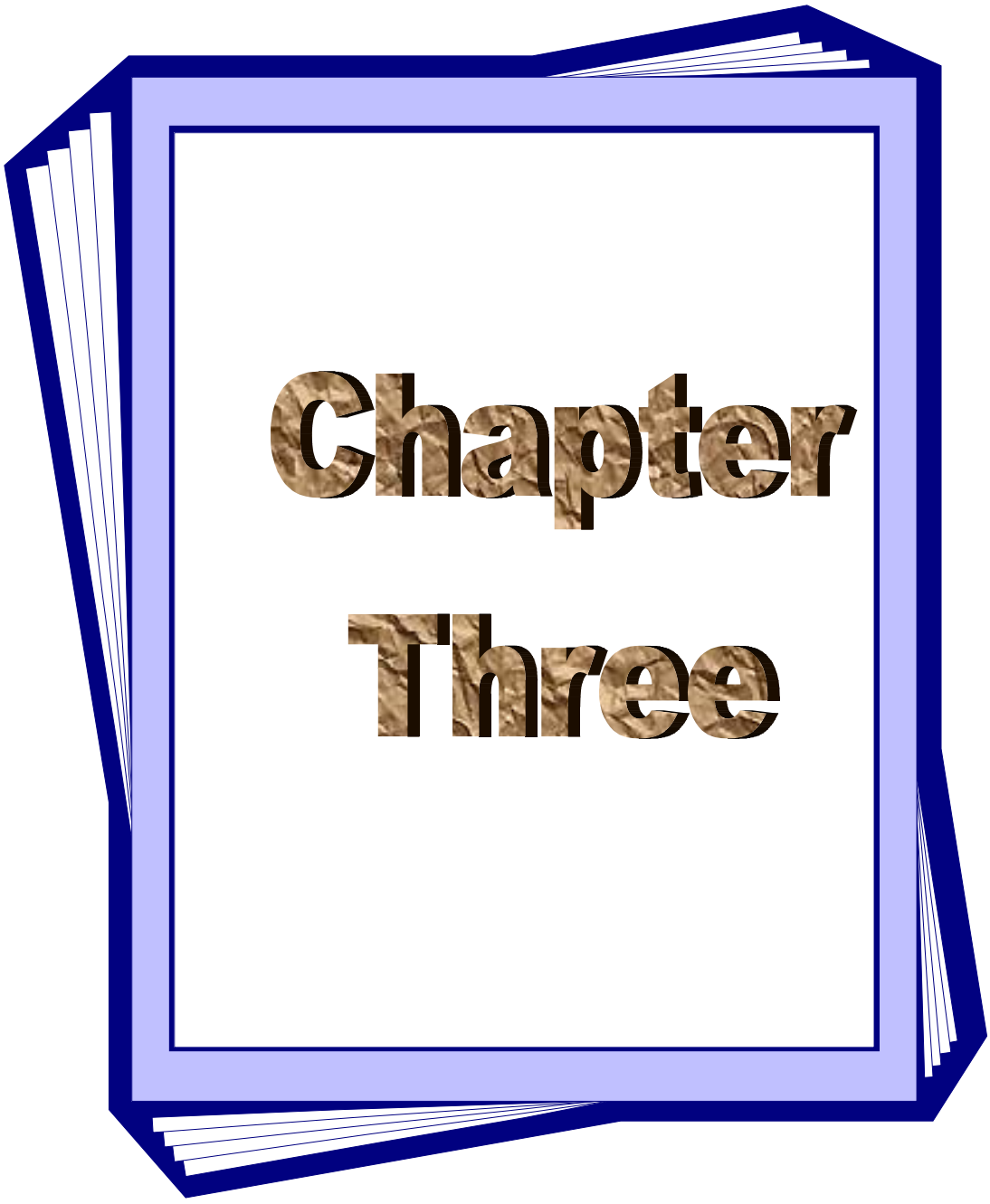
Scream was developed by the IBM researchers Coppersmith, Halevi, and Jutla in 2002 [Hal.02]. It is a purely software-oriented stream cipher. The design is based on the ideas behind the SEAL stream cipher [Rog.44], but considered to be more secure. The so-called “toy cipher” denoted Scream0 uses the AES S-box whereas the Scream stream cipher uses secret S-boxes, generated by the key.

Recently, a new European project eSTREAM [eST.05] has started, and at the first stage of the project 35 new proposals were received by May 2005. Although many previous stream ciphers were broken, collected cryptanalysis experience allowed strengthening new proposals significantly, and there are many of them that are strong against different kinds of attacks. One such good proposal was the new stream cipher Grain.

The stream cipher Grain was developed by a group of researchers M. Hell, T. Johansson, and W. Meier, and was especially designed for being

very small and fast in hardware implementation. It uses the key of length 80 bits and the IV is 64 bits, its internal state is of size 160 bits. Grain uses a nonlinear feedback shift register (NLFSR) and a linear feedback shift register (LFSR), and the idea to use NLFSR is quite new in modern cryptography [Joh.02].

Dragon is a word oriented stream cipher [Che.05] submitted to the eSTREAM project, designed by a group of researchers, Ed Dawson et al. It is a word oriented stream cipher that operates on key sizes of 128 and 256 bits. The original idea of the design is to use a nonlinear feedback shift register (NLFSR) and a linear part (counter), combined by a filter function to generate a new state of the NLFSR and produce the keystream. The internal state of the cipher is 1088 bits, which is updated by a nonlinear function denoted by F. This function is also used as a filter function producing the keystream. The idea to use a NLFSR is quite modern, and there are not many cryptanalysis techniques on NLFSRs yet developed.



Basic Concepts

CHAPTER THREE

BASIC CONCEPTS

3.1 Introduction

The growth of the Internet has made government intelligence and police agencies nervous. They say that widely available encryption software could make wiretapping more difficult; their common reaction is to try to restrict the strength of encryption algorithms or require that spare copies of the keys are available somewhere for them to seize.

Cryptography is the study of mathematical techniques which are related to the aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication [Sti.95].

This chapter introduces the basic concepts in different fields in mathematics, which the cryptography and cryptanalysis are needed, specially, in stream cipher systems, which will be discussed in details in chapter two.

3.2 Basic Mathematical Concepts

3.2.1 Number Theory

Number theory, in mathematics, is primarily the theory of the properties of integers (whole numbers) such as parity, **divisibility**, **primality**, **additivity**, and **multiplicativity**, etc. In the next subsections we will investigate more detailed discussions about numbers.

Theorem (3.1): (the fundamental theorem of arithmetic) [Apo.98]

Any positive integer n>1 can be written uniquely in the following prime factorization form:

$$n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k} = \prod_{i=1}^{k} p_i^{\alpha_i} \quad \ldots(3.1)$$

where $p_1<p_2<\ldots<p_k$ are primes, and $\alpha_1, \alpha_2, \ldots, \alpha_k$ are positive integers.

Example (3.1) [Yan.00**]**: The following are prime factorization of n for n=1999, 2000, …, 2010.

1999	=	1999	,	2000	=	$2^4.5^3$	,	2001	=	3.23.29
2002	=	2.7.11.13	,	2003	=	2003	,	2004	=	2^3.3.167
2005	=	5.401	,	2006	=	2.17.59	,	2007	=	3^2.223
2008	=	2^3.251	,	2009	=	7^2.41	,	2010	=	2.3.5.67

Definition (3.1)[And.94**]:** Let a and b be two integers, not both zero. The largest divisor d s.t. d|a and d|b is called the **greatest common divisor** (gcd) of a and b, which is denoted by gcd(a,b).

Definition (3.2)[And.94**]:** Let a and b be two integers, not both zero. d is a common multiple of a and b, the least common multiple (lcm) of a and b, is the **smallest common multiple**, which is denoted by lcm(a,b).

Definition (3.3)[And.94**]:** Integers a and b are called **relatively prime** if gcd(a,b)=1. we say that integers $n_1,n_2,\ldots n_k$ are relatively prime if, whenever $i\neq j$, we have $\gcd(n_i,n_j)=1$, $\forall i,j,\ 1\leq i,j\leq k$.

Theorem (3.2)[Apo.98**]**: Suppose a and b are two positive integers.

If $a=\prod_{i=1}^{k} p_i^{\alpha_i}$ and $b=\prod_{i=1}^{k} p_i^{\beta_i}$, then

$\gcd(a,b)=\prod_{i=1}^{k} p_i^{\varepsilon_i}$, where $\varepsilon_i=\min(\alpha_i,\beta_i)$, $\forall i$, $1\le i\le k$.

$\text{lcm}(a,b)=\prod_{i=1}^{k} p_i^{\delta_i}$, where $\delta_i=\max(\alpha_i,\beta_i)$, $\forall i$, $1\le i\le k$.

Theorem (3.3)[Apo.98]: Suppose a and b are two positive integers, then

$$\text{lcm}(a,b)=\frac{a.b}{\gcd(a,b)}.$$

Example (3.2): Since the prime factorization of 240 and 560 are:

$240=2^4.3.5$ and $560=2^4.5.7$, then the:

$\gcd(240,560)=2^{\min(4,4)}.3^{\min(1,0)}.5^{\min(1,1)}.7^{\min(0,1)}=2^4.3^0.5^1.7^0=80.$

$\text{lcm}(240,560)=2^{\max(4,4)}.3^{\max(1,0)}.5^{\max(1,1)}.7^{\max(0,1)}=2^4.3^1.5^1.7^1=1680.$

$$\text{lcm}(240,560)=\frac{240*560}{\gcd(240,560)}=\frac{134400}{80}=1680$$

3.2.2 Algebra Description of Logic Circuits [Riv.97]

In electronically logical circuits (which are subject to the Boolean algebra), there are small circuits called Gates which are, for example, part from transistors, diodes, capacitors, and etc, these gates are shown in figure (3.1):

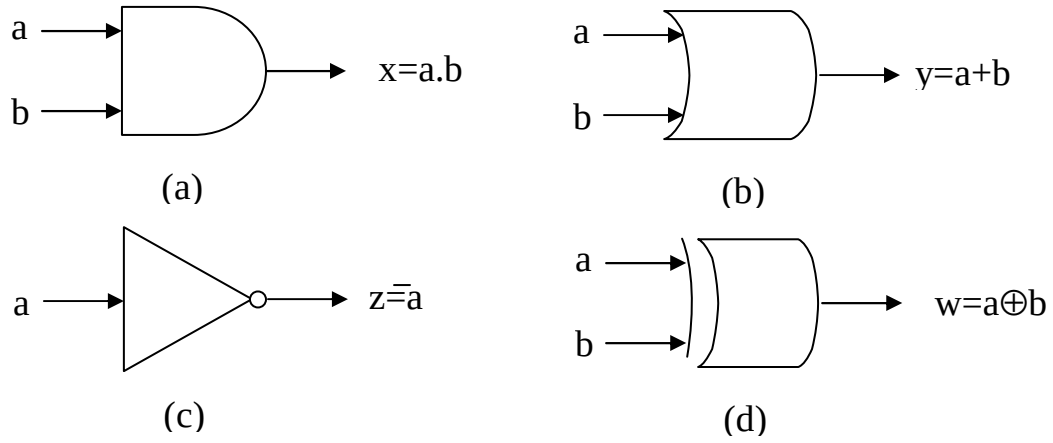


Figure (3.1) The Boolean gates.

(a).The gate AND: is multiplying the input variables.

(b).The gate OR: summing the input variables.

(c).The gate NOT: complement of the input variable.

(d).The gate XOR: summing XOR the input variables.

These gates are shown in the table (3.1).

Table (3.1) The truth tables of the four gates.

•	**0**	**1**
0	0	0
1	0	1

+	**0**	**1**
0	0	1
1	1	1

a	$\bar{\mathbf{a}}$
0	1
1	0

⊕	**0**	**1**
0	0	1
1	1	0

3.2.3 Polynomials [All.85]

Definition (3.4): P(x) is called polynomial from degree n:

$P(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$,

where x is the variable of the polynomial, $a_i \in \mathbb{F}, i=0,1,\ldots,n$ are called coefficients of the polynomial.

Definition (3.5): the polynomial $P(x) \in \mathbb{F}[x]$, where $\mathbb{F}[x]$ is the ring of all polynomial for the variable x, irreducible if P(x) = q(x)s(x), where q(x) or s(x) is of degree 0, otherwise P(x) is reducible.

Definition (3.6): If P(x) can be reformulate as follows:

$P(x) = (x-\alpha_1)(x-\alpha_2)\dots(x-\alpha_n)$,

Then $\alpha_i \in \mathbb{F}, i=1,\dots,n$ are called the roots of P(x).

Definition (3.7): the polynomial P(x) of degree r on filed GF(2) called primitive polynomial when $P(x)|(x^p+1)$, where $p=2^r-1$, and no less than p.

3.2.4 Sequences

Definition (3.8)[Gil.02]: The **sequence** in the field $\mathbb{F}$ is a function *f*, whose domain is the set of non negative (or could be positive) integer, s.t. $f:\mathbb{Z}\rightarrow\mathbb{F}$, and its denoted by $S=\{S_n\}_{n=0}^{+\infty}$.

Definition (3.9)[Gil.02]: The Sequence S is **periodic** when $\exists p \in \mathbb{Z}^+$ s.t. $s_0=s_p, s_1=s_{p+1},\dots$, the minimum p is the **period** of S.

3.3 Cryptology Terminology

Cryptography (from the Greek Kryptós, "**hidden**" and gráphein, "**to write**") is the study of principles and techniques by which information can be concealed in ciphertexts and later revealed by legitimates users employing the secret key, but in which it is either impossible or computationally infeasible for an unauthorized person to do so. **Cryptanalysis** (from the Greek Kryptós, and analyéin "**to loosen**") is the science (and art) of recovering information from ciphertexts without knowledge of the key. Both terms are subordinate to the more general term **Cryptology** (from the Greek Kryptós, and logos, "**word**"). The cryptography concerned in **Encryption** and **Decryption** processes [Mot.95].

Now we have to present some important notations:

- **Message space** M: a set of strings (plaintext messages) over some alphabet, that needs to be encrypted.
- **Ciphertext space** C: a set of strings (ciphertexts) over some alphabet that has been encrypted.
- **Key space** K: a set of strings (keys) over some alphabet, which includes the encryption key e_k and the decryption key d_k.
- The **Encryption** process (algorithm) E: $Ee_k(M)=C$.
- The **Decryption** process (algorithm) D: $Dd_k(C)=M$.

The algorithms E and D must have the property that:

$$Dd_k(C)=Dd_k(Ee_k(M))=M.$$

The above situations shown in figure (3.2).

Figure (3.2) Encryption Process.

3.4 Cryptosystems

Cryptanalysis is the science and study of methods of breaking ciphers. It is a system identification problem, and the goal of **Cryptography** is to build systems that are hard to identify [Kon.81]. To attack a cryptographic system successfully the cryptanalysis is forced to be based on subtle approaches, such as knowledge of at least part of the text encrypted, knowledge of characteristic features of the language used,..., with some luck. However, in practice, some of this information may be inaccurate, imprecise, or missing, which, in turn, causes to decrease the possibility of attacking and increasing the time or the resources required by the analyst. The **Cryptosystem** are the systems which use the encryption and decryption processes, these systems can be classified as in figure (3.3).

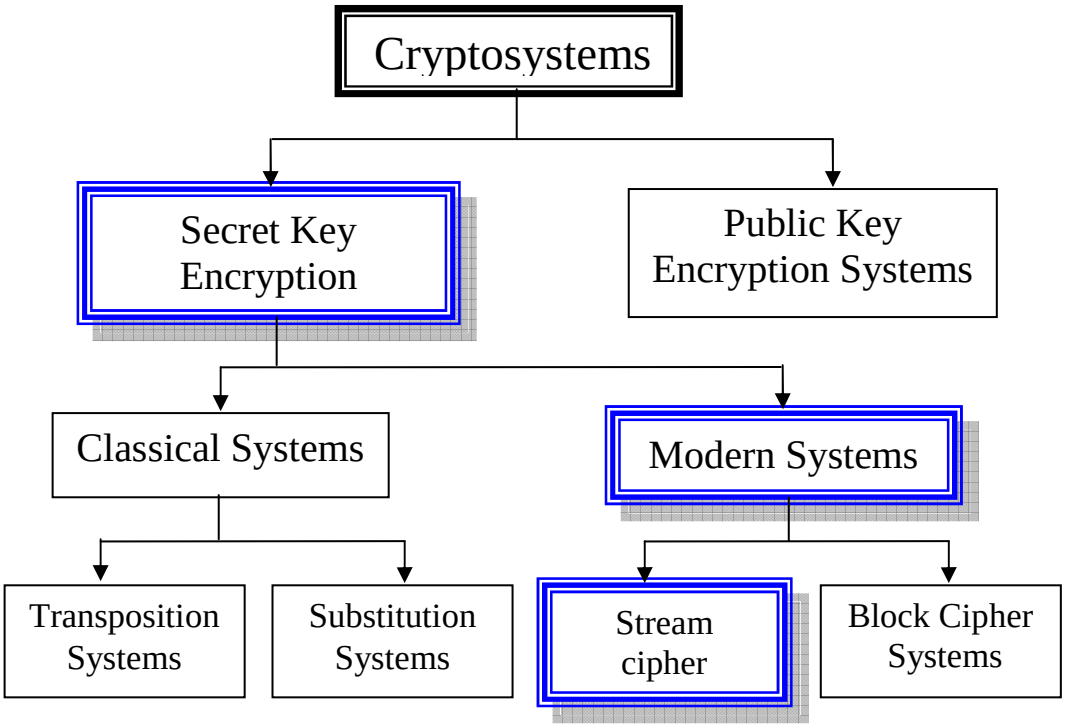


Figure (3.3) Cryptosystems classification

There are essentially two different types of cryptographic systems (cryptosystems), these cryptosystems are described in the next two subsections [Fra.94, Joh.02].

3.4.1 Public Key Cryptographic System

It is also called **asymmetric cryptosystems**. In a public key (**non-secret key**) cryptosystem (see figure (3.4)), the encryption key e_k and decryption key d_k are different, that is $e_k \neq d_k$.

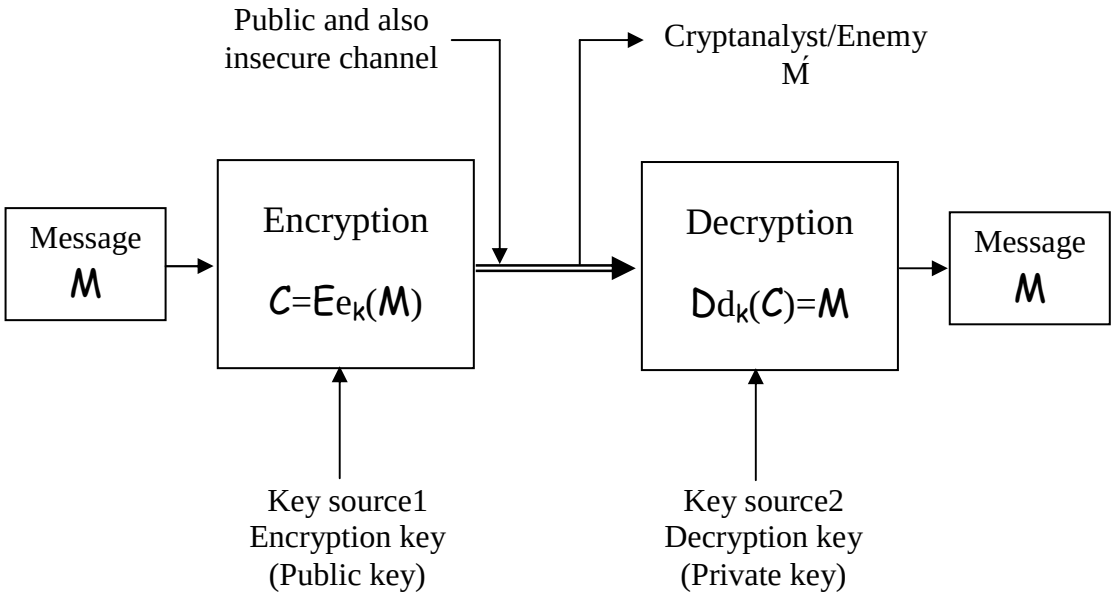


Figure (3.4) Modern public-key cryptosystem, $e_k \neq d_k$.

3.4.2 Secret Key Cryptographic System

It's also called **symmetric cryptosystems**. In a conventional secret-key cryptosystem (see figure (3.5)), the same key ($e_k=d_k=k \in K$), called **secret key**, used in both encryption and decryption; we are interest in this type of cryptosystems.

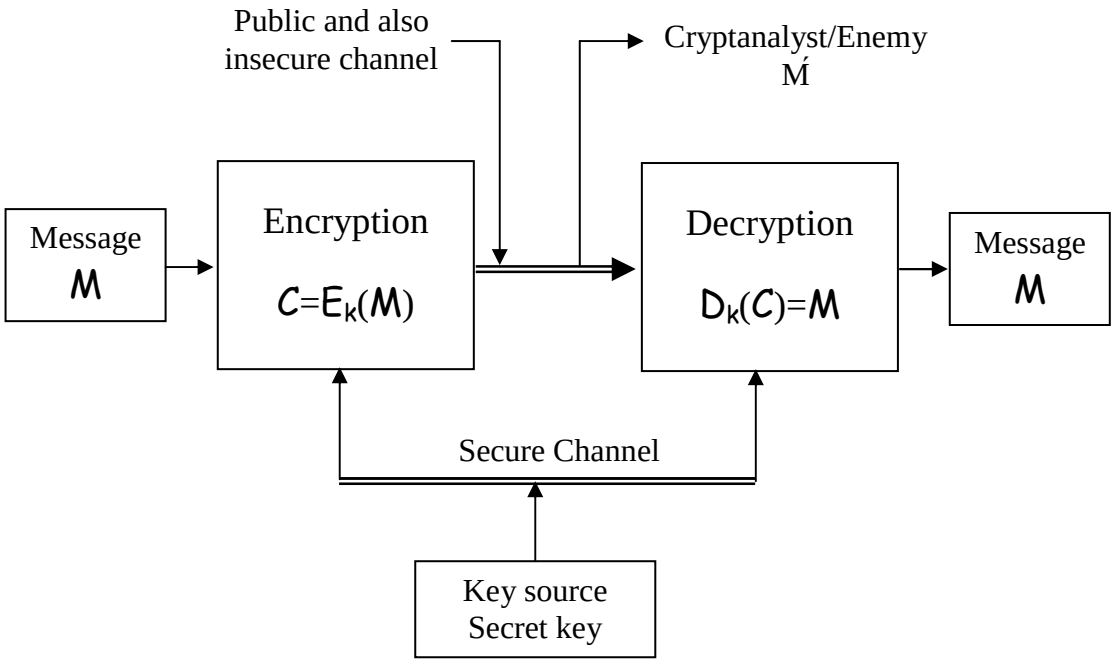


Figure (3.5) Conventional secret-key cryptosystems

The sender uses an invertible transformation *f* defined by:

$f : M \xrightarrow{k} C$

So produce the ciphertext:

$c=(E_k(m)), m \in M$ and $c \in C$.

and transmits it over the public insecure channel to the receiver. The key k should also be transmitted to the legitimate receiver for decryption but via a secure channel since the legitimate receiver knows the key k, he can decrypt c by transformation f^{-1} defined by:

$f^{-1} : C \xrightarrow{k} M$

and obtain:

$D_k(c)=D_k(E_k(m))= m$, $c \in C$ and $m \in M$,

and it's the original plaintext message.

