AN ALGORITHM TO ANALYZE SUBSTITUTION PERMUTATION NETWORK RESISTANCE TO LINEAR AND DIFFERENTIAL CRYPTANALYSIS

KASHIF ALI

AN ALGORITHM TO ANALYZE SUBSTITUTION PERMUTATION NETWORK RESISTANCE TO LINEAR AND DIFFERENTIAL CRYPTANALYSIS

By

KASHIF ALI

A thesis submitted to the

School of Graduate Studies

in partial fulfillment of the

requirements for the degree of

Master of Engineering

Faculty of Engineering and Applied Science

Memorial University of Newfoundland

2007

St. John's Newfoundland Canada

Abstract

In this thesis, we propose a practical novel algorithm, called the Two-Round Iterative (TRI) algorithm that analyzes the block cipher structure referred to as a Substitution Permutation Network (SPN). The algorithm characterizes the resistance of the cipher to linear cryptanalysis and differential cryptanalysis. By finding the best or close to best linear approximation and differential characteristics of the cipher, the algorithm can be used to find the number of plaintext/ciphertext pairs required to mount either attack on the cipher successfully. An important feature of the algorithm is that the complexity of algorithm is linear in terms of number of rounds and hence is able to give results in practical time.

In this thesis, the algorithm has been applied to 16-bit ciphers to verify the effectiveness of the algorithm in finding optimal linear biases and differential probabilities. Further, it is applied to realistic 64-bit ciphers based on 8×8 and 4×4 S-boxes that possess good cryptographic properties. In addition to the TRI algorithm, we have also developed two algorithms that are guaranteed to find the optimal linear approximation and differential characteristic and applied them to 16-bit ciphers in order to examine the TRI algorithm efficiency and effectiveness. It is shown that the TRI algorithm is effective in finding the best or close to the best linear approximation and differential characteristic and the corresponding linear bias and differential probability. Also the TRI algorithm can be practically applied to realistically sized ciphers (e.g. 64-bits) where the other algorithms are too inefficient to be practical. Experimental data is presented in the form of figures and tables which demonstrate the usefulness of the TRI algorithm in characterizing the security level of realistic SPN block ciphers.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Dr. H.M. Heys, my supervisor, for his timely guidance and direction during the course of my research and thesis writing. Dr. Heys' technical knowledge, caring attitude and able leadership have been a source of inspiration during my Master of Engineering program. I have a very high regard for his gentleness and kind attitude.

I would also like to thank Dr. R. Venkatesan, Associate Dean of Graduate Studies, for his moral support and guidance during my stay at the University.

My special thanks are reserved to my parents, Mr. Anwar Ali and Mrs. Kahkashan Anwar, and my two younger brothers, Atif Ali and Shahid Ali, who provided exceptional moral support during the course of my study.

I would also like to extend my gratitude to my uncle Mr. Haseen Khan and my aunt Mrs. Fatima Khan for being my guardian in Canada and providing me with all the resources that I needed for my stay and successful completion of my program. Their honest advice always helped me to improve myself professionally and personally.

Lastly, I would like to thank all of my friends and members of Computer Engineering Research Lab (CERL) for providing all the technical help that I needed during the program.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Cryptography is derived from a Greek word meaning "the science of hidden or secret writing" [1]. The use of mathematical functions in cryptography allows a cryptographer to develop encryption routines and digital signatures, which have a great use in computer security [1]. The encryption used in a cryptosystem is the practice of hiding messages so that they cannot be read by anyone else except the intended recipient. Thus encryption is a method of converting a plaintext message into its corresponding ciphertext message based on a key [2]. The method of encrypting and decrypting a message is called a cipher.

1.1 Characterization of a Cryptographic System

Cryptanalysis is the process of attacking and analyzing a cipher by the knowledge of ciphertext data or plaintext/ciphertext pairs and the nature of the algorithm used for the encryption and decryption [3] [4]. The main idea behind the cryptanalytic attack is to deduce the plaintext or the key being used in the cipher. If the attacker manages to find the right key then he will be able to deduce the encrypted message and can possibly even manipulate the data of some systems, thus posing a threat to information security.

A cryptographic system can be categorized in two ways [4]. Firstly, a cryptosystem can be categorized based on the number of keys used. If both the sender and the receiver use the same key for encryption and decryption, then the system is

referred to as a symmetric key or secret-key cryptosystem [1]. However if the sender and the receiver use different keys for encryption and decryption, then the system is called an asymmetric key or public key cryptosystem [1]. An asymmetric cipher uses a public key for encryption and a private (secret) key for decryption. Symmetric key encryption is much faster as compared to asymmetric key encryption; however, the problem with symmetric key is key exchange [1] [4].

Secondly, a cryptographic system can be classified according to the way the plaintext is processed. A block cipher processes an input block of bits to produce an output block of bits [5]. In contrast, a stream cipher will continuously process input bits to produce output bits one at a time [4]. The practicality of both these ciphers depends on the type of application for which the cipher is intended.

1.2 Symmetric-key and Asymmetric-key Algorithms

A symmetric-key cipher is an algorithm that uses the same cryptographic keys for encryption and decryption [2]. Both the sender and the receiver share the secret key over a private information channel. The major advantage of a symmetric-key algorithm over an asymmetric-key algorithm is that it is much less computationally intensive [1]. However, the shared secret key needs to be kept secure during distribution. Due to this reason, symmetric-key algorithms are often not preferred for authentication and authorization purposes [1]. A symmetric-key cryptosystem is shown in Figure 1.1 where M is plaintext, C is ciphertext and K is the cipher key.

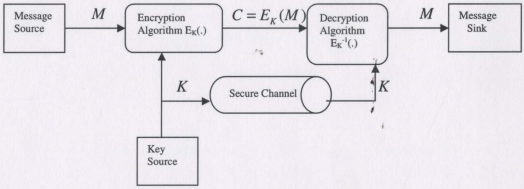

Figure 1.1: Symmetric-key Cryptosystem

Contrary to a symmetric-key algorithm, an asymmetric-key algorithm is an algorithm that allows a user to communicate securely without having prior access to a shared secret key. This is done by using a pair of cryptographic keys, designated as public key and private key, which are related mathematically. In an asymmetric-key algorithm, the private key is kept secret, while the public key may be widely distributed [2]. Hence the sender encrypts the information by using the public key of the receiver and the receiver decrypts the information using its private key. Asymmetric-key algorithms are widely used for authentication purposes during a communication between two parties. Besides this, it is also used in digital signatures and during the key exchange to establish a key for symmetric-key encryption during a communication. Some of the popular asymmetric-key algorithms are RSA and Digital Signature Standard (DSS) cryptosystem [4]. An asymmetric-key cryptosystem is shown in Figure 1.2 where M is plaintext, C is ciphertext, K_S is private key and K_P is public key.

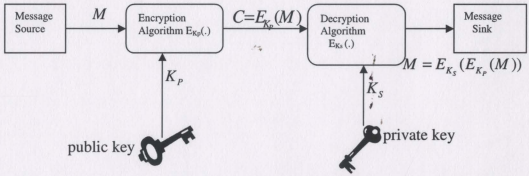

Figure 1.2: Asymmetric-key Cryptosystem

1.3 Block Ciphers and Stream Ciphers

In cryptography, a block cipher is a cipher which operates on fixed-length groups of bits, termed *blocks*, with an unvarying keyed transformation [4]. When encrypting, a block cipher takes an *N*-bit block of plaintext as input and outputs a corresponding *N*-bit block of ciphertext. The exact transformation is controlled by a second input called the cipher key. The decryption process is similar to the encryption process. The decryption algorithm takes an *N*-bit block of ciphertext together with the key bits, and yields the original *N*-bit block of plaintext.

A block cipher consists of two paired algorithms, one for encryption, *E*, and another for decryption, E^{-1}. Both algorithms accept two inputs: an input block of size *N* bits and a key of size *k* bits, yielding an *N*-bit output block. For any one fixed key, decryption is the inverse function of encryption, and is represented by equation (1.1)

$$E_K^{-1}(E_K(M)) = M \tag{1.1}$$

for any block of input M and key K. Many block ciphers can be categorized as Feistel networks [4], or as more general Substitution Permutation Networks (SPNs) [6]. Arithmetic operations, logical operations (especially XOR), S-boxes and various permutations are all frequently used as components in block ciphers. Some of the most widely used block ciphers are Data Encryption Standard (DES) [2] [4], Triple DES [4], and Advanced Encryption Standard (AES) [4].

On the other hand, a stream cipher is a cipher in which the plaintext is encrypted one bit at a time, and in which the transformation of successive bits varies during the encryption. Typically, ciphertext bits are generated by XORing plaintext bits with a pseudo random sequence of bits referred to as key stream [2] [4]. The encryption of each bit is dependent on the current state of the key stream generator. Stream ciphers are often used in applications where plaintext comes in quantities of unknowable length, for example, a secure wireless connection. If a block cipher were to be used in this type of application, the designer would need to choose either transmission efficiency or implementation complexity, since block ciphers cannot directly work on blocks shorter than their block size. The major advantage of a stream cipher is that the stream cipher algorithms are typically faster than the block cipher algorithms [2]. Also, the stream cipher has a low error propagation rate. However, the stream cipher requires synchronization between the transmitter and the receiver end because the same random stream should be available at the sender and the receiver. Hence the need for the synchronization limits its usefulness in low bandwidth connections. One of the widely used stream ciphers in software is RC4 [4].

1.4 Motivation

Linear cryptanalysis and differential cryptanalysis are two of the most fundamental cryptanalytic attacks on symmetric-key block ciphers. Linear cryptanalysis was first introduced by Matsui and was successfully applied to cryptanalyze the Data Encryption Standard (DES) cipher by using linear approximations [7] [8]. On the other hand, differential cryptanalysis was introduced by Biham and Shamir, and was used to cryptanalyze DES by using differential characteristics [9]. Even though both the attacks were initially targeted at DES, they can be and are used to characterize the security level of all block ciphers. One of the common block cipher structures is referred as a Substitution Permutation Network (SPN). It is used in many current modern day ciphers like DES and the Advanced Encryption Standard (AES). Hence, it is of interest to characterize the applicability of linear cryptanalysis and differential cryptanalysis to the SPN structure.

