EXAMINATION OF BLOCK CIPHERS RIJNDAEL, SAFER K-64 AND A SLIDE ATTACK ON SPECTR-H64

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

BY

SELÇUK KAVUT

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

JANUARY 2002

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. Tayfur ÖZTÜRK
Director

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

Prof. Dr. Mübeccel DEMİREKLER
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 TANIK (Chairperson)

Assoc. Prof. Dr. Melek D. YÜCEL

Prof. Dr. Ersan AKYILDIZ

Prof. Dr. Kemal LEBLEBİCİOĞLU

Dr. Tolga TÜFEKÇİ

ABSTRACT

EXAMINATION OF BLOCK CIPHERS RIJNDAEL, SAFER K-64 AND A SLIDE ATTACK ON SPECTR-H64

Kavut, Selçuk

M.S., Department of Electrical and Electronics Engineering

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

January 2002, 90 pages

In this thesis, some of the symmetric (single-key) block ciphers, Rijndael, Safer K-64, selected among the most popular ones in the open literature, are studied; and a slide attack on Spectr-H64 is presented. Since the substitution boxes (s-boxes) of Rijndael and Safer K-64 apply nonlinear transformations operating on each intermediate cipher result independently, their characteristics have significant effects on the strength of the block ciphers. Therefore, the s-box characteristics of Rijndael and Safer K-64 are investigated for the criteria of completeness, avalanche, strict avalanche, bit independence, nonlinearity and XOR table distribution. The overall performance for different rounds of Rijndael and Safer

K-64, in terms of Avalanche Weight Distribution (AWD) criterion, is also evaluated.

Diffusion characteristics of Spectr H-64 in each round are then compared to those of Rijndael and Safer K-64 in terms of AWD criterion and a weakness in the round transformation of Spectr-H64 is observed. This weakness is exploited first to break one round of Spectr-H64, and then to develop a slide attack on the overall cipher, which works for 2^{32} elements of the key space (out of 2^{256} keys). Moreover, 2^{128} weak keys of Spectr-H64 are found, for which encryption becomes the same function as decryption. Corresponding to each weak key, 2^{32} fixed points are observed, i.e., the ciphertext is the same as the plaintext after encryption.

Key words: Block ciphers, Rijndael, Safer K-64, Spectr-H64, s-box, slide attacks.

ÖZ

RIJNDAEL, SAFER K-64 BLOK ŞİFRELERİNİN İNCELENMESİ VE SPECTR-H64'E BİR KAYMA SALDIRISI

Kavut, Selçuk
Yüksek Lisans, Elektrik and Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Melek D. Yücel

Ocak 2002, 90 sayfa

Bu tezde, günümüzde açık literatürdeki popüler simetrik (tek-anahtarlı) blok şifre algoritmalarından bazıları olan Rijndael ve Safer K-64 üzerinde çalışılmış, ayrıca Spectr-H64'e bir kayma saldırısı yapılmıştır . Rijndael ve Safer algoritmalarının yerleştirme kutuları (s-kutuları), her çevrimde elde edilen şifrelenmiş ara değerlere doğrusal olmayan dönüşümler uyguladıkları için, bu algoritmalarının güvenliği üzerinde önemli etkilere sahiptir. Bu yüzden, Rijndael ve Safer s-kutularının özellikleri eksiksizlik, çığ, katı çığ, ikil bağımsızlığı, doğrusal olmama ve XOR tablo dağılımı ölçütlerine göre araştırılmıştır. Rijndael ve Safer K-64'ün genel başarımları ayrıca Çığ Ağırlık Dağılımı (ÇAD) ölçütüne göre, farklı çevrim sayıları için bulunmuştur.

Spectr-H64'ün yayılma özellikleri, Rijndael ve Safer K-64 şifrelerinin yayılma özellikleri ile her çevrim için karşılaştırılmış ve Spectr-H64 çevrim dönüşümünde bir zayıflık gözlenmiştir. Bu zayıflık önce Spectr-H64'ün bir çevrimini kırmak için kullanılmış ve daha sonra, anahtar uzayının (toplam 2^{256} anahtar içinden) 2^{32} elemanı için geçerli olan ve şifrenin tümüne yönelik bir kayma saldırısı geliştirilmiştir. Ayrıca, Spectr-H64'ün şifreleme ve şifre çözme işlemlerinin aynı olmasına neden olan 2^{128} tane zayıf anahtarı bulunmuş, ve her zayıf anahtar için 2^{32} tane sabit nokta (algoritma girdi ve çıktısının aynı olduğu durum) gözlenmiştir.

Anahtar Sözcükler: Blok şifre, Rijndael, Safer K-64, Spectr-H64, s-kutusu, kayma saldırısı.

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to Assoc. Prof. Dr. Melek D. Yücel. I have been fortunate to receive her supervision, valuable guidance and encouragement throughout this thesis.

I am grateful to my dear family for their love, encouragement and support.

I would like to express my sincere appreciation to İzzet Özçelik, Özgür İnce, Raed Murad and Ahmet Akbulut for their encouragement and support.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF SYMBOLS

f	Cryptographic function, which maps n input bits to m output bits; it may represent either an s-box or a block cipher according to the context
$\mathbf{i}$, i	n-bit input difference vector, and its integer value respectively
$\mathbf{j}$, j	m-bit mask vector of output bits, and its integer value respectively
$\mathbf{X}$, $\mathbf{X}_i$	n-bit input vectors, so that $\mathbf{X}_i = \mathbf{X} \oplus \mathbf{i}$
e_k	Unit vector with a single 1 in position k
$\boldsymbol{\delta}_f^i$	m-bit avalanche (output difference) vector corresponding to the input difference vector $\mathbf{i}$, at the output of the cryptographic function f
δ_k^i	k^{th} avalanche variable, which is the k^{th} entry of $\boldsymbol{\delta}_f^i$ (if used with the subscript j, corresponds to the linear combination of the avalanche variables, $\boldsymbol{\delta}_f^i$ masked by $\mathbf{j}$)
$w(\delta_k^i)$	Total change in the k^{th} avalanche variable over all possible input vectors
δ_j^i	The generalized avalanche variable $(\boldsymbol{\delta}_f^i \cdot \mathbf{j})$, or the "Avalanche vector masked by $\mathbf{j}$", obtained by masking the avalanche vector $\boldsymbol{\delta}_f^i$ by the output mask vector $\mathbf{j}$
$w(\delta_j^i)$	Total change in the generalized avalanche variable (or the "avalanche vector masked by $\mathbf{j}$"), over all possible input vectors

$p_{AVAL}(i)$	AVAL (avalanche) parameter, the change probability of output bits, corresponding to an input difference vector **i**
$p_{SAC}(i, j)$	SAC (strict avalanche criterion) parameter, the change probability of a linear combination of output bits masked by **j**, corresponding to an input difference vector **i**
$BIC(\boldsymbol{\delta}_f)$	BIC (bit independence criterion) parameter, the maximum absolute value of the normalized covariance (also known as the correlation coefficient, but not the correlation) of the avalanche variable pairs, corresponding to the cryptographic function f
$XOR_f(\mathbf{i}, \delta)$	An entry in the XOR (exclusive or) table of the cryptographic function f, indexed by (**i**, δ), corresponding to the input difference vector **i**, and the output difference vector δ
NLM_f	Nonlinearity parameter of the cryptographic function f
$\in_{AVAL}$	Maximum absolute relative error for the avalanche criterion (AVAL)
$\in_{SAC}$	Maximum absolute relative error for the strict avalanche criterion (SAC)
$\in_{BIC}$	Maximum absolute relative error for the bit independence criterion (BIC), which is also equal to $BIC(\boldsymbol{\delta}_f)$
$wt(.)$	Hamming weight (number of nonzero elements) of a vector
$R_r^{e_i}$	Resemblance parameter corresponding to input difference e_i, and r rounds of encryption
$r_j(i)$	Autocorrelation function of the output bits masked by **j**

CHAPTER 1

INTRODUCTION

1.1 Cryptography Background

Cryptography, the study of secret (*crypto-*) writing (*-graphy*), is the art and science of transforming information into an intermediate form which secures that information while in storage or in transit.

Plaintext is the original, readable message. It is convenient to think of plaintext as being actual language characters, but may be any other symbols or values (such as arbitrary computer data) which need to be protected. *Encryption* is the general term for hiding information or message; encrypted message is called *ciphertext*. Ciphertext contains the same information as the original plaintext, but hides the original information. The process of extracting information, which was hidden by encryption, is called *decryption*. All modern cryptographic systems usually make use of a *key* to control encryption and decryption.

Cryptanalysis is the science of recovering the plaintext of a message from the ciphertext without access to the key. The branch of mathematics encompassing both cryptography and cryptanalysis is called *cryptology*.

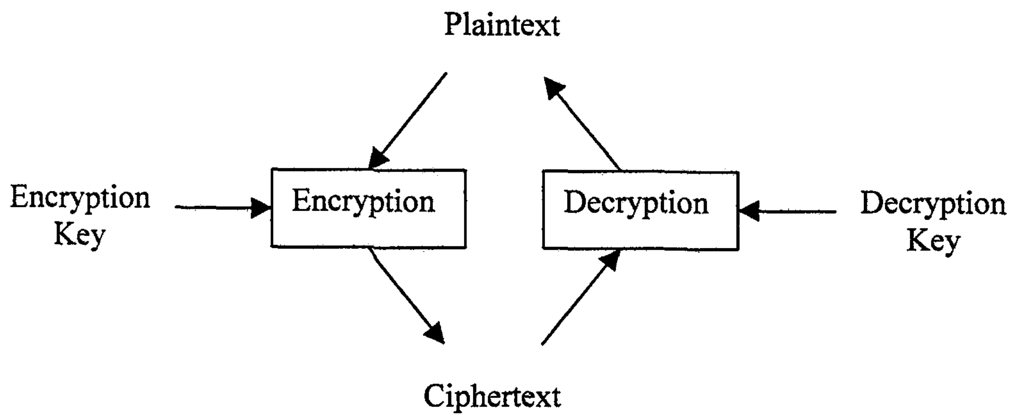

Figure 1.1 A Basic Cryptosystem

The actual mathematical function used to encrypt and decrypt messages is called a *cryptographic algorithm* or *cipher*. There are two general forms of key-based cyptographic algorithms: *symmetric* (or *secret-key*, or *single-key*) and *asymmetric* (or *public-key*, or *double-key*). A basic structure of a cyptosystem is shown in Fig.1.1.

In a public-key cryptosystem, the key that is used to encrypt the message is different from the key used to decrypt the message. The encryption key, known as the *public key* is used to encrypt a message, but the message can only be decrypted by the person who has the decryption key, known as the *private key*. Resulting from its design principles, deriving the private key from the public key is computationally infeasible.

Symmetric algorithms have one key that is used both to encrypt and decrypt the message; therefore, these algorithms require the sender and the recipient to share the same key. All the security of a symmetric algorithm entirely lies in the key, thus the key must be kept secret. There are two types of symmetric algorithms: *stream ciphers* and *block ciphers*. Stream ciphers operate on plaintext one bit at a time. Block ciphers operate on groups of bits called blocks. The block ciphers examined in this thesis are AES-Rijndael, Safer K-64 and Spectr-H64.

Both the symmetric and public-key algorithms can be broken given infinite time and resources. Modern cryptography relies on making it computationally infeasible to break an algorithm; this means that while it is theoretically possible, the time-scale and resources involved make it completely unrealistic. If an algorithm is presumed to be perfect, then the only method of breaking it relies on trying every possible key until the resulting ciphertext makes sense. This type of attack is called *brute-force attack.*

1.2 Scope of Thesis

The main purpose of this study is to examine cryptographic properties of the block ciphers Advanced Encryption Standard (AES)-Rijndael, Safer K-64 and to develop a slide attack on Spectr-H64. Some security test criteria of block ciphers appearing in literature such as completeness, avalanche, strict avalanche, bit independence, XOR table distribution, nonlinearity and Avalanche Weight Distribution (AWD) are investigated. The block ciphers AES-Rijndael and Safer K-64 are examined in detail with respect to all these criteria and the results are compared.

We also evaluate the overall performance of Rijndael, Safer K-64 and Spectr-H64 round by round, in terms of Avalanche Weight Distribution criterion. We find a weak diffusion property of Spectr-H64's round transformation, which can be observed through the AWD criterion, and exploit this weakness to develop a chosen plaintext slide attack on Spectr-H64, which works for 2^{32} elements of the key space (out of 2^{256} keys). Moreover, 2^{128} weak keys of Spectr-H64 are found, for which encryption becomes the same function as decryption. Corresponding to each weak key, 2^{32} fixed points are observed, i.e., the ciphertext is the same as the plaintext after encryption.

1.3 Outline of Thesis

Chapter 1 gives a brief introduction to the basic concepts of modern cryptography and summary of the thesis.

In Chapter 2, the information about the structures of the block ciphers AES-Rijndael, Safer K-64 and Spectr-H64 is given in detail.

In Chapter 3, some well-known cryptographic test criteria appearing in the literature such as completeness, avalanche, strict avalanche, bit independence, XOR table distribution, nonlinearity and Avalanche Weight Distribution (AWD) are described.

In Chapter 4, the test results for the block ciphers AES-Rijndael, Safer K-64 and Spectr-H64 are presented.

In Chapter 5, we break one-round encryption of Spectr-H64 and develop a slide attack on the overall algorithm of Spectr-H64. In addition, we find a large set of weak keys, and observe some fixed points.

Finally, Chapter 6 discusses the results of the work done in this thesis.

CHAPTER 2

BLOCK CIPHERS

In symmetric cryptography, a single key is used for both encryption and decryption. The sender uses the key to encrypt the plaintext and sends the ciphertext to the recipient. The recipient applies the same key to decrypt the message and recover the plaintext. With this form of cryptography, the key must be known to both the sender and the recipient; that, in fact, is the secret.

There are several widely used symmetric key cryptography schemes and they are generally categorized as being either block ciphers or stream ciphers. A block cipher is so called because it encrypts blocks of data at a time; the same plaintext block will always be encrypted into the same ciphertext when using the same key. Block ciphers can also be used to construct hash functions, and message authentication codes. Due to their flexibility for encryption, block ciphers have an important place in cryptographic algorithms. Stream ciphers operate on a single bit, byte, or word at a time, and implement a feedback mechanism so that the same plaintext will yield different ciphertext every time it is encrypted.

This chapter gives sufficient information about block ciphers Rijndael, Safer K-64 and Spectr-H64, on which this thesis is studied.

2.1 Rijndael (Advanced Encryption Standard)

2.1.1 Introduction

Rijndael [1] is a symmetric block cipher with variable block length and variable key length, designed by Joan Daemen and Vincent Rijmen; and selected as the Advanced Encryption Standard (AES) by National Institute of Standards and Technologies (NIST). The plaintext/ciphertext block length and the key length can be independently specified to 128, 192, or 256 bits. The number of iteration rounds given in Table 2.1 depends on the values corresponding to the block length and the key length of the cipher.

Table 2.1 Number of Rounds of Rijndael as a Function of Block and Key Lengths

Key length in bits	*Input block length in bits*		
	128	192	256
128	10	12	14
192	12	12	14
256	14	14	14

All nine combinations of the block length and the key length differ from each other only in their key schedules and in the number of rounds used. Therefore, examining the diffusion property of Rijndael with 128-bit block length and 128-bit key length also gives some information about the diffusion properties of other eight versions of the block cipher.

2.1.2 Algorithm

The computational graph of the encryption process of Rijndael with 128-bit block length and 128-bit key length is shown in Fig.2.1. From the figure, it is seen that the encryption process consists of an initial round key addition, 9 similar rounds and a final round. The round keys K_0, K_2, .., K_{10} are derived from the 128-bit user selected round key K_0 in a manner that will be explained in section 2.1.3.

In the encryption process, four different group operations on intermediate cipher results are performed, namely,

- bit-by-bit XOR operation on pairs of bytes, denoted as $\oplus$,
- nonlinear byte substitution, which operates on each byte of the intermediate cipher result independently, denoted as S,
- byte-wise permutations,
- polynomial multiplication over Z_{2^8}, denoted as $\otimes c(x)$.

The substitution table (or s-box) is obtained through the non-linear byte substitution; which is in turn implemented by two transformations:

- First taking multiplicative inverse in $GF(2^8)$. In binary representation '0000 0000' is mapped onto itself.
- Then applying an affine transformation over Z_2 defined by:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \tag{2.1}$$

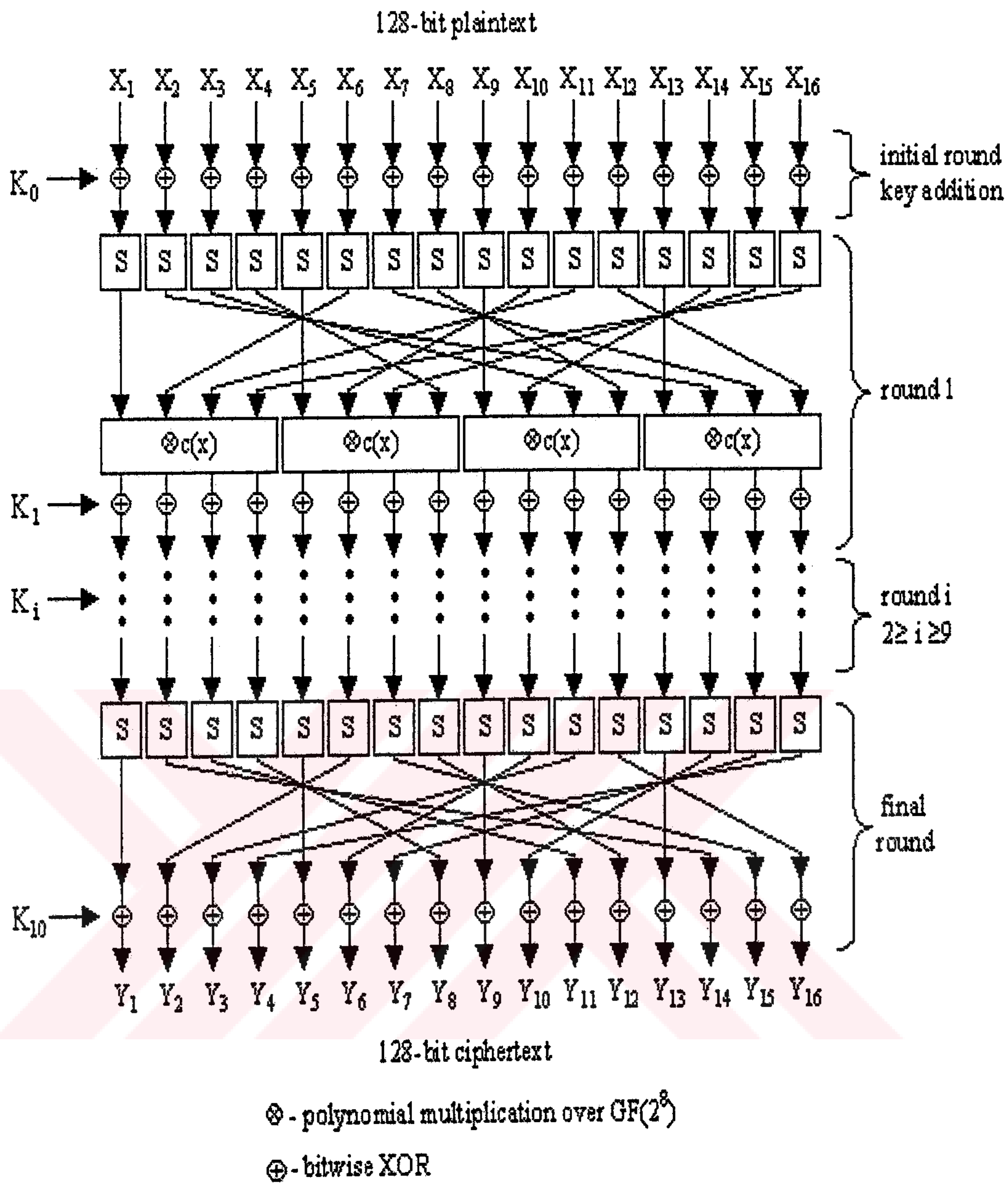

Figure 2.1 Encryption Scheme of Rijndael with 128-bit Block and Key Length

where (x_0, x_1, ..., x_7) is the multiplicative inverse of the byte at the input of the s-box.

The polynomial multiplication over $Z_2^{\,8}$, denoted as $\otimes c(x)$, is performed as follows: Each 4-byte vector of intermediate cipher result is considered as a polynomial over Z_{2^8} and multiplied modulo $x^4 + 1$ with a fixed polynomial c(x), given by

$$c(x) = \text{'03'}\, x^3 + \text{'01'}\, x^2 + \text{'01'}\, x + \text{'02'} \tag{2.2}$$

where the coefficients of c(x) are represented in hexadecimal form.

The inverse of the polynomial multiplication ⊗c(x) is performed similarly. Every 4-byte vector is transformed by multiplying it with a specific multiplication polynomial d(x), which is the multiplicative inverse of c(x), given by

$$d(x) = \text{'0B'}\, x^3 + \text{'0D'}\, x^2 + \text{'09'}\, x + \text{'0E'} \tag{2.3}$$

In the round transformation of Rijndael, the only nonlinear step is the first transformation, and the bytewise permutation is applied before the polynomial multiplication. In the inverse of a round, the transformations in the round are replaced with their inverses and the order of the transformations is reversed; consequently the inverse of the nonlinear step will end up being the last step of the inverse round and the inverse of the bytewise permutation is applied after the inverse of the polynomial multiplication. However, the structure of Rijndael is such that the sequence of transformations of decryption is equal to that of the cipher itself, with the transformations replaced by their inverses and a change in the key schedule (see section 2.1.3).

2.1.3 Key Schedule

The 128-bit user selected secret key, which is directly used as the round key K_0, is first partitioned into 4 subblocks W_0, W_1, W_2 and W_3. The transformation shown in Fig.2.2 is then applied for all $k \in \{1, 2, \ldots, 10\}$, which gives us ($W_{4k}$, W_{4k+1}, W_{4k+2}, W_{4k+3}).

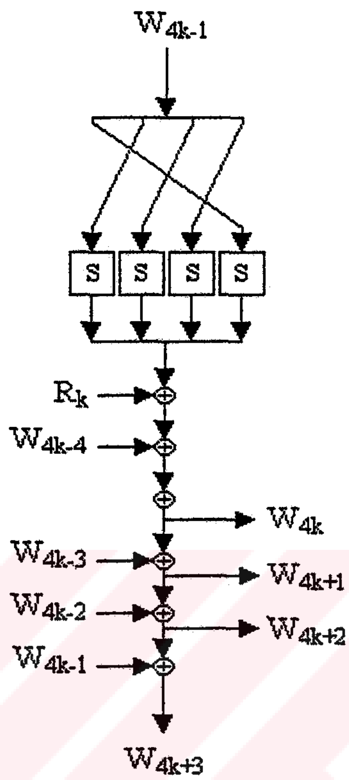

Figure 2.2 Key Schedule Structure of Rijndael with 128-bit Block and Key Length ($k \in \{1, 2, \ldots, 10\}$)

In Fig. 2.2, R_k denotes the round constants and it is defined, in the hexadecimal form, by:

$$R_k = (RC_k, \text{'00'}, \text{'00'}, \text{'00'}) \quad \textbf{(2.4)}$$

with RC_k representing an element in Z_{2^8} with a value x^{k-1} so that:

$$RC_1 = 1 \text{ (i.e. '01')} \quad \textbf{(2.5a)}$$

$$RC_k = x \text{ (i.e. '02')} \cdot RC_{k-1} \quad \textbf{(2.5b)}$$

It is seen from Fig. 2.2 that the 32-bit vector W_{4k-1} is first partitioned into four bytes, and the nonlinear byte substitution is applied after a simple bytewise permutation. Then, the resulting vector is XORed with R_k and W_{4k-4} yielding W_{4k}. The vectors W_{4k+1}, W_{4k+2}, W_{4k+3} are obtained through XOR operations of the vector pairs W_{4k} and W_{4k-3}, W_{4k+1} and W_{4k-2}, W_{4k+2} and W_{4k-1}, respectively.

After applying the transformation shown in Fig. 2.2 for all k, the round keys are determined by:

$$K_k = (W_{4k}, W_{4k+1}, W_{4k+2}, W_{4k+3}) \qquad \textbf{(2.6)}$$

For decryption process of Rijndael, the inverse of the polynomial multiplication is applied to all round keys except the first and last one (K_0 and K_{10}). Then the resulting round keys are used in reverse order.

2.2 Safer K-64

2.2.1 Introduction

Secure **A**nd **F**ast **E**ncryption **R**outine with a **K**ey of **64** bits (Safer K-64) is an iterated block cipher with 64-bit plaintext and ciphertext blocks, designed by Massey [2]. Safer K-40 and Safer K-128 were proposed afterwards, which differ from Safer K-64 in their key schedules and the number of rounds used.
After Knudsen found a weakness in the key schedule of the algorithm and suggested a stronger key schedule [3], this stronger key schedule was adopted by Massey and the new cipher was named Safer SK (Safer with a Strengthened Key Schedule).

2.2.2 Algorithm

The detailed encryption structure of Safer K-64 is shown in Fig. 2.3. The encryption algorithm of Safer K-64 consists of r rounds of identical transformations that are applied in sequence to the plaintext, followed by an output transformation, to produce the ciphertext. For secure encryption, at least $r = 6$ is recommended, but larger values of r may be used for even greater security (maximum $r = 10$). The round keys (K_1, K_2, .., K_{2r+1}) are derived from the 64-bit user selected round key K_1 in a manner that will be explained in section 2.2.3. Each round is controlled by two 64-bit round keys, and the output transformation is controlled by one round key. The output transformation is the mixed XOR/byte-addition of the r^{th} round output with the round key K_{2r+1}.

In the round transformation of Safer K-64, the first step is the mixed XOR/byte-addition of the round input with the round key K_{2i-1}. The eight bytes of the result are then passed through a nonlinear layer and individually subjected to one of two different s-boxes. One of these s-boxes is named as "exponentiating s-box", which is labelled "S" in Fig. 2.3, and it is formed according to some rule as:

- if the input byte into the s-box is an integer x, the corresponding output byte is 45^x modulo 257, except that the output byte is 0 if the modular result 256, which occurs for $x = 128$.

The other s-box named as "logarithm taking s-box", which is labelled "S^{-1}" in Fig.2.3, and it is formed according to the following rule:

- if the input byte into the s-box is an integer x, the corresponding output byte is $\log_{45}(x)$, except that the output byte is 128 if the input byte $x = 0$.