There are many different types of secret key cryptosystems [Yan.00].

3.5 Cryptosystems Cryptanalysis

Unlike designing a cipher system, breaking a cipher system sometimes highly depends on guess and luck. Whatever method of cryptanalysis is used, the analyst must always expend some amount of time and resources (defined as work factor) to reach his goal. By increasing available resources, the time required to attack the cipher successfully can often be reduced.

Consequently, there is a relationship between cost and time for any given cryptanalytic attack against a cipher. Also, there is a relationship established between the value, of the information obtained by breaking a cipher, and time. These two relationships can be used to determine the practical secrecy of the cipher. Usually, cryptanalysis involves high-speed computers, and complex, sophisticated computer programs. This includes: first, computer processors, which may include special purpose hardware to execute the logical and arithmetic operations needed to obtain the solution, second, computer storage for the analysis programs and its data, and third human resources to devise and write analysis programs, gather data, and oversee the analysis [Mey.82].

The basic concepts of cryptanalysis were developed as a branch of applied mathematics; the cryptanalysis uses the following tools [Mey.82]:

1. Probability theory and statistics.
2. Linear algebra.
3. Abstract algebra (group theory).
4. Complexity theory.

3.6 Data Security [Men.96]

An encryption scheme is said to be **breakable** if a third party, without prior knowledge of the key pair (e_k, d_k), can systematically recover plaintext from corresponding ciphertext within some appropriate time frame.

An encryption scheme can be broken by trying all possible keys to see which one the communicating parties are using (assuming that the class of encryption functions is public knowledge). This is called an **exhaustive search** of the key space. It follows then that the number of keys (i.e., the size of the key space) should be large enough to make this approach computationally infeasible. It is the objective of a designer of an encryption scheme that this be the best approach to break the system.

The size of the key space is the number of encryption/decryption key pairs that are available in the cipher system. A key is typically a compact way to specify the encryption transformation (from the set of all encryption transformations) to be used. For example, a transposition cipher of block length t has t! encryption functions from which to select. Each can be simply described by a permutation which is called the key.

It is a great temptation to relate the security of the encryption scheme to the size of the key space. A necessary, but usually not sufficient, condition for an encryption scheme to be secure is that the key space be large enough to preclude exhaustive search. For instance, the simple substitution cipher has a key space of size $26! \approx 4 \times 10^{26}$. The polyalphabetic substitution cipher of (3) alphabets has a key space of size $(26!)^3 \approx 7 \times 10^{79}$. Exhaustive search of either key space is completely infeasible, yet both ciphers are relatively weak and provide little security.

3.7 Stream Cipher Systems

In **stream ciphers**, the message units are bits, and the key is usual produced by a **random bit generator** (see figure (3.6)). The plaintext is encrypted on a bit-by-bit basis.

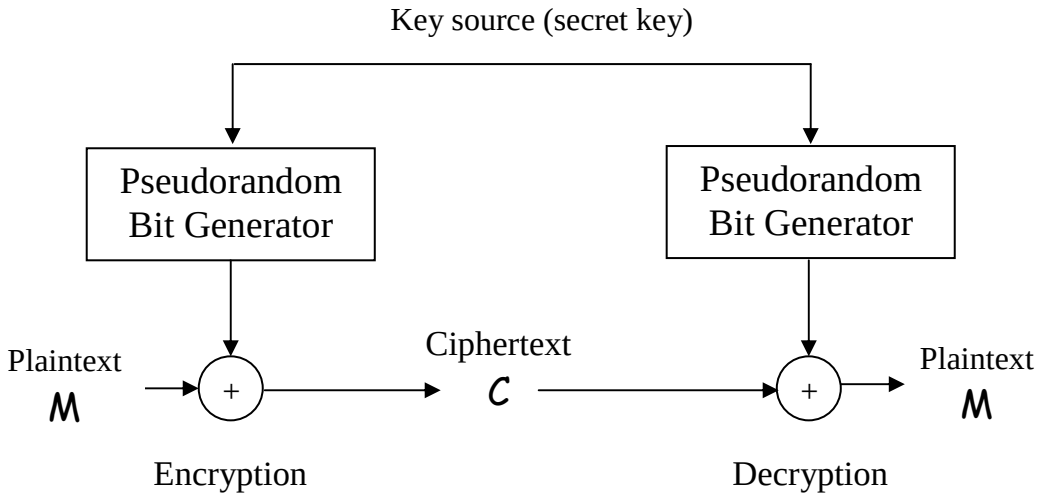


Figure (3.6) A stream (Bit) cipher.

The key is fed into random bit generator to create a long sequence of binary signals. This "key-stream" k is then mixed with plaintext m, usually by a bit wise XOR (Exclusive-OR modulo 2 addition) to produce the ciphertext stream, using the same random bit generator and seed.

Stream ciphers are an important class of encryption algorithms. They encrypt individual characters (usually binary digits) of a plaintext message one at a time, using an encryption transformation which varies with time. By contrast, block ciphers tend to simultaneously encrypt groups of characters of a plaintext message using a fixed encryption transformation. Stream ciphers are generally faster than block ciphers in hardware, and have less complex hardware circuitry. They are also more appropriate, and in some cases mandatory (e.g., in some telecommunications applications), when buffering is limited or when characters must be individually processed as they are received. Because they have limited or no error

propagation, stream ciphers may also be advantageous in situations where transmission errors are highly probable [Riv.97].

3.8 Combination Generator

3.8.1 Linear Feedback Shift Register

A **feedback shift register** is made up of two parts: a shift register and a **feedback function** (see figure (3.7)). The shift register is a sequence of bits, (the length of a shift register is figured in bits). Each time a bit is needed, all of the bits in the shift register are shifted 1 bit to the right. The new left-most bit is computed as a function of the other bits in the register. The output of the shift register is 1 bit, often the least significant bit. The period of a shift register is the length of the output sequence before it starts repeating.

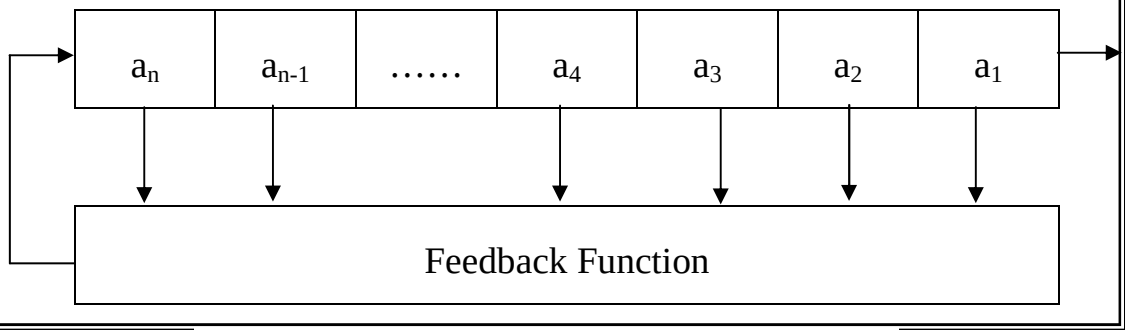


Figure (3.7) Feedback Shift Register.

Cryptographers have liked stream ciphers made up of shift registers: They are easily implemented in digital hardware. We will only touch on the mathematical theory.

Ernst Selmer, the Norwegian governments' chief cryptographer, worked out the theory of shift register sequences in 1965 [Sel.66]. Solomon Golomb, an NSA mathematician, wrote a book with Selmers results and some of his own [Gol.82, Sel.66].

The simplest kind of feedback shift register is a **Linear Feedback Shift Register** (LFSR), as described in figure (3.8). The feedback function is simply the XOR of certain bits in the register; the list of these bits is called a **tap sequence**. Because of the simple feedback sequence, a large body of mathematical theory can be applied to analyzing LFSRs. Cryptographers like to analyze sequences to convince themselves that they are random enough to be secure. LFSR's are the most common type of shift registers used in cryptography.

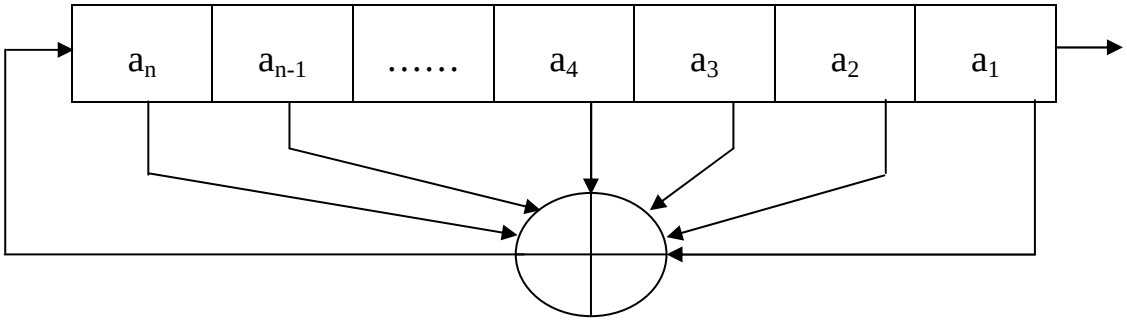


Figure (3.8) Linear Feedback Shift Register.

In order for a particular LFSR to be a **maximal-period LFSR**, the polynomial formed from a tap sequence plus the constant 1 must be a primitive polynomial mod 2. The degree of the polynomial is the length of

the shift register. A primitive polynomial of degree n is an irreducible polynomial.

Linear feedback shift registers (LFSRs) are used in many of the keystream generators that have been proposed in the literature. There are several reasons for this [Men.96]:

1. LFSRs are well-suited to hardware implementation.
2. They can produce sequences of large period.
3. They can produce sequences with good statistical properties.
4. Because of their structure, they can be readily analyzed using algebraic techniques.

A linear feedback shift register (LFSR) of length L consists of L stages (or delay elements) numbered 0,1,…,L−1, each capable of storing one bit and having one input and one output; and a clock which controls the movement of data. During each unit of time the following operations are performed:

1. The content of stage 0 is output and forms part of the output sequence.
2. The content of stage i is moved to stage i−1 for each i, $1 \leq i \leq L-1$.
3. The new content of stage L−1 is the feedback bit sj which is calculated by adding together modulo 2 the previous contents of a fixed subset of stages 0,1,…,L−1.

Every output sequence (i.e., for all possible initial states) of an LFSR $\langle L, C(D) \rangle$ is periodic if and only if the connection polynomial C(D) has degree L.

If $C(D) \in Z_2[D]$ is a primitive polynomial of degree L, then $\langle L, C(D) \rangle$ is called a **maximum-length LFSR**. The output of a maximum-length LFSR with non-zero initial state is called an **m-sequence**.

The basic approach to designing a keystream generator using LFSR is simple. First you take one or more LFSR, generally of different lengths and with different feedback polynomials. (If the lengths are all relatively prime and the feedback polynomials are all primitive, the whole generator is **maximal** length). The key is the initial state of the LFSR. Every time you want a bit, **shift** the LFSR once (this is sometimes called clocking). The output bit is a function, preferably a nonlinear function, of some of the bits of the LFSR. This function is called the **combining function**, and the whole generator is called a combination generator.

Three general methodologies for destroying the linearity properties of LFSRs are discussed in this section [Men.96]:

1. Using a nonlinear combining function on the outputs of several LFSRs.
2. Using a nonlinear filtering function on the contents of a single LFSR.
3. Using the output of one (or more) LFSRs to control the clock of one (or more) other LFSRs.

3.8.2 Generator Components

One approach is to use n LFSRs in parallel; their outputs combined using an n-input binary **Boolean function** or **combining function** (CF).

Figure (3.9) shows the design of n-LFSR's generator with combining function.

Because LFSRs are inherently linear, one technique for removing the linearity is to feed the outputs of several parallel LFSRs into a non-linear Boolean function to form a **combination generator**. Various properties of such a combining function are critical for ensuring the security of the resultant scheme, for example, in order to avoid correlation attacks [Rec.04, Mis.98, Mat.95].

Since a well-designed system should be secure against known plaintext attacks, an LFSR should never be used by itself as a keystream generator. Nevertheless, LFSRs are desirable because of their very low implementation costs.

For essentially all possible secret keys, the output sequence of an LFSR based keystream generator should have the following properties:

1. large period;
2. large linear complexity; and
3. good statistical properties.

It is emphasized that these properties are only **necessary** conditions for a keystream generator to be considered cryptographically secure. Since mathematical proofs of security of such generators are not known, such generators can only be deemed **computationally secure** after having withstood sufficient public security [Men.96].

The LFSRs in an LFSR-based keystream generator (Figure (3.9)) may have known or secret connection polynomials. For known connections, the secret key generally consists of the initial contents of the component

LFSRs. For secret connections, the secret key for the keystream generator generally consists of both the initial contents and the connections.

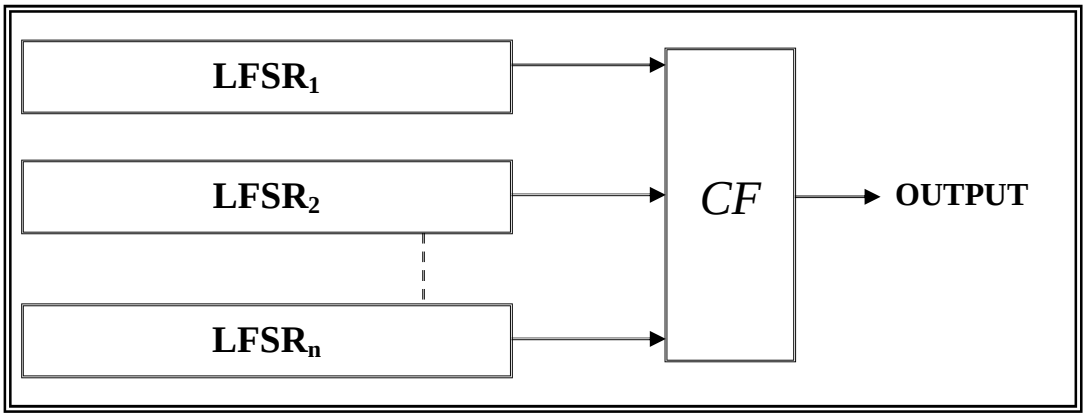


Figure (3.9) n-LFSR's Generator with Combining Function

3.8.3 Examples of Known Generators

Some examples of known keystream generators are introduced.

1. Linear Generator [Sch.96]

The **Linear generator**, illustrated in figure (3.9), is defined by n-maximum-length LFSRs whose lengths $r_1, r_2, \ldots, r_n$, where $n \in Z^+$ are pair wise relatively prime, with XOR combining function:

$$F(x_1,x_2,..,x_n) = x_1 \oplus x_2 \oplus \ldots \oplus x_n \quad \ldots (2.1)$$

This generator considered weak, despite of his good randomness, because of his weak linear complexity.

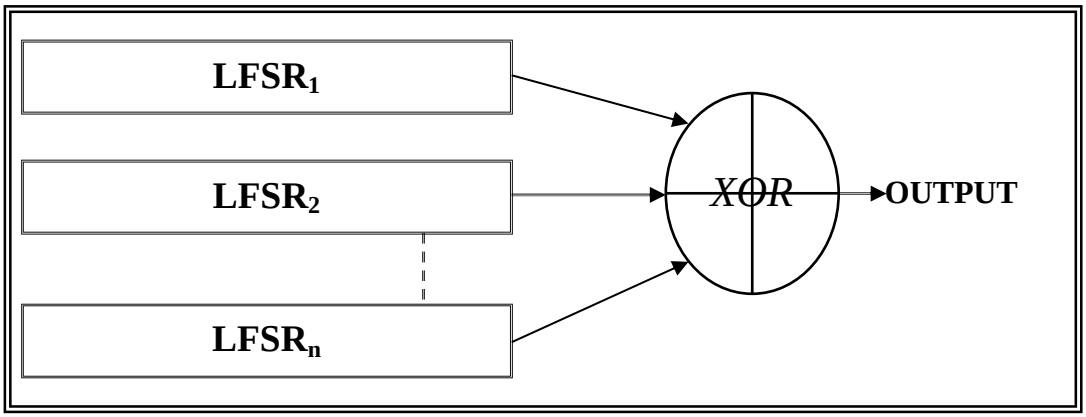


Figure (3.10xc) n-Linear Generator.

2. Product Generator [Sch.96]

The **Product generator**, illustrated in figure (3.10), is defined by n-maximum-length LFSRs whose lengths $r_1, r_2, \ldots, r_n$, where $n \in Z^+$ are pair wise relatively prime, with AND combining function:

$$F(x_1,x_2,..,x_n) = x_1 \bullet x_2 \bullet \ldots \bullet x_n = \prod_{i=1}^{n} x_i$$

This generator considered weak, despite of his good linear complexity, because of his weak randomness.

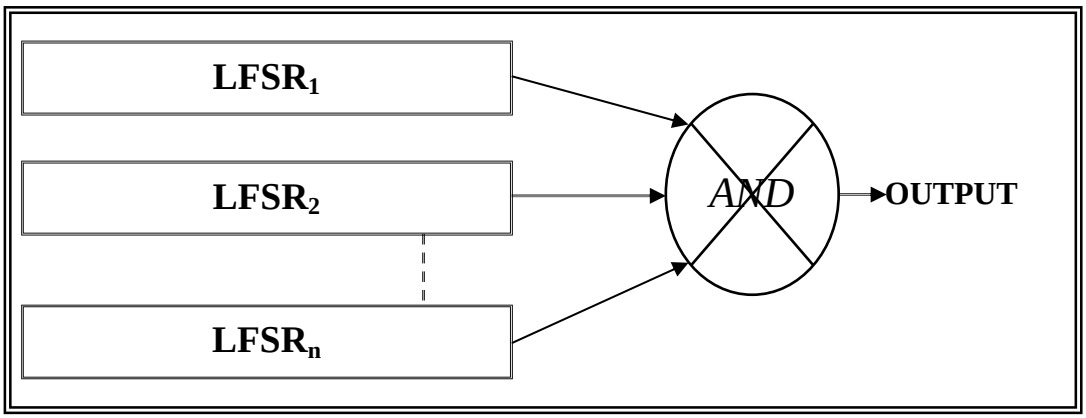


Figure (3.11) n-Product Generator.

3.9 Attacking of Stream Cipher Systems

3.9.1 Attacking of Stream Cipher Methods

The attacking methods of stream cipher could be classified as displayed in Figure (3.11).

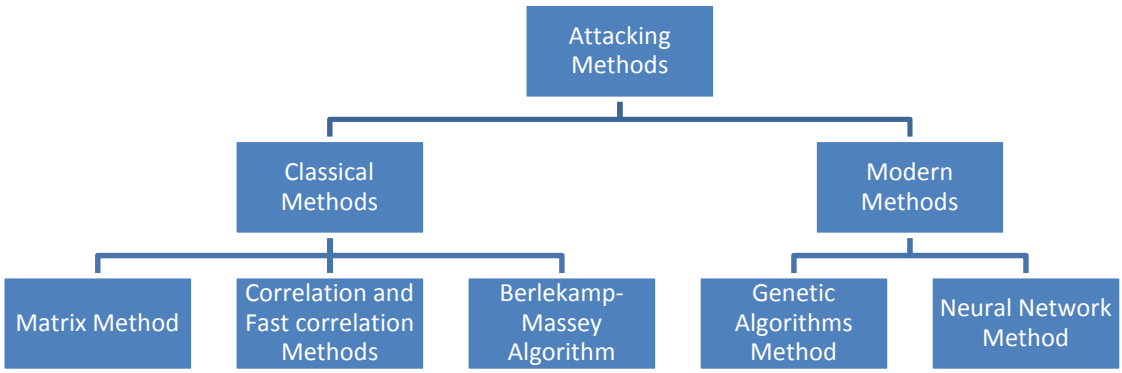


Figure (3.12) Attacking methods of Stream Cipher.

The classification is based on the information processing approaches and the tools that are utilized by the cryptanalysis. From figure (3.12), clearly the classification has two major parts: classical methods, and modern methods. The classical methods include the most efficient methods used for attacking the stream cipher, but they are not the only methods in this area; they are: matrix method, Massey algorithm, and correlation methods (including fast correlation). These methods usually may use some or all cryptanalysis requirements and tools given in this section. However, the modern methods depend on different approaches for information processing, namely biological like processing. We classify them as modern methods since they depend on new tools, which are:

1. Genetic Algorithms [Gol.89].
2. Neural Networks [Hec.90].

These new tools can greatly facilitate cryptanalysis, as we shall explain in this chapter. In fact, the variety of new technologies increases the possibilities of attack. So that, each major advance in information technology must change our ideas about data security [Dav.89]. The requirements summed up by the phrase "data security" do not stay the same; they change rapidly as the technology changes. One of the reasons for the speed of development in cryptography is, of course, the presence of cryptanalyst [Bek.82].

(a). Classical Cryptanalysis Methods

Although there is a considerable literature on the design of cryptography system, relatively little public domain information exists on techniques for cryptanalysis. In this section we present an overview of the most efficient methods used to attack stream cipher (linear and non-linear type):

1. Matrix Method

Meyer et al [Mey.82, Aug.85], have demonstrated a method of breaking an “N-stage” LFSR given (2N) consecutive bits of known plaintext. The basic method is to set up the matrix equation: K = MC

$$\begin{bmatrix} K_{n+1} \\ K_{n+2} \\ K_{n+3} \\ \vdots \\ K_{2n} \end{bmatrix} = \begin{bmatrix} M_n & M_{n-1} & M_{n-2} & \cdots & M_1 \\ 0 & M_n & M_{n-1} & \cdots & M_2 \\ 0 & 0 & M_n & \cdots & M_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & M_n \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ \vdots \\ C_n \end{bmatrix}$$

where

M : is the matrix of successive shifts of the first n bits of plaintext.