Our work is motivated by Matsui's algorithm on DES, where he developed a tool to linearly and differentially cryptanalyze DES ciphers. In our work we wanted to develop a similar tool to apply to SPNs, such that it is efficient and practical to apply to realistically sized ciphers.

1.5 Objective of the Thesis

The primary objective of the thesis is to develop an automated tool to analyze the resistance of SPNs to linear and differential cryptanalysis. To this aim, we have developed a practical algorithm called the Two-Round Iterative (TRI) algorithm that tries to find the best linear approximation and differential characteristic of the SPN cipher in

an automated way. By finding the best or close to best linear approximation and differential characteristic of the cipher, the algorithm can be used to find the number of plaintext/ciphertext pairs required to mount either attack (linear/differential) on the cipher successfully. An important feature of the TRI algorithm is that the complexity of algorithm is linear in terms of number of rounds and hence is able to give results in practical time for realistically-sized cipher. We have shown in our work that the TRI algorithm can be practically applied to realistically-sized ciphers (e.g. 64-bit block ciphers) where other algorithms are too inefficient. Hence, the TRI algorithm is a useful tool to examine the properties of SPN ciphers including S-boxes and the permutation structure that are necessary for good cryptographic resistance to linear and differential cryptanalysis.

1.6 Outline of the Thesis

The outline of the thesis is listed below:

In Chapter 2, we will study three different types of practically realizable SPN ciphers; one is a 16-bit SPN cipher using 4×4 S-boxes and the other two are 64-bit cipher using 4×4 S-boxes and 8×8 S-boxes. A discussion is given along with an example to attack the SPN cipher linearly and differentially.

Chapter 3 deals with a number of algorithms that have been studied and developed related to linear cryptanalysis and differential cryptanalysis. These algorithms find the best linear bias or differential probability for an SPN block cipher and a corresponding linear approximation or differential characteristic. Some preliminary results are also given in this chapter which shows our algorithm is a useful tool to look

into properties of S-boxes and cipher structures that are necessary for good cryptographic resistance to linear and differential cryptanalysis.

In Chapter 4, detailed discussions given on the results obtained using our efficient non-optimal algorithm and the optimal algorithm (in terms of largest bias or differential probability). These results are shown in the form of tables and figures, and comparisons are made between the optimal solution and the non-optimal solution where necessary.

In Chapter 5, concluding remarks are made related to our work and some limitations related to our algorithm are noted. The limitations of our algorithm can be considered for the future research work.

Chapter 2

Background Review of SPNs

Feistel proposed the use of substitutions and permutations in implementing a strong cipher in 1973 [10]. He was inspired by Claude Shannon's work that introduced alternating confusion and diffusion functions in implementing a powerful product cipher [11]. The methodology of diffusion is to make a complex statistical relationship between plaintext and ciphertext so that it prevents the cryptanalyst from deducing any key. This is achieved by having each plaintext bit to affect all ciphertext bits. However, even after diffusion, if the attacker manages to form some statistical relationship with the knowledge of plaintext and ciphertext bits, then the concept of confusion implemented in the cipher structure thwarts the discovery of key bits [4]. The idea of confusion is to make an intricate relationship between ciphertexts and the encryption key bits so that the deduction of key bits is not viable. This is typically achieved by implementing a complex nonlinear substitution in the cipher structure. The SPNs that we will study in the thesis have the characteristics suggested by Shannon and Feistel. We are analyzing SPNs because they are still widely used in today's modern cipher design as in DES and AES [4].

In this chapter, we describe SPNs and examine the applicability of linear and differential cryptanalysis to SPN structure.

2.1 Description of SPNs

A basic SPN is a symmetric-key block cipher structure [12]. Each round of cipher structure consists of a substitution block, a permutation block and key mixing. The N-bit plaintext after processing through R rounds gives the N-bit ciphertext. In the substitution stage, an N-bit block is divided into m-bit sub-blocks. Each of the m-bit sub-blocks is fed into a bijective $m \times m$ substitution box (S-box). An S-box is a mapping $\{0,1\}^m \rightarrow \{0,1\}^m$ [13]. A substitution block is immediately followed by a permutation block where bit-wise transposition takes place. In general, the permutation is an invertible linear transformation [14] [15]. All rounds in an SPN are alike except the last round where the permutation is not done. A good SPN cipher should incorporate a sufficient number of rounds R and a good design of S-boxes (A good S-box have mathematical properties that make the cipher hard to analyze). Cryptanalysis will be difficult if the number of rounds in the cipher is large, even if the S-box design is relatively weak (A weak S-box have mathematical properties that make the cipher easy to analyze). Similarly, if the S-box design is strong then a more secure cipher can be obtained for lesser number of rounds. Larger S-boxes are more resistant to linear and differential cryptanalysis but require large lookup tables to implement. Another disadvantage of large S-boxes is the resulting complexity in hardware implementation. Hence a limit of m is usually 8 to 10 for practical implementation [16]. A sample 4×4 S-box and a permutation mapping are shown in Table 2.1 and Table 2.2 respectively. The S-box mapping shown in Table 2.1 is taken from the DES cipher [4]. In this table, the leftmost bit is represented as the most significant bit in the hexadecimal notation. In Table 2.2, the number represents the bit position in the block.

The key mixing in an SPN is achieved by using bitwise XOR between the key bits associated with a round (referred to as subkey bits) and the input data block to a round. There are $R+1$ subkeys that are obtained by running a key scheduling algorithm with the cipher key as an input. However in our studies we will not consider a key scheduling algorithm to generate subkeys. Instead, all bits of the subkeys are independently generated and are unrelated to each other. This assumption is consistent with the presentation of linear and differential cryptanalysis.

Input	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Output	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

Table 2.1: S-box Representation (in hexadecimal)

Input	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Output	1	5	9	13	2	6	10	14	3	7	11	15	4	8	12	16

Table 2.2: Permutation of Bits

For an SPN, decryption is similar to encryption and can be viewed as simply having the data run backward through the network. The S-box and the permutation mapping in the decryption network are the inverse of the encryption mapping. The subkeys used for the decryption are the same as encryption except they are applied in reverse order. Since no permutation is applied in the last round, the encryption and

decryption network look alike. In the following section we will look at the structure of SPNs that are discussed in the thesis.

2.2 Specific Ciphers Studied in Thesis

In the thesis we will study three types of SPNs:

- 16-bit SPN using 4×4 S-boxes (Cipher-A)
- 64-bit SPN using 8×8 S-boxes (Cipher-B)
- 64-bit SPN using 4×4 S-boxes (Cipher-C)

2.2.1 16-bit SPN using 4×4 S-boxes (Cipher-A)

A 16-bit SPN structure is shown in Figure 2.1 [16]. From the Figure 2.1, we can see that the 16-bit plaintext after processing through R rounds of the cipher gives the 16-bit ciphertext. In the figure R is assumed to be 4. Throughout the thesis we will use one mapping for all S-boxes in all the rounds of cipher. Advantages of the use of one S-box mapping are the ease of study, consistency with the AES structure and a compact hardware and software implementation. In this work some good S-boxes are used during the simulations that have good cryptographic properties like the DES 4×4 S-boxes. The permutation is straightforward as the output i of S-box j is connected to input j of S-box i as shown in Table 2.2.

The subkey layer simply represents the bit-by-bit XOR of the subkey and the data block. Thus cipher structure is not a realistically sized cipher. With only a 16-bit block, the cipher would be susceptible to several security flaws. However, this structure is a

useful one to consider when examining the practicality of cipher analysis and design techniques.

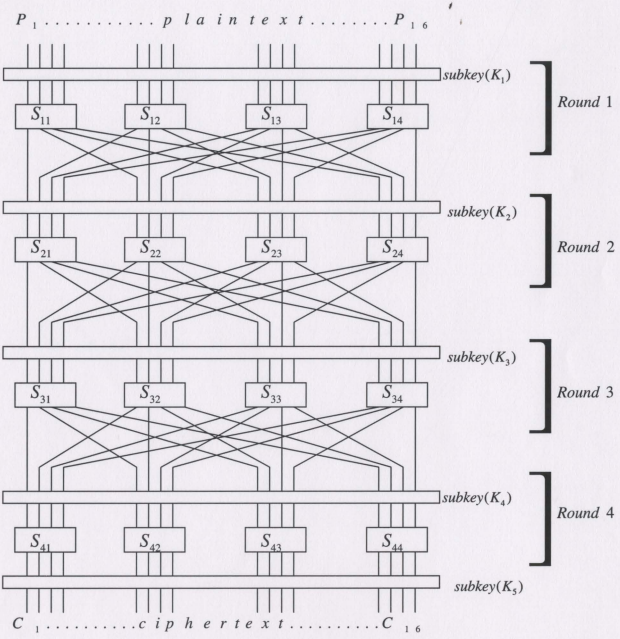

Figure 2.1: 16-bit SPN using 4×4 S-boxes (Cipher-A)

2.2.2 64-bit SPN using 8×8 S-boxes (Cipher-B)

A 64-bit SPN structure is shown in Figure 2.2 [13] with the number of rounds given as R=3. The 64 input bits are divided into eight sub-blocks and run into eight S-boxes as shown in Figure 2.2. This structure is an extension of the structure discussed in Section 2.2.1. This is a more realistic cipher structure as compared to 16-bit SPN using 4×4 S-boxes and could be used as the basis for a secure block cipher. A 64-bit round key is mixed in each round which can be obtained by running the key scheduling algorithm [4] or generating the bits independently. The transposition of bits and subkey layer in the cipher structure are the same as discussed in Section 2.2.1.

