T.C.
BAHÇEŞEHİR ÜNİVERSİTESİ

TIME/MEMORY TRADEOFF ATTACK

ON

EDITING GENERATOR

MS Thesis

Elif KOPARAN ERMİŞ

İSTANBUL, 2010

T.C.
Bahçeşehir Üniversitesi
The Graduate School Of
Natural And Applied Sciences

TIME/MEMORY TRADEOFF ATTACK ON EDITING GENERATOR

MS Thesis

Elif KOPARAN ERMİŞ

Supervisor: Prof. Dr. Emin ANARIM

İSTANBUL, 2010

T.C.
BAHÇEŞEHİR ÜNİVERSİTESİ
The Graduate School Of
Natural and Applied Sciences

Name of the thesis : Time/Memory Tradeoff Attack on Editing Generator
Name/Last Name of the Student : Elif KOPARAN ERMİŞ
Date of Thesis Defence : 14.04.2010

The thesis has been approved by the Graduate School of Natural and Applied Sciences.

Yrd. Doç. Dr. Tunç Bozbura
Director

This is to certify that we have read this thesis and that we find it fully adequate in scope, quality and content, as a thesis for the degree of Master of Science.

Examining Comittee Members	Signature
Prof. Dr. Emin ANARIM	_____
Doç Dr. Adem KARAHOCA	_____
Yrd. Doc. Dr. Tevfik AYTEKİN	_____

T.C.
BAHÇEŞEHİR ÜNİVERSİTERSİ
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği

Tezin Başlığı : Düzenleme Üreteci üzerine Zaman/Bellek Takas Saldırısı
Öğrencinin Adı Soyadı : Elif KOPARAN ERMİŞ
Tez Savunma Tarihi : 14.04.2010

Bu yüksek lisans tezi Fen Bilimleri Enstitüsü tarafından onaylamıştır.

Yrd. Doç. Dr. Tunç Bozbura
Enstitü Müdürü

Bu tez tarafımızca okunmuş, nitelik ve içerik açısından bir Yüksek Lisans tezi olarak yeterli görülmüş ve kabul edilmiştir.

Tez Sınav Jürisi Üyeleri	İmza
Prof. Dr. Emin ANARIM	_____
Doç Dr. Adem KARAHOCA	_____
Yrd. Doc. Dr. Tevfik AYTEKİN	_____

ABSTRACT

TIME/MEMORY TRADEOFF ATTACK

ON

EDITING GENERATOR

ERMİŞ KOPARAN, Elif

M.S. Department of Computer Engineering

Supervisor: Prof. Dr. Emin ANARIM

April, 2010, 69 pages

In 1980 Hellman introduced a Linear Feedback Shift Register (LFSR) based attack called Time/Memory Tradeoff (TMTO) attack for breaking block ciphers. After that Babbage (1995) and Golic (1997) pointed out, this TMTO attack applicable to stream ciphers. In the time phase of the TMTO attack, the time sequence divided by windows that has a certain period. Preparation of this thesis, LFSR based Editing Generator, introduced by Gong and Jiang in 2005, the aim is attacked to capture the initial states of the LFSRs. But during the attack, has been discovered that in a particular created length time sequence has a recurrent window periods in itself. And these unique fewer windows show that the time sequence in the online phase of the attack has fewer bits than the given TMTO attack's time sequence.

Keywords: editing generator, time memory tradeoff

ÖZET

DÜZENLEME ÜRETECİ

ÜZERİNE

ZAMAN/BELLEK TAKAS SALDIRISI

ERMİŞ KOPARAN, Elif

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Danışmanı: Prof. Dr. Emin ANARIM

Nisan, 2010, 69 sayfa

Zaman/Bellek Takas (TMTO) saldırısı ilk olarak 1980 yılında Hellman tarafından blok şifrelemelerde çalışacak biçimde tasarlanmış ve ardından Babbage (1995) ve Golic (1997) tarafından katar şifrelemeye uyarlanmıştır. TMTO 'nun katar şifrelemeye uyarlanan hali zaman kısmında yer alan ve belli bir periyodu tamamlayan diziyi pencerelere bölmek biçiminde geliştirilmiştir. Bu tez sırasında 2005 yılında Gong ve Jiang tarafından öne sürülen Doğrusal Geri Beslemeli Kaydırmalı Yazmaç (LFSR) tabanlı Düzenleme Üreteci (Editing Generator) için TMTO atağı kullanılmış ve iki LFSR'ı oluşturan birincil diziler ele geçirilmeye çalışılmıştır. Fakat bu atak üzerinde yapılan çalışma sonucunda zaman kısmında belli bir uzunlukta oluşturulan dizinin kendi içerisinde tekrarlayan periyodlara sahip olduğu görülmüştür. Bu ise, verilen zaman dizisi içerisinde yer alan belli sayıdaki tekil pencerelerin daha az sayıda olduğunu, başka bir deyişle bu zaman dizisinin daha kısa olabileceğini göstermektedir. Tez içerisinde de bu konuya ağırlık verilmiştir.

Anahtar Kelimeler: düzenleme üreteci, zaman bellek takas saldırısı

ACKNOWLEDGMENTS

This thesis is dedicated to **my husband Orhan ERMİŞ and my family** for their patience and understanding during my master's study and the writing of this thesis.

I would like to express my gratitude to **Prof. Dr. Emin ANARIM** for not only being such great supervisors but also encouraging and challenging me throughout my academic program.

I also thank my **colleagues** and my friend **Okan ŞAKAR,** for their patience and help.

TABLE OF CONTENTS

ABSTRACT v

ÖZET vi

ACKNOWLEDGMENTS vii

LIST OF TABLES x

LIST OF FIGURES xi

LIST OF SYMBOLS / ABBREVIATIONS xii

1. INTRODUCTION 1

2. CRYPTOLOGY 4

2.1 CRYPTOGRAPHY 4

2.2 CRYPTANALYSIS 7

3. STREAM CIPHERS 10

3.1 LINEAR FEEDBACK SHIFT REGISTERS (LFSRs) 11

3.2 BOOLEAN FUNCTIONS 14

3.3 TYPES OF STREAM CIPHERS 16

3.4 SECURITY OF STREAM CIPHERS 20

4. COMPRESSION ALGORITHMS 23

4.1 A COMPRESSION MODEL FOR PSEUDO-RANDOM GENERATION 24

4.2 TYPES OF COMPRESSION ALGORITHMS 24

4.2.1 The SSG Algorithm 24

4.2.2 The BSG Algorithm 26

4.2.3 The ABSG Algorithm 27

4.2.4 The MBSG Algorithm: 27

4.2.5 The Editing Generator: 28

4.2.6.1 The Properties Of Randomness Of Ternary M-Sequences 28

4.2.6.2 Construction 29

4.2.6.2.1 Stop-and-Go Generator 30

4.2.6.2.2 Shrinking Generator 30

4.2.6.3 The Properties of Randomness 34

4.2.6.3.1 Period and Linear Span 34

4.2.6.4 Security Analysis 34

4.2.6.4.1 Parity-Check Attack 35

5. TIME/MEMORY TRADEOFF ATTACK *36*

5.1 INTRODUCTION 36

5.2 HELLMAN'S AND OECHSLIN'S TMTO ATTACKS 37

5.3 TIME-MEMORY-DATA TRADEOFF ATTACKS 39

5.4 TMTO ATTACKS ON STREAM CIPHERS 40
6. TIME/MEMORY TRADEOFF ATTACK ON 42
EDITING GENERATOR *42*
6.1 BG ATTACK (UNMODIFIED TRADEOFF) 42
6.2 HELLMAN' S ATTACK 44
6.3 COMBINED VERSION OF THE BG ATTACK AND HELLMAN' S TMTO ATTACK 45
6.4 OUR ATTACK 45
6.4.1 Online Phase 46
6.4.2 Preprocessing Phase 48
6.4.3 TMTO Attack on Bit Search Type Stream Ciphers 50
6.4.4 Our Attack on Bit Search Type Stream Ciphers 51
6.4.4.1 SSG 51
6.4.4.2 BSG, ABSG, MBSG 51
7. CONCLUSION 53
REFERENCES 54

LIST OF TABLES

TABLE 3.2.1: TRUTH TABLE……..*14*

TABLE 3.2.2: ATOMIC FUNCTIONS TRUTH TABLE……..*15*

TABLE 6.1.1: MEMORY TABLE……*43*

TABLE 6.1.2: TIME TABLE……*44*

TABLE 6.4.3.1: TMTO ATTACK VALUES……*50*

TABLE 6.4.4.2.1: OUR ATTACK VALUES ON THE OTHE STREAM CIPHERS.....*52*

LIST OF FIGURES

Figure 1: The Basic Block Cipher 6
Figure 2: Basic Stream Cipher 7
Figure 3: The Linear Feedback Shift Register 12
Figure 4: Nonlinear Combination Generator 17
Figure 5: Nonlinear Filter Generator 17
Figure 6: Stop and Go Generator 18
Figure 7: Shrinking Generator 19
Figure 8: The Shrinking Generators 25
Figure 9: Model of the Clock-Control Generator and Shrinking Generator. 29
Figure 10: Inserting operation in Clock-Control Generator 30
Figure 11: Deleting operation in Shrinking Generator 31
Figure 12 : Constructing Hellman's Table 37

LIST OF SYMBOLS / ABBREVIATIONS

⊕:	XOR, Exclusive or, modulo 2 addition
LFSR:	Linear Feedback Shift Register
PRNG:	Pseudo Random Number Generator
EG:	Editing Generator
SSG:	Self Shrinking Generator
BSG:	Bit Search Generator
MBSG:	Modified Bit Search Generator
SG:	Shrinking Generator
ANF:	Algebraic Normal Form
AES:	Advanced Encryption Standard
DES:	Data Encryption Standard
OTP:	The one-time pad
TMTO:	Time-Memory Tradeoff
BG:	Babbage Golic Attack
SB:	Shamir Biryukov Attack
***N′*:**	Number of the primitive polynomials
LCM:	Least Common Multiple

1. INTRODUCTION

It seems reasonable to assume that people have tried to conceal information in written form since writing was developed and examples survive in stone inscriptions and papyruses showing that many ancient civilizations including the Egyptians, Hebrews and Assyrians all developed cryptographic systems.

Cryptography is a science that deals with hiding the content of the messages that will be transmitted one point to another. Development in technology leads people to make more secure algorithms. They also called as compression algorithms.

In computer science and information theory, data compression or source coding is the process of encoding information using fewer bits than an unencoded representation would use through use of specific encoding schemes. But in cryptographic compression algorithms have to contain least data in a maximum size that it can hide from the original one.

The cryptographic compression algorithms are especially used to add nonlinearity to the pseudo-random sequences. We can use compression function while compressing the output of the Linear Feedback Shift Register (LFSR) like pseudo-random number generator. These compression algorithms delete or insert bits to the original sequence to prevent the keystream from attacks. Decimation based compression algorithms are good example for cryptographic compression algorithms like SSG, BSG, MBSG, ABSG and EG.

Cryptographic compression algorithms separate to two bases, symmetric and asymmetric encryption. Symmetric-key algorithms are a class of algorithms for cryptography that use trivially related, often identical, cryptographic keys for both decryption and encryption.