C : is the unknown matrix of switch states.

K. : is the key matrix.

Thus the switch values can be obtained by inverting M and solving:

$$C = M^{-1} K$$

Therefore, the entire key will be known completely. Although finding the inverse of matrix is not trivial (the time taken to find the inverse matrix over GF(2) is proportional to $O(N^3)$, it is a straightforward process. But this method is weak, since if less than 2N consecutive bits are known this may not be enough to determine the entire sequence.

Beker and Piper [Bek.82], showed two methods of breaking an “N-stage” LFSR given (2N) bits of the known plaintext sequence where the bits are not necessarily consecutive. These methods are based on trying every possibility for filling unknown entries in the sequence, and for each one they merely use the techniques of the matrix equation, so that, they involve a mixture of guessing and solving simultaneous equations. The choice between these two methods is usually made by seeing which will require fewer trials.

2. Berlckamp-Massey Shift Register Synthesis Algorithm

The iterative algorithm introduced by BerleKamp [Mas.69] for decoding Bose-Chaudhuri-Hocquenghem (BCH) codes, provides efficient solution to the problem of synthesizing the shortest linear feedback shift register capable of generating a finite sequence of bits. The algorithm leads to the polynomial of smallest possible degree (N) as providing (2N) bits from shift register of length (N) [Hen.89].

3. Correlation and Fast Correlation Methods

Siegenthaler [Sie.85], in 1985, demonstrated a method that the $LFSR_i$ part of the key can be found independently of the other $LFSR_s$ parts, by using the "divide and conquer" technique. This method showed that the number of trials to find the key can be reduced significantly in those cases where a correlation exists between the output of the running key generator employed in a stream cipher, and $LFSR_i$ sequence with correlation probability P (up to 0.75). These generators have been broken for LFSR lengths ($N < 50$).

Meier and Staffelbach [Sta.89], in 1989, developed two algorithms (A and B), which are much faster than the above attack and are demonstrated to be successful against shift registers of considerable length N ($N>>50$), provided that the number t of feedback taps is small ($t< 10$ if $P \leq 0.75$).

Both A and B algorithms are developed by using the statistical model of Siegenthaler. On the other hand, for correlation probabilities $P<0.75$ the attacks are proven to be infeasible against long LFSRs if they have a greater number of taps (roughly $N \geq 100$ and $t \geq 10$).

(b). Modern Cryptanalysis Methods

The modern methods of cryptanalysis are based on new and different approaches to minimize the time and the cost of attacking. These methods, as displayed in figure (3.10), are directed to utilize genetic algorithms and neural network concepts.

1. Genetic Algorithms Methods

Genetic Algorithms are search algorithms based on the mechanics of natural selection and natural genetics. They use concepts drawn from the theory of evolution to "breed" progressively better solutions to problems with very large solution spaces [Obe.89]. With their ability to efficiently search huge solution spaces, genetic algorithms would seem a natural candidate for use in cryptanalysis [Mat.93].

Matthews [Mat.93], demonstrated the use of genetic algorithms to break classical transposition ciphers by finding the transposition sequence used.

Spillman, et al [Spi.93-1] showed the use of genetic algorithms in the cryptanalysis of simple substitution ciphers.

Spillman and Rechard [Spi.93-2], used a new method to attack knapsack cipher using genetic algorithms to decrease the time for breaking.

However, there are a number of cryptographic algorithms that cannot be attacked using genetic algorithms, a large number of cryptographic algorithms, including those underpinning many rotor-based systems (for example stream cipher systems), appear vulnerable to attack by genetic algorithms [Mat.93].

2. Neural Networks Method

The general interest in the neural networks arises from their fascinating properties which enable them to exceed the limitations of traditional information processing.

Although it seems that neural network is a valuable tool for use in cryptanalysis, neural network is not a panacea. Therefore an important question must be answered: Does cryptanalysis make a suitable problem to be solved by Neural Networks? To answer, we must know the characteristics that must be exhibited by the problem to be suitable for solution by neural network. They are [Dav.91]:

1. The rule used in solving the problem may be unknown, or very difficult to explain or formalize.
2. The problem makes use of noisy, or incomplete data.
3. The problem may evolve. ?
4. The problem needs very high speed processing.
5. There may be no current technical solutions.

3.9.2 Types of Attack [Riv.97]

In this section, some techniques of cryptanalysts should be discussed. The general assumption that is usually made is that the opponent knows the cryptosystem being used. This is usually referred to as Kerckhoff's principle. Of course, if the opponent does not know the cryptosystem being used, that will make his task more difficult. But we do not want to base the security of a cryptosystem on the (possibly shaky) premise that the opponent does not know what system is being employed. Hence, our goal

in designing a cryptosystem will be to obtain security under Kerckhoff's principle.

We want to differentiate between different levels of attacks on cryptosystems. The most common types are enumerated as follows.

1. **Ciphertext-only**: The opponent possesses a string of ciphertext, y.
2. **Known plaintext**: The opponent possesses a string of plaintext, x, and the corresponding ciphertext, y.
3. **Chosen plaintext**: The opponent has obtained temporary access to the encryption machinery. Hence he can choose a plaintext string, x, and construct the corresponding ciphertext string, y.
4. **Chosen ciphertext**: The opponent has obtained temporary access to the decryption machinery. Hence he can choose a ciphertext string, y, and construct the corresponding plaintext string, x.

In each case, the object is to determine the key that was used. Clearly, these four levels of attacks are enumerated in increasing order of strength.

3.9.3 Classes of Attacks [Men.96]

Over the years, many different types of attacks on cryptographic primitives and protocols have been identified. The discussion here limits consideration to attacks on encryption and protocols. The attacks of the adversaries can mount may be classified as follows:

1. A passive attack is one where the adversary only monitors the communication channel. A passive attacker only threatens confidentiality of data.

2. An active attack is one where the adversary attempts to delete, add, or in some other way alter the transmission on the channel. An active attacker threatens data integrity and authentication as well as confidentiality.

3.10 Concept of Basic Efficiency Criteria

As known before, any stream cipher key generator consists of two basic units; they are sequence(s) of bit stream and **C**ombining **F**unction (CF) for the key generator. Any weakness in any one of these units means clear weakness in output key generator sequence, so there are some conditions must be available in key generator before it is constructed.

Next, we will introduce the sequence efficiency in order to use the sequence as encryption key. It is important to mention that the "efficient sequence" is compatible to "efficient key generator". The **basic criteria** of key generator efficiency can be defined as the ability of key generator and its sequence to withstand the mathematical analysis which the cryptanalyst can be applied on them.

The criteria of key generator efficiency depend on some/all elements of basic units of key generator, for this reason these may intersect each other's. An increased criterion may cause negative effect on the other's, which may be increase or decrease the ability of key generator efficiency. For instance, it is not necessary that the linear complexity of key generator be high as possible to gain efficient key generator but it is very important that the efficient key generator has balance CF (balance output bits and balance different strings in the produced sequence) to produce pseudo random sequence.

In the following sections, we will introduce these basic criteria. In order to show how they can be estimated. First we have to define some samples.

Let key generator consists of n input sequences S_i, $1 \le i \le n$, let $S=\{s_0,s_1,\ldots\}$ be the output sequence which product from the key generator and s_j, $j=0,1,\ldots$ represents the elements j of S.

3.11 Periodicity (P) [Men.96]

Definition (3.10): Let $S=s_0,s_1,s_2,\ldots$ be an infinite sequence. The subsequence consisting of the first n terms of S is denoted by $S^n = s_0, s_1, s_2,\ldots, s_{n-1}$.

Definition (3.11): The sequence $S=s_0,s_1,s_2 \ldots$ is said to be **n-periodic** if $s_i=s_{i+n}$ for all $i \ge 0$. The sequence S is **periodic** if it is n-periodic for some positive integer n. The period of a periodic sequence S is the smallest positive integer n for which S is n-periodic. If S is a periodic sequence of period n then the cycle of S is the subsequence S^n.

Let P(S) represent the period of the sequence S, let $P(S_i)$ be the period of the sequence S_i, $1 \le i \le k$, then

$$P(S) = \text{lcm}(P(S_1),P(S_2),\ldots,P(S_k)) \quad \ldots(3.2)$$

Of course if $P(S_i)$ are relatively prime to each other $\forall i$, $1 \le i \le k$, then

$$P(S)=\prod_{i=1}^{k} P(S_i) \quad \ldots(3.3)$$

3.12 Linear complexity (LC)

This subsection summarizes selected results about the linear complexity of sequences. All sequences are assumed to be binary sequences.

Definition (3.12): An LFSR is said to **generate** a finite sequence S^n if there is some initial state for which the output sequence of the LFSR has S^n as its first n terms.

Definition (3.13) [Rue.86-1,Rue.86-2]: The **linear complexity** of an infinite binary sequence S, denoted **LC(S)**, is defined as follows:

(i). if S is the zero sequence S = 0, 0, 0, . . . , then LC(S) = 0;

(ii). if no LFSR generates S, then LC(S) = ∞;

(iii). otherwise, LC(S) is the length of the shortest LFSR that generates S.

Definition (3.14): The **linear complexity** of a finite binary sequence S^n, denoted $LC(S^n)$, is the length of the shortest LFSR that generates a sequence having S^n as its first n terms.

3.12.1 Berlekamp-Massey Algorithm [Mas.69,Bla.87]

The Berlekamp-Massey algorithm is an efficient algorithm for determining the linear complexity of a finite binary sequence S^n of length n (see Definition (3.14)). The algorithm takes n iterations, with the N^{th} iteration computing the linear complexity of the subsequence S^N consisting of the first N terms of S^n. The theoretical basis for the algorithm is Remark (3.1).

Definition (3.15) Consider the finite binary sequence $S^{N+1}=s_0,s_1,...,s_{N-1}, s_N$. For $C(D)=1+c_1D+\ldots+c_rD^r$, let $\langle r,C(D)\rangle$ be an LFSR that generates the subsequence $S^N =s_0, s_1,..., s_{N-1}$. The next discrepancy d_N is the difference between s_N and the $(N+1)^{st}$ term generated by the LFSR:

$$d_N=(s_N+\sum_{i=1}^{r} c_i s_{N-i}) \bmod 2 \qquad \ldots(3.3)$$

Remark (3.1): Let $S^N = s_0, s_1,..., s_{N-1}$ be a finite binary sequence of linear complexity $LC=LC(S^N)$, and let $\langle r,C(D)\rangle$ be an LFSR which generates S^N.

(i). The LFSR $\langle r,C(D)\rangle$ also generates $S^{N+1} = s_0, s_1,..., s_{N-1}, s_N$ if and only if the next discrepancy d_N is equal to 0.

(ii). If $d_N = 0$, then $LC(S^{N+1}) = r$.

(iii). Suppose $d_N=1$. Let m the largest integer<N such that $LC(S^m)<LC(S^N)$, and let $\langle LC(S^m),B(D)\rangle$ be an LFSR of length $LC(S^m)$ which generates S^m. Then $\langle r',C'(D)\rangle$ is an LFSR of smallest length which generates S^{N+1}, where

$$r' = \begin{cases} r, & \text{if } r > N/2, \\ N+1-r, & \text{if } r \le N/2. \end{cases} \qquad \ldots(3.4)$$

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

Berlekamp-Massey algorithm

INPUT: a binary sequence $S^n = s_0, s_1, s_2,\ldots, s_{n-1}$ of length n.

OUTPUT: the linear complexity $Lc(S^n)$ of S^n, $0 \le Lc(S^n) \le n$.

PROCESS: 1. Initialization. C(D)←1, r←0, m← −1, B(D)←1, N←0.

2. While (N <n) do the following:

2.1Compute the next discrepancy d. d←$(s_N + \sum_{i=1}^{r} c_i s_{N-i})$mod 2.

2.2 If d = 1 then do the following:

T (D)←C(D), C(D)←C(D) + B(D).D^{N-m}.

If r≤ N/2 then r←N + 1 − r, m←N, B(D)←T (D).

2.3 N←N + 1.

3. Return(r).

Example (3.3) (Berlekamp-Massey algorithm), Table (3.2) shows the steps of Berlekamp-Massey algorithm for computing the linear complexity of the binary sequence S^n = 0, 0, 1, 1, 0, 1, 1, 1, 0 of length n=9. This sequence is found to have linear complexity 5, and an LFSR which generates it is $\langle 5, 1+D^3+D^5 \rangle$.

Remark (3.2): Let S^n be a finite binary sequence of length n, and let the linear complexity of S^n be LC. Then there is a unique LFSR of length LC which generates S^n if and only if LC $\leq$ n/2.

Table (3.2) Steps of the Berlekamp-Massey algorithm of example (3.3).

s_N	d_N	T(D)	C(D)	r	m	B(D)	N
-	-	-	1	0	-1	1	0
0	0	-	1	0	-1	1	1
0	0	-	1	0	-1	1	2
1	1	1	$1+D^3$	3	2	1	3
1	1	$1+D^3$	$1+D+D^3$	3	2	1	4
0	1	$1+D+D^3$	$1+D+D^2+D^3$	3	2	1	5
1	1	$1+D+D^2+D^3$	$1+D+D^2$	3	2	1	6
1	0	$1+D+D^2+D^3$	$1+D+D^2$	3	2	1	7
1	1	$1+D+D^2$	$1+D+D^2+D^5$	5	7	$1+D+D^2$	8
0	1	$1+D+D^2+D^5$	$1+D^3+D^5$	5	7	$1+D+D^2$	9

3.12.2Non-Linearity of Combining Functions [Rue.87,Kla.94]

One general technique for destroying the linearity inherent in LFSRs is to use several LFSRs in parallel. The keystream is generated as a nonlinear function *f* of the outputs of the component LFSRs. Such keystream generators are called **nonlinear combination generators**, and *f* is called the **combining function**. The remainder of this subsection demonstrates that the function *f* must satisfy several criteria in order to withstand certain particular cryptographic attacks.

Definition (3.16) A product of m distinct variables is called an **m^{th} order product** of the variables. Every Boolean function $f(x_1, x_2,..., x_n)$ can be written as a modulo 2 sum of distinct m^{th} order products of its variables, $0 \leq m \leq n$; this expression is called the **algebraic normal form of *f***. The nonlinear order of *f* is the maximum of the order of the terms appearing in its algebraic normal form.

Example (3.4) the Boolean function $f(x_1,x_2,x_3,x_4,x_5) = 1 \oplus x_2 \oplus x_3 \oplus x_4x_5 \oplus x_1x_3x_4x_5$ has nonlinear order 4. Note that the maximum possible nonlinear order of a Boolean function in n variables is n. Remark (3.3) demonstrates that the output sequence of a nonlinear combination generator has high linear complexity, provided that a combining function *f* of high nonlinear order is employed.

Remark (3.3): Suppose that n maximum-length LFSRs, whose lengths r_1, $r_2,\ldots,r_n$ are pairwise distinct and greater than 2, are combined by a nonlinear function $f(x_1, x_2,..., x_n)$ which is expressed in algebraic normal form. Then the linear complexity of the keystream is $f(r_1, r_2,\ldots, r_n)$. (The expression $f(r_1,r_2,\ldots,r_n)$ is evaluated over the integers rather than over Z_2).

3.13 Randomness (R)

Random number generation is an important primitive in many cryptographic mechanisms. For example, keys for encryption

transformations need to be generated in a manner which is unpredictable to an adversary. Generating a random key typically involves the selection of random numbers or bit sequences. Random number generation presents challenging issues [Men.96].

Any random-number generator on a computer (at least, on a finite-state machined) is, by definition, periodic. Anything that is periodic is, by definition, predictable. And if something is predictable, it cannot be random. A true random-number generator requires some random input; a computer cannot provide that [Bau.95].

This section introduces basic concepts relevant to random and pseudorandom bit generation, considers a technique as a sample for pseudorandom bit generation, and describes statistical tests designed to measure the quality of a random bit generator.

Definition (3.17) [Ben.99]: A **random bit generator** is a device or algorithm which outputs a sequence of statistically independent and unbiased binary digits.

Remark (3.4) [Ben.99]: (**random bits vs. random numbers**) A random bit generator can be used to generate (uniformly distributed) random numbers. For example, a random integer in the interval [0,n] can be obtained by generating a random bit sequence of length $\log_2^{|n+1|}$ bits, and converting it to an integer; if the resulting integer exceeds n, one option is to discard it and generate a new random bit sequence.

Definition (3.18) [Ben.99]: A **Pseudo Random Bit Generator** (PRBG) is a deterministic algorithm which, given a truly random binary sequence of length k, which “appears” to be random. The input to the **Pseudo Random**

Bit Generator (PRBG) is called the **seed**, while the output of the PRBG is called a **pseudorandom bit sequence**.

A minimum security requirement for a pseudorandom bit generator is that the length k of the random seed should be sufficiently large so that a search over 2^k elements (the total number of possible seeds) is infeasible for the adversary [Mot.95].

3.13.1 Requirements for Random Number Generators [Cod.97]

Ideally a pseudo-random number generator would produce a stream of numbers that:

1. Are uniformly distributed.
2. Are uncorrelated.
3. Never repeats itself.
4. Satisfy any statistical test for randomness.
5. Are reproduceable (for debugging purposes).
6. Are portable (the same on any computer).
7. Can be changed by adjusting an initial “seed” value.
8. Can easily be split into many independent subsequences.
9. Can be generated rapidly using limited computer memory.

In practice it is impossible to satisfy all these requirements exactly. Since a computer uses finite precision arithmetic to store the state of the generator, after a certain period the state must match that of a previous

iteration, after which the generator will repeat itself. Also, since the numbers must be reproduceable, they are not truly random, but generated by a deterministic iterative process, and therefore cannot be completely uncorrelated.

3.13.2 Statistical Tests [Ben.99,Mau.92]

Definition (3.19): A **statistical hypothesis**, denoted H_0, is an assertion about a distribution of one or more random variables.

A test of a statistical hypothesis is a procedure, based upon the observed values of the random variables, that leads to the acceptance or rejection of the hypothesis H_0. The test only provides a measure of the strength of the evidence provided by the data against the hypothesis; hence, the conclusion of the test is not definite, but rather probabilistic.

Definition (3.20): The significance level of the test of a statistical hypothesis H_0 is the probability of rejecting H_0 when it is true.

In this section, H_0 will be the hypothesis that a given binary sequence was produced by a random bit generator. If the significance level cat of a test of H_0 is too high, then the test may reject sequences that were, in fact, produced by a random bit generator. On the other hand, if the significance level of a test of H_0 is too low, then there is the danger that the test may accept sequences even though they were not produced by a random bit generator. It is, therefore, important that the test be carefully designed to have a significance level that is appropriate for the purpose at hand; a significance level or between 0.001 and 0.05 might be employed in practice. A statistical test is implemented by specifying a statistic on the random sample. Statistics are generally chosen so that they can be efficiently computed, and so that they (approximately) follow an N(0,1) or

a χ^2 distribution. The value of the statistic for the sample output sequence is computed and compared with the value expected for a random sequence.

Suppose that a statistic X for a random sequence follows a χ^2 distribution with υ degrees of freedom, and suppose that the statistic can be expected to take on larger values for nonrandom sequences. To achieve a significance level of α, a threshold value x_α is chosen (using Chi square table) so that $Pr(X>x_\alpha)=\alpha$. If the value X_s of the statistic for the sample output sequence satisfies $X_s>x_\alpha$, then the sequence fails the test; otherwise, it passes the test.

3.13.3 Golomb's Concept of Randomness [Men.96,Gol.82]

Golomb's randomness postulates are presented here for historical reasons they were one of the first attempts to establish some necessary conditions for a periodic pseudorandom sequence to look random. It is emphasized that these conditions are far from being sufficient for such sequences to be considered random.

Definition (3.21): Let S be a sequence. A **run** of S is a subsequence of S consisting of consecutive 0's or consecutive 1's which is neither preceded nor succeeded by the same symbol.
A run of 0's is called a **gap**, while a run of 1's is called a **block**.

Definition (3.22): Let $S= s_0,s_1,s_2\ldots$ be a periodic sequence of period n. The **autocorrelation function** of S is the integer-valued function C(t) defined as:

$$n.C(t)=\sum_{i=0}^{n-1}(2s_i-1)\cdot(2s_{i+t}-1),\ 0\le t\le n-1.$$

The autocorrelation function C(t) measures the amount of similarity between the sequence S and a shift of S by t positions. If S is a random periodic sequence of period n, then n.C(t) can be expected to be quite small for all values of t, $0<t<n$.

Definition (3.23): Let S be a periodic sequence of period n. Golomb′s randomness postulates are the following:

R1: In the cycle S^n of S, the number of 1′s differs from the number of 0′s by at most 1.

R2: In the cycle S^n at least half the runs have length 1, at least one-fourth have length 2, at least one-eighth have length 3, etc., as long as the number of runs so indicated exceeds 1. Moreover, for each of these lengths, there are (almost) equally many gaps and blocks.

R3: The autocorrelation function C(t) is two-valued. That is for some integer K:

$$n.C(t)=\sum_{i=0}^{n-1}(2s_i-1)\cdot(2s_{i+t}-1)=\begin{cases}n & ,t=0\\ K & ,1\le t\le n-1\end{cases} \qquad \ldots(3.5)$$

Theorem (3.4): The output sequence of any maximum shift register satisfies Golomb's postulates.
The mathematical proof of this theorem can be found in [Gol.82].

Definition (3.24): A binary sequence which satisfies Golomb′s randomness postulates is called a **pseudo-noise sequence** or a **pn-sequence**.

Pseudo-noise sequences arise in practice as output sequences of maximum-length linear feedback shift registers.

Example (3.5): (pn-sequence) Consider the periodic sequence S of period N=15 with cycle S^{15}=011001000111101. The following shows that the sequence S satisfies Golomb′s randomness postulates.

R1: The number of 0′s in S^{15} is 7, while the number of 1′s is 8.

R2: S^{15} has 8 runs. There are 4 runs of length 1 (2 gaps and 2 blocks), 2 runs of length 2 (1 gap and 1 block), 1 run of length 3 (1 gap), and 1 run of length 4 (1 block).

R3: The autocorrelation function C(t) takes on two values :C(0)=1 and C(t)=1/15 for 1≤t≤14. Hence, S is a pn-sequence.