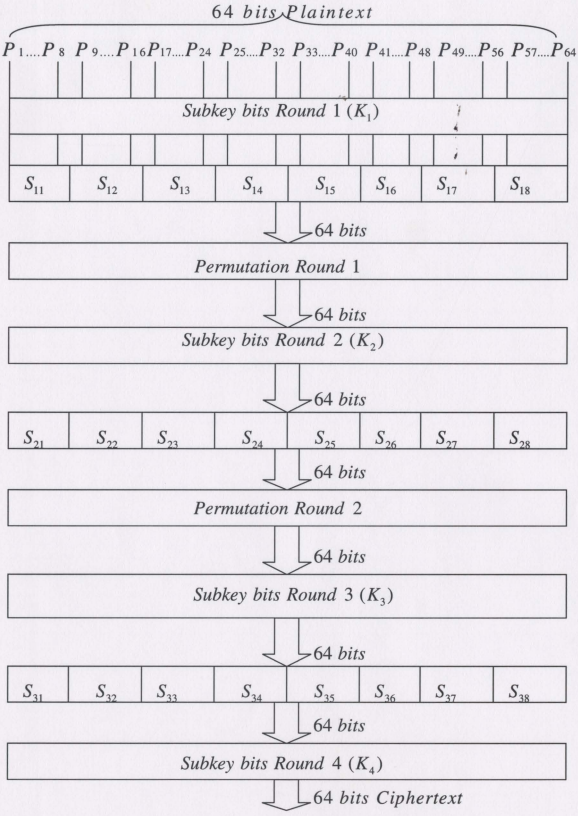

Figure 2.2: 64-bit SPN using 8×8 S-boxes (Cipher-B)

2.2.3 64-bit SPN using 4×4 S-boxes (Cipher-C)

The third SPN under consideration is a type of SPN architecture called Extended Tree-structured SPN (TS-SPN) [14]. The advantage of using an Extended TS-SPN that, while providing good diffusion properties, it provides a single permutation for all the rounds in the cipher and can be used for any number of rounds for a given value of N and m where N is the number of plaintext / ciphertext bits and m is the number of input / output bits from an S-box. Hence the hardware implementation of Extended TS-SPN is easy as compared to a basic TS-SPN [14] which requires different permutations in each round. In our thesis, we will consider an Extended TS-SPN that has m=4 (i.e. uses 4×4 S-boxes) and N=64 as shown in Figure 2.3. The permutation of bits for m=4 and N=64 is shown in Table 2.3.

Input	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Output	1	5	9	13	17	21	25	29	33	37	41	45	49	53	57	61
Input	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Output	2	6	10	14	18	22	26	30	34	38	42	46	50	54	58	62
Input	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
Output	3	7	11	15	19	23	27	31	35	39	43	47	51	55	59	63
Input	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
Output	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64

Table 2.3: Permutation of Bits for Cipher-C

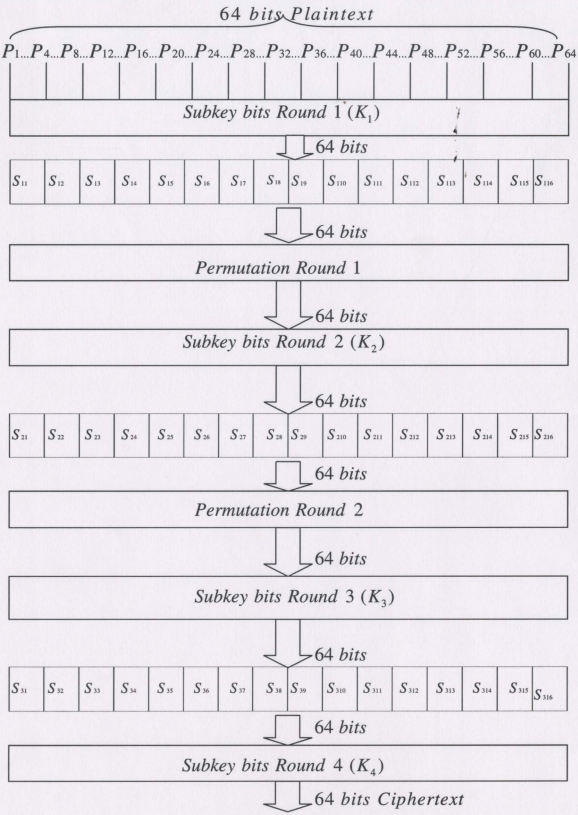

Figure 2.3: 64-bits SPN using 4×4 S-boxes (Cipher-C)

2.3 Advanced Encryption Standard (AES)

The AES is a block cipher which was proposed by Rijmen and Daemen [17] [18] and adopted by National Institute of Standards and Technology (NIST) as a US government standard [18]. Subsequently, it has been widely used in providing data security in a large variety of applications. The cipher can have different block lengths and key lengths of 128, 192 or 256 bits. Some of the important characteristics of AES are listed below [4]:

- It provides resistance against all known attacks.
- It offers speed and code compactness on a wide range of platforms.
- The design of the cipher is simple.

The structure of AES incorporates substitution and permutation blocks in each round. Just like basic SPNs it processes the entire data block in parallel during each round of substitution and permutation. However, while similar to a basic SPN, the permutation layer which is essentially a linear transformation (using XOR of bits) of the bits in the data block [4]. The linear transformation layer is computed of two components: "shift rows" (essentially a permutation of bytes within the block) and "mix column" which computes a mixing of bits by computing over Galois field GF (2^8). The block diagram for one round encryption of AES cipher is shown in Figure 2.4.

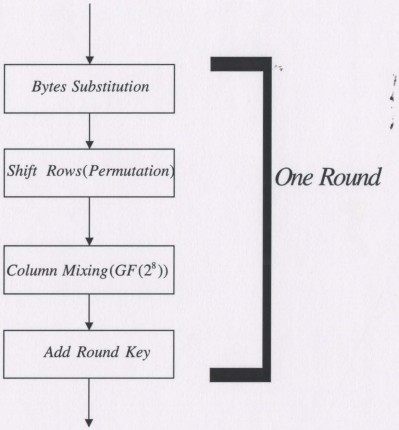

Figure 2.4: Block Diagram for One Round Encryption of AES [4]

2.4 Linear Cryptanalysis

In many real world applications the attacker of a cipher can possess information on a random set of plaintexts and ciphertexts. Linear cryptanalysis provides a mechanism to obtain subkey bits from the knowledge of plaintext and ciphertext bits [7] [19]. Linear cryptanalysis is known as an effective attack on various block cipher structures. The attack is more efficient than exhaustively searching the subkey bits. It is considered to be a known plaintext attack because the attacker has the knowledge of the set of plaintexts and the corresponding ciphertexts [16] [20].

The basic principle of linear cryptanalysis is to analyze the S-boxes and extend the properties of the S-boxes to the entire cipher structure [7]. The idea is to find a statistically tendency towards a linear relation between a portion of plaintext bits and ciphertext bits, where linearity refers to bitwise XOR operation between bits as shown in Equation (2.1):

$$X_{i1} \oplus X_{i2} \oplus ... X_{iu} \oplus Y_{j1} \oplus Y_{j2} \oplus ... \oplus Y_{jv} = 0 \tag{2.1}$$

where X_i represents the i^{th} bit of the input $X = [X_1, X_2 ...]$ and Y_j represents the j^{th} bit of the output $Y = [Y_1, Y_2 ...]$. The Equation (2.1) shows the bitwise XOR relationship between the u input bits and v output bits.

In the Equation (2.1), if we randomly choose u and v independent bits for which probability of zero is equal to 1/2 and place them in Equation then the probability that expression holds true is 1/2. Linear cryptanalyst exploits the likelihood of Equation (2.1) to hold true to deviate from probability of 1/2. The deviation of value can be towards zero or towards one. The amount by which the probability of linear expression deviates from 1/2 is referred as linear probability bias. Hence if the expression holds with the probability p_L, then the linear probability bias is calculated as $\varepsilon = p_L - 1/2$. The weakness of the cipher depends upon the value of p_L. If p_L=1, then expression in Equation (2.1) exactly signifies linear behavior and if p_L=0 then the relationship is affine in the cipher. In both the cases, it symbolizes weak cipher characteristics. Thus, the higher the magnitude of ε, the lesser the number of plaintext and ciphertext pairs is required, and hence the easier will be the linear attack.

When setting up a linear attack the Equation (2.1) can also be written to incorporate key bits in the form of Equation (2.2).

$$X_{i1} \oplus X_{i2} \oplus ... X_{iu} \oplus Y_{j1} \oplus Y_{j2} \oplus ... \oplus Y_{jv} = K_{l1} \oplus K_{l2} \oplus ... K_{lw} \tag{2.2}$$

where K_l is the l^{th} bit of the key $K = [K_1, K_2 ...]$ and K is the complete set of subkey bits. The notation w is the total number of key bits involved during the approximation. During the attack, the key bits K are fixed but unknown.

If the right hand side of the equation is equal to zero, then ε will have the same sign (+ or -) as the bias of the expression involving the subkey sum and, if the right hand side of equation is equal to one then ε will have the opposite sign.

The piling-up lemma, as proposed by Matsui [16], is used to calculate the probability of n independent random binary variables, $X_1, X_2, ..., X_n$, summing to 0, and is given by Equation (2.3):

$$Prob(X_1 \oplus X_2 \oplus ... X_n = 0) = 1/2 + 2^{n-1} \prod_{i=1}^{n} \varepsilon_i \tag{2.3}$$

The Equation (2.3) can be represented in form of Equation (2.4):

$$Prob(X_1 \oplus X_2 \oplus ... X_n = 0) = 1/2 + \varepsilon_{1,2,3,...n} \tag{2.4}$$

where $\varepsilon_{1,2,3,...n}$ symbolizes the bias of $X_1 \oplus X_2 \oplus ... X_n = 0$.

2.4.1 Construction of Linear Approximation Table or Bias-table

In this section, we will look into the creation of the linear approximation table or bias-table of S-boxes. In the SPN structure, the only non-linearity is due to the presence of S-boxes. So a linear approximation is made using Equation (2.1) for the non-linearity introduced by the S-boxes in a cipher. We will explain the characteristics of S-boxes in

the context of the16-bit SPN using 4×4 S-boxes. A 4×4 S-box has an S-box mapping similar to the one shown in Table 2.4. The input vector $X=[X_1,X_2,X_3,X_4]$ and the output vector $Y=[Y_1,Y_2,Y_3,Y_4]$ takes part in the linear approximation. The S-box that we used to illustrate the construction of bias-table is from the second row of the S-box $S_1[4]$.

Input	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Output	0	F	7	4	E	2	D	1	A	6	C	B	9	5	3	8

Table 2.4: S-box Representation (in hexadecimal)

Consider the linear expression $X_4 \oplus Y_1 \oplus Y_2 = 0$. It can be inferred from Table 2.5 that expression $X_4 \oplus Y_1 \oplus Y_2 = 0$ is true for exactly 10 out of maximum 16 possible cases, where each corresponds to one of the possible values for X. Hence the probability bias ε is equal to $\frac{10}{16}-\frac{1}{2}=\frac{2}{16}$.