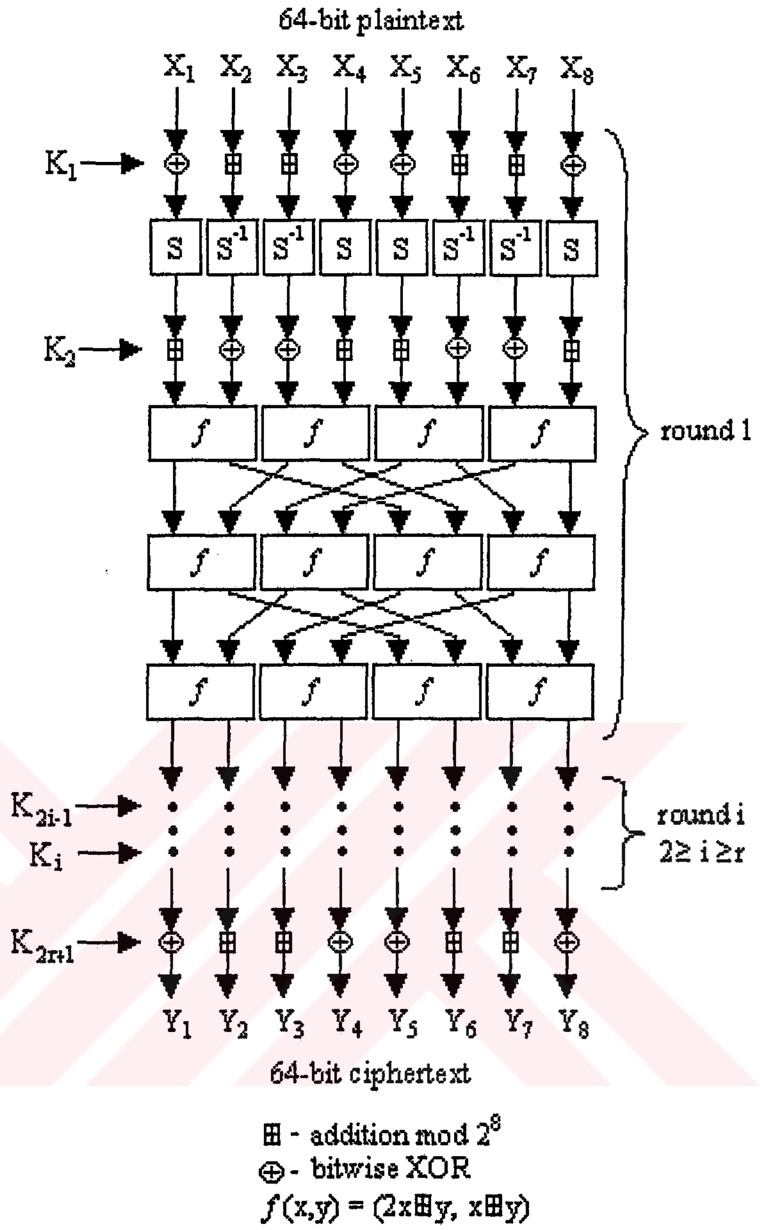

Figure 2.3 Encryption Scheme of Safer K-64 (*r* rounds)

After the nonlinear transformation performed by the s-boxes, the bytes of the resulting vector are passed through another mixed XOR/byte-addition with the round key K_{2i}. The last step of the round transformation is the three-level linear layer of the boxes that are labelled "*f*" in Fig. 2.3. The transformation performed by the *f*-boxes, which is defined in the figure, allows the cipher to achieve the desired diffusion rapidly.

In the decryption algorithm of Safer K-64, all the transformations in the encryption process are first replaced with their inverses. More specifically, the XOR/byte-addition layers are converted to the XOR/byte-subtraction layers, the three-level linear layer is inverted, and also the nonlinear layer of s-boxes is inverted. Then, the inverse transformations are applied in reverse order, and the round keys are also reversed. However, note that, since the logarithm taking s-box is obtained by inverting the exponentiating s-box and vice versa, the inverse of the nonlinear layer is easily obtained by interchanging the s-boxes. Therefore, the overall decryption structure of the algorithm consists of an input transformation that is the mixed XOR/byte-subtraction of the ciphertext block from the round key K_{2r+1}, followed by r rounds of decryption under the control of the round keys in reverse order.

2.2.3 Key Schedule

The key schedule structure of Safer K-64, i.e., the procedure for generating the round keys $K_2, K_3, \ldots, K_{2r+1}$ from the user-selected round key K_1, is indicated in Fig. 2.4. The quantities $B_2, B_3, \ldots, B_{2r+1}$ are the constants called "additive key biases", which are intended to behave as random numbers.

Letting $B_i[j]$ denote the j^{th} byte of B_i, and $K_i[j]$ denote the j^{th} byte of K_i, the round keys are obtained through the following equations:

$$B_i[j] = 45^{45^{((9i+j) \bmod 257)}} \bmod 257 \qquad \textbf{(2.7)}$$

$$K_i[j] = (K_1[j] <<<(3i-1)) + B_i[j] \bmod 256 \qquad \textbf{(2.8)}$$

where $i \in \{2, \ldots, 2r+1\}$, $j \in \{1, 2, \ldots, 8\}$, and "<<<3" is the cyclic rotation by 3 bits to the left.

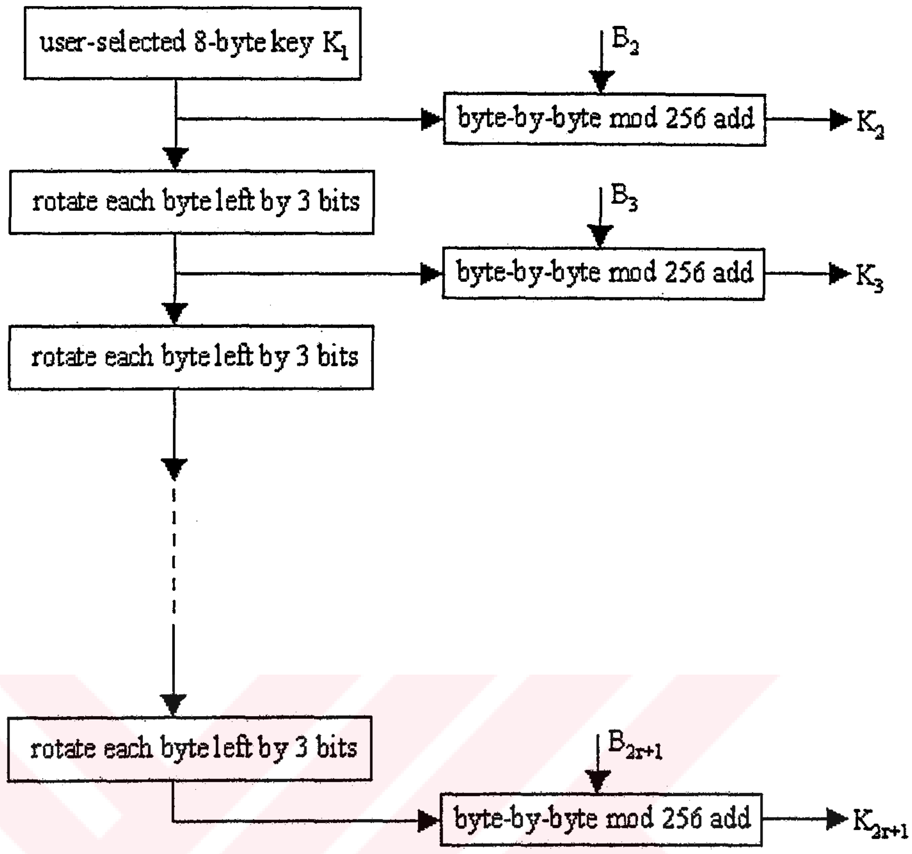

Figure 2.4 Key Schedule Structure of Safer K-64

2.3 Spectr-H64

2.3.1 Introduction

Spectr-H64 [4] is a secret key block cipher, recently designed by N. D. Goots, A. A. Moldovyan and N. A. Moldovyan. It is based on the data-dependent permutations and data-dependent transformation of round keys, aiming to make the cipher strong against differential and linear cryptanalysis.

Spectr-H64 has a 256-bit user-selected secret key, and encrypts and decrypts data in blocks of 64 bits. The cipher is so constructed that the decryption structure is the same as the encryption structure once the decryption round keys have been determined as explained in Section 2.3.3.

The design strategy of Spectr-H64 is oriented to the following objectives [4]:

- Cryptoscheme should be iterative in structure,
- Cryptoscheme should be based on fast operations (data-dependent permutations, XOR, fixed permutations, special fast nonlinear functions),
- Key scheduling should be very simple in order to provide high encryption speed in the case of frequent change of keys.

2.3.2 Algorithm

The algorithm is designed as a sequence of the initial transformation IT, 12 iterative rounds, and the final transformation FT. The overall encryption structure is shown in Fig. 2.5. The round keys Q_{IT}, Q_1, Q_2, …, Q_{12} and Q_{FT} are derived from the user selected 256-bit secret key $K = (K_1, K_2, \ldots, K_8)$ (see section 2.3.3).

The initial and final transformations perform data-dependent permutations, and in the round transformation of Spectr-H64 the following operations are used: cyclic rotation "$>>>k$" by fixed amount k, XOR operation "$\oplus$", nonlinear function G_{AB}, data-dependent permutations $P_{32/80}$, $P^{-1}_{32/80}$ and extension operation E_{AB}, where the subscripts A and B denote 32-bit round keys used by the boxes G and E.

The extension operation E_{AB} shown in Fig. 2.5 is used to form a 80-bit control vector, which is represented as

$$E_{AB}(U) = (V_1, V_2, V_3, V_4, V_5) = V,$$

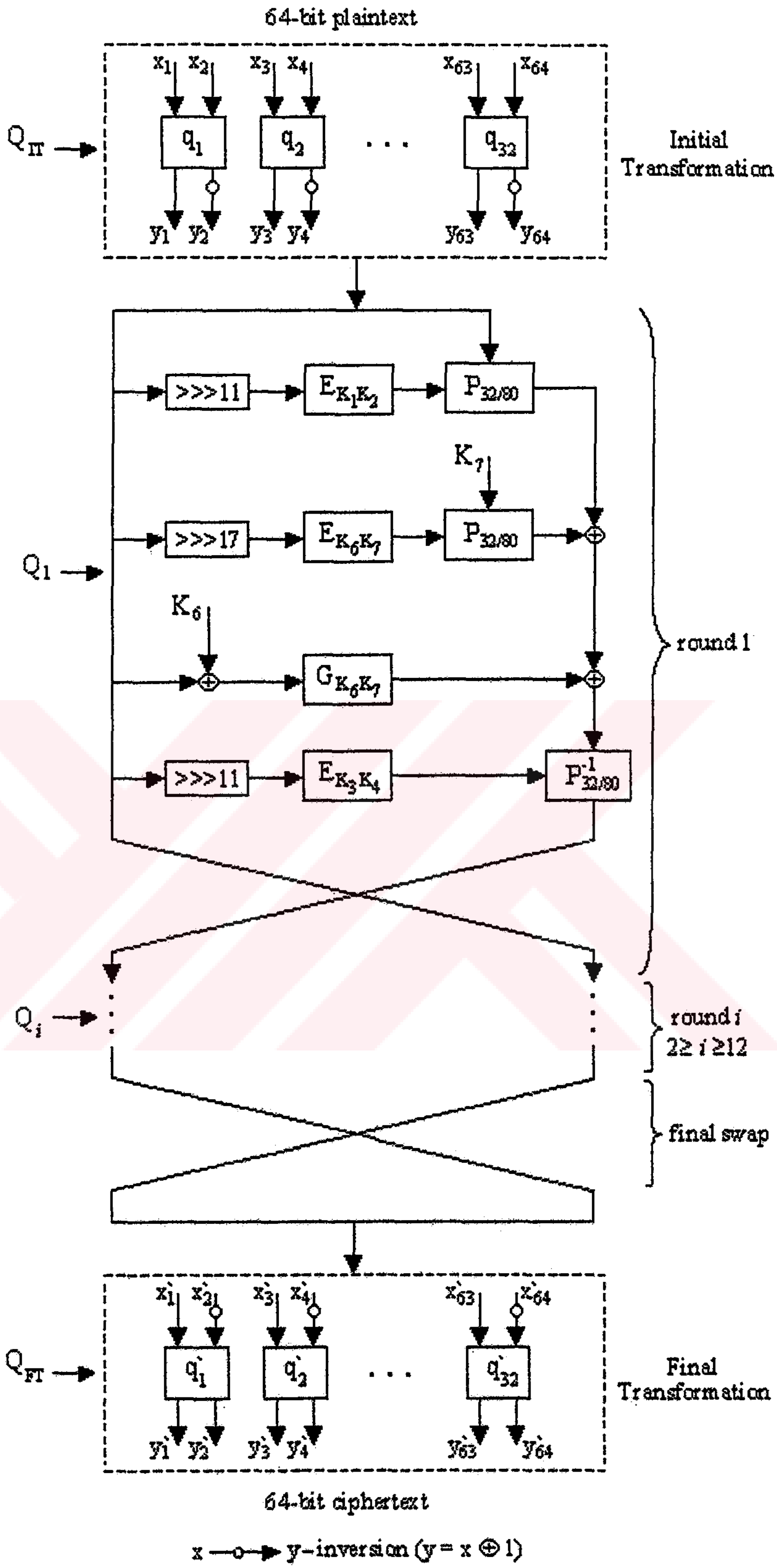

Figure 2.5 Overall Encryption Scheme of Spectr-H64

where $U \in \{0, 1\}^{32}$, $V_1, V_2, \ldots, V_5 \in \{0, 1\}^{16}$ and $V \in \{0, 1\}^{80}$. The vectors V_1, $V_2, \ldots, V_5$ are determined according to the following equations:

$$V_1 = U_R \tag{2.9a}$$

$$V_2 = \pi\,((U \oplus A)_R) \tag{2.9b}$$

$$V_3 = \pi`\,((U \oplus B)_R) \tag{2.9c}$$

$$V_4 = \pi`\,((U \oplus B)_L) \tag{2.9d}$$

$$V_5 = \pi\,((U \oplus A)_L) \tag{2.9e}$$

where the subscripts L and R denote left and right half of the vectors respectively and the fixed permutations π and $\pi`$ are defined for $Z \in \{0, 1\}^{32}$ as

$$\pi\,(Z) = (Z_R{>}{>}{>}1, Z_L{>}{>}{>}1) \tag{2.10}$$

$$\pi`\,(Z) = (Z_R{>}{>}{>}5, Z_L{>}{>}{>}5) \tag{2.11}$$

The data-dependent permutation applied on the input $X \in \{0, 1\}^{32}$ by the $P_{32/80}$ box (Fig. 2.6) produces the output $Y \in \{0, 1\}^{32}$:

$$Y = P_{32/80}(X,V),$$

where $V = (v_1, v_2, \ldots, v_{80})$ is the control vector formed by the extension operation. The $P_{32/80}$ box consists of the 80 $P_{2/1}$ boxes arranged in 5 layers. Each $P_{2/1}$ box has one control bit v_i ($i \in \{1, \ldots, 80\}$), 2-bit input vector and 2-bit output vector. If the control bit $v_i = 0$, then the input vector is directly carried to the output, otherwise the input bits are interchanged. From Fig.2.6 it is seen that the $P_{32/80}$ box applies four permutations after the first 4 layers. It is important to observe that the initial values of the control bits (v_1, v_2, ..., v_{16}) of the first layer are equal to the right half part of 11-bit cyclically rotated form of the left half plaintext (see Fig.2.5 and Eq. (2.9a)).

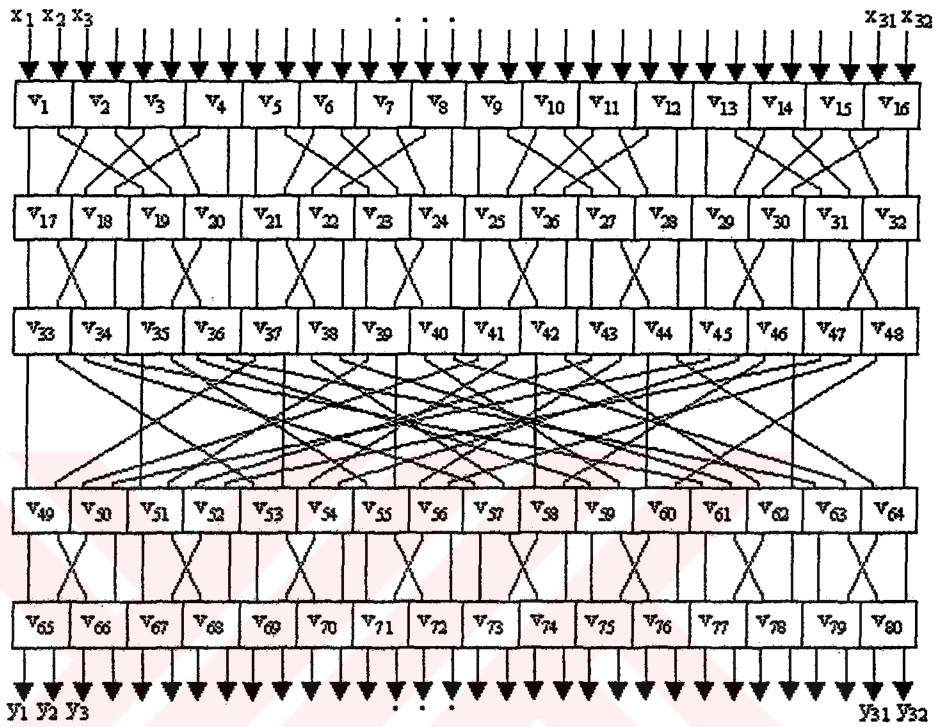

Figure 2.6 Structure of the $P_{32/80}$ Box

The operation performed by $P^{-1}_{32/80}$ box is the inverse of the operation applied by the $P_{32/80}$ box. Therefore, the control bits of the last layer of $P^{-1}_{32/80}$ box are also equal to the right half part of 11-bit cyclically rotated form of the left half plaintext.

The nonlinear operation G_{AB} shown in Fig. 2.5 is represented as

$$G_{AB}\,(W) = X,$$

where X,W $\in$ $\{0, 1\}^{32}$. The nonlinear operation G_{AB} is performed by the following equation:

$$X = M_0 \oplus M_1 \oplus (M_2 \otimes A) \oplus (M_2 \otimes M_5 \otimes B) \oplus (M_3 \otimes M_5) \oplus (M_4 \otimes B) \qquad \textbf{(2.12)}$$

where binary vectors M_1, M_2, ..., M_5 are expressed recursively through W as follows:

$$M_0 = (m_1^{(0)}, m_2^{(0)}, \ldots, m_{32}^{(0)}) = W \qquad \textbf{(2.13a)}$$

$$M_j = (m_1^{(j)}, m_2^{(j)}, \ldots, m_{32}^{(j)}) = (1, m_1^{(j-1)}, m_2^{(j-1)}, \ldots, m_{31}^{(j-1)}) \qquad \textbf{(2.13b)}$$

for all j = 1, 2, ..., 5.

The initial (IT) and final (FT) transformations of the algorithm are represented as:

$$Y = IT(X, Q_{IT})$$

$$Y` = FT(X`, Q_{FT})$$

where X, X` ,Y, Y` $\in$ $\{0,1\}^{64}$, and Q_{IT}, Q_{FT} $\in$ $\{0,1\}^{32}$. As shown in Fig.2.5, the initial transformation uses 32 bits of $Q_{IT} = \{q_1, q_2, \ldots, q_{32}\}$ as the control bits of its 32 $P_{2/1}$ boxes, which interchange the two input bits whenever $q_i = 1$. Each even indexed bit at the $P_{2/1}$ box output is then inverted.

The final transformation is defined as the inverse of the initial transformation, i.e., each even indexed bit of the input block is first inverted and then each pair of bits with indices 2i-1 and 2i is interchanged whenever the control bit $q_i` = 1$, where $\{q_1`, q_2`, \ldots, q_{32}`\} = Q_{FT}$.

The computational graph of the decryption process is essentially same as that of the encryption process, the only change being in the key schedule of the algorithm as explained in the following section.

2.3.3 Key Schedule

The round keys Q_{IT}, Q_1, Q_2, …, Q_{12}, Q_{FT} are derived from the original key $K = (K_1, K_2, \ldots, K_8)$ ($K \in \{0, 1\}^{256}$ and $K_1, K_2, \ldots, K_8 \in \{0, 1\}^{32}$) as shown in each column of Table 2.2. Notice that, in Table 2.2, $Q_{IT}, Q_{FT} \in \{0, 1\}^{32}$, $Q_1, Q_2, \ldots, Q_{12} \in \{0, 1\}^{192}$ and the intermediate key values O_{2j-1} and O_{2j}, for $j \in \{1, 2, 3, 4\}$, are derived as $O_{2j-1} = K_{2j-1+e}$ and $O_{2j} = K_{2j-e}$; where $e = 0$ in the encryption and $e = 1$ in the decryption modes.

Table 2.2 Key Schedule of Spectr-H64

Q_{IT}	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}	Q_{11}	Q_{12}	Q_{FT}
	K_1	K_8	K_5	K_4	K_1	K_6	K_7	K_4	K_2	K_6	K_5	K_3	
	K_2	K_6	K_7	K_3	K_2	K_8	K_5	K_3	K_1	K_8	K_7	K_4	
O_1	O_6	O_1	O_2	O_5	O_7	O_3	O_4	O_8	O_6	O_1	O_2	O_5	O_2
	O_7	O_4	O_3	O_8	O_6	O_1	O_2	O_5	O_7	O_4	O_3	O_8	
	K_3	K_5	K_6	K_2	K_4	K_7	K_6	K_1	K_4	K_5	K_8	K_1	
	K_4	K_7	K_8	K_1	K_3	K_5	K_8	K_2	K_3	K_7	K_6	K_2	

CHAPTER 3

SECURITY TEST CRITERIA FOR BLOCK CIPHERS

In 1949, Shannon published a vital paper for modern cryptography, introducing the fundamental theory of symmetric cryptosystems [5]. In that paper, he presented the principles of "confusion" and "diffusion". Since then, these principles have become the essential properties that block ciphers must have and methods of achieving good diffusion and confusion have lied at the heart of block cipher design.

Confusion obscures the relationship between the elements of plaintext and the elements of ciphertext, while diffusion spreads the influence of the plaintext elements over the ciphertext elements. Both principles try o achieve the same goal: *hiding the statistical features of plaintext* Depending upon these principles, some well-known cryptographic test criteria appearing in the literature such as completeness, avalanche, strict avalanche, bit independence, XOR table distribution, nonlinearity and Avalanche Weight Distribution (AWD) are described in this chapter.

3.1 Completeness and Avalanche Criteria

The property of completeness was defined by Kam and Davida [6]. If a cryptographic transformation is complete, then every input bit depends upon every output bit. If a cryptographic transformation is not complete, it is possible to find a pair of input and output bits such that flipping the input bit does not cause a change in the output bit for all input vectors.

The idea of avalanche was introduced by Feistel [7]. Avalanche is a desirable cryptographic property, which is necessary to ensure that a small difference between two plaintexts results in a seemingly random difference between the two corresponding ciphertexts. When some input value is concerned, changing one bit in that value, half of the output bits is expected to change; if "avalanche criterion" is to be satisfied.

For a cryptographic transformation f, which maps n input bits to m output bits; there are 2^n different inputs. If input vectors $\mathbf{X}$ and $\mathbf{X}_i$ differ by the vector $\mathbf{i}$ ($\mathbf{X}_i = \mathbf{X} \oplus \mathbf{i}$) the output difference vector called the "avalanche vector" is computed as:

$$\boldsymbol{\delta}_f^i = f(\mathbf{X}) \oplus f(\mathbf{X}_i) = [\delta_1^i \; \delta_2^i \; \ldots \; \delta_m^i], \qquad \delta_k^i \in Z_2 \tag{3.1}$$

whose elements are called the avalanche variables, and $i \in [0, 2^n-1]$ is an integer, for which the binary representation is given by the n-bit vector $\mathbf{i}$.

One can find the total change of the k'th avalanche variable over the whole input alphabet of size 2^n, corresponding to the input difference vector $\mathbf{i}$, by considering all input pairs ($\mathbf{X}, \mathbf{X}_i$):

$$\mathrm{w}(\delta_k^i) = \sum_{\text{for all } \mathbf{X} \in Z_2^n} \delta_k^i \tag{3.2}$$

Since there are 2^n terms in the summation given above, Eq. (3.2) should be divided by 2^n to be used as an estimate of probability.

In order to obtain the change of all avalanche variables corresponding to the input difference vector **i**, we sum the normalized form of Eq. (3.2) over all output bits:

$$(½)^n \sum_{k=1}^{m} \mathrm{w}(\delta_k^{\,i}) \qquad \textbf{(3.3)}$$

If Eq. (3.3) is normalized by m it can also be used as a probability estimate

$$p_{AVAL}(i) = (1/(m \cdot 2^n)) \sum_{k=1}^{m} \mathrm{w}(\delta_k^{\,i}) \qquad \textbf{(3.4)}$$

An nxm function is said to satisfy the avalanche criterion exactly, if the probability, $p_{AVAL}(i)$, defined above is equal to one half for all $\mathbf{i}\neq 0$. Although traditionally, the definition of the avalanche criterion is restricted only to single-bit difference vectors [7]; yet, we prefer to use a more general definition considering all possible input difference vectors **i**.

$p_{AVAL}(i)$ can take values in the range [0, 1], and it should be interpreted as the probability of change of all output bits when the input difference vector equals **i**. If $p_{AVAL}(i)$ is close to ½ for all i, then the s-box satisfies the avalanche criterion within a small error region.

3.2 Strict Avalanche Criterion

Webster and Tavares combined the criteria of completeness and avalanche into Strict Avalanche Criterion (SAC) [8]. According to their definition, a function

$f: Z_2^n \rightarrow Z_2^m$ satisfies SAC if for all $k \in \{1, 2, ..., n\}$, $l \in \{1, 2, ..., m\}$, flipping the k'th input bit changes the l'th output bit with the probability of exactly ½. However, in this work, we prefer to use a more general definition, and consider any difference vector **i**≠0 (or corresponding integer $i \in [1, 2^n-1]$) at the input, and any possible linear combination of the output bits masked by the vector **j** (expressed either as the integer $j \in [1, 2^m-1]$ or corresponding m-bit binary vector **j**). So, an s-box is said to satisfy SAC if for all $i \in [1, 2^n-1]$, $j \in [1, 2^m-1]$:

$$(\tfrac{1}{2})^n \, w(\delta_j^i) = \tfrac{1}{2} \qquad \textbf{(3.5)}$$

where δ_j^i should now be interpreted as a generalized avalanche variable, which is equal to some linear combination of the avalanche variables, or the "avalanche vector masked by **j**, ($\boldsymbol{\delta}_f^i \cdot \mathbf{j}$)". Eq. (3.5) can be used to define a SAC parameter:

$$p_{SAC}(i, j) = (\tfrac{1}{2})^n \, w(\delta_j^i) \qquad \textbf{(3.6)}$$

$p_{SAC}(i, j)$ can take values in the range [0, 1], and it should be interpreted as the probability of change of the output bits masked by the vector **j** when the input is changed by the difference vector **i**. If $p_{SAC}(i, j)$ is close to ½ for all (i, j) pairs, then the s-box is said to satisfy SAC within a small error region. It is easy to demonstrate that an s-box, which satisfies SAC, must satisfy both completeness and avalanche criteria, but the satisfaction of avalanche criterion does not necessarily imply the satisfaction of SAC.