3.13.4 Standard Statistical Randomness Tests [Gol.82, Bek.82, Ben.99]

Let $S=s_0,s_1,s_2\ldots,s_{n-1}$ be a binary sequence of length n. This subsection presents five statistical tests that are commonly used for determining whether the binary sequence s possesses some specific characteristics that a truly random sequence would be likely to exhibit. It is emphasized again that the outcome of each test is not definite, but rather probabilistic. If a sequence passes all five tests, there is no guarantee that it was indeed produced by a random bit generator.

It is important to mention that the frequency, run and auto correlation test are called the **Main Binary Standard Randomness Tests** (MBSRT).

Before we shed light on the five basic tests, we have to construct the law of Chi-square which we really used is:

Assume that the outcome of a random experiment falls into one of k categories, and assume by hypothesis that p_i is the probability that the outcome falls into category i, assume that L independent observation is

made, and let Q_i be the number of observation falling into category i, in order to test the hypothesis the quantity T is compared:

$$T=\sum_{i=1}^{k}\frac{(Q_i - L.p_i)^2}{L.p_i} \qquad \ldots(3.6)$$

If the hypothesis is true, the value T is distribute according to the χ^2 distribution with υ=k-1 degree of freedom, the hypothesis is rejected if Q_i and Lp_i are too different, i.e. if T is too big, that means we set some pass mark x_0 and reject the hypothesis if T greater than x_0, α will be the significance level of the test, of course $E_i=Lp_i$ s.t. E_i is the expected value of occurrence of outcome i.

I.Frequency test (monobit test)

The purpose of this test is to determine whether the number of 0′s and 1′s in S are approximately the same, as would be expected for a random sequence. Let n_0, n_1 denote the observed number of 0′s and 1′s in S, respectively. The expected value is n/2.

The statistic used is:

$$X_1=\sum_{i=0}^{1}\frac{(n_i - n/2)^2}{n/2}=\frac{(n_0 - n_1)^2}{n} \qquad \ldots(3.7)$$

which approximately follows a χ^2 distribution with 1 degree of freedom.

II.Serial test (two-bit test)

The purpose of this test is to determine whether the number of occurrences of 00, 01, 10, and 11 as subsequences of s are approximately the same, as would be expected for a random sequence. Let n_0, n_1 denote the number of 0′s and 1′s in s, respectively, and let n_{00},

n_{01}, n_{10}, n_{11} denote the observed number of occurrences of 00,01,10,11 in s, respectively. Note that $n_{00}+n_{01}+n_{10}+n_{11}=n-1$ since the subsequences are allowed to overlap. The expected value is (n-1)/4.

The statistic used is:

$$X_2=\sum_{i=0}^{1}\sum_{j=0}^{1}\frac{(n_{ij}-(n-1)/4)^2}{(n-1)/4} \qquad \ldots(3.8)$$

which approximately follows a χ^2 distribution with 3 degrees of freedom.

III.Poker test

Let m be a positive integer such that m≥3, and let k=m. Divide the sequences into k non-overlapping parts each of length m, and let n_i be the observed number of occurrences of the i^{th} type of sequence of length m, 0≤i≤m. The poker test determines whether the sequences of length m each appear approximately the same number of times in S, as would be expected for a random sequence. The expected value of the string which consists of i (1's) with no consideration to arrangement of (1's) is:

$$E_i=C_i^m\cdot\frac{1}{2^m}\cdot\frac{n}{m}$$

The statistic used is:

$$X_3=\sum_{i=0}^{m}\frac{(n_i-C_i^m\cdot\frac{1}{2^m}\cdot\frac{n}{m})^2}{C_i^m\cdot\frac{1}{2^m}\cdot\frac{n}{m}} \qquad \ldots(3.9)$$

which approximately follows a χ^2 distribution with υ=m degrees of freedom. Note that the poker test is a generalization of the frequency test: setting m=1 in the poker test yields the frequency test.

IV. Runs test

The purpose of the runs test is to determine whether the number of runs (of either zeros or ones) of various lengths in the sequence S is as expected for a random sequence. The expected number of gaps (or blocks) of length i in a random sequence of length n is:

$$E_i = \frac{n - i + 3}{2^{i+2}}$$

Let k be equal to the largest gap (block). Let B_i, G_i, be the observed number of blocks and gaps, respectively, of length i in S for each i, $1 \le i \le k$. The statistic used is:

$$X_{4,0} = \sum_{i=1}^{k} \frac{(G_i - E_i)^2}{E_i}, \; X_{4,1} = \sum_{i=1}^{k} \frac{(B_i - E_i)^2}{E_i} \qquad \ldots(3.10)$$

which approximately follows a χ^2 distribution with biggest gap (block) as a degrees of freedom.

V. Autocorrelation test

The purpose of this test is to check for correlations between the sequence S and (non-cyclic) shifted versions of it. Let τ be a fixed integer, $1 \le \tau \le n/2$. The expect value $E=(n-\tau)/2$. The number of bits in S not equal to their τ-shifts is:

$$s^{\tau} = \left\{ s_i^{\tau} = s_i \oplus s_{i+\tau} \right\}_{i=1}^{n-\tau}, \qquad \ldots(3.11)$$

where $\oplus$ denotes the XOR operator.

Let $n_0(\tau)$ and $n_1(\tau)$ denote the observed number of 0′s and 1′s in $A(\tau)$, respectively. The statistic used is:

$$X_5 = \frac{\left(n_0(\tau) - \frac{n-\tau}{2}\right)^2}{\frac{n-\tau}{2}} + \frac{\left(n_1(\tau) - \frac{n-\tau}{2}\right)^2}{\frac{n-\tau}{2}} = \frac{(n_0(\tau) - n_1(\tau))^2}{n - \tau} \qquad \ldots(3.12)$$

which approximately follows a χ^2 distribution with $\upsilon=1$ degrees of freedom.

Example (3.6): (basic statistical tests)

Consider the (non-random) sequence S of length n = 160 obtained by replicating the following sequence four times: 11100 01100 01000 10100 11101 11100 10010 01001.

I. **Frequency test:** n_0=84, n_1=76, and the value of the statistic X_1 is 0.4.

II. **Serial test:** n_{00}=44, n_{01}=40, n_{10}=40, n_{11}=35, expected value is E=39.75, and the value of the statistic X_2 is 1.025.

III. **Poker test:** Here m=3. The blocks #"000"=5, #("001"+"010"+ "001")=28, #("011"+"110"+"101")=12, #"111"=7, expected values are E_0=6.667, E_1=20.001, E_2=20.001, E_3=6.667 and the value of the statistic X_3 is 6.834.

IV. **Runs test:** Here E_1=20.25, E_2=10.0625, E_3=5, and k=3. There are 25, 4, 5 blocks of lengths 1, 2, 3, respectively, and 8, 20, 12 gaps of lengths 1, 2, 3, respectively. The value of the statistic X_4 is 31.7913.

V. **Autocorrelation test:** If τ=3, $n_0(3)$=80 and $n_1(3)$=77. The value of the statistic X_5 is 0.115.

For a significance level of α=0.05, the threshold values for X_1, X_2, X_3, X_4, and X_5 are 3.8415, 7.8415, 7.8415, 31.787, and 0.115, respectively. Hence, the given sequence S passes the frequency, serial, poker and autocorrelation tests, but fails the runs test.

3.13.5 CRYPT-X Package of Randomness Tests

Designer and users of encryption algorithms used in cipher systems need a systematic approach in examining their cipher prior to use, to ensure that they are safe from cryptanalytic attack. In this manner we will introduce a new package of randomness instead of the mentioned five tests.

CRYPT-X [Gus.94] is a microcomputer package that is intended to be used to test either large binary strings that are to be used as keystream in stream ciphers or block cipher algorithms.

3.13.5.1 Statistical Tests Applied

The package uses a number of statistical distributions including the standard normal and the chi-squared distribution [Bha.77].

The **standard normal distribution** is used to compare a sample measure obtained from the cipher with expected measure of hypothesized distribution. The test statistic for the standard normal distribution is $z=(x-\mu)/\sigma$, where x is the sample measure and μ and σ are the expected measure and variance of the hypothesized distribution.

The **chi-squared distribution** is used to compare the goodness-of-fit of the observed frequencies of a sample measure to the corresponding expected frequencies of the hypothesized distribution.

The test statistic is:

$$\chi^2=\sum_{i=1}^{n}\frac{(o_i-e_i)^2}{e_i} \qquad \ldots(3.13)$$

Where o_i and e_i are the respective observed and expected frequencies of all possibilities of the measure. This result in the number of degree of freedom ν being one less than the resulting number of e_i values obtained i.e. $\nu=n-1$.

Algorithms calculating the tail-area probabilities of the standard normal and chi-squared distributions have been incorporated into the package. If the tail-area probability α of the statistic is extremely small ($\alpha<0.001$ for large sample)

then it would be interpreted that the cipher does not satisfy the test applied [Gus.94].

3.13.5.2 Stream Cipher Tests

The security of a stream cipher depends on the keystream appearing random. In most cases the keystream is formed by a deterministic generator which produces a periodic sequence. If the stream deviated significantly from randomness in some fashion a cryptanalyst may be able to use some the decrease in entropy caused by this deviation. The method applied in this package examines the hypothesis that the string was based on Bernoulli trials [Bha.77], for which

$$\Pr(z(t)=1) = \Pr(z(t)=0) = \frac{1}{2} \qquad \ldots(3.14)$$

Tests employed in the package to examine this hypothesis are the **Frequency** tests on the original stream and the 1st and 2nd **Binary Derivative** stream, **Change Point** test, **Subblock** (**Poker**) test and **Run** test. The details of these tests are as follows:

1. **Frequency Test**: the aim of this test to determine how the proportion of ones in the sample stream of length n bits fit into the hypothesized distribution where the proportion of ones is 0.5. Using n_1 as the number of ones, the standard normal test statistic is:

$$z = 2\sqrt{n}\left(\frac{n_1}{n} - 0.5\right) \qquad \ldots(3.15)$$

2. **Binary Derivative Test**: as described in [Car.88] is a new stream found by the modulo-two addition of successive bits in the stream. In forming the 1st binary derivative we are looking at the overlapping 2-tuples 00, 01, 10, 11 in the original stream. The proportion of ones in the 1st binary

derivative gives the proportion of the total of 01 and 10 patterns in the original stream.

3. **Change Point**: at each bit position t in the stream the proportion of ones to the point is compared to the proportion of ones in the remaining stream. The bit where the maximum change occurs is called the change point. This test determines whether this change is significant for a binomial distribution where the proportion of ones in the stream is expected to be 0.5 [Pet.79].

4. **Subblock Test: (poker)** its partitions the stream into F hands of length m bits. For a stream of size n, where $F/2^m \geq 5$, the total number of hands is $\lfloor n/m \rfloor$, where $\lfloor \ \rfloor$ denotes the integer value. The aim of this test is to show that there is an equal number of each of 2^m possible hands. If f_i denotes the frequency of hand pattern i, then the test statistic used is [Bra.88]:

$$\chi^2 = \left(\frac{2^m}{F}\right) \sum_{i=0}^{2^m-1} f_i^{\,2} - F \qquad \ldots(3.16)$$

This compared with chi-squared distribution with degree of freedom equal to 2^m-1.

5. **Runs Test**: this test counts the number of runs of ones (blocks) and runs of zeros (gaps) for each possible run length. For random data there should be an equal number of blocks and gaps. The expected number of blocks (gaps) of length i is:

$$e_i = \frac{\frac{n-i-1}{2} + 2}{2^{i+1}} \qquad \ldots(3.17)$$

A chi-squared test is performed on the bit stream to test for the goodness-of-fit of the number of blocks and gaps to this distribution. The chi-squared statistic is calculated as:

$$\chi^2=\sum_{i=1}^{k}\frac{(B_i-e_i)^2}{e_i}+\sum_{i=1}^{k}\frac{(G_i-e_i)^2}{e_i} \qquad \ldots(3.18)$$

Where B_i is the number of blocks of length i, G_i is the number of blocks of length i, and Σ denotes the summation over all possible run k of length i such that $e_i \geq 5$. This is compared with 2k-2 degree of freedom.

3.14 Correlation Immunity (CI) [Sie.84, Sie.85, Che.91, Cla.96]

Remark (3.5): (**correlation attacks**) Suppose that n maximum-length LFSRs $R_1,R_2,...,R_n$ of lengths r_1, r_2,..., r_n are employed in a nonlinear combination generator. If the connection polynomials of the LFSRs and the combining function f are public knowledge, then the number of different keys of the generator is $\prod_{i=1}^{n}(2^{r_i}-1)$.(A key consists of the initial states of the LFSRs) Suppose that there is a correlation between the keystream and the output sequence of R_1, with correlation probability Pr>1/2. If a sufficiently long segment of the keystream is known (e.g., as is possible under a known-plaintext attack on a binary additive stream cipher), the initial state of R_1 can be deduced by counting the number of coincidences between the keystream and all possible shifts of the output sequence of R_1, until this number agrees with the correlation probability Pr. Under these conditions, finding the initial state of R_1 will take at most $2^{r_i}-1$ trials. In the case where there is a correlation between the keystream and the output sequences of each of $R_1,R_2,...,R_n$, the (secret) initial state of each LFSR can be determined independently in a total of about $\prod_{i=1}^{n}(2^{r_i}-1)$ trials; this number is far smaller than the total number of different keys.

In a similar manner, correlations between the output sequences of particular subsets of the LFSRs and the keystream can be exploited.

In view of Remark (3.5), the combining function f should be carefully selected so that there is no statistical dependence between any small subset of the n LFSR sequences and the keystream. This condition can be satisfied if *f* is chosen to be m^{th}-order correlation immune.

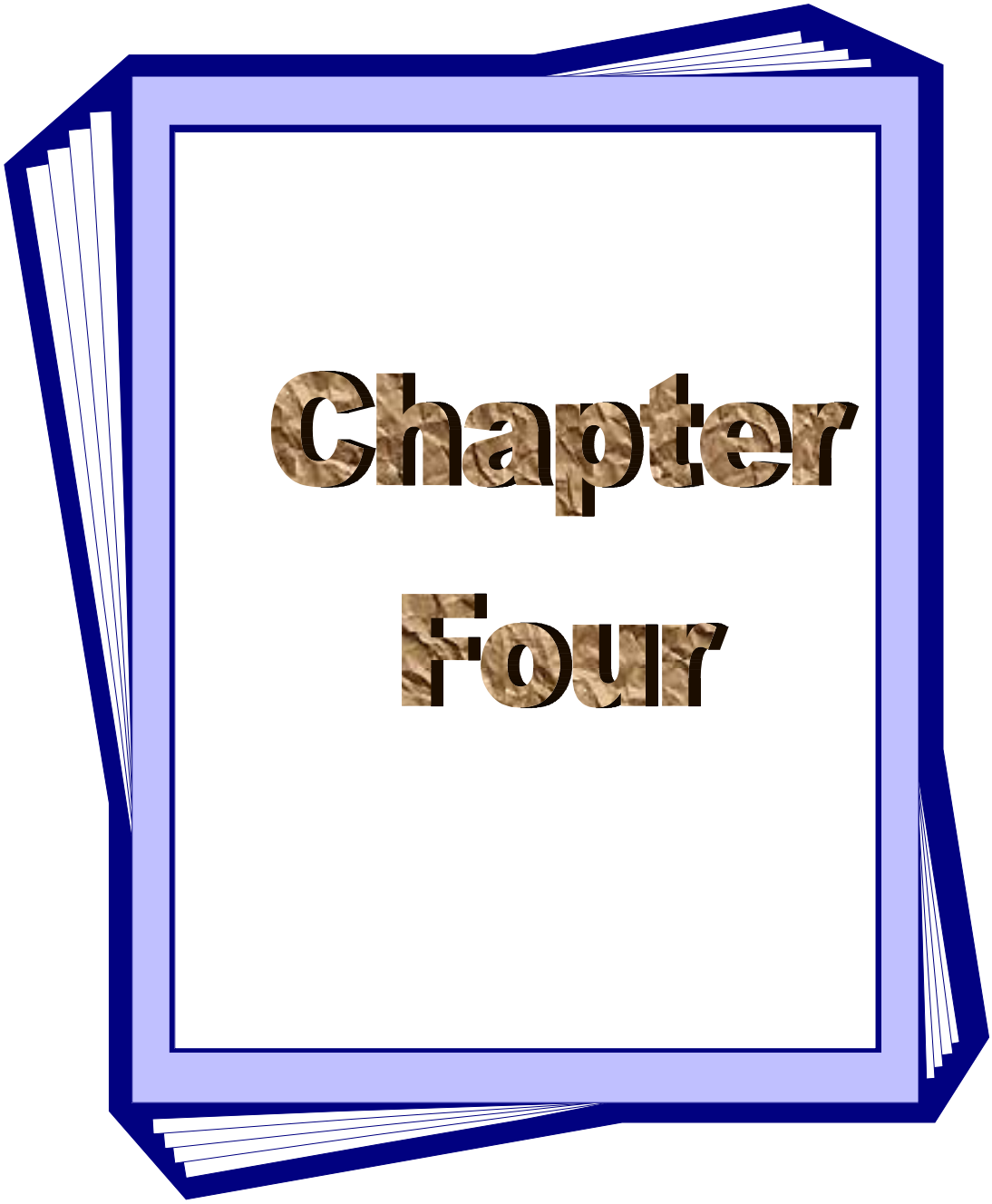
Definition (3.25) Let $X_1,X_2,...,X_n$ be independent binary variables, each taking on the values 0 or 1 with probability 1/2. A Boolean function $f(x_1,x_2,...,x_n)$ is m^{th}-order correlation immune if for each subset of m random variables $X_{i_1}, X_{i_2}, ..., X_{i_m}$ with $1 \le i_1 < i_2 < \ldots < i_m \le n$, the random variable $Z = f(X_1,X_2,...,X_n)$ is statistically independent of the random vector $(X_{i_1}, X_{i_2}, ..., X_{i_m})$; equivalently, $I(Z; X_{i_1}, X_{i_2}, ..., X_{i_m})=0$.

Example (3.6): the function $f(x_1,x_2,...,x_n)= x_1 \oplus x_2 \oplus \ldots \oplus x_n$ is $(n-1)^{th}$ order correlation immune.

In light of (3.6), the following shows that there is a tradeoff between achieving high linear complexity and high correlation immunity with a combining function.

Remark (3.6): If a Boolean function $f(x_1,x_2,...,x_n)$ is m^{th}-order correlation immune, where $1 \le m < n$, then the nonlinear order of *f* is at most $n-m$. Moreover, if *f* is balanced (i.e., exactly half of the output values of *f* are 0) then the nonlinear order of *f* is at most $n-m-1$ for $1 \le m \le n-2$.

The tradeoff between high linear complexity and high correlation immunity can be avoided by permitting memory in the nonlinear combination function *f*. This point is illustrated by the summation generator.



Objective of Dissertation
&
Dissertation Outlines

CHAPTER FOUR

OBJECTIVE OF DISSERTATION & OUTLINES

4.1 Objective of Dissertation

The practical goals of this Dissertation, divided into two main practical parts:

1. The First part is a **defense side**. Two stream cipher systems are proposed, these systems are act as a new keygenerators based on LFSR's with high order non-linear combining functions. These two generators have good statistical properties, these generators are:
   - Modified Geffe Generator.
   - 5-Threshold Generator.
2. The second part is **attack side**. The two new proposed stream ciphers are analyzed to estimate the initial values of the combined LFSR's in the two cryptosystems by adopting the application of numerical solution on them. This done by solving the linear equations systems of the generated sequences from the two generators. This part focuses on achieve a basic and important goal, which is represented by the cryptanalysis of cryptosystems.

The Dissertation results obtained with implementation in Delphi version 7.0 visual programming language exploiting the object oriented tools of this language in solving this kind of problems.

4.2 Dissertation Outlines

The solution of our problem has been covered through five chapters, the first two chapters are considered to provide the main concepts of this thesis, while the proposed new stream cipher systems and the attack of the proposed systems are introduced in chapter five.

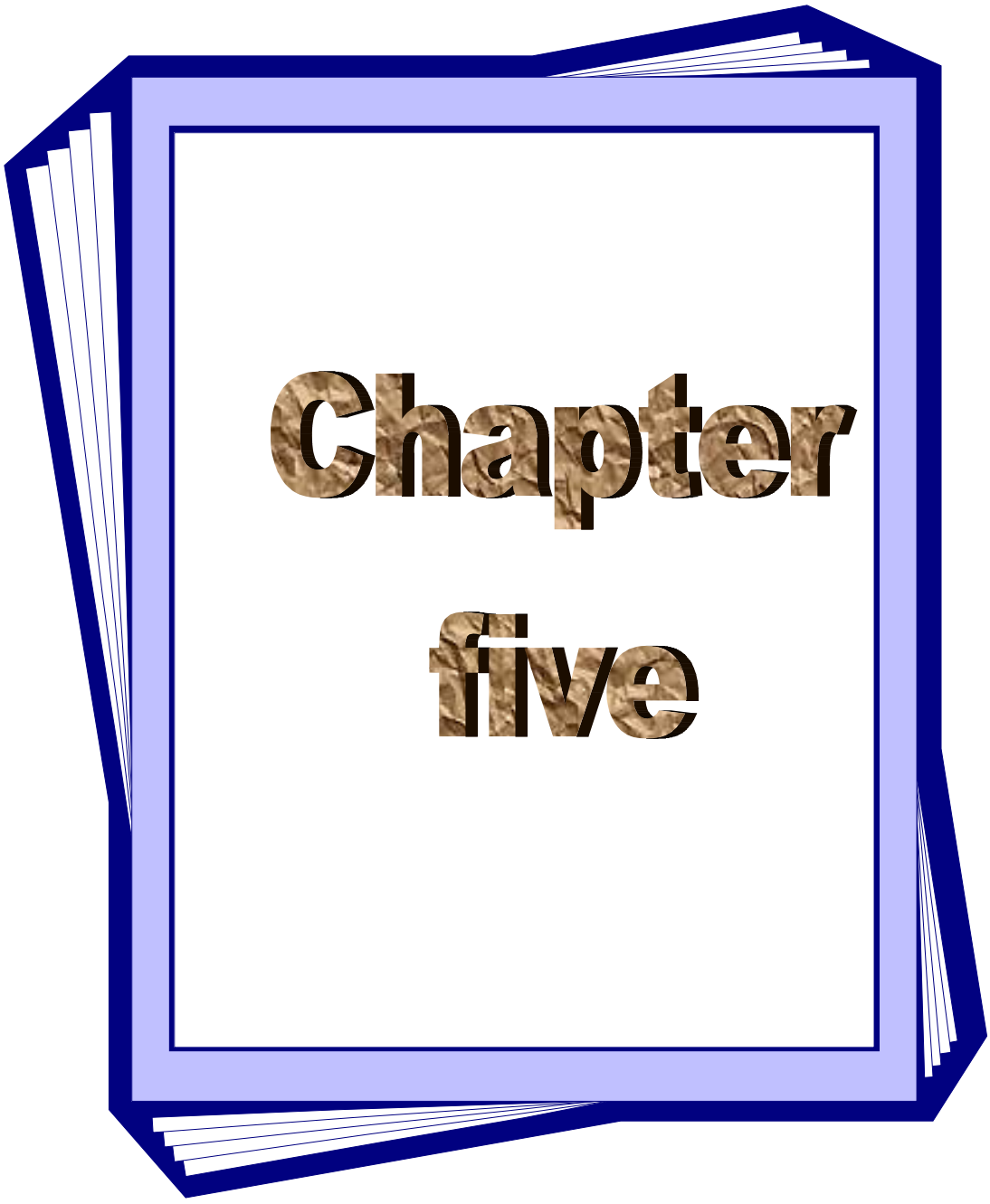
Chapter three: This chapter introduces the basic concepts in different fields in mathematics, which the cryptography and cryptanalysis are needed, specially, in stream cipher systems. This chapter shows the importance of cryptology and its classification, then showing some classical and modern cryptanalysis methods. Lastly, we will illustrate the types of attack introduces the basic efficient criteria (periodicity, linear complexity, randomness and correlation immunity) concept and the basic mathematical background of them.