Similarly, consider the linear expression $X_1 \oplus X_2 \oplus X_3 \oplus X_4 \oplus Y_4 = 0$. The probability bias ε for this expression is equal to $\frac{12}{16}-\frac{1}{2}=\frac{4}{16}$. The affine relationship can be seen from the expression $X_2 \oplus Y_2 \oplus Y_3 \oplus Y_4 = 0$. The value computed for ε is equal to $\frac{6}{16}-\frac{1}{2}=\frac{-2}{16}$. For the linear cryptanalysis the magnitude of ε plays a pivotal role in the complexity of attacking a cipher; hence, the affine relationship is equally used to get the best result.

A complete list of all the linear approximations of the S-box can be viewed in Table 2.6. The row indicates the input sum or masking input bits and is denoted by ΓX. Similarly, the column indicates the output sum or masking output bits and is denoted by ΓY. Each element in the Table 2.6 is the decimal numerator value for every possible ε value, such that ε is given by table value divided by 16.

We can deduce some important properties from the bias-table [16]. Firstly, the values in the bias-table will always be an even number. Secondly, the probability that any sum of a non-empty subset of input bits is equal to the sum involving no output bits is 1/2. Thirdly, the bias value for the linear combination involving no input bits and no output bits is 1/2. The resulting value 8 can be seen in the first row and first column of the bias-table. It can also be noted from the bias-table that the sum of any row or column is either +8 or -8. Similar concepts can be considered for the bias-table of an 8×8 S-box where the table contains 256 rows and 256 columns and the bias is given by the table value divided by 256.

X_1	X_2	X_3	X_4	Y_1	Y_2	Y_3	Y_4	$Y_1 \oplus Y_2$	$X_1 \oplus X_2 \oplus X_3 \oplus X_4$	$Y_2 \oplus Y_3 \oplus Y_4$
0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1	0	1	1
0	0	1	0	0	1	1	1	1	1	1
0	0	1	1	0	1	0	0	1	0	1
0	1	0	0	1	1	1	0	0	1	0
0	1	0	1	0	0	1	0	0	0	1
0	1	1	0	1	1	0	1	0	0	0
0	1	1	1	0	0	0	1	0	1	1
1	0	0	0	1	0	1	0	1	1	1
1	0	1	0	0	1	1	0	1	0	0
1	0	1	1	1	1	0	0	0	0	1
1	1	0	0	1	0	0	1	1	0	1
1	1	0	1	0	1	0	1	1	1	0
1	1	1	0	0	0	1	1	0	1	0
1	1	1	1	1	0	0	0	1	0	0

Table 2.5: Sample Linear Approximation of S-box

(ΓX,ΓY)	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	-2	-2	-2	6	2	2	2	2
2	0	2	-2	-4	0	2	2	0	0	2	2	0	0	2	-2	4
3	0	2	2	0	4	-2	2	0	-2	0	4	2	-2	0	0	-2
4	0	2	-2	4	-2	0	0	-2	0	2	2	0	-2	0	4	2
5	0	2	2	0	2	4	0	-2	2	-4	0	-2	0	2	2	0
6	0	0	0	0	2	2	2	2	0	0	0	0	2	-6	2	2
7	0	0	0	0	2	2	-6	2	2	2	2	2	0	0	0	0
8	0	0	0	0	-2	2	2	-2	2	2	2	2	4	0	0	-4
9	0	0	0	0	-2	2	2	6	0	0	0	0	-2	2	2	-2
A	0	2	-2	4	2	0	0	2	-2	0	0	-2	4	2	-2	0
B	0	2	2	0	-2	-4	0	2	4	-2	2	0	2	0	0	2
C	0	-2	2	4	0	2	2	0	2	0	0	2	-2	0	-4	2
D	0	-2	-2	0	4	-2	2	0	4	2	-2	0	0	2	2	0
E	0	-4	4	0	0	0	0	0	-2	2	2	-2	2	2	2	2
F	0	4	4	0	0	0	0	0	0	4	-4	0	0	0	0	0

Table 2.6: Linear Approximation Table or Bias-table

2.4.2 Linear Approximation of the Overall Cipher Structure

Once the bias-table is constructed for the S-boxes, we can find the linear approximation of the whole cipher structure by using Equation (2.1). Since we assume that each S-box is independently related to each other in the cipher structure, the overall bias value is found by concatenating the linear approximation of each S-box. Using Equation (2.1), we find an expression between plaintext bits and the second last round output bits. A subset of the subkey bits that follow the last round are obtained by using the linear expression that constitute the plaintext bits and second last round output bits.

The attack on the cipher structure is explained with reference to the 16-bit SPN using 4×4 S-boxes where each S-box in the SPN is assumed to be the S-box of Table 2.4. In Figure 2.5, a sample attack is illustrated. From Figure 2.5 we can see the active S-boxes and the corresponding ΓX and ΓY values for each round. The approximation is made on the S-boxes involved that are referred as active S-boxes, as shown in Table 2.7.

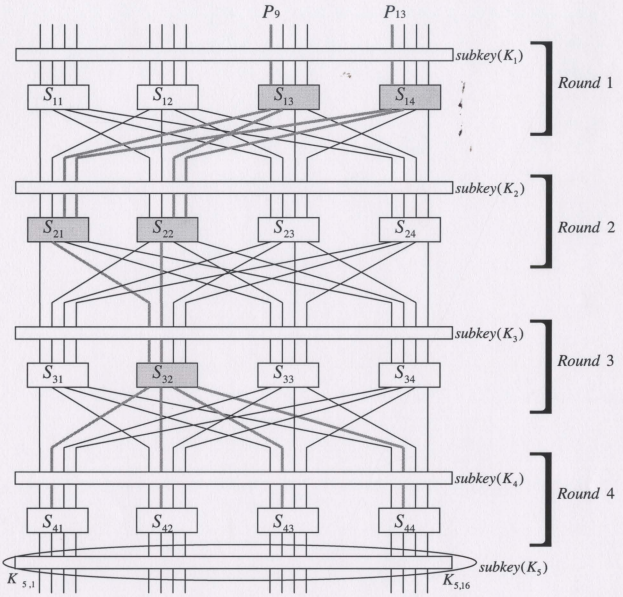

Figure 2.5: Linear Cryptanalysis of Cipher-A

ACTIVE S-BOX $S_{i,j}$: LINEAR EXPRESSION	LINEAR PROBABILITY: p_L	LINEAR BIAS : ε
$S_{13} : X_1 = Y_1 \oplus Y_2$	12/16	4/16
$S_{14} : X_1 = Y_1 \oplus Y_2$	12/16	4/16
$S_{21} : X_3 \oplus X_4 = Y_2$	12/16	4/16
$S_{22} : X_3 \oplus X_4 = Y_2$	12/16	4/16
$S_{32} : X_1 \oplus X_2 = Y_1 \oplus Y_2 \oplus Y_3 \oplus Y_4$	10/16	2/16

Table 2.7: Approximation Equations for Active S-boxes

Let U_i represent the 16-bit block of bits for the input, V_i represent the 16-bit block of bits for the output of the round i S-boxes and $U_{i,j}$ and $V_{i,j}$ represents the j^{th} bit of block U_i and V_i respectively. The bits are numbered from left to right from 1 to 16. Similarly, let K_i represent the subkey block of bits XORed at the input to round i. The linear approximation expression of active S-boxes involved for each round is shown in Table 2.8.

ROUND NUMBER: n	LINEAR EXPRESSION
1	$U_{1,9} = V_{1,9} \oplus V_{4,10}$ $U_{1,13} = V_{1,13} \oplus V_{1,14}$
2	$V_{2,2} = U_{2,3} \oplus U_{2,4}$, $U_{2,3} = V_{1,9} \oplus K_{2,3}$ $V_{2,6} = U_{2,7} \oplus U_{2,8}$, $U_{2,4} = V_{1,13} \oplus K_{2,4}$ $U_{2,7} = V_{1,10} \oplus K_{2,7}$, $U_{2,8} = V_{1,14} \oplus K_{2,8}$
3	$U_{3,5} \oplus U_{3,6} = V_{3,5} \oplus V_{3,6} \oplus V_{3,7} \oplus V_{3,8}$ $U_{3,5} = V_{2,2} \oplus K_{3,5}$, $U_{3,6} = V_{2,6} \oplus K_{3,6}$
4	$U_{4,2} = V_{3,5} \oplus K_{4,2}$, $U_{4,6} = V_{3,6} \oplus K_{4,6}$ $U_{4,10} = V_{3,7} \oplus K_{4,10}$, $U_{4,14} = V_{3,8} \oplus K_{4,14}$

Table 2.8: Linear Expression for Each Round

The plaintext bits P_i are related to input bits of round one with the Equation $U_{1,9} = P_9 \oplus K_{1,9}$ and $U_{1,13} = P_{13} \oplus K_{1,13}$. Solving the linear expression defined in Table 2.8, the final expression obtained involves the portion of the plaintext bits, the second last round output bits and the subkey bits, and is given by

$$P_9 \oplus P_{13} \oplus U_{4,2} \oplus U_{4,6} \oplus U_{4,10} \oplus U_{4,14} = \sum_{\oplus} K \tag{2.5}$$

where $\sum_{\oplus} K$ is the XOR sum of the subkey bits involved in the approximations of Table 2.8. The bias value for the whole cipher structure is calculated using the piling-up lemma and is equal to

$$\varepsilon = 2^4 \times (\frac{4}{16})^4 \times \frac{2}{16} = \frac{1}{128}.$$

The value $\sum_{\oplus} K$ is fixed and can be either 0 or 1. Thus Equation (2.5) represents the form of Equation (2.1) if $\sum_{\oplus} K = 0$. If $\sum_{\oplus} K = 1$ then we will have affine relationship and the bias value of the cipher structure will be equal to $-\frac{1}{128}$. Since in linear cryptanalysis the magnitude of bias plays the important role in determining the last round subkey bits and hence finding the number of known plaintext and ciphertext pairs for the attack, the value of $\sum_{\oplus} K$ doesn't affect the end result. The number of plaintext and ciphertext pairs that are required during the attack is roughly equal to ε^{-2} as suggested by Matsui [7] [16].