The encryption key is trivially related to the decryption key, in that they may be identical or there is a simple transformation to go between the two keys. The keys, in practice, represent a shared secret between two or more parties that can be used to maintain a private information link.

Asymmetric Encryption is a form of Encryption where keys come in pairs. Asymmetric Encryption is also known as Public Key Cryptography, since users typically create a matching key pair, and make one public while keeping the other secret. Users can "sign" messages by encrypting them with their private keys. This is effective since any message recipient can verify that the user's public key can decrypt the message, and thus prove that the user's secret key was used to encrypt it. If the user's secret key is, in fact, secret, then it follows that the user, and not some impostor, really sent the message.

A pseudorandom number generator (PRNG) is an algorithm for generating a sequence of numbers that approximates the properties of random numbers. Pseudo-random sequence generators (PRSG) based on LFSRs are very common structures in practice due to their efficient hardware implementation. Linear Feedback Shift Registers are used in many of the keystream generators that have been proposed in the literature. The key point that makes LFSRs important in stream cipher design is that they can produce random looking numbers to the given key value. LFSRs produce random sequences with using their linear function. The resulting output sequence is a linearly dependent structure and the input sequence can be easily evaluated within a small set of algebraic operations from any subpart of the output sequence.

The most important subject in stream ciphers which are the subset of symmetric key encryption is to design a system that produces random looking output sequences. They use pseudorandom number generators as a secret key. Stream ciphers typically execute at a higher speed than block ciphers and have lower hardware complexity. However, stream ciphers can be susceptible to serious security problems if used incorrectly.

They can be viewed as approximating the action of a proven unbreakable cipher, the one-time pad (OTP), sometimes known as the Vernam cipher. A one-time pad uses a keystream of completely random digits. The keystream is combined with the plaintext digits one at a time to form the ciphertext. Binary stream ciphers are often constructed using linear feedback shift registers (LFSRs) because they can be easily implemented in hardware and can be analysed mathematically.

Time/Memory Tradeoff first presented by Hellman in 1980 breaking for the block ciphers. Recently, Babbage (1995) and Golic (1997) independently pointed out, Hellman's TMTO applicable to the stream ciphers. TMTO attacks on stream ciphers are a serious security threat and the resistance to this class of attacks is an important criterion in the design of a modern stream cipher. These attacks are especially effective against stream ciphers where a variant of the TMTO attack can make use of multiple data to reduce the off-line and the on-line time complexities of the attack.

In the thesis we study with a new stream cipher called Editing Generator (EG). Editing generator is a combined model of the clock-control generator (viewed as insertion) and the shrinking generator (viewed as deletion). The resulting sequences are also ternary sequences. The resulting sequence an edited sequence, denoted as *Edit*(*A*, *B*), the sequence *A* a *base sequence*, and the sequence *B* a *control sequence* (Gong and Jiang, 2005).

We implement the time/memory tradeoff attack to editing generator with a new approach. In our approach we show that the time sequence in the online phase of the attack has fewer bits than the given TMTO attack's time sequence.

2. CRYPTOLOGY

Cryptography is the practice and study of hiding information. Modern cryptography intersects the disciplines of mathematics, computer science, and engineering. It consists of two subfields called cryptography and cryptanalysis. The cryptography is deals with the protection of data, developing new algorithms, protocols, systems, etc. The cryptanalysis is the study of methods for obtaining the meaning of encrypted information, without access to the secret information which is normally required to do so.

2.1 CRYPTOGRAPHY

The main goals of modern cryptography can be seen as: authentication, non-repudiation, data integrity and data confidentiality.

Authentication: Consists of two components, the fact that data has not been modified and the fact that you know who the sender is.

Non-Repudiation: Protects against denial by one of the entities involved in a communication of having participated in all or part of the communication.

Data Integrity: A data integrity service guarantees that the content of the message, that was sent, has not been tampered with.

Confidentiality: The protection of data from unauthorized disclosure.

Generally, besides the usage of security services the cryptographic algorithms also take an important point in our security structures. We can firstly group the cryptographic algorithms in two, the symmetric key encryption algorithms and the asymmetric key encryption.

Before defining the types of encryption algorithms, we have to define the concept, encryption. Encryption is a way to hide the content of the data with using set of rules and a secret key. The considerations of encryption stated with lots of principle. The most known one is that the Kerckhoff's Principle Stamp and Low (2007), due to this principle our encryption method is publicly known and the secret key will only be known for parties who use the secure communication line.

We have also introduced the basic terms that are used in the cryptographic encryption algorithm. These terms are:

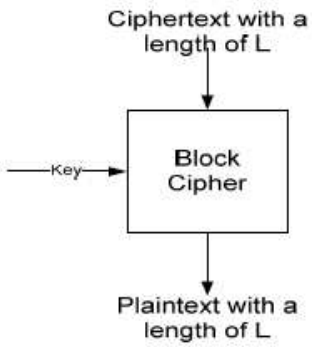
- ***Secret Key*** is a value for which we use to alter the input message.
- ***Plaintext***, also known as cleartext, is usable data. It is data either before encryption or after successful decryption.
- ***Ciphertext*** is encrypted data. Plaintext cannot be deduced from properly encrypted ciphertext.

There are also methods that do not need any secret key for producing a ciphertext, for example, the hash functions, which we will not introduce it in our study.

The modern study of symmetric-key ciphers relates mainly to the study of block ciphers and stream ciphers and to their applications. Symmetric-key cryptography refers to encryption methods in which both the sender and receiver share the same key.

The term asymmetric key encryption also known, more generally, called *public key* cryptography in which two different but mathematically related keys are used - a *public* key and a *private* key. We generally use asymmetric key encryption for key distribution systems. As mentioned above the symmetric key encryption techniques can be classified into two groups, the block ciphers and the stream ciphers.

A block cipher encryption algorithm might take (for example) a 128-bit block of plaintext as input, and output a corresponding 128-bit block of ciphertext. The exact transformation is controlled using a second input — the secret key.



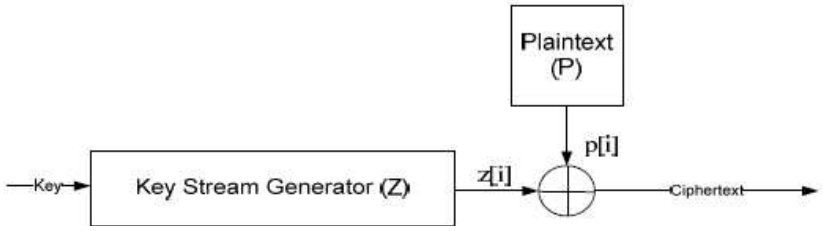
Stallings, W., 2003. *Cryptogrphy and the Network Security*. New Jersey: Prentice Hall, 3rd Edition

Figure 1: The Basic Block Cipher

An early and highly influential block cipher design was the Data Encryption Standard (DES), developed at IBM and published as a standard in 1977. A successor to DES, the Advanced Encryption Standard (AES), was adopted in 2001.

In cryptography, a **stream cipher** is a symmetric key cipher where plaintext bits are combined with a pseudorandom cipher bit stream (keystream), typically by an exclusive-or (xor) operation. Stream ciphers represent a different approach to symmetric encryption from block ciphers.

Block ciphers operate with a fixed transformation on large blocks of plaintext data; stream ciphers operate with a time-varying transformation on individual plaintext digits, see (Rueppel *et al*. 1986).



Seren, Ü., 2007. *Analysis of Compression Techniques and Memory Bit Efffects on Compression for Pseudo-Random Generation* Engineering Department.

Figure 2: Basic Stream Cipher

Stream ciphers typically execute at a higher speed than block ciphers and have lower hardware complexity.

2.2 CRYPTANALYSIS

Although the actual word "*cryptanalysis*" is relatively recent (it was coined by William Friedman in 1920), methods for breaking codes and ciphers are much older. Cryptanalysis (from the Greek *kryptós*, "hidden", and *analýein*, "to loosen" or "to untie") is the study of methods for obtaining the meaning of encrypted information, without access to the secret information which is normally required to do so. Typically, this involves finding a secret key. In non-technical language, this is the practice of codebreaking or cracking the code.

The most common cryptanalysis technique is the brute force attack, which is a method of defeating a cryptographic scheme by systematically trying a large number of possibilities; for example, a large number of the possible keys in a key space in order to decrypt a message.

For symmetric-key ciphers, a brute force attack typically means a brute-force search of the key space; that is, testing all possible keys in order to recover the plaintext used to produce a particular ciphertext.

We know that the upper bound for the complexity of an attack is the complexity of the brute force attack for every algorithm. So our purpose has to decrease the number of possibilities to make the cryptanalysis more efficient. The cryptanalysis of the algorithm is a complicated work that the analysis of the system that we consider the length of the ciphertext, plaintext and the secret key and algebraic, statistical, etc. like properties of the algorithm to break the cipher.

According to the Stallings (2003), the most common types of cryptanalytic attacks and their properties for the Kerckhoff's Principle are listed below:

Ciphertext-Only Attack (COA): Much known, ciphertext attack is an attack model for cryptanalysis where the attacker is assumed to have access only to a set of ciphertexts.

Known-Plaintext Attack (KPA): This is an attack model for cryptanalysis where the attacker has samples of both the plaintext and its ciphertext and is at liberty to make use of them to reveal further secret information such as secret keys and code books.

Chosen-Plaintext Attack (CPA): An attack model for cryptanalysis which presumes that the attacker has the capability to choose arbitrary plaintexts to be encrypted and obtain the corresponding ciphertexts.

Chosen-Ciphertext Attack (CCA): This attack model for cryptanalysis in which the cryptanalyst gathers information, at least in part, by choosing a ciphertext and obtaining its decryption under an unknown key.

Chosen Text Attack: The encryption algorithm, ciphertext to be decoded, plaintext message chosen by the cryptanalyst, together with its corresponding ciphertext generated with the secret key and the purported ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key.

3. STREAM CIPHERS

As mentioned before stream cipher is a symmetric key cipher where plaintext bits are combined with a pseudorandom cipher bit stream, typically by a xor operation. Stream ciphers can be classified into three groups: the one time pad, the synchronous stream ciphers and the self-synchronous stream ciphers to the (Menezes *et al.* 1997).

In some resources, one time pad cannot be accepted as a stream cipher. The one-time pad (OTP) is an encryption algorithm in which the plaintext is combined with a secret random key or *pad*, which is used only once and which has been proven to be impossible to crack if used correctly. The most known example for one time pad type stream ciphers the Vernam Cipher; see (Menezes *et al.* 1997). While a one-time pad cipher is provably secure (provided it is used correctly), it is generally impractical since the key is the same length as the message.

A synchronous stream cipher a stream of pseudo-random digits is generated independently of the plaintext and ciphertext messages, and then combined with the plaintext (to encrypt) or the ciphertext (to decrypt). Properties of synchronous stream cipher are defined below.

- In a synchronous stream cipher, the sender and receiver must be exactly in step for decryption to be successful. If digits are added or removed from the message during transmission, synchronization is lost. To restore synchronization, various offsets can be tried systematically to obtain the correct decryption. Another approach is to tag the ciphertext with markers at regular points in the output.

- A ciphertext digit that is modified (but not deleted) during transmission does not affect the decryption of other ciphertext digits.

- If, however, a digit is corrupted in transmission, rather than added or lost, only a single digit in the plaintext is affected and the error does not propagate to other parts of the message. This property is useful when the transmission error rate is high; however, it makes it less likely the error would be detected without further mechanisms. Moreover, because of this property, synchronous stream ciphers are very susceptible to active attacks - if an attacker can change a digit in the ciphertext, he might be able to make predictable changes to the corresponding plaintext bit; for example, flipping a bit in the ciphertext causes the same bit to be flipped in the plaintext.

There are one more structures that are really an important issue in stream ciphers called the Linear Feedback Shift Registers (LFSRs).

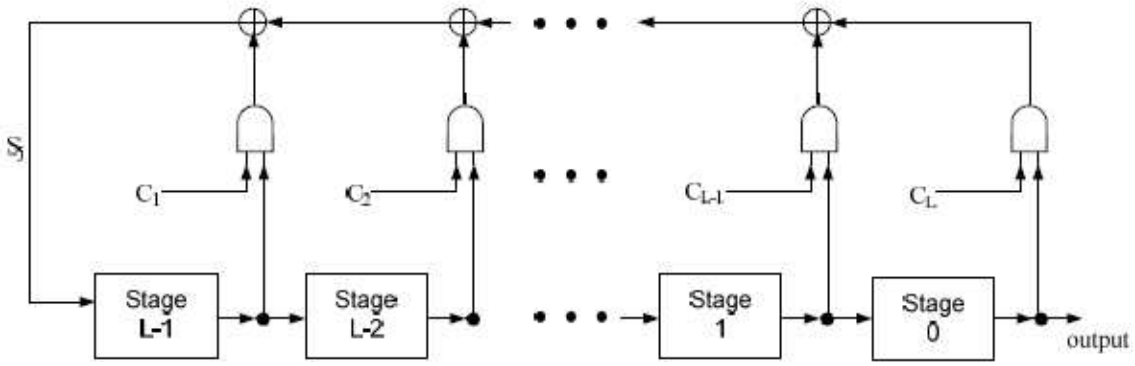
3.1 LINEAR FEEDBACK SHIFT REGISTERS (LFSRs)

Linear Feedback Shift Registers are used in many of the keystream generators that have been proposed in the literature. The key point that makes LFSRs important in stream cipher design is that they can produce random looking numbers to the given key value.

There are several reasons that make LFSRs important:

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

The following figure defines the working principle of the LFSR with a length of L, each C_i represent the feedback coefficient, the closed semi-circles shows the logical AND gates, and the feedback bit S_j is the modulo 2 sum of the contents of those stages i, $0 \leq i \leq L-1$, for which $C_{L-i} = 1$ (Menezes *et al.* 1997).



Jansen, S.J.A., 2004. *Stream Cipher Design: Make your LFSR jump!* : Workshop Record ECRYPT State of the Art of Stream Ciphers, pp. 94-108

Figure 3: The Linear Feedback Shift Register

Definition 2.1: Figure 2.1 denotes polynomial $F(D) = 1+C_1D+C_2D^2+\ldots+C_LD^L$. This polynomial is called connection polynomial which is also called the feedback polynomial Jansen (2004, pp. 94-108). And defined as

$$F(D) := \sum_{i=0}^{L} C_i D^i$$

The degree of the connection polynomial is equal to the length of the LFSR.

Definition 2.2: Assume that, we have an LFSR with a length of L, the L^{th} order recursion is commonly represented by its Characteristic Polynomial, C(D), also of degree L Jansen (2004, pp. 94-108), as shown in below:

$$C(D) := \sum_{i=0}^{L} C_i D^{L-i}$$

Definition 2.3: The functions F and C are reciprocal of each other. That means, this relation is expressed as $C(D) = D^{L}F(D^{-1})$ Jansen (2004, pp. 94-108).

Another way to look at the LFSR is to consider it as a Linear Finite State Machine as in Jansen (2004, pp. 94-108). In this case the state of the LFSM is represented by a vector $\sigma^t = (\sigma^t_{n-1}, \sigma^t_{n-2}, ..., \sigma^t_0)$ denotes the content of memory cell M_i after t transitions. As the finite state machine is linear, transitions from one state to the next can be described by a multiplication of the state vector with a transition matrix T, i.e. $\sigma_{t+1} = \sigma_t T$, for $t \geq 0$. The transition matrix is given below:

$$T = \begin{pmatrix} 0 & 0 & \dots & 0 & c_L \\ 1 & 0 & \dots & 0 & c_{L-1} \\ 0 & 1 & \dots & 0 & c_{L-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & c_1 \end{pmatrix}$$

It can be seen that the matrix is equal to the so called companion matrix of the polynomial C(D).

The characteristic polynomial of T in linear algebra sense, i.e. det(DI-T), precisely equals this polynomial and, hence, C(T) = 0. So the companion matrix plays the role of a root of C and, consequently it can be used to form solutions of the recursion equation.

Definition 2.4: Assume that we have an LFSR with a period of L. If the LFSR produce a sequence with a length of 2^{L-1} without any recursion, these LFSRs are called the maximum length LFSR.

At last, we must be careful about the initial key value of the LFSR. Because, if we use the key with all zero will makes LFSR to produce a sequence of all zero.

3.2 BOOLEAN FUNCTIONS

Boolean functions are another important element of the stream cipher concept.

Definition 2.5: In Carlet (2006), Boolean functions $\boldsymbol{f: F^n}_{\boldsymbol{2}} \rightarrow \mathbf{F_2}$ map binary vectors of length n to the finite field F_2.

There are 2^{2^n} distinct Boolean functions to the n different binary input variables and we denote the set of Boolean functions in n variables by B_n. According to the Carlet (2006), among the classical representations of Boolean functions, the one in which is most usually used in cryptography and coding is the n-variable polynomial representation over , F_2 of the form

$$f(x) = \bigoplus_{i \in P(N)} a_i \left(\prod_{j=i} x_j \right) = \bigoplus_{i \in P(N)} a_i x^i,$$

where P(N) denotes the power set of N = {1, 2, …, n}.

Every coordinate x_i appears in this polynomial with exponents at most 1, because every bit in F_2 equals its own square.

This representation belongs to $F_2[x_1,...,x_n]/(x_1^2 \oplus x_1,...,x_n^2 \oplus x_n)$. This is called the Algebraic Normal Form (ANF) (Carlet 2006).

Definition 2.6: According to the Carlet (2006), $w_H(f) := \#\{ x \in F_2^n : f(x) \neq 0\}$ is the Hamming weight of a Boolean function while the Hamming distance between two such functions is

$$\#\{ x \in F_2^n : f(x) \neq g(x)\} = w_H(f \oplus g)$$

In other words, Hamming weight is the number of ones in the vector and the Hamming distance of two functions is that the Hamming weight of modulo two additions of these two functions.

Example: Let us consider the function *f* whose truth-table is

Table 3.2.1: Truth Table

x_1	x_2	x_3	$f(x)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

It is the sum (modulo 2 or not, no matter) of the atomic functions f_1, f_2 and f_3 whose truth-tables are

Table 3.2.2: Atomic functions truth table

x_1	x_2	x_3	$f_1(x)$	$f_2(x)$	$f_3(x)$
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	0
1	1	0	0	0	0
1	1	1	0	0	1

The function $f_1(x)$ takes value 1 if and only if $1 \oplus x_1 = 1$, $1 \oplus x_2 = 1$ and $x_3 = 1$, that is if and only if $(1 \oplus x_1)(1 \oplus x_2)\ x_3 = 1$. Thus the ANF of f_1 can be obtained by expanding the product $(1 \oplus x_1)(1 \oplus x_2)\ x_3$. After similar observations on f_2 and f_3, we see that the ANF of f equals $(1 \oplus x_1)(1 \oplus x_2)\ x_3 \oplus x_1(1 \oplus x_2)\ x_3 \oplus x_1x_2x_3 = x_1x_2x_3 \oplus x_2x_3 \oplus x_3$.

3.3 TYPES OF STREAM CIPHERS

Generally, stream ciphers are divided into three groups:

- Nonlinear Combination Generators
- Nonlinear Filtering Generators
- Clock-Controlled Generators

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.

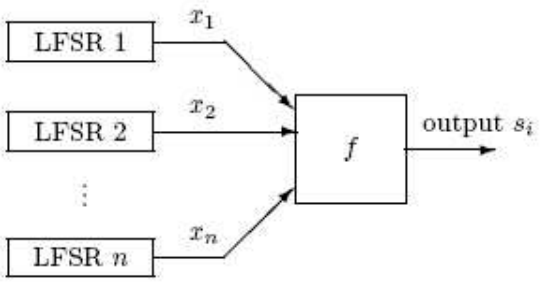
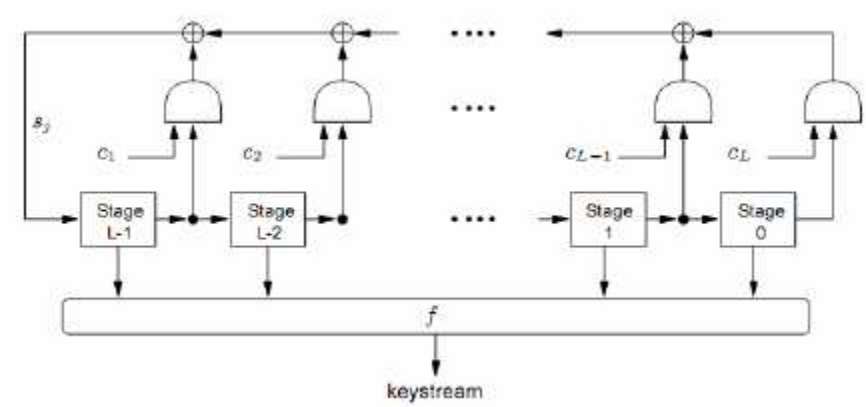


Figure 4: Nonlinear Combination Generator

Definition 3.3.1: According to the (Menezes *et al.* 1997), the definition of a nonlinear combination generator is like that 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 addition 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 is the maximum of the order of the terms appearing in its algebraic normal form.

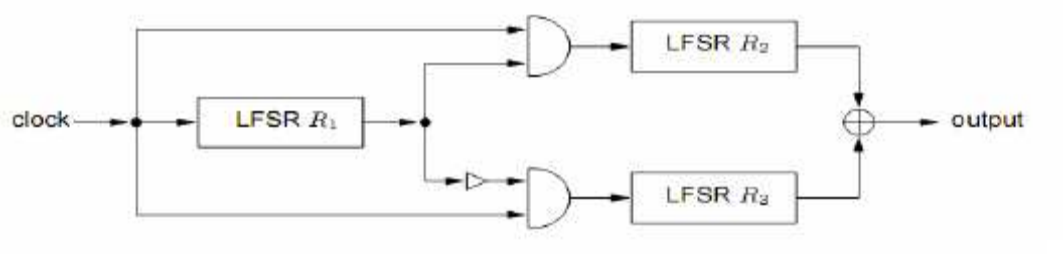
The nonlinear filter generator has different design principle. There is one LFSR and its different elements are used as an input of the Boolean function.



Seren, Ü., 2007. *Analysis of Compression Techniques and Memory Bit Effects on Compression for Pseudo-Random Generation* İstanbul : M.S. Thesis, Boğaziçi University, Electrical and Electronics Engineering Department

Figure 5: Nonlinear Filter Generator

The clock-controlled generators can be expressed into two groups. The first one is the stop and go generator. In this type of clock-controlled generator one LFSR is used to clock the other two LFSRs as shown in the following figure.



Seren, Ü., 2007. *Analysis of Compression Techniques and Memory Bit Effects on Compression for Pseudo-Random Generation* İstanbul : M.S. Thesis, Boğaziçi University, Electrical and Electronics Engineering Department

Figure 6: Stop and Go Generator

We can summarize the operation of alternating step generator, as in (Menezes *et al.*1997), as follows:

1. Register R_1 is clocked.

2. If the output of R_1 is 1 then:

R_2 is clocked; R_3 is not clocked but its previous output bit is repeated.

(For the first clock cycle, the "previous output bit" of R_3 is taken to be 0.)

3. If the output of R_1 is 0 then:

R_3 is clocked; R_2 is not clocked but its previous output bit is repeated.

(For the first clock cycle, the "previous output bit" of R is taken to be 0.)

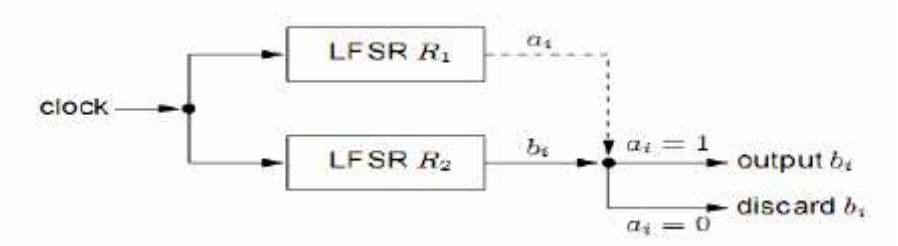
4. The output bits of R_2 and R_3 are XOR 'ed; the resulting bit is part of the keystream.

Second type of clock-controlled generators are that the Shrinking Generator. The working principle of the Shrinking generator is as follows summarizing the operations of this generator in the following three steps (Menezes et al. 1997):

1. Registers R_1 and R_2 are clocked.

2. If the output of R_1 is 1, the output bit of R_2 forms part of the keystream.

2. If the output of R_1 is 0, the output bit of R_2 is discarded.



Seren, Ü., 2007. *Analysis of Compression Techniques and Memory Bit Effects on Compression for Pseudo-Random Generation* İstanbul : M.S. Thesis, Boğaziçi University, Electrical and Electronics Engineering Department

Figure 7: Shrinking Generator

This two generator is important because editing generator is a combined model of the clock-control generator and the shrinking generator.

3.4 SECURITY OF STREAM CIPHERS

For a stream cipher to be secure its keystream must have a large period and it must be impossible to *recover the cipher's key* or internal state from the keystream. Cryptographers also demand that the keystream be free of even subtle biases that would let attackers *distinguish* a stream from random noise, and free of detectable relationships between keystreams that correspond to *related keys* or related nonces.
This should be true for all keys (there should be no *weak keys*), and true even if the attacker can *know* or *choose* some *plaintext* or *ciphertext.* Following concepts are playing an important role to attack the system.

- **Time Complexity:** Time complexity refers to a function describing how much time it will needs to be processed to apply attack and to reach success.
- **Data Complexity:** It can be defined as the required amount of keystream material which is needed to guarantee the success of attack.
- **Memory Complexity:** It can be defined as the required memory to attack the cipher. It is just like a combination of both the time and data complexity.

The important cryptanalysis methods are listed below

Trade – Off Attacks: The trade-off attack proposes a solution that lies in between the two solutions. The precomputation time is still on the order of 2^n, but the memory complexity is $2^{2n/3}$ and the inversion of a single value requires only $2^{2n/3}$ function evaluations. The main purpose of the trade-off attacks is to decrease the search complexity of the exhaustive search into data and time complexity, see Babbage (2006) for detailed information.

Algebraic Attacks: These types of attacks try to find a linear equation with a higher degree between input and output of the stream cipher. The recently developed algebraic attacks apply to all keystream generators whose internal state is updated by a linear transition function, including LFSR-based generators. For the latest journal see Armknecht, F. *et al., 2006.*

Distinguishing Attacks: Broadly an attack in which the attacker is given a black box containing either an instance of the system under attack with an unknown key, or a random object in the domain that the system aims to emulate. While it is conceivable that distinguishing attacks applied to stream ciphers might yield information about the key, or about future keystream, many distinguishing attacks on stream ciphers do not, by themselves, compromise the cipher in its normal operation. For example, RC4 is vulnerable to a distinguishing attack of order 2^{31} bytes (Hawkes and Rose, 2002).

Correlation Attacks: Class of known plaintext attacks for breaking stream ciphers whose keystream is generated by combining the output of several LFSRs using a Boolean function. Correlation attacks exploit a statistical weakness that arises from a poor choice of the Boolean function it is possible to select a function which avoids correlation attacks, so this type of cipher is not inherently insecure.

Dictionary Attack: This essentially involves running through a dictionary of words in the hope that the key (or the plaintext) is one of them. This type of attack is often used to determine passwords since people usually use easy to remember words. So this type of attack can be prevented by avoiding the words present in the dictionaries as key or password. Alphanumeric characters can be a better option against dictionary attack.

Guess and Determinate Attack: The strategy used here is to guess a few of the unknown variables in the cipher, and from the guessed values deduce the values of other unknown variables. This is often slower compared to exhaustive key search due to nonlinearities and irregularities in the cipher. Because of this, an assumption is made that makes the cipher more linear. If the probability that the assumption holds is p, the expected number of tries until the assumption holds are 1/p.

Side Channel Attack: "Side channel" attacks are based on "Side channel information". Side channel information is information retrieved from the physical implementation instead of theoretic weaknesses. Any information that can be measured, and is dependent on the key, state, or plaintext. Side channel analyses are of concern because the attacks can be mounted quickly and also be implemented using readily available hardware costing only a few dollars to thousands of dollars. The amount of time required to attack and analysis depends on the types of attack (power analysis, timing attack,etc). In a timing attack the attacker tries to break a cipher by analyzing the execution time for encryption or decryption. It can be done if the encryption or decryption time depends on the input. The cryptosystems often take slightly different time to process different inputs. This is usually the case for asymmetric algorithms.

4. COMPRESSION ALGORITHMS

In computer science and information theory, **data compression** or **source coding** is the process of encoding information using fewer bits than an unencoded representation would use through use of specific encoding schemes. The cryptographic compression algorithms have to contain least data in a maximum size that it can hide from the original one.

The cryptographic compression algorithms are especially used to add nonlinearity to the pseudo-random sequences. As we mentioned that pseudo-random sequences has weaknesses to the Algebraic attacks.

We can use compression function while compressing the output of the LFSR like pseudo-random number generator. The compression function takes n bits of input and produces m bits of output, where $n \geq m$ can be classified to the value of output rate.

Decimation based compression algorithms are good example for cryptographic compression algorithms. These are used as a second structure to compress the output of the PRNG structure. The most common decimation algorithms are SSG, BSG, ABSG and MBSG.

4.1 A COMPRESSION MODEL FOR PSEUDO-RANDOM GENERATION

To make the keystream more secure, we will use some compression algorithms. These compression algorithms - like the Editing Generator - delete or insert bits to the original sequence to prevent the keystream from attacks that are related with the algebraic or correlation type properties of pseudo-random outputs.

In Gouget and Sibert (2006), the term random input sequences is introduced, those sequences that follow the uniform distribution of binary words: each word *w* is a prefix of a random input sequence with probability $1/2^{|w|}$, and all words are assumed to be independent.

4.2 TYPES OF COMPRESSION ALGORITHMS

4.2.1 The SSG Algorithm

The self-shrinking generator is a modified version of the shrinking generator and was first presented in Meier and Staffelbach (1995).

The self-shirinking generator(SSG) requires only one LFSR *A,* whose length will be denoted by *L.* The LFSR generates an m-sequence $(a_i)_{i\geq 0}$ in the usual way. The selection rule is the same as for shrinking generator, using the even bits $a_0, a_2, \ldots$ as S-Bits and the odd bits $a_1, a_3, \ldots$ as A-Bits in the above sense. Thus, the self-shrinking rule requires a tuple (a_{2i}, a_{2i+1}) as input and outputs a_{2i} iff $a_{2i} = 1$. The close relationship between shrinking and self-shrinking generator is shown in figure 8.

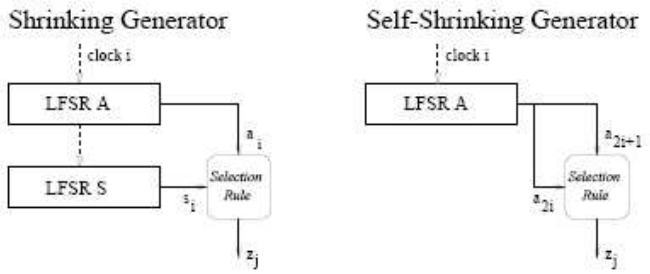


Figure 8: The Shrinking Generators

Assume that we have a random variable X | X = {x_0, x_1, x_2 …}, which was generated by LFSR, used as an input sequence to the SSG. The random variable Z | Z = {z_0, z_1, z_2…}, which was generated from X, is accepted as output. The output rate of the Self-Shrinking Generator is 1/4. According to the definition in (Zenner *et al.* 2001), SSG algorithm searches the bits that have the even position, if the value of bit is 1, sets the output bit as the latter bit of the even positioned bit, else the value of even positioned bit is 0, and the algorithm gives no output. Summarize the SSG algorithm as follows:

```
Set i := 0, j := 0
while (true)
    if (x[i] ≠ 0) z[j] := x[i+1]
    j := j + 1
    i := i + 2
```

The output rate calculation is very simple for SSG algorithm. Assume that we have an evenly distributed input sequence, the occurrence of 1 in even position is 1/2 and also we know that the algorithm gives one output bit to the given two input bits that means we have L/2 output bits, if all of the bits at the even positions are 1 with the input sequence length of a L. Under the conditions that are mentioned above, we can easily say that we have the output rate of 1/4.

4.2.2 The BSG Algorithm

The Bit Search Generator algorithm is a kind of cryptographic compression algorithm that takes a pseudorandom input with size of L and produces the output with a size of L/3. This algorithm was firstly proposed in Gouget and Sibert (2004). The random variable X | X = $\{x_0, x_1, x_2 \ldots\}$, which was generated by the LFSR, is accepted as an input to the BSG. The random variable Z| Z = $\{z_0, z_1, z_2, \ldots\}$, which was constructed from X, is accepted as output. The BSG algorithm works like this, first of all the x_0 from input bit is set as a search bit and then, the algorithm starts to search the bit which has the same value with x_0. Assume that we find the correct bit at the position l, if there are no bits between x_l and the search bit, then, the resulting bit will be 0 for output, otherwise, the output bit will be 1. The working principles of BSG algorithm in pseudo code format is as shown in below:

```
Set i := -1, j := -1
while(true)
    i := i + 1
    j := j + 1
    b := x[i]
    i := i + 1
    if (x[i] = b) set z[j] := 0
    else
    while(s[i] ≠ b) i := i + 1
```

Average output rate of the BSG is equals 1/3 (this is obviously the case for random input sequences), whereas the average output rate of both the Shrinking and the Self-Shrinking Generator is 1/4.

4.2.3 The ABSG Algorithm

The ABSG algorithm is the improved version of the BSG algorithm which was proposed in (Gouget *et al.* 2005). The working principle of the ABSG is most likely to the BSG algorithm except the determination of the output bits.

In ABSG algorithm, the output bit is selected from the input bit, the second bit of the codeword is used as an output bit and there is no change in the searching process. Summarize the algorithm as pseudo code below:

```
Set i := 0, j := 0
while(true)
    b := x[i]
    z[j] := x_{i+1}
    i := i + 1
    while(x[i] = b̄ ) i := i + 1
        i := i + 1
        j := j + 1
```

4.2.4 The MBSG Algorithm:

MBSG algorithm is the Modified Bit Search Generator, was firstly introduced in (Gouget *et al.* 2005). MBSG has the output rate of 1/3 for evenly distributed inputs. This algorithm searches the subsequences $b0^i1$, where $i \geq 0$ and b is selected as the output bit.

```
Set i := 0, j := 0
while (true)
    y_i := x_i
    i := i + 1
    while (x_i = 0) i := i + 1
    i := i + 1
    j := j + 1
```

4.2.5 The Editing Generator:

Pseudo-random sequences are commonly used in cryptology. It can be implemented as either key stream generators in stream cipher systems or pseudo-random number generators in session key generators. Pseudo-random sequence generators (PRSG) based on linear feedback shift registers (LFSR) are most common structures in practice due to their efficient hardware implementation.

This new generator is called an editing generator which is a combined model of the clock-control generator (viewed as insertion) and the shrinking generator (viewed as deletion) was firstly proposed in Gong and Jiang (2005). Constructed by using two ternary LFSRs and the resulting sequences are also ternary sequences.

4.2.6.1 The Properties Of Randomness Of Ternary M-Sequences

As mentioned in (Gong and Jiang, 2005) *Let A be a ternary m-sequence of degree n.*

(1) The least period of A is $3^n - 1$.

(2) It satisfies the balance property, i.e., each non-zero element in Z_3 occurs 3^{n-1} times in one period of A and zero element occurs $3^{n-1} - 1$ times in one period.

(3) It satisfies the following run property. In each period,

(a) for $1 \leq k \leq n-2$, the runs of each element in Z_3 of length k occur $2^2 3^{n-k-2}$ times,

(b) the runs of each nonzero element of length $n - 1$ occur once,

(c) the runs of the zero element occur twice,

(d) the run of each nonzero element of length n occurs once.

(4) 2-level autocorrelation.

(5) Each nonzero n – tuple occurs exactly once.

The following is an example of a ternary m-sequence of degree 3 generated by $f(x) = x^3 – x^2 – 2$.

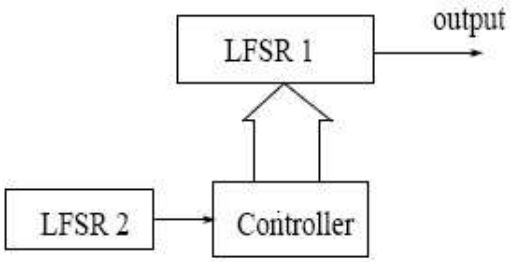
0011102112101

0022201221202

4.2.6.2 Construction

In this section, introducing the construction of the editing generator and derive some basic properties on randomness. First, take a look at the operations of the stop and-go generator, a simple model of clock-control generators, and the shrinking generator. Both of these generators can be represented in the following model Figure 9.

Let $A = \{ a_i \}$ and $B = \{ b_j \}$ be two binary sequences generated by LFSR 1 and LFSR 2 respectively.



Gong, G., Jiang, S., 2005. *The Editing Generator and Its Cryptanalysis*, International Journal of Wireless and Mobile.

Figure 9: Model of the Clock-Control Generator and Shrinking Generator.

4.2.6.2.1 Stop-and-Go Generator

Output sequence: $S = \{ s_k \}$ whose elements are given by

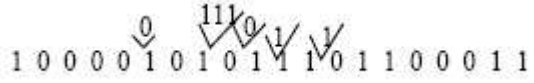
$$s_{k_j} = a_{i_j}, j = 0,1,..., where\ k_j = j, i_j = \sum_{t=0}^{j} b_t - 1\ and\ set\ a_{-1} = 0$$

In the other words, at time j, if $b_j = 1$, the generator outputs the current output bit of LFSR1. Otherwise, the generator outputs the previous output bit (inserting).

For example, let

$$A = 1\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ \cdots$$
$$B = 1\ 1\ 1\ 1\ 1\ \underline{0}\ 1\ 1\ 1\ \underline{0\ 0\ 0}\ 1\ \underline{0}\ 1\ \underline{0}\ 1\ 1\ \underline{0\ 0}\ \cdots$$

The inserting process is shown in Figure 10.



Gong, G., Jiang, S., 2005. *The Editing Generator and Its Cryptanalysis*, International Journal of Wireless and Mobile.

Figure 10: Inserting operation in Clock-Control Generator

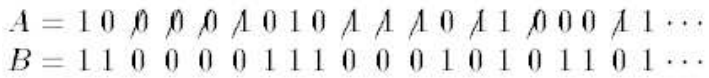
The first 20 output bits are: 10000010111100111110

4.2.6.2.2 Shrinking Generator

Output sequence: $S = \{s_k\}$, at time j, if $b_j = 1$, then the generator outputs the current bit a_j of A. Otherwise this bit is discarded, i.e.,

$$s_{k_j} = a_{i_j}, j = 0,1,..., where\ i_j = j, k_j = \sum_{t=0}^{i} b_t$$

For example,



Gong, G., Jiang, S., 2005. *The Editing Generator and Its Cryptanalysis*, International Journal of Wireless and Mobile.

Figure 11: Deleting operation in Shrinking Generator

The first 10 output bits from the shrinking generator are: 1001001001

The new generator, the editing generator, is to combine these two operations together, in order to resist to the known attacks to those two control models when they are used separately.

Definition 1: Let $A = \{a_i\}$ and $B = \{b_j\}$ be two ternary sequences generated by LFSRs 1 and 2, respectively. The output sequence $S = \{s_k\}$ is determined by the following rules. At time j, if $b_j = 2$, the generator discards the current output element in LFSR 1; else if $b_j = 1$, the generator outputs the current output element in LFSR 1. Otherwise, the generator outputs the previous output element in LFSR 1. We will call it an editing generator, the resulting sequence an edited sequence, denoted as *Edit*(A, B), the sequence A a *base sequence*, and the sequence B a *control sequence*.

This definition can be written into the following pseudocode:

Algorithm 1 (*Edit* (*A*, *B*))**:** Algorithm for generating edited sequences

Input: A and B, two ternary LFSR sequences
Output: an edited sequence S of length m

1. **Procedure** $Edit(A, B, S, m)$
2. Set $a_{-1} = 0$
3. Set $k = j = i = 0$
4. **while** $(j < m)$ **do**
   (a) **if** $b_j = 2$, **then** set $i = i + 1$ // discarding the current output element in A
   // (deleting an element from A)
   (b) **else if** $b_j = 1$, **then** set
   $s_k = a_i$; $k = k + 1$; $i = i + 1$ //outputing the current element in A
   // (keeping an element of A)
   (c) **else** set $s_k = a_{i-1}$; $k = k + 1$ // outputting the previous output element in A
   //(inserting an element into A)
   (d) $j = j + 1$
5. **return** S.

A = 2 0 1 2 2 1 2 0 2 0 0 0 1 11 0 2 1 1 2 1 0 1 0 0 2 2

B = 2 1 1 2 1 0 1 0 0 2 2 2 01 2 2 1 2 0 2 0 0 0 1 1 1 0.

The process of generating the first 12 elements of the edited sequence S is shown as follows:

2 11 2 1
2 0 1 2 2 1 2 0 0 2 0 0 1 1
= 0 1 2 2 1 1 1 2 0 1 1 1

Remark 1: The elements in an edited sequence are determined by the domain from which the base sequence A takes its elements. The domain from which the control sequence B takes its elements represents a set of strategies used for editing. Thus one may consider a general model for *Edit*(A, B) where A is a sequence whose elements taken from $Z_t = \{0, 1, \ldots, t-1\}$ and B is a sequence whose elements, taken from Z_r. We will continue to investigate the editing generator in this general setting in the future.

The properties of randomness of the edited sequence *Edit*(*A*, *B*) depend on the base sequence *A* and the control sequence *B*. In the rest of this paper, the case that both these two sequences are chosen as modified ternary *m*-sequences of degree *n*. The definition of a modified ternary *m*-sequence of degree *n* is as follows.

Definition 2: If a sequence is constructed by adding one zero into one of two zero runs of length $n - 1$ in a ternary *m*-sequence of degree *n*, then we call it a modified ternary *m*-sequence of degree *n*, *mm*-sequence for short.

From Property 1, we know that a modified ternary *m*-sequence of degree *n* has period 3^n, each element in Z_3 occurs exactly 3^{n-1} times and each *n*-tuple in $Z_3^{(n)}$ occurs exactly once in one period of 3^n.

Property 1: Let $S = Edit(A, B)$ where *A* and *B* are two *mm*-sequence *s* of degree *n*. Then we have the following matches.

$$i_3^n = 2.3^{n-1} \quad and \quad k_3^n = 2.3^{n-1}$$

$$i_{2.3}^n = 4.3^{n-1} \quad and \quad k_{2.3}^n = 4.3^{n-1}$$

$$i_{3.3}^n = 2.3^n \quad and \quad k_{3.3}^n = 2.3^n$$

Moreover, $2 \cdot 3^n$ is a period of *S*.

4.2.6.3 The Properties of Randomness

Derive the least period of the edited sequence *Edit* (*A*, *B*), a large lower bound for the linear span, ratio of linear span to period, and occurrences of symbols.

4.2.6.3.1 Period and Linear Span

Theorem 1: *Let $S = \{s_k\}$ be the edited sequence Edit(A, B) where both $A = \{a_i\}$ and $B = \{b_j\}$ are mm-sequence of degree n. Then Per(S), the least period of S, is either $2 \cdot 3^n$ or 3^n.*

Theorem 2: *Let S = Edit(A, B), where A and B are mm-sequences of degree n. Then LS(S), the linear span of S, is lower bounded by*

$$LS(S) > 3^{n-1}$$

Theorem 3: *Let $S = Edit(A, B)$ where both $A = (a_0, a1, ..., a3n^{-1})$ and $B = (b_0,b_1,...,b_3^{n-1})$ are mm-sequences of degree n. Then each element in Z^3 occurs at least 3^{n-1} times in one period of length $2 \cdot 3^n$ of S.*

For proves of Theorem1, Theorem2 and Theorem3 (Gong and Jiang, 2005).

4.2.6.4 Security Analysis

The security of the edited sequence $S = Edit(A, B)$ where $A = \{a_i\}$ and $B = \{b_j\}$ are mm-sequences of degree *n.* Assume that a portion of the key stream sequence *S,* without loss of generality, S_0^{m-1} is known.

The primitive polynomials that generate $\{a_i\}$ and $\{b_j\}$ are also assumed to be known. The objective of attacks is to find out the initial states of $\{a_i\}$ and $\{b_j\}$.

4.2.6.4.1 Parity-Check Attack

In (Gong and Jiang, 2005) they showed that one can recover the initial states of B and A by searching for the initial state of B and then carrying out a consistency test. Suppose a segment of key stream S_0^{m-1} is known.

Algorithm for searching sequence B:

- Generate candidate sequence B by guessing its initial state and then derive sequence B' from B by removing symbol 2. Furthermore, he computes B'' from B by removing all 0's.
- Compute sequence S' from S_0^{m-1} by removing s_i if $b'= 0$, $i =0,1,\ldots m-1$
- Let $t(i)$ be the ith index of symbol '1' in sequence B''. Partially recover A by setting $a_{t(i)} = s_i'$.
- Write a linear equation in terms of the initial state of A for each determined a_i in Step 3. Check the consistency of the linear system. If it is not consistent, go to step 1. Otherwise, compute the initial state of A.

The performance of this algorithm in each round, steps 1–3 need only $O(m)$. Step 4 needs $O(mn^3)$. Thus to be successful, it needs $O(3^n mn^3)$. In general, $m = O(n)$, it follows the running time is $O(3^n n^4)$, which is essentially the cost to search for the initial state of B.

5. TIME/MEMORY TRADEOFF ATTACK

Time-Memory Tradeoff (TMTO) attacks on stream ciphers are a serious security threat and the resistance to this class of attacks is an important criterion in the design of a modern stream cipher. TMTO attacks are especially effective against stream ciphers where a variant of the TMTO attack can make use of multiple data to reduce the off-line and the on-line time complexities of the attack.

5.1 INTRODUCTION

The problem of inverting a one-way function is a basic problem in cryptanalysis. For example, the problem of deducing the encryption key from a plaintext/ciphertext pair can be modeled as such. Two extreme solutions of the problem are exhaustive key search and the table attack. In exhaustive key search, the time complexity of the attack is N encryptions (where N is the number of possible keys), the memory requirement is negligible, and there is no preprocessing phase. In the table attack, the time complexity of the on-line phase of the attack is negligible, while the memory requirement is N memory cells, and a one-time preprocessing phase with time complexity of N encryptions is required.

In Hellman (1980) presented a trade-off between these two attacks. Hellman's attack uses pre-computed tables of total size M which allow to reduce the on-line time complexity T. The values of M and T satisfy the relation $N^2 = TM^2$ (up to logarithmic factors), and thus, a convenient choice of M and T is $M = T = N^{2/3}$. The attack also requires a pre-computation phase with time complexity of N encryptions. This phase is usually neglected in the treatment of TMTO attacks, since once it is performed, its results can be re-used for multiple attacks. In Biryukov and Shamir (2000) showed that if the attacker has access to multiple data points, the tradeoff curve obtained in Hellman's attack can be improved.

If the number of available data points is D, the attacker can produce an attack which satisfies $N^2 = TM^2D^2$, and has a preprocessing step of N/D operations. However, this attack is applicable only for $T \geq D^2$.

5.2 HELLMAN'S AND OECHSLIN'S TMTO ATTACKS

Start with the basic TMTO attack of Hellman (1980) against block ciphers.

Let f : {0, 1, . . . ,N−1} → {0, 1, . . . ,N − 1} be the function that the attacker tries to invert. The pre-processing phase of Hellman's attack consists of constructing several tables. To construct each table, the attacker chooses m random starting points, and from each starting point x, he computes the chain $SP = x, f(x), f^2(x) = f(f(x)), \ldots, f^t(x) = EP$, as shown in Figure 12.

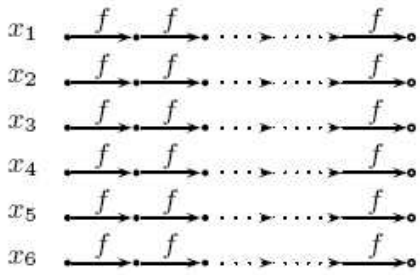


Figure 12 : Constructing Hellman's Table

The pairs (EP, SP) are stored in a table. The attacker constructs t such tables $T_0,\ldots,T_{t-1}$, each for a different function f_i that is usually a slight modification of the original f. In the online phase of the attack, the attacker is given $z = f(y)$ and has to find y. For all $0 \leq i \leq t - 1$, he applies f_i repeatedly to $f_i(y)$ (that can be easily computed given $f(y)$) to get the sequence $f_i(y), f_i^2(y),\ldots, f_i^t(y)$, for each new value the attacker checks whether the obtained value is an end point in the table T_i.

If a value appears as an end point in the table, the attacker takes the corresponding starting point and applies f_i sequentially, until $f_i(y)$ is reached. The point encountered just before $f_i(y)$ is supposedly y.

The time complexity of the attack is t^2 applications of f and t^2 database accesses (or $T = t^2$), and the memory required for the attack is $M = mt$. Each of the tables "covers" mt points (m chains of length t each), and thus all the t tables cover about mt^2 states. Since the tables should cover most of the N possible states, we have $N \approx (mt)t = mt^2$, and hence $N^2 = TM^2$ is the tradeoff curve obtained for this attack. The time complexity of the pre-processing phase is N, but as we noted before, this phase is usually neglected in the analysis of TMTO attacks.

In 2003, Oechslin presented a different method to construct the tables in the TMTO attack. In the new method, a single table (called the Rainbow table) is constructed. This time, mt starting points are chosen, and the constructed chains are of the form x, $f_0(x)$, $f_1(f_0(x))$, $f_2(f_1(f_0(x)))$, . . . , $f_{t-1}(\ldots(f_0(x)))$, where the functions f_i are the same as used in Hellman's scheme. This technique reduces the effect of collisions in the table, and hence allows to cover most of the state space by a single table.

On the other hand, the on-line phase of the attack is more complicated. First, the attacker checks whether $f_{t-1}(y)$ (computed from the given $f(y)$) appears as an end point in the table. If not, she computes $f_{t-1}(f_{t-2}(y))$ and checks whether this value appears in the table. If not, he computes $f_{t-1}(f_{t-2}(f_{t-3}(y)))$, and so on. Once an end point is encountered, the attack proceeds as Hellman's attack.

The time complexity of the attack is $1+2+\ldots+(t-1) \approx t^2/2$ applications of f and t database accesses, and the memory requirement is mt. Since this time most of the states are covered by the single table, we have $N \approx mt^2$, and hence the obtained tradeoff curve is $N^2 = 2TM^2$.

5.3 TIME-MEMORY-DATA TRADEOFF ATTACKS

Biryukov and Shamir (2000) showed that if the attacker has access to multiple data points, the tradeoff curve obtained in Hellman's attack can be improve[1]. It is important to note that the multiple data points stand for pairs of (unknown input, known output) to the function the attacker tries to invert, and the attacker is satisfied with finding any one of the unknown inputs. If the number of available data points is D, the attacker can construct tables that cover only N/D of the N states, and by the birthday paradox one of the data points is supposed to fall into the covered states. In the on-line phase of the attack, the attacker repeats the attack for all the data points, and once one of the points is covered by the table, the secret key can be retrieved.

In this case there are only t/D tables, and thus the time complexity of the attack remains t^2 while the memory requirement is reduced to mt/D. Since this time we have $N/D \approx mt^2$, the obtained tradeoff curve is $N^2 = TM^2D^2$. The time complexity of the pre-processing phase is reduced to N/D.

Biryukov and Shamir (2000) noted that the tradeoff curve $N^2 = TM^2D^2$ can be obtained only if $T \geq D^2$, since only in this case at least one "full" table consisting of m chains of length t each, where $mt^2 \approx N$, can be constructed. If $T < D^2$ and a single "smaller" table is used in the attack, the resulting time complexity is Dt and the memory requirement is m. However, since in this case we have $N/D \approx mt$, the obtained tradeoff curve is $TM^2D^2 = D^3tm^2 > D^2t^2m^2 = D^2(N/D)^2 = N^2$ (where we used the inequality $D > t$ following from the assumption $D^2 > T$).

Another technique allowing to extend the tradeoff curve $N^2 = TM^2D^2$ to the range $T < D^2$ is the BSW-sampling technique, introduced by Biryukov (*et. al.* 2001) and examined by

[1] The idea of exploiting multiple data points in TMTO attacks was first presented in Babbage (1995), Golic (1997). Biryukov and Shamir (2000) were the first to combine this idea with Hellman's attack.

Biryukov and Shamir (2000). This technique can be applied if for some easily distinguishable subset of the output values of *f*, the inputs leading to these outputs can be efficiently enumerated. For detailed example, look Dunkelman and Keller (2008).

5.4 TMTO ATTACKS ON STREAM CIPHERS

TMTO attacks on stream ciphers can be divided into two classes, according to the one-way function the attacker tries to invert: Inverting the internal state to output transformation, and inverting the (key, IV) to output transformation.

The first class of attacks is presented in (Babbage 1995, Golic 1997, Biryukov e*t. al* 2001, Biryukov and Shamir 2006) try to invert the function (Internal State → Output Prefix). In attacks of this class the trade-off curve provided by Hellman's basic attack can be improved using multiple data points. Each additional output bit provides the attacker with an additional data point, and if the attacker uses D such points, the tradeoff curve can be improved to $N^2 = TM^2D^2$, for $T \geq D^2$. Attacks of this class were used to break several stream ciphers, including A5/1(Golic 1997) and LILI-128 (Saarinen 2002).

The second class of attacks is presented in (Hong and Sarkar 2005) try to invert the function (Secret Key → Output Prefix).

The second class of attacks cannot use multiple data points from the same output stream, but the on-line time complexity of the attacks is less than that of exhaustive key search, independently of the size of the internal state. The security of a stream cipher against the second class of attacks is increased if the cipher uses a publicly known Initial Value (IV). If the attacker does not know the IV in advance, he cannot use it in the pre-processing phase of the TMTO attack. The current approach in this situation is to treat the IV as part of the secret key (Hong and Sarkar 2005, Biryukov e*t. al*. 2006).

As a result, the amount of secret key material is increased, but at the same time, the attack can use multiple data points. The data points are obtained from encryptions using the same secret key and different IVs. As a result, the tradeoff curve $N^2 = TM^2D^2$, for $T \geq D^2$, is again possible, where N is the number of possible keys multiplied by the number of possible IVs. In order to prevent second class of attacks, as well as other attacks, it is suggested in (Hong and Sarkar 2005, Biryukov e*t. al*. 2006) to require stream ciphers to have IV at least as long as the secret key.

6. TIME/MEMORY TRADEOFF ATTACK ON EDITING GENERATOR

In the preprocessing phase the attacker explores the general structure of the cryptosystem, and summarizes his findings in large tables. In the realtime phase, the attacker is given actual data produced from a particular unknown key, and his goal is to use the precomputed tables in order to find the key as quickly as possible.

- **N:** the size of the search space.
- **P:** the time required by the preprocessing phase of the attack.
- **M:** the amount of random access memory available to the attacker.
- **T:** the time required by the real-time phase of the attack.
- **D:** the amount of real-time data available to the attacker.

The size N of the search space is determined by the number of internal states (or output rate, in other words) of the bit generator, which can be different from the number of keys.

6.1 BG ATTACK (UNMODIFIED TRADEOFF)

In the preprocessing (offline) phase of the attack can be thought as preparing candidates for the keystream. In this phase, the states of the LFSRs generated randomly and also corresponding output values are evaluated. According to the assumption of Babbage (1995) and Golic (1997), (this attack will be called BG attack in short), the search space is divided into two equal parts as shown in below:

$$N = MT \qquad \text{By choosing } T = M, \qquad T = N^{1/2} \text{ and } M = N^{1/2}$$

According to the structure of the Editing Generator, the sequences in memory table can be declared in 3-tuple (x_i, y_i, z_i), where each x_i and y_i are the ternary bits in LFSR 1 and LFSR2, and each z_i is the corresponding output sequence. Because of the window size of the online phase is the *log(N)* bits, in the offline phase the output sequence must be the first *log(N)* bits of the output sequence. At the end of the preparation of the memory table, the table must be sorted in the increasing order of the output sequence. The structure of the memory table is as shown in Table 6.1.1.

Table 6.1.1: Memory Table

LFSR1	LFSR2	Output
3^n bits	3^n bits	2.3^{n-1} bits
3^n bits	3^n bits	2.3^{n-1} bits
.	.	.
.	.	.
.	.	.
3^n bits	3^n bits	2.3^{n-1} bits

In the real-time phase of the attack, *D* windows are selected from the captured sequence. Because of the window size is *log(N) bits,* then the attacker must pick $D + log(N) - 1$ generated bits, and this corresponds to the *D* possible windows of *log(N)* consecutive bits. It lookups each *log(N)* bits from the data in the sorted table.

As mention in the remark, the period of the ternary LFSRs is 3^n for the *mm-sequence* and *n* is the order of the LFSRs. If I suppose that the number of the 0s equally distributed in periods than the output rate will be 2.3^{n-1}. Then the search space will be $N = 3^{2.3^{n-1}}$. According to the assumption of the BG attack the search space, *N* will be divided as $N^{1/2}$ time (*T*) and $N^{1/2}$ memory (*M*). As far as the search space of the Editing Generator is concerned, the size of the preprocessing table, *M* will be $3^{3^{n-1}}$ and the time table will be $3^{3^{n-1}}$.

The structure of the windows in the online phase or time table in other words is as shown in Table 6.2.1.

Table 6.1.2: Time Table

$1 \ldots log(3^{2.3^{n-1}})$
$2 \ldots log(3^{2.3^{n-1}})+1$
.
.
.
$D \ldots D + log(3^{2.3^{n-1}}) - 1$

In the real-time phase, I must have at least $3^{2.3^{n-1}/2}$ length cipher text, where I will calculate the keystream from this sequence. After dividing the key sequence into windows, I will search the memory table according to the windows. If at least one possible window is found in the table, the secret key values for the LFSRs can be handled from the table, and if there are no false alarms, the attack is accomplished.

6.2 HELLMAN' S ATTACK

The starting point of the attack on stream ciphers is Hellman's original tradeoff attack on block ciphers, which considers the random function *f* that maps the key *x* to the ciphertext block *y* for some fixed chosen plaintext. In the application of Hellman's attack to the stream ciphers, the difference between the BG attack is increasing the size of the search space to increase the number of random sequences in the offline phase and also to increase the number of windows in the online phase. Hence, the time-memory tradeoff values are as shown in below:

$$N^2 = TM^2, \quad P = N, D = 1$$

By choosing $T = M$, Hellman gets the particular tradeoff point $T = N^{2/3}$ and $M = N^{2/3}$

$$N = 3^{2.3^{n-1}} \Rightarrow M = N^{2/3} = 3^{4.3^{n-2}}, T = 3^{4.3^{n-2}}$$

6.3 COMBINED VERSION OF THE BG ATTACK AND HELLMAN' S TMTO ATTACK

Shamir and Biryukov (2000), combined the two types of tradeoff attacks to obtain a new attack on stream ciphers (I will call this attack as SB Attack) whose parameters satisfy the following relation:

$$P=N/D \quad and \quad TM^2D^2 = N^2 \quad for\ any \quad D^2 \leq T \leq N.$$

Preprocessing Time : $P = N^{2/3}$
Attack Time : $T = N^{2/3}$
Disk Space : $M = N^{1/3}$
Available Data : $D = N^{1/3}$

For $N = 3^{2.3^{n-1}}$ the parameters $P = T = N^{2/3} = 3^{4.3^{n-2}}$, and $M = D = N^{1/3} = 3^{2.3^{n-2}}$

6.4 OUR ATTACK

In our assumption, the structure of the linear feedback shift registers must be concerned. Because, they all have periods and coefficients (according to the primitive polynomial structure) make them unique.

To clarify the structure of our attack, I can summarize the possible disadvantages of time memory tradeoff attack like that, first of all memory tables that is described in the former cases (both in BG, Hellman and BS attacks) are chosen randomly, which is contradictory to the LFSR structure (as far as the coefficients of primitive polynomials is concerned), because in some random values it is not possible to generate a stream like that. With the random values, I may not find any proper sequence in the time table or these random value sequences lead to high number false alarms. Secondly, the nature of the time memory tradeoff attack is to divide search space into time and memory phases, which I declared before, but again the LFSR-based stream cipher is concerned, the output of the cipher has some period and this period is proportional to the output rate. Because of this in the next part I will start from the Online Phase of the attack.

6.4.1 Online Phase

In the BG attack the search space determines with the thought that memory (M) and time (T) equal to each other but choosing the *mm-sequence* for the LFSRs, I have the memory size $M = 3^n$ and consideration of the equally distributed 0s, I have $2/3M$ output. That means for $M = 3^n$ I have 2.3^{n-1} outputs. Because when 0 reads in the control sequence, one bit in the base sequence can't read by the stream cipher. For the BG attack they will be divided equally and their details shown in below.

$$N = M.T = 3^{2.3^{n-1}}, \qquad M = 3^{3^{n-1}}, T = 3^{3^{n-1}}$$

Still, considering the two ternary LFSRs for random selections, each of their size is 3^n and the search space of the LFSRs, N will be $3^{3^{2n}}$.

But in our attack, with the dependency to output rate of the cipher text, I can't determine the M and T equal to each other.

Because as mentioned before LFSR-based stream cipher is depended with the period, that is proportional with the output rate. Although the length of the cipher text is $3^{N/2}$, the sequence not be unique in itself and windows also.

Definition 6.4.1.1 (Output Window): According to the structure of the stream cipher, subsequences that could give output (i.e, each 3-bit give 2-bit output in editing generator, according to this 3-bit accepted as an output window).

Continuation with this definition, assume that LSFR1 and LFSR2 have the same length. In one period of operation of the L1 and L2, the output window of the LFSRs 2.3^{n-1} and the control sequence L2 has the period 3^{n}. The relation of the LFSR sequence and the output window is given with the below theorem.

Theorem 4: For any LFSR based stream cipher f and for some $k \in Z^{+}$, which has a periodic input x there must be periodic output window, y with respect to ratio of $|x|$ to $|y|$.

$$lcm\ (|x|, |y|) = k.|x|$$

Proof. By the Definition 2.2 given in Section 2, LFSR with a length of L has a period in at most $P^n - 1$ bit for any given $P, n \in Z^{+}$.

Basis: For $k = 1$, this can be accepted as an output of LFSR or its permutation instead of compression $y=f(x)$, where x is LFSR sequence and y is output window. In other words this is guarantied that there is a linear mapping between x and y.

$$|y| = |x| \qquad \text{where} \qquad |x| = P^n - 1,\ |y| = P^n - 1$$

Recursive Step: Assume that $P = 2$ and $k \in Z^{+}$ For k = 2, …, n

For $k = 2$ $|x| = 2^{n-1}$, $\left|y\right| = \dfrac{2^n - 1}{2}$ $\Rightarrow$ Because of the $2^n - 1$ is odd number $y \notin Z^{+}$.

So there must be 1 uncompleted search subsequence.

The period of y will be *lcm* (|*x*| , |*y*|) = 2 |*x*|

. .
. .
. .

For k = n, the output period will be lcm (|*x*| , |*y*|) = *n* . |*x*|

Closure:

So that for k = n+1, *lcm* (|*x*| , |*y*|) = (n+1) |*x*|

We can generalize the result to the rational numbers. Assume that for some $k \in Q^+$ where $k = \frac{a}{b}$ $a, b \in Z^+$.

$$lcm\ (|x|, |y|) = b.\ |x|$$

6.4.2 Preprocessing Phase

Now, from the online phase I have the value of *T*. This value and with finding the *N,* I obtain the memory (*M*) size to generate offline tables.

As mentioned before using the random sequences in LFSRs are not giving us effective results. Because of that both for the two LFSRs, I use primitive polynomials. To generate all primitive polynomials over the order between 6 and 9, I use the internet online program[2] to assign the coefficients for the two LFSRs.

To determine the number of the primitive polynomials (*N'*) over GF(3), I use Euler's Totient Function.

[2] The programs detail links is given in section References.

This formula is an Euler product and is often written in the equivalent form

$$\varphi(n) = n.\prod_{p|n}\left(1-\frac{1}{p}\right)$$

with the product ranging only over the distinct primes p dividing n.

This formula gives us the number of the coefficients consistent from the primitive polynomials. And this N' equal to number of the primitive polynomials of degree n over GF(p).

$$N' = \varphi(p^n - 1)/n$$

Considering the n bit initial states of the LFSRs, I have 3^n different *n-tuple*. This *n-bit* initial state creates with probability 3^n.

$$a_{n+k} = c_{n-1}a_{n-1+k} + ... + c_1 a_{1+k} + c_0 a_k, k = 0,1,.$$

With these two component the search space

$$N \cong \binom{N'.3^n}{2} < 3^{3^n}$$

After finding the search space N, now I can determine the memory M. Using the memory tradeoff formula $N = M.T$. The value of T from the online phase and have N from the offline phase, I can find the memory to create the preprocessing tables.

$$M = \frac{N}{T} = \frac{\binom{3^n.N'}{2}}{2.3^n} = N'(3^n.N' - 1).2^{-2}$$

6.4.3 TMTO Attack on Bit Search Type Stream Ciphers

As mentioned in the previous chapters I explain three different types Time/Memory Tradeoff attacks, Babbage-Golic, Hellman and Biryukov-Shamir. Although they use the same method for the attack, the selected values for their time and memory are different. These different selections of the time and memory determine the search space *N*.

For the LFSR based stream ciphers, SSG has the LFSR period 2^n-1 and the output rate is 1/4, ABSG, MBSG and BSG has the LFSR period 2^n-1 and the output rate is 1/3, finally for the Editing Generator EG the LFSR period 3^n-1 and the output rate is 2.3^{n-1}. For their time and memory values on three TMTO attacks look table 6.4.3.1.

Table 6.4.3.1: TMTO Attack Values

	BG $N=M.T$		Hellman $N^2 = T.M^2$		BS $N^2 = T.M^2.D^2$	
	$M = N^{1/2}$	$T = N^{1/2}$	$M = N^{2/3}$	$T = N^{2/3}$	$M = N^{1/3}$	$T = N^{2/3}$
SSG	$2^{(2^n-1)/8}$	$2^{(2^n-1)/8}$	$2^{(2^n-1)/6}$	$2^{(2^n-1)/6}$	$2^{(2^n-1)/12}$	$2^{(2^n-1)/6}$
ABSG	$2^{(2^n-1)/6}$	$2^{(2^n-1)/6}$	$2^{(2^n-1)/9}$	$2^{(2^n-1)/9}$	$2^{(2^n-1)/9}$	$2^{2.(2^n-1)/9}$
MBSG	$2^{(2^n-1)/6}$	$2^{(2^n-1)/6}$	$2^{(2^n-1)/9}$	$2^{(2^n-1)/9}$	$2^{(2^n-1)/9}$	$2^{2.(2^n-1)/9}$
BSG	$2^{(2^n-1)/6}$	$2^{(2^n-1)/6}$	$2^{(2^n-1)/9}$	$2^{(2^n-1)/9}$	$2^{(2^n-1)/9}$	$2^{2.(2^n-1)/9}$
EG	$3^{3^{n-1}}$	$3^{3^{n-1}}$	$3^{4.3^{n-2}}$	$3^{4.3^{n-2}}$	$3^{2.3^{n-2}}$	$3^{4.3^{n-2}}$

6.4.4 Our Attack on Bit Search Type Stream Ciphers

Our attack's starting point is, there have subsequence bits that haven't read/see by the stream cipher in the online phase of the TMTO attack. That leads the number of unique windows fewer than the given standard TMTO formula results and the size of the *T* will be smaller. The stream ciphers that are shown in below will be examined according to this assumption.

6.4.4.1 SSG

For the Self Shrinking Generator, the stream cipher divides the bits into couples, read the bits that starts with 1 and outputs the next bit. Considering our base thought the output window of the stream cipher will be *L*/2. With these values the number of the windows given below.

$$|x| = 2^{n-1}, \qquad |y| = \frac{2^{n-1}}{2} \qquad \Rightarrow \qquad lcd\,(\,|x|\,,\,|y|\,) = 2\,.\,2^{n-1}$$

6.4.4.2 BSG, ABSG, MBSG

The BSG, ABSG and the MBSG algorithm's methods are different but they are similar with their output rate (for this algorithms, output rate is equal to the number of output window), *L*/3.

The BSG algorithm searches for the patterns $\bar{b}\,b^{i}\,\bar{b}$, where i ≥ 0 and b∈ {0, 1}. If i is equal to 0, then the output will be 0, else the output will be 1.

For the ABSG searching process same with the BSG plus if i is equal to zero the output bit will be $\overline{b}$, otherwise *b*.

The MBSG algorithm searches the subsequences $b0^i1$, where i ≥ 0 and b is selected for the output bit.

Thinking of their methods and output rate, the number of the windows using in our attack's online phase will be same for all three stream cipher. For the input of LFSR |*x*| and the keystream |*y*| the value T given in below.

$$|x| = 2^{n-1}, \qquad |y| = \frac{2^{n-1}}{3} \qquad \Rightarrow \qquad lcm(\,|x|\,,\,|y|\,) = 3\,.\,2^{n-1}$$

The table summarizes the results that our attack's values on the other stream ciphers.

Table 6.4.4.2.1: Our Attack Values on the Other Stream Ciphers

	Our Attack	
	$M = N / T,\ N \cong \binom{N'.P^n}{2}$	|x|: *input of LFSR,*|y|: *keystream* $T = lcm(\|x\|,\|y\|)$ *number of windows*
SSG	$N'.(2^{n-1}.N' - 1).2^{-2}$	2.2^{n-1}
ABSG	$N'.(2^{n-1}.N' - 1)/6$	3.2^{n-1}
MBSG	$N'.(2^{n-1}.N' - 1)/6$	3.2^{n-1}
BSG	$N'.(2^{n-1}.N' - 1)/6$	3.2^{n-1}
EG	$N'.(3^{n}.N' - 1).2^{-2}$	2.3^{n}

7. CONCLUSION

The main purpose of the Time/Memory Tradeoff attacks is to decrease the search complexity of the exhaustive search into data and time complexity. TMTO uses pre-computed tables of total size *M* which allow to reduce the online time complexity *T*. According to BG attack, the search space (*N*) divided into two equal parts which are online (*T*) and offline phase (*M*). The window size of the online phase is the *log(N)* bits, in the offline phase the output sequence must be the first *log(N)* bits of the output sequence.

For the Editing Generator the number of the windows stabilizes with the value 2.3^n, which is also the number of unique sequences that can be captured from known plaintext attack. That means it is unnecessary to divide the search space into some proportion of time and memory in LFSR-based stream ciphers because the time complexity is fixed. Being the case when the time complexity decreases the memory complexity also decreases. Also the random selection of input sequences for memory table can increase the false alarms, because of the possibility of randomly choosing non-producible sequence and also this part needs some pre-pre-work for finding primitive polynomials. Our attack shows that applying this method, the complexity of the attack proportional with the output rate. This is already equal with the security complexity of the given algorithm, $O(3^n)$. To sum up, under the consideration of reasons that I found, the TMTO attack's time complexity *T* is smaller than the given TMTO values that applied to LFSR-based stream ciphers.

REFERENCES

Armknecht F., Carlet C., Gaborit P., Kunzli S., Meier W. and Ruatta O., 2006. *Efficient Computation of Algebraic Immunity for Algebraic and Fast Algebraic Attacks*, In Eurocrypt 2006, May 28-June 1, Saint Petersburg, Russia.

Babbage, S., 2006. *"A Space/Time Trade-Off in Exhaustive Search Attacks on Stream Ciphers*". Newbury : Vodafone Ltd.

Biryukov A., Shamir A., 2000. *Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers*, Advances in Cryptology, proceedings of ASIACRYPT 2000, Lecture Notes in Computer Science, Springer-Verlag, Vol. 1976, pp. 1–13.

Biryukov A., Shamir A., Wagner D., 2001. *Real Time Cryptanalysis of A5/1 on a PC*, proceedings of FSE 2000, Lecture Notes in Computer Science, Springer-Verlag, Vol. 1978, pp. 1–18.

Biryukov, A., Mukhopadhyay, S., Sarkar, P., 2006. *Improved Time-Memory Tradeoffs with Multiple Data*, proceedings of Selected Areas in Cryptography 2005, Lecture Notes in Computer Science, Springer-Verlag, Vol. 3897, pp. 245–260.

Carlet, C., 2006. *"Boolean Functions for Cryptography and Error Correction Codes"*. Cambridge : Cambridge University Press.

Ergüler, İ., Anarım, E., 2006. *The Editing Bit-Search Generator*, Boğaziçi University Technical Report.

Golic, J. Dj., 1997. *Cryptanalysis of Alleged A5 Stream Cipher*, Advances in Cryptology, proceedings of EUROCRYPT '97, Lecture Notes in Computer Science, Springer-Verlag Vol. 1233, pp. 239–255.

Gong, G., Jiang, S., 2005. *The Editing Generator and Its Cryptanalysis*, International Journal of Wireless and Mobile.

Gouget, A., Sibert, H., 2004. *The Bit Search Generator*. The State of the Art of Stream Ciphers: Workshop Recofrd, pp. 60-68.
Gouget, A., Sibert, H., Berbain, C., Courtois, N., Debbraize, B., Mitchell, C., 2005 *Analysis of Bit-Search Generator and Sequence Compression Technique* : Fast Software Encryption.

Gouget, A., Sibert, H., 2006. *How to Strengthen Pseudo-random Generators by Using Compression* : EUROCRYPT 2006, Lecture Notes in Computer Sciences, Springer-Verlag, Vol. 4004, pp.129–146.

Hawkes, P. and Rose G., 2002. *The applicability of distinguishing attacks against stream ciphers*. in Proceedings of the Third NESSIE Workshop, 2002.
http://eprint.iacr.org/2002/142.pdf

Hellman, M.E., 1980. *A Cryptanalytic Time-Memory Tradeoff*, IEEE Transactions on Information Theory, Vol. 26, No. 4, pp. 401–406.

Hong, J., Sarkar, P., 2005. *Rediscovery of Time Memory Tradeoffs*, IACR Eprint Report 2005/090. Available on-line at http://eprint.iacr.org/2005/090.

Jansen, S.J.A., 2004. *Stream Cipher Design: Make your LFSR jump!* : Workshop Record ECRYPT-State of the Art of Stream Ciphers, pp. 94-108.

Meier, W., Staffelbach, O., 1995. *The Self-Shrinking Generator*. Lecture Notes in Computer Science, Springer-Verlag, Vol. 950, pp. 205-214.

Menezes, A.J., van Oorschot, P.C., Vanstone, S.A., 1997. *Handbook of Applied Cryptography* : CRC Press.

Oechslin, P., 2003. *Making a Faster Cryptanalytic Time-Memory Trade-Off*, Advances in Cryptology, proceedings of CRYPTO 2003, Lecture Notes in Computer Science, Springer-Verlag, Vol. 2729, pp. 617–630.

Rueppel, R.A., 1986. *Analysis* and. *Design of Stream Ciphers*, Springer-Verlag.

Saarinen, M.J.O., 2002. *A Time-Memory Tradeoff Attack Against LILI-128*, Lecture Notes in Computer Science, Springer-Verlag, Vol. 2365, pp. 129-134.

Stallings, W., 2003. *Cryptogrphy and the Network Security.* New Jersey : Prentice Hall, 3rd Edition

Zenner, E., Krause, M., Lucks, S., 2001. *Improved Cryptanalysis of the Self-Shrinking Generator.* Proceedings of the 6th Australasian Conference on Information Security and Privacy, pp. 21 – 35.

http://www.theory.csc.uvic.ca/~cos/inf/neck/PolyInfo.html

http://www.theory.cs.uvic.ca/~cos/gen/poly.html

ÖZGEÇMİŞ

Adı Soyadı: Elif KOPARAN ERMİŞ

Sürekli Adresi: Acıbadem mah. Pomak sok. Manolya apt. 7/4 Kadıköy/İST.

Doğum Yeri Tarihi: Antalya - 14.07.1982

Yabancı Dili: İngilizce

İlk Öğretim : Akdeniz Koleji - 1997

Orta Öğretim : Antalya Lisesi - 2000

Lisans : Mimar Sinan Üniversitesi - 2004

Yüksek Lisans : Bahçeşehir Üniversitesi

Enstitü Adı : Fen Bilimleri Enstitüsü

Program Adı : Bilgisayar Mühendisliği

Çalışma Hayatı :	Aras Kargo A.Ş.	2008 – Devam Ediyor
	Sirius Group Consulting	2007 – 2008