Chapter four: This chapter illustrates the main objective of this these and the basic outlines.

Chapter Five: in this chapter, two new proposed stream cipher systems are introduced, and we attempts to attack the new cryptosystems which are proposed in chapter four by using known plaintext.

Chapter six: this chapter includes the main discussion of this thesis by applying the basic efficient criteria on these proposed cryptosystems to proof the efficiency of these cryptosystems, then solving the linear systems of each cryptosystem to complete the attacking of the same cryptosystems.

Chapter seven: finally, this chapter presents the conclusions as well as some recommendations.



Design & Analysis of New Stream Cipher Generators

CHAPTER FIVE

DESIGN & ANALYSIS OF NEW STREAM CIPHER GENERATORS

5.1 Introduction

LFSR and Combining Function (CF) are considered as basic units to construct key generator that used in stream cipher systems [Sch.96].

In this Dissertation the developing of Geffe generator will introduced, the developing done by increasing LFSR's from (3) to (5) with new combining function which has good statistical properties. The suggested generator called Modified Geffe generator.

Another generated will be shown, the threshold generator, which will be modified to be consists from (5) LFR's with high complexity and balance CF, the proposed generator called 5-Threshold generator.

In this chapter we will discuss the details and the basic efficiency criteria of the two proposed generators.

Let the Key Generator (KG) consist of n LFSR's have lengths $r_1,r_2,..,r_n$ respectively with $CF=F_n(x_1,x_2,\ldots,x_n)$, s.t. $x_i \in \{0,1\}$ $1 \le i \le n$, represents the output of $LFSR_i$, let $S=\{s_0,s_1,\ldots\}$ be the sequence product from KG and s_j, $j=0,1,\ldots$ represents elements of S. let S_i be the sequence i product from $LFSR_i$ with a_{ij} elements $1 \le i \le n$, $j=0,1,\ldots,$.

5.2 Basic Efficiency Criteria (BEC)

In this section we will summarized the basic efficiency criteria which are detailed in chapter two.

(a). Periodicity

The sequence S has period P(S) when $s_0=s_{P(S)}, s_1=s_{P(S)+1}, \ldots$, the period of $LFSR_i$ denotes by $P(S_i)$, P(S) and $P(S_i)$ are least possible positive integers (see equation (3.2)).

The period of S which product from key generator depends on the LFSR unit only and there is no effect of CF unit.

P(S) will has lower bound when $r=r_i \ \forall 1\leq i\leq n$, and upper bound when $P(S_i)$ are relatively prime with each other therefore $P(S_r)\leq P(S)\leq \prod_{i=1}^{n} P(S_i)$.

The objective is that key generator must have an upper bound to P(S) (as in equation (3.3)).

It's known earlier that $P(S_i) \leq 2^{r_i}-1$, and if the $LFSR_i$ has maximum period then $P(S_i)= 2^{r_i}-1$ [Gol.82].

Theorem (3.1) [AlS.09]

$$P(S)=\prod_{i=1}^{n}(2^{r_i}-1) \qquad \ldots(5.1)$$

if and only if the following conditions are holds:

1. $GCD_n(P(S_i))=1$,.
2. the period of each LFSR has maximum period ($P(S_i)= 2^{r_i}-1$).

(b). Randomness

For our purposes, a sequence generator is pseudo-random if it has this property: It looks random. This means that it passes all the statistical tests of randomness that we can find [Sch.97].

The sequence that is satisfied the 3-randomness properties called PRS [Gol.82]. The randomness criterion depends on LFSR's and CF units, therefore from the important conditions to get Pseudo Random Sequence is, the sequence must be maximal and CF must be balance.

(c). Linear Complexity

The Linear Complexity is defined as the length, of the shortest LFSR (which is equivalent LFSR) that can mimic the generator output. Any sequence generated by a finite-state machine over a finite field has a finite linear complexity [Mas.69].

(d). Correlation Immunity

Correlation can be defined as the relation between the sequence of CF=F_n from the key generator and the sequences that are combined each other by CF. This relation caused because of the non-linearity of the function F_n. The correlation probability CP(x), in general, represents the ratio between the numbers of similar binaries of two sequences to the length of the compared part of them. F_n has m^{th} order CI, if the output z of F_n is statistically independent from m output from m-sequences $(x_1,x_2,...,x_m)$, of n combined sequences s.t. $m \leq n$.

5.3 Geffe Generator

This generator introduced by Geffe in 1973 [Gef.73]. Geffe generator defined by three maximum-length LFSRs whose lengths r_1, r_2, r_3 are pair wise relatively prime, with nonlinear combining function:

$$GF_3(x_1,x_2,x_3) = x_1*x_2 \oplus (1 \oplus x_2)*x_3 = x_1*x_2 \oplus x_2*x_3 \oplus x_3 \qquad \ldots(5.2)$$

The middle LFSR in the combining function of Geffe generator acts as a traffic policeman, such that if the output of LFSR2 "0" then the output of LFSR1 take apart in the output sequence; else the output of LFSR3 will contribute in the output sequence.

Figure (5.1) shows the block diagram of Geffe generator.

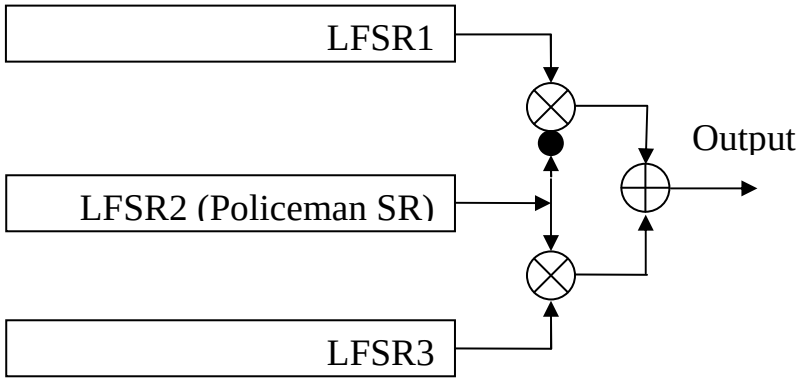


Figure (5.1) Geffe generator [Gef.73].

The keystream generated has period $(2^{r_1}-1)(2^{r_2}-1)(2^{r_3}-1)$ and linear complexity $LC=r_1r_2+r_2r_3+r_3$. The Geffe generator is cryptographically weak because information about the states of LFSR1 and LFSR3 leaks into the output sequence.

The truth table of non-linear combining function $GF_3(x_1,x_2,x_3)$ is shown in table (5.1).

Table (5.1) Truth table of GF_3 for Geffe generator.

x_1	x_2	x_3	GF_3
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0

1	1	1	1
0.75	0.5	0.75	0.5
Correlation Probability (CP_i) for each LFSR			Ratio of "0"

Note: the shaded cells indicate the similarity between x_i and the output of GF_3.

From table (5.1) we can conclude that this generator will generate sequences with good randomness properties.

Despite having high period and moderately high linear complexity, the Geffe generator succumbs to correlation attacks [Sch.97].

5.4 Modified Geffe Generator

Know we would improve Geffe generator by choosing 5 LFSR's instead of 3 LFSR's, if the output of LFSR3 is "0" then we choose the xoring of LFSR1 and LFSR2, otherwise we choose the xoring of LFSR4 and LFSR5. The CF of this generator is:

$$GF_5(x_1,x_2,x_3,x_4,x_5)=(x_1\oplus x_2)*(x_3\oplus 1)\oplus x_3*(x_4\oplus x_5) \quad \ldots(5.3\text{-a})$$

Or it can be written as follows:

$$GF_5(x_1,x_2,x_3,x_4,x_5)=x_1\oplus x_2\oplus x_1x_3\oplus x_2x_3\oplus x_3x_4\oplus x_3x_5 \quad \ldots(5.3\text{-b})$$

so the proposed generator called **Modified-Geffe generator**.

Figure (5.2) shows the block diagram of Modified Geffe Generator.

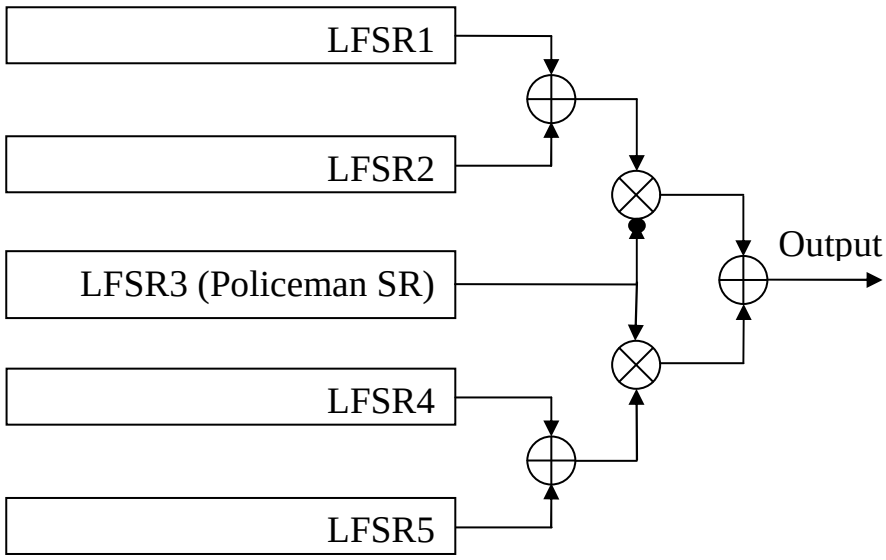


Figure (5.2) Block diagram Modified Geffe generator.

5.5 Threshold Generator

This generator introduced by Brüer in 1983 [Brü.83]. Threshold generator defined by three maximum-length LFSRs whose lengths r_1, r_2, r_3 are pair wise relatively prime, with nonlinear combining function:

$$TF_3(x_1,x_2,x_3) = x_1*x_2 \oplus x_1*x_3 \oplus x_2*x_3 \quad \ldots(5.4)$$

This generator depends on function called majority function. This function sums (ordinary sum) the outputs of the three LFSR's then divide by (2); if the result greater than (3/2) the output is "1"; else the output is "0" to form the output sequence.

The truth table of non-linear combining function $TF_3(x_1,x_2,x_3)$ is shown in table (5.2).

Table (5.2) Truth table of TF_3 for Threshold generator.

x_1	x_2	x_3	TF_3
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0.75	0.75	0.75	0.5
Correlation Probability (CP_i) for each LFSR			Ratio of "0"

Figure (5.3) shows the block diagram of Threshold generator.

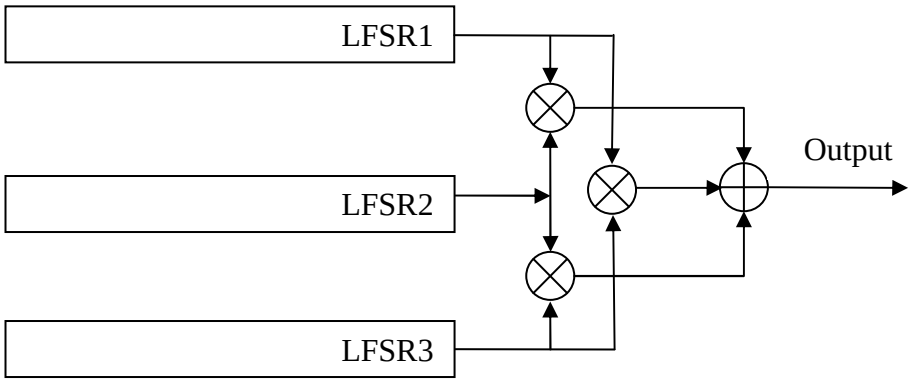


Figure (5.3) Threshold generator [Brü.83].

The keystream generated has period $(2^{r_1}-1)(2^{r_2}-1)(2^{r_3}-1)$ and linear complexity $LC=r_1r_2+r_1r_3+r_2r_3$. The Threshold generator is cryptographically weak because information about the states of LFSR1, LFSR2 and LFSR3 leaks into the output sequence.

From table (3.2) we can conclude that this generator will generate sequences with good randomness properties.

Despite having high period and moderately high linear complexity, the Threshold generator succumbs to correlation attacks [Sch.97].

5.6 Modified Threshold Generator

In this Dissertation, we propose to use (5) LFSR's instead of (3), where the CF of this generator is:

$$TF_5(x_1,x_2,x_3,x_4,x_5)=x_1x_2x_3+x_1x_2x_4+x_1x_2x_5+x_1x_3x_4+x_1x_4x_5+x_2x_3x_4+x_2x_3x_5+ x_1x_2x_3+x_2x_4x_5+x_3x_4x_5+x_1x_2x_3x_4+x_1x_2x_3x_5+x_1x_2x_4x_5+ x_1x_3x_4x_5+x_2x_3x_4x_5 \quad \ldots(5.5)$$

so this system is called **modified 5-Threshold System**.

Figure (5.4) shows the block diagram of Modified 5-Threshold Generator.

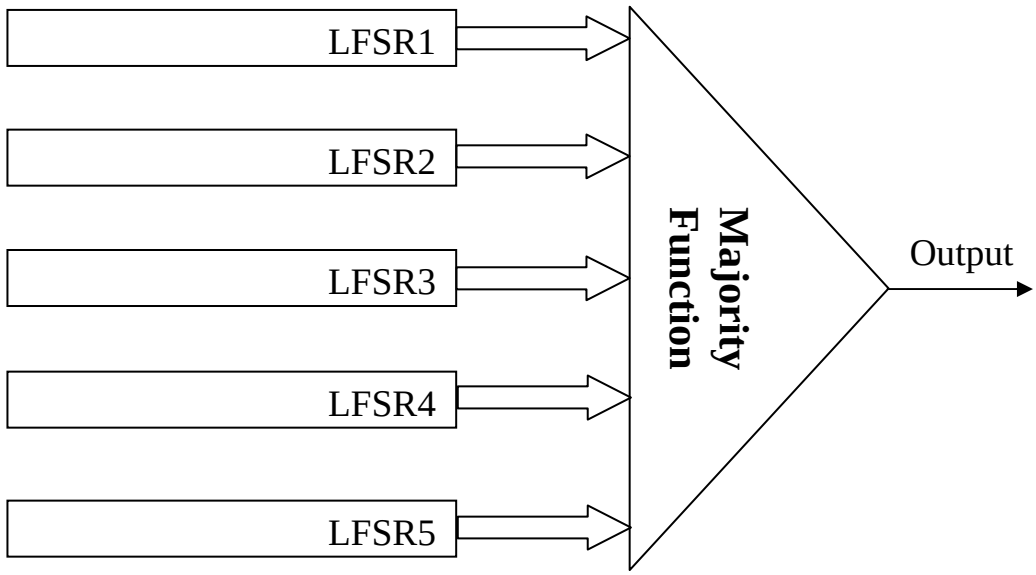


Figure (5.4) Block diagram Modified 5-Threshold generator.

5.7 Attacking the Proposed Cryptosystems

Golomb used the recurrence relation to find the next state values of single LFSR depending on initial values, s.t. he can be considered the first who can construct a linear equations system of a single LFSR. Attacking of key generator means attempt to find the initial values of the combined LFSR's.

This Dissertation introducing the analysis of the two new generators which are proposed in sectiones 5.5 and 5.6; The Modified Geffe generator and Modified Threshold generator. The analysis process divided into three stages. First, a Golomb's method introduced to construct a linear equations system of a single LFSR. Secondly, this method developed to construct a linear equations system of key generator (a LFSR system) where the effect of combining function of LFSR is obvious. Lastly, before solving the linear equations system, the existence and the uniqueness of the solution must be tested, then solving the linear equations system using one of the classical methods like Gauss Elimination. Find the solution of linear equations system means find the initial values of the generator.

The LFSR System (LFSRS) consists of two main basic units, the feedback function and initial state values [Sch.97]. The second one is, the Combining Function (CF), which is a Boolean function [Whi.05].

Most of all Stream Cipher System's are depending on these two basic units.

This work aims to find the initial values of every LFSR in the system depending on the following information:

1. The length of every LFSR and its feedback function are known.
2. The CF is known.
3. The output sequence S (keystream) generated from the LFSRS is known, or part of it, practically, that means, a probable word attack be applied [Sch.97].

This work consists of three stages, constructing system of linear equations, testing the existence and the uniqueness of the solution of this system, and lastly, solving the system of linear equations.

5.8 Constructing System of Linear Equations (SLE's)

5.8.1 Single LFSR

Before involving in solving the System of Linear Equations (SLE's), it should show how could be the SLE's of a single LFSR constructed, since it's considered a basic unit of LFSRS. Let's assume that all LFSR that are used are maximum LFSR, that means, Period $P(S)=2^r-1$, where r is LFSR length.

Let SR_r be a single LFSR with length r, let $A_0=(a_{-1},a_{-2},\ldots,a_{-r})$ be the initial value vector of SR_r, s.t. a_{-j}, $1\leq j\leq r$, be the component j of the vector A_0, in another word, a_{-j} is the initial bit of stage j of SR_r, let $C_0^T=(c_1,\ldots,c_r)$ be the feedback vector, $c_j\in\{0,1\}$, if $c_j=1$ that means the stage j is connected. Let $S=\{s_i\}_{i=0}^{m-1}$ be the sequence (or $S=(s_0,s_1,\ldots,s_{m-1})$ read "S vector") with length m generated from SR_r. The generating of S depending on the following equation [Gol.82]:

$$s_i = a_i = \sum_{j=1}^{r} a_{i-j} c_j \quad i=0,1,\ldots \qquad \ldots(5.5)$$

Equation (5.5) represents the linear recurrence relation.

The objective is finding the A_0, when r, C_0 and S are known.

Let M be a r×r matrix called the generating matrix or characteristic matrix, which is describes the initial phase of SR_r, where the 1st column is the vector C_0 and the rest columns are the Identity matrix I without the last column.

$M=(C_0|I_{r\times r-1})$, where $M^0=I$.

Let A_1 represents the new initial state of SR_r after one shift, s.t.

$$A_1=A_0\times M=(a_{-1},a_{-2},\ldots,a_{-r})\begin{pmatrix} c_1 & 1 & \cdots & 0 \\ c_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ c_r & 0 & \cdots & 0 \end{pmatrix} = (\sum_{j=1}^{r} a_{-j} c_j, a_{-1},\ldots,a_{1-r}).$$

In general,

$$A_i=A_{i-1}\times M,\ i=1,2,\ldots \qquad \ldots(5.6)$$

A_i represents the initial state of SR_r after i shifts.

Equation (5.6) can be considered as a recurrence relation, so we have:

$$A_i=A_{i-1}\times M=A_{i-2}\times M^2=\ldots=A_0\times M^i \qquad \ldots(5.7)$$

The matrix M^i represents the i phase of SR_r, equations (5.6,7) can be considered as a Markov Process s.t., A_0, is the initial probability distribution, A_i represents probability distribution and M be the transition matrix [Pap.01].

Notice that:

$M^2=[C_1C_0|I_{r\times r-2}]$ and so on until get $M^i=[C_{i-1}\ldots C_0|I_{r\times r-i}]$, where $1\le i<r$.

When $C_P=C_0$ then $M^{P+1}=M$.

Now let's calculate C_i s.t.

$$C_i=M\times C_{i-1},\ i=1,2,\ldots \qquad \ldots(5.8)$$

Where C_i is the feedback vector after i shifts.

Equation (5.5) can be rewritten as:

$$A_0\times C_i=s_i\ ,\ i=0,1,..,r-1 \qquad \ldots(5.9)$$

When i=0 then $A_0\times C_0=s_0$ is the 1^{st} equation of the SLE's,

i=1 then $A_0\times C_1=s_1$ is the 2^{nd} equation of the SLE's, and

i=r-1 then $A_0\times C_{r-1}=s_{r-1}$ is the r^{th} equation of the SLE's.

In general:

$$A_0\times \mathbf{C}=S \qquad \ldots(5.10)$$

C represents the matrix of all C_i vectors s.t.

$$\mathbf{C}=(C_0C_1\ldots C_{r-1}) \qquad \ldots(5.11)$$

The SLE can be formulated as:

$$Y=[\ \mathbf{C}^T|S^T] \qquad \ldots(5.12)$$

Y represents the extended matrix of the SLE.

Example (5.1)

Let the SR_4 has $C_0^T=(0,0,1,1)$ and S=(1,0,0,1), by using equation (5.8), we get:

$$C_1=M\times C_0=\begin{pmatrix}0&1&0&0\\0&0&1&0\\1&0&0&1\\1&0&0&0\end{pmatrix}\begin{pmatrix}0\\0\\1\\1\end{pmatrix}=\begin{pmatrix}0\\1\\1\\0\end{pmatrix}$$, in the same way, $C_2=\begin{pmatrix}1\\1\\0\\0\end{pmatrix}$,$C_3=\begin{pmatrix}1\\0\\1\\1\end{pmatrix}$

From equation (4.6) we have:

$A_0\begin{pmatrix}0&0&1&1\\0&1&1&0\\1&1&0&1\\1&0&0&1\end{pmatrix}$=(1,0,0,1), this system can be written as equations:

s.t. A_0 is the initial state vector of SR_r,

$a_{-3}+a_{-4}=1$

$a_{-2}+a_{-3}=0$

$a_{-1}+a_{-2}=0$

$a_{-1}+a_{-3}+a_{-4}=1$

(for simplicity the sign (-) can be omitted).