Once an R-1 round linear approximation is determined with high probability bias, the last round partial subkey bits are recovered by partially decrypting the last round of cipher for all possible values for the subkey bits and for all plaintext ciphertext pairs. In our example, the target partial last round subkey bits are $K_{5,1}$ to $K_{5,16}$. A counter variable is incremented for each possible value for the target subkey, each time the linear expression in Equation (2.5) holds true. The linear expression is tested for a large number of plaintext and ciphertext pairs. The correct partial subkey will have counter variable value that differs maximum from half the number of plaintext and ciphertext pairs. The result will be practically correct because the correct partial subkey will result in the linear

approximation holding true with large bias (similar to the calculated ε). An incorrect key will be equivalent to a random guess and has probability equal to ½ of satisfying the linear expression. Hence the correct partial subkey bits will result in a bias given by

$$|\varepsilon_{prac}| = |count - N_{PC}/2|/N_{PC} \quad (2.6)$$

where ε_{prac} is the practical bias value obtained by running the test on the obtained plaintext / ciphertext data, *count* is the counter variable that keeps the count when linear expression is true and N_{PC} is the number of plaintext and ciphertext pairs. The value of ε_{prac} should be close to the theoretical bias of $\varepsilon = \frac{1}{128}$.

In the example above, all key bits in the last round are obtained at once by the approximation. But the approximation illustrated does not give a large bias value. Hence another approach would be to partially attack last round subkey bits that involve large bias value but with less number of known plaintext and ciphertext pairs. When the key bits are recovered partially, the methodology discussed above is applied continuously till all subkey bits in last round are recovered. Once all last round subkey bits are decrypted, the subkey bits in other rounds can be easily attacked by working backwards through the cipher using the same plaintext / ciphertext data.

2.5 Differential Cryptanalysis

Unlike linear cryptanalysis, differential cryptanalysis is a chosen plaintext attack [9] [21]. In this approach, the attacker chooses plaintext input and tries to inspect the output in order to obtain the subkey bits. In differential cryptanalysis, the attacker tries to find high probability matches between plaintext differences and the differences between

the last rounds of cipher [16]. Consider the m-bit input vector $X = [X_1, X_2, ...X_m]$ and the m-bit output vector $Y = [Y_1, Y_2, ...Y_m]$. The notations X_α and X_β represent the two inputs used for the difference during the differential attack, while Y_α and Y_β signify their corresponding outputs respectively. The input difference and output difference are given by $\Delta X = X_\alpha \oplus X_\beta$ and $\Delta Y = Y_\alpha \oplus Y_\beta$ respectively, where $\oplus$ represents bitwise XOR operation. The notation $(\Delta X, \Delta Y)$ is referred to as a difference pair.

The idea of the differential attack is to find the maximum probability p_D for the differential pair. The value of p_D should be greater than $\frac{1}{2^m}$. We will call a differential characteristic as a sequence of ΔX and ΔY over the rounds of the cipher such that ΔY for the j^{th} round corresponds to ΔX for the $j+1$ round.

2.5.1 Construction of Difference-table

In this section, we will look into the creation of the difference-table of S-boxes. In the SPN structure, the differential characteristic of the overall cipher structure is constructed by exploiting the properties of S-boxes. The high probability of a difference pair for each round is concatenated to find good overall differential probability p_D and the corresponding differential characteristic. The subkey bits of the cipher are not present in the difference expression unlike the linear expression in linear cryptanalysis. This is because of the differential characteristic: the subkey bits are present as output difference in the j^{th} round and as input difference in round $j+1$, and hence they cancel out each other when the difference expression is evaluated. We will explain the creation of S-

boxes in the context of the 16-bit SPN using 4×4 S-boxes. The input vector $X = [X_1, X_2, X_3, X_4]$ and the output vector $Y = [Y_1, Y_2, Y_3, Y_4]$ takes part in the difference pair of S-boxes. The S-box that we used for the construction of difference-table is from the second row of the S-box S_1 [4] as shown in Table 2.4.

X	Y	$\Delta Y(\Delta X = 0010)$	$\Delta Y(\Delta X = 0111)$
0000	0000	0111	0001
0001	1111	1011	0010
0010	0111	0111	0101
0011	0100	1011	1010
0100	1110	0011	1010
0101	0010	0011	0101
0110	1101	0011	0010
0111	0001	0011	0001
1000	1010	0110	0010
1001	0110	1101	0101
1010	1100	0110	1001
1011	1011	1101	0010
1100	1001	1010	0010
1101	0101	1101	1001
1110	0011	1010	0101
1111	1000	1101	0010

Table 2.9: Sample Difference Pairs of the S-box

A sample difference pair of the S-box of Table 2.4 is constructed as shown in Table 2.9. The output difference value ΔY is calculated for a given ΔX by

$$X_r \oplus \Delta X = X_{sr} \Rightarrow Y_{sr} \oplus Y_r = \Delta Y \tag{2.7}$$

where subscripts r and sr correspond to row number in Table 2.9. When the r^{th} row element of vector X is bitwise XORed with a fixed ΔX, then the resultant value is located at row sr in the column of vector X. The notation $\Rightarrow$ signifies the bijective

mapping between X and Y. The Equation (2.7) can be explained by an example. If X_r = 0001 and ΔX = 0010, then $X_r \oplus \Delta X = X_{rr}$ = 0011. The corresponding value of output Y_r when input X_r = 0011 is equal to 0100. The value Y_{rr} when XORed with Y_r = 1111 results in ΔY = 1011 as shown in the second row of column three in Table 2.9.

The difference-table is created by counting the number of occurrences of the ΔY value for a given ΔX value. From the Table 2.9 the number of occurrences of ΔY = 0010 when ΔX = 0111 is equal to 6. Therefore in the difference-table as shown in Table 2.10 the value corresponding to $(\Delta X = 0111, \Delta Y = 0010)$ is equal to 6. The resulting differential probability for the $(\Delta X, \Delta Y)$ pair will be equal to 6/16. The values shown in the difference-table are represented in decimal form.

From the difference-table we can view some interesting properties [16]. All the values in the table are positive and even. The sum of all elements in a row and column is equal to 2^m. We also notice that when ΔX = 0 or ΔY = 0, all the values in the row and column are equal to 0 except when ΔX = ΔY = 0 the table value is equal to 16.

$(\Delta X, \Delta Y)$	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	2	0	0	0	2	0	0	0	2	8	0	0	2
2	0	0	0	4	0	0	2	2	0	0	2	2	0	4	0	0
3	0	4	0	0	2	0	2	0	2	0	2	0	0	0	0	4
4	0	0	0	6	0	2	0	0	0	0	2	0	0	2	2	2
5	0	2	2	0	2	0	2	0	2	2	0	0	0	0	0	4
6	0	0	0	0	0	2	2	0	0	4	0	0	0	2	6	0
7	0	2	6	0	0	4	0	0	0	2	2	0	0	0	0	0
8	0	0	0	0	0	0	0	4	0	4	2	2	0	0	2	2
9	0	0	2	0	0	4	2	0	2	0	0	4	2	0	0	0
A	0	0	2	0	6	0	0	0	0	0	2	0	2	4	0	0
B	0	4	0	2	0	0	2	0	4	0	0	2	0	0	2	0
C	0	2	0	0	6	0	0	0	0	2	4	0	2	0	0	0
D	0	0	0	0	0	2	4	2	4	0	0	0	0	2	0	2
E	0	2	2	2	0	0	0	6	0	2	0	0	0	0	2	0
F	0	0	2	0	0	2	0	0	2	0	0	4	2	2	2	0

Table 2.10: Difference-table of the S-box

2.5.2 Differential Characteristic of the Overall Cipher Structure

The presence of the key does not have any effect on the differential characteristic of the cipher [16] [22]. Hence the input difference going to the S-box will have the same effect if the subkey is present or absent. The overall differential characteristic is obtained by concatenating the S-box difference pairs in each round, such that overall differential characteristic involves the plaintext bits and the input bits of the last round of S-boxes. The subset of the subkey bits that follow the last round are obtained by using the differential characteristic that associated with the plaintext bits and second last round output bits [16] [23].

The attack on the cipher structure is explained with reference to 16-bit SPN using 4×4 S-boxes. In Figure 2.6, a sample attack on the 4 round cipher is illustrated. Figure 2.6 shows the active S-boxes, and the corresponding ΔX and ΔY values for each round. The difference pairs of the S-boxes involved that are referred to as active S-boxes, are shown in Table 2.11.

ACTIVE S-BOXES: S_{ij}	$(\Delta X, \Delta Y)$	DIFFERENTIAL PROBABILITY p_D
S_{14}	(1,C)	8/16
S_{21}	(1,C)	8/16
S_{22}	(1,C)	8/16
S_{31}	(C,4)	6/16
S_{32}	(C,4)	6/16

Table 2.11: Difference Pairs of Active S-boxes

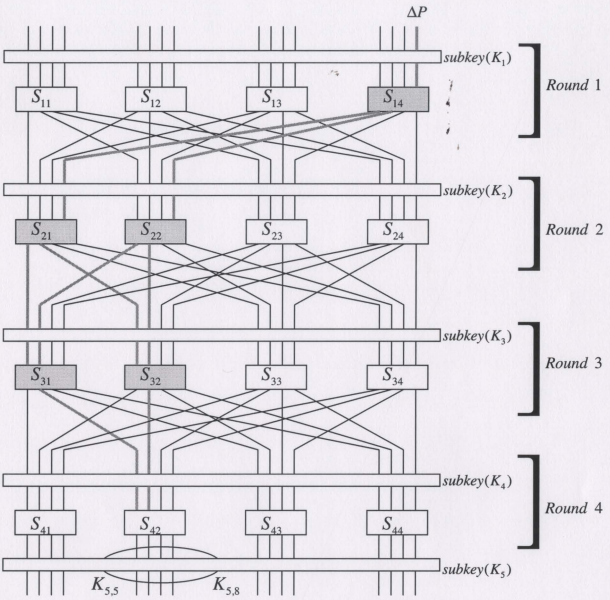

Figure 2.6: Differential Cryptanalysis of Cipher-A

The overall differential probability of the cipher is given by $P_D = \prod p_D$ [16], where p_D is the differential probability of each active S-box involved in the differential characteristic of cipher. In other words, the overall differential probability is simply the

product of all active S-box probabilities. Then from Figure 2.6, the partial subkey bits that are attacked are $K_{5,5}$ to $K_{5,8}$. Therefore the differential probability P_D used to obtained subkey bits $K_{5,5}$ to $K_{5,8}$ is equal to $(\frac{8}{16})^3 \times (\frac{6}{16})^2$. The number of plaintext / ciphertext pairs required to recover the partial subkey bits successfully is given by $N_{PC} \approx v / P_D$, where N_{PC} is the number of plaintext ciphertext required and v is a small constant factor [16].