3.3 Bit Independence Criterion

The idea of bit independence criterion (BIC) was introduced by Webster and Tavares [8]. For a given set of avalanche vectors generated by complementing a single plaintext bit, all avalanche variables should be pairwise independent. Alternatively, consider two n-bit input vectors which differ by **i**, with the corresponding avalanche vector $\boldsymbol{\delta}_f^i$. If f meets the bit independence criterion, then the k'th and the l'th bits of $\boldsymbol{\delta}_f^i$ change independently for all k, l ($1 \leq k$, $l \leq m$ with $k \neq l$), and input difference vectors $i \in [1,\ 2^n-1]$). (We should mention that the traditional definition of BIC is only interested in single-bit input differences; however, we prefer the general definition, which takes all input differences into account.)

To measure the bit independence property, one needs the correlation coefficient between the k'th and l'th components of the avalanche vector $\boldsymbol{\delta}_f^i$, evaluated over all input pairs **X** and $\mathbf{X}_i$, which differ by i ($\mathbf{X}_i = \mathbf{X} \oplus \mathbf{i}$). The bit independence parameter corresponding to the effect of the input change **i** on the k'th and l'th bits of $\boldsymbol{\delta}_f^i$ is the absolute value of this correlation coefficient:

$$\begin{aligned} BIC^i(\delta_k, \delta_l) &= |(\mathrm{E}\{\delta_k^i \delta_l^i\} - \mathrm{E}\{\delta_k^i\}\mathrm{E}\{\delta_l^i\}) / \sigma_k \sigma_l| \\ &= |\, cov(\delta_k^i, \delta_l^i) / \sigma_k \sigma_l \,| \end{aligned} \tag{3.7}$$

where E{.} denotes the average value and "σ" stands for the standard deviation of the related avalanche variable. Then the overall BIC parameter is defined as:

$$BIC(\boldsymbol{\delta}_f) = \max_{\substack{1 \leq i \leq 2^n-1 \\ 1 \leq k,\, l \leq m \\ j \neq k}} BIC^i(\delta_k, \delta_l) \tag{3.8}$$

$BIC(\boldsymbol{\delta}_f)$ is defined in the range [0, 1]. It is ideally equal to zero, and in the worst case it is equal to one.

3.4 XOR Table Distribution

XOR table of an s-box gives information about the security of the block cipher against differential cryptanalysis. The essence of a differential attack is that it exploits particular high-valued entries in the XOR tables of s-boxes employed by a block cipher.

The XOR table [9] of an $n \times m$ s-box is a $2^n \times 2^m$ matrix. The rows of the matrix represent the change in the output of the s-box. An entry in the XOR table of an s-box indexed by ($\mathbf{i}$, δ) indicates the number of input vectors $\mathbf{X}$ which, when changed by $\mathbf{i}$, result in the specific output difference of $\delta = f(\mathbf{X}) \oplus f(\mathbf{X} \oplus \mathbf{i})$:

$$XOR_f(\mathbf{i}, \delta) = \# \{ \mathbf{X} \mid f(\mathbf{X}) \oplus f(\mathbf{X} \oplus \mathbf{i}) = \delta \} \qquad \textbf{(3.9)}$$

where $\mathbf{i} \in Z_2^n$ and $\delta \in Z_2^m$.

Note that an entry in the XOR table can only take an even value, and the sum of all values in a row is always 2^n.

As entries with high values in the XOR table are particularly useful to differential cryptanalysis, a necessary condition for an s-box to be immune to differential cryptanalysis is that, it does not have large values in its XOR table.

3.5 Nonlinearity

The nonlinearity parameter $NLM_f(z)$, of a cipher f: $Z_2^n \rightarrow Z_2^m$ for a given nonzero vector $\mathbf{z} = (\mathbf{j}, \mathbf{w}, c) \in Z_2^{n+m+1}$ (and corresponding integer $z \in [1, 2^{n+m+1}-1]$) is defined as the number of cases over all cipher inputs $\mathbf{X} \in Z_2^n$ such that the affine function ($\mathbf{w} \cdot \mathbf{X} \oplus c$) and the nonzero linear combination ($\mathbf{j} \cdot f(\mathbf{X})$) differ from each other [10]:

$$NLM_f(z) = \# \{ \mathbf{X} \mid (\mathbf{j} \cdot f(\mathbf{X})) \neq \mathbf{w} \cdot \mathbf{X} \oplus c \} \tag{3.10}$$

In Eq. (3.10) $\mathbf{j} \in Z_2^m$, $\mathbf{w} \in Z_2^n$ and $c \in Z_2$. The overall nonlinearity measure, NLM_f, of the cipher f is defined as:

$$NLM_f = \min_z NLM_f(z) \tag{3.11}$$

For a cipher f not to be susceptible to linear cryptanalysis, NLM_f is required to be as close as possible to its maximum value, given by $(2^{n-1} - 2^{-1}2^{n/2})$ for perfectly nonlinear functions. Small values of NLM_f around zero show that the cipher f is close to affine functions, and susceptible to linear cryptanalysis [10].

3.6 Avalanche Weight Distribution (AWD)

Most of the above mentioned criteria are used mainly for testing the s-boxes of block ciphers; and they are not very practical for application to the overall cipher. We propose the Avalanche Weight Distribution (AWD) criterion [11] for fast and rough analysis of overall diffusion properties of block ciphers. This criterion examines whether for quite similar plaintext pairs, $\mathbf{X}$ and $\mathbf{X}_i$, which differ by i, the Hamming weight of the corresponding avalanche vector $\boldsymbol{\delta}_f^i$, is sufficiently close to the binomial distribution around $m/2$, m being the dimension of output vectors of the cipher.

CHAPTER 4

TEST RESULTS OF RIJNDAEL AND SAFER K-64

In this thesis, in order to implement the block ciphers and simulate the security tests, the programming languages C and Matlab are used. All security tests related to s-boxes can be efficiently simulated in Matlab. However, for the AWD criterion that is applied to the overall algorithm, the programming language C is preferred since it takes shorter time (see Appendix E).

4.1 Avalanche Criterion

Although Eq. (3.4) is not satisfied exactly by the s-box of Rijndael, $p_{AVAL}(i)$ values are found to be very close to one half. Indeed, it is more logical to expect the criterion given by Eq. (3.4) to be satisfied within an error range of $\pm\in$, which we call the "relative error interval"; so that, the modified avalanche criterion within $\pm\in_i$ is [12, 13]:

$$(\tfrac{1}{2})(1-\in_i) \leq p_{AVAL}(i) \leq (\tfrac{1}{2})(1+\in_i) \qquad \textbf{(4.1)}$$

Corresponding overall relative absolute error $\in_{AVAL}$ of the s-box can be found from Eq. (4.1) as:

$$\in_{AVAL} = \max_{1 \le i \le 2^n - 1} | 1 - 2\, p_{AVAL}(i) | \qquad \textbf{(4.2)}$$

Relative absolute error for the avalanche criterion is obtained as "0.0625", evaluating Eq. (3.4) and Eq. (4.2) successively, for the s-box of Rijndael.

4.2 Strict Avalanche Criterion

The s-box of Rijndael does not satisfy SAC exactly; but it satisfies SAC within a very small relative error interval, as in the case of avalanche criterion. So, considering Eq. (3.6) within an error interval as in Eq. (4.1), overall relative absolute error for SAC, $\in_{SAC}$, can be found as [12, 13]:

$$\in_{SAC} = \max_{\substack{1 \le i \le 2^n - 1 \\ 1 \le j \le 2^m - 1}} | 1 - 2\, p_{SAC}(i,j) | \qquad \textbf{(4.3)}$$

It can be shown that $[1\text{-}2\, p_{SAC}(i,j)]$ is equal to the autocorrelation function $r_j(i)$ of the Boolean function, corresponding to the output bits masked by j [23]. The authors of Rijndael have used the idea in [21], which states that for invertible s-boxes operating on bytes, the maximum input/output correlation can be made as low as 0.1250 [1]. Computing Eq. (3.6) and Eq. (4.3) successively, we obtain $\in_{SAC}$ as "0.1250" for Rijndael, and confirm that the lowest limit is achieved.

4.3 Bit Independence Criterion

The situation for BIC is a little bit different from Avalanche Criterion and SAC, as BIC is analysed according to the $BIC(\boldsymbol{\delta}_f)$ value of an s-box, which is already defined by Eq. (3.8) as the highest value of correlation coefficient corresponding to any two avalanche variables; hence the relative absolute error, $\in_{BIC}$, for this criterion is:

$$\in_{BIC} = BIC(\delta_f) \qquad \textbf{(4.4)}$$

$\in_{BIC}$ is obtained, calculating Eq (3.7) and Eq. (3.8) as "0.1350" for Rijndael's s-box. In Table 4.1, we firstly summarize the parameters obtained for the s-boxes of Rijndael and Safer K-64 [11, 14], using the traditional definitions.

Table 4.1 Relative Absolute Error Values of Rijndael and Safer K-64, for Weight-One Input Differences and Separate Output Bits

Cipher & S-box	$\in_{AVAL}$	$\in_{SAC}$	$\in_{BIC}$
S-box of Rijndael	0.0352	0.1250	0.1341
Exponentiating s-box of Safer K-64	0.5039	1	1
Logarithm taking s-box of Safer K-64	0.0313	0.2500	0.3150

Secondly, using the advanced definitions of the mentioned criteria, and considering all input differences $i \in [0, 2^n-1]$, and all nonzero linear combinations at the output, with output masks $j \in [0, 2^m-1]$; we obtain the results in Table 4.2.

Table 4.2 Relative Absolute Error Values of Rijndael and Safer K-64, for All Input Differences and All Linear Combinations of Output Bits

Cipher & S-box	$\in_{AVAL}$	$\in_{SAC}$	$\in_{BIC}$
S-box of Rijndael	0.0625	0.1250	0.1350
Exponentiating s-box of Safer K-64	0.5039	1	1
Logarithm taking s-box of Safer K-64	0.3906	0.6875	0.5318

4.4 XOR Table Distribution

The XOR table is a matrix of size 256 × 256, whose entries are calculated by Eq. (3.9). We divide it into 8 pieces, so that each piece is 32 × 256, and tabulate the maximum entry for each piece together with the corresponding maximum entry of the exponentiation and the logarithm taking s-boxes of Safer K-64 in Table 4.3 for comparison. In [21], it is claimed that the maximum entry in the XOR table for an s-box operating on bytes can be made as low as 4, as we found experimentally for Rijndael's s-box. It is apparent that XOR values of Rijndael are much more uniformly distributed than those of Safer K-64; which makes it securer against differential cryptanalysis.

Table 4.3 Maximum Entries of the XOR Table for Rijndael and Safer K-64

	Logarithm taking s-box of Safer K-64	*Exponentiating s-box of Safer K-64*	*S-box of Rijndael*
1st piece	12	12	4
2nd piece	22	16	4
3rd piece	16	22	4
4th piece	22	12	4
5th piece	12	128	4
6th piece	16	16	4
7th piece	16	22	4
8th piece	128	16	4

4.5 Nonlinearity

The nonlinearity distribution is defined as the number of $\mathbf{z} = (\mathbf{j}, \mathbf{w}, c) \in Z_2^{n+m+1}$ vectors corresponding to a specific value of $NLM_f(z)$, given by Eq.(3.10) as $NLM_f(z) = \# \{ \mathbf{X} \mid (\mathbf{j} \cdot f(\mathbf{X})) \neq \mathbf{w} \cdot \mathbf{X} \oplus c \}$. Hence the horizontal axis shows all possible $NLM_f(z)$ values in the range $(0, 2^n)$. The minimum value on the horizontal axis corresponds to NLM_f defined by Eq. (3.11). The nonlinearity distribution of an s-box, which is symmetric around the midpoint $NLM_f(z) = 2^n$-1, gives us information about the susceptibility of the s-box to linear cryptanalysis. In Fig.4.1, the nonlinearity distribution for the s-box of Rijndael is sketched.

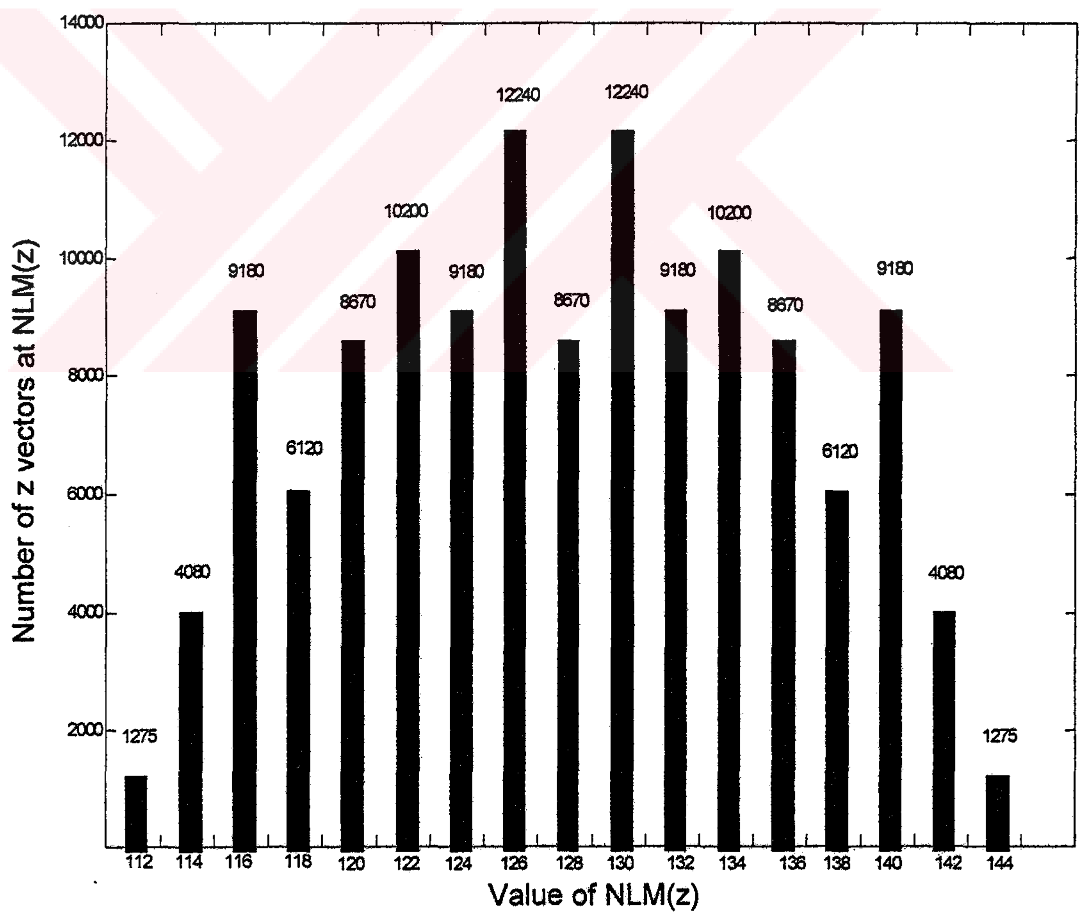

Figure 4.1 Nonlinearity Distribution for the S-box of Rijndael

Examining Fig. 4.1, the s-box of Rijndael does not seem to be susceptible to linear cryptanalysis. Because, it is observed that the s-box of Rijndael has got 1275 **z** vectors with the lowest $NLM_f(z) = 112$. For those 1275 **z** vectors, equation

$$\mathbf{j}\cdot f(\mathbf{X}) \neq \mathbf{w}\cdot\mathbf{X} \oplus c \tag{4.5}$$

is satisfied for 112 plaintexts. Hence the probability that Eq. (4.5) is correct is equal to 112/256 = 0.4375. Since this probability is very close to ½ (bias is 0.5-0.4375=0.0625), the s-box of Rijndael can be said to be "not susceptible to linear cryptanalysis".

Nonlinearity distribution curves of Safer's exponentiating and logarithm taking s-boxes both yield the same curve (Fig. 4.2), which results from the fact that the logarithm taking s-box is obtained by inverting the exponentiating s-box and vice versa.

In Fig. 4.2(a), $NLM_f(z)$ parameter taking smaller values than 98 is not seen due to the corresponding small number of z vectors. Therefore, we sketch details of Fig. 4.2(a) in Fig. 4.2(b), by expanding the vertical scale, so that the lowest $NLM_f(z)$ value is seen. From Fig. 4.2(b), it is observed that Safer's s-boxes have got only one z vector with the lowest $NLM_f(z) = 82$. Hence, for that specific z vector, the probability that Eq.(4.5) is correct equals to 82/256 = 0.3203, which has a remarkably larger bias (0.5-0.3203=0.1797) than that of Rijndael's s-box; hence one can claim that Safer is more susceptible against linear cryptanalysis.

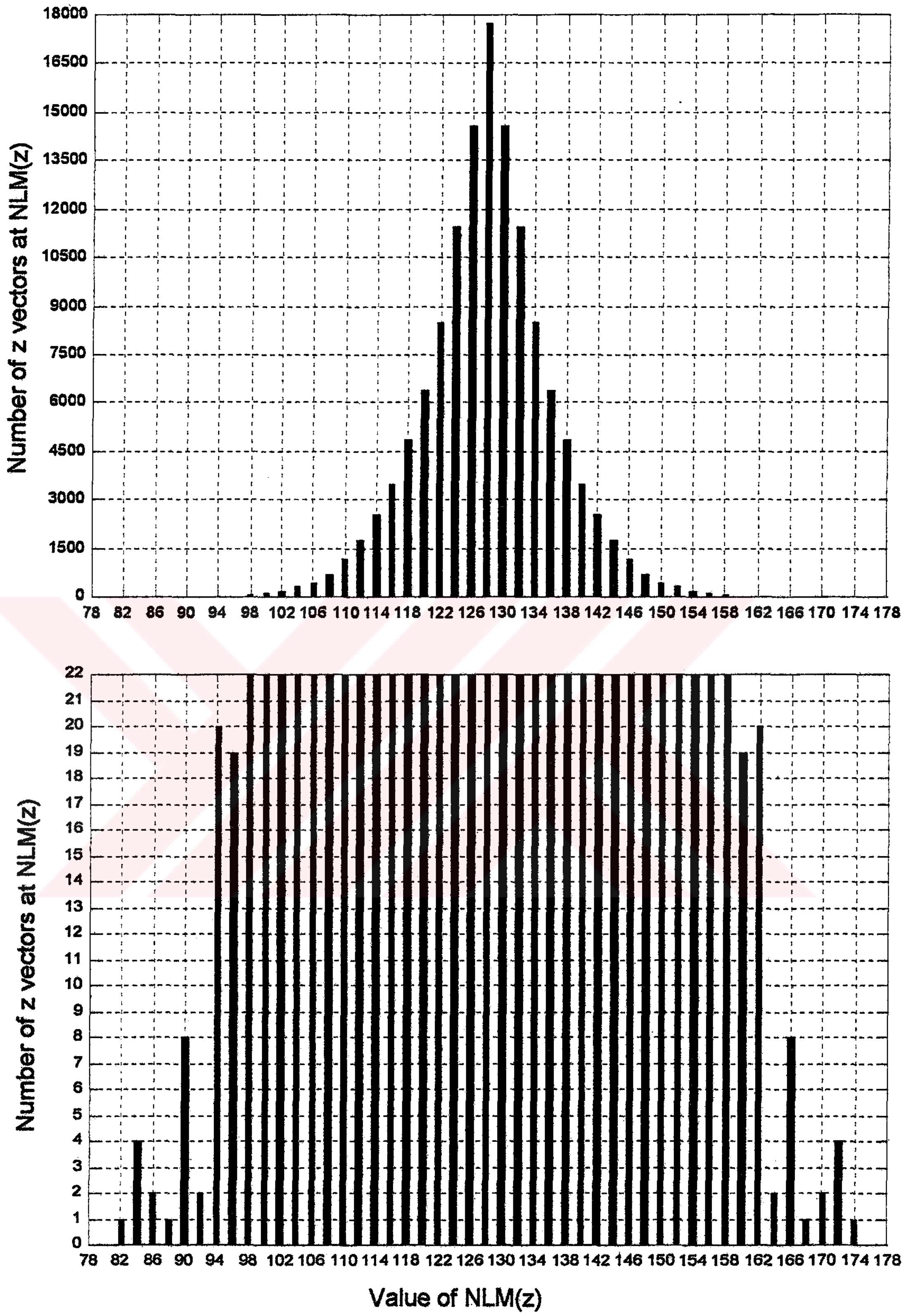

Figure 4.2 Nonlinearity Distribution Curves for the S-boxes of Safer (a) Overall Picture (b) Details

In fact, the mentioned z vector with the lowest $NLM_f(z)$ value equals to $(\mathbf{j},\mathbf{w},c)$ = ((0, 0, 1, 1, 0, 0, 1, 0), (0, 0, 1, 1, 1, 0, 1, 0),1) for the exponentiating s-box of Safer; while, it equals to $(\mathbf{j},\mathbf{w},c)$ = ((0, 0, 1, 1, 1, 0, 1, 0), (0, 0, 1, 1, 0, 0, 1, 0),1) for the logarithm taking s-box. Notice the interchange of the vectors **j** and **w**, which results from the fact that both the exponentiating s-box and the logarithm taking s-box are the inverse transforms of each other. The overall nonlinearity measure, NLM_f, can also be obtained using the Walsh-Hadamard Transform. The Walsh-Hadamard Transforms for Rijndael's s-box, Safer's exponentiating s-box, and Safer's logarithm taking s-box are given in Appendix D.

4.6 Avalanche Weight Distribution (AWD)

We use the following test procedure [11, 14] to examine the diffusion properties of Rijndael with n=128 bit input block and key lengths for the criterion of Avalanche Weight Distribution (AWD);

Step 1- A plaintext $\mathbf{X}$ is chosen at random and the plaintext $\mathbf{X}_{e_i}$ is calculated so that the difference between $\mathbf{X}$ and $\mathbf{X}_{e_i}$ is e_i, i.e., $\mathbf{X}_{e_i} = \mathbf{X} \oplus e_i$ (where e_i is the n-bit unit vector in position i); hence, $\mathbf{X}$ and $\mathbf{X}_{e_i}$ differ only in bit $i \in \{1, 2, \ldots, 128\}$.

Step 2- $\mathbf{X}$ and $\mathbf{X}_{e_i}$ are submitted to r-rounds of Rijndael for encryption under a random key,

Step 3- From the resultant ciphertexts $\mathbf{C} = f(\mathbf{X})$ and $\mathbf{C}_{e_i} = f(\mathbf{X}_{e_i})$, the Hamming weight of the avalanche vector $wt(\boldsymbol{\delta}^{e_i}) = k$ is calculated, where $k \in \{1, \ldots, 128\}$,

Step 4- The value of the k^{th} element of the avalanche weight distribution array is incremented by one, i.e., $AWD\_array[k] = AWD\_array[k] + 1$,

Step 5- The steps 2-4 are repeated for at least 10000 randomly chosen input vectors $\mathbf{X}$, and the values in the avalanche weight distribution array are sketched versus its index k, as the histogram of possible weights corresponding to e_i.

With the help of the test procedure explained above, 128 figures, each corresponding to an input difference e_i are obtained for each round of Rijndael. It is observed that none of the curves with 1-round of encryption satisfies the AWD criterion, i.e., they do not resemble a binomial distribution around a mean value of 64. The reason why they are unacceptable is that; after 1-round of encryption all the avalanche vectors have very small Hamming weights, gathered around a mean value of ~16 and none of which exceeding 30. However, after 2-rounds of encryption, all AWD curves start to resemble the expected binomial distribution around 64. First and second round AWD curves corresponding to an input difference of e_1 are depicted in Fig. 4.3; as representatives of other one-bit input differences, which all yield very similar curves.

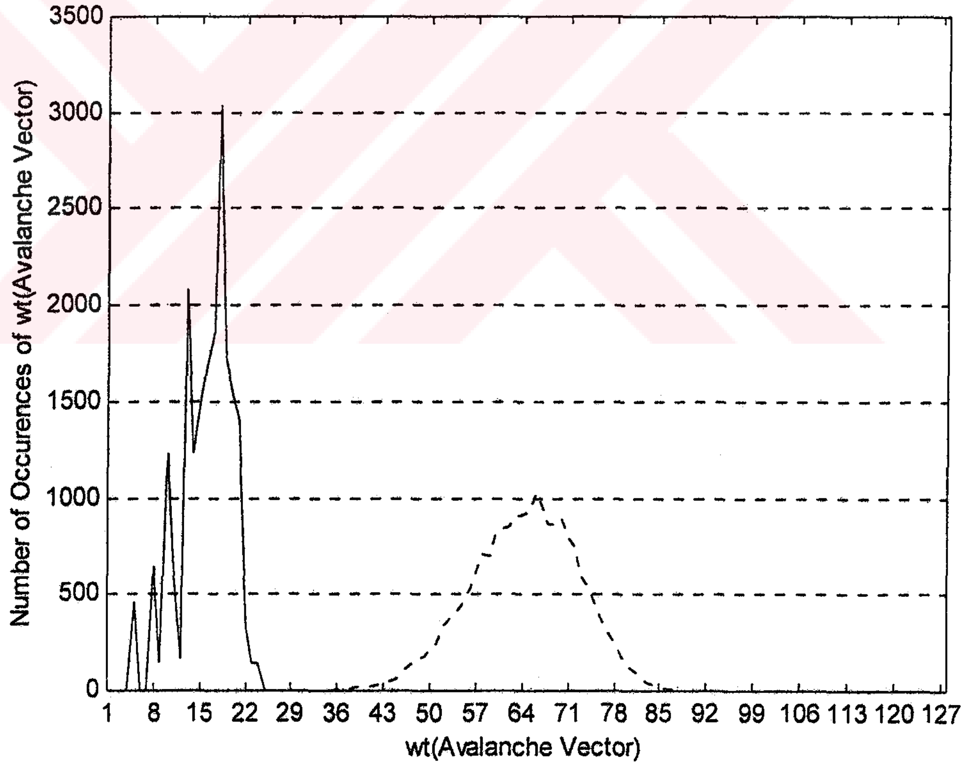

Figure 4.3 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Rijndael for e_1

For comparison, a representative of AWD curves of Safer K-64 is depicted in Fig.4.4 for the first and second rounds of the cipher. This figure as compared to Fig.4.3, clearly shows that Safer K-64 has much better AWD characteristics, which resembles the ideal binomial distribution, even at the end of the first round.

We can define a normalized measure of closeness between the evaluated AWD curves and the ideal binomial distribution. subtracting the sum of normalized error magnitudes from unity, as:

$$R_r^{e_i} = 1 - \frac{1}{2N}\sum_{k=0}^{m}\left| AWD\_array[k] - \frac{N}{2^m}\binom{m}{k}\right| \qquad \textbf{(4.6)}$$

where m is the length of the ciphertext and N is the number of plaintexts, e_i is the

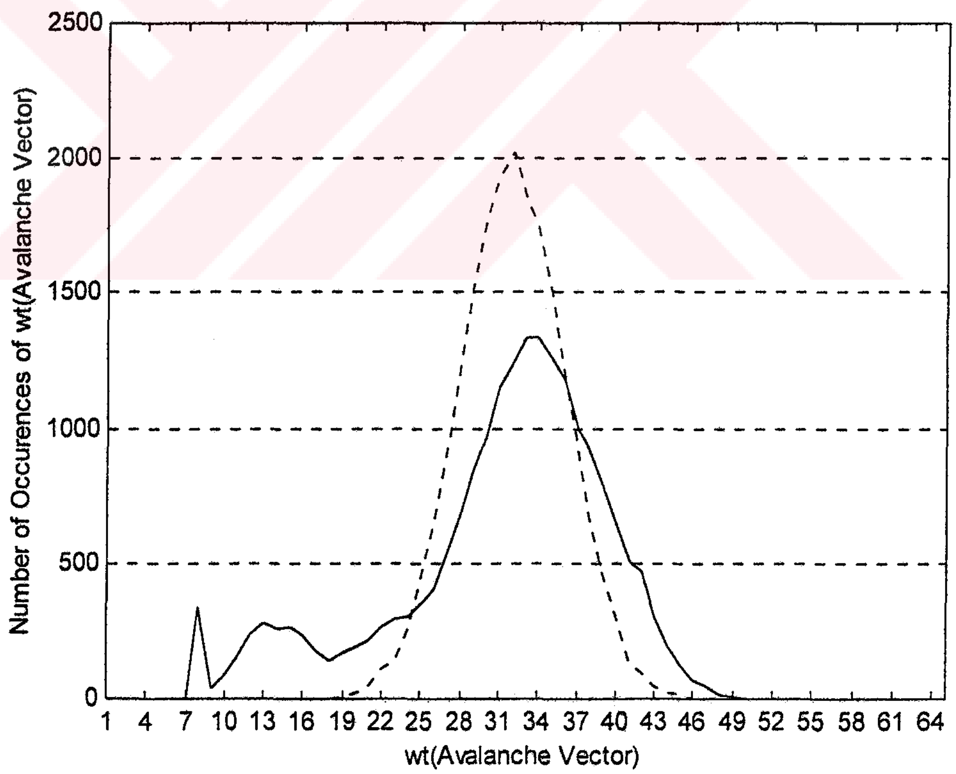

Figure 4.4 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Safer K-64 for e_1

input difference, and r is the number of rounds. Then, the overall first round and second round AWD curves of Rijndael and Safer K-64 yield a closeness-to-ideal (or resemblance) parameter as given in Table 4.4. It is observed from this table that, although the third and higher rounds of Rijndael resemble the ideal binomial curve (of mean value $m/2$=64) more than 98%, the resemblance parameter is around 81% at the second iteration and 0% at the first. On the other hand, similarly evaluated curves of Safer K-64 are found to be much closer [11] to the ideal binomial distribution (of mean value $m/2$=32) than Rijndael values given in Table 4.4. The AWD curves of Rijndael and Safer K-64 are given in Appendix A and Appendix B respectively, together with corresponding resemblance percentages.

Table 4.4 Resemblance Percentages $R_r^{e_1}$ for Rijndael and Safer K-64

	Input difference vector	
Round r	*Rijndael*	*Safer K-64*
1	0	0.7245
2	0.8122	0.9899
3	0.9847	0.9872
4	0.9832	0.9890
5	0.9857	0.9850
6	0.9851	0.9892
7	0.9843	
8	0.9863	
9	0.9866	
10	0.9860	

CHAPTER 5

SLIDE ATTACK ON SPECTR-H64

Spectr-H64 is a secret-key block cipher, recently designed by N. D. Goots, A. A. Moldovyan and N. A. Moldovyan. It is based on the data-dependent permutations and data-dependent transformation of subkeys, with 64-bit block length and 256-bit key length. In this chapter, we observe that some single bit differences at the first round input of Spectr-H64 are reflected to the round output as single bit differences. More specifically, after one round encryption of the two plaintexts P and P_i, which differ in the i^{th} bit of the right half part, only one bit of the resulting output vectors changes as can be seen in Appendix C. We utilize this weakness to extract 128 key bits after one round of the algorithm, using one known plaintext and 32 adaptively chosen plaintexts.

We then apply the slide attack [15, 16] for the 2^{32} weak keys of the form $K = (K_1, K_1, K_1, K_1, K_1, K_1, K_1, K_1)$, which cause each round to be identical. This attack requires about 2^{17} chosen plaintexts and 2^{64} steps (a step being equivalent to the comparison of two 32-bit blocks).

We also find that Spectr-H64 has a set of 2^{128} weak keys of the form $K = (K_1, K_1, K_2, K_2, K_3, K_3, K_4, K_4)$, which includes the above mentioned 2^{32} weak keys. For this set of keys, encryption is the same function as decryption; thus, double encryption reveals the plaintext. For each of these weak keys, we observe that there are 2^{32} fixed points, i.e., the ciphertext becomes the same as the plaintext after encryption.

5.1 Slide Attacks

The first step in "sliding" direction can be dated back to a 1978 paper by Grossman and Tuckerman [17]. Afterwards, Biham's work on related-key cryptanalysis [18], and Knudsen's early work [19] are the milestones in this direction.

Slide attacks are, in general, independent of the exact properties of the iterated round function and the number of rounds, which is not the case for the conventional cryptanalytic tools such as differential and linear cryptanalysis for which each additional round requires an exponential effort from the attacker [15, 16]. A typical slide attack exploits the *self similarity* of a block cipher and views the cipher as a product of identical transformations F_Q (P), where Q is the round key (here F might include more than one round of the cipher). The only requirement on F is that it can be broken easily once an input-output pair is obtained for the identical transformation. Calling the identical transformation $F_Q = F$, and the overall encryption function $E = F \circ F \circ \ldots \circ F = F^r$, the crucial observation leading to the slide attack [15, 16] is

$$P' = F(P) \text{ and } C = F^r(P) \text{ implies } C' = F^r(P') = F^r(F(P)) = F(F^r(P)) = F(C).$$

Hence, a standard slide attack tries to find plaintext-ciphertext pairs (P, C) and (P', C') with $C' = F(C)$. Such pairs are called *slid pairs*, and once a slid pair is found, an extra relation $P' = F(P)$ is obtained.

5.2 Breaking One Round of Spectr-H64

We compare one round diffusion characteristics of some ciphers using the Avalanche Weight Distribution (AWD) curves. In Fig. 5.1, we sketch one round AWD curves corresponding to a single bit plaintext difference for the block ciphers AES-Rijndael, Safer K-64, and Spectr-H64. We observe that perfect diffusion cannot be obtained in a single round; however, when we complement a single input bit, none of these ciphers except Spectr-H64 always causes a single output bit change. More specifically, calling one round encryption of Spectr-H64, $F(\mathrm{P})$, where P is 64-bit input vector; the weakness we observe is that, for a single bit input difference vector e_i, $F(\mathrm{P} \oplus e_i) = F(\mathrm{P}) \oplus e_j$, for all $i \in \{33, 34, \ldots, 64\}$, and $j \in \{1, 2, \ldots, 32\}$.

The reason why one bit difference vector propagates without any weight change in the first round is that, for the input vectors P and P_i with identical left half parts; all control bits of the data-dependent permutations $\mathrm{P}_{32/80}$ and $\mathrm{P}^{-1}_{32/80}$ and the output of the nonlinear function G_{AB} remain the same under the same key. Hence, the difference e_i between the right half parts of P and P_i is only permuted and reflected to the first round output as a single bit change at position j. Due to the same reasons, multiple-bit input differences in the right half part also propagate without any weight change, and the number of complemented bits remain the same at the first round output. The initial transformation does not improve this behaviour, since its bit permutations between adjacent pairs cannot mix separate halves of the plaintext with each other. Hence, as far as the propagation of the right half input difference is concerned, the first round of the algorithm can be modelled as a cascade connection of $\mathrm{P}_{32/80}$ and $\mathrm{P}^{-1}_{32/80}$ boxes, as shown in Fig. 5.2. The first 5 layers in this figure belong to the permutation $\mathrm{P}_{32/80}$ and the last 5 layers belong to the inverse permutation $\mathrm{P}^{-1}_{32/80}$. The XOR boxes between $\mathrm{P}_{32/80}$ and $\mathrm{P}^{-1}_{32/80}$ boxes are not included in Fig. 5.2 since they do not have any effect on difference vectors.

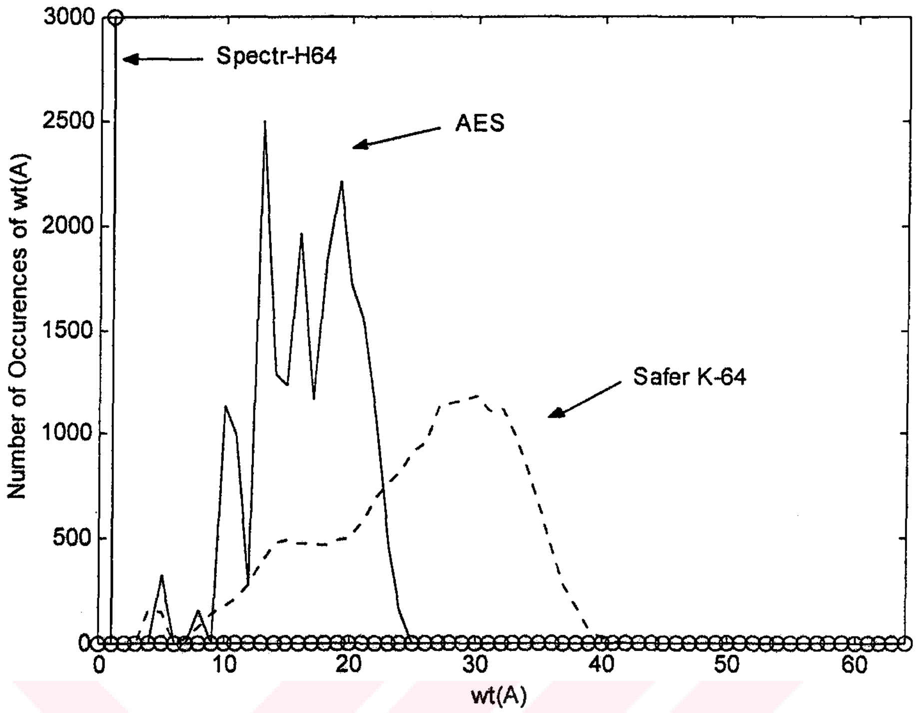

Figure 5.1 One Round AWD Curves of Spectr-H64, Safer K-64 and Rijndael

Another weakness of the round transformation, exploited to extract the key bits after the first round, is the fact that the control vector V_l used in Layer 1 of $P_{32/80}$ and Layer 10 of $P^{-1}_{32/80}$ boxes is completely known. Initially, V_l is equal to the right half part of 11-bit cyclically rotated form of the left half plaintext (see (2.9a) and Fig.2.5).

In the following, we use the mentioned two weakness of Spectr-H64 to describe how the control bits of one permutation layer are found using all control bits of the previous permutation layer and propagation of one bit input differences e_i for $i \in \{33, 34, \ldots, 64\}$. Knowing the output difference bits (e_j) caused by the input difference bits (e_i), the control vector bits are found completely, which are then used to obtain 32×4 key bits K_1, K_2, K_3 and K_4 by applying the inverse of the extension transformation.

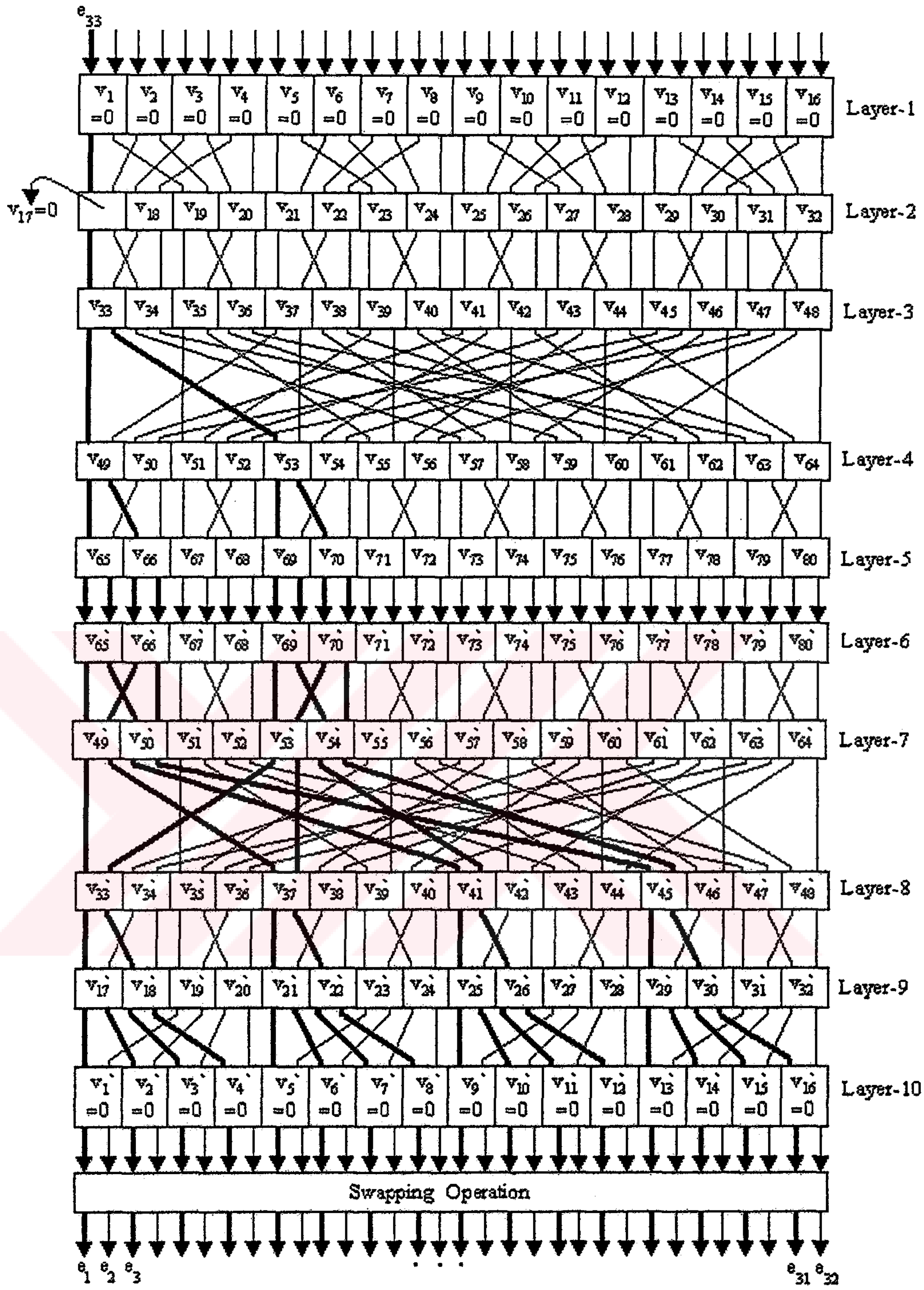

Figure 5.2 Combination of the Transformations $P_{32/80}$ and $P^{-1}_{32/80}$ (illustrating the propagation of one bit input difference (e_{33}) after one round encryption of Spectr-H64 without the initial transformation)

In Fig. 5.2, we sketch the first round propagation of one bit input difference vector e_{33} (corresponding to the state of all zero control bits at the first and last layers) without considering the initial transformation. Notice that only the right halves of propagating differences are shown in the figure since the left half differences are all zero vectors. All control bits of Layer 1 and Layer 10 are known since the plaintext is known (we assume $v_i = v_i' = 0$ for i = 1, 2, ..., 16). The first control bit v_{17} of Layer 2 is found using the propagation of input difference e_{33}. If one of the output bits indicated by bold lines in Fig. 5.2 is complemented in response to the input difference e_{33}, then $v_{17} = 0$; otherwise $v_{17} = 1$. As a second step, the value of v_{18} is found similarly, considering the propagation of either e_{37} or e_{39}, which are controlled by v_{18}. Knowing the positions of the output bits in response to each single input bit difference, one can then obtain all control bits of Layer 2. Notice that carefully selected 16 single bit input differences and their corresponding output bits are enough to extract 16 control bits of Layer 2.

Now since we know all the control bits of Layer 2, we can also obtain the input vectors of Layer 3 and corresponding output vectors of Layer 10; therefore the control bits of Layer 3 are found similarly. In this way all control bits are found and then the key bits K_1, K_2, K_3 and K_4 are obtained using a known plaintext-ciphertext pair (P, C) and 32 adaptively chosen plaintext-ciphertext pairs (P_i, C_j) of one round encryption for each value of i ($P_i = P \oplus e_i$, $C_j = C \oplus e_j$, $i \in \{33, 34, \ldots, 64\}$, $j \in \{1, 2, \ldots, 32\}$).

If the initial transformation is included, we observe a similar distribution of output difference bits, as for one round of Spectr-H64 without the initial transformation, in response to a single input bit difference. Therefore, the propagation of one bit input difference can also be exploited similarly to extract the key bits K_1, K_2, K_3 and K_4 after one round of Spectr-H64 with the initial transformation.

5.3 Applying Slide Attack on Spectr-H64

In this section, firstly the slide attack is applied to a modified variant of Spectr-H64, without the initial and final transformations, for the 2^{32} weak keys, and then it is shown how the initial and final transformations affect the number of operations necessary to break the algorithm for the same keys.

In order to find the degree of self similarity in Spectr-H64, we first examine the key schedule given in Table 2.2. It can be observed that Spectr-H64 has 2^{128} weak keys of the form $K = (K_1, K_1, K_2, K_2, K_3, K_3, K_4, K_4)$ for which encryption is the same function as decryption, and thus double encryption reveals the plaintext. However, in this case, the key scheduling does not yield periodic subkeys. More specifically, all round subkeys are different with the exception that $Q_1 = Q_9$, $Q_3 = Q_{10}$ and $Q_4 = Q_{12}$, therefore slide attack does not apply.

On the other hand, it is observed that if all K_i's ($i \in \{1, 2, ..., 8\}$) are equal to each other, the same subkeys are produced for each round, and $Q_1 = Q_2 = ... = Q_{12} = Q$ (see Table 2.2). Spectr-H64 can then be viewed as a product of identical permutations for this key subspace of 2^{32} weak keys in which all K_i's are equal to each other. Our attack is applicable for only these keys.

The overall algorithm consists of the initial transformations, 12 identical round transformations and the final transformation for the mentioned 2^{32} weak keys. Because of the weakness described in the previous section, once we obtain a pair (P, P') of one round encryption where $P' = F(P)$, the 128 key bits of K_1, K_2, K_3 and K_4 can be extracted in a negligible time. Therefore the *easy cryptanalysis* requirement on F is satisfied.

We first consider the slide attack on a modified variant of Spectr-H64 without the initial and final transformations, as illustrated in Fig. 5.3 in simplified form, where

f denotes the overall transformation applied to the right half part of the round input.

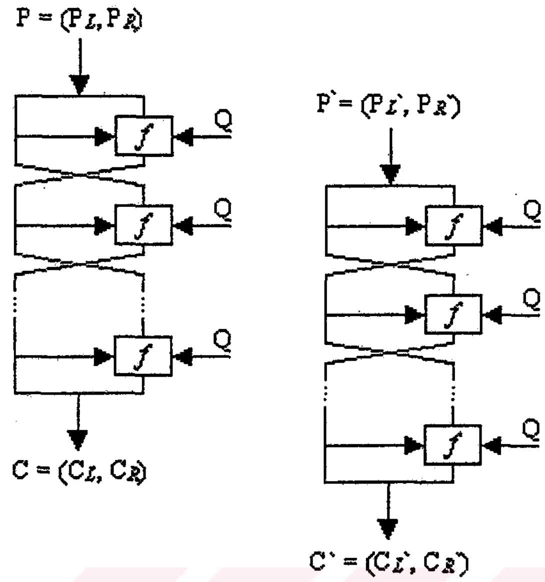

Figure 5.3 Illustration of Slide attack on Spectr-H64 (applicable for the 2^{32} keys, without initial and final transformations; if $C_L` = C_R$, then P` is the one round encryption of P)

A chosen plaintext slide attack encrypts two pools of chosen plaintexts P = (P_L, P_R) and P` = ($P_L`$, $P_R`$), where $P_L = P_R`$ is fixed, and P_R and $P_L`$ both take $2^{16}$ random values. Then, checking whether $C_L` = C_R$, we expect to find a vector P` in the second pool which is an exact one round encryption of the element P from the first pool with high probability, by the birthday paradox. Therefore, we can find a slid pair in 2^{32} checking operations (= comparison of two 32-bit blocks). After finding a slid pair, other adaptively chosen 32 plaintext-ciphertext pairs of the first round are easily found with the help of the procedure explained below:

1. Encrypt 32 plaintexts $P_i = P \oplus e_i$, and 32 plaintexts $P_j` = P` \oplus e_j$ corresponding to the slid pair (P, P`); for all $i \in \{33, 34, \ldots, 64\}, j \in \{1, 2, \ldots, 32\}$, and obtain $32 \times 32$ ciphertext pairs $C_i$ and $C_j`$. (Notice that the subscript of the

ciphertext simply indicates the corresponding plaintext, $F^r(P_i) = C_i$, and it does not imply that $C_i = C \oplus e_i$.)

2. Check whether $C_{iL}` = C_{jR}$ for each ciphertext pair; if they are equal, corresponding plaintext pair $P_i$ and $P_j`$ satisfy the equation $F(P_i) = P_j`$.

Notice that since one bit input differences cause one bit output differences after one round encryption of Spectr-H64, the above procedure works. Now, one can extract the 128 key bits used for the first round, as explained in the previous section. Since our attack works for the 2^{32} weak keys in which all K_i's ($i \in \{1, 2, ..., 8\}$) are equal to each other, knowing the extracted key bits is equivalent to knowing all the key bits. The attack is independent of the number of rounds of the cipher, and requires 2^{17} chosen plaintexts and 2^{32} steps (= checking operations).

Next, we consider the effect of the initial and final transformations on the cryptanalysis of Spectr-H64. One can observe from Fig. 2.5 that, if successive bits of the plaintext entering the same $P_{2/1}$ boxes are chosen the same, the initial transformation output is independent of the control bits. Hence, in the attempt of finding a slid pair, where one assigns the plaintexts $P = (P_L, P_R)$ and $P` = (P_L`, P_R`)$, such that $P_L = P_R`$, we can choose P_L and $P_R`$ in the above mentioned form that the initial transformation cannot affect. For example, if $P_L$ and $P_R`$ are chosen as all zero vectors (see Fig.5.4), the effect of the initial transformation is removed; however, the final transformation still remains effective unless $Q_{FT_L} = Q_{FT_R}$. If $Q_{FT_L} = Q_{FT_R}$, the left and right halves of the final transformation inputs are subject to the same permutation; and one can find a match for the vectors $C_L`$ and $C_R$ in $2^{32}$ checking operations. On the other hand, if $Q_{FT_L} \neq Q_{FT_R}$, we have to guess all possible ($2 \times 2^{16}$) input vectors of the final transformation in order to find a match between $C_L`$ and C_R.

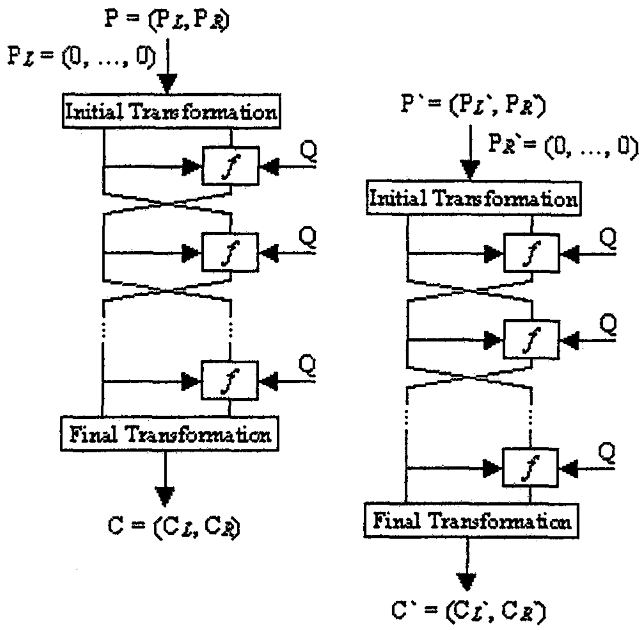

Figure 5.4 Illustration of the Effect of Initial and Final Transformations on the Slide Attack Shown in Fig.5.3

The slide attack on Spectr-H64 also requires 2^{17} chosen plaintexts, as for the slide attack on the modified variant of Spectr-H64. However, since we have to make $2^{16} \times 2^{16}$ checking operations among the vectors C_L` and C_R for each possible Q_{FT}, time complexity increases remarkably. Instead of the 2^{32} steps (a step being equivalent to the comparison of two 32-bit blocks) required by the slide attack on the modified variant of Spectr-H64, the chosen plaintext slide attack on Spectr-H64 requires 2^{64} steps. In addition, there may be some false matches between the vectors C_L` and C_R while guessing the vectors Q_{FT_L} and Q_{FT_R}, which can be checked immediately during the process of extracting the key bits. It should be seen that the correctness for the guesses of the left and right halves of the vector Q_{FT} can also be checked by a trial encryption if we exploit the restriction on the key bits. In this case, the secret-key can be found directly without the cryptanalysis process of one round encryption. After finding a slid pair, 32 adaptively chosen plaintexts can be found using a similar procedure, which is explained for the attack on the modified variant. The key bits are extracted after breaking one round of Spectr-H64 with the initial transformation, as explained in the previous section.

5.4 Weak Keys and Fixed Points

We observe that Spectr-H64 has a set of 2^{128} weak keys of the form $K = (K_1, K_1, K_2, K_2, K_3, K_3, K_4, K_4)$, for which encryption becomes the same function as decryption since the round subkeys for decryption become the same as the round subkeys of encryption (see Table 5.1); thus, double encryption reveals the plaintext.

Table 5.1 Round Subkeys of Spectr-H64 Used for Both Encryption and Decryption, for the Keys of the Form $K = (K_1, K_1, K_2, K_2, K_3, K_3, K_4, K_4)$

Q_{IT}	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}	Q_{11}	Q_{12}	Q_{FT}
	K_1	K_4	K_3	K_2	K_1	K_3	K_4	K_2	K_1	K_3	K_3	K_2	
	K_1	K_3	K_4	K_2	K_1	K_4	K_3	K_3	K_1	K_4	K_4	K_2	
K_1	K_3	K_1	K_1	K_3	K_4	K_2	K_2	K_4	K_3	K_1	K_1	K_3	K_1
	K_4	K_2	K_2	K_4	K_3	K_1	K_1	K_3	K_4	K_2	K_2	K_4	
	K_2	K_3	K_3	K_1	K_2	K_4	K_3	K_1	K_2	K_3	K_4	K_1	
	K_2	K_4	K_4	K_1	K_2	K_3	K_4	K_1	K_2	K_4	K_3	K_1	

Notice that the final transformation is the inverse of the initial transformation, if $Q_{IT} = Q_{FT}$, and that if we suppose the i^{th} round subkey of the encryption process is $Q_i = (A, B, C, D, E, F)$, where $A, B,.., F \in \{0,1\}^{32}$, and $i = 1, 2 ,..., 12$, then the $(13\text{-}i)^{th}$ round subkey of the decryption process is $Q_{(13-i)} = (E, F, C, D, A, B)$. Therefore, from Table 5.1, one can observe that the last six rounds of encryption decrypt the intermediate cipher result obtained after the first six rounds of encryption, if the left and right halves of the intermediate cipher result are equal to each other, for the mentioned set of 2^{128} weak keys. Hence, there are 2^{32} fixed points for each weak key, which demonstrate that there may be problems with using Spectr-H64 to build a secure hash function.

CHAPTER 6

CONCLUSIONS

The designers of Rijndael have taken one of the candidate s-box constructions in [21] that makes use of the mapping $x \rightarrow x^{-1}$ in Z_{2^8}, which is proposed for designing strong ciphers against cryptanalysis. Through our experimental results for the criteria of avalanche, strict avalanche and bit independence, it is shown that the s-box of Rijndael satisfies these criteria within very small values of relative absolute errors, such as $\in_{AVAL} = 6.25\%$, $\in_{SAC} = 12.5\%$ and $\in_{BIC} = 13.5\%$. Compared to the relative absolute errors of the exponentiating s-box of Safer family given in Table 4.2, as $\in_{AVAL} = 50.39\%$, $\in_{SAC} = 100\%$ and $\in_{BIC} = 100\%$, it is apparent that Rijndael's s-box parameters are much better than those of Safer. The sensitivity of the exponentiation box of Safer [11,14] to the input vector of e_{128}, which yields $\in_{SAC}$ and $\in_{BIC}$ values of 100%, can make the cipher vulnerable to differential cryptanalysis. On the other hand; overall performance of Safer K-64 measured with respect to the AWD criterion is better than that of Rijndael, at least at the end of the first and second rounds, as can be seen by comparing Fig. 4.3 and Fig. 4.4.

Our observations on the block ciphers Rijndael and Safer K-64 do not indicate that one of these ciphers is superior to the other; however, they should be interpreted as additional randomness tests [20]; as well as independent confirmation of analogous results [23,11,14].

It should be emphasized that, Safer K-64 is the best cipher, in terms of AWD criterion, among all three ciphers investigated in this thesis. The AWD criterion is satisfied with a high resemblance percentage around 98% at the second round of Rijndael and at the fourth round of Spectr; however, Safer K-64 satisfies the criterion with the same percentage at its second round.

The slide attack that we define is applicable for a small key subspace (one out of 2^{192}), so it does not indicate that Spectr-H64 is a weak cipher, but it should be interpreted as an independent confirmation of the observation in [15], which states that auto-key ciphers and data-dependent transformations are potentially vulnerable to conventional slide attacks. On the other hand, the 2^{128} weak keys, and the 2^{32} fixed points for each weak key that we observe demonstrate that the key scheduling of the algorithm should be improved.

As a final remark, the key scheduling of Spectr-H64 yields four round periodicity for 2^{64} weak keys of the form $K = (K_1, K_1, K_1, K_1, K_2, K_2, K_2, K_2)$, which may also make the cipher vulnerable to conventional slide attacks [15]. This periodicity may be suspected to make the cipher a candidate also for advanced slide attacks [16]; but we think that the techniques of advanced slide attacks are not applicable for Spectr-H64, because of the data-dependent permutations of the round subkeys.

REFERENCES

1. Daemen J. and Rijmen V.: AES Proposal: Rijndael. NIST Publication, 1999.

2. Massey J. L.: SAFER K-64: A Byte-Oriented Block-Ciphering Algorithm. Fast Software Encryption, Proceedings of Cambridge Security Workshop, Cambridge, U.K., LNCS 809, Springer Verlag, pp.1-17, 1994.

3. Knudsen L. R.: A Key Schedule Weakness in SAFER K-64. Advances in Cryptology, Proc. Crypto'95, LNCS 963, Springer Verlag, pp. 274-286, 1995.

4. Goots N. D., Moldovyan A. A. and Moldovyan N. A.: Fast Encryption Algorithm Spectr-H64. Proceedings of International Workshop MMM-ACNS-2001, St. Petersburg, Russia, LNCS 2052, Springer Verlag, pp. 275-286, May 21-23 2001.

5. Shannon C. E.: Communication Theory of Secrecy Systems. Bell Systems Technical Journal, Vol. 28, pp. 656-715, 1949.

6. Kam J. B. and Davida G. I.: Structured design of substution-permutation encryption networks. IEEE Transactions on Computers, Vol. C-28, No. 10, pp. 747-753, October 1979.

7. H. Feistel: Cryptography and computer privacy. Scientific American, Vol. 228, No. 5, pp. 15-23, May 1973.

8. Webster A. F. and Tavares S. E.: On the Design of S-boxes. Advances in Cryptology. Proceedings of CRYPTO'85, New York, Springer Verlag, pp. 523-534, 1986.

9. Shamir A. and Biham E.: Differential Cryptanalysis of DES-like Cryptosystems. Journal of Cryptology, Vol. 4, No. 1, pp. 3-72, 1991.

10. Meier W. and Staffelbach O.: Nonlinearity Criteria For Cryptographic Functions. Advances in Cryptology, Proc. EUROCRYPT'89, Springler-Verlag, pp. 549-562, 1989.

11. Aras E.: Analysis of Security Criteria for Block Ciphers. M.S. Thesis, Middle East Technical University, Ankara, Türkiye, September 1999.

12. Vergili I.: Statistics on Satisfaction of Security Criteria for Randomly Generated S-boxes. M.S. Thesis, Middle East Technical University, Ankara, Türkiye, June 2000.

13. Vergili I. and Yücel M. D.: On Satisfaction of Some Security Criteria for Randomly Chosen S-Boxes. Proc. 20th Biennial Symp. on Communications, Kingston, pp. 64-68, May 2000.

14. Aras E. and Yücel M. D.: Some Cryptographic Properties of Exponentiation and Logarithm Taking S-Boxes. Proc. 20th Biennial Symp. on Communications, Kingston, Canada, pp. 69-73, May 2000.

15. Biryukov A. and Wagner D.: Slide Attacks. Proceedings of FSE'99, LNCS 1636, Springer Verlag, pp. 245-259, 1999.

16. Biryukov A. and Wagner D.: Advanced Slide Attacks. Proceedings of Eurocrypt'2000, LNCS 1807, Springer Verlag, pp. 589-606, 2000.

17. Grossman E. K., Tucherman B.: Analysis of a Weakened Feistel-like Cipher. 1978 International Conference on Communications, Alger Press Limited, pp.46.3.1-46.3.5, 1978.

18. Biham E.: New Types of Cryptanalytic Attacks Using Related Keys. J. of Cryptology, Vol.7, pp.229-246, 1994.

19. Knudsen L. R.: Cryptanalysis of LOKI91. Proceedings of AUSCRYPT'92, LNCS 718, Springer Verlag, pp.196-208, 1993.

20. Soto J. and Basham L.: Randomnes Testing of the Advanced Encryption Standard Finalist Candidates. NIST Publication, March 2000.

21. Nyberg, K.: Differentially Uniform Mappings For Cryptography. Advances in Cryptology, Proceedings Eurocrypt'93, LNCS 765, T.Helleseth, Ed., Springer Verlag, pp.55-64, 1994.

22. Meier W., Staffelbach O.: Nonlinearity Criteria For Cryptographic Functions. Advances in Cryptology, Proceedings of Eurocrypt'89, Springer Verlag, pp.549-562, 1989.

23. Daemen, J.: Annex to AES Proposal Rijndael. Chapter 5: Propagation and Correlation, June 1998.

APPENDIX A

AWD CURVES OF RIJNDAEL

AWD curves of the block cipher Rijndael for one-bit input differences e_2, e_4, e_8, e_{16}, e_{32}, e_{64}, e_{100}, e_{110}, e_{120}, and e_{128} are given in figures A.1 through A.10; together with corresponding resemblance percentages $R_r^{e_i}$.that is defined by Eq. (4.6).

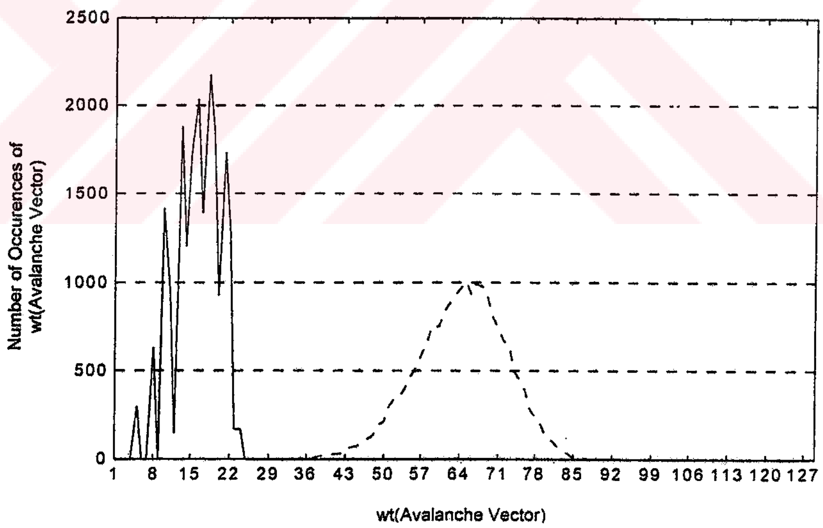

Figure A.1 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Rijndael for e_2, with $R_1^{e_2} = 0$ and $R_2^{e_2} = 0.8167$

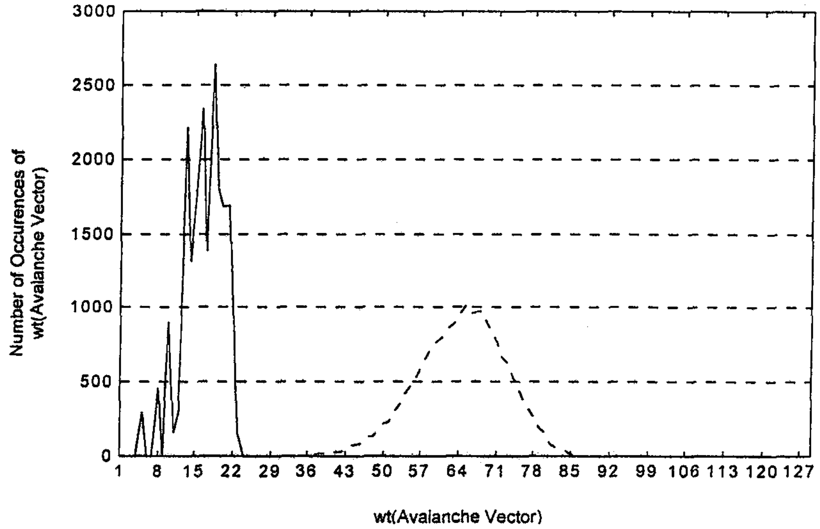

Figure A.2 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Rijndael for e_4, with $R_1^{e_4} = 0$ and $R_2^{e_4} = 0.8147$

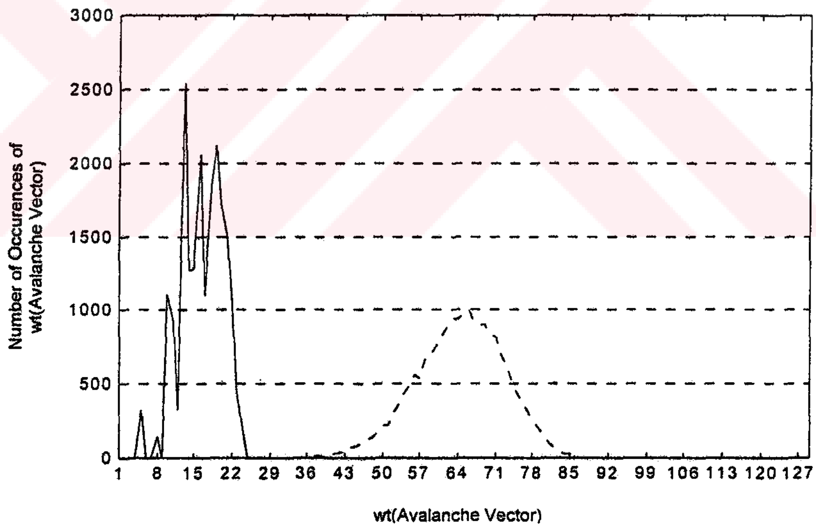

Figure A.3 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Rijndael for e_8, with $R_1^{e_8} = 0$ and $R_2^{e_8} = 0.8075$

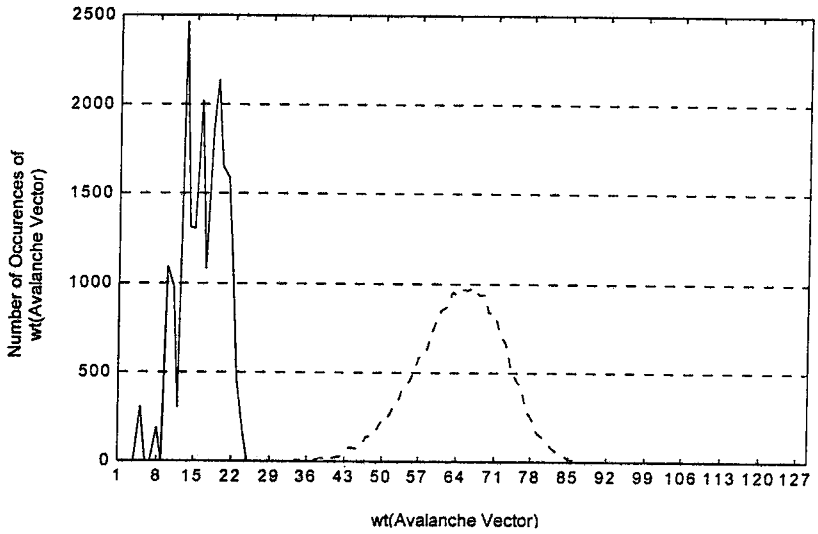

Figure A.4 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Rijndael for e_{16}, with $R_1^{e16} = 0$ and $R_2^{e16} = 0.8103$

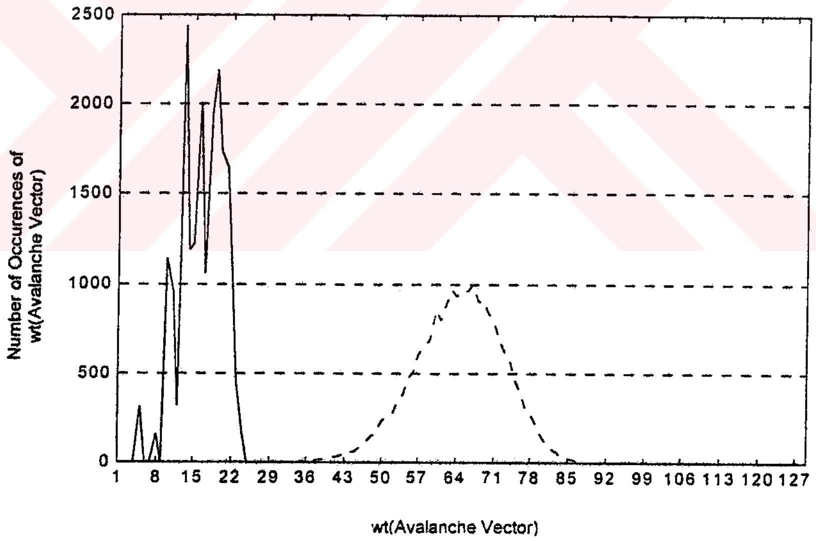

Figure A.5 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Rijndael for e_{32}, with $R_1^{e32} = 0$ and $R_2^{e32} = 0.8139$

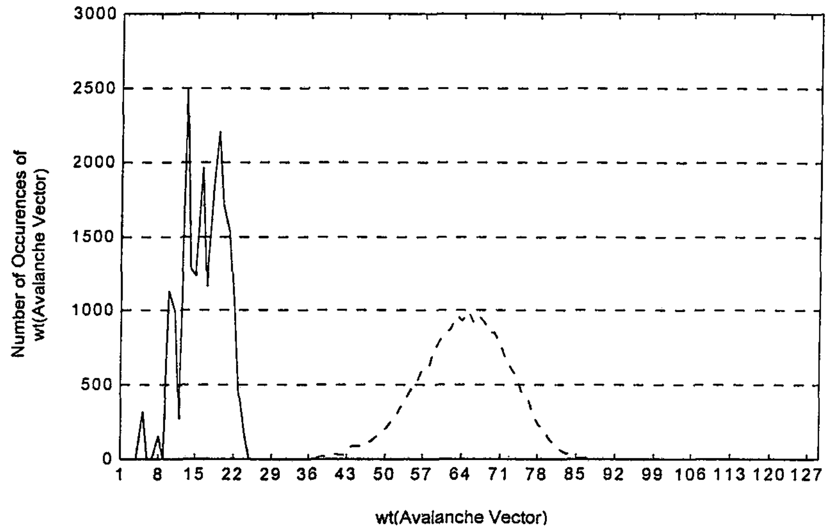

Figure A.6 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Rijndael for e_{64}, with $R_1^{e64} = 0$ and $R_2^{e64} = 0.8124$

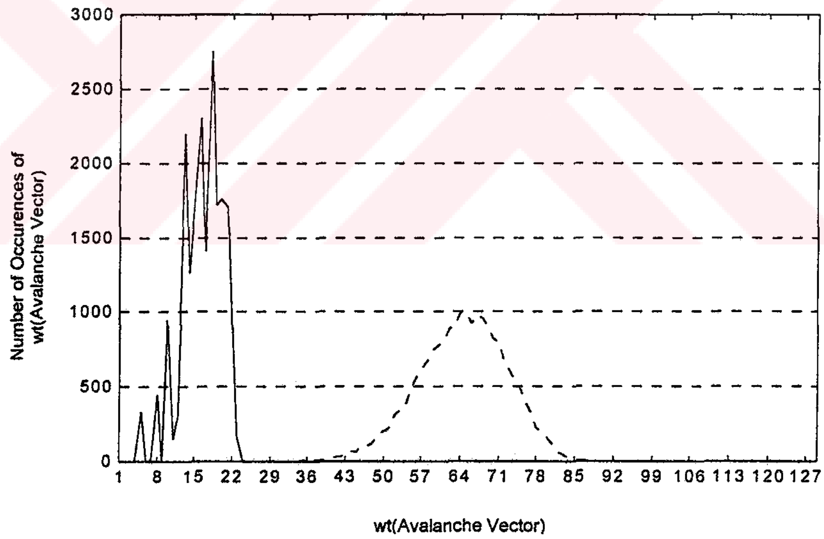

Figure A.7 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Rijndael for e_{100}, with $R_1^{e100} = 0$ and $R_2^{e100} = 0.8191$

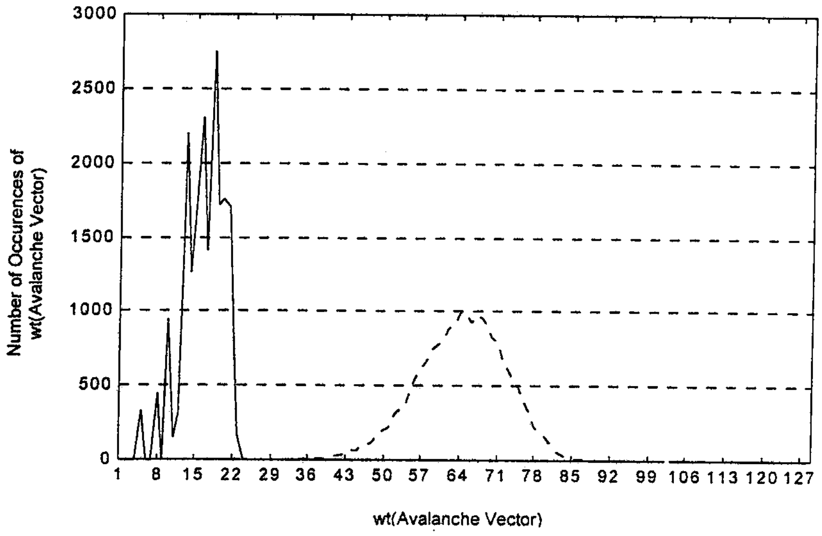

Figure A.8 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Rijndael for e_{110}, with $R_1^{e_{110}} = 0$ and $R_2^{e_{110}} = 0.8089$

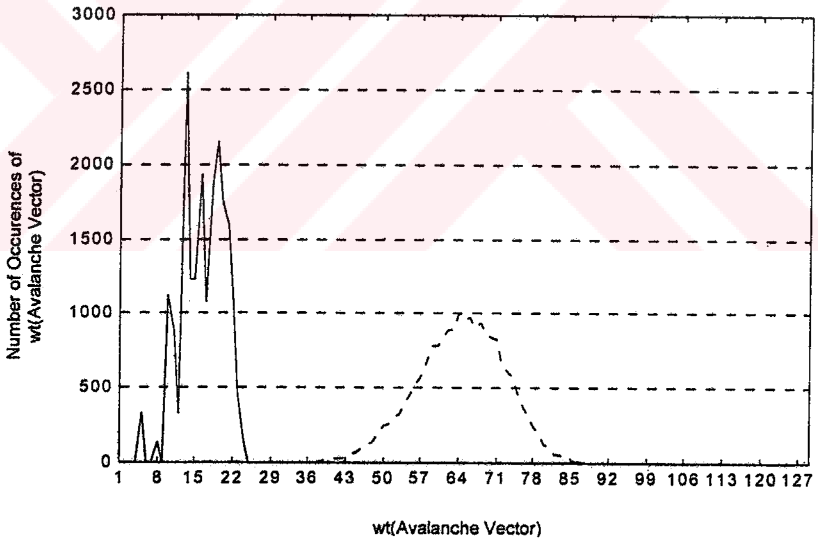

Figure A.9 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Rijndael for e_{120}, with $R_1^{e_{120}} = 0$ and $R_2^{e_{120}} = 0.8121$

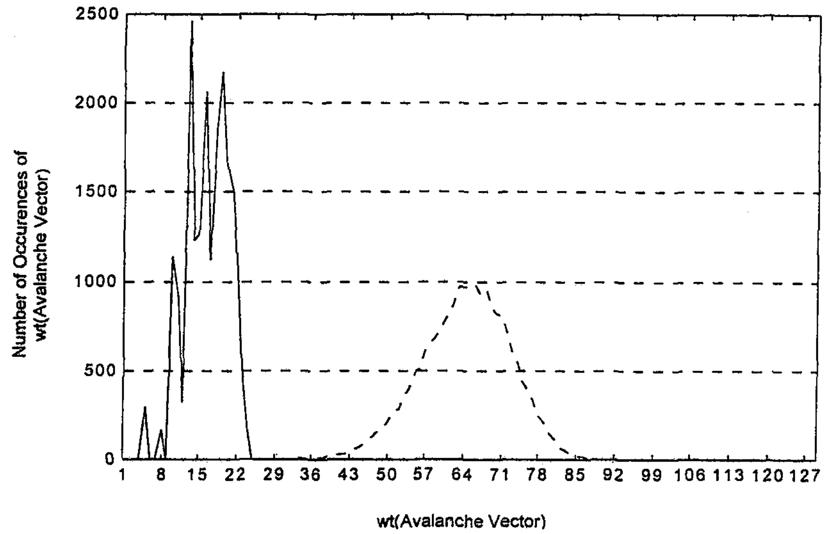

Figure A.10 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Rijndael for e_{128}, with $R_1^{e_{128}} = 0$ and $R_2^{e_{128}} = 0.8109$

APPENDIX B

AWD CURVES OF SAFER K-64

AWD curves of the block cipher Safer K-64 for one-bit input differences e_2, e_4, e_8, e_{16}, e_{32}, e_{40}, e_{45}, e_{50}, e_{55}, and e_{64} are given in figures B.1 through B.10; together with corresponding resemblance percentages $R_r^{e_i}$.that is defined by Eq. (4.6).