Then the SLE's after using formula (4.8) is:

$$Y=\left[\begin{array}{cccc|c}0&0&1&1&1\\0&1&1&0&0\\1&1&0&0&0\\1&0&1&1&1\end{array}\right] \qquad \ldots(5.13)$$

5.8.2 Constructing A SLE's for Modified Geffe Generator

In general for any generator, let's have n of SR_{r_j} with length r_j, j=1,2,…,n, with feedback vector $C_{0j}=\begin{pmatrix}c_{01j}\\c_{02j}\\\\c_{0r_jj}\end{pmatrix}$,and has unknown initial value vector $A_{0j}=(a_{-1j},\ldots,a_{-rjj})$, so SR_{r_j} has $M_j=(C_{0j}|I_{r_j\times r_j-1})$

By using recurrence equation (5.8),

$$C_{ij}=M_j\times C_{i-1,j},\ i=1,2,\ldots \qquad \ldots(5.14)$$

by using equation (5.9):

$A_{0j}\times C_{ij}=s_{ij}$, i=0,1,…,r-1 and $S_j=(s_{0j},s_{1j},\ldots,s_{m-1,j})$.

S_j represents the output vector of SR_{r_j}, which of course, is unknown too. m represents the number of variables produced from the LFSR's with consider to CF, in the same time its represents the number of equations which are be needed to solve the SLE's. Of course, there is n of SLE's (one SLE's for each SR_{r_j} with unknown absolute values).

Now, let A_0 be the extended vector to (m) variables, which consists of initial values from all LFSR's and **C** is the matrix of C_i vectors considering the CF, C_i represents the extended vector of all feedback vectors C_{ij}, then $A_0 \times \mathbf{C} = S$.

From CF, the number of new variables are:

$m = r_1+r_2+r_1r_3+r_2r_3+r_3r_4+r_3r_5$.

The initial value is:

$A_0 = A_{01}+A_{02}+A_{01}A_{03}+A_{02}A_{03}+A_{03}A_{04}+A_{03}A_{05} = (x_0, x_1, \ldots, x_{m-1})$,

s.t. $x_0 = a_{-11}$, $x_1 = a_{-21}, \ldots, x_{m-1} = a_{-r_4 4} a_{-r_5 5}$

(this arrangement is not standard so it can be changed according to the researcher requirements or to the way how can know the number and the form of the multiplying variables).

For simplicity let's denote the unknowns of $LFSR_1$ by 'a', $LFSR_2$ by 'b', and so on, let's denote the unknowns of SR_5 by 'e', therefore:

$$x_0 = a_1,\ x_1 = a_2, \ldots, x_{m-1} = d_{r4} e_{r5} \qquad \ldots(5.15)$$

We called the variables $a_i * b_j * \ldots * e_k$ **origin variables**, while x_j the **new variables**.

In the same way, equation (5.15) can be applied on the feedback vector C_{ij}:

$$C_i = C_{i1}+C_{i2}+C_{i1}C_{i3}+C_{i2}C_{i3}+C_{i3}C_{i4}+C_{i3}C_{i5}$$

And the sequence S will be:

$$S = S_1+S_2+S_1S_3+S_2S_3+S_3S_4+S_3S_5,$$

s.t. $s_i = s_{i1}+s_{i2}+s_{i1}s_{i3}+s_{i2}s_{i3}+s_{i3}s_{i4}+s_{i3}s_{i5}$,

where s_i is the element i of S.

So the SLE's can be obtained by equation (5.10).

Example (5.2)

Let’s have the following feedback vectors for 5 LFSR's with lengths 2,3,4,5 and 6:

$C_{01}=\begin{pmatrix}1\\1\end{pmatrix}$, $C_{02}=\begin{pmatrix}1\\0\\1\end{pmatrix}$, $C_{03}=\begin{pmatrix}1\\0\\0\\1\end{pmatrix}$, $C_{04}=\begin{pmatrix}0\\1\\0\\0\\1\end{pmatrix}$, $C_{05}=\begin{pmatrix}1\\0\\0\\0\\0\\1\end{pmatrix}$ then m=69.

Let the output sequences be:

S=(0,1,0,1,0,0,1,1,1,0,1,…,0,0,0,0,0,1,1,0,1,1,0,0),

$C_{01}=C_{31}=\ldots=C_{65,1}=C_{68,1}=\begin{pmatrix}1\\1\end{pmatrix}$,$C_{11}=C_{41}=...=C_{63,1}=C_{66,1}=\begin{pmatrix}0\\1\end{pmatrix}$

$C_{21}=C_{51}=\ldots=C_{64,1}=C_{67,1}=\begin{pmatrix}1\\0\end{pmatrix}$.

$C_{02}=\ldots=C_{63,2}=\begin{pmatrix}1\\0\\1\end{pmatrix}$,$C_{12}=\ldots=C_{69,2}=\begin{pmatrix}1\\1\\1\end{pmatrix}$,$C_{22}=\ldots=C_{64,2}=\begin{pmatrix}0\\1\\1\end{pmatrix}$,

$C_{32}=\ldots=C_{65,2}=\begin{pmatrix}1\\1\\0\end{pmatrix}$,$C_{42}=\ldots=C_{66,2}=\begin{pmatrix}0\\0\\1\end{pmatrix}$,$C_{52}=\ldots=C_{67,2}=\begin{pmatrix}0\\1\\0\end{pmatrix}$,

$C_{62}=\ldots=C_{68,2}=\begin{pmatrix}1\\0\\0\end{pmatrix}$.

$C_{03}=\ldots=C_{65,3}=\begin{pmatrix}1\\0\\0\\1\end{pmatrix}$,$C_{13}=\ldots=C_{66,3}=\begin{pmatrix}1\\0\\1\\1\end{pmatrix}$,$C_{23}=\ldots=C_{67,3}=\begin{pmatrix}1\\1\\1\\1\end{pmatrix}$,

$C_{33}=\ldots=C_{68,3}=\begin{pmatrix}0\\1\\1\\1\end{pmatrix}$,

And so on until we get:

$C_{12,3}=\ldots.=C_{57,3}=\begin{pmatrix}0\\0\\1\\0\end{pmatrix}$, $C_{13,3}=\ldots=C_{58,3}=\begin{pmatrix}0\\1\\0\\0\end{pmatrix}$,$C_{14,3}=\ldots=C_{59,3}=\begin{pmatrix}1\\0\\0\\0\end{pmatrix}$.

In the same process we get C_{i4} and C_{i5}, for i=1,2,…,69.

by applying equation (5.8), C_0^T will be:

C_0^T=(1,1,1,0,1,1,0,0,1,0,0,0,0,1,0,0,1,…,0,0,0,0,0,0,1,0,0,0,0,1).

Therefore,

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

1 1 1 0 1 1 0 0…0 0 1 0 0 0 0 1 | 0
1 0 0 1 0 1 0 0…1 1 1 0 0 0 1 1 | 1

Y = .

…(5.16)

.

1 0 0 1 1 1 1 1…0 0 1 1 1 1 1 1 | 0
0 1 1 1 1 0 0 0…1 1 0 1 1 1 1 1 | 0

5.8.3 Constructing A SLE's for Modified Threshold Generator

For the Modified Threshold system, let A_0 be the extended vector for m variables, which consists of initial values from all LFSR's and **C** is the matrix of C_i vectors considering the CF, C_i represents the extended vector of all feedback vectors C_{ij}, then $A_0 \times \mathbf{C}=S$.

From CF the number of variables (m) are:

$m=r_1r_2r_3+r_1r_2r_4+r_1r_2r_5+r_1r_3r_4+r_1r_4r_5+r_2r_3r_4+r_2r_3r_5+r_1r_2r_3+r_2r_4r_5+r_3r_4r_5+r_1r_2r_3r_4$

$+r_1r_2r_3r_5+r_1r_2r_4r_5+r_1r_3r_4r_5+r_2r_3r_4r_5$.

The initial value is:

$$A_0=A_{01}A_{02}A_{03}+A_{01}A_{02}A_{04}+A_{01}A_{02}A_{05}+A_{01}A_{03}A_{04}+A_{01}A_{04}A_{05}+A_{02}A_{03}A_{04}+$$
$$A_{02}A_{03}A_{05}+A_{01}A_{02}A_{03}+A_{02}A_{04}A_{05}+A_{03}A_{04}A_{05}+A_{01}A_{02}A_{03}A_{04}+$$
$$A_{01}A_{02}A_{03}A_{05}+A_{01}A_{02}A_{04}A_{05}+A_{01}A_{03}A_{04}A_{05}+A_{02}A_{03}A_{04}A_{05}$$
$$=(x_0,x_1,\ldots,x_{m-1}),$$

s.t. $x_0=a_{-11}a_{-12}a_{-13}$, $x_1=a_{11}a_{12}a_{23},\ldots,x_{m-1}=a_{r_2 2}a_{r_3 3}a_{r_4 4}a_{r_5 5}$

thus

$$x_0=a_1b_1c_1,\ x_1=a_1b_1c_2,\ldots,x_{m-1}=b_{r2}c_{r3}d_{r4}e_{r5} \qquad \ldots(5.17)$$

In the same way, equation (5.17) can be applied on the feedback vector C_{ij}:

$$C_i=C_{i1}C_{i2}C_{i3}+C_{i1}C_{i2}C_{i4}+C_{i1}C_{i2}C_{i5}+C_{i1}C_{i3}C_{i4}+C_{i1}C_{i4}C_{i5}+C_{i2}C_{i3}C_{i4}+C_{i2}C_{i3}C_{i5}$$
$$+C_{i1}C_{i2}C_{i3}+C_{i2}C_{i4}C_{i5}+C_{i3}C_{i4}C_{i5}+C_{i1}C_{i2}C_{i3}C_{i4}+C_{i1}C_{i2}C_{i3}C_{i5}+C_{i1}C_{i2}C_{i4}C_{i5}$$
$$+C_{i1}C_{i3}C_{i4}C_{i5}+C_{i2}C_{i3}C_{i4}C_{i5}.$$

And the sequence S will be:

$$S=S_1S_2S_3+S_1S_2S_4+S_1S_2S_5+S_1S_3S_4+S_1S_4S_5+S_2S_3S_4+S_2S_3S_5+S_1S_2S_3+S_2S_4S_5$$
$$+S_3S_4S_5+S_1S_2S_3S_4+S_1S_2S_3S_5+S_1S_2S_4S_5+S_1S_3S_4S_5+S_2S_3S_4S_5$$

s.t.

$$s_i=s_{i1}s_{i2}s_{i3}+s_{i1}s_{i2}s_{i4}+s_{i1}s_{i2}s_{i5}+s_{i1}s_{i3}s_{i4}+s_{i1}s_{i4}s_{i5}+s_{i2}s_{i3}s_{i4}+s_{i2}s_{i3}s_{i5}+s_{i1}s_{i2}s_{i3}+s_{i2}s_{i4}s_{i5}$$
$$+s_{i3}s_{i4}s_{i5}+s_{i1}s_{i2}s_{i3}s_{i4}+s_{i1}s_{i2}s_{i3}s_{i5}+s_{i1}s_{i2}s_{i4}s_{i5}+s_{i1}s_{i3}s_{i4}s_{i5}+s_{i2}s_{i3}s_{i4}s_{i5},$$

where s_i is the element i of S.

So the SLE's can be obtained by equation (5.10).

Remark (5.1):

It's important to mention that the (m) equations which are needed to solve the system must be linearly independent in order to guarantee obtained unique solution.

Example (5.3)

Let’s have the following feedback vectors for 5 LFSR's with lengths 2,3,4,5 and 6:

$C_{01}=\begin{pmatrix}1\\1\end{pmatrix}$, $C_{02}=\begin{pmatrix}1\\0\\1\end{pmatrix}$, $C_{03}=\begin{pmatrix}1\\0\\0\\1\end{pmatrix}$, $C_{04}=\begin{pmatrix}0\\1\\0\\0\\1\end{pmatrix}$, $C_{05}=\begin{pmatrix}1\\0\\0\\0\\0\\1\end{pmatrix}$ then m=1624.

Let the output sequences be: S=(1,1,1,1,1,1,0,…,0,1,1,1,0,1,1,0,0)

$C_{0,1}=C_{3,1}=C_{6,1}=C_{9,1}=\ldots=C_{1620,1}=C_{1623,1}=\begin{pmatrix}1\\1\end{pmatrix}$

$,C_{1,1}=C_{4,1}=C_{7,1}=C_{10,1}=...=C_{1618,1}=C_{1621,1}=\begin{pmatrix}0\\1\end{pmatrix}$,

$C_{21}=C_{51}=C_{81}=C_{11,1}=\ldots=C_{1619,1}=C_{1622,1}=\begin{pmatrix}1\\0\end{pmatrix}$.

$C_{0,2}=\ldots=C_{1617,2}=\begin{pmatrix}1\\0\\1\end{pmatrix}$,$C_{1,2}=\ldots=C_{1618,2}=\begin{pmatrix}1\\1\\1\end{pmatrix}$,$C_{2,2}=\ldots=C_{1619,2}=\begin{pmatrix}0\\1\\1\end{pmatrix}$

$,C_{3,2}=\ldots=C_{1620,2}=\begin{pmatrix}1\\1\\0\end{pmatrix}$,$C_{4,2}=\ldots=C_{1621,2}=\begin{pmatrix}0\\0\\1\end{pmatrix}$,$C_{5,2}=\ldots=C_{1622,2}=\begin{pmatrix}0\\1\\0\end{pmatrix}$

$,C_{6,2}=\ldots=C_{1623,2}=\begin{pmatrix}1\\0\\0\end{pmatrix}$.

$C_{0,3}=\ldots=C_{1609,3}=\begin{pmatrix}1\\0\\0\\1\end{pmatrix}$,$C_{1,3}=\ldots=C_{1610,3}=\begin{pmatrix}1\\0\\1\\1\end{pmatrix}$,$C_{2,3}=\ldots=C_{1611,3}=\begin{pmatrix}1\\1\\1\\1\end{pmatrix}$,$C_{3,3}=\ldots$

$=C_{1612,3}=\begin{pmatrix}0\\1\\1\\1\end{pmatrix}$,

And so on until we get:

$C_{12,3}=\ldots.=C_{1621,3}=\begin{pmatrix}0\\0\\1\\0\end{pmatrix}$, $C_{13,3}=\ldots=C_{1622,3}=\begin{pmatrix}0\\1\\0\\0\end{pmatrix}$,$C_{14,3}=\ldots=C_{1623,3}=\begin{pmatrix}1\\0\\0\\0\end{pmatrix}$.

In the same process we get $C_{i,4}$ and $C_{i,5}$, for i=1,2,…,1623.

by applying equation (5.8), C_0^T will be:

C_0^T=(1,0,0,1,0,0,0,0,1,0,0,1, 1,0,0,1,0,0,0,0,1,0,0,1,…,1,0,0,0,0,1).

Therefore, then the augmented (1623×1623) matrix Y will be:

$$Y=\begin{pmatrix} 1\,0\,0\,1\,0\,0\,0\,0\ldots0\,0\,0\,0\,1\,0\,0\,1 \mid 1\\ 0\,0\,0\,0\,0\,0\,0\,0\ldots1\,1\,0\,0\,0\,0\,0\,0 \mid 1\\ \vdots\\ 0\,0\,0\,0\,1\,1\,1\,1\ldots0\,0\,0\,0\,0\,0\,0\,0 \mid 0\\ 0\,1\,1\,1\,0\,0\,0\,0\ldots0\,0\,0\,0\,0\,0\,0\,0 \mid 0 \end{pmatrix} \quad \ldots(5.18)$$

5.9 Test the Uniqueness of The Solution of SLE's

Since the system of m variables, then there are 2^m-1 equations, but only m linearly independent equations are needed to solve the system. If the system contains dependent equations, then the system has no unique solution. So first it should test the uniqueness of solution of the system by many ways like calculating the rank of the system matrix ($r(\mathbf{C}^T)$) or by finding the determinant of the matrix. If the rank equal the matrix degree ($deg(\mathbf{C}^T)$), then the system has unique solution, else ($r(\mathbf{C}^T) < deg(\mathbf{C}^T)$) the system has no unique solution.

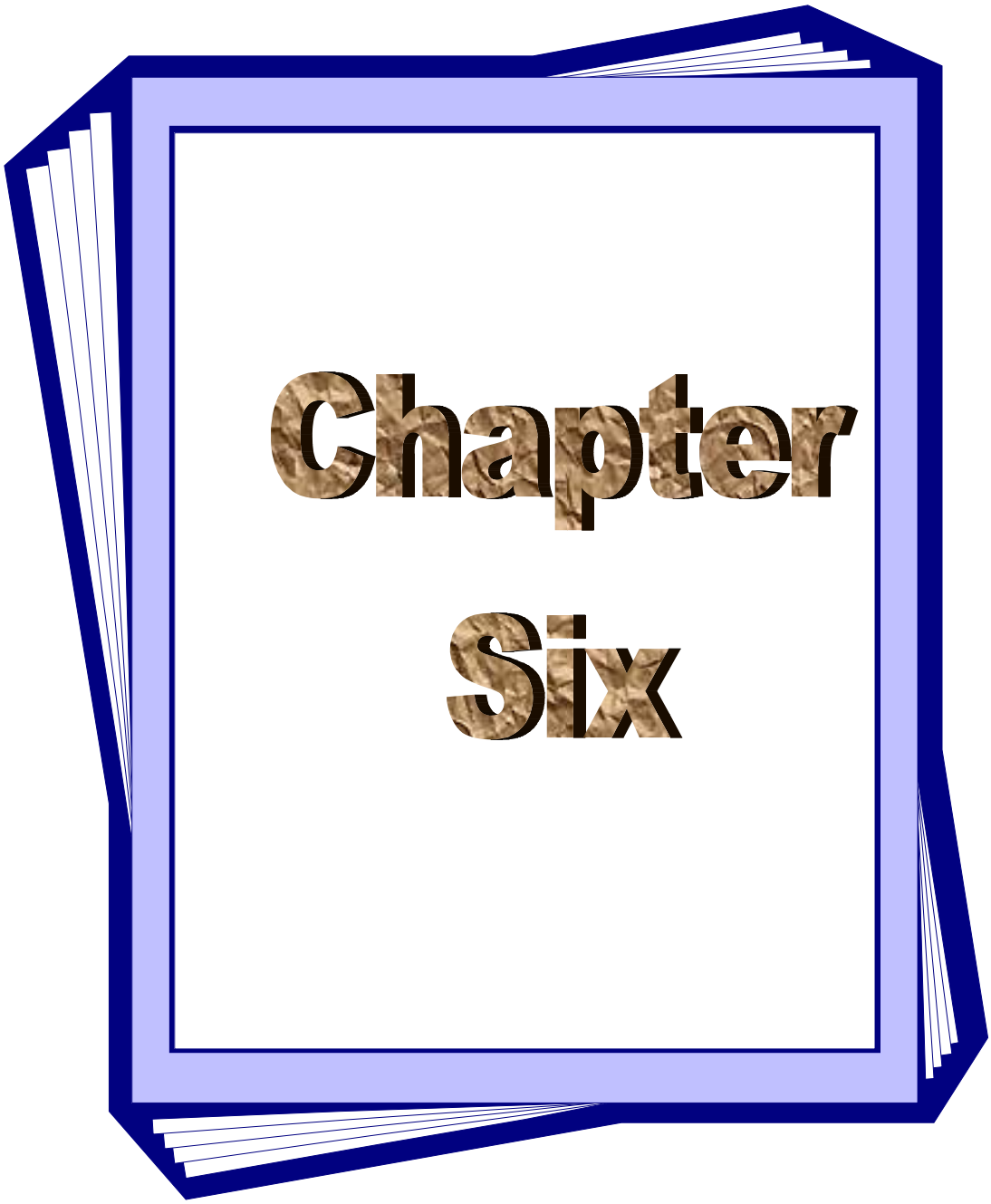
In order to calculate the $r(\mathbf{C}^T)$ it has to use the elementary operations to convert the $\mathbf{C}^T$ matrix to a simplest matrix by making, as many as possible of, the matrix elements zero's. The elementary operations should be applied in the rows and columns of the matrix $\mathbf{C}^T$, if it converts to Identity matrix then $r(\mathbf{C}^T)=deg(\mathbf{C}^T)=m$, then we can judge that $\mathbf{C}^T$ has unique solution [Jaa.05].

Example (5.4)

Let's have the matrix $\mathbf{C}^T=\begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}$, by using the elementary operations, the matrix can be converted to the matrix $\mathbf{C}^{T*}=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$,

this matrix has rank =4=$deg(\mathbf{C}^T)$ then the matrix has unique solution.

For modified Geffe (Threshold) generator, we obtain that the SLE's has unique solution; of course we have to choose 69 (1024) independent equations not all are in sequence order for the SLE's in example (5.2) (example (5.3)).



Discussion of Dissertation Results

CHAPTER SIX

DISCUSSION OF DISSERTATION RESULTS

6.1 Introduction

In this chapter, first we will discuss the details of applying the basic efficiency criteria on the two proposed generators to calculate the efficiency of each cryptosystems which are proposed in the previous chapter. Secondly, We will attack the linear equations system of each proposed cryptosystem solution suggested in chapter five. The attack represented by solving the linear equation system using Gauss Elimination method after modified it to be sautible to implemented in GF(2).

let the Key Generator (KG) consist of n LFSR's have lengths $r_1,r_2,..,r_n$ respectively with $CF=F_n(x_1,x_2,\ldots,x_n)$, s.t. $x_i \in \{0,1\}$ $1 \le i \le n$, represents the output of $LFSR_i$, let $S=\{s_0,s_1,\ldots\}$ be the sequence product from KG and s_j, j=0,1,… represents elements of S. let S_i be the sequence i product from $LFSR_i$ with a_{ij} elements $1 \le i \le n$, j=0,1,…,.

6.2 Implementation of BEC on Modified Geffe Generator

(a). Periodicity

The periodicity of this generator can calculate as follows:

$$\mathbf{P(S)=} \text{l.c.m}\,(2^{r_1}-1, 2^{r_2}-1, 2^{r_3}-1, 2^{r_4}-1, 2^{r_5}-1)$$

If the $LFSR_i$ has maximum period then $P(S_i)= 2^{r_i}-1$ [Gol.82].

$$P(S)=\prod_{i=1}^{5}(2^{r_i}-1).$$

Example (6.1)

if r_i=2,3,…,6 for i =1,2,...,5, then:

P(S)=l.c.m(3,7,15,31,63)

=l.c.m($3^1.5^0.7^0.31^0$, $3^0.5^0.7^1.31^0$, $3^1.5^1.7^0.31^0$, $3^0.5^0.7^0.31^1$, $3^2.5^0.7^1.31^0$)

= $3^{max(0,1,2)}$. $5^{max(0,1)}$. $7^{max(0,1)}$. $31^{max(0,1)}$= $3^2.5^1.7^1.31^1$=9765.

(b). Randomness

From the truth table of CF of modified Geffe, notice that the ratio of number of 0's (2^4=16) to the total output of the function = 32 (2^5=32) is 0.5, and so as the number of 1's, that's indicates that this generator can generates random sequence. The truth table of CF is shown in table (6.1).

Table (6.1) Truth table of GF_5 of Modified Geffe generator.

x_1	x_2	x_3	x_4	x_5	F_5
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	0
0	0	1	0	0	0
0	0	1	0	1	0
0	0	1	1	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	0	1	0
0	1	0	1	0	0
0	1	0	1	1	1
0	1	1	0	0	0
0	1	1	0	1	1
0	1	1	1	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	0	1	0
1	0	0	1	0	0
1	0	0	1	1	1
1	0	1	0	0	0
1	0	1	0	1	1
1	0	1	1	0	1
1	0	1	1	1	1
1	1	0	0	0	0
1	1	0	0	1	1
1	1	0	1	0	1
1	1	0	1	1	1
1	1	1	0	0	1
1	1	1	0	1	1
1	1	1	1	0	1
1	1	1	1	1	1
0.5	0.5	0.5	0.5	0.5	0.5
Correlation Probability (CP_i) for each LFSR					Ratio of "0"

We have to test the randomness of the Modified Geffe generator using two samples denoted by c1 and c2 with length 50000 bits and 100000 bits respectively.

CRYPT-X [Gus.94] package used to test the output results of CRKG tested by using group of tests. Table (6.2) shows the Frequency test, 1st and 2nd Binary Derivative test. While Change Point test shown in table (6.3). Subblock (Poker) and Run tests described in table (6.4).

Table (6.2) GF_5 tested by frequency and binary derivative tests.

Test	Ex.	Length	n_1	mean(1)	prop(1)	α
Freq.	c1	50000	24750	25000.0	0.4950	0.5826
	c2	100000	49930	50000.0	0.4993	0.8468
1st BD	c1	50000	25115	25000.0	0.5023	0.9927
	c2	100000	50010	50000.0	0.5001	0.9725
2nd BD	c1	50000	25195	25000.0	0.5039	0.3994
	c2	100000	50360	50000.0	0.5036	0.2943

Table (6.3) GF_5 tested by Change point test.

Ex.	Length	Change point (Cp)	n_1 before Cp	prop(1) before Cp	prop(1) after Cp	α
c1	50000	24001	12307	0.4905	0.5017	0.5182
c2	100000	48066	25279	0.4941	0.5031	0.4452

Table (6.4) GF_5 tested by subblock and Run tests.

Test	Ex.	Size	χ^2-value	DF	mean(1)	prop(1)	α
Subblock	c1	2	0.4456	3	-------	-------	0.9307
	c2		1.2338				0.7449
	c1	4	10.3570	15	-------	-------	0.9967
	c2		9.5872				0.8449
	c1	8	215.9064	255	-------	-------	0.9645
	c2		246.6523				0.6389
Run	c1	-----	19.7353	20	2980	2981	0.4746
	c2		13.4825	22	5253	5254	0.9188

Where $\mathbf{n_1}$ denotes the number of ones, **mean(1)** is the mean of ones, **prop(1)** is the proportion of ones, and lastly, α is the tail-area probability or can be called significance probability.

(c). Linear Complexity

Let's denotes the Linear Complexity for the generated sequence by LC(S), then it can by calculated by:

$LC(S)= r_1+r_2+r_1r_3+r_2r_3+r_3r_4+r_3r_5$.

Example (6.2)

Let's use the same information mentioned in example (6.1), then:

LC(S)=2+3+2*4+3*4+4*5+4*6 = 69

Now BerlyCamp-Massey algorithm will applied to estimate the LC of GF_5, table (6.5) shows LC(GF_5) for various generated sequences lengths.

Table (6.5) LC(GF_5) of various sequences length.

Ex.	***Length***	***LC(GF_5)***	***Ratio***
1	4000	1993	0.49825
2	10000	5003	0.5003
3	20000	10354	0.5177

(d). Correlation Immunity

Notes from table (6.5) (from the shaded cells) that the number of similarity between x_i and the output of CF is 16 bits from the total number 32 bits $\forall i$, then the correlation probability (CP_i) can be calculated as follows:

$CP_i = 16/32 = 0.5$, for i = 1,2,…,5.

Let's denotes the Correlation Immunity for the generated sequence by CI(S), then it can by calculated by:

CI(S) = 5,

since the number of immune variables $x_i = 5$.

This indicates that modified Geffe generator is immune and it cannot be attacked by correlation attack or fast correlation attack, while the origin Geffe generator is not immune [Sie.84] (compare tables (5.1) and (6.1)).

Naturally, the same results can be obtained from the above tables by using the real time run of the generator. Table (6.6) shows these results from various comparison sequences.

Table (6.6) CP and CI results for various sequences lengths.

Ex.	Length	CP					CI
		x_1	x_2	x_3	x_4	x_5	
c1	20000	0.502	0.521	0.572	0.529	0.511	4
c2		0.494	0.497	0.500	0.498	0.499	5
c1	40000	0.503	0.471	0.502	0.530	0.495	5
c2		0.499	0.499	0.502	0.498	0.499	5

6.3 Implementation of BEC on Modified 5-Threshold Generator

(a). Periodicity

The periodicity of this generator can calculate as follows:

$$\mathbf{P(S)}=\text{l.c.m}(2^{r_1}-1,2^{r_2}-1,2^{r_3}-1,2^{r_4}-1,2^{r_5}-1)$$

if the $LFSR_i$ has maximum period then $P(S_i)= 2^{r_i}-1$ [Gol.82].

$$P(S)=\prod_{i=1}^{5}(2^{r_i}-1).$$

Example (6.3)

if r_i=2,4,5,6,8 for i =1,...,5, then:

P(S)=l.c.m(3,15,31,63,255)

=l.c.m($3^1.5^0.7^0.17^0.31^0$, $3^1.5^1.7^0.17^0.31^0$, $3^0.5^0.7^0 17^0.31^1$,

$3^2.5^0.7^1.17^0.31^0$, $3^1.5^1.7^0.17^1.31^0$)

$= 3^{max(0,1,2)}.5^{max(0,1)}.7^{max(0,1)}.17^{max(0,1)}.31^{max(0,1)}$

$= 3^2.5^1.7^1.17^1 31^1$=166005.

(b). Randomness

From the truth table of TF_5 of modified 5-Threshold, notice that the ratio of number of 0's to the total output of the function = 32 (2^5=32) is 0.5, this mean the number of 0's = 16 and so as the number of 1's, that's indicates that this generator can generates random sequence.

The truth table of TF_5 is shown in table (6.7).

Table (6.7) Truth table of TF_5 of modified 5-Threshold generator.

x_1	x_2	x_3	x_4	x_5	F_5
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	0
0	0	1	0	0	0
0	0	1	0	1	0
0	0	1	1	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	0	1	0
0	1	0	1	0	0
0	1	0	1	1	1
0	1	1	0	0	0
0	1	1	0	1	1
0	1	1	1	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	0	1	0
1	0	0	1	0	0
1	0	0	1	1	1
1	0	1	0	0	0
1	0	1	0	1	1
1	0	1	1	0	1
1	0	1	1	1	1
1	1	0	0	0	0
1	1	0	0	1	1
1	1	0	1	0	1
1	1	0	1	1	1
1	1	1	0	0	1
1	1	1	0	1	1
1	1	1	1	0	1
1	1	1	1	1	1
0.6875	0.6875	0.6875	0.6875	0.6875	0.5
Correlation Probability (CP_i) for each LFSR					Ratio of "0"

We have to test the randomness of the Modified Geffe generator using two samples denoted by c1 and c2 with length 5000 bits and 10000 bits respectively.

CRYPT-X [Gus.98] package used to test the output results of 5-Threshold Generator tested by using group of tests. Table (6.8) shows the Frequency test, 1st and 2nd Binary Derivative test. While Change Point test shown in table (6.9). Subblock (Poker) and Run tests described in table (6.10).

Table (6.8) TF_5 tested by frequency and binary derivative tests.

Test	Ex.	Length	n_1	mean(1)	prop(1)	α
Freq.	c1	50000	25035	25000.0	0.5007	0.6550
	c2	100000	50110	50000.0	0.5011	0.3642
1st BD	c1	50000	25005	25000.0	0.5001	0.9303
	c2	100000	49930	50000.0	0.4993	0.5468
2nd BD	c1	50000	24890	25000.0	0.4978	0.9897
	c2	100000	50160	50000.0	0.5016	0.1894

Table (6.9) TF_5 tested by Change point test.

Ex.	Length	Change point (Cp)	n_1 before Cp	prop(1) before Cp	prop(1) after Cp	α
c1	50000	23749	11852	0.5025	0.4990	0.5548
c2	100000	51533	26012	0.4997	0.5021	0.6250

Table (6.10) TF_5 tested by subblock and Run tests.

Test	Ex.	Size	χ^2-value	DF	mean(1)	prop(1)	α
Subblock	c1	2	0.2461	3	-------	-------	0.9698
	c2		2.2141				0.8580
	c1	4	6.7490	15	-------	-------	0.9642
	c2		10.7923				0.7923
	c1	8	229.6054	255	-------	-------	0.8711
	c2		241.5842				0.7190
Run	c1	-----	17.9685	26	23847	23847	0.9997
	c2		17.2414	28	41955	41954	0.9437

(c). Linear Complexity

The Linear Complexity is defined as the length, of the shortest LFSR (which is equivalent LFSR) that can mimic the generator output. Any sequence generated by a finite-state machine over a finite field has a finite linear complexity [Mas.69].

Let's denotes the Linear Complexity for the generated sequence by LC(S), then it can by calculated by:

$LC(S)=r_1r_2r_3+r_1r_2r_4+r_1r_2r_5+r_1r_3r_4+r_1r_4r_5+r_2r_3r_4+r_2r_3r_5+r_1r_2r_3+r_2r_4r_5+r_3r_4r_5+$
$r_1r_2r_3r_4+r_1r_2r_3r_5+r_1r_2r_4r_5+r_1r_3r_4r_5+r_2r_3r_4r_5.$

Example (6.4)

Let's use the same information mentioned in example (6.1), then:

$LC(S)=2*3*4+2*3*5+2*3*6+2*4*5+2*4*6+2*5*6+3*4*5+3*4*6+$
$3*5*6+4*5*6+2*3*4*5+2*3*4*6+2*3*5*6+2*4*5*6+3*4*5*6$
$=1624$

Now BerlyCamp-Massey algorithm will applied to estimate the LC of CRKG, Table (6.11) shows $LC(TF_5)$ for various sequences length from the two outputs.

Table (6.11) $LC(GF_5)$ of various sequences length.

Ex.	***Length***	***LC(GF_5)***	***Ratio***
1	4000	1958	0.4895
2	10000	5107	0.5107
3	20000	9906	0.4953

(d). Correlation Immunity

Notes from table (6.7) (from the shaded cells) that the number of similarity between x_i and the output of CF is 22 bits from the total number 32 bits $\forall i$, then the correlation probability (CP_i) can be calculated as:

$CP_i = 22/32 = 0.6875$, for i = 1,2,…,5.

Let's denotes the Correlation Immunity for the generated sequence by CI(S), then it can by calculated by:

CI(S) = 0,

since the number of immune $x_i = 0$.

This indicates that this generator can be attacked by correlation attack or fast correlation attack [Sie.84].

Naturally, the same results can be obtained from the above tables by using the real time run of the generator. Table (6.12) shows these results from various comparison sequences.

Table (6.12) CP and CI results for various sequences lengths.

Ex.	**Length**	**CP**					**CI**
		x_1	**x_2**	**x_3**	**x_4**	**x_5**	
c1	20000	0.679	0.653	0.622	0.699	0.701	0
c2		0.684	0.691	0.664	0.682	0.677	0
c1	40000	0.655	0.678	0.671	0.685	0.670	0
c2		0.683	0.671	0.669	0.673	0.681	0

6.4 Solving the SLE's

After be sure that the SLE's has unique solution, the SLE's can be solved by using one of the most common classical methods, its **Gauss Elimination method**. This method chosen since it has lower complexity than other methods. As known, this method depending on two main stages, first, converting the matrix Y to up triangular matrix, and the second one, is finding the converse solution [Jaa.05].

6.4.1 Solving SLE's for Single LFSR

Applying Gauss elimination on SLE's generated from single LFSR is easy than solving SLE's generated from system of LFSR's or generator since there is no combining function (especially if it's not non-linear function). After solving we obtain the initial values of the attacked LFSR directly, since we have system with degree equal to the length of LFSR.

Example (6.5) shows the solving of a single SLE's for one LFSR.

Example (6.5)

Let's use the matrix Y of equation (5.13), after applying the elementary operations, and then the up triangular matrix is:

$$Y' = \left[\begin{array}{cccc|c} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{array}\right]$$

Now applying the backward solution to get the initial value vector:

A_0=(0,0,0,1).

6.4.2 Solving SLE's for Modified Geffe Generator

The SLE's of 5_LFSR's is more complicated than SLE's of a single LFSR, specially, if the CF is high order (non-linear) function. First, it should solve the variables which are consists of multiplying more than one initial variable bits of the combined LFSR's. we notice that every system has its own SLE's system because of the CF, so it has own solving method.

As an example to find the variables a_{ij} of modified Geffe generator, after solving the SLE's we found that (6) variables (x_k) equal (1) only from the whole number of variables (69 variables), s.t.:

$x_1=x_4=x_{12}=x_{24}=x_{44}=x_{68}=1$; elsewhere $x_k=0$ …(6.1)

From equation (5.15), we know that every x_k is consists from product of (1) or (2) unknowns, where a_i,b_j,c_k,d_l,e_n are initial values the five LFSR's contribute in 5-modified Geffe generator s.t. i=1,2,3,4,5. The SLE's Y which mentioned in example (6.2) will be solved in the next example.

Example (6.6)

From equation (6.1), and table (6.13):

Table (6.13) finding origin variables from new variable (x_k) for GF_5.

i	k=L	Combination of r_j	Indices of variables
1	1	r_1-1	a_2
2	4	r_1+r_2-1	b_3
3	12	$r_1+r_2+r_1*r_3$-1	a_2c_4
4	24	$r_1+r_2+r_1*r_3+r_2*r_3$-1	b_3c_4
5	24	$r_1+r_2+r_1*r_3+r_2*r_3+r_3*r_4$-1	C_4d_5
6	68	$r_1+r_2+r_1*r_3+r_2*r_3+r_3*r_4+r_4*r_5$-1	C_4e_6

$x_0=a_1=0$, $x_1=a_2=1$, $x_2=b_1=0$, $x_3=b_2=0$, $x_4=b_3=1$, and $x_{24}=c_4*d_5=1$, this means $c_4=d_5=1$ and so on until we found all the initial values of all LFSR's contribute the modified Geffe generator. But for example c_1 (d_1,e_1) not appears in the above group so definitely it's "0", and so on for other variables.

After applying the above process we get:

- $A_{01}=(a_1,a_2)=(a_{-11},a_{-21})=(0,1)$.
- $A_{02}=(b_1,b_2,b_3)=(a_{-12},a_{-22},a_{-32})=(0,0,1)$.
- $A_{03}=(c_1,c_2,c_3,c_4)=(a_{-13},a_{-23},a_{-33},a_{-43})=(0,0,0,1)$.
- $A_{04}=(d_1,d_2,d_3,d_4,d_5)=(a_{-14},a_{-24},a_{-34},a_{-44},a_{-54})=(0,0,0,0,1)$.
- $A_{05}=(e_1,e_2,e_3,e_4,e_5,e_6)=(a_{-15},a_{-25},a_{-35},a_{-45},a_{-55},a_{-65})=(0,0,0,0,0,1)$.

6.4.3 Solving SLE's for Modified Threshold Generator

For the modified Threshold generator, it's going to solve the variables d_k, $1\leq k\leq m-1$, then solving the initial values a_{-ij} since x_k is represented by multiplying three initial bits in 10 terms, and four initial bits in 5 terms. In another word, every system has its own SLE's style to be solved because of the CF, so it has own solving method.

As an example to find the variables a_{-ij} of modified Threshold generator, after solving the SLE's we found that 25 variables (x_k) equal (1) from the whole number of variables, for example:

$$x_{23}=x_{53}=x_{89}=x_{129}=x_{177}=x_{237}=x_{297}=x_{369}=x_{459}=x_{579}=x_{699}=x_{843}=x_{1023}=$$
$$x_{1263}=x_{1623}=1;\ \text{elsewhere}\ x_k=0 \qquad \ldots(6.2)$$

From equation (5.17), we know that every x_k is consists from product of three or four unknowns, where a_i,b_j,c_k,d_l,e_n are initial values of the five LFSR's contribute in modified Threshold generator s.t. i=1,2,3,4,5. The SLE's system Y which mentioned in example (6.2) will be solved in the next example.

Example (6.7)

From equation (6.2), and table (6.14):

Table (6.14) finding origin variables from new variables (x_k) for TF_5.

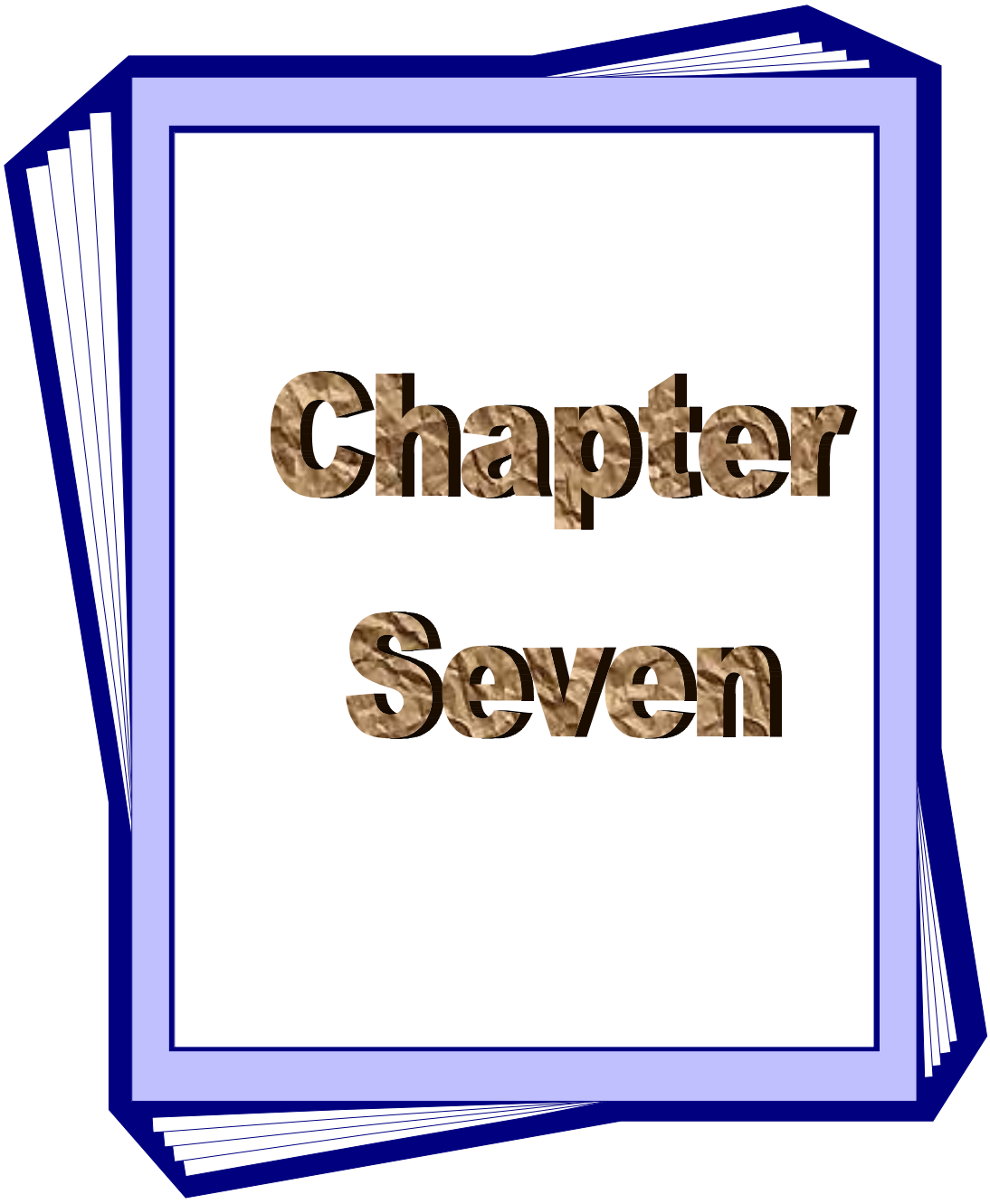
i	k=L	Combination of r_j	Indices of variables
1	23	$L=r_1*r_2*r_3-1$	$a_2b_3c_4$
2	53	$L=L+r_1*r_2*r_4$	$a_2b_3d_5$
3	89	$L=L+r_1*r_2*r_5$	$a_2b_3e_6$
4	129	$L=L+r_1*r_3*r_4$	$a_2c_4d_5$
5	177	$L=L+r_1*r_3*r_5$	$a_2c_4e_6$
6	237	$L=L+r_1*r_4*r_5$	$a_2d_5e_6$
7	297	$L=L+r_2*r_3*r_4$	$b_2c_4d_5$
8	369	$L=L+r_2*r_3*r_5$	$b_2c_4e_6$
9	459	$L=L+r_2*r_4*r_5$	$b_2d_5e_6$
10	579	$L=L+r_3*r_4*r_5$	$c_4d_5e_6$
11	699	$L=L+r_1*r_2*r_3*r_4$	$a_2b_3c_4d_5$
12	843	$L=L+r_1*r_2*r_3*r_5$	$a_2b_3c_4e_6$
13	1023	$L=L+r_1*r_2*r_4*r_5$	$a_2b_3d_5e_6$
14	1263	$L=L+r_1*r_3*r_4*r_5$	$a_2c_4d_5e_6$
15	1623	$L=L+r_2*r_3*r_4*r_5$	$b_3c_4d_5e_6$

$x_{23}=a_2*b_3*c_4=1$, this means $a_2=b_3=c_4=1$ and $x_{53}=a_2*b_3*d_5=1$ this means $a_2=b_3=d_5=1$ and so on until we found all the initial values of all LFSR's contribute the modified Threshold generator. But for example $a_2*b_3*c_4*d_1=0$, since $a_2=b_3=c_4=1$ then d_1 definitely equal "0", and so on for other variables.

After applying the above process we get:

- $A_{01}=(a_1,a_2)=(a_{-11},a_{-21})=(0,1)$.
- $A_{02}=(b_1,b_2,b_3)=(a_{-12},a_{-22},a_{-32})=(0,0,1)$.
- $A_{03}=(c_1,c_2,c_3,c_4)=(a_{-13},a_{-23},a_{-33},a_{-43})=(0,0,0,1)$.

- $A_{04}=(d_1,d_2,d_3,d_4,d_5)=(a_{-14},a_{-24},a_{-34},a_{-44},a_{-54})=(0,0,0,0,1)$.
- $A_{05}=(e_1,e_2,e_3,e_4,e_5,e_6)=(a_{-15},a_{-25},a_{-35},a_{-45},a_{-55},a_{-65})=(0,0,0,0,0,1)$.



Conclusions and Recommendations

CHAPTER SEVEN

CONCLUSIONS AND RECOMMENDATIONS

7.1 Introduction

The research contribution based on investigating constructing, testing uniqueness and solving the system of linear equations of single LFSR and some suggested new different systems like: Modified Geffe and Threshold generators. These SLE's are solved using binary Gauss elimination method which modified to be suitable to work in GF(2) filed.

7.2 Conclusions

The following are some points concluded from this study:

1. If we change our attack from known plain attack to cipher attack only, which means, changing in the sequence S (non-pure absolute values), so we shall find a new technique to isolate the right equations in order to solve the LES.

2. It is not hard to construct a SLE's of any other LFSR systems; of course, we have to know all the necessary information (CF, the number of combined LFSR's, their lengths and tapping).

3. We can find the determinant of the augmented matrix Y to investigate the uniqueness of the solution of the SLE's.

4. In non-linear generators, we notice that we don’t need all the produced variables to solve the SLE's.

5. Notice that the two suggested non-linear systems (Modified Geffe and Threshold) are passed all basic efficiency criteria specially the randomness, but these system

can be cryptanalyses as seen in chapter four, so the designer must pay attention to the method of analysis discussed on this thesis.

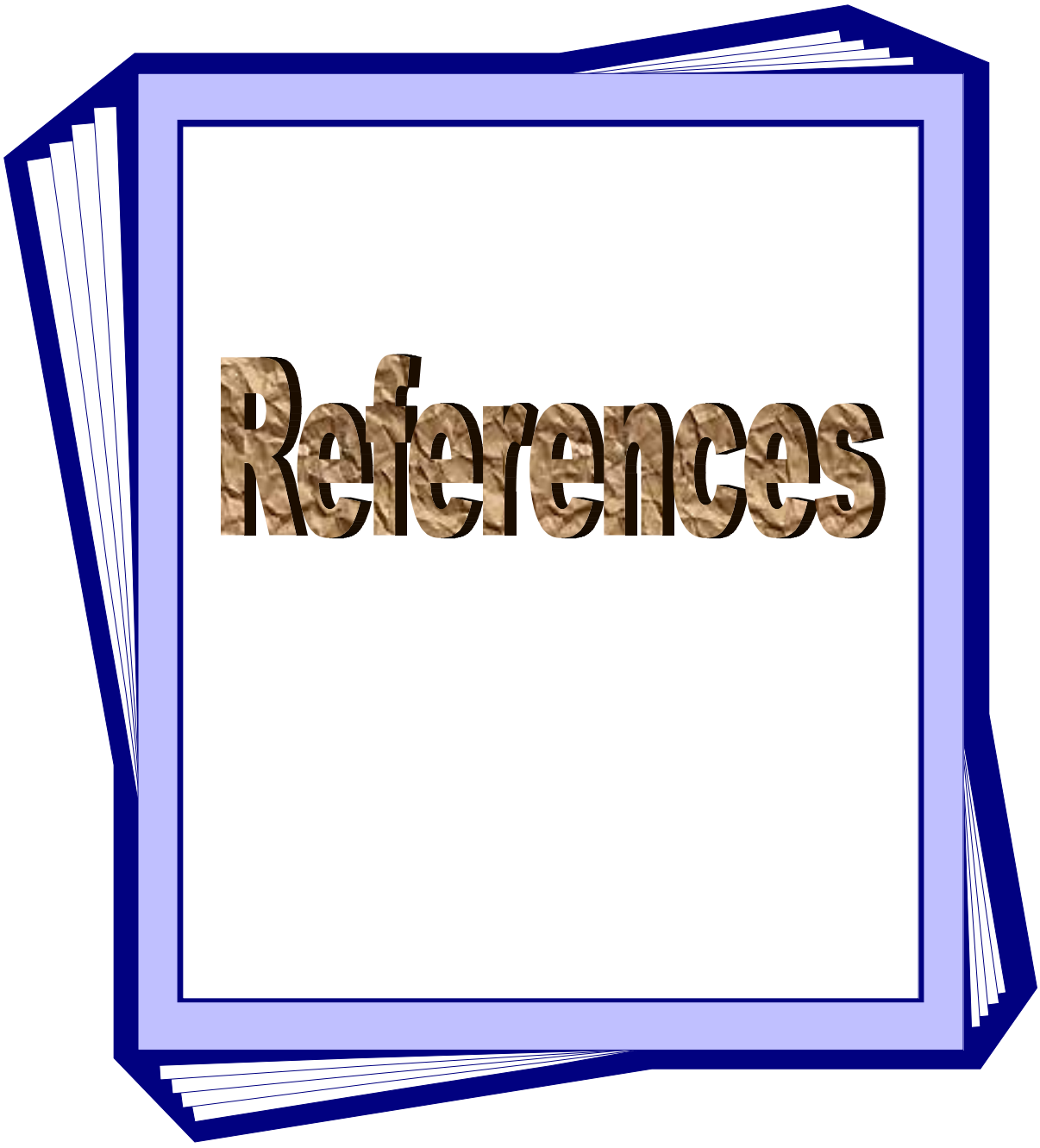
7.3 Recommendations

This Thesis recommends the following suggestion as future works:

1. We propose to use other methods of solving SLE's, like Gauss-Jordan, Gauss-Sidel,…,etc.

2. As future work, we suggest construct SLE's for almost known and famous generators published in cryptography conferences and journals.

3. We suggest to use some methods to solve the non-linear system of Geffe or Threshold generators, like Newton-Raphson method, this method remains the number of unknowns more less then when it converted to SLE's, for instance, in Modified Geffe the number of variables are (69) variables, while when using the new technique, the number of variables is just (20) variables.



References

References

[All.85]. Allenby, R. B., "***Rings, Fields and Groups***", Springer-Verlag, 1985.

[AlS.09]. Al-Shammari, A. G., "***Mathematical Modeling and Analysis Technique of Stream Cipher Cryptosystems***", Ph. D. Thesis, University of Technology, Applied Sciences, 2009.

[And.94]. Andrews, G. G, "***Number Theory***", Dover Publications, October 1994.

[Apo.98]. Apostol, T. M., "***Introduction to Analytic Number Theory***", Corrected 5th Printing, Undergraduate Texts in Mathematics, Springer-Verlag, 1998.

[Aug.85]. August, D., "***Information Theoretic Approach to Secure LFSR Ciphers***", Cryptologia, pp. 351-359, October, 1985.

[Bau.95]. Baum, U. and Blackburn, S. "***Clock Controlled Pseudorandom Generators on Finite Groups***", K.U Leuven Workshop Cryptographic Algorithms, Springer-Verlag, 1995.

[Bek.82]. Beker, H. and Piper, F., "***Cipher Systems: The Protection of Communications***", John Wiley & Sons, New York, 1982.

[Ben.99]. Bennett, D. J. "***Randomness***", Harvard University Press, October 1999.

[Bha.77]. Bhattacharyya, G., and Johnson, R., "***Statistical Concepts and Methods***", John Wiley and Sons, New York, 1977.

[Bla.87]. Blahut, R. E., "***Principles and Practice of Information Theory***", Addison-Wesley, Reading, Massachusetts, 1987.

[Bra.88]. Brassard, G., "***Modern Cryptology: A Tutorial***", LNCS 325, Springer-Verlag, New York, 1988.

[Bri.99]. Briceno, M., Goldberg, I., and Wagner, D., "***A Pedagogical Implementation of A5/1***", Available at http://jya.com/a51-pi.htm (accessed August 18, 2003), 1999.

[Brü.83].Brüer, J. O., "***On Nonlinear Combinations of Linear Shift Register Sequences***" Internal Report LITH-ISY-1-0572,1983.

[Car.88]. Carrol, J., and Robins, L., "***Computer Cryptanalysis***", Technical Report No. 223, Dept. of Computer Science, The University of Western Ontario, London, Ontario, 1988.

[Che.05]. Chen, K. and et al, "***Dragon: A Fast Word Based Stream Cipher***", eSTREAM, ECRYPT Stream Cipher Project, Report 2005/006 (2005-04-29), 2005.

[Che.91]. Chepyzhov, V. and Smeets, B., "***On a Fast Correlation Attack on Certain Stream Ciphers***", Advances in Cryptology–EUROCRYPT '91 (LNCS 547), 176–185, 1991.

[Cla.96]. Clark, A., Golić, J. , and Dawson, E., "***A Comparison of Fast Correlation Attacks***", D. Gollmann, editor, Fast Software Encryption, Third International Workshop (LNCS1039), 145–157, Springer-Verlag, 1996.

[Cod.97]. Coddington, P. D., "***Random Number Generators for Parallel Computers***", Northeast Parallel Architectures Center, 111 College Palace, Syracuse University, Syracuse, NY 13244-4100, U.S.A. Ver. 1.1, 1997.

[Dav.91]. Davalo, E. and Nairn P., "***Neural Networks***", Machllan, 1991.

[eST.05]. eSTREAM**:** ECRYPT Stream Cipher Project, IST-2002-507932. Available at http://www.ecrypt.eu.org/stream/ (accessed September 29, 2005), 2005.

[Flu.00]. Fluhrer, S. R. and McGrew, D. A., "***Statistical Analysis of the Alleged RC4 Keystream Generator***", In B. Schneier, editor,

Fast Software Encryption 2000, volume 1978 of Lecture Notes in Computer Science, pages 19–30. Springer-Verlag, 2000.

[Fra.94]. Fraleigh, J. B., "***A First Course in Abstract Algebra***" 5th Edition, Addison-Wesley, 1994.

[Gef.73].Geffe, P. R., "***How to Protect Data with Ciphers that are Really Hard to Break***", Electronics pp. 99-101, Jan. 4, 1973.

[Gil.02]. Gilbert, W. J. "***Modern Algebra with Applications***", Wiley-Interscince, March 2002.

[Gol.82]. Golomb, S.W., "***Shift Register Sequences***" San Francisco: Holden Day 1967,(Reprinted by Aegean Park Press in 1982).

[Gol.89]. Goldberg, D. E., "***Genetic Algorithm in Search, Optimization, and Machine Learning***", Boston: Addison-Wesley, 1989.

[Gus.94]. Gustafson, H., Dawson, E., Nielsen, L. and Caelli, W.,"***A Computer Package for Measuring the Strength of Encryption Algorithms***", Computers & Security, 13 (1994), 687–697.

[Gus.96-1]. Gustafson, H., "***Statistical Analysis of Symmetric Ciphers***", PhD thesis, Queensland University of Technology, 1996.

[Gus.96-2]. Gustafson, H., Dawson, E., and Golić, J., "***Randomness Measures Related to Subset Occurrence***", E. Dawson and J. Golić, editors, Cryptography: Policy and Algorithms, International Conference, Brisbane, Queensland, Australia, July 1995 (LNCS 1029), 132–143, 1996.

[Hal.02]. Halevi, S., Coppersmith, D., and Jutla, C. S., "***Scream: A Software Efficient Stream Cipher***", In J. Daemen and V. Rijmen, editors, Fast Software Encryption 2002, volume 2365 of Lecture Notes in Computer Science, pages 195–209. Springer-Verlag, 2002.

[Hec.90]. Hecht-Nielsen, R., "***Neurocomputing***", Addison-Wesley Publishing Co. Inc., U. S. A., 1990.

[Hen.89]. Henkel, W., "***Another Description of the BerleKamp-Massey Algorithm***", IEEE Proceedings, vol. 136, pt. 1, no. 3, pp. 197-200, June, 1989.

[Jaa.05]. Jaan Kiusalaas, "***Numerical Methods in Engineering with MATLAB***", Cambridge University Press, 2005.

[Jan.90]. Jansen, C. J. and Boekee, D. E., "***On the Significance of the Directed Acyclic Word Graph in Cryptology***", Advances in Cryptology– AUSCRYPT '90 (LNCS 453), 318–326, 1990.

[Joh.02]. Johansson, T. and Jönsson, F., "***On the Complexity of Some Cryptographic Problems Based on the General Decoding Problem***", IEEE Transactions on Information Theory, 48(10): 2669–2678, 2002.

[Kla.94]. Klapper, A., "***The Vulnerability of Geometric Sequences Based on Fields of Odd Characteristic***", Journal of Cryptology,7(1994),33–51.

[Kon.81]. Konheim, A. G., "***Cryptography: A Primer***", John Wiley and Sons, Inc., 1981.

[Man.01]. Mantin, I. and Shamir, A., "***Practical Attack on Broadcast RC4***", In M. Matsui, editor, Fast Software Encryption 2001, volume 2355 of Lecture Notes in Computer Science, pages 152–164. Springer- Verlag, 2001.

[Mas.69]. Massey, J. L., "***Shift-register synthesis and BCH decoding***", IEEE Transactions on Information_Theory, 15 (1969), 122–127.

[Mat.93]. Matthews, R. A. J., "***The Use of Genetic Algorithms in Cryptanalysis***", Cryptologia, vol. XVII, no 2, pp. 187-201, April, 1993.

[Mat.95]. Matt, J. B., "***Stream Ciphers Technical Report TR-701***", RSA Laboratories, 1995.

[Mau.92]. Maurer, U. M., "***A Universal Statistical Test for Random Bit Generators***", Journal of Cryptology, v. 5, n. 2, 1992.

[Men.96]. Menezes, A. P. van Oorschot, P. and Vanstone, S., "***Handbook of Applied Cryptography",*** CRC Press, 1996.

[Mey.82]. Meyer, C. H. and Matyas, S. M, "***Cryptography: A New Dimension in Computer Data Security***", John Wiley and Sons, New York, 1982.

[Mis.98]. Mister, S., "***Cryptanalysis of RC4-like Stream Ciphers***", M.Sc. thesis, Queen's University, May 1998.

[Mot.95]. Motwani, R. and Raghavan, P., "***Randomized Algorithms***", Cambridge University Press, 1995.

[Obe.89]. Obermeier, K. K. and Barron, J. J., "***Time to Get Fired Up***", Byte, pp. 217-224, August 1989.

[Pap.01]. Papoulis, A. "***Probability Random Variables, and Stochastic Process***", McGraw-Hill College, October, 2001.

[Pau.03]. Paul, S. and Prenel, B., "***Analysis of non-fortuitous predictive States of the RC4 Keystream Generator***", In T. Johansson and S. Maitra, editors, Progress in Cryptology—INDOCRYPT 2003, volume 2904 of Lecture Notes in Computer Science, pages 52–67. Springer-Verlag, 2003.

[Pau.04]. Paul, S. and Prenel, B., "***A New Weakness in the RC4 Keystream Generator and an Approach to Improve the Security of the Cipher***", In B. Roy and W. Meier, editors, Fast Software Encryption 2004, volume 3017 of Lecture Notes in Computer Science, pages 245–259. Springer-Verlag, 2004.

[Pet.79]. Pettitt, A. N., "***A Non-Parametric Approach to the Change Point Problem***", Applied Statistic, 28(2), 126-135, 1979.

[Rec.04]. Rechberger, C., "***Side Channel Analysis of Stream Ciphers***", Master's Thesis, Institute for Applied Information Processing and Communications (IAIK) Graz University of Technology, Graz, Austria, 2004.

[Riv.97]. Rivest, R. L., "***Hand Book of Applied Cryptography***", John Wiley & Sons, 1997.

[Rog.94]. Rogaway, P. and Coppersmith, D., "***A Software- Optimized Encryption Algorithm***", In R. J. Anderson, editor, Fast Software Encryption'93, volume 809 of Lecture Notes in Computer Science, pages 56–63. Springer-Verlag, 1994.

[Rue.86-1]. Rueppel, R. A., "***Analysis and Design of Stream Ciphers***" Springer-Verlag, Berlin, 1986.

[Rue.86-2]. Rueppel, R. A., "***Linear Complexity and Random Sequences***", Advances in Cryptology–EUROCRYPT'85 (LNCS 219), 167–188, 1986.

[Rue.87]. Rueppel, R. A. and Staffelbach, O. J., "***Products of Linear Recurring Sequences with Maximum Complexity***", IEEE Transactions on Information Theory, 33 (1987), 124–131.

[Sch.96]. Schneier, B., "***Applied Cryptography: Protocols, Algorithms, and Source Code in C***", John Wiley & Sons, New York, 2nd edition, 1996.

[Sel.66]. Selmer, E. S., "***Linear Recurrence over Finite Field***", University of Bergen, Norway, 1966.

[Sie.84]. Siegenthaler, T., "***Correlation-Immunity of Nonlinear Combining Functions for Cryptographic Applications***", IEEE Transactions on Information Theory, 30 (1984), 776–780.

[Sie.85]. Siegenthaler, T., "***Decrypting a Class of Stream Ciphers using Ciphertext Only***", IEEE Transactions on Computers, 34 (1985), 81–85.

[Sma.03]. Smart, N., "***Cryptography: An Introduction***", McGraw-Hill Education, 2003. ISBN 0-077-09987-7.

[Spi.93-1]. Spillman, R., Janssen M., Nelson B., and Kepner, M, "***Use Of A Genetic Algorithm In The Cryptanalysis Of Simple Substitution Ciphers***", Cryptologia, vol.16, no. 1, pp. 31-44, 1993.

[Spi.93-2]. Spillman, R., and Rechard, A., "***Cryptanalysis of Knapsack Ciphers Using Genetic Algorithms***", Cryptologia, Vol. 17, 1993.

[Sta.89]. Staffelbach, O. and Meier, W., "***Fast Correlation Attacks on Certain Stream Ciphers***", Journal of Cryptology, 1 (1989), 159–176.

[Sti.95]. Stinson, D. R., "***Cryptography: Theory and Practice***" CRC Press, 1995.

[Whi.05]. Whitesitt, J. E., "***Boolean Algebra and its Application***", Addison-Wesley, Reading, Massachusetts, April, 2005.

[Yan.00]. Yan, S. Y., "***Number Theory for Computing***", Springer-Verlag Berlin Heidelberg, New York, 2000.

[Zol.04]. Zoltak, B., "***VMPC one-way function and stream cipher***", In B. Roy and W. Meier, editors, Fast Software Encryption 2004, volume 3017 of Lecture Notes in Computer Science, pages 210–225. Springer-Verlag, 2004.

الملخص

هو احد أهم فئات خوارزميات التشفير. عادة ما يتم به Stream Cipherالتشفير الانسيابي تشفير رمز منفرد (عادة ما يكون ثنائي رقمي واحد) من النص الواضح للرسالة في الوقت الواحد، باستخدام دالة تحويل شفري تتغير مع الوقت. متتابعات المسجل الزاحف والتي تتولد من المسجل تستخدم في مجال علم التشفير ونظرية الترميزLFSRالزاحف الخطي ذو التغذية المرتدة

إن الدورة، التعقيد الخطي، العشوائية و ممانعة الارتباط اعتبرت مقاييس أساسية لقياس كفاءة مولد المفاتيح. لقد اعتمد مولد المفاتيح على المسجل الزاحف و الدالة المركبة ، باعتبارهما تمثل الوحدات الأساسية لبناء نظم التشفير الانسيابي.

هذا البحث تركز في امرين مهمين هما تصميم وتحليل نظامي تشفير جديدة مقترحة. هذان النظامان تم تطويرهما من نظامين معروفين، النظامان المقترحان سميت مولدي جيف وثريشولد المحورة.

من خلال هذا البحث، تم حساب مقاييس الكفاءة الاساسية للمولدين المقترحين نظريا قبل التنفيذ او البناء العملي (ماديا أو برمجيا). وقد اجتاز هذان النظامان هذه المقاييس بنجاح.

واخيرا، فقد تم تحليل المولدين المحورين المقترحين من خلال بناء، وفحص وحدانية ثم حل نظام المعادلات الخطية للمتتابعة المخرجة من المولدين بالرغم من تعقيدهما العالي. ان حل نظام المعادلات الخطية يعني ايجاد القيم الابتدائية للمسجلات الزاحفة المشاركة في المولد الواحد.

ان اهم ما يمكن استنتاجه من هذا البحث هو: ان كل مولد يجب ان يتجتاز مقاييس الكفاءة الاساسية ولكن هذا غير كافي ليكون امين ضد انواع المهاجمات.

بسم الله الرحمن الرحيم

...وقل ربي 114 علماً

صدق الله العلي العظيم

سورة طه الآية (114)

جامعة سانت كليمنتس العالمية

تحليل نظم التشفير الانسيابي ذات المسجلات الزاحفة

رسالة مقدمة الى
جامعة سانت كليمنتس
وهي جزء من متطلبات نيل درجة دكتوراه فلسفة في علوم الرياضيات

مقدمة من قبل الطالب
ثائر سامي رشيد الربيعي

بأشراف
ا.م.د.حسين نعمه الحسيني

Baghdad 2012