Once an R-1 round differential characteristic is determined with high probability, the last round partial subkey bits are recovered by partially decrypting the last round of cipher. So in our example the target partial last round subkey bits are $K_{5,5}$ to $K_{5,8}$. For each guess of the target subkey bits, a counter variable is incremented each time the difference to the last round of cipher is consistent with the characteristic. The differential characteristic is tested for a large number of plaintext and ciphertext pairs N_{PC}. The correct partial subkey will have maximum value for the counter variable. An incorrect key will be equivalent to the random guess and has very low probability. Hence the correct partial subkey bits is given by probability

$$P_D = count / N_{PC} . \tag{2.8}$$

It is expected that, for the correct target partial subkey, the value of P_D calculated from Equation (2.8) should be close to the theoretical value of P_D calculated as the product of the differential probabilities of he active S-boxes.

2.6 Conclusion

Linear cryptanalysis and differential cryptanalysis are effective attacks on block ciphers. In order to provide the security against linear and differential cryptanalysis, S-boxes should be carefully selected (i.e. they should not have high biases or differential probabilities) and the cipher structure must necessitate a large number of active S-boxes in the attack. For example, the AES design uses an S-box and SPN structure in its cipher construction which is resistive to linear and differential attacks. In the next chapter, we will look into number of algorithms that have been studied and developed related to the analysis of linear and differential cryptanalysis of block ciphers. The objective of the algorithms is to find the best linear approximation and differential characteristic for the block cipher structure.

Chapter 3

Analysis and Implementation of Algorithms

This chapter reviews a number of algorithms that have been studied and developed by us related to linear cryptanalysis and differential cryptanalysis. These algorithms find the best bias or differential probability for a block cipher and a corresponding linear approximation or differential characteristic. The first algorithm to be studied is an algorithm developed by Matsui [24], in which he found the best linear approximation probability or differential probability of the DES cipher. His approach provided motivation to find a tool and developed an algorithm that finds the best linear approximation and differential characteristic of a block cipher structure like a Substitution Permutation Network (SPN). Such an algorithm can be used as a tool to analyze the effectiveness of the attacks against SPNs, and thereby improve the design and selection of cipher components.

In the first section, a Depth First Search (DFS) algorithm is discussed. The search for the best path within the cipher structure should have minimum time complexity. For all the algorithms discussed in this chapter the searching mechanism is DFS based and the algorithms discussed are similar except that they differ in the pruning mechanism and the decision criterion for optimizing the running time of the algorithm.

In the second section, the approach used by Matsui in finding the best linear approximation of the DES cipher is discussed. In later sections, original algorithms developed related to the SPN structure are introduced. The algorithms that are developed by us are given below:

1. Intelligent Pruning Mechanism (IPM) using DFS Algorithm
2. Modified Matsui (MM) Algorithm
3. Two-Round Iterative (TRI) Algorithm

In each section, algorithm is described along with the advantages and limitations of the approach. Some preliminary results are also shown to strengthen the viewpoint. Pseudocode, figures and tables are shown when necessary to explain the algorithms more clearly and precisely.

3.1 Depth First Search (DFS) Algorithm

In this section, the function of the Depth First Search (DFS) algorithm is discussed before considering its implementation in algorithms developed for SPNs. DFS broadly operates on a directed graph or an undirected graph [25]. A tree structure is a way of representing the hierarchical nature of a structure in a graphical form. Every finite tree structure has a member called the *root node* that is the topmost node in the hierarchy of the tree structure. A node is the "*parent*" of another node if it is one step higher in the hierarchy and closer to root node. A node is the "*sibling*" of other nodes if it shares the same parent node. A node is called a "*child*" node if it is connected to a parent node and is one level lower in the hierarchy than a parent node. The purpose of branches is to connect pairs of nodes in a tree structure. The objective of DFS is to search through all the nodes in a tree structure by starting at the root node. DFS is a search that advances by expanding the first child node of the search tree that appears and thus going deeper and deeper until a goal node is found, or until it hits a node that has no children. Then the search backtracks, returning to the most recent node it had not finished exploring [26].

The overall complexity of the execution time of the DFS algorithm is $O(|V|+|E|)$ where V represents the set of nodes and E represents the set of branches in the graph [26]. Therefore it is of linear order in terms of the number of nodes and the number of branches. Since the complexity of a search cannot be better than a linear order, this search procedure will be an optimal one if it is necessary to visit all nodes in the search of the graph. However if the number of nodes is large (e.g. 2^{68}) the searching mechanism can still be slow. The searching mechanism can be made to run faster by avoiding the visit of some nodes, if it is certain that the desired information is not present there. The general pseudo code for the DFS algorithm is shown in Figure 3.1.

```
Procedure Round-1:

BEGIN
 FOR all nodes w DO
  array[w]= 0                    // initialize all node unvisited
  parent[w]=NULL                 // initialize all parent of tree as NULL
 END FOR
 FOR all nodes w DO
    IF array[w] = = 0 THEN call Procedure DFS(w)
EXIT

Procedure DFS(node w):
 Visit (w)                       // traversing a node
 array[w]=1                      // node visited so change the flag
 FOR all node v adjacent to w DO
    IF array[v]= = 0  THEN       // check if sibling visited
       parent[v]=w
      call Procedure DFS(v)
    END IF
 END FOR
END Procedure
```

Figure 3.1: General Pseudocode for DFS Algorithm [26]

In this thesis we consider the basic SPN architecture for finding the maximum probability to be used in differential cryptanalysis and the maximum bias as applicable to linear cryptanalysis. In our algorithms, the nature of the use of DFS is to find optimal branches through all layers of the tree such that some branches and nodes do not need to be visited if they are clearly not optimal. This we call *pruning*. Hence pruning tries to reduce search space within the tree structure. When any of the algorithms that are discussed in the thesis are executed on, an implicit tree structure is formed. Each node in the tree structure is equivalent to ΓX_i for linear cryptanalysis and ΔX_i for differential cryptanalysis, where i is the round number of the SPN and $\Gamma X_i / \Delta X_i$ represent the linear mask/difference of the input to a round. Each layer in the tree structure symbolizes the corresponding round number for the SPN. The purpose of a DFS algorithm is to search the branches through each layer of the tree structure to find where the optimal solution resides. In the case of linear cryptanalysis, an optimal solution is an approximation with the maximum bias. For differential cryptanalysis, an optimal solution is a characteristic with the maximum probability. If DFS traverses through all nodes in the tree structure then the order of complexity with respective to number of nodes is approximately equal to $2^{N \times n}$, such that there are 2^N nodes in each round and n is the number of rounds involved in the approximation.

3.2 Matsui's Algorithm on DES

Matsui exploited the Feistel structure [10] of DES and developed a practical algorithm that can be used in attacking or analyzing the resistance of DES using linear and differential cryptanalysis [24]. In his work, Matsui makes use of induction to find the

best n-round probability[1] B_n from the knowledge of best i-round probabilities B_i $(1 \le i \le n-1)$ [24]. The algorithm by nature is recursive. In his algorithm, Matsui uses *greedy approach* and *pruning mechanism*. The greedy approach tries to find the good approximation / characteristic while searching in the tree structure. Besides that, pruning tries to reduce search space within the tree structure by avoiding a visit to the branches and nodes that will clearly be not optimal. The pseudocode for the algorithm is presented in [24].

Matsui's results show that his approach is good enough to get the best probability [24] in good time for the DES structure, but this approach will not necessarily yield the best probability efficiently for other block cipher structures such as SPNs. A modified version of Matsui's approach for the SPN structure discussed in Section 3.4 proves the above claim.

3.3 Intelligent Pruning Mechanism (IPM) using DFS Algorithm

In the next section, we will consider the modified version of Matsui's approach to SPNs, but we first consider a straight-forward application of a DFS based algorithm using a greedy approach. The algorithm finds the best linear approximation or best differential characteristics for the SPNs. This algorithm basically uses a greedy based approach and a pruning mechanism while searching the tree structure using DFS so that branches leading to suboptimal solutions can be avoided. The breadth of the tree structure searched grows dramatically with the increase in number of rounds, size of S-boxes and block size.

[1] For simplicity we shall typically refer to "probability" when discussing the algorithms, where, in the case of linear cryptanalysis, this refers to "bias", and for differential cryptanalysis, this refers to the differential probability.

In this section, discussion of the algorithm is in the context of linear cryptanalysis. Similar concepts can be applied using differential cryptanalysis. The two key aspects that are incorporated in the algorithm are the *greedy approach* and the *intelligent pruning mechanism*. The term *greedy approach* in the context of the algorithm determines the order of the branches to be searched while the *intelligent pruning mechanism* determines when branches are too bad to be considered for further searching in the algorithm, based on a conditional check. In other words, in order to reduce the number of search operations and avoid unnecessary traversing of nodes and branches within the tree structure, the intelligent pruning mechanism is used to find the best bias of the network as soon as possible.

The greedy aspect is applied on the substitution block because it produces the non-linearity in the system. During the linear cryptanalysis, each input to an $m \times m$ S-box has 2^m possible output combinations, to select as branches that lead to further searching. The greedy approach of our algorithm basically considers the arrangement of these masking output bits from the S-box with respect to the masking input bits and the active S-box involved during such transformation. If the masked output bits lead to the right solution quickly, then the solution is achieved with fewer search operations. Thus in our algorithm, we try to find the appropriate $(\Gamma X_i, \Gamma Y_i)$ value from the bias-table for each round i using a greedy approach, so that the resulting solution of n rounds is the best one. The greedy approach is useful because it helps to find the maximum bias by minimizing active S-boxes (Hamming weight) and maximizing S-box bias value (bias-table). The two greedy approaches for the selection criteria of ΓY_i for the output bits of an S-box that are used in the algorithm by us are listed below:

1. For a given S-box, firstly arrange output values ΓY_i from the bias-table for a given input value ΓX_i in descending order of bias, and then arrange ΓY_i of same bias based on Hamming weight[2] of ΓY_i in ascending order.
2. For a given S-box, firstly arrange output values ΓY_i from the bias-table based on Hamming weight for a given input value ΓX_i in ascending order, and then arrange ΓY_i of same Hamming weight based on bias in descending order.

The intelligent pruning mechanism used in the algorithm tests the boundary condition

$$[p_1 \times p_2 \times p_3 \times ..., p_i \times Z_{n-i}] \geq \overline{B_n} \tag{3.1}$$

and acts accordingly. In the boundary condition notation p_i signifies the bias of the i^{th} round, Z_{n-i} corresponds to maximum possible bias for the *n-i* round and $\overline{B_n}$ is the current estimate of the best *n*-round bias. If the boundary condition is false, the algorithm avoids the search further down the hierarchical tree structure because the best solution does not reside in that space.

Another important intelligent pruning mechanism used in the algorithm is the reduction in the number of branches at the root of the tree structure. This means that if not all masking input-output pair combinations are searched, then the search time can be reduced significantly. Hence, in round one, for all possible candidates for output combinations ΓY_1 , the maximum bias is found by looking at the bias-table. If there is more than one ΓX_1 then only one ΓX_1 is used, as other values can be found by looking at ΓY_1 and the bias p_1. (The explanation in this paragraph is in context of linear

[2] Hamming weight is referred to the number of input / output bits active

cryptanalysis.) In the case of differential cryptanalysis, the bias-table is replaced by the difference-table and ΓY_1 and ΓX_1 are replaced by ΔY_1 and ΔX_1 respectively. The pruning approach discussed here is used not only in the IPM algorithm but also used in the MM algorithm for SPN and the TRI algorithm, which will be discussed in later sections.

Both the greedy approaches give the same final result theoretically and practically but the motive is to investigate which one of the two greedy approaches gives the result faster for a given network and S-box. The pseudocodes of the algorithms for the linear and differential cryptanalysis are shown in Figure 3.2 and Figure 3.3, respectively. The inputs to the algorithms are the network structure and the S-box properties. The output of the algorithms is $\overline{B_n}$, the estimate of current n-round bias / differential probability. The pseudocode can also be modified to keep track of the actual approximation / characteristic. The notations that are used in the pseudo code are explained below:

- Variable n is equal to R-1, where R is the total number of rounds in the cipher structure. This is because, as discussed in chapter 2, the attacks work by finding the best linear approximation or differential characteristic for R-1 rounds.
- $p_i = (\Gamma X_i, \Gamma Y_i)$ is the bias of the i^{th} round for the linear cryptanalysis using the piling-up lemma.
- $p_i = (\Delta X_i, \Delta Y_i)$ is the probability of the i^{th} round for differential cryptanalysis.
- $[p_1, p_2, p_3, ..., p_n]$ is computed as n-round bias for the linear cryptanalysis using the piling-up lemma and as n-round differential probability for the differential cryptanalysis.

- $\Gamma X_i = perm(\Gamma Y_{i-1})$ represents the permutation of the mask for the output bits of round $i-1$ for linear cryptanalysis.
- $\Delta X_i = perm(\Delta Y_{i-1})$ represents the permutation of the difference for the output bits of round $i-1$ for the differential cryptanalysis.
- $p_n = \max_{\Gamma Y}(\Gamma X_n, \Gamma Y)$ is the probability calculated as the magnitude of the maximum value of the masking output bits for a given masking input bits ΓX_n.
- $p_n = \max_{\Delta Y}(\Delta X_n, \Delta Y)$ is the probability calculated as the magnitude of the maximum value of the difference output bits for a given difference input bits ΔX_n.
- Z_{n-i} equal to $2^{n-i} \times |\{\max(\Gamma X, \Gamma Y)/2^m\}|$, where $\max(\Gamma X, \Gamma Y)$ is the maximum value in the bias-table for the linear cryptanalysis. For differential cryptanalysis, Z_{n-i} equal to $\max(\Delta X, \Delta Y)/2^m$ from the difference-table.
- Term $\overline{B_n}$ is the current estimate of the maximum n-round bias / differential probability of the cipher, B_n.

Procedure Round 1

BEGIN

$\overline{B_n} = 0$

FOR each candidate for ΓX_1 and ΓY_1 **DO**

$p_1 = (\Gamma X_1, \Gamma Y_1)$

IF $[p_1, Z_{n-1}] \geq \overline{B_n}$ **THEN** call Procedure Round (i)

END FOR

EXIT

Procedure Round (i)

FOR each candidate for ΓY_i **DO**

$\Gamma X_i = perm(\Gamma Y_{i-1})$

$p_i = (\Gamma X_i, \Gamma Y_i)$

IF $[p_1, p_2, p_3, \ldots p_i, Z_{n-i}] \geq \overline{B_n}$ **THEN** call Procedure Round (i+1)

END FOR

RETURN

Procedure Round (n)

$\Gamma X_n = perm(\Gamma Y_{n-1})$

$p_n = \max_{\Gamma Y}(\Gamma X_n, \Gamma Y)$

IF $[p_1, p_2, p_3, \ldots p_n] \geq \overline{B_n}$ **THEN** $\overline{B_n} = [p_1, p_2, p_3, \ldots p_n]$

RETURN

Figure 3.2: Pseudocode for IPM Algorithm for Linear Cryptanalysis.

The greedy selection criteria are examined on an SPN network consisting of 6 rounds of 4×4 S-boxes as shown in Figure 3.4. This is Cipher-A as discussed in Chapter 2. The DES S-box is used for the study is given in Table 2.1. For an SPN, the linear cryptanalyst requires a $R-1$ round approximation to mount a linear attack successfully,

where R is the number of rounds in the cipher. Hence the result of computation for 5 rounds is presented, as the overall cipher has 6 rounds.

Procedure Round 1

BEGIN
$\overline{B_n} = 0$
FOR each candidate for ΔX_1 and ΔY_1 **DO**
$p_1 = (\Delta X_1, \Delta Y_1)$
IF $[p_1, Z_{n-1}] \geq \overline{B_n}$ **THEN** call Procedure Round (i)
END FOR
EXIT

Procedure Round (i)

FOR each candidate for ΔY_i **DO**
$\Delta X_i = perm(\Delta Y_{i-1})$
$p_i = (\Delta X_i, \Delta Y_i)$
IF $[p_1, p_2, p_3, \ldots p_i, Z_{n-i}] \geq \overline{B_n}$ **THEN** call Procedure Round (i+1)
END FOR
RETURN

Procedure Round (n)

$\Delta X_n = perm(\Delta Y_{n-1})$
$p_n = \max_{\Delta Y}(\Delta X_n, \Delta Y)$
IF $[p_1, p_2, p_3, \ldots p_n] \geq \overline{B_n}$ **THEN** $\overline{B_n} = [p_1, p_2, p_3, \ldots p_n]$
RETURN

Figure 3.3: Pseudocode for IPM Algorithm for Differential Cryptanalysis.

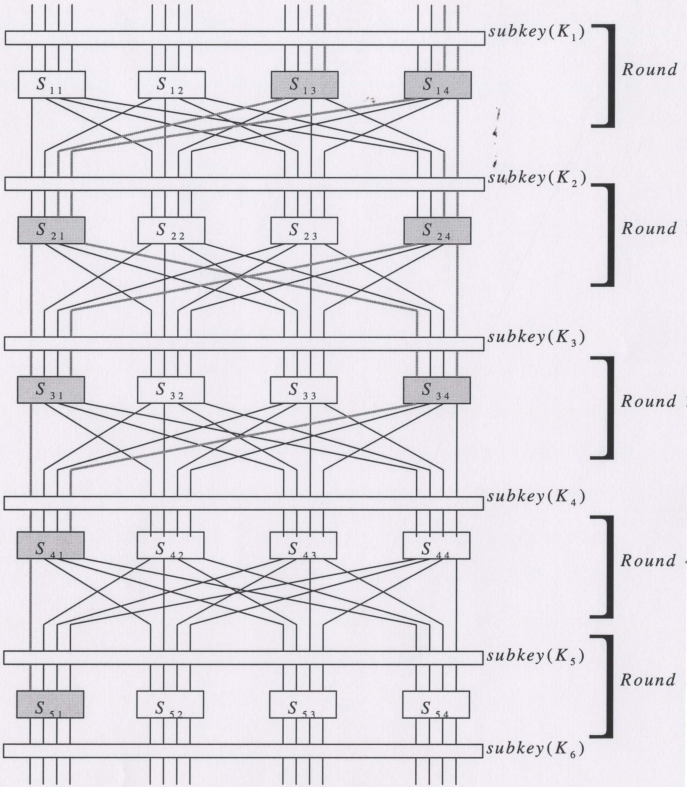

Figure 3.4: Linear Approximation of SPN Consisting of 6 Rounds of 4×4 S-boxes.

0	0	3	3
3	0	0	3
9	0	0	9
9	0	0	0
8	0	0	0

Table 3.1: Tabular Representation of Linear Approximation for Cipher-A.

Table 3.1 is the representation of Figure 3.4 in tabular form. All the active S-boxes in Figure 3.4 are shaded, which is equivalent to each cell in the table that has non-zero values. All the non-zero values in the table, in hexadecimal form are equivalent to the inputs going to the active S-boxes, where the leftmost bit represents the most significant bit. For example in round 1, two dark input lines going into S-box S_{13} and S_{14} in binary are equivalent to 0011, which in table form can be viewed in row one column 3 and column 4, respectively, and is represented by hexadecimal value 3. Table 3.1 and Figure 3.4 shows the actual masking input and output bits involved for each round along with the active S-boxes in order to obtain the maximum bias value for the network. Since tabular form is more convenient than the figure representation, in our work results will be shown in tabular form except where figure representation is necessary. The tabular representation applies similar to differential cryptanalysis.

There can be more than one linear approximation to get the same bias in SPN architecture. For the 6-round cipher, the maximum bias B_5 is equal to 0.01483. The maximum value for any active S-box is equal to 6, which is obtained from the bias-table

of the S-box. In the above case, there are 8 active S-boxes that yield this result. For both greedy approaches, the same result is obtained and the result is optimal (i.e. the best bias is determined). However due to the nature of the S-box and small network architecture there is not much time difference for the two greedy approaches. There can be significant time difference if a more realistically sized cipher consisting of 8 rounds of 8×8 S-boxes is considered. We will discuss this further in the subsequent paragraphs.

In Table 3.2, we present the best 7 round linear approximation for an 8 round of Cipher-B. Again each rectangle that is not equal to zero represents an active S-box while non-zero values correspond to the inputs in each round. Each row symbolizes a cipher round. The maximum bias value B_7 of the network is 4.5e-7[3]. The results are achieved by running the TRI algorithm for SPN that will be discussed and implemented in the Section 3.6.

0	0	0	0	0	0	37	0
0	0	0	0	0	0	2	0
2	0	0	0	0	0	0	0
80	0	0	0	0	0	0	0
0	80	0	0	0	0	0	0
0	40	0	0	0	0	0	0
0	0	0	0	40	0	0	0

Table 3.2: Tabular Representation of Linear Approximation for Cipher-B.

The algorithm discussed in Section 3.3 cannot be used in this case because the network size is too large and the execution time of algorithm increases dramatically. The

[3] "eX" to represent 10^X

purpose of showing the above result is that a better result can be achieved if the behavior of the S-box and the cipher structure is known. In the above case, the maximum value in the bias table is 32 when the Hamming weight of the output bits is greater than 2. But a better overall bias value is obtained when Hamming weight of output bits is one and has a maximum value of 18. If the first greedy approach is used then in round two there will be at least 3 active S-boxes initially which will eventually result in very low $\overline{B_n}$ and large number of active S-boxes for the overall cipher structure. (Hence, a good estimate for $\overline{B_n}$ is obtained slowly because the masking output bits have low biases when Hamming weight is equal to one.) This will deteriorate the searching mechanism as the number of nodes in the tree structure increases dramatically and the probability that the search is in the right direction is low. However, if the second greedy approach is employed then a good estimate for $\overline{B_n}$ is achieved quickly and the search can be expedited rapidly. The result suggests that maximum bias of Cipher-B is attained when the Hamming weight of the output bits is low and, as a result, less number of active S-boxes is required. Hence we will use the second greedy approach for all of our algorithms.

Although the result obtained using the IPM algorithm is optimal, its execution time is still large even after using the above greedy approach and pruning mechanism. For the larger networks, the resulting execution time for the algorithm is prohibitive.

3.4 Modified Matsui (MM) Algorithm

Matsui's recursive algorithm [24] cannot be used in SPN structures as it is, due to the distinct nature of the DES and SPN structures. Even though both are block cipher structures, in DES, substitution and permutation operate only on half of the block at a

time. On the other hand, in the SPN architecture, substitution and permutation operate on the whole block at once. Figure 3.5 and 3.6 presents the pseudocode of Matsui's algorithm modified to operate on SPNs for linear and differential cryptanalysis, respectively. The pseudocode for the linear and differential cases is almost the same except for the difference in the test condition for Procedure Round-1 and Procedure Round-i. In the linear case, the test condition or boundary condition given by

$$[p_1, p_2, p_3, .. p_i, B_{n-i}] \geq \overline{B_n} \tag{3.2}$$

is governed by the piling-up lemma, while in the differential case, the boundary condition given by Equation (3.2) is the product of probabilities of the individual rounds. The modified Matsui approach is somewhat better than the IPM algorithm discussed in Section 3.3 because it has a close boundary condition as stated above. On the other hand in the IPM algorithm, the boundary condition given by (3.1) is not a close boundary condition because it has no knowledge of best $n-1$ rounds linear biases (or similarly differential probabilities). However, the MM algorithm uses the same greedy approach as IPM and the same first round simplifications as IPM.

In the IPM algorithm, the boundary condition is not true until the search reaches far down in the tree structure. Hence, it is not as efficient as the MM algorithm. While both of the algorithms discussed guarantee the best n round linear biases and differential probabilities, they are limited in their practicality because of the large execution time for large networks.

The notations used in the pseudocode have the meaning as mentioned in Section 3.3. In the algorithm the term $\overline{B_n}$ represents the current estimate of the maximum n-round bias / differential probability of cipher, B_n. Note that the determination of B_n

requires the previous determination of B_{n-1}. Hence, the algorithm iteratively computes the best bias / probability for approximation / characteristics of increasing number of rounds.

Procedure Round 1

BEGIN
$\overline{B_n} = 0$
FOR each candidate for ΓX_1 and ΓY_1 **DO**
$p_1 = (\Gamma X_1, \Gamma Y_1)$
IF $[p_1, B_{n-1}] \geq \overline{B_n}$ **THEN** call Procedure Round (i)
END FOR
EXIT

Procedure Round (i)

FOR each candidate for ΓY_i **DO**
$\Gamma X_i = perm(\Gamma Y_{i-1})$
$p_i = (\Gamma X_i, \Gamma Y_i)$
IF $[p_1, p_2, p_3, .., p_i, B_{n-i}] \geq \overline{B_n}$ **THEN** call Procedure Round (i+1)
END FOR
RETURN

Procedure Round (n)

$\Gamma X_n = perm(\Gamma Y_{n-1})$
$p_n = \max_{\Gamma Y}(\Gamma X_n, \Gamma Y)$
IF $[p_1, p_2, p_3, .., p_n] \geq \overline{B_n}$ **THEN** $\overline{B_n} = [p_1, p_2, p_3, .., p_n]$
RETURN

Figure 3.5: Pseudocode for MM Algorithm for Linear Cryptanalysis

Procedure Round 1

$\overline{B_n} = 0$

BEGIN

FOR each candidate for ΔX_1 and ΔY_1 **DO**

$p_1 = (\Delta X_1, \Delta Y_1)$

IF $[p_1, B_{n-1}] \geq \overline{B_n}$ **THEN** call Procedure Round (i)

END FOR

EXIT

Procedure Round (i)

FOR each candidate for ΔY_i **DO**

$\Delta X_i = perm(\Delta Y_{i-1})$

$p_i = (\Delta X_i, \Delta Y_i)$

IF $[p_1, p_2, p_3, .., p_i, B_{n-i}] \geq \overline{B_n}$ **THEN** call Procedure Round (i+1)

END FOR

RETURN

Procedure Round (n)

$\Delta X_n = perm(\Delta Y_{n-1})$

$p_n = \max_{\Delta Y}(\Delta X_n, \Delta Y)$

IF $[p_1, p_2, p_3, .., p_n] \geq \overline{B_n}$ **THEN** $\overline{B_n} = [p_1, p_2, p_3, .., p_n]$

RETURN

Figure 3.6: Pseudocode for MM Algorithm for Differential Cryptanalysis

3.5 Limitations, Issues Encountered for the First Two Algorithms

Both the IPM and the MM approach in the SPN architecture have their own limitations. The complexity of both the algorithms increases considerably with the increase in the number of rounds and the number of active S-boxes in each round. While working on the smaller 16-bit SPN architecture (Cipher-A), the algorithms ceases to give the result for the structures that have greater than 8 rounds in reasonable amount of time.

This result is not surprising because the DFS algorithm has to traverse most of the nodes and branches to make sure that result obtained is the best one. The greedy based approach cannot make a good decision for the best result after every round until it gets a better bias after scanning the whole cipher structure in the case of IPM. Hence it has to run almost all the way down the structure to make a decision. The MM algorithm also suffers from the same problem. Hence these two approaches are not efficient for larger networks.

Despite these issues, preliminary investigation on the 16-bit SPN architecture using the above two approaches gives some indication that the best result is acquired when a small number of S-boxes is involved in the linear approximation or differential cryptanalysis. Such a result can be viewed, for example, from Table 3.1 where, for the 5 round linear approximation out of 20 S-boxes just 8 active S-boxes are needed to get the best result. Hence, the experimental results signals that some logical constraints may be put on the searching mechanism so that result is attained at good pace.

Given the aforementioned realizations, design objectives for a new algorithm were determined:

1. Pruning during the search should be improved by using some strict constraints so that it can be used for ciphers of more than 8 rounds, i.e., larger, more secure cipher structures.
2. The algorithm should be scalable to a large network such as 64-bit SPN structure based on 8×8 S-boxes and the 64-bit SPN structure based 4×4 on S-boxes.
3. Complexity of the algorithm should be reduced from a potential exponential factor to some linear factor when the number of rounds is increased.

In order to meet the above design objectives the two algorithms discussed in Sections 3.3 and Section 3.4 requires modification. The searching mechanism can be improved or speedup with respect to SPN structure in the following ways:

1. Put constraints on the number of active S-boxes in each round.
2. Find the maximum linear bias or differential probability value rapidly so that unnecessary branches can be eliminated.
3. Carefully limit the number of S-boxes for the first round and intelligently chooses masking input-output pairs or difference input-output pairs from the bias-table or difference-table such that overall result can be maximized.
4. Examine how closely the practical result is near to theoretical upper bound. The upper bound occurs when each round has one S-box and the maximum bias / probability value from the bias table / difference-table.
5. Efficiently implement the inner sub procedures of algorithm so that it reduces the overall complexity of the algorithm.

3.6 Two-Round Iterative (TRI) Algorithm

Considering the factors as stated in Section 3.5, a new algorithm has been developed that incorporates most of the above features. After studying the bias table in linear analysis or difference-table in differential analysis and the SPN architecture we found out that if we reduce the calculation in early rounds we can save lots of trivial search. So the new algorithm collects some good results after intelligently pruning the first two rounds, then uses these results to determine good results for the outcome of the next two rounds. These results are then used for the next two rounds and the process

continues to derive good results for the outcome of n rounds. Therefore the approach is divided into several steps with each step being of similar complexity. In total there are $n/2$ steps and hence the algorithm is linear in the number of rounds n.

This approach really fits the objective and serves the purpose of getting a good result in superior time. This approach is also a greedy based approach except that the search aggressively prunes branches of the tree with the potential of producing a non-optimal result. That is, while it is expected that the result will be a good linear bias or differential probability, it is not guaranteed to be the best. The overall bias probability or differential probability is found by collecting some good biases or probabilities after every two rounds. Then decisions are made by using these biases or probabilities assuming that it will lead to the best results after the next two rounds.

The intuition behind the approach is to collect a list of close to optimal results after every two rounds, i.e., the list of results obtained after round 2, are used to calculate good results after round 4. The process is iteratively repeated after every two rounds until the good result is obtained after n rounds. The assumption in the approach is that by biasing the result for n, on good results for n-2, the overall result will be good, if not optimal. As we shall see in Chapter 4, the results obtained using this approach gives good biases or probabilities for n round linear approximation or differential characteristics.

We refer to this algorithm as the Two-Round Iterative or TRI algorithm. