TWO VERSIONS OF THE STREAM CIPHER SNOW

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ERDEM YILMAZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

DECEMBER 2004

Approval of the Graduate School of Natural and Applied Sciences

_____
Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____
Prof. Dr. İsmet Erkmen
Head of Department

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

_____
Assoc. Prof. Dr. Melek D. Yücel
Supervisor

Examining Committee Members

Prof. Dr. Yalçın Tanık (METU,EE)_____

Assoc. Prof. Dr. Melek D. Yücel (METU,EE)_____

Prof. Dr. Murat Aşkar (METU,EE)_____

Assoc. Prof. Dr. Ferruh Özbudak (METU,MATH)_____

Asst. Prof. Dr. A. Özgür Yılmaz (METU,EE)_____

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

Name, Last Name: Erdem Yılmaz

Signature :

ABSTRACT

TWO VERSIONS OF THE STREAM CIPHER SNOW

Yılmaz, Erdem

M.Sc., Department of Electrical and Electronics Engineering

Supervisor: Assoc. Prof. Dr. Melek D. Yücel

December 2004, 60 pages

Two versions of SNOW, which are word-oriented stream ciphers proposed by P. Ekdahl and T. Johansson in 2000 and 2002, are studied together with cryptanalytic attacks on the first version. The reported attacks on SNOW1.0 are the “guess-and-determine attack”s by Hawkes and Rose and the “distinguishing attack” by Coppersmith, Halevi and Jutla in 2002. A review of the distinguishing attack on SNOW1.0 is given using the approach made by the designers of SNOW in 2002 on another cipher, SOBER-t32. However, since the calculation methods for the complexities of the attack are different, the values found with the method of the designers of SNOW are higher than the ones found by Coppersmith, Halevi and Jutla.

The correlations in the finite state machine that make the distinguishing attack possible and how these correlations are affected by the operations in the finite state machine are investigated. Since the substitution boxes (S-boxes) play an important

role in destroying the correlation and linearity caused by Linear Feedback Shift Register, the s-boxes of the two versions of SNOW are examined for the criteria of Linear Approximation Table (LAT), Difference Distribution Table (DDT) and Auto-correlation Table distributions.

The randomness tests are performed using NIST statistical test suite for both of the ciphers. The results of the tests are presented.

Keywords: Stream Cipher, SNOW, S-box, Distinguishing Attack, Randomness Tests

ÖZ

SNOW AKAN ŞİFRESİNİN İKİ UYARLAMASI

Yılmaz, Erdem

Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü

Tez yöneticisi: Doç. Dr. Melek D. Yücel

Aralık 2004, 60 sayfa

P. Ekdahl ve T. Johansson tarafından 2000 ve 2002 yıllarında önerilen ve kelime odaklı akan şifrelerden olan SNOW'un iki uyarlaması ve birinci uyarlamasına yapılmış kriptanaliz atakları üzerinde çalışılmıştır. SNOW1.0 için rapor edilen bu ataklar, 2002 yılında, Hawkes ve Rose'un "tahmin et ve belirle", Coppersmith, Halevi and Jutla'nın "ayırt etme" ataklarıdır. SNOW'un tasarımcılarının 2002'de başka bir şifre, SOBER-t32, için yaptığı yaklaşım kullanılarak, bu tezde SNOW1.0'e yapılmış olan ayırt etme atağı yinelenmiştir. Fakat atak karmaşıklıklarını hesaplama yöntemleri farklı olduğundan, SNOW'un tasarımcılarının yöntemiyle bulunan değerler Coppersmith, Halevi ve Jutla'nın bulduklarından fazladır.

Ayırt etme atağını mümkün kılan sonlu durum makinesindeki benzeşmeler ve bu benzeşmelerin Sonlu Durum Makinesi'ndeki işlemlerden nasıl etkilendiği araştırılmıştır. Yerleştirme kutuları Doğrusal Geri Beslemeli Kaydımalı Yazdırgaç'ın

sebep olduğu doğrusallık ve benzeşmelerin ortadan kaldırılmasında önemli bir rol oynadığı için, SNOW'un yerleştirme kutularının özellikleri, Doğrusal Yaklaşım Tablosu, Fark Dağılımı Tablosu ve Oto-korelasyon Tablosu dağılımı ölçütlerine göre incelenmiştir.

Ayrıca NIST istatiksel test ortamı kullanılarak her iki algoritma için de rassallık testleri yapılmış ve test sonuçları sunulmuştur.

Anahtar Kelimeler: Akan Şifre, SNOW, Yerleştirme Kutusu, Ayırt Etme Atağı, Rassallık Testleri

To My Family

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my advisor, Assoc. Prof. Dr. Melek D. Yücel for her guidance, encouragement and support in every stage of this research.

I am also grateful to my home-mates and colleagues for their encouragement and support.

Finally, I would like to express my deep gratitude to all who have encouraged and helped me at the different stages of this work.

And my parents and sister, I thank them for everything.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Traditionally, different systems have been used by governments and the military forces to prevent national or military secrets to be revealed by enemies. Today, also when we use our credit card or an ATM, watch pay-per-view channels, or buy something on the web, some security systems are used to offer protection. Achieving security requires some technical skills, which are provided through cryptography. There are two different types of cryptographic techniques, which are called *symmetric key cryptography* and *public key cryptography*.

In public key cryptography, the keys are not symmetric. There are two types of keys, which are called the *public key* and the *private key*; which belong to the same person. The public key is used to encrypt a message, but the message can only be decrypted by the person who has the private key. These keys are generated so that, one can not obtain the private key from the public key.

In symmetric key cryptography, encrypting and decrypting operations are done using the same key both on the sender and receiver sides. These types of algorithms are also called as secret key algorithms. The key in symmetric key cryptography is the single critical parameter that is to be kept secret. There are two types of symmetric key algorithms: *stream ciphers* and *block ciphers.*

Block ciphers encrypt the plaintext in blocks. These types of algorithms take the plaintext in fixed-length blocks as input and give the ciphertext again in fixed-length blocks as output. In most of the cases, the block lengths for the plaintext and the ciphertext are the same. Stream ciphers encrypt the plaintext character by character (or bit by bit). These types of algorithms produce a stream of bits, which is called the keystream of the algorithm. This keystream is used to encrypt or decrypt the plaintext or the ciphertext.

Stream ciphers have several properties that make them suitable for use in telecommunication applications. But apart from the security tried to obtain, the main property that makes stream ciphers distinguishable from block ciphers is that they are in general fast and have low hardware complexity.

Before 1999, there was not much interest on stream ciphers and most of the stream ciphers were bit-oriented, however they do not perform well in software. After the call for the NESSIE[1] project, the interest on stream ciphers has started to rise significantly.

An open call in March 2000 led to the submission of forty cryptographic primitives to the NESSIE project. There were five stream ciphers submitted to NESSIE.

- LEVIATHAN
- SOBER-t16 and SOBER-t32
- BMGL
- LILI-128
- SNOW

Among these, four of them, SOBER-t16, SOBER-t32, LILI-128 and SNOW, are based on linear feedback shift registers.

The NESSIE evaluation process was an open process which means that apart from the evaluations made by NESSIE partners, NESSIE project welcomes comments and evaluations from all over world. The evaluation process was divided into two phases and after the second phase, none of the ciphers was recommended by NESSIE. Because, every stream cipher was exposed to an attack faster than exhaustive search.

The stream cipher SNOW was submitted by Patrik Ekdahl and Thomas Johansson, to provide not only the security aspects but also a good performance in software. The first version of SNOW (SNOW1.0) passed the first phase of the NESSIE evaluation, but could not pass the second phase due to two attacks reported. One was a guess-and-determine attack [Hawkes, Rose; 2002] and the other was a distinguishing attack [Coppersmith, Halevi, Jutla; 2002]. These attacks

[1] The NESSIE project is a three year project (2000-2002) that is funded by the European Union's *Fifth Framework Programme*. The main objective of the NESSIE project is to put forward a portfolio of strong cryptographic primitives of various types.

revealed some weaknesses in the design and a new improved version of the cipher, SNOW2.0, was developed. Although the reasons for some weaknesses in SNOW1.0 are known, the exact reasons for the strong correlations in the FSM are not known.

In this study, the reasons behind the changes applied to SNOW1.0 to improve the security are studied and searched. Among the changes that have been made, one was the substitution box. So, both of the substitution boxes are examined. Their LAT (Linear Approximation Table), DDT (Difference Distribution Table) and Auto-correlation Tables are formed and comparisons are made. Some tests are performed to see how the correlations in the FSM are affected by the changes in the s-box and other operations in the FSM, and the reasons for large correlations are investigated.

In [Ekdahl, Johansson; 2002b], Ekdahl and Johansson mount a distinguishing attack on Sober-t32 [Hawkes, Rose; 2000] whose structure is very similar to SNOW1.0. Their distinguishing attack is very similar to the one applied on SNOW1.0 [Coppersmith, Halevi, Jutla; 2002]. In this study, we give a detailed review of the distinguishing attack on SNOW1.0 using the description of the attack on SOBER-t32 [Ekdahl, Johansson; 2002b]. However, since the methods of calculation for the complexities of the attack are different, the values found with the method used in [Ekdahl, Johansson; 2002b] are higher than the ones in [Coppersmith, Halevi, Jutla; 2002].

Randomness tests are performed using NIST statistical test suite for both of the ciphers. The results of the tests are presented and compared.

Chapter 1 gives an overview of the cryptography and a summary of the thesis.

In Chapter 2, an introduction to stream ciphers and Boolean functions are given. The properties of LFSRs and s-boxes, which play important roles in a stream cipher system, are described.

In Chapter 3, the information about the structures of two versions of SNOW is given.

In Chapter 4, the examination of s-boxes and finite state machine is presented. A review of the distinguishing attack is given using the approach in [Ekdahl,

Johansson; 2002b] made for SOBER-t32; together with the search for the correlations in the FSM while changing the operations in the FSM. The results of the randomness tests are presented.

In Chapter 5, concluding remarks are discussed along with future work for possible improvements.

CHAPTER 2

STREAM CIPHERS

A *stream cipher* is a cryptographic technique that encrypts binary digits individually, using a transformation that changes with time. This is contrasted to a *block cipher*, where a block of binary data is encrypted simultaneously, with the transformation usually being constant for each block.

In specific applications, stream ciphers are more appropriate than block ciphers [NESSIE Sec. Rep.; 2003 *and* Robshaw, RSA Lab. Tech. Rep.; 1995] :

- Stream ciphers are generally faster than block ciphers, especially in hardware.
- Stream ciphers have less hardware complexity and less memory requirements for both hardware and software.
- Stream ciphers process the plaintext character by character, so no buffering is required to accumulate a full plaintext block (unlike block ciphers).
- *Synchronous* stream ciphers (Section 2.1) have no error propagation.

Most stream ciphers are based on simple devices that are easy to implement and run efficiently. A common example of such a device is the linear feedback shift register (LFSR) [Rueppel;1986]. Such simple devices produce predictable output given some previous output. This is due to the linear property of the device. Therefore, in order to use LFSRs in cryptographical primitive, and particularly in a stream cipher, the linearity must be destroyed. Thus, Boolean functions and S-boxes are introduced together with their basic properties.

2.1 Stream Ciphers

Stream ciphers are divided into two sets called *synchronous* and *self-synchronous.*

Definition 2.1: In a synchronous stream cipher, the keystream is generated independently of the plaintext and the ciphertext.

A synchronous stream cipher can be represented by a finite state machine, as illustrated in Figure 2.1. Now let's take a look at the encryption process. It can be described at time $t \geq 0$ by the equations

$$\begin{aligned} \sigma_{t+1} &= f(\sigma_t, k), \\ z_t &= g(\sigma_t, k), \\ c_t &= h(z_t, m_t), \end{aligned} \tag{2.1}$$

where σ_0 is the *initial state* and may depend on the key *k*. *f* is the *next-state function, g* is the function which produces the *keystream* z_t, $t \geq 0$ and h is the output function which combines the keystream and plaintext to produce the ciphertext c_t, $t \geq 0$.

One of the most common types of synchronous stream cipher is the binary additive stream cipher. A binary additive stream cipher is a synchronous stream cipher where the plaintext, ciphertext, and keystream all are binary sequences, and furthermore, the encryption function *h* (output function) is the simple XOR operation, i.e.,

$$c_i = m_i \oplus z_i \ . \tag{2.2}$$

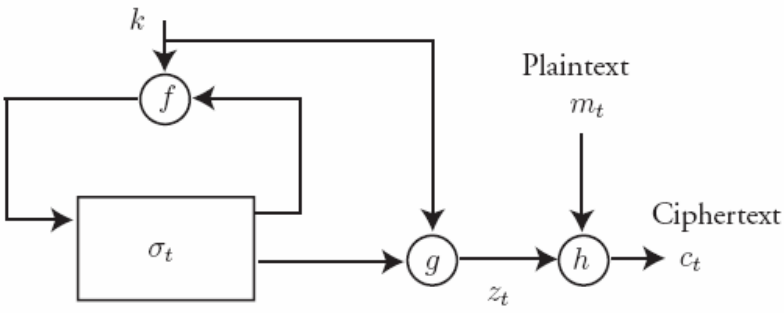


Figure 2.1 : General structure of a synchronous stream cipher

Since the keystream from a synchronous keystream generator neither depends on the plaintext, nor on the ciphertext, there is no error propagation. That is, if a certain symbol in the ciphertext has been corrupted by transmission error, the rest of the ciphertext will not be affected.

To be able to decrypt correctly, the receiver has to be in perfect synchronization with the sender. If synchronization is lost, the decryption will not work correctly and the information is lost. Thus, there is a need for mechanisms for detecting lost synchronization and for re-initialization. Due to the synchronization property, synchronous stream ciphers are vulnerable to active attacks, where an adversary can insert or delete symbols to the ciphertext sequence. It will also make it possible for an adversary to change some of the ciphertext symbols and still create a valid ciphertext sequence. Thus, we need to use additional techniques to guarantee message authentication.

A frame based communication protocol can be used to defeat this synchronization problem. In this protocol, message sequence is first divided into smaller *frames* which are numbered with a frame number. We then add a feature called an *Initialization Value (IV),* which is publicly known and used in the initialization of the stream cipher together with the secret key. Now, with a fixed key but with a changing IV, the stream cipher will produce different sequences of keystream material for each IV. For each frame the receiver tries to decrypt, he looks at the public frame number attached to the frame of encrypted information and pre-initializes the stream cipher with the new frame number as IV and the secret key, and then decrypts the information. If synchronization is lost for a single frame, it will only affect a small amount of information, until a new frame arrives and he can synchronise.

Definition 2.2 : A self-synchronizing or asynchronous stream cipher is a stream cipher where the keystream is generated as a function of the key, k, and at most t previous ciphertext symbols.

As for synchronous stream ciphers, we can define a state σ_i also for a self-synchronizing stream cipher. Here the state is taken as the t previous ciphertext symbols,

$$\sigma_i = (c_{i-1}, c_{i-2}, ..., c_{i-t}) . \tag{2.3}$$

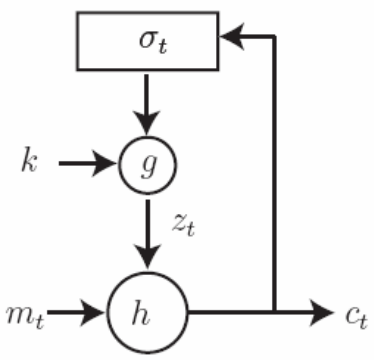


Figure 2.2 : General structure of a self-synchronizing stream cipher

The ith keystream symbol, z_i, is generated as a function, denoted by *g,* of the initial state and the key,

$$z_i = g(\sigma_i, k). \tag{2.4}$$

From the definition of the state, we observe that we need to have an initial state defined by a initial value for i < 0. This initial value may be public. The principle of self-synchronizing stream ciphers is illustrated in Figure 2.2.

Since the state depends only on the last t ciphertext symbols, the key-stream will automatically be re-synchronized after a limited time, if some ciphertext symbols are lost during transmission. If a single error occurs on the channel, the decryption of the next t ciphertext symbols will be affected. Thus, the error propagation is worse for self-synchronizing stream ciphers compared with synchronous stream ciphers. The self-synchronization property will also make it harder to detect insertion or deletion of false ciphertext digits by an active adversary. Thus, there is a need for additional methods to guarantee message authentication.

2.2 Linear Feedback Shift Registers

Linear Feedback Shift Registers (LFSRs) are the most commonly used devices as keystream generators. LFSRs produce sequences having large periods and good statistical properties, they are well-suited for hardware implementations

and there are mathematical techniques to analyse them. For this reason it will be a good start to use LFSRs while designing a stream cipher.

A linear feedback shift register produces a sequence, **s** = $s_0,s_1,s_2,....$, satisfying the linear recurrence function,

$$s_n = \sum_{j=1}^{l} c_j s_{n-j}, \qquad n = l, l+1,...., \tag{2.5}$$

where *l* is the length of the LFSR, $s_i \in F_q, i \geq 1$. The general form of a linear feedback shift register is illustrated in Figure 2.3. An LFSR consists of *l* delay elements, where each delay element, also called *stage*, can store an element, or digit, in F_q. The *l* stages, ($s_{n-l},s_{n-l+1},.....,s_{n-1}$), are together called the *state* of the shift register. Each *feedback coefficient* c_j, *j=1,....,l* is an element in F_q. Using the feedback coeffients, we define the *feedback polynomial,* or connection ploynomial, to be $g(x) = 1 - c_1 x + c_2 x^2 + ... + c_{l-1} x^{l-1} + c_l x^l$. As an alternative to the feedback polynomial one can use the *characteristic polynomial,* $f(x) = x^l - c_1 x^{l-1} - c_2 x^{l-2} - ..... - c_{l-1} x - c_l$. The first *L* output symbols ,$s_0,s_1,s_2,....,s_{l-1}$ are initially loaded into the *l* stages. These symbols loaded into the LFSR, together form the *initial state.*

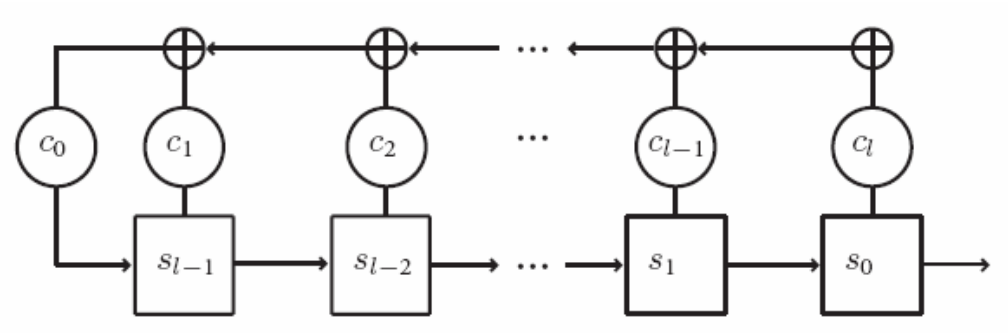


Figure 2.3 : Linear Feedback Shift Register of length *l*

Since there are only a finite number of possible states, q^l,the sequence produced by the LFSR must repeat itself after a finite period, i.e., for every starting state we can find a T such that $s_t = s_{t+T}, t \geq 0$. The period depends on the

properties of the feedback polynomial, and for our use in stream ciphers, we confine ourselves to the following definition and theorem regarding the period.

Definition 2.3 : The feedback polynomial $g(x)$ is called *irreducible* if it can not be written as the product of two polynomials with coefficients in F_q and positive degree. If the root *x* of an irreducible polynomial $g(x)$ of degree *l* is a generator of the multiplicative group of all the non-zero elements of F_{q^l}, $g(x)$ is called *primitive* polynomial.

Theorem 2.1 : Consider an LFSR of length *l* and feedback polynomial $g(x)$, where $g(x)$ is a primitive polynomial of degree *l* over F_q.Then each of the $q^l - 1$ non-zero initial states of the LFSR produces the sequence with period $q^l - 1$.

Then, all possible states except the all zero state will appear during a period. An LFSR with a primitive feedback polynomial is also called *maximum-length* LFSR, and the sequence generated is called a *maximum-length* LFSR.

Definition 2.4 : The *linear complexity* of a sequence **s** = $s_0,s_1,s_2,....,s_i \in F_q$, denoted $L(s)$, is the length of the shortest LFSR that generates the sequence.

Given at least $2L(s)$ output symbols of an LFSR, the linear complexity can be determined with the Berlekamp-Massey algorithm [Massey; 1969]. Thus, LFSRs have good statistical properties and can be a useful block for stream ciphers, but some further work is required to prevent attacks that make use of the inherent linearity (Section 2.3).

There are many different considerations that we must keep in mind when we consider the suitability of a keystream generated by some stream cipher. Some of these considerations are period, linear complexity and statistical measure of the keystream.

- **Period :** If the period of the keystream is too short, then different parts of the plaintext will be encrypted with the repeating keystream and this causes a severe weakness. A good assesment is necessary regarding the period of the keystream while designing a stream cipher. Practically, the period should

be long enough so that the same portion of the keystream is not used more than once.

- **Linear Complexity :** It is an indication for how difficult a sequence might be to replicate. While a high linear complexity is a necessary condition, the following example shows that it is not a sufficient condition. Consider the sequence consisting of a single 1 with the remaining bits set to 0. In this case the linear complexity is equal to the length of the sequence. However it is clear that as a keystream such a sequence is useless since all bits except the starting bit are zero.

  Rueppel [Rueppel; 1984] proposes the use of the *linear complexity profile* in the analysis of stream ciphers. After each bit is added to the keystream the linear complexity of the sequence seen so far is calculated; the value of the linear complexity can be plotted against the number of bits that have been examined, thereby giving a 'profile' of the sequence. Rueppel established that the linear complexity profile for a perfectly random source closely follows the line $y = x/2$.

- **Statistical measures :** A wide range of different statistical tests can be applied to a sequence to assess how well it was generated by a perfectly random source. (Appendix A).

But note that properties like large period, large linear complexity and a good statistical behaviour are necessary but not sufficient conditions for a stream cipher to be considered cryptographically secure.

2.3 Introducing nonlinearity

If an LFSR is used as a sequence generator in a stream cipher system, nonlinearity has to be introduced to the output stream. There are a number of standard tecniques that can be used to form a highly nonlinear output sequence. But before mentioning these techniques, Boolean functions and substitution boxes, which have a great role in destroying the linearity caused by LFSRs, will be presented.

2.3.1 Boolean Functions

A Boolean function, $f(x)$ takes a binary vector $x=(x_1,x_2,...,x_n), x_i \in \mathrm{F}_2$ $1 \le i \le n$, as input and outputs one bit, i.e.,

$$f : \mathrm{F}_2^n \to \mathrm{F}_2$$

The vectors in F_2^n ,in ascending lexicographic order, are denoted by $\alpha_0, \alpha_1, ..., \alpha_{2^n-1}$. As vectors in F_2^n and integers in $\left[0, 2^n-1\right]$ have a natural one-to-one correspondence, it allows us to switch from a vector in F_2^n to its corresponding integer in $\left[0, 2^n-1\right]$, and vice versa.

The *sequence* of f is defined as $((-1)^{f(\alpha_0)}, (-1)^{f(\alpha_1)}, ..., (-1)^{f(\alpha_{2^n-1})})$, while the *truth table* of f is defined as $(f(\alpha_0), f(\alpha_1), ..., f(\alpha_{2^n-1}))$. A Boolean function $f(x)$ can also uniquely be expressed in algebraic normal form.

$f(x_1,x_2,....,x_n) = a_0 + a_1x_1 + ... + a_nx_n + a_{12}x_1x_2 + a_{13}x_1x_3 + ... + a_{12....n}x_1x_2.....x_n$,

where addition and multiplication are in F_2. For a Boolean function

$$f(x_1,x_2,x_3) = x_1 + x_1x_2 + x_2x_3$$

the truth table is shown in Table 2.1.

Definition 2.5 : The algebraic degree of a Boolean function f is defined to be the number of variables in the highest order product of f, when f is written in algebraic normal form. The algebraic degree of f is denoted by $\deg(f)$.

We call $f(x) = a_1x_1 + ... + a_nx_n + c$ an *affine function*, where $x=(x_1,x_2,...,x_n)$ and $a_j, c \in GF(2)$. In particular, f will be called a *linear function* if $c=0$. The sequence of an affine (linear) function will be called an affine (linear) sequence.

An n variable Boolean function f is *balanced* if the output column in the truth table contains an equal number of 0's and 1's. Alternatively, f is balanced if $P(f(x)=0) = P(f(x)=1) = 1/2$, when x is chosen uniformly in F_2^n.

Table 2.1 : The Truth Table of the Boolean function $f(x_1, x_2, x_3) = x_1 + x_1x_2 + x_2x_3$

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
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

F_2^n is the set of all Boolean functions in n variables, and let A_n be the set of all affine functions in n variables. The *Hamming distance* between two functions $f(x), g(x) \in F_2^n$ is defined as,

$$d_H(f, g) = \left| \left\{ x \mid f(x) \neq g(x), x \in \mathrm{F}_2^n \right\} \right|. \tag{2.6}$$

Definition 2.6 : *Nonlinearity* of a Boolean function $f(x)$, denoted by N_f, is the Hamming distance to the nearest affine function, i.e.,

$$N_f = \min_{g \in A_n} d_H(f, g). \tag{2.7}$$

A high nonlinearity is desirable property since it will decrease the correlation between the output and the input variables or a linear combination of input variables. This property is very important while designing a nonlinear combining generator (Section 2.3.3).

Definition 2.7 : An n variable Boolean function is defined to be *t*-th order *correlation immune* , if for any *t*-tuple of independent identically distributed binary random variables $X_{i_1}, X_{i_2}, ..., X_{i_t}$, we have

$$I(X_{i_1}, X_{i_2}, ..., X_{i_t}; Y) = 0, \quad 1 \le i_1 < i_2 < ... < i_m \le n, \tag{2.8}$$

where $Y = f(X_1, X_2, ..., X_n)$, and $I(X;Y)$ denotes the mutual information.

Siegenthaler [Siegenthaler; 1984] showed that there is a tradeoff between the algebraic degree and the order of correlation immunity.

Theorem 2.2 : Let $f(x)$ be a balanced Boolean function in n variables of algebraic degree d which is t-th order correlation immune. Then the following upper bound [Siegenthaler; 1984] must hold

$$\begin{aligned} &d + t \le n - 1 \quad if \ 1 \le t \le n - 2 \\ &d + t \le n \qquad if \ t = n - 1. \end{aligned} \tag{2.9}$$

A Boolean function that is both balanced and t-th order correlation immune is called a *t-resilient* function.

The properties above are often investigated through Walsh Transform.

Definition 2.8 : For a Boolean function, $f : \mathrm{F}_2^n \to \mathrm{F}_2$, the *Walsh Transform* of $f(x)$ is defined to be the real valued function $F(w)$ over the vector space F_2^n given by

$$F(w) = \sum_x (-1)^{f(x) \oplus w \cdot x} \tag{2.10}$$

where the dot product (sum of component-wise products) of vectors x and w is defined as $x \cdot w = \langle x, w \rangle = x_1 w_1 + ... + x_n w_n$.

The component-wise product of two vectors x and w is a vector denoted by $x * w$.

The Hamming distance between a Boolean function $f(x)$ and an affine function $g(x) = w \cdot x \oplus c$, where $c \in \mathrm{F}_2$, can be calculated with the Walsh transform as

$$d_H(f, g) = 2^{n-1} - \frac{(-1)^c F(w)}{2}. \tag{2.11}$$

Thus the nonlinearity of $f(x)$ can be obtained from the Walsh transform as

$$N_f = 2^{n-1} - \frac{1}{2}\max_w |F(w)|. \tag{2.12}$$

Theorem 2.3 : A Boolean function is *t*-th order correlation immune if and only if

$$F(w) = 0, \quad \forall w \in \mathrm{F}_2^n \,|\, 1 \le w_H(w) \le t, \tag{2.13}$$

where $w_H(w)$ is the Hamming weight of w, i.e., the number of nonzero positions in w.

A Boolean function $f(x)$ is balanced if and only if $F(0) = 0$. Hence we see that Walsh transform is an important tool when investigating properties of Boolean functions.

Definition 2.9 : Let f be a function on F_2^n. For a vector $\alpha \in \mathrm{F}_2^n$, denote by $\xi(\alpha)$ the sequence of $f(x \oplus \alpha)$. Thus $\xi(0)$ is the sequence of f itself and $\xi(0) * \xi(\alpha)$ is the sequence of $f(x) \oplus f(x \oplus \alpha)$. Define the auto-correlation of f with a shift α by

$$\Delta(\alpha) = \langle \xi(0), \xi(\alpha) \rangle, \tag{2.14}$$

which is also equal to $\sum_{x \in V_n} (-1)^{f(x)} (-1)^{f(x \oplus \alpha)}$.

Definition 2.10 : The *Sylvester-Hadamard matrix* (or *Walsh-Hadamard matrix*) of order 2^n, denoted by H_n, is generated by the recursive relation

$$H_n = \begin{pmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{pmatrix}, \; n = 0,1,2,...., \; H_0 = 1. \tag{2.15}$$

Each row (column) of H_n is a linear sequence of length 2^n.

Let ξ be the sequence of a function f on F_2^n. Then, another definition of nonlinearity of f, N_f can be calculated by

$$N_f = 2^{n-1} - \frac{1}{2}\max\{|\langle \xi, l_i \rangle|, 0 \le i \le 2^n - 1\} \tag{2.16}$$

where l_i is the ith row of H_n, $i = 0,1,...,2^n - 1$, and

$$(\Delta(\alpha_0), \Delta(\alpha_1), ..., \Delta(\alpha_{2^n-1}))H_n = (\langle \xi, l_0 \rangle^2, \langle \xi, l_1 \rangle^2, ..., \langle \xi, l_{2^n-1} \rangle^2) \quad (2.17)$$

where α_i is the binary representation of an integer i and l_i is the ith row of H_n, $i = 0,1,...,2^n - 1$. And also that note $\langle \xi, l_i \rangle$ is equal to Walsh transform of f, $F(w)$.

As a conclusion Boolean functions should have some properties when used as a combining function (Section 2.3.3) in a stream cipher sysytem:

Algebraic degree A high algebraic degree is desirable since it increases the linear complexity of the resulting keystream.

Nonlinearity A high nonlinearity gives a weaker correlation between the input variables and the output variable and increases the resistance to correlation attacks.

Correlation immunity A high correlation immunity forces the attacker to consider several input variables jointly and thus decreaes the vulnerability of divide-and-conquer attacks.

2.3.2 S-Boxes

An s-box (Substitution box) can be considered as a vector output Boolean function.

An $n \times m$ s-box is a mapping from F_2^n to F_2^m, i.e., $F = (f_0, f_1, ..., f_m)$, where n and m are integers with $n \geq m \geq 1$ and each component function f_j is a function on F_2^n.

Lemma 2.11 : A function $F = (f_1, f_2, ..., f_m)$, where each f_i, $1 \leq i \leq m$, is a Boolean mapping $\mathrm{F}_2^n \to \mathrm{F}_2$, is uniformly distributed (balanced) if and only if all nonzero linear combinations of $f_1, f_2, ..., f_m$ are balanced.

The concept of nonlinearity can be extended to the case of an s-box.

Definition 2.12 : The standard definition of the *nonlinearity* of $F = (f_0, f_1, ..., f_m)$ is

$$N_F = \min_g \{N_g \mid g = \bigoplus_{j=1}^{m} c_j f_j,\ c_j \in GF(2), g \neq 0\} \quad (2.18)$$

LAT (Linear Approximation Table), DDT (Difference Distribution Table) and *Auto-Correlation Tables* are some of the criterions that are used to measure the security of s-boxes. The tables of an $n \times m$ s-box is a $2^n \times 2^m$ matrix.

First we introduce three more notations, $k_j(\alpha), \Delta_j(\alpha)$ and η_j, associated with an s-box $F = (f_1, f_2, ..., f_m)$. Then we will define the three tables mentioned above.

Definition 2.13: Let $F = (f_1, f_2, ..., f_m)$ be an $n \times m$ s-box, $\alpha \in \mathrm{F}_2^n$, $j = 0,1,...,2^m - 1$ and $\beta_j = (b_1, ..., b_m)$ be the vector in F_2^m that corresponds to the binary representation of j. In addition set $g_j = \bigoplus_{u=1}^{m} b_u f_u$ be the jth linear combination of the component functions of F. Then we define

1. $k_j(\alpha)$ as the number of times $F(x) \oplus F(x \oplus \alpha)$ equals $\beta_j \in \mathrm{F}_2^m$ while x runs through F_2^n once,
2. $\Delta_j(\alpha)$ as the auto-correlation of g_j with a shift α,
3. η_j as the sequence of g_j.

   Then $\langle \eta_j, l_i \rangle$ is the Walsh transform of g_j.

In LAT, the rows, indexed by the vectors in F_2^n, represent the coefficients of a linear boolean function, while the columns, indexed by the vectors in F_2^m, represent the coefficients for a linear combination of component functions of F. An entry in the table indicates the number of matches between input vectors for which the values of a linear function and a linear combination of component functions of F minus 2^{n-1}. In addition it can be defined as

$$LAT(i, j) = 2^{n-1} - d_H(g_j, l_i(x)), \quad l_i(x) \in A_n \tag{2.19}$$

An entry in the *j*-th column of the table equals to half of the Walsh transform of g_j. Below is the corresponding matrix for LAT table.

$$LAT = \begin{pmatrix} \langle \eta_0, l_0 \rangle / 2 & \cdots & \langle \eta_{2^m-1}, l_0 \rangle / 2 \\ \vdots & \ddots & \vdots \\ \langle \eta_0, l_{2^n-1} \rangle / 2 & \cdots & \langle \eta_{2^m-1}, l_{2^n-1} \rangle / 2 \end{pmatrix} \tag{2.20}$$

In DDT, the rows, indexed by the vectors in F_2^n, represent the changes in the inputs, while the columns, indexed by the vectors in F_2^m, represent the change in the output of the s-box. An entry in the table indexed by (α, β) indicates the number of input vectors which, when changed by α (in the sense of bit-wise XOR), result in a change in the output by β (also in the sense of bit-wise XOR).

$$DDT = \begin{pmatrix} k_0(\alpha_0) & \cdots & k_{2^m-1}(\alpha_0) \\ \vdots & \ddots & \vdots \\ k_0(\alpha_{2^n-1}) & \cdots & k_{2^m-1}(\alpha_{2^n-1}) \end{pmatrix} \tag{2.21}$$

In Auto-correlation table, the rows indexed by the vectors in F_2^n, represent the changes in the inputs, while the columns, indexed by the vectors in F_2^m, represent the coefficients for a linear combination of component functions of F. An entry in the table indexed by (α, β) indicates the auto-correlation of g_j with a shift α.

$$Auto-correlation\ table = \begin{pmatrix} \Delta_0(\alpha_0) & \cdots & \Delta_{2^m-1}(\alpha_0) \\ \vdots & \ddots & \vdots \\ \Delta_0(\alpha_{2^n-1}) & \cdots & \Delta_{2^m-1}(\alpha_{2^n-1}) \end{pmatrix} \tag{2.22}$$

In [Zhang, Zheng, Imai; 1998], a relationship between these tables are shown. Instead of *correlation immunity distribution table* used in [Zhang, Zheng, Imai; 1998] we use LAT. So, the relations are rewritten. An entry in LAT is equal to the half of the square root of the corresponding entry in correlation immunity distribution table.

- $ACT = DDT \cdot H_m$ (2.23)
- $ACT = 2^{-n} H_n \cdot LAT$ (2.24)

2.3.3 Some Classic Stream Cipher Designs

In this section mainly the LFSR based stream cipher designs are presented. There are three obvious ways to generate an alternative output. These are *Nonlinear combination generators, Nonlinear filter generators* and *Clock-controlled generators.* They are not only used lonely but also combined to get more complex and hopefully more secure stream ciphers.

In a nonlinear combination generator several linear feedback shift registers are used in parallel. The generator consists of n LFSRs to, whose outputs are combined in a Boolean function f. The principle of a combination generator is illustrated in Figure 2.4.

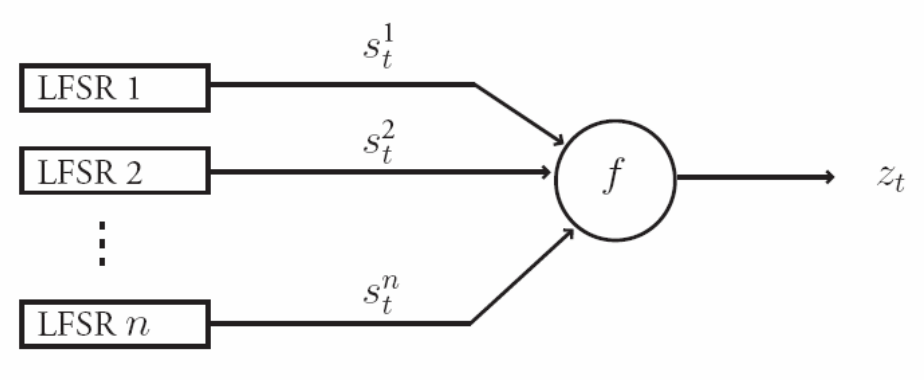


Figure 2.4 : Nonlinear combination generator

To get a secure nonlinear combination generator we need to find a function that is correlation immune and have high nonlinearity. But there is a tradeoff between these properties (Theorem 2.2). To eliminate this tradeoff, the memoryless function f can be replaced by a finite state machine with memory. And to increase both correlation immunity and nonlinearity we must employ a large number of LFSRs.

Instead of using several LFSRs one can use one single LFSR and generate the keystream as a nonlinear function f of the stages of the LFSR. Such a keystream generator called a nonlinear filter generator. The function f is then called filtering function. The principle of nonlinear filter generators is illustrated in Figure 2.5. Also for a nonlinear filter generators we can replace the memoryless

filtering function with a finite state machine. One such cipher which has recently been proposed is SNOW (Chapter 3).

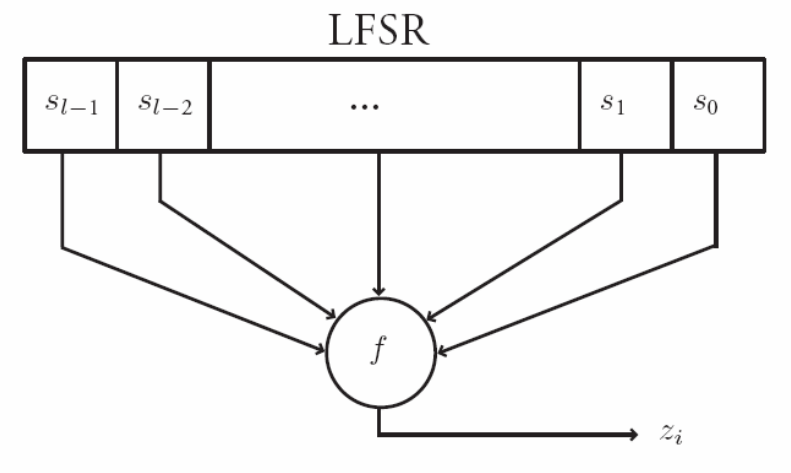


Figure 2.5 : Nonlinear filter generator

The third method used in designing stream ciphers is by clock-controlled generators. In a clock-controlled generator, the output of one or several LFSRs controls the clocking of other shift registers. Two examples of clock-controlled generators are the shrinking generator and the alternating step generator. Another clock-controlled generator that is used in practice is the cipher A5 used in GSM phones.

CHAPTER 3

DESCRIPTION OF SNOW1.0 AND SNOW2.0

SNOW is designed by Patrick Ekdahl and Thomas Johansson [Ekdahl, Johansson; 2000] and submitted to NESSIE project. It has excellent performance, several times faster than AES. After the report of distinguishing and guess-and-determine attacks on the first version, a new version of SNOW, which is called SNOW2.0 [Ekdahl, Johansson; 2002a], is proposed. The new version is said to be more secure and faster than SNOW1.0. In this chapter, first SNOW1.0 and then SNOW2.0 are described. A detailed analysis of SNOW1.0 related with the distinguishing attack is given in Section 4.2.

3.1 Description of SNOW1.0

SNOW1.0 is a keystream generator based on a LFSR defined over $\mathrm{F}_{2^{32}}$, where the nonliearity is provided by a Finite State Machine (FSM). It uses a 128-bit or 256-bit key and has an internal memory of 576 bits. The generator is shown in Figure 3.1, where we denote addition in $\mathrm{F}_{2^{32}}$ by the symbol $\oplus$, addition modulo 2^{32} by the symbol $\boxplus$, multiplication by $\alpha \in \mathrm{F}_{2^{32}}$ by $\textcircled{\alpha}$ and a cyclic shift of 7 steps to the left by $<<<$. SNOW uses an LFSR of length 16, feeding a finite state machine. The FSM consists of two 32 bit registers, called $R1$ and $R2$, as well as a some operations to calculate the output and the next value of $R1$ and $R2$.

The operation of the cipher is as follows. First, the key initialization is done. This procedure provides initial values for the LFSR as well as for the $R1, R2$ registers in the finite state machine. Next, the first 32 bits of the keystream are calculated by bitwise adding the output of the FSM and the last entry of the LFSR.

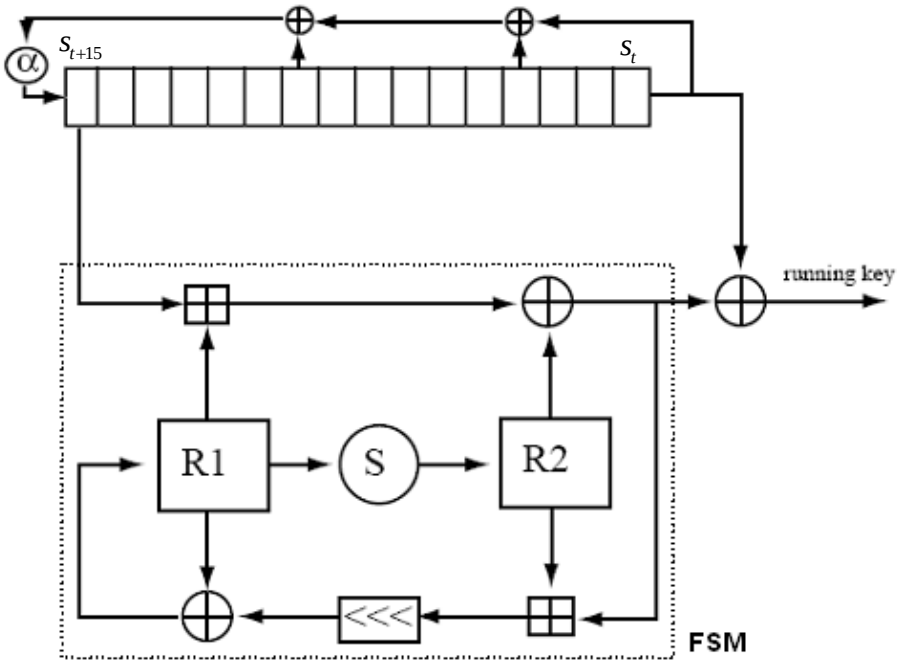


Figure 3.1 : A schematic picture of SNOW1.0

And after every clocking of the whole cipher, next 32 bits of the keystream are calculated.

The LFSR has a primitive feedback polynomial over $\mathrm{F}_{2^{32}}$ which is

$$p(x) = x^{16} \oplus x^{13} \oplus x^{7} \oplus \alpha^{-1}, \tag{3.1}$$

where $\mathrm{F}_{2^{32}}$ is generated by the irreducible polynomial

$$\pi(x) = x^{32} \oplus x^{29} \oplus x^{20} \oplus x^{15} \oplus x^{10} \oplus x \oplus 1 \tag{3.2}$$

over F_2, and $\pi(\alpha) = 0$. Furthermore let $s_{15}, s_{14}, ..., s_0 \in \mathrm{F}_{2^{32}}$ be the initial state of the LFSR, each s_{t+i} being an element of $\mathrm{F}_{2^{32}}$.

Now, let $R1_t$, $R1_{t+1}$ and $R2_t$, $R2_{t+1}$ denote the values of $R1$ and $R2$ at time $t, t+1$, respectively. The LFSR can also be defined by a linear recurrence relation

$$s_{t+16} = \alpha \cdot (s_t \oplus s_{t+3} \oplus s_{t+9}), \tag{3.3}$$

where $\cdot$ denotes multiplication in $\mathrm{F}_{2^{32}}$ and stored values $(s_{t+15}, ..., s_t)$ corresponds to the state of the LFSR. The output of the FSM at time t is computed as

$$f_t = (s_{t+15} \boxplus R1_t) \oplus R2_t. \tag{3.4}$$

The 32-bit output of the cipher at time t is computed as

$$z_t = f_t \oplus s_t . \tag{3.5}$$

The next state of the FSM is computed as

$$R1_{t+1} = R1_t \oplus ROT(f_t \boxplus R2_t, 7) \tag{3.6}$$

$$R2_{t+1} = S(R1_t), \tag{3.7}$$

where $ROT(A,B)$ denotes the cyclic rotation of A by B bits towards the most significant bit, and $S()$ is defined by four invertible 8-bit s-boxes and a bit permutation.

The s-box operation works as follows (Figure 3.2). The input x is split into 4 bytes. Each of the bytes enters a nonlinear mapping from 8 bits to 8 bits.

The nonlinear mapping is defined to be

$$r = w^7 \oplus \beta^2 \oplus \beta \oplus 1, \tag{3.8}$$

where the arithmetics are in F_{2^8}. w and r are input and output vectors which are considered as representing elements in F_{2^8} using the polynomial base $\{\beta^7,...,\beta,1\}$ generated by the irreducible polynomial $\pi(x) = x^8 \oplus x^5 \oplus x^3 \oplus x \oplus 1$ and $\pi(\beta) = 0$. In Figure 3.2, γ denotes $\beta^2 \oplus \beta \oplus 1$, which also represents the vector element (00000111) in F_{2^8}.

After the nonlinear mapping, the bits in the resulting word are permuted. The permutation is described by

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
3	10	20	24	0	14	17	29	7	13	18	25	5	12	13	27

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	8	21	26	4	9	19	31	2	11	16	28	6	15	22	30

which should be interpreted as the 31st bit position is mapped to the 3rd , the 30th bit is mapped to the 10th, etc.

There are two modes of operation specified for SNOW1.0. These are the standard mode and IV (initial vector) mode. In the standard mode, only a secret key, called k, is used to form the seed. But in IV mode the generator is initialized using

two variables, the secret key k and a known initialisation value (IV). This means that for a given secret key k, the generator produces a set of pseudo-random number sequences, one for each IV value. The details of the modes of operation and key initialisation can be found in [Ekdahl, Johansson; 2000].

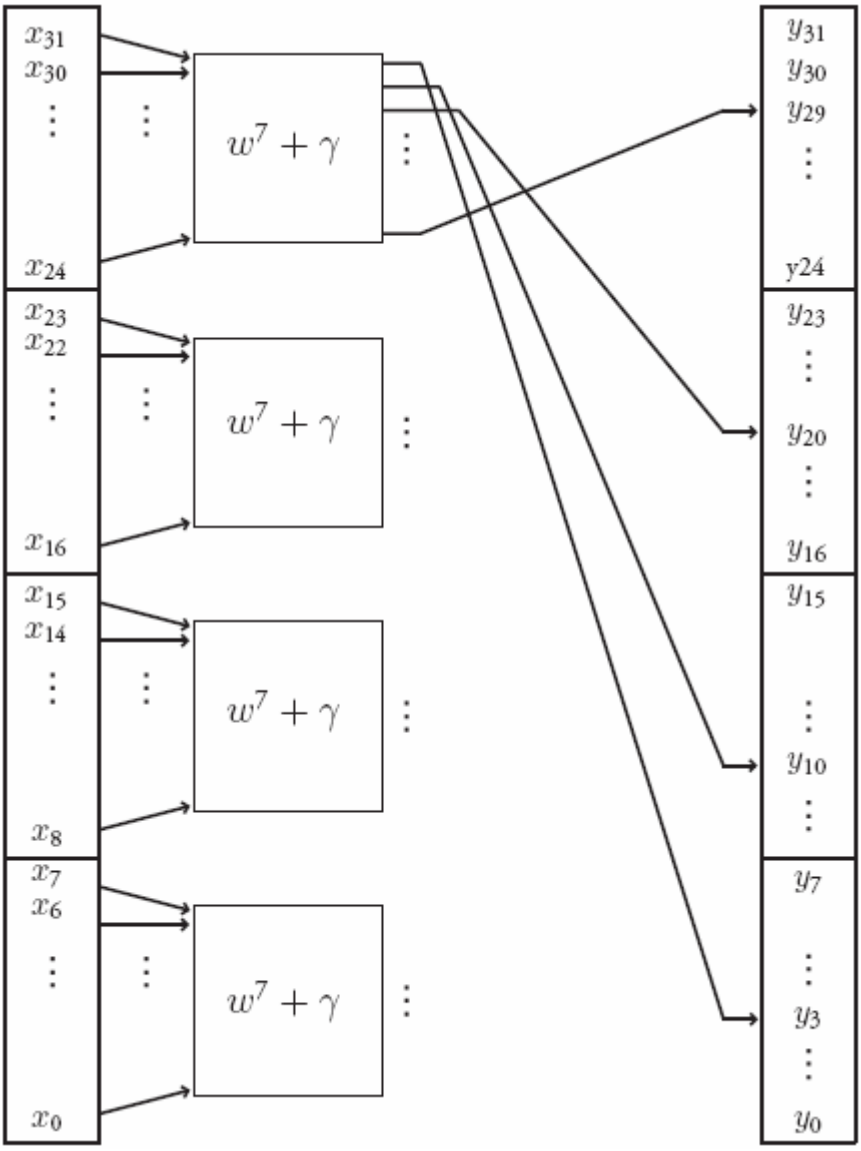


Figure 3.2 : The s-box in SNOW1.0

3.2 Description of SNOW2.0

SNOW2.0 [Ekdahl, Johansson; 2002a] is schematically a small modification of SNOW1.0, see Figure 3.3. The word size and LFSR length are the same, but the feedback polynomial is different. The Finite State Machine (FSM) has two input words, taken from the LFSR.

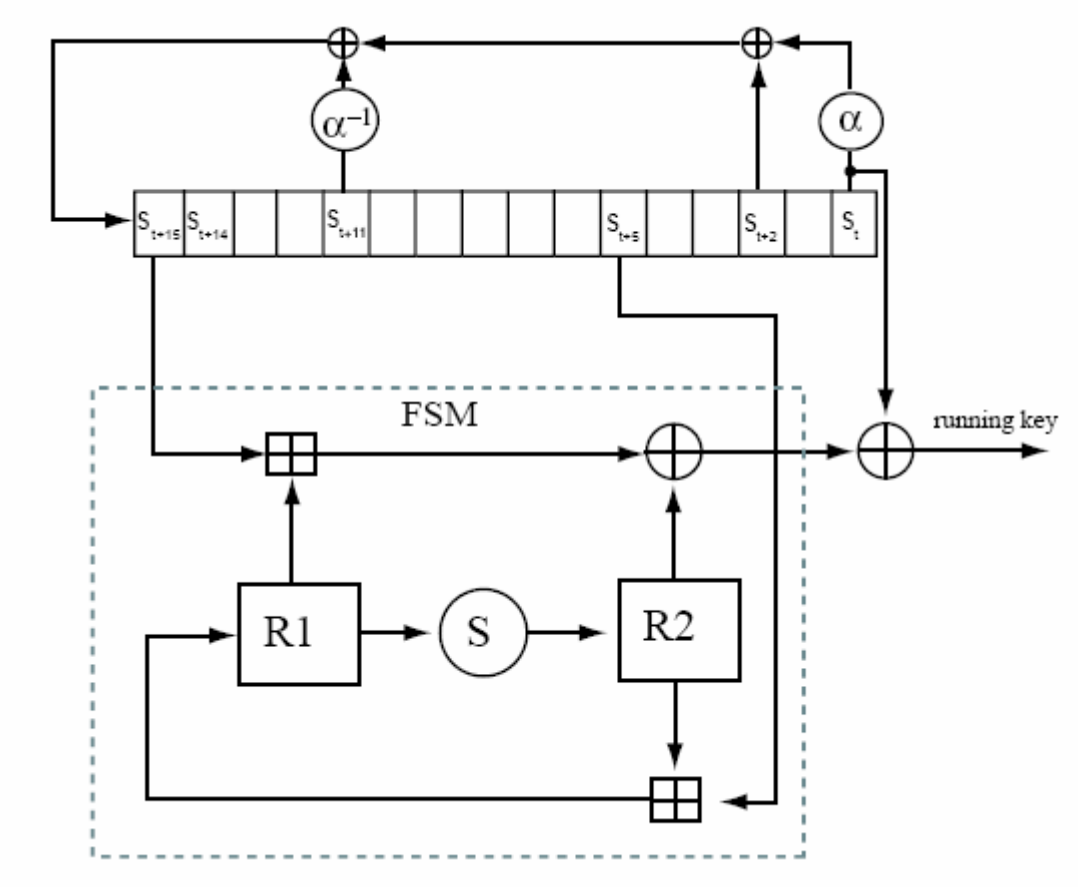


Figure 3.3 : A schematic picture of SNOW2.0

There is a small difference in the operation of the cipher. In the first version, after the key initialization, the first symbol was read out before the cipher was clocked. But in the second version it is read out after the cipher is clocked once.

The feedback polynomial of SNOW2.0 is given by

$$\pi(x) = \alpha x^{16} \oplus x^{14} \oplus \alpha^{-1} x^5 \oplus 1 \in \mathrm{F}_{2^{32}}[x], \tag{3.9}$$

where α is a root of $x^4 \oplus \beta^{23} x^3 \oplus \beta^{245} x^2 \oplus \beta^{48} x \oplus \beta^{239} \in \mathrm{F}_{2^8}[x]$,

and β is a root of $x^8 \oplus x^7 \oplus x^5 \oplus x^3 \oplus 1 \in \mathrm{F}_2[x]$.

The input to the FSM is (s_{t+15}, s_{t+5}) and the output of the FSM is calculated as

$$f_t = (s_{t+15} \boxplus R1_t) \oplus R2_t, \quad t \geq 0 \tag{3.10}$$

and the keystream is given by

$$z_t = f_t \oplus s_t, \quad t \geq 1. \tag{3.11}$$

The next state of the FSM is computed as

$$R1_{t+1} = s_{t+5} \boxplus R2_t \quad \text{and} \tag{3.12}$$

$$R2_{t+1} = S(R1_t) \quad t \geq 0. \tag{3.13}$$

The s-box used in SNOW1.0 was changed with the one of Rijndael [Daemen, Rijmen; 2002] in SNOW2.0. K. Nyberg, in [Nyberg; 93], inspired J. Daemen and V. Rijmen to use a mapping $\mathrm{x} \to \mathrm{x}^{-1}$ in the design of Rijndael. Nyberg presented the impressive properties of this mapping. Actually, the high nonlinearity property, which we also examine in Section 4.1, was first proven in the work of Carlitz and Uchiyama [Carlitz, Uchiyama; 1957].

It is a permutation on $\mathbf{Z}_{2^{32}}$. Let $w = (w_3, w_2, w_1, w_0)$ be the input to the s-box, where $w_i, i = 0...3$ is the four bytes of w. Assume w_3 to be the most significant byte. Let

$$w = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{pmatrix} \tag{3.14}$$

be a vector representation of the input to the s-box. First the Rijndael s-box, denoted S_R, is applied to each byte.

$$\begin{pmatrix} S_R[w_0] \\ S_R[w_1] \\ S_R[w_2] \\ S_R[w_3] \end{pmatrix} \tag{3.15}$$

Then the MixColumn Transformation of Rijndael's round function is applied which can be computed as a matrix multiplication,

$$\begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{pmatrix} = \begin{pmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x \end{pmatrix} \begin{pmatrix} S_R[w_0] \\ S_R[w_1] \\ S_R[w_2] \\ S_R[w_3] \end{pmatrix} \tag{3.16}$$

where (r_3, r_2, r_1, r_0) are the output bytes from the s-box. These bytes are concatenated to form the word output from the s-box $r = S(w)$.

SNOW2.0 has only IV mode of operation and length of the IV is 128 bit. The details of the key initialization can be found in [Ekdahl, Johansson; 2002a].

3.3 Implementation Performances of SNOW

In addition to the improvements made in the cryptograhical sense, the implementation performance of SNOW1.0 was also improved in SNOW2.0. The implementation performances, that are tabulated in Table 3.1, are taken from [Ekdahl, Johansson; 2002a]. It is told that the tests were run on a PC with Intel 4 processor running at 1.8GHz, 512 Mb of memory.

Table 3.1 : Number of cycles needed for key setup and keystream generation on a Pentium 4 @1.8GHz

	SNOW1.0	**SNOW2.0**
Key sutup	925	937
Keystream generation	47	38

CHAPTER 4

EXAMINATION OF SNOW1.0 AND SNOW2.0

The NESSIE evaluation process consists of two phases. The first version of SNOW (SNOW1.0) has passed the first phase of the NESSIE evaluation, but could not pass the second phase due to two attacks reported (Table 4.1). One is a guess-and-determine attack [Hawkes, Rose; 2002] and the other is a distinguishing attack [Coppersmith, Halevi, Jutla; 2002]. Actually there are two methods reported for the guess-and-determine attack and they have different process and data complexities. In section 4.2.1, we give a review of the distinguishing attack on SNOW1.0 by using the description [Ekdahl, Johansson; 2002b] of the attack on SOBER-t32 [Hawkes, Rose; 2000]. Since the structure of SOBER-t32 is similar to SNOW1.0, i.e., both of the ciphers consist of an LFSR and a nonlinear function, the same approach can be made to SNOW1.0 as well. However, since the methods of calculation for the complexities of the attack are different, the values found with the method used in [Ekdahl, Johansson; 2002b] are higher than the ones in [Coppersmith, Halevi, Jutla; 2002].

Guess-and-determine attacks exploit the relationships between internal values (such as the recurrence relationship in a shift register), and the relationship used to construct the key-stream values from the internal values. It is a kind of key recovery attack. In this attack some internal values are guessed and then the relationships are used to determine other internal values. The cipher is said to be "*broken*" when a complete internal state has been determined from the guessed values. When this attack is applied to SNOW1.0, it is aided by the unfortunate choice of inputs to the recurrence relation, which is

Table 4.1 : Complexities of attacks on SNOW1.0 and SNOW2.0

	SNOW1.0		SNOW2.0	
	Data Complexity	***Process Complexity***	***Data Complexity***	***Process Complexity***
Guess-and-Determine Attack (Method 1)	2^{64}	2^{256}	-	-
Guess-and-Determine Attack (Method 2)	2^{95}	2^{224}	-	-
Distinguishing Attack	2^{95}	2^{100}	2^{225}	2^{225}

$$s_{t+16} = \alpha(s_t \oplus s_{t+3} \oplus s_{t+9}). \tag{4.1}$$

There is a distance of 3 words between s_t and s_{t+3}, and a distance of $6 = 2 \times 3$ words between s_{t+3} and s_{t+9}. Thus, by squaring (4.1)

$$s_{t+32} = \alpha^2(s_t \oplus s_{t+6} \oplus s_{t+18}) \tag{4.2}$$

we see that $(s_{t+i} \oplus s_{t+i+6})$ can be considered as a single input to either equation. The attacker does not need to determine both s_{t+i} and s_{t+i+6} explicitly, but only the exor sum will be enough to use in (4.1) and (4.2). If the linear recurrence did not have this property, then it is likely that fewer state words could be derived from the guessed words, and the attacker would be unable to derive a full state from the guessed words. To overcome this weakness, the recurrence relation is changed in SNOW2.0.

Distinguishing attacks against stream ciphers are basically established on the way of considering ciphers in two parts, linear and nonlinear. Linear approximation of the nonlinear part is found and then combined with the linear part. In SNOW1.0, obviously the linear part is the LFSR and the nonlinear part is the FSM. In [Coppersmith, Halevi, Jutla; 2002], it is shown that large correlations found in the FSM by the help of the linear approximation can be turned into a distinguishing attack.

In SNOW1.0, the linearity caused by the linear feedback shift register is destroyed using a filtering generator with memory, i.e., a Finite State Machine (FSM). In today's cryptography, both in block ciphers and stream ciphers,

nonlinearity is generally achieved by substitution boxes (s-box). Due to the weaknesses found in s-boxes, linear and differential cryptanalysis can be done. Although these cryptanalysis techniques are mostly applied to block ciphers, it can also be applied to a word-oriented stream cipher.

In this chapter, first the s-boxes of SNOW1.0 and SNOW2.0 are examined. A review of the distinguishing attack is given using the approach in [Ekdahl, Johansson;2002b] made for SOBER-t32 and the effect of the s-boxes to the distinguishing attack and how the correlations are affected by the operations in the finite state machine are investigated. Randomness tests are performed using NIST Statistical Test Suite and the results of these tests are presented.

4.1 Examination of S-Boxes

The s-box of SNOW1.0, which was mentioned in Section 3.1, has the nonlinear mapping

$$r = w^7 \oplus \beta^2 \oplus \beta \oplus 1, \tag{4.3}$$

where the arithmetics are in F_{2^8}. $w = (w_7, w_6, ..., w_0)$ and $r = (r_7, r_6, ..., r_0)$ are input and output vectors which are considered as representing elements in F_{2^8} using the polynomial base $\{\beta^7, ..., \beta, 1\}$ generated by the irreducible polynomial $\pi(x) = x^8 \oplus x^5 \oplus x^3 \oplus x \oplus 1$ and $\pi(\beta) = 0$. Equation (4.3) can be rewritten as

$$r = w^7 \oplus c_1, \tag{4.4}$$

where c_1 denotes the coostant vector element in F_{2^8}, which is equal to (00000111).

In SNOW2.0, the s-box of Rijndael [Daemen, Rijmen; 2002] is used. It is implemented by two transformations:

- First taking the multiplicative inverse in $GF(2^8)$. In binary representation '00000000' is mapped onto itself.
- Then applying an affine transformation over $\mathbf{Z}_2$ defined by:

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

where $x = (x_7, x_6, ..., x_0)$ is the multiplicative inverse of the byte at the input of the s-box.

The non-linear mapping of the s-box of Rijndael can be considered in the form of Equation (4.4) as

$$r = aw^{-1} \oplus c_2, \tag{4.5}$$

where a and c_2 denotes constant elements in F_{2^8}. Let $\alpha \in \mathrm{F}_{2^8}$, then $\alpha^{2^8-1} = 1$. So, α^{-1} is equal to α^{2^8-2}, which is also equal to α^{254}. Hence, equation (4.5) becomes

$$r = aw^{254} \oplus c_2. \tag{4.6}$$

4.1.1 Linear Approximation Table

The examination of the s-boxes starts with the formation of the Linear Approximation Tables. In this section, we present the distribution of LAT values that we calculate for the s-boxes of SNOW1.0 and SNOW2.0 (See Appendix A).

A LAT table is a good indicator that shows the susceptibility of a cipher to linear cryptanalysis. Nonlinearity measurement is used to evaluate this susceptibility. Nonlinearity for an $n \times m$ s-box is defined as

$$N_S = \min_g \{N_g \mid g = \bigoplus_{j=1}^{m} c_j f_j,\ c_j \in GF(2), g \neq 0\} \tag{4.7}$$

where f_i, $1 \le i \le m$, of $S = (f_1, f_2, ..., f_m)$ is a Boolean mapping $\mathrm{F}_2^n \to \mathrm{F}_2$. This can be expressed for an $n \times n$ s-box in terms of maximum magnitude LAT element as

$$N_S = 2^{n-1} - \max_{\mathbf{w},\mathbf{c} \in \mathrm{F}_2^n} \left| LAT(\mathbf{w}, \mathbf{c}) \right| \tag{4.8}$$

where $\mathbf{w}$ and $\mathbf{c}$ denote the rows and columns of LAT.

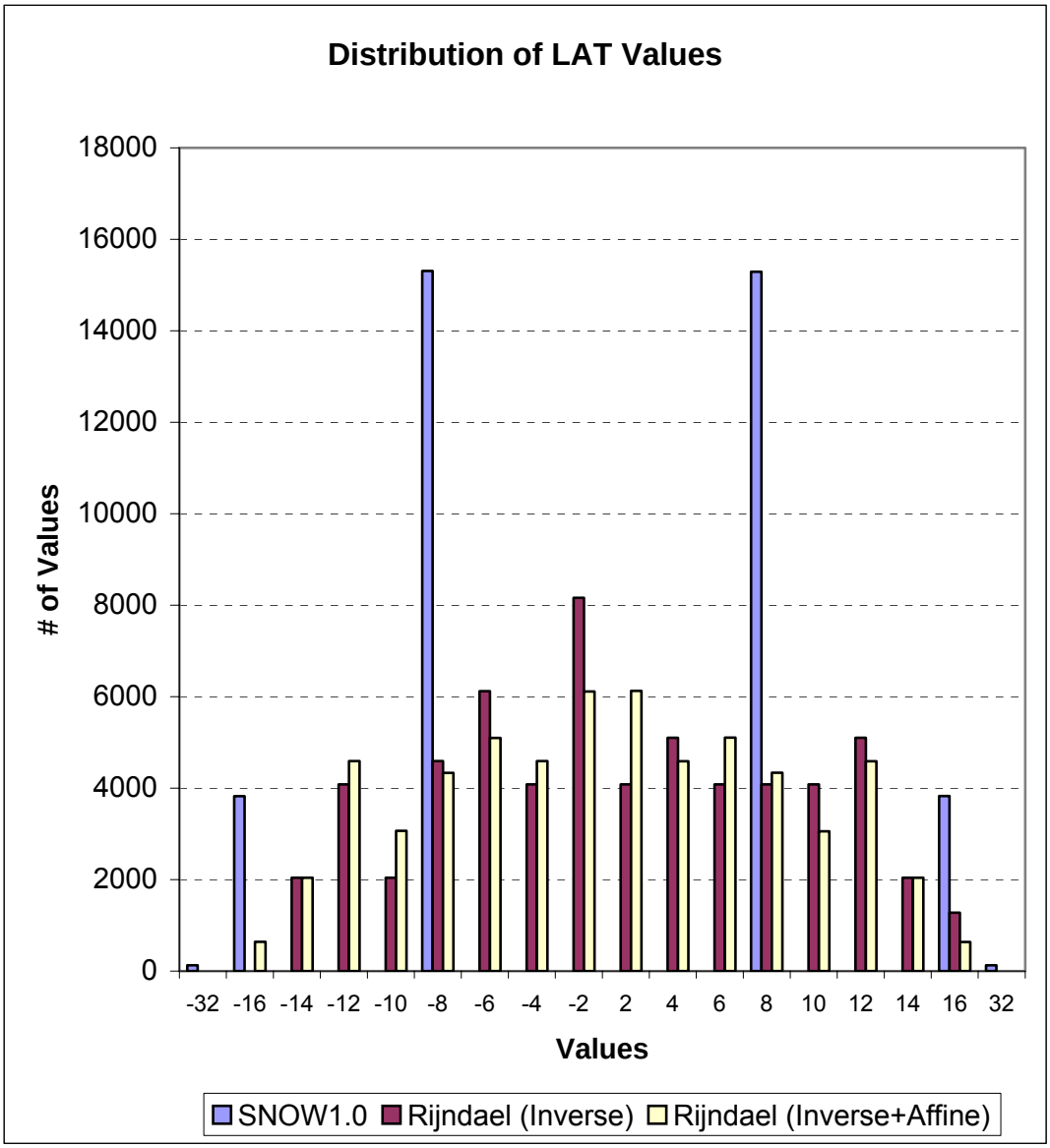


Figure 4.1 : Distribution of LAT values for SNOW1.0 and Rijndael

Table 4.2 : Details for Figure 4.1

	-32	-16	-14	-12	-10	-8	-6	-4	-2	2	4	6	8	10	12	14	16	32
SNOW1.0	128	3824				15308							15292				3826	127
Rijndael (Inverse)		0	2040	4080	2040	4590	6120	4080	8160	4080	5100	4080	4080	4080	5100	2040	1275	
Rijndael (Inverse+Affine)		640	2040	4592	3064	4334	5096	4592	6112	6128	4588	5104	4336	3056	4588	2040	635	

The elements of LAT can be expressed as

$$LAT(\mathbf{w},\mathbf{c}) = 2^{n-1} - d(\mathbf{c} * F(\mathbf{x}), \mathbf{w} * \mathbf{x}) \ . \quad (4.9)$$

In Figure 4.1, the distribution of LAT values for SNOW1.0 and Rijndael are shown together. The details of this figure are tabulated in Table 4.2.

Examining Figure 4.1, the maximum magnitude in the LAT elements is 32. This leads to a nonlinearity of 96=128-32. This nonlinearity measurement can also be treated as the number of inputs for which the equality of a linear boolean function and a component function (of F) holds. That is, for 255 (127+128) equation pairs, 96 plaintexts satisfies the equality. Then the probability that these equations hold can be calculated as $96/256 = 0.375$ (bias from $1/2$ is 0.125).

For Rijndael two diagrams are tabulated, one for multiplicative inverse function, the other for multiplicatice inverse function and affine transformation. The aim was to see that there is no effect of affine transformation on the nonlinearity. The reason for affine transformation seems to be the elimination of the “zero to zero” map in multiplicative inverse function.

The maximum magnitude LAT element for Rijndael is 16. So, the nonlinearity is 112=128-116 and there are 1275 equation pairs for which the probability that those equations hold is $112/256 = 0.4375$ (bias from $1/2$ is 0.0625).

Whenever these calculated probabilities are not close to $1/2$, it means that the examined s-box is susceptible to linear cryptanalysis. Thus, regarding the above calculated probabilities for SNOW1.0 and Rijndael, one can say that s-box of SNOW1.0 is more susceptible to linear cryptanalysis.

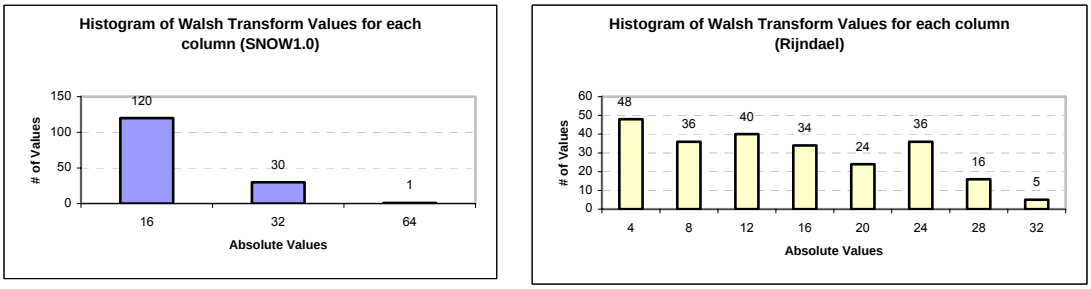
Walsh transform values can be used to evaluate the strength of an s-box. Every column of LAT is half of the Walsh transform values of a linear combination function $g(x)$ given by

$$g(x) = \bigoplus_{j=1}^{8} c_j f_j(x), \ c_j \in GF(2). \quad (4.10)$$

Hence, in order for a Boolean function to be highly nonlinear, the absolute value of its Walsh transform should not take large values. The s-boxes of both SNOW1.0 and Rijndael consist of 8 Boolean functions and all Boolean functions perform well for each of the ciphers. All Boolean functions have the same histogram of Walsh transform values. Figure 4.2 (a) and (b) show the histograms of Walsh transform values for all linear combinations of 8 Boolean functions for the s-boxes of

SNOW1.0 and Rijndael, respectively. Notice that compared to SNOW1.0, Rijndael has a more uniform distribution of Walsh transform values.

The Walsh transforms of one of the Boolean functions of SNOW1.0 and Rijndael are given in Figure 4.3 and 4.4, respectively.



a) SNOW1.0 **b)** Rijndael

Figure 4.2 : Histogram of Walsh Transform values for each combination of component functions of F for **a)** SNOW1.0 **b)** Rijndael

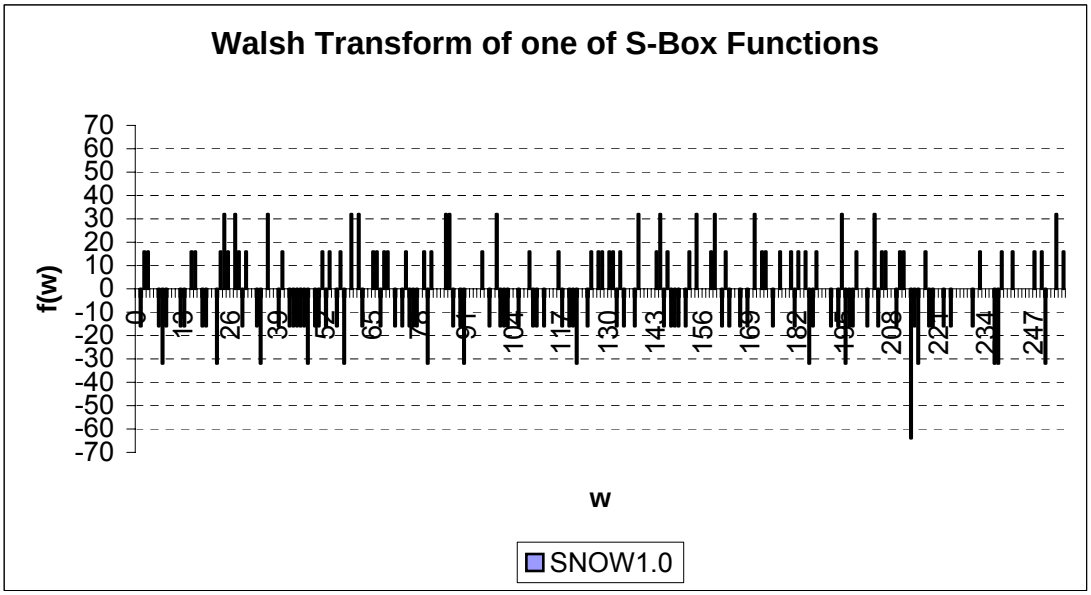


Figure 4.3 : Walsh Transform of one of s-box functions of SNOW1.0

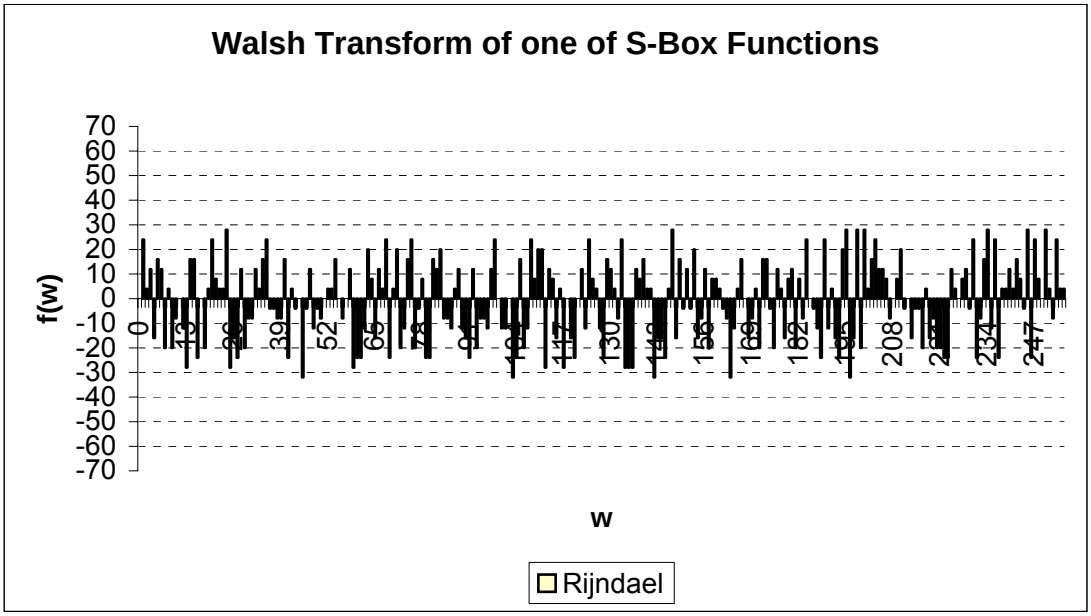


Figure 4.4 : Walsh Transform of one of s-box functions of Rijndael

4.1.2 Difference Distribution and Auto-correlation Tables

The Difference Distribution Table is a matrix of size 256 x 256, whose entries are calculated by Equation 2.21. The maximum entry in DDT determines the security level against differential cryptanalysis.

The s-boxes of SNOW1.0 and Rijndael are 6 and 4-differentially uniform, respectively. So, the s-box of SNOW1.0 is more susceptible to differential cryptanalysis. And as we see in Figure 4.6, DDT values of both ciphers are uniformly distributed.

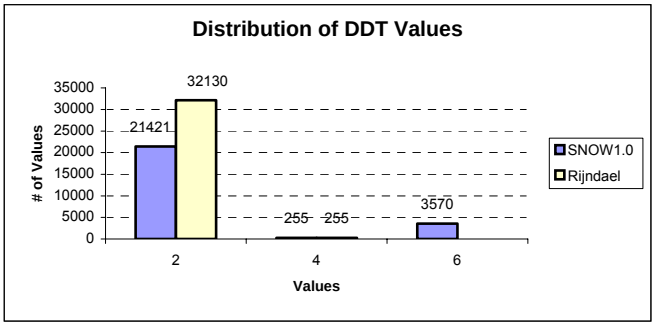
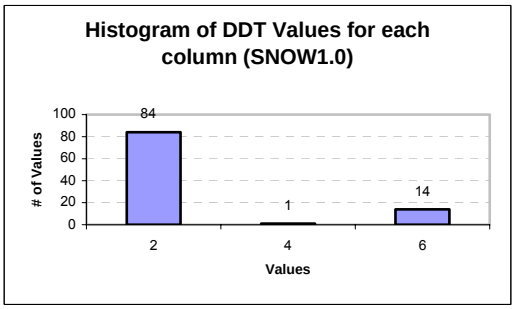
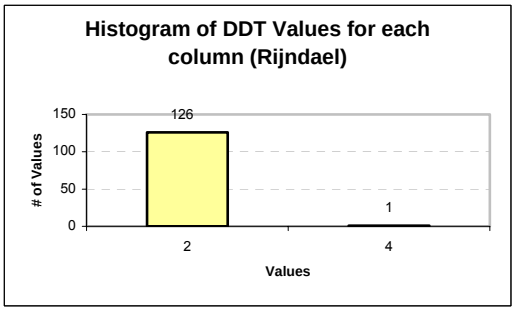


Figure 4.5 : Distribution of DDT values





a) SNOW1.0 **b)** Rijndael

Figure 4.6 : Histogram of DDT values for each column

We now tabulate the auto-correlation values for linear combination functions $g(x)$ given in Equation (4.10). The entries of the tables are calculated by Equation 2.22.

As we see in Figure 4.7, there are a lot of entries at higher values in the auto-correlation table of SNOW1.0. But for Rijndael, the entries are distributed uniformly at lower values.

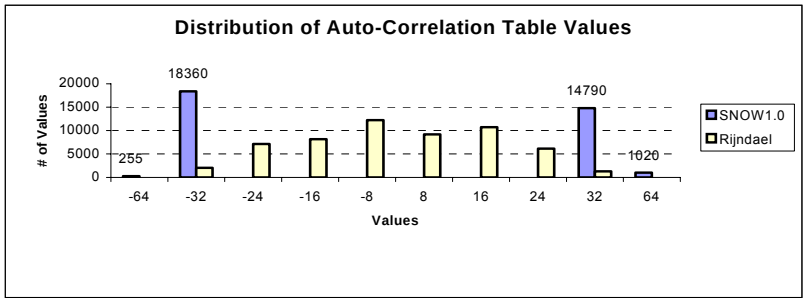
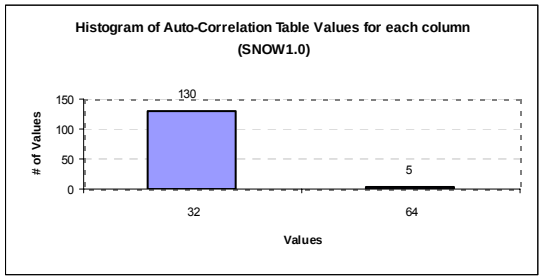
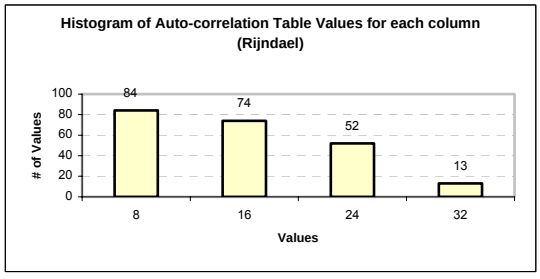


Figure 4.7 : Distribution of auto-correlation table values





a) SNOW1.0 **b)** Rijndael

Figure 4.8 : Histogram of auto-correlation table values for each column

4.2 Analysis of the Finite State Machine

One of the attacks against SNOW1.0 was a distinguishing attack by Coppersmith *et al* [Coppersmith, Halevi, Jutla; 2002]. The basic idea of this attack, developed earlier by Golic [Golic; 1996], is to distinguish the outputs of a keystream generator from a truly random bit sequence, by applying the technique of linear cryptanalysis on block ciphers. Quite interestingly, in the same year the designers of SNOW also applied a distinguishing attack to SOBER-t16 and t-32 [Ekdahl, Johansson; 2002b].

In a distinguishing attack, the internal state of the target keystream generator is divided into the linear part and nonlinear part, and the best approximation of the non-linear part is found. The attack is based on combining the linear approximation of the nonlinear part with the linear recurrence, defined through the feedback polynomial.

In SNOW1.0, the linear recurrence relation of LFSR in Figure 4.9 is given by

$$s_{t+16} = \alpha \cdot (s_t \oplus s_{t+3} \oplus s_{t+9}) \tag{4.11}$$

and the corresponding characteristic polynomial is

$$p(x) = x^{16} \oplus x^{13} \oplus x^7 \oplus \alpha^{-1}. \tag{4.12}$$

To be able to apply a distinguishing attack against SNOW1.0, we need a polynomial with 0-1 coefficients, i.e., we have to eliminate the element $\alpha \in \mathrm{F}_{2^{32}}$. Repeated squaring of this polynomial will still yield a valid linear recurrence equation for the considered linear recurrence of SNOW1.0. The exponentiation with 2^{32} gives

$$p(x)^{2^{32}} = x^{16\times 2^{32}} \oplus x^{13\times 2^{32}} \oplus x^{7\times 2^{32}} \oplus \alpha^{-1\times 2^{32}}. \tag{4.13}$$

Since $\alpha \in \mathrm{F}_{2^{32}}$, $\alpha^{-1\times 2^{32}} = \alpha^{-1}$ and addition of (4.11) and (4.12) gives

$$p(x)^{2^{32}} \oplus p(x) = x^{16x2^{32}} \oplus x^{13x2^{32}} \oplus x^{7x2^{32}} \oplus x^{16} \oplus x^{13} \oplus x^7. \tag{4.14}$$

Since the coefficient α, which does not belong to F_2, is eliminated, one has a linear relation with 6 terms that holds for each single bit position. Dividing (4.14) by x^7, the resulting linear recurrence equation is given by

$$s_{t+16\times 2^{32}-7} \oplus s_{t+13\times 2^{32}-7} \oplus s_{t+7\times 2^{32}-7} \oplus s_{t+9} \oplus s_{t+6} \oplus s_t = 0. \tag{4.15}$$

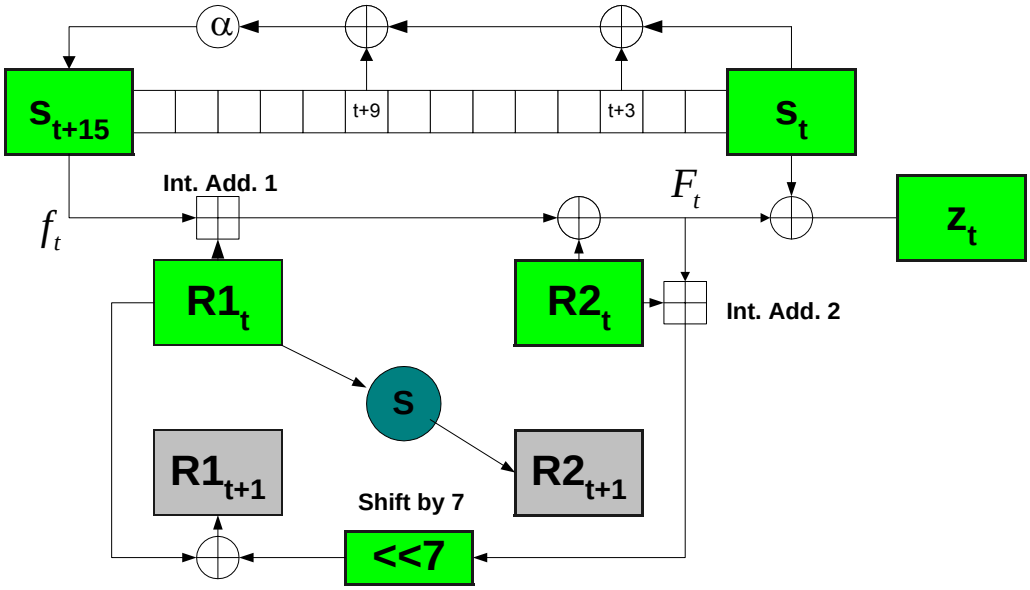


Figure 4.9 : Schematic view of SNOW1.0

For the nonlinear part, the linear approximation of FSM, found by [Coppersmith, Halevi, Jutla; 2002], is

$$(f_t)_{15} \oplus (f_t)_{16} \oplus (f_{t+1})_{22} \oplus (f_{t+1})_{23} = (F_t)_{15} \oplus (F_{t+1})_{23} \tag{4.16}$$

where f_t is the input to FSM and F_t is the output to FSM in Figure 4.9. The bias of the approximation is found as at least $2^{-9.3}$ [Coppersmith, Halevi, Jutla; 2002]. We will try to explain how (4.16) is obtained in Section 4.2.2; after giving a review of the distinguishing attack [Coppersmith, Halevi, Jutla; 2002] by using the approach in [Ekdahl, Johansson; 2002b], made for a distinguishing attack on SOBER-t32.

4.2.1 Review of the Distinguishing Attack on SNOW1.0

To understand the core idea of the attack, let's write the output of the FSM as

$$F_t = \Omega_t \oplus w_t \tag{4.17}$$

where Ω_t is for the linear operations of the words from LFSR and w_t is for the noise (nonlinearity) introduced by the integer addition and substitution box.

Now, given the FSM output, $F_0, F_1, ..., F_{N-1}$, of length N, we can use the linear recurrence relation (4.15) to calculate

$$F_{t+16\times 2^{32}-7} \oplus F_{t+13\times 2^{32}-7} \oplus F_{t+7\times 2^{32}-7} \oplus F_{t+9} \oplus F_{t+6} \oplus F_t =$$

$$\Omega_{t+16\times 2^{32}-7} \oplus w_{t+16\times 2^{32}-7} \oplus \Omega_{t+13\times 2^{32}-7} \oplus w_{t+13\times 2^{32}-7} \oplus \Omega_{t+7\times 2^{32}-7} \oplus w_{t+7\times 2^{32}-7} \oplus$$

$$\Omega_{t+9} \oplus w_{t+9} \oplus \Omega_{t+6} \oplus w_{t+6} \oplus \Omega_t \oplus w_t \qquad (4.18)$$

Since all the Ω_j terms only depend on the words from the LFSR, they will be equal to zero as a result of (4.15). Then, we have

$$F_{t+16\times 2^{32}-7} \oplus F_{t+13\times 2^{32}-7} \oplus F_{t+7\times 2^{32}-7} \oplus F_{t+9} \oplus F_{t+6} \oplus F_t =$$

$$w_{t+16\times 2^{32}-7} \oplus w_{t+13\times 2^{32}-7} \oplus w_{t+7\times 2^{32}-7} \oplus w_{t+9} \oplus w_{t+6} \oplus w_t . \qquad (4.19)$$

Using Matsui's *Piling-Up Lemma*, which was introduced in [Matsui; 1993], we can calculate the probability that the noise variables sum to zero.

Piling-Up Lemma [Matsui; 1993]:

For n independent, random binary variables $X_1, X_2, ..., X_n$,

$$P(X_1 \oplus X_2 \oplus ... \oplus X_n = 0) = 1/2 + 2^{n-1}\prod_{i=1}^{n} \varepsilon_i \qquad (4.20)$$

where ε_i represents the probability that X_i is equal to zero.

Denote the left hand side of (4.19) by F_t^* and the bias of (4.16) by ε_{23}. Then, using (4.20), we have

$$P_\varepsilon((F_t^*)_{23} = 0) = \frac{1}{2} + 2^5 \varepsilon_{23}^6 , \qquad (4.21)$$

where 2^5 accounts for all possible cases for an exor operation with 6 terms to be equal to zero. Then, the total bias of the cipher is calculated as $2^5 \cdot 2^{-9.3\times 6} = 2^{-50.8}$.

But how many outputs do we need in order to distinguish this biased output of FSM from a truly random source? The probability of error while distinguishing the outputs of FSM is

$$P_e = 2^{-nC(P_\varepsilon, P_u)} \quad \text{[Cover, Thomas; } \textit{Elements of Inf. Theory.} \text{ 1991]} \qquad (4.22)$$

where is n the number of outputs, $P_\varepsilon = \frac{1}{2} + \varepsilon$, $P_u = \frac{1}{2}$ (uniform distribution) and $C(P_\varepsilon, P_u)$ is the *Chernoff Information*

$$C(P_\varepsilon, P_u) = -\min_{0\le\lambda\le 1} \log_2 \left(\sum_{x\in N} (P_\varepsilon(x))^\lambda (P_u(x))^{1-\lambda} \right) \qquad (4.23)$$

The exact value of λ is difficult to obtain. So, to get an upper bound in (4.22), the summation in (4.23) should be minimized. To achieve this we can take λ as 0.5. After making calculations the approximate value for $C(P_\varepsilon, P_u)$ is found to be ε^2. Using $P_e = 0.5$ we derive the number of outputs needed to distinguish the output of SNOW1.0 from a truly random bit sequence as $2^{101.6} (= 1/\varepsilon^2)$ and the process complexity is same.

4.2.2 Approximating the FSM

In order to verify the approximation of the FSM and to construct a test mechanism we have clearly rederived the approximation.

There are two non-linear operations in the FSM: one is the substitution box and the other is the addition modulo 2^{32}. To simplify the calculations, the substitution box is ignored and the addition modulo 2^{32} is approximated with the xor operation (and one extra bit is also taken into account to approximate the carry bit). After making these changes, Figure 4.10 is achieved for SNOW1.0.

The input of FSM is related to its output with the equation

$$(f_{t+1})_x + (R1_{t+1})_x \oplus (R2_{t+1})_x = (F_{t+1})_x \qquad (4.24)$$

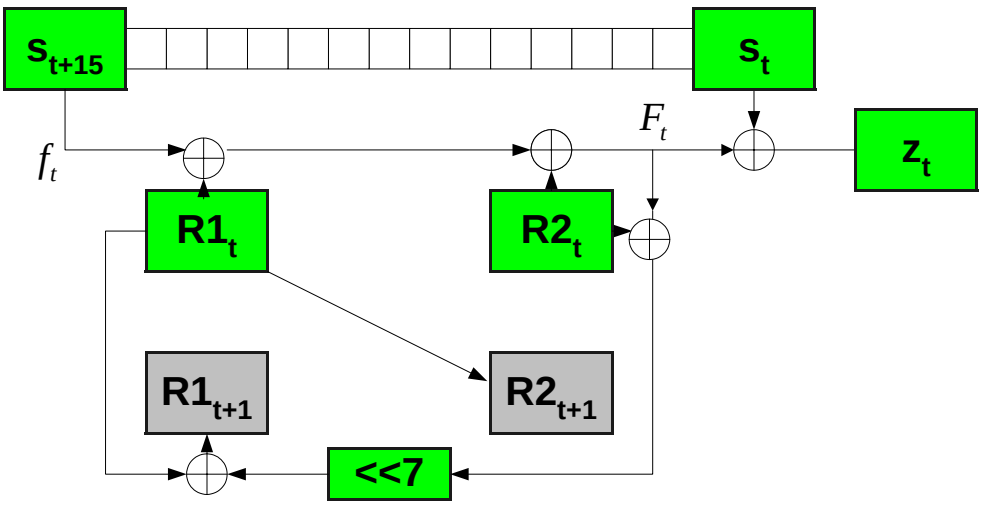


Figure 4.10 : Schematic view of SNOW1.0 with the approximation of the FSM

The substitutions, (4.25), (4.26) and (4.27), are inserted in (4.24) for two rounds calculations.

$$R2_{t+1} = S(R1_t) \;\Rightarrow\; R2_{t+1} \approx R1_t \tag{4.25}$$

$$(f_{t+1})_x + (R1_{t+1})_x = (f_{t+1})_x \oplus (R1_{t+1})_x \oplus (f_{t+1})_{x-1} \tag{4.26}$$

$$(R1_{t+1})_x = (R1_t)_x \oplus (R2_t)_{x-7} + (F_t)_{x-7} \tag{4.27}$$

Then, we have the equation

$$(f_{t+1})_x \oplus (R1_t)_x \oplus (R2_t)_{x-7} + (F_t)_{x-7} \oplus (f_{t+1})_{x-1} \oplus (R1_t)_x = (F_{t+1})_x . \tag{4.28}$$

By using (4.24) we can achieve (4.29).

$$(f_t)_{x-7} + (R1_t)_{x-7} \oplus (R2_t)_{x-7} = (F_t)_{x-7} \tag{4.29}$$

Substituting (4.29) instead of $(F_t)_{x-7}$ in (4.28) and approximating the integer addition, we relate the input of FSM to its output for two rounds.

$$(f_t)_x \oplus (f_t)_{x+1} \oplus (f_{t+1})_{x+7} \oplus (f_{t+1})_{x+8} = (F_t)_x \oplus (F_{t+1})_{x+8} \tag{4.30}$$

where $0 \le x \le 23$.

The correlations are searched for all possible x values. The cipher algorithm is run 500.000.000 times in order to find the bias of the equation. The biases, that are found, are listed in Table 4.3.a. The biggest correlation is achieved when x is equal to 15 and the value of the bias is approximately $2^{-9.34}$. This is almost the same as the one that Coppersmith has found.

4.2.3 Changing the S-Box

In part 4.1, the s-boxes of the two versions of SNOW were examined. There, we have concluded that the s-box of SNOW1.0 is weaker in all aspects that we have considered. In order to see the effect of s-boxes to the correlations between the input and output of FSM, the s-box of SNOW1.0 is changed with that of Rijndael. The biases that are found, are listed in Table 4.3.b . Examining the table, the largest correlation value is observed as $2^{-14.3}$ when x is equal to 12. Then, the total bias of the cipher is calculated as $2^5 \cdot 2^{-14.23\times 6} = 2^{-80.38}$ and the number of outputs needed to perform a distinguishing attack is $2^{160.8}$, which is slower than the exhaustive search for the 128-bit key, but not for the 256-bit key.

Only the change of the substitution box makes a great deal of improvement in the strength of the cipher against the distinguishing attack of Coppersmith *et al*. [Coppersmith, Halevi, Jutla; 2002]. We can conclude that one of the reasons for the large correlations is the disadvantages of the first s-box with respect to the second one.

Table 4.3 : Experimentally found correlation values in the FSM

x	Bias
0	0.000019
1	0.000000
2	0.000005
3	-0.000005
4	0.000010
5	-0.000035
6	0.000008
7	0.000004
8	-0.000001
9	0.000020
10	0.000032
11	-0.000028
12	0.000027
13	-0.000019
14	0.000007
15	**0.001545**
16	0.000041
17	0.000015
18	-0.000045
19	0.000001
20	0.000014
21	0.000011
22	0.000006
23	0.000045

$= 2^{-9.34}$

a) SNOW1.0

x	Bias
0	0.000023
1	0.000011
2	0.000009
3	0.000008
4	-0.000013
5	0.000004
6	0.000014
7	0.000044
8	0.000004
9	-0.000020
10	0.000016
11	0.000008
12	**0.000052**
13	0.000036
14	0.000008
15	-0.000013
16	-0.000015
17	-0.000007
18	-0.000027
19	-0.000008
20	-0.000016
21	0.000008
22	0.000008
23	0.000036

$= 2^{-14.23}$

b) The s-box of SNOW1.0 is changed with that of Rijndael.

4.2.4 Changing the "Integer Additions" with "Additions in $F_{2^{32}}$"

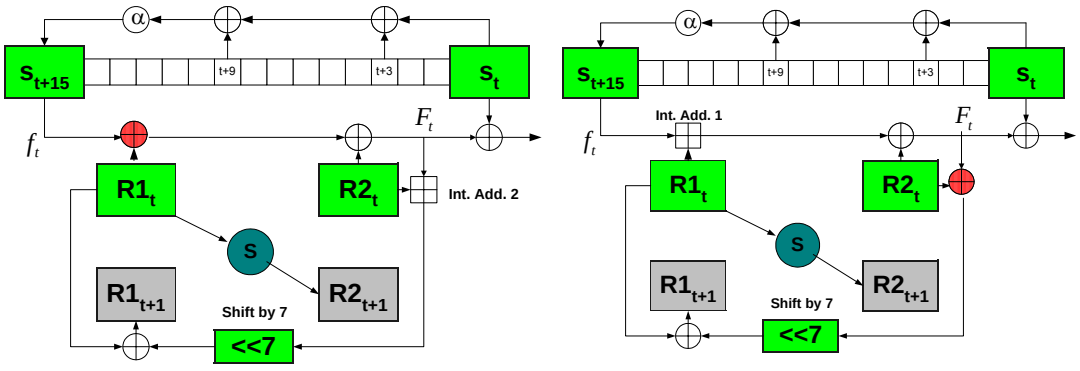
To discover the effects of two integer additions (Figure 4.9) on the correlations in the FSM, we change them with "additions in $F_{2^{32}}$" one by one and search for the correlations. The modified verisons SNOW1.0 are shown in Figure 4.11 (a) and (b).

To obtain the approximations of the FSM, again we make the calculations in Section 4.2.2 by taking the modifications into account. Thus, we obtain the equations (4.31) and (4.32) for Figures 4.11 (a) and (b), respectively.

$$(f_t)_x \oplus (f_t)_{x+1} \oplus (f_{t+1})_{x+8} = (F_t)_x \oplus (F_{t+1})_{x+8} \tag{4.31}$$

$$(f_t)_x \oplus (f_t)_{x+1} \oplus (f_{t+1})_{x+7} \oplus (f_{t+1})_{x+8} = (F_{t+1})_{x+8} \tag{4.32}$$

The corresponding results for the correlation searches are tabulated in Table 4.4 (a) and (b). The results show that changing *Int. Add. 1* makes impressively much contribution to the strength of the cipher. This contribution is almost the same as what we have achieved when we change the s-box of SNOW1.0 with that of Rijndael. Though, it is quite likely that there is an interaction between the s-box and *Int. Add. 1* in the original version of SNOW1.0. Whereas changing the *Int. Add. 2* slightly decreases the strength of the cipher.



a) *Int. Add. 1* is replaced by addition in $F_{2^{32}}$.

b) *Int. Add. 2* is replaced by addition in $F_{2^{32}}$.

Figure 4.11 : Schematic view of the modified versions of SNOW1.0

Table 4.4 : Experimentally found correlation values

x	Bias
0	0.000009
1	0.000004
2	0.000036
3	0.000012
4	0.000010
5	0.000035
6	0.000018
7	0.000020
8	0.000017
9	0.000032
10	0.000030
11	0.000011
12	**0.000047**
13	0.000012
14	0.000027
15	0.000003
16	0.000004
17	0.000016
18	0.000002
19	0.000010
20	0.000036
21	0.000033
22	0.000026
23	0.000015

$= 2^{-14.38}$

a) *Int. Add. 1* is replaced by addition in $\mathrm{F}_{2^{32}}$.

x	Bias
0	0.000011
1	0.000001
2	0.000038
3	0.000019
4	0.000011
5	0.000007
6	0.000002
7	0.000001
8	0.000038
9	0.000045
10	0.000020
11	0.000010
12	0.000023
13	0.000018
14	0.000008
15	**0.002001**
16	0.000029
17	0.000001
18	0.000008
19	0.000022
20	0.000023
21	0.000029
22	0.000025
23	0.000001

$= 2^{-8.96}$

b) *Int. Add. 1* is replaced by addition in $\mathrm{F}_{2^{32}}$.

4.2.5 Eliminating the “Shift by 7” Operation

In [Ekdahl, Johansson; 2002a], it is told that the reason for the large correlations may be the interaction between the s-box and the shift operation. To discover this we eliminate the “*Shift by 7*” operation from the FSM and search for the correlations. The modified version of SNOW1.0 is shown in Figure 4.12. Approximating the FSM we have the equation

$$(f_t)_x \oplus (f_t)_{x+1} \oplus (f_{t+1})_x \oplus (f_{t+1})_{x+1} = (F_t)_x \oplus (F_{t+1})_{x+1} \qquad (4.33)$$

The largest correlation, that we have found, is $2^{-7.45}$ when x is equal to 5 (Table 4.5). Compared to the correlation found in the original version SNOW1.0, which is $2^{-9.34}$, this result is quite bigger and in contradiction with the prediction made in [Ekdahl, Johansson; 2002a].

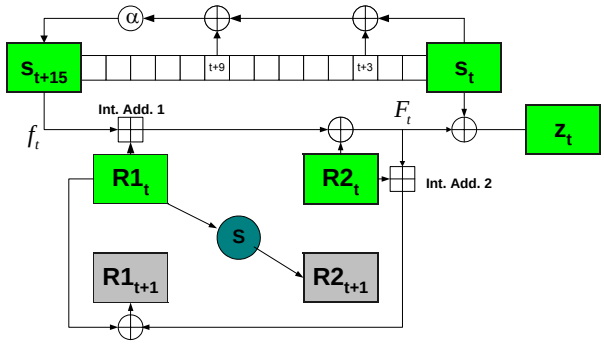


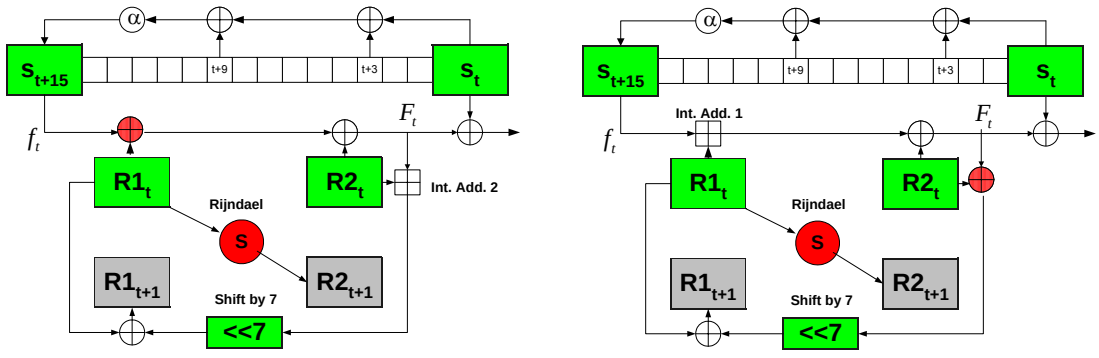
Figure 4.12 : Schematic view of SNOW1.0 without "*Shift by 7*" operation

Table 4.5 : Results of the correlation search after eliminating the "Shift by 7" operation

x	Bias		x	Bias
0	0.000004		15	0.000005
1	0.000030		16	0.000008
2	0.000027		17	0.011453
3	0.000030		18	0.000216
4	0.000021		19	0.000026
5	**0.005725**	$= 2^{-7.45}$	20	0.000110
6	0.000021		21	0.000028
7	0.000053		22	0.002247
8	0.000008		23	0.000057
9	0.000030		24	0.000343
10	0.001379		25	0.000000
11	0.000001		26	0.000742
12	0.000008		27	0.000016
13	0.000038		28	0.002269
14	0.000060		29	0.000002

4.2.6 Both Changing the S-Box and Other Operations

As a final modification, in addition to the s-box we change the integer additions one by one. The modified versions are shown in Figure 4.13. Since the approximations that we obtain in Section 4.2.4 are valid again, while searching for the correlations we use the equations (4.31) and (4.32). The results are tabulated in Table 4.6. Notice that the correlations obtained ($2^{-14.29}$ and $2^{-14.31}$) are approximately the same as the one obtained by only changing the s-box ($2^{-14.23}$). That is to say, there is no interaction between the s-box of Rijndael and the integer additions. When we also take the results obtained in Section 4.2.4 into account, we can say that using a stronger s-box dominates the system and don't let any interactions happen.



a) *Int. Add. 1* is replaced by addition in $\mathrm{F}_{2^{32}}$.

b) *Int. Add. 1* is replaced by addition in $\mathrm{F}_{2^{32}}$.

Figure 4.13 : Both the s-box and the integer additions are changed in SNOW1.0

Table 4.6 : Results of the correlation searches after changing the s-box and replacing integer additions by "additions in $F_{2^{32}}$".

x	Bias
0	-0.000007
1	-0.000039
2	0.000017
3	-0.000007
4	-0.000003
5	-0.000022
6	0.000019
7	-0.000007
8	0.000022
9	0.000018
10	**0.000050**
11	-0.000003
12	-0.000024
13	-0.000009
14	-0.000007
15	0.000041
16	-0.000037
17	0.000004
18	-0.000023
19	-0.000016
20	-0.000026
21	-0.000017
22	-0.000005
23	0.000014

$= 2^{-14.29}$

a) *Int. Add. 1* is replaced by addition in $F_{2^{32}}$.

x	Bias
0	-0.000022
1	0.000033
2	-0.000017
3	0.000004
4	0.000004
5	-0.000035
6	-0.000005
7	-0.000012
8	-0.000004
9	0.000009
10	0.000045
11	-0.000002
12	**0.000049**
13	0.000002
14	-0.000033
15	-0.000028
16	-0.000008
17	-0.000039
18	-0.000030
19	0.000016
20	-0.000025
21	0.000008
22	0.000045
23	0.000008

$= 2^{-14.31}$

b) *Int. Add. 1* is replaced by addition in $F_{2^{32}}$.

4.3 Results of Randomness Tests

As a last but rough comparison, we use the NIST Statistical Test Suite to accomplish the randomness tests on SNOW1.0, SNOW2.0 and also on the FSM of SNOW1.0. In order to prepare the test data, SNOW1.0 and SNOW2.0 are implemented with Borland C++ development environment.

The NIST Statistical Test Suite consists of 16 core statistical tests that, under different parameter inputs, can be viewed as 189 statistical tests. But the results of the 16 core tests will be demonstrated. Brief descriptions of the statistical tests are given in Appendix B. The detailed explanations for the tests and the test suite can be found in [Rukhin; 2001].

The test data prepared is 300×2^{20} bits, that is 300 (sample size) binary sequences of length 2^{20} . The significance level is chosen as 0.01. For each binary sequence and each statistical test, a *P-value* is reported.

A P-value is the probability of obtaining a test statistic as large or larger than the one observed if the sequence is random. Hence, values below significance level (0.01) may be thought as nonrandom.

Two evaluations are made for each statistical test and sample. First, the proportion of binary sequences in a sample that passed the statistical test is calculated (Figure 4.14 (a) and (b)). It is observed that both of the ciphers passed the tests with a good result.

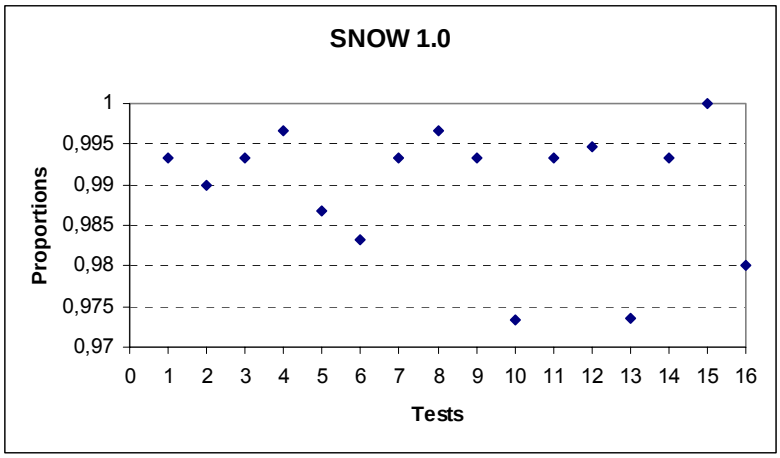


Figure 4.14 (a) : Proportion of sequences passing a test for SNOW1.0

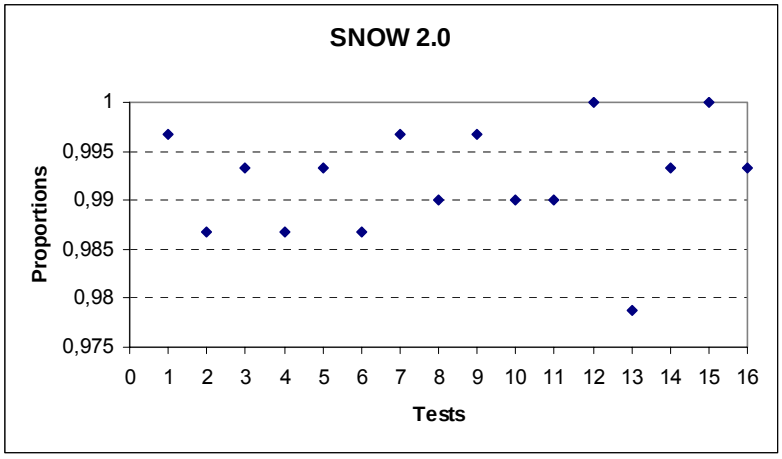


Figure 4.14 (b) : Proportion of sequences passing a test for SNOW2.0

The statistical tests applied are:

1. Frequency Test
2. Frequency Test within a Block
3. Cumulative Sums (Cusum) Test
4. Runs Test
5. Test for the Longest Run of Ones in a Block
6. Binary Matrix Rank Test
7. Discrete Fourier Transform (Spectral) Test
8. Non-overlapping Template Matching Test
9. Overlapping Template Matching Test
10. Maurer's "Universal Statistical" Test
11. Approximate Entropy Test
12. Random Excursions Test
13. Random Excursions Variant Test
14. Serial Test
15. Lempel-Ziv Compression Test
16. Linear Complexity Test

The second evaluation is the calculation of the $P\text{-}value_T$ of the p-values which is done to ensure uniformity. If $P\text{-}value_T \geq 0.0001$, then the sequences can be considered to be uniformly distributed. When we look at Table 4.7, we can see that

all of the p-value$_T$ s are greater than 0.0001. So, both of the ciphers have uniform distribution of p-values.

Table 4.7 : *P-value$_T$* of the p-values for each statistical test on SNOW1.0 and SNOW2.0

	P-value of the P-values	
	SNOW1.0	SNOW2.0
1. Frequency Test	0.481416	0.862344
2. Frequency Test within a Block	0.474986	0.671779
3. Cumulative Sums (Cusum) Test	0.664861	0.049770
4. Runs Test	0.062821	0.609377
5. Test for the Longest Run of Ones in a Block	0.706149	0.195163
6. Binary Matrix Rank Test	0.733228	0.299251
7. Discrete Fourier Transform (Spectral) Test	0.015241	0.122325
8. Non-overlapping Template Matching Test	0.822534	0.236810
9. Overlapping Template Matching Test	0.150906	0.266680
10. Maurer's "Universal Statistical" Test	0.888137	0.401199
11. Approximate Entropy Test	0.514124	0.142602
12. Random Excursions Test	0.915031	0.130453
13. Random Excursions Variant Test	0.689019	0.437274
14. Serial Test	0.828458	0.834308
15. Lempel-Ziv Compression Test	*	*
16. Linear Complexity Test	0.928071	0.372502

We have also applied these tests to the output of the linear approximation of the FSM in SNOW1.0, which is

$$\sigma = (f_t)_{15} \oplus (f_t)_{16} \oplus (f_{t+1})_{22} \oplus (f_{t+1})_{23} \oplus (F_t)_{15} \oplus (F_{t+1})_{23}\,. \tag{4.34}$$

The results obtained from NIST Statistical Test Suite are shown in Figure 4.15 and Table 4.8. The results show that the randomness properties are bad. Moreover, some statistical tests, especially the frequency test, result in failure. This is consistent with the fact that Equation (4.34) has a large bias.

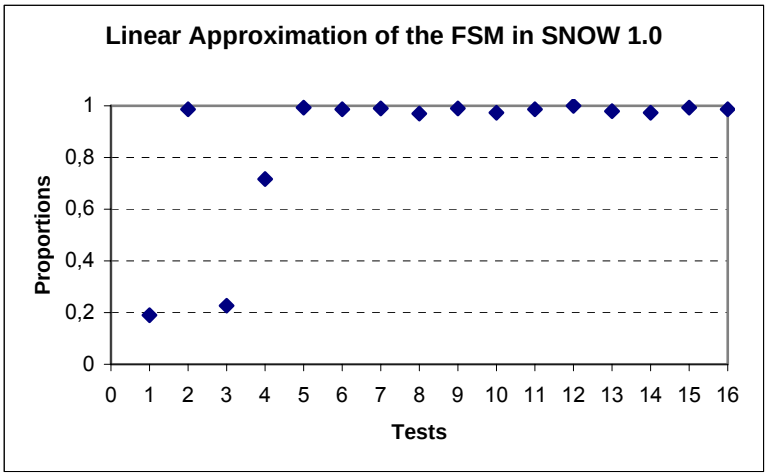


Figure 4.15 : Proportion of sequences passing a test for the linear approximation of the FSM in SNOW1.0

Table 4.8 : *P-value*$_T$ of the p-values for each statistical test on the FSM of SNOW1.0

	P-value of the P-values
1. Frequency Test	0.000000 *
2. Frequency Test within a Block	0.455937
3. Cumulative Sums (Cusum) Test	0.000000 *
4. Runs Test	0.000000 *
5. Test for the Longest Run of Ones in a Block	0.856907
6. Binary Matrix Rank Test	0.746572
7. Discrete Fourier Transform (Spectral) Test	0.547637
8. Non-overlapping Template Matching Test	0.000011 *
9. Overlapping Template Matching Test	0.050845
10. Maurer's "Universal Statistical" Test	0.148094
11. Approximate Entropy Test	0.000068 *
12. Random Excursions Test	0.066882
13. Random Excursions Variant Test	0.105618
14. Serial Test	0.007694
15. Lempel-Ziv Compression Test	0.089121
16. Linear Complexity Test	0.175049

CHAPTER 5

CONCLUSIONS

The designers of the stream cipher SNOW1.0 proposed SNOW2.0 after two attacks have been reported. One of the important changes made was the change of the substitution box. Through our examination on the substitution box of SNOW1.0, nonlinearity and differential uniformity values are found as 96 and 6, respectively. Although these values can not be considered as the values of a weak cipher, when compared with the corresponding values of the s-box taken from Rijndael, which are 112 (nonlinearity) and 4 (differential uniformity), they are inferior. The histogram of Walsh transform values for the s-box of SNOW1.0 (Figure 4.2) shows that for all linear combination functions, Walsh transform distributions are the same. This impressive property is also observed in the s-box of Rijndael.

Using the approach in [Ekdahl, Johansson; 2002b] for the distinguishing attack on SOBER-t32, a review of the distinguishing attack on SNOW1.0 is given. The methods of calculation of the complexities are different in [Ekdahl, Johansson; 2002b] and [Coppersmith, Halevi, Jutla; 2002]. Using the method in [Ekdahl, Johansson; 2002b] we found that we can distinguish the output from a random source using $2^{101.6}$ keystream outputs. This method is more inefficient than the one in [Coppersmith, Halevi, Jutla; 2002], where the data complexity of the attack is 2^{95}. Corresponding process complexity is found as $2^{101.6}$ using the method in [Ekdahl, Johansson; 2002b]; whereas it is 2^{100} in [Coppersmith, Halevi, Jutla; 2002].

Only the change of the s-box in SNOW1.0 with that of Rijndael makes a great deal of improvement in the strength of the cipher against the distinguishing attack, i.e., the number of outputs needed to perform a distinguishing attack is $2^{160.8}$, which is faster than the exhaustive search for the 256-bit key; whereas in SNOW1.0 it is $2^{101.6}$, which is faster than exhaustive search for both 128-bit and 256-bit keys. Also, our correlation search results show that the reason for large correlations is the interaction between the s-box and the first one of the integer addition units rather

than the “*Shift by 7*” operation as expected in [Ekdahl, Johansson; 2002a]. Using a stronger s-box dominates the system and does not let any interactions to happen in the FSM.

REFERENCES

[Arnault, Berger, Necer; 2002] F. Arnault, T. P. Berger, and Abdelkader Necer. A New Class of Stream Ciphers Combining LFSR and FCSR Arhitectures. A. Menezes, P. Sarkar, editors, *INDOCRYPT 2002*, LNCS 2551, pp.22-33, 2002. Springer-Verlag 2002.

[Carlitz, Uchiyama; 1957] L. Carlitz and S. Uchiyama, “Bounds for exponential sums,” Duke Math. J. 24, 1957.

[Coppersmith, Halevi, Jutla; 2002] D. Coppersmith, S.Halevi, C.Jutla. Cryptanalysis of Stream Ciphers with Linear Masking. In M. Yung, editor, Advances in Cryptology- CRYPTO 2002, volume 2442 of *Lecture Notes in Computer Science*, pages 515-532. Springer Verlag, 2002.

[Cover, Thomas; *Elements of Inf. Theory.* 1991] T. Cover and J. A. Thomas. *Elements of Information Theory.* Wiley series in Telecommunication. Wiley, 1991.

[Daemen, Rijmen; 2002] J. Daemen, V. Rijmen. *The Design of Rijndael*. Springer Verlag, 2002.

[Ekdahl, Johansson; 2000] P. Ekdahl, T. Johansson. SNOW- a new stream cipher. In *Proceedings of First Open NESSIE Workshop,2000.*

[Ekdahl, Johansson; 2002a] P. Ekdahl, T. Johansson. A new version of the stream cipher SNOW. In K. Nyberg and H. Heys, editors, *Selected Areas in Cryptography*- SAC 2002, volume 2595 of *Lecture Notes in Computer Science*, pages 47-61. Springer Verlag, 2002.

[Ekdahl, Johansson; 2002b] P. Ekdahl, T. Johansson, “Distinguishing attacks on SOBER-t16 and t-32,” *Fast Software Encryption, FSE 2002,* Springer –Verlag, LNCS 2365, pp.210-224, 2002.

[Golic; 1996] J. Golic, “Linear models for keystream generator,” *IEEE Trans. Computers,* vol. C-45, pp. 41-49, 1996.

[*Handbook of Applied Cryptography; 1997*] A. Menezes, P. von Oorschot and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[Hawkes, Rose; 2000] Primitive specification and supporting documentation for SOBER-t32 submission to NESSIE. In *Proceedings of the First Open NESSIE Workshop,* 13-14 November 2000, Heverlee, Belgium.

[Hawkes, Rose; 2002] P. Hawkes and G. G. Rose. Guess-and-determine attacks on SNOW. In K. Nyberg and H. Heys, editors, *Selected Areas in Cryptography*- SAC 2002, volume 2595 of *Lecture Notes in Computer Science*, pages 37-46. Springer Verlag, 2002.

[Massey; 1969] J. R. Massey. Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, 15:122-127, 1969.

[Matsui; 1993] M. Matsui. Linear Cryptanalysis Method for DES cipher. In T. Helleseth, editor, *Advances in Cryptology- EUROCRYPT’93* volume 765 of *Lecture notes in Computer Science*, pages 386-397. Springer Verlag, 1994.

[NESSIE Sec. Rep.; 2003] NESSIE Security Report, NES/ DOC/ ENS/ WP5/ D20/ 2, February 19, 2003.

[Nyberg; 1993] K. Nyberg. “Differentially uniform mappings for cryptography,” *Advances in Cryptology- EUROCRYPT’93*.

[Robshaw, RSA Lab. Tech. Rep.; 1995] M. J. B Robshaw, Stream Ciphers, RSA Laboratories Technical Report TR-701, Version 2.0-July 25, 1995.

[Rueppel; 1984] R. A. Rueppel. *New Approaches to Stream Ciphers,* PhD Thesis, Swiss Federal Institute of Technology, Zurich,1984.

[Rueppel;1986] R. A. Rueppel. Analysis and Design of Stream Ciphers. Springer-Verlag, 1986 [p.103].

[Rukhin; 2001] A. Rukhin, et. al., A Statistical Test Suite for the Validation of Random Number Generators and Pseudorandom Number Generators for Cryptographic Applications. *NIST Special Publication 800-22,* 2001. The test suite is available at http://csrc.nist.gov/rng/rng2.html, accessed in April, 2004.

[Schneier, *Applied Cryptography;* 1996] B. Schneier. *Applied Cryptography: Protocols, Algorithms and Source Codes in C*. John Wiley and Sons, Newyork, 2nd edition, 1996.

[Siegenthaler; 1984] Siegenthaler. Correlation Immunity of Nonlinear Combining Functions for Cryptographic Applications, *IEEE Transactions on Information Theory*, 30:776-780, 1984.

[Soto, Basham; NIST, 2000] Soto J. and Basham L.: Randomness Testing of the Advanced Encryption Standard Finalist Candidates. NIST Publication, March 2000.

[STORK; 2003] STORK. Strategic Roadmap for Crypto. Available at http://www.stork.eu.org, accessed in May, 2004.

[Watanabe, Biryukov, Canniere; 2003] D. Watanabe, A. Biryukov, and C. D. Canniere. A Distinguishing Attack of SNOW2.0 with Linear Masking Method. In *Selected Areas in Cryptography*- SAC 2003. To be published in *Lecture Notes in Computer Science*. Springer Verlag, 2003.

[Zhang, Zheng, Imai; 1998] Xian-Mo Zhang, Yulian Zheng and Hideki Imai. Relating Differential Distribution Tables to Other Properties of Substitution Boxes. *Designs , codes and Cryptography*, vol.19, pp.45-63,1998.

APPENDIX A

S-Boxes of SNOW1.0 and Rijndael

SNOW1.0 S-Box:

$\{S(\mathbf{x})|_{\mathbf{x}=0}^{255}\} = \{$7, 6, 135, 248, 226, 214, 193, 42, 53, 244, 246, 199, 120, 189, 155, 15, 185, 46, 59, 41, 16, 250, 5, 203, 36, 161, 240, 126, 107, 162, 171, 200, 166, 32, 205, 35, 104, 138, 48, 27, 137, 180, 234, 49, 44, 68, 255, 208, 74, 213, 121, 117, 109, 222, 89, 216, 252, 10, 210, 57, 254, 21, 84, 3, 132, 134, 28, 14, 130, 4, 183, 128, 87, 1, 152, 80, 111, 140, 142, 82, 51, 33, 220, 37, 153, 71, 239, 123, 230, 114, 75, 186, 60, 85, 139, 38, 154, 201, 93, 8, 164, 34, 94, 127, 129, 253, 90, 174, 66, 218, 39, 150, 151, 91, 125, 76, 160, 241, 67, 78, 188, 141, 95, 29, 56, 165, 81, 30, 73, 131, 98, 50, 115, 211, 43, 176, 52, 177, 172, 61, 119, 169, 31, 116, 147, 22, 122, 158, 192, 103, 110, 9, 170, 178, 229, 18, 206, 197, 69, 0, 196, 11, 156, 223, 113, 133, 202, 19, 64, 187, 224, 25, 79, 190, 143, 221, 99, 108, 163, 219, 26, 102, 13, 167, 149, 124, 184, 168, 24, 179, 97, 146, 235, 245, 212, 62, 20, 209, 86, 198, 175, 106, 55, 136, 191, 144, 217, 45, 159, 72, 23, 207, 233, 173, 40, 194, 54, 232, 12, 83, 225, 157, 17, 77, 145, 92, 105, 70, 242, 181, 231, 227, 249, 148, 237, 236, 182, 96, 204, 215, 112, 251, 101, 247, 63, 118, 243, 2, 195, 58, 47, 65, 238, 228, 88, 100}

RIJNDAEL S-Box:

$\{S(\mathbf{x})|_{\mathbf{x}=0}^{255}\} = \{$99, 124, 119, 123, 242, 107, 111, 197, 48, 1, 103, 43, 254, 215, 171, 118, 202, 130, 201, 125, 250, 89, 71, 240, 173, 212, 162, 175, 156, 164, 114, 192, 183, 253, 147, 38, 54, 63, 247, 204, 52, 165, 229, 241, 113, 216, 49, 21, 4, 199, 35, 195, 24, 150, 5, 154, 7, 18, 128, 226, 235, 39, 178, 117, 9, 131, 44, 26, 27, 110, 90, 160, 82, 59, 214, 179, 41, 227, 47, 132, 83, 209, 0, 237, 32, 252, 177, 91, 106, 203, 190, 57, 74, 76, 88, 207, 208, 239, 170, 251, 67, 77, 51, 133, 69, 249, 2, 127, 80, 60, 159, 168, 81, 163, 64, 143, 146, 157, 56, 245, 188, 182, 218, 33, 16, 255, 243, 210, 205, 12, 19, 236, 95, 151, 68, 23, 196, 167, 126, 61, 100, 93, 25, 115, 96, 129, 79, 220, 34, 42, 144, 136, 70, 238, 184, 20, 222, 94, 11, 219, 224, 50, 58, 10, 73, 6, 36, 92, 194, 211, 172, 98, 145, 149, 228, 121, 231, 200, 55, 109, 141, 213, 78, 169, 108, 86, 244, 234, 101, 122, 174, 8, 186, 120, 37, 46, 28, 166, 180, 198, 232, 221, 116, 31, 75, 189, 139, 138, 112, 62, 181, 102, 72, 3, 246, 14, 97, 53, 87, 185, 134, 193, 29, 158, 225, 248, 152, 17, 105, 217, 142, 148, 155, 30, 135, 233, 206, 85, 40, 223, 140, 161, 137, 13, 191, 230, 66, 104, 65, 153, 45, 15, 176, 84, 187, 22}

APPENDIX B

Description of Statistical Tests

Monobit Test: The purpose of this test is to determine whether the number of ones and zeros in a sequence are approximately the same as would be expected for a truly random sequence.

Block Frequency Test: The purpose of the block frequency test is to determine whether the number of ones and zeros in each of M non-overlapping blocks created from a sequence appear to have a random distribution.

Cumulative Sums Forward (Reverse) Test: The purpose of the cumulative sums test is to determine whether the sum of the partial sequences occurring in the tested sequence is too large or too small.

Runs Test: The purpose of the runs test is to determine whether the number of runs of ones and zeros of various lengths is as expected for a random sequence. In particular, this test determines whether the oscillation between such substrings is too fast or too slow.

Long Runs of Ones Test: The purpose of this test is to determine whether the longest run of ones within the tested sequence is consistent with the longest run of ones that would be expected in a random sequence.

Rank Test: The purpose of this test is to check for linear dependence among fixed length substrings of the original sequence.

Discrete Fourier Transform (Spectral) Test: The purpose of this test is to detect periodic features (i.e., repetitive patterns that are near each other) in the tested sequence that would indicate a deviation from the assumption of randomness.

Aperiodic Templates Test: The purpose of this test is to reject sequences that exhibit too many occurrences of a given non-periodic (aperiodic) pattern.

Periodic Template Test: The purpose of this test is to reject sequences that show deviations from the expected number of runs of ones of a given length.

Universal Statistical Test: The purpose of the test is to detect whether or not the sequence can be significantly compressed without loss of information. A compressible sequence is considered to be nonrandom.

Approximate Entropy Test: The purpose of the test is to compare the frequency of overlapping blocks of two consecutive/adjacent lengths (*m* and *m+1*) against the expected result for a normally distributed sequence.

Random Excursion Test: The purpose of this test is to determine if the number of visits to a state within a random walk exceeds what one would expect for a random sequence.

Random Excursion Variant Test: The purpose of this test is to detect deviations from the distribution of the number of visits of a random walk to a certain state.

Serial Test: The purpose of this test is to determine whether the number of occurrences of *m*-bit overlapping patterns is approximately the same as would be expected for a random sequence.

Lempel-Ziv Complexity Test: The purpose of the test is to determine how far the tested sequence can be compressed. The sequence is considered to be non-random if it can be significantly compressed

Linear Complexity Test: The purpose of this test is to determine whether or not the sequence is complex enough to be considered random.