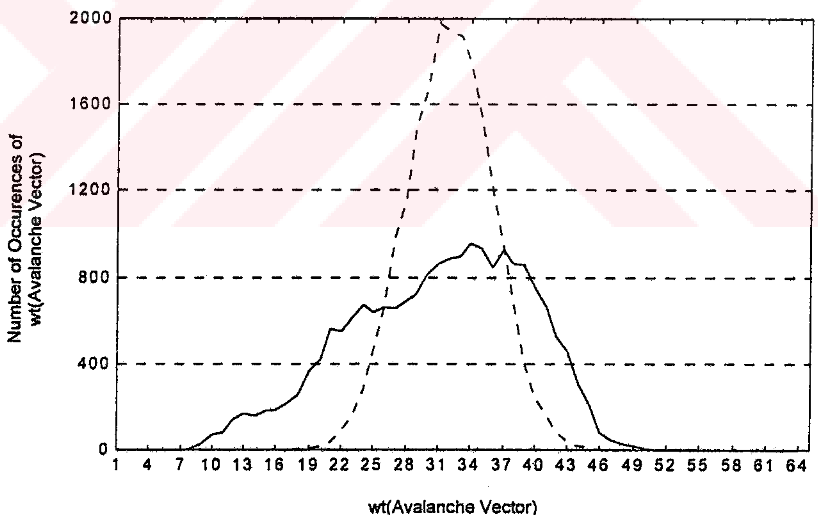

Figure B.1 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Safer K-64 for e_2, with $R_1^{e_2} = 0.6274$ and $R_2^{e_2} = 0.9857$

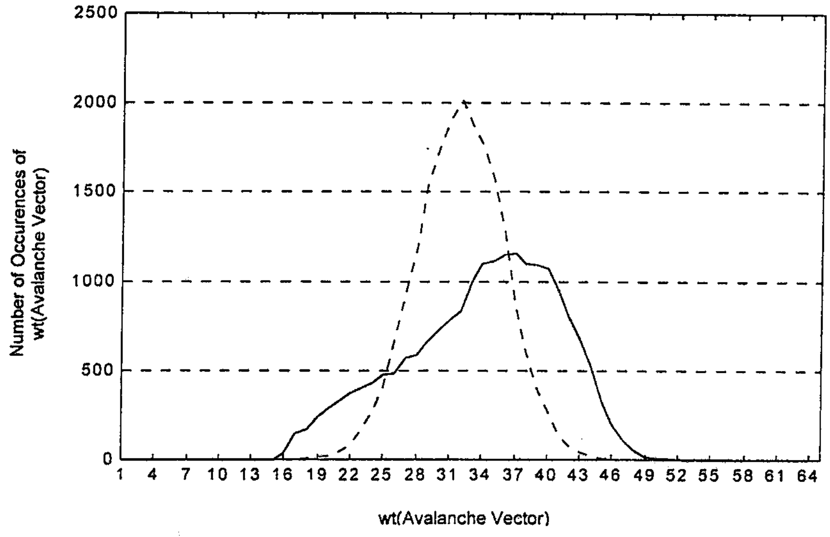

Figure B.2 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Safer K-64 for e_4, with $R_1^{e_4} = 0.6333$ and $R_2^{e_4} = 0.9857$

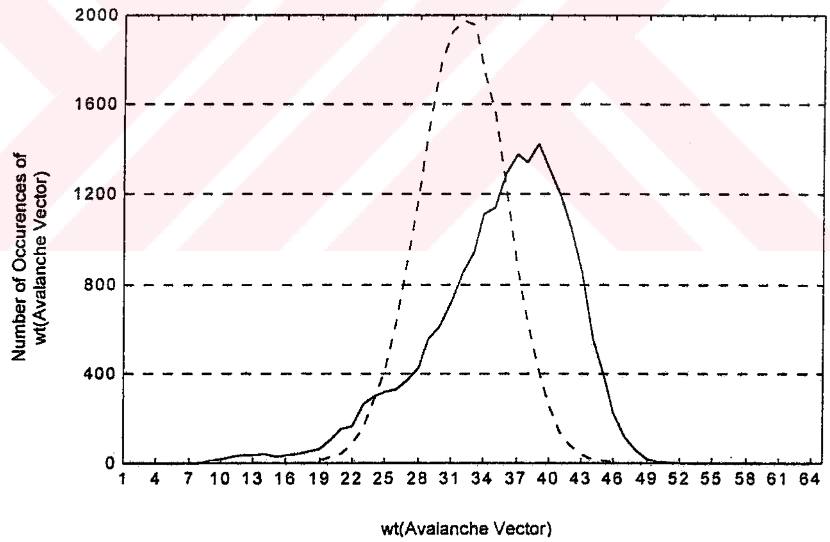

Figure B.3 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Safer K-64 for e_8, with $R_1^{e_8} = 0.5903$ and $R_2^{e_8} = 0.9911$

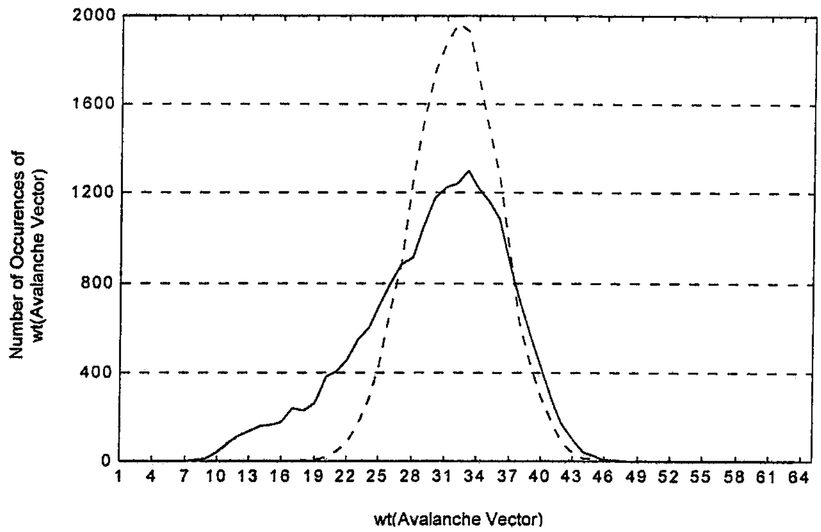

Figure B.4 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Safer K-64 for e_{16}, with $R_1^{e_{16}} = 0.7759$ and $R_2^{e_{16}} = 0.9893$

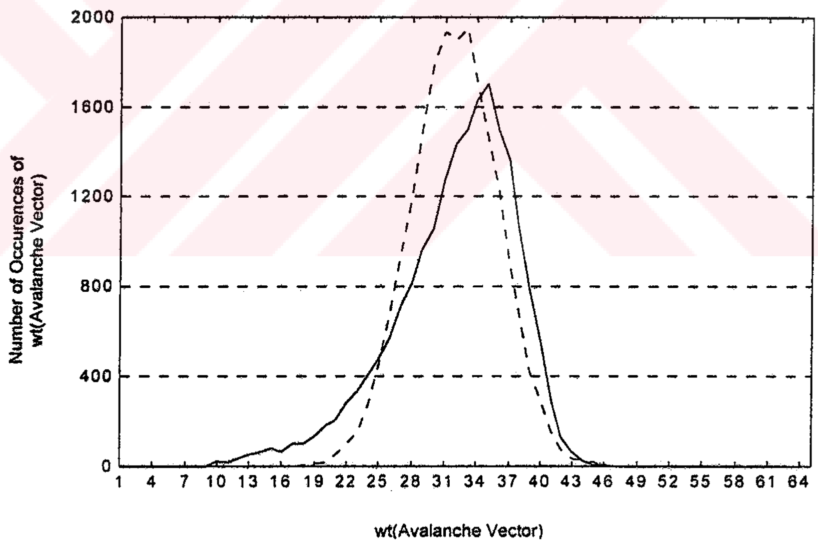

Figure B.5 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Safer K-64 for e_{32}, with $R_1^{e_{32}} = 0.8154$ and $R_2^{e_{32}} = 0.9877$

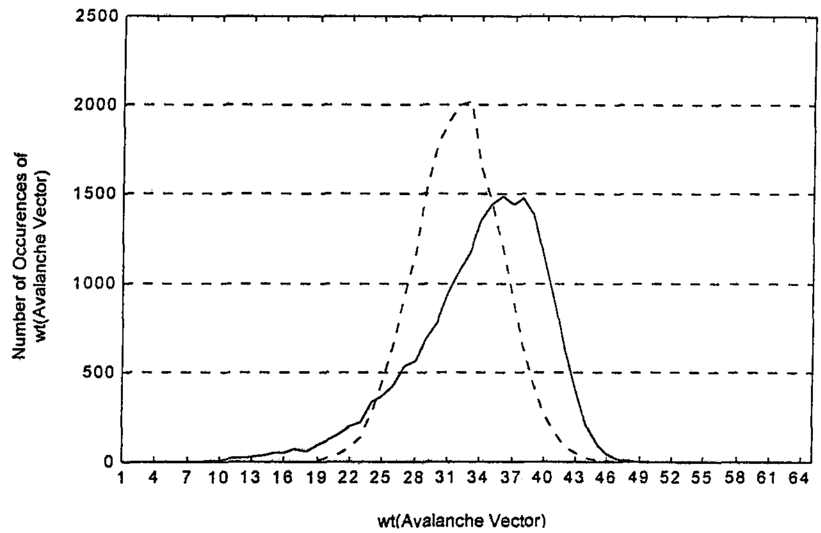

Figure B.6 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Safer K-64 for e_{40}, with $R_1^{e_{40}} = 0.6877$ and $R_2^{e_{40}} = 0.9896$

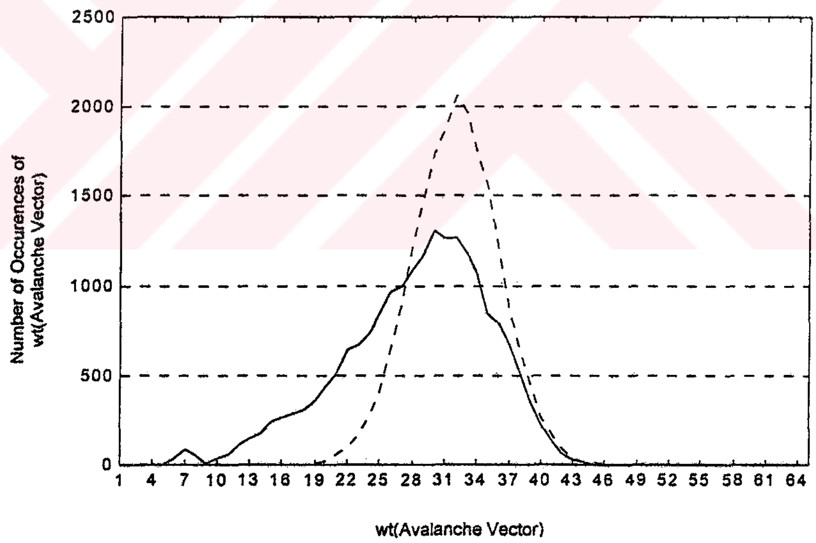

Figure B.7 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Safer K-64 for e_{45}, with $R_1^{e_{45}} = 0.7301$ and $R_2^{e_{45}} = 0.9860$

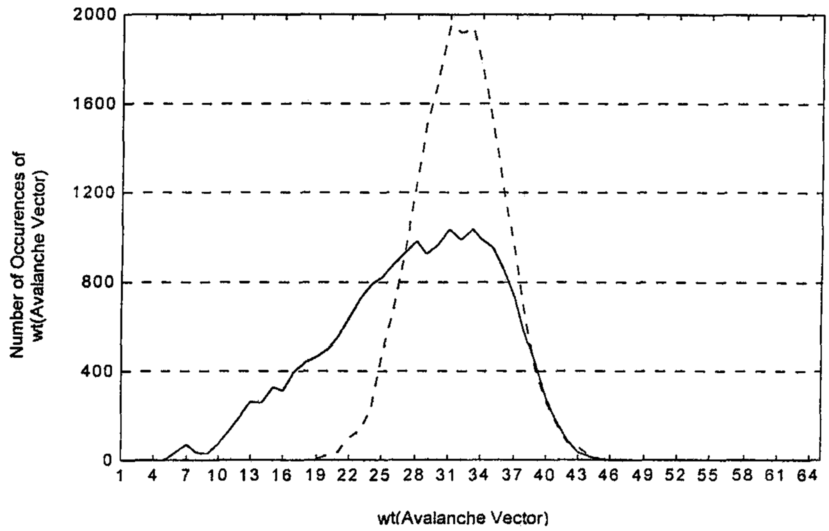

Figure B.8 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Safer K-64 for e_{50}, with $R_1^{e_{50}} = 0.6847$ and $R_2^{e_{50}} = 0.9884$

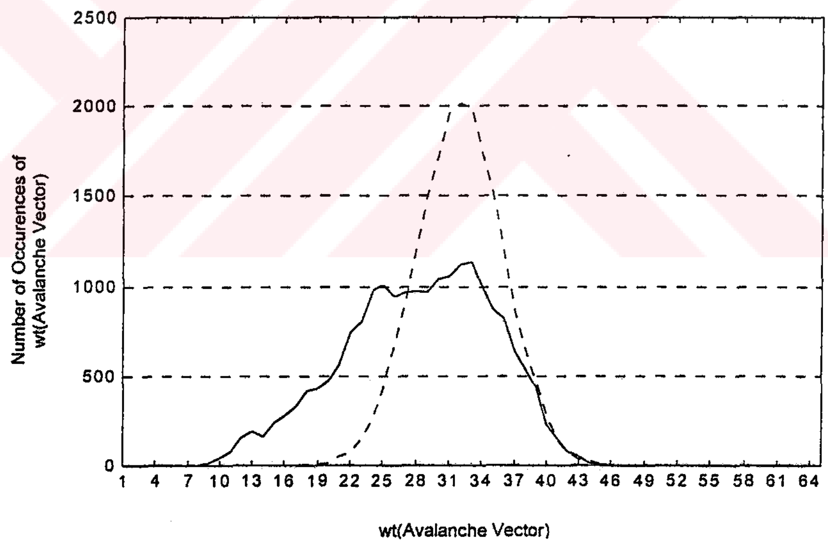

Figure B.9 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Safer K-64 for e_{55}, with $R_1^{e_{55}} = 0.6867$ and $R_2^{e_{55}} = 0.9882$

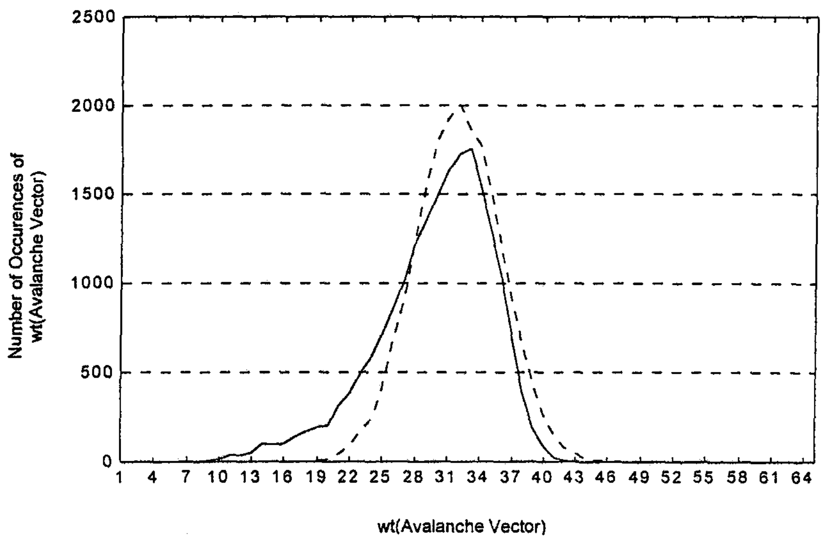

Figure B.10 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Safer K-64 for e_{64}, with $R_1^{e_{64}} = 0.8528$ and $R_2^{e_{64}} = 0.9880$

APPENDIX C

AWD CURVES OF SPECTR-H64

AWD curves of the block cipher Spectr-H64 for one-bit input differences e_2, e_4, e_8, e_{16}, e_{32}, e_{40}, e_{45}, e_{50}, e_{55}, and e_{64} are given in figures C.1 through C.10; together with corresponding resemblance percentages $R_r^{e_i}$.that is defined by Eq. (4.6).

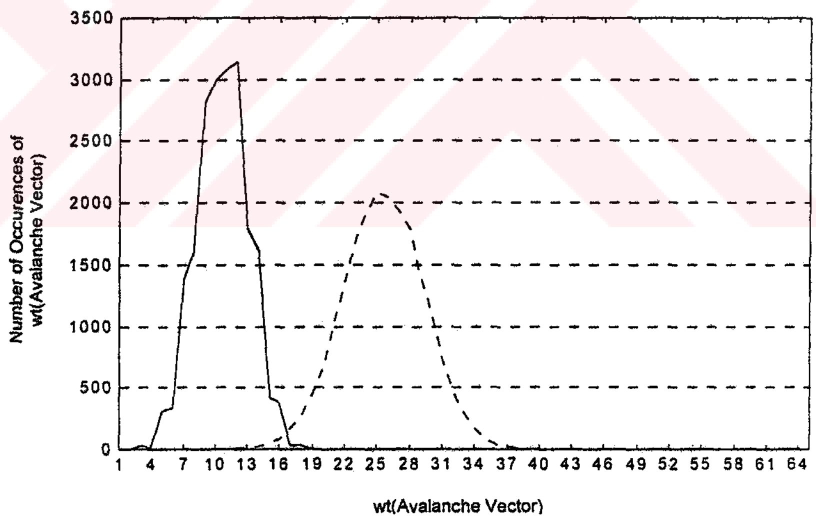

Figure C.1 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Spectr-H64 for e_2, with $R_1^{e_2} = 0.0003$ and $R_2^{e_2} = 0.4091$

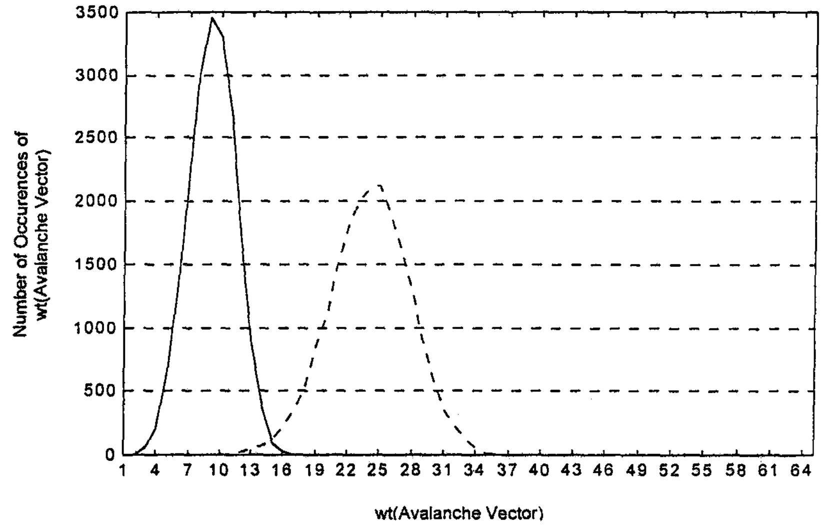

Figure C.2 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Spectr-H64 for e_4, with $R_1^{e_4} = 0$ and $R_2^{e_4} = 0.3077$

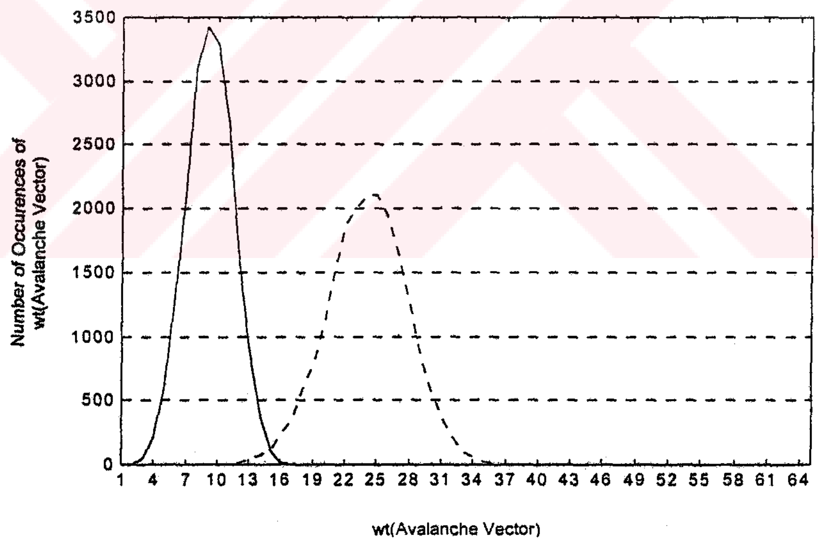

Figure C.3 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Spectr-H64 for e_8, with $R_1^{e_8} = 0$ and $R_2^{e_8} = 0.3060$

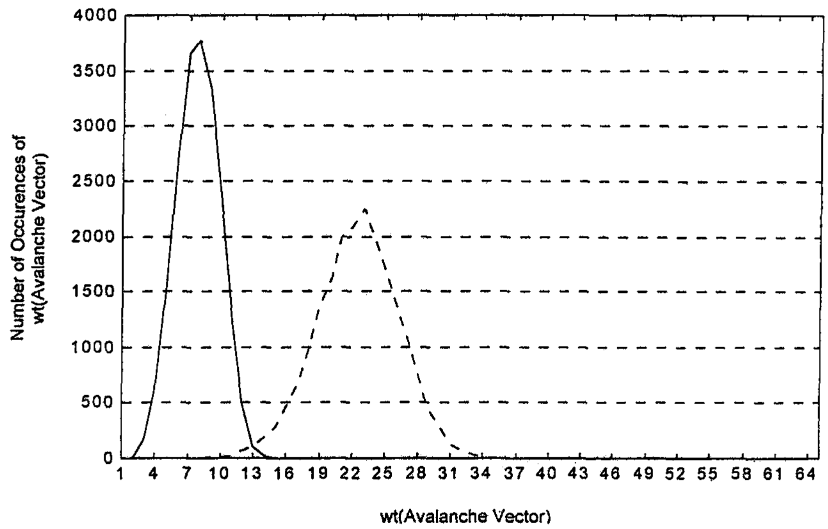

Figure C.4 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Spectr-H64 for e_{16}, with $R_1^{e_{16}} = 0$ and $R_2^{e_{16}} = 0.2170$

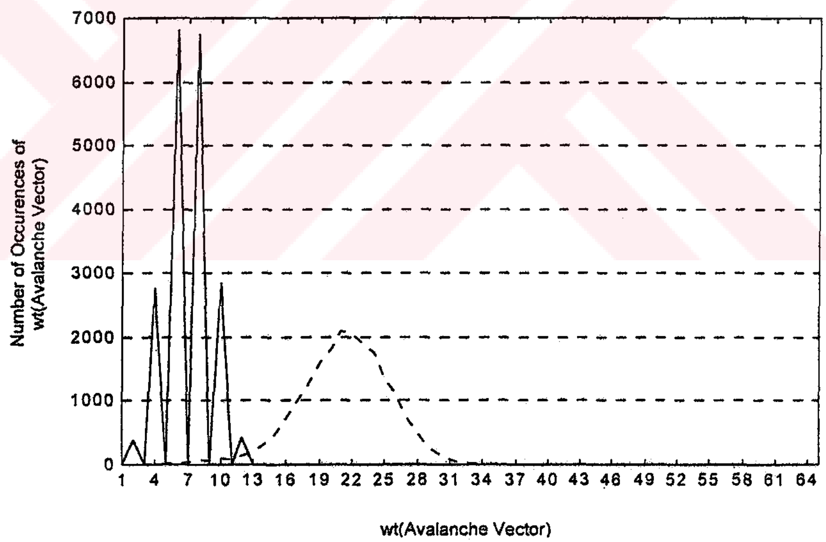

Figure C.5 First Round (*solid*) and Second Round (*dashed*) AWD Curves of Spectr-H64 for e_{32}, with $R_1^{e_{32}} = 0$ and $R_2^{e_{32}} = 0.1746$

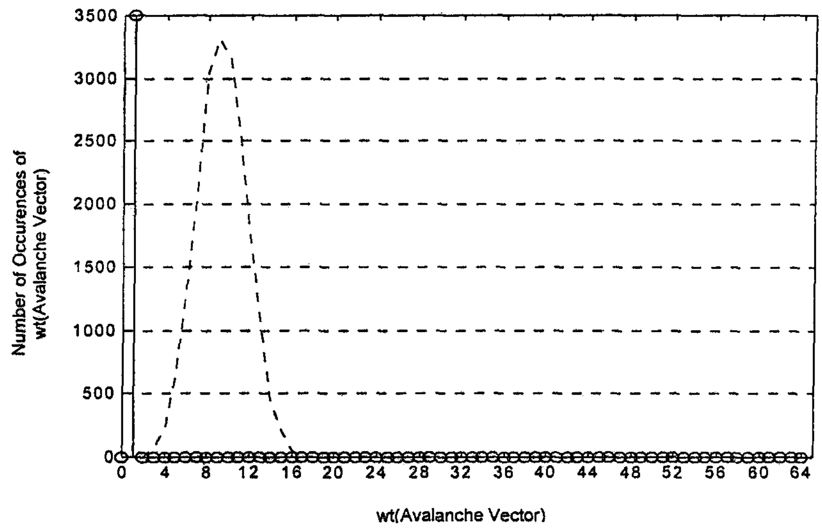

Figure C.6 First Round (*marked with* "o") and Second Round (*dashed*) AWD Curves of Spectr-H64 for e_{40}, with $R_1^{e_{40}} = 0$ and $R_2^{e_{40}} = 0.0001$

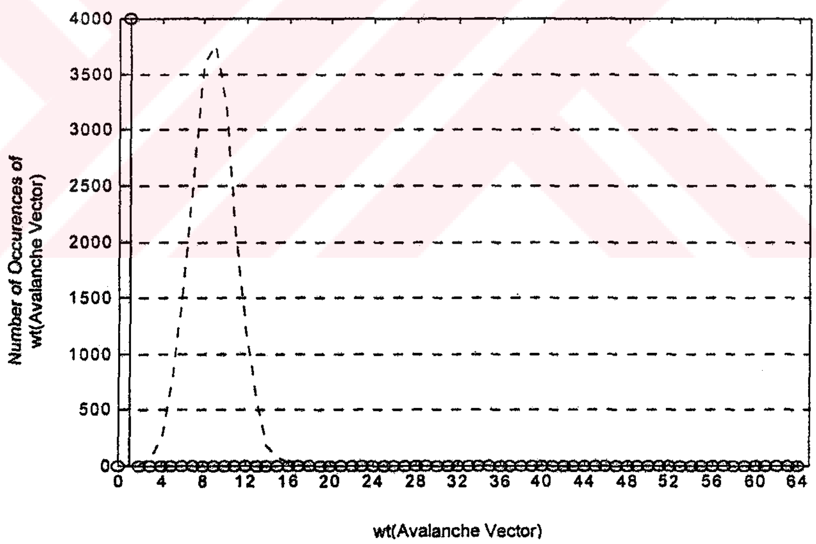

Figure C.7 First Round (*marked with* "o") and Second Round (*dashed*) AWD Curves of Spectr-H64 for e_{45}, with $R_1^{e_{45}} = 0$ and $R_2^{e_{45}} = 0.0001$

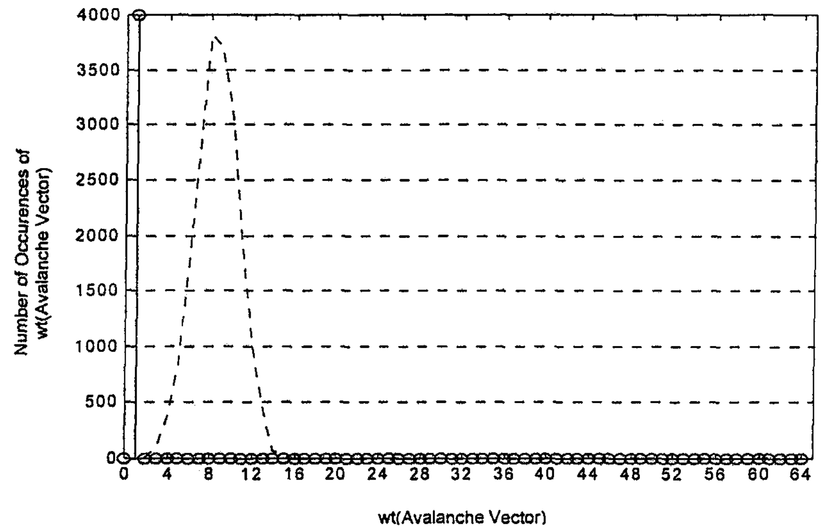

Figure C.8 First Round (*marked with* "o") and Second Round (*dashed*) AWD Curves of Spectr-H64 for e_{50}, with $R_1^{e_{50}} = 0$ and $R_2^{e_{50}} = 0.0002$

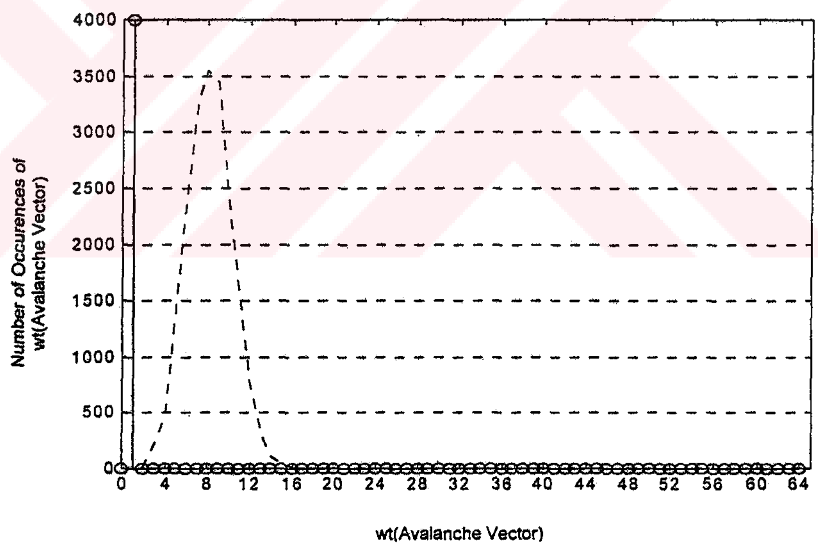

Figure C.9 First Round (*marked with* "o") and Second Round (*dashed*) AWD Curves of Spectr-H64 for e_{55}, with $R_1^{e_{55}} = 0$ and $R_2^{e_{55}} = 0.0001$

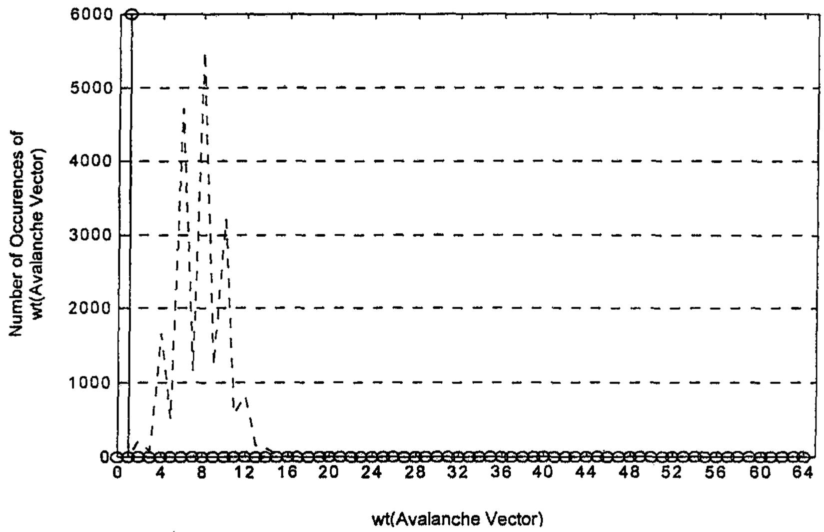

Figure C.10 First Round (*marked with* “o”) and Second Round (*dashed*) AWD Curves of Spectr-H64 for e_{64}, with $R_1^{e_{64}} = 0$ and $R_2^{e_{64}} = 0.0001$

APPENDIX D

WALSH-HADAMARD TRANSFORMS OF RIJNDAEL'S S-BOX, SAFER'S EXPONENTIATING S-BOX, AND SAFER'S LOGARITHM TAKING S-BOX

Let f: $Z_2^n \to Z_2$ be a Boolean function that produces a single-bit result for each possible combination of values from n Boolean variables. A Boolean function $f(\mathbf{X})$ is called an affine function of $\mathbf{X} = (x_1, x_2, \ldots, x_n) \in Z_2^n$, if it is in the form

$$f(\mathbf{X}) = a_1 \otimes x_1 \oplus a_2 \otimes x_2 \oplus \ldots \oplus a_n \otimes x_n \oplus c = \mathbf{w} \bullet \mathbf{X} \oplus c \qquad \textbf{(A.1)}$$

where $a_1, a_2, \ldots, a_n, c$ belong to Z_2, $\mathbf{w} = (a_1, a_2, \ldots, a_n) \in Z_2^n$, and $\otimes$, $\oplus$ & $\bullet$ respectively denote addition, multiplication and inner product in Z_2. $f(\mathbf{X})$ is called linear if $c = 0$. The sequence f_s of the Boolean function $f(\mathbf{X})$ is defined for all possible values $\mathbf{X} = X_i$ as:

$$f_s = \{(-1)^{f(X_0)}, (-1)^{f(X_1)}, \ldots, (-1)^{f(X_{2^n-1})}\} = \{(-1)^{f(\mathrm{X})}\}|_{\mathbf{X} = X_i} \qquad \textbf{(A.2)}$$

where X_i is the n-bit vector corresponding to the binary representation of the integer $i = 0, \ldots, 2^n - 1$. In the space of Boolean functions, sequences of all linear functions form an orthogonal basis with respect to the inner product operation in the field of real numbers. The representation of a Boolean function $f(\mathbf{X})$ with respect to this basis is called the Walsh-Hadamard Transform [22, 23], or spectrum of $f(\mathbf{X})$:

$$W\{f(\mathbf{X})\} = F(\mathbf{w}) = \sum_{x \in Z_2^n} (-1)^{f(\mathbf{X})}(-1)^{\mathbf{w}\cdot\mathbf{X}} = f_s \bullet (\mathbf{w}\cdot\mathbf{X})_s \tag{A.3}$$

In (A.3), f_s and $(\mathbf{w}\cdot\mathbf{X})_s$ stand for the sequences of $f(\mathbf{X})$, and 2^n possible linear functions $\mathbf{w}\cdot\mathbf{X}$ respectively, and the inner product $\bullet$ (not $\cdot$) is defined in the field of real numbers. As a result, $F(\mathbf{w})$ takes even integer values in the interval $[-2^n, 2^n]$. It can be shown that the overall nonlinearity measure NLM_f, defined in Section 3.5, by equations (3.10) and (3.11) is equivalent to the following equation:

$$NLM_f = 2^{n-1} - (\tfrac{1}{2}) \max_{\mathbf{w}} \{|f_s \bullet (\mathbf{w}\cdot\mathbf{X})_s|\} = 2^{n-1} - (\tfrac{1}{2}) \max_{\mathbf{w}} \{|F(\mathbf{w})|\} \tag{A.4}$$

Hence, in order for a Boolean function $f(\mathbf{X})$ to be highly nonlinear, the absolute value of its Walsh-Hadamard Transform $|F(\mathbf{w})| = |f_s \bullet (\mathbf{w}\cdot\mathbf{X})_s|$ should not take large values.

Since the s-boxes of Rijndael and Safer map 8-bit input vector $\mathbf{X} = (x_1, x_2, \ldots, x_8)$ to 8-bit output vector $\mathbf{Y} = (y_1, y_2, \ldots, y_8)$, each s-box consists of 8 Boolean functions $f(\mathbf{X}) = y_1, f(\mathbf{X}) = y_2, \ldots, f(\mathbf{X}) = y_8$. The Walsh-Hadamard Transforms of the Boolean functions for Rijndael's s-box, Safer's exponentiating s-box, and Safer's logarithm taking s-box are given in figures D.1 through D.8, figures D.9 through D.16, and figures D.17 through D.24, respectively. Moreover, for Safer's exponentiating s-box, the Walsh-Hadamard Transform of the Boolean function "$f(\mathbf{X}) = (0, 0, 1, 1, 0, 0, 1, 0) \cdot (y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8) = y_3 \oplus y_4 \oplus y_7$" is given in Fig. D.25, corresponding to the vector $\mathbf{j} = (0, 0, 1, 1, 0, 0, 1, 0)$ in $\mathbf{z} = (\mathbf{j}, \mathbf{w}, c)$ with the lowest $NLM_f(z)$ value (see section 4.5). The Walsh-Hadamard

Transform of the Boolean function $f(\mathbf{X}) = y_3 \oplus y_4 \oplus y_5 \oplus y_7$ for Safer's logarithm taking s-box is also given in Fig. D.26, corresponding to the vector $\mathbf{w} = (0, 0, 1, 1, 1, 0, 1, 0)$ in $\mathbf{z} = (\mathbf{j}, \mathbf{w}, c)$ with the lowest $NLM_f(z)$ value.

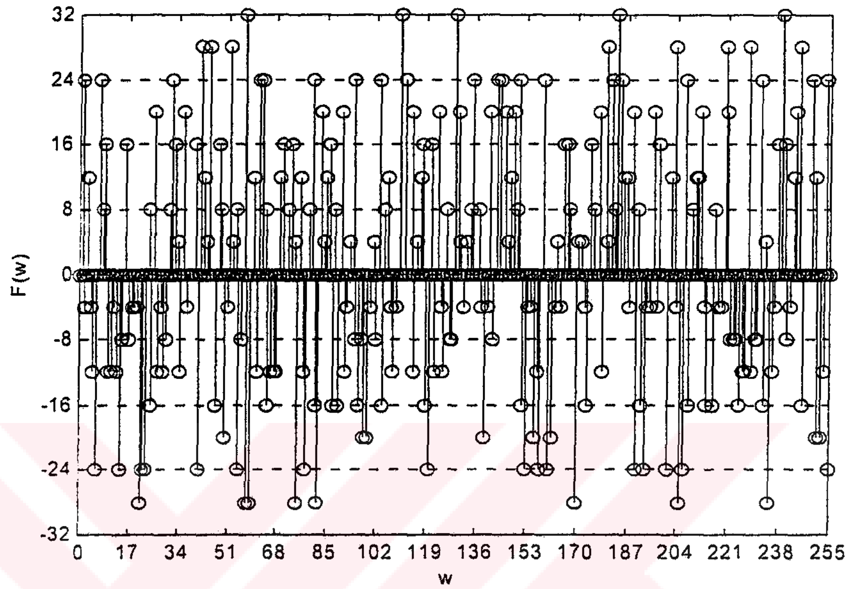

Figure D.1 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_1$ of Rijndael's s-box

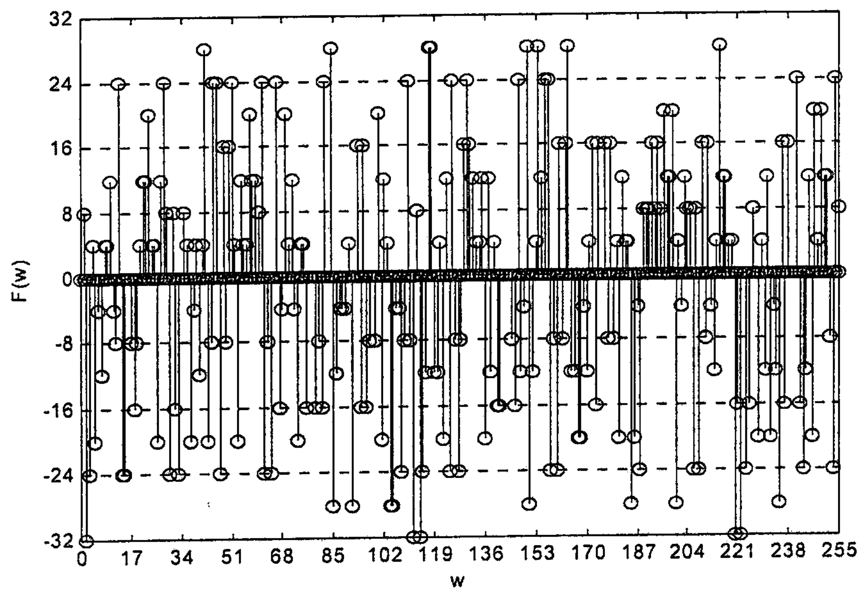

Figure D.2 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_2$ of Rijndael's s-box

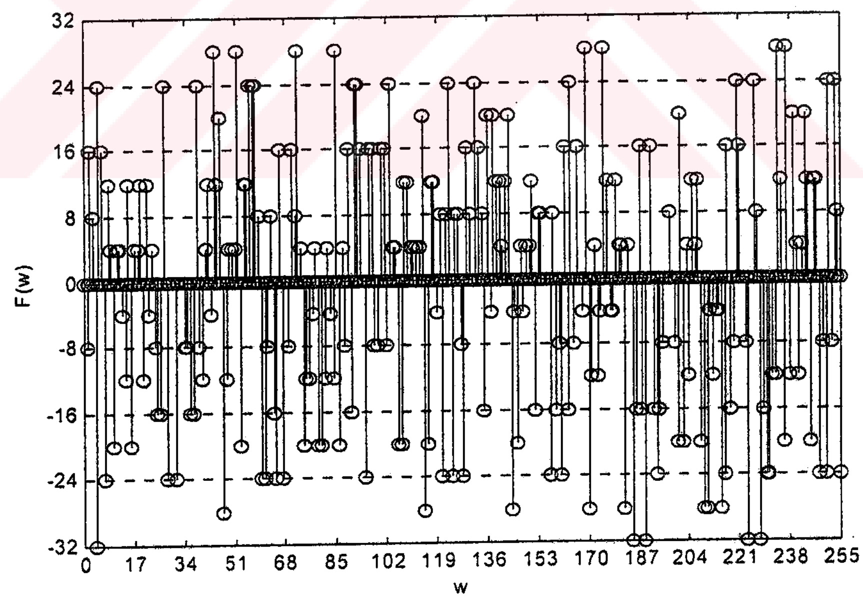

Figure D.3 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_3$ of Rijndael's s-box

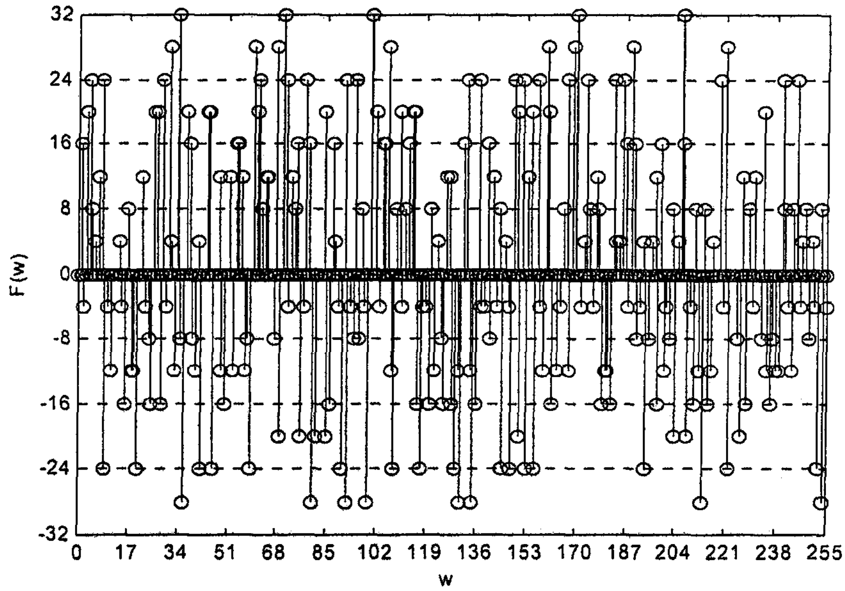

Figure D.4 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_4$ of Rijndael's s-box

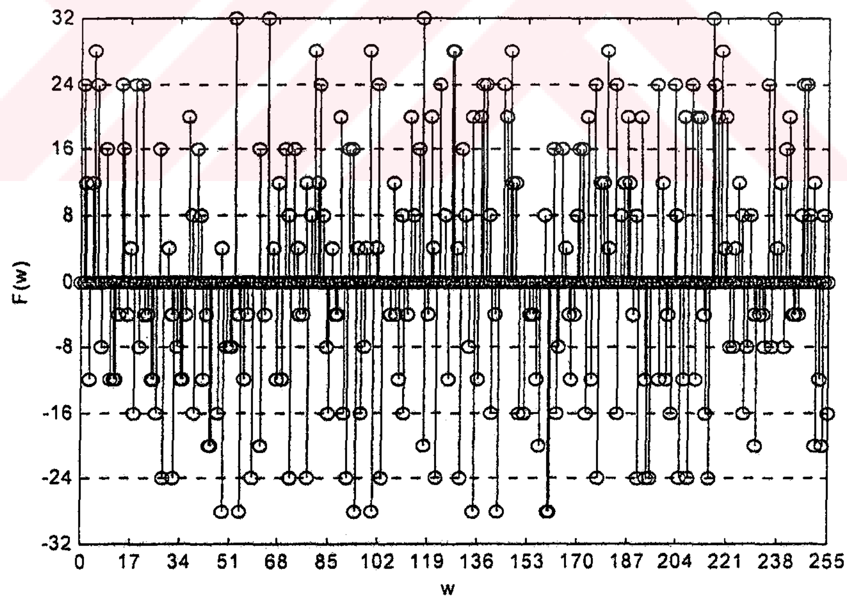

Figure D.5 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_5$ of Rijndael's s-box

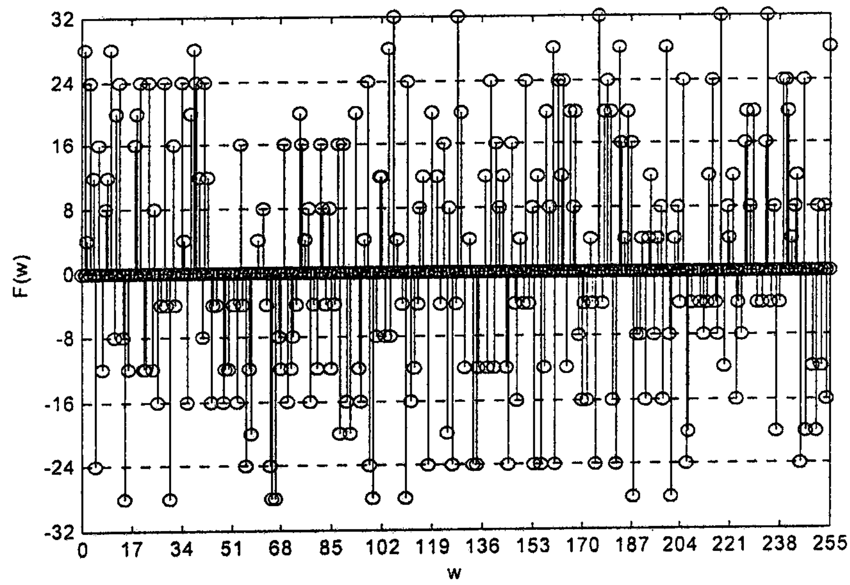

Figure D.6 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_6$ of Rijndael's s-box

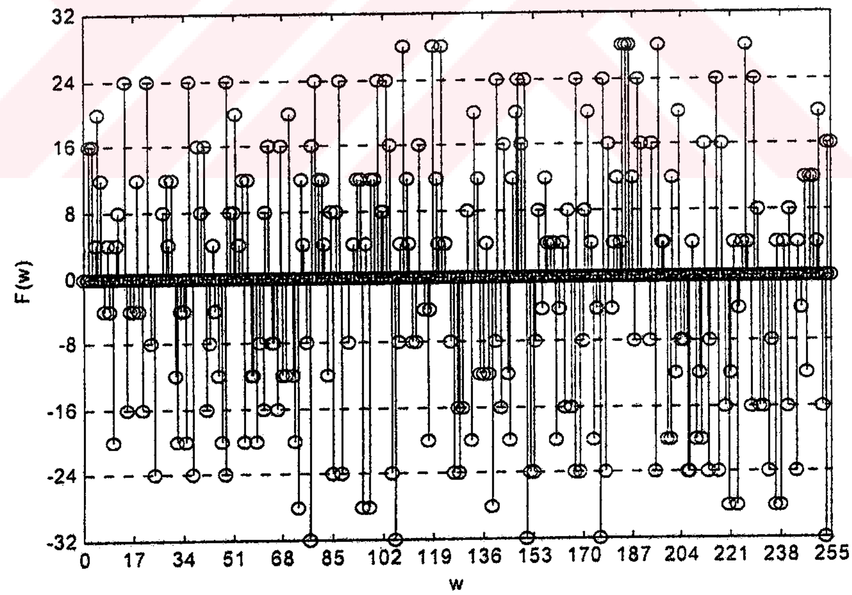

Figure D.7 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_7$ of Rijndael's s-box

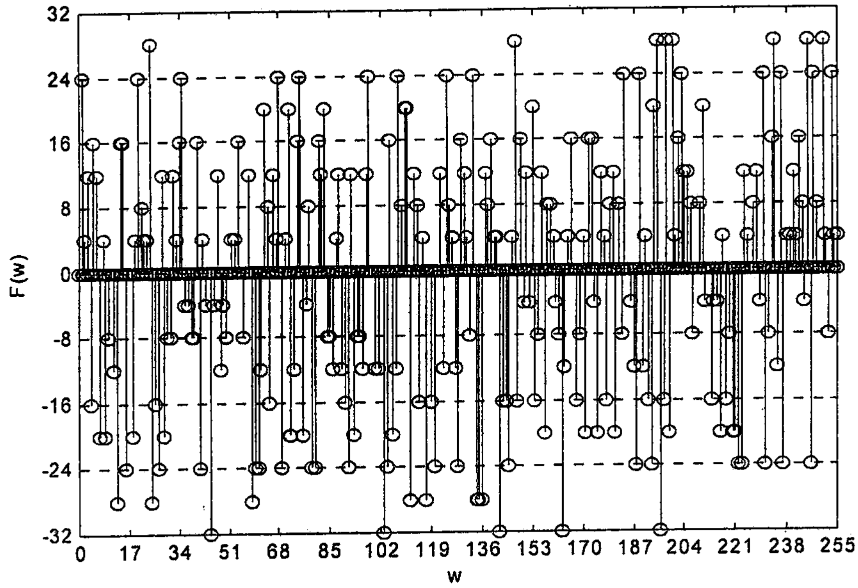

Figure D.8 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_8$ of Rijndael's s-box

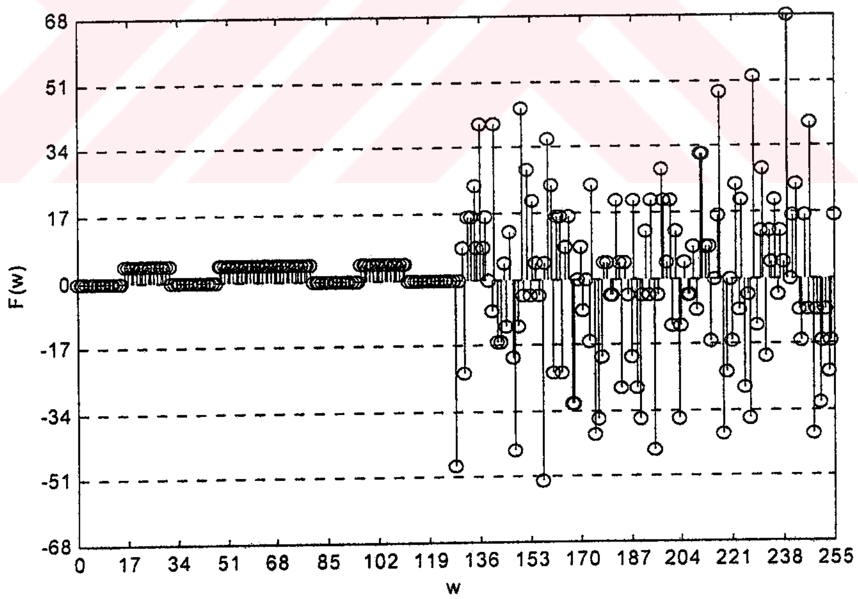

Figure D.9 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_1$ of Safer's exponentiating s-box

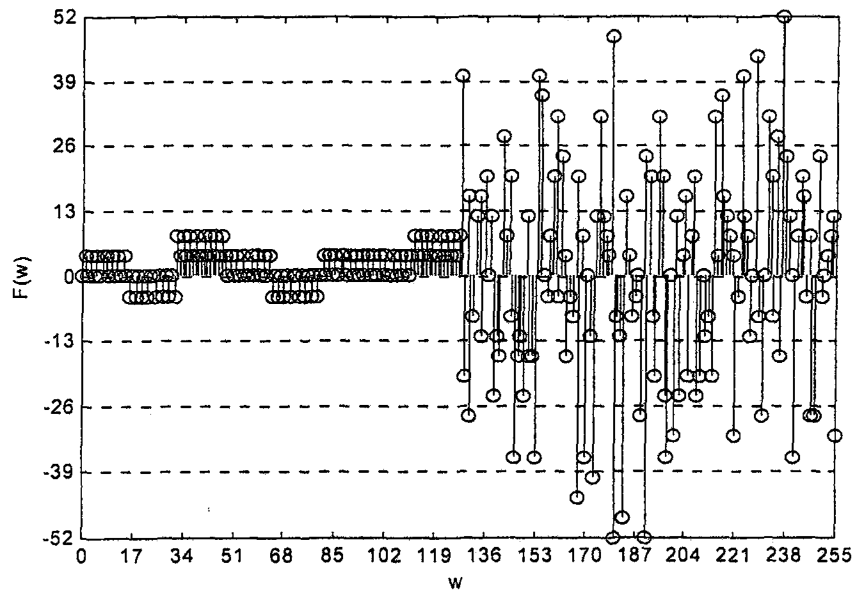

Figure D.10 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_2$ of Safer's exponentiating s-box

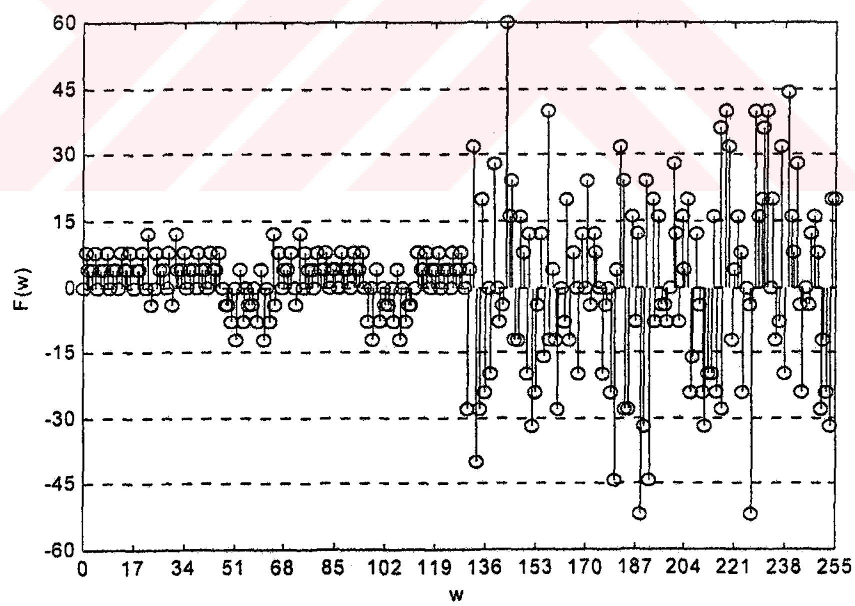

Figure D.11 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_3$ of Safer's exponentiating s-box

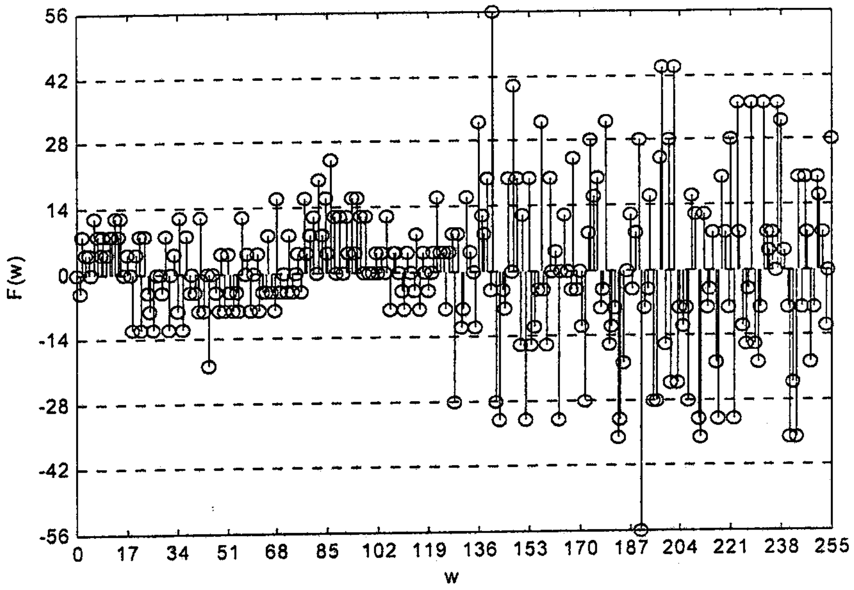

Figure D.12 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_4$ of Safer's exponentiating s-box

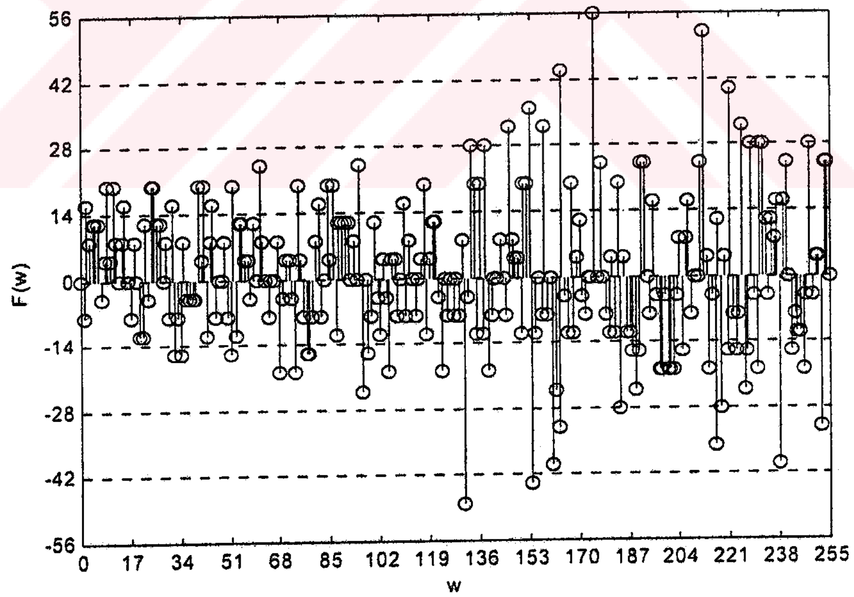

Figure D.13 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_5$ of Safer's exponentiating s-box

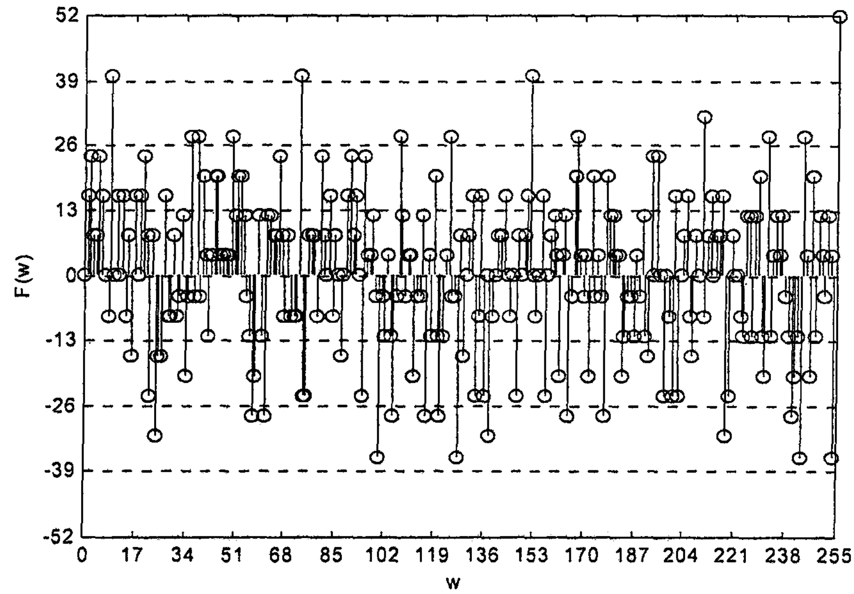

Figure D.14 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_6$ of Safer's exponentiating s-box

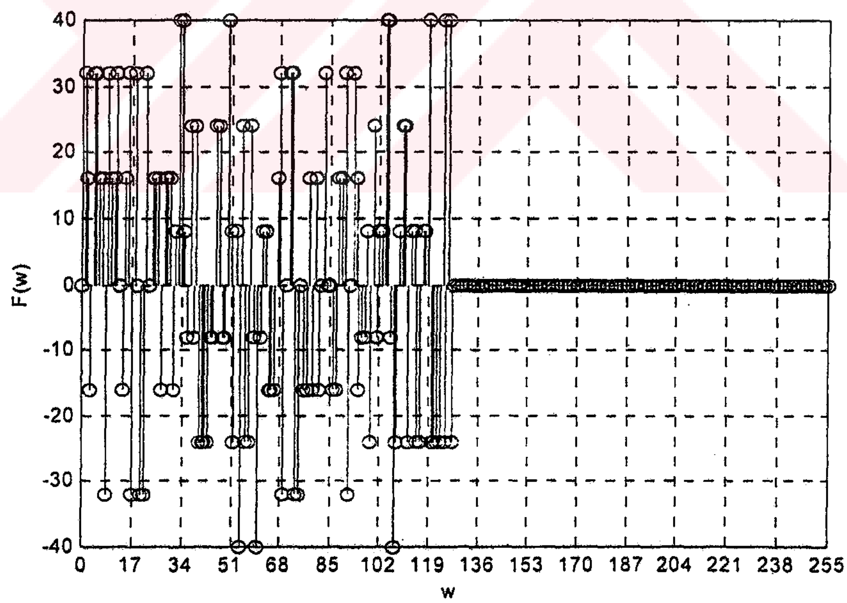

Figure D.15 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_7$ of Safer's exponentiating s-box

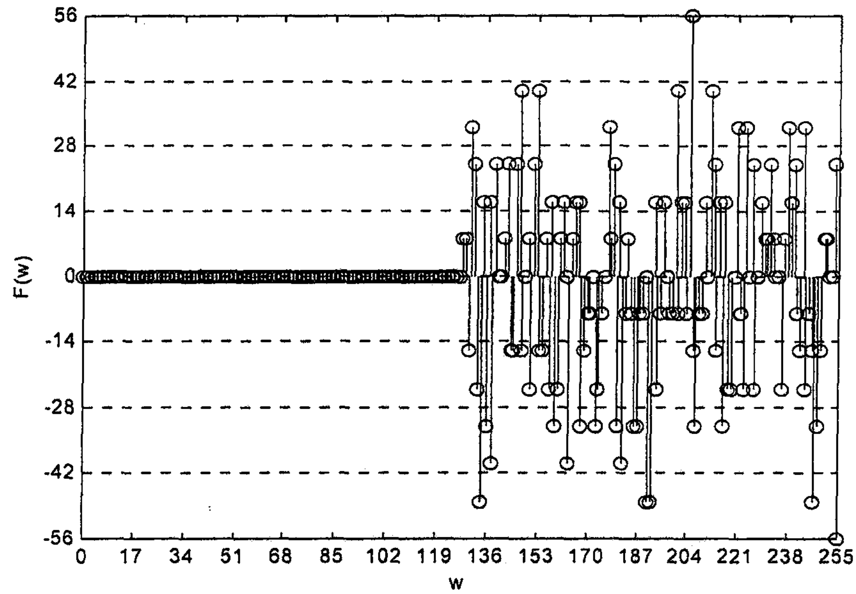

Figure D.16 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_8$ of Safer's exponentiating s-box

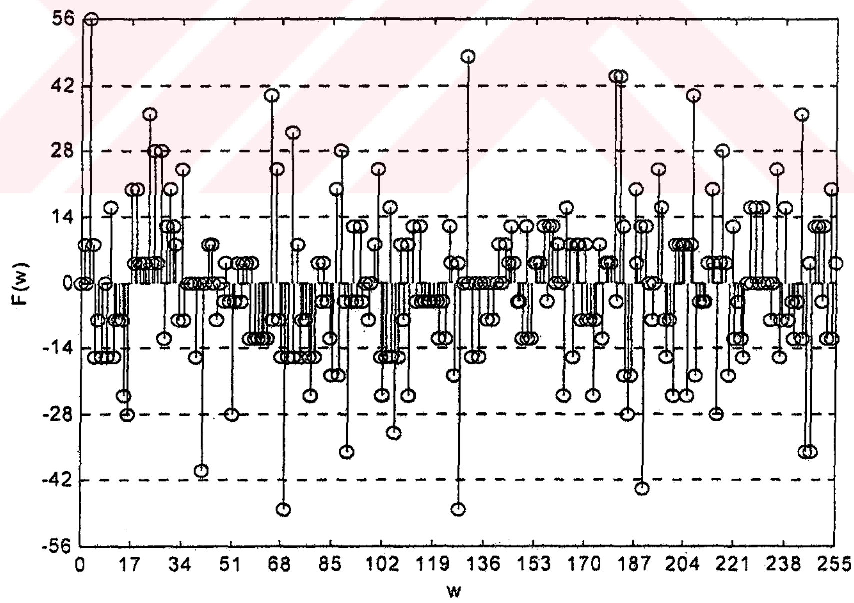

Figure D.17 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_1$ of Safer's logarithm taking s-box

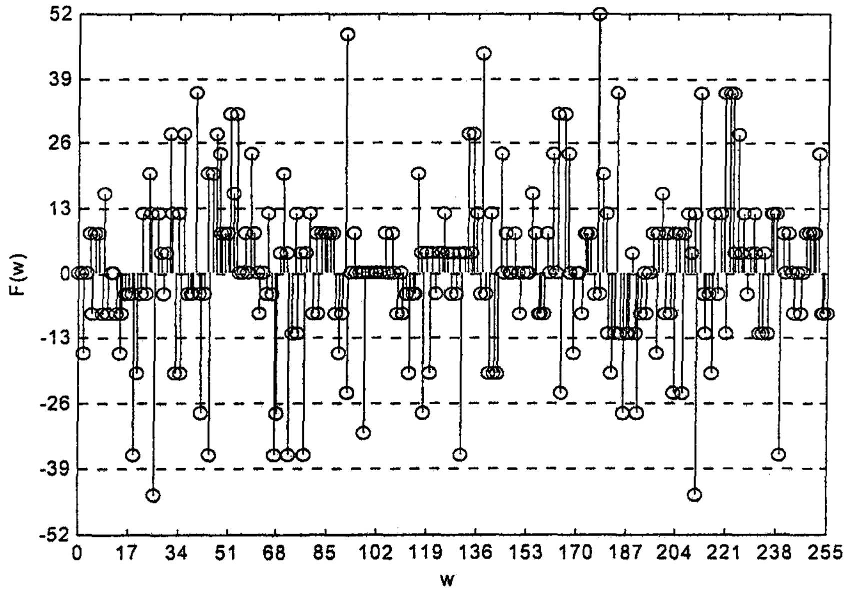

Figure D.18 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_2$ of Safer's logarithm taking s-box

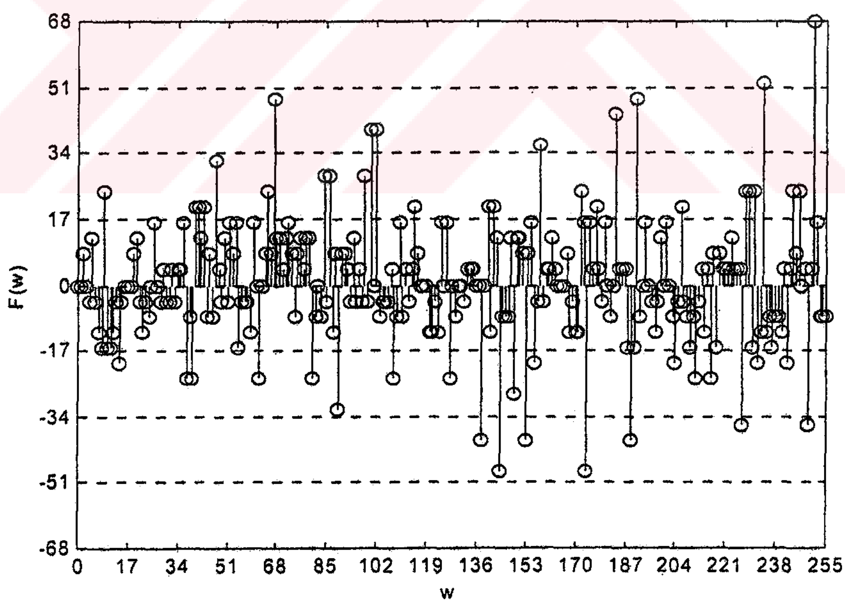

Figure D.19 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_3$ of Safer's logarithm taking s-box

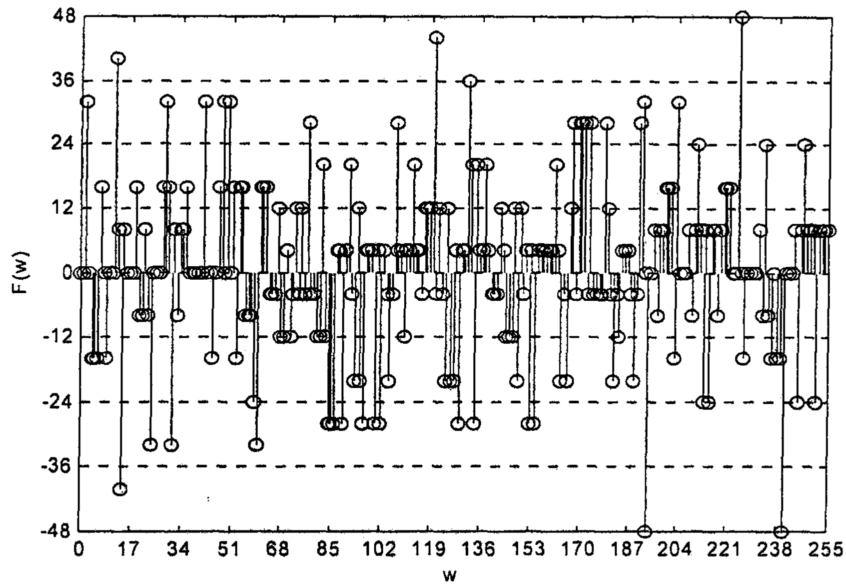

Figure D.20 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_4$ of Safer's logarithm taking s-box

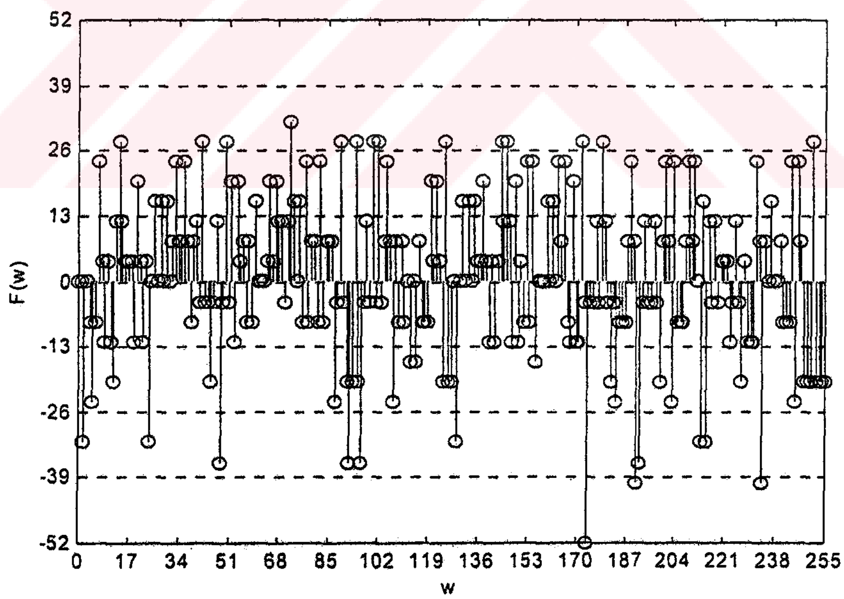

Figure D.21 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_5$ of Safer's logarithm taking s-box

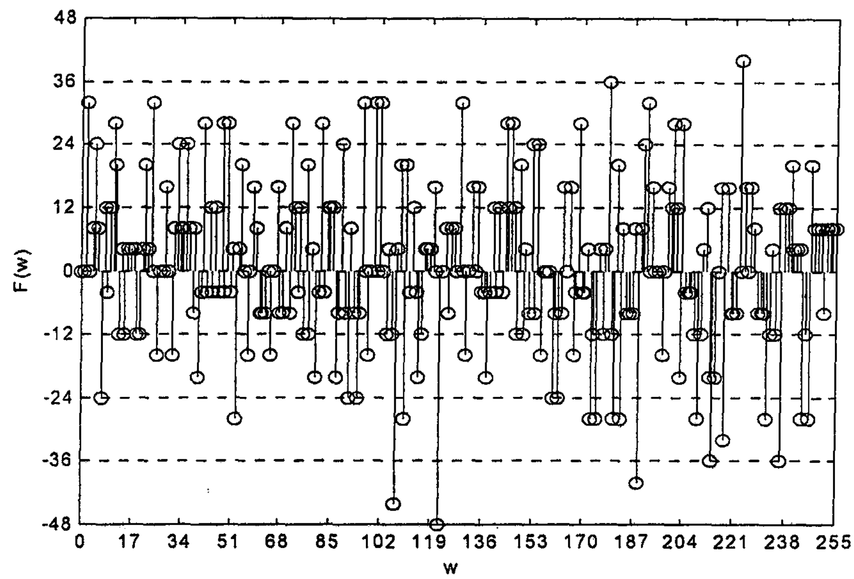

Figure D.22 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_6$ of Safer's logarithm taking s-box

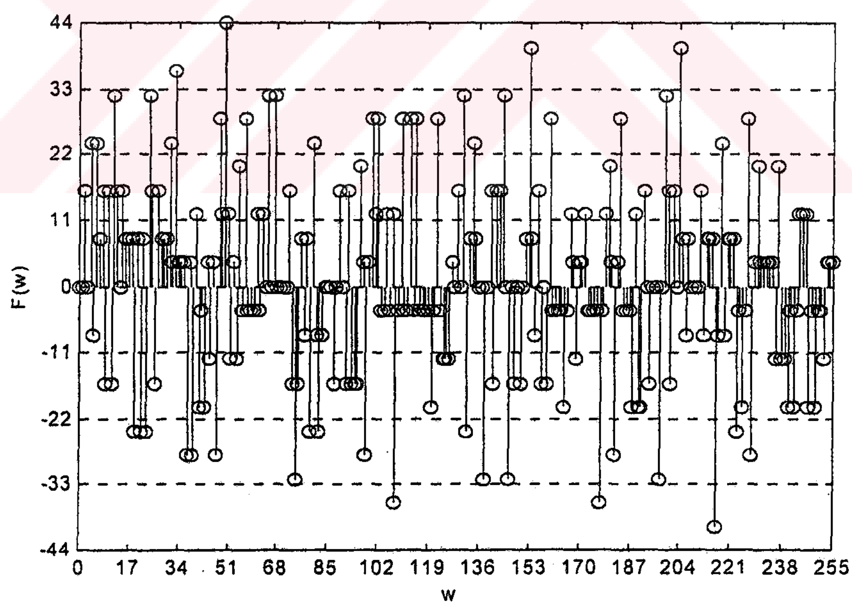

Figure D.23 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_7$ of Safer's logarithm taking s-box

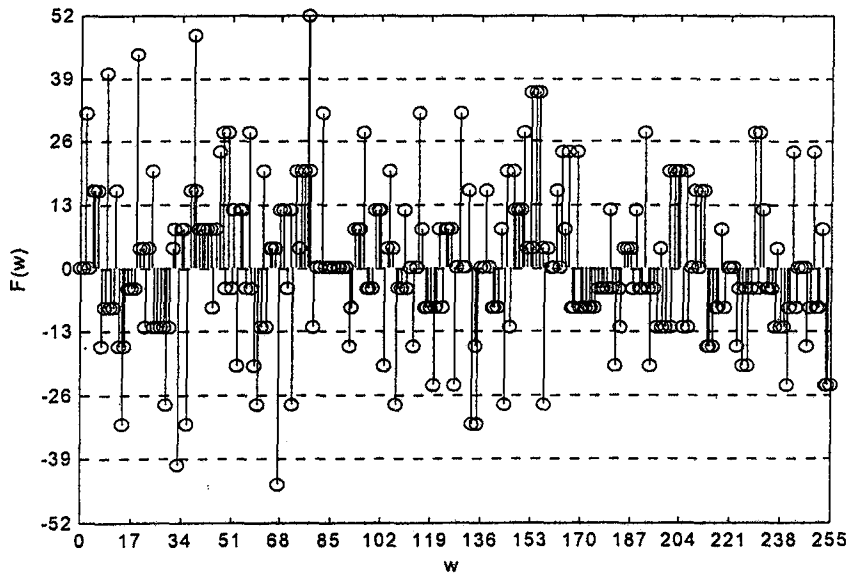

Figure D.24 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_8$ of Safer's logarithm taking s-box

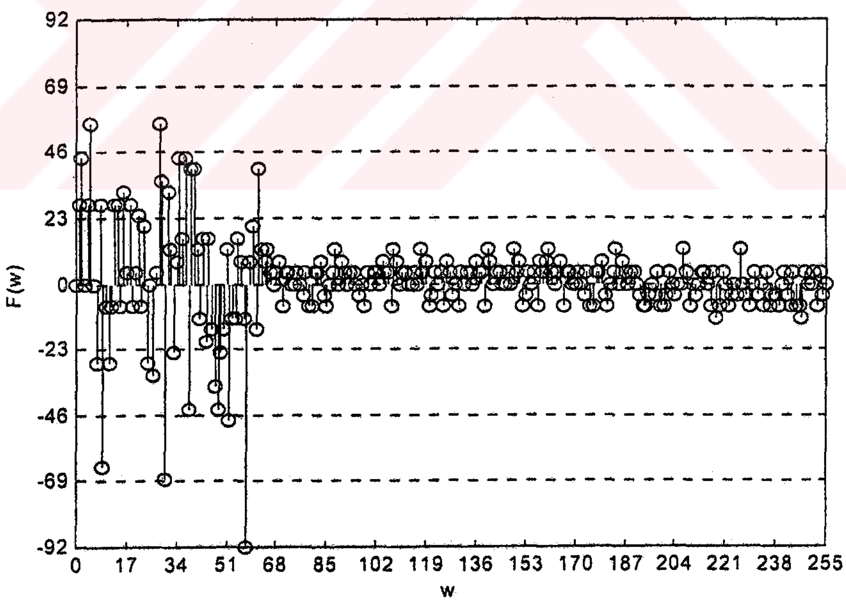

Figure D.25 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_3 \oplus y_4 \oplus y_7$ of Safer's exponentiating s-box

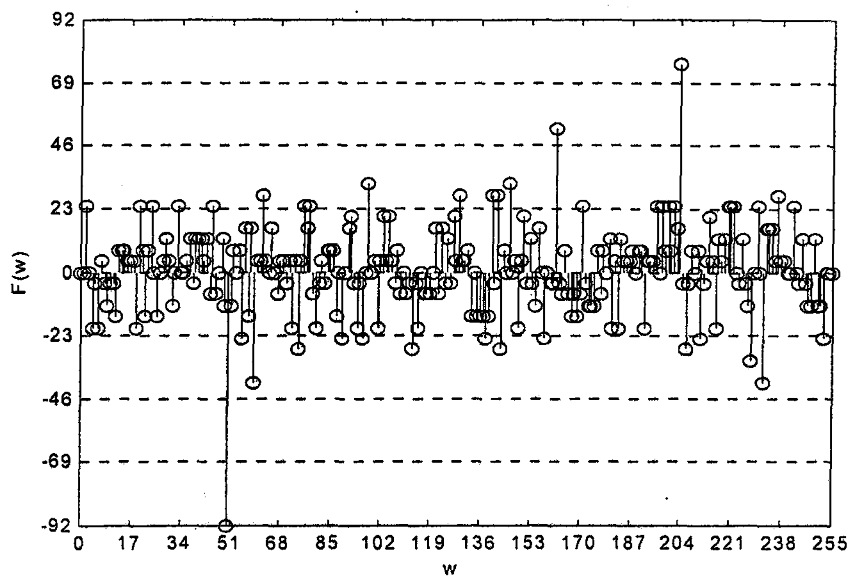

Figure D.26 Walsh-Hadamard Transform for the Boolean function $f(\mathbf{X}) = y_3 \oplus y_4 \oplus y_5 \oplus y_7$ of Safer's logarithm taking s-box

APPENDIX E

SIMULATION ASPECTS, TEST VALUES, AND AVAILABILITY OF SOURCE CODES

In this thesis, in order to implement the block ciphers and simulate the security tests, the programming languages C and Matlab are used. All security tests related to s-boxes can be efficiently simulated in Matlab. However, for the AWD criterion that is applied to the overall algorithm, the programming language C is preferred since it takes shorter time.

Although we write the programs independently, the block ciphers Rijndael* and Safer K-64 are among the most popular in the open literature, therefore their source codes and the test values can be easily obtained through internet. For Safer-K64, the source code in Pascal and the test values are also available [2]**.

Since the block cipher Spectr-H64 is recently introduced, it is not easy to reach the source code and/or the test values. Therefore the following test results, which are obtained through the authors' e-mail address***, are given in Table E.1. In the table, the test values are represented in hexadecimal form, and $K \in \{0, 1\}^{256}$, $P \in \{0, 1\}^{64}$, $C \in \{0, 1\}^{64}$, and P_1, P_2, C_1, C_2, K_1, K_2, ..., K_8, $\in \{0,1\}^{32}$.

* A number of implementations in various programming languages including C and Matlab, and the test values, i.e., the plaintext and the ciphertext values under a certain secret key, can be reached through the internet address "htpp://www.esat.kuleuven.ac.be/~rijmen/rijndael/".

** The conference paper [2] can be downloaded through "http://citeseer.nj.nec.com/322820.html".

*** spectr @ vicom.ru

Table E.1 Test Values for Spectr-H64

Secret-Key (K)	Plaintext (P)	Ciphertext (C)
K = (K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8) = (00000000, 00000000 00000000, 00000000 00000000, 00000000 00000000, 00000000)	P = (P_1, P_2) = (00000000, 00000000)	C = (C_1, C_2) = (DDBD8DDC, 94732BCD)
	P = (P_1, P_2) = (AAAAAAAA, AAAAAAAA)	C = (C_1, C_2) = (4D0392A2, 8D54EF5A)
	P = (P_1, P_2) = (55555555, 55555555)	C = (C_1, C_2) = (F2E4C199, EDAF181C)
K = (K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8) = (11121314, 21222324 31323334, 41424344 51525354, 61626364 71727374, 81828384)	P = (P_1, P_2) = (FBFCFDFD, F7F4F9F5)	C = (C_1, C_2) = (49A2EA0F, FBFCB5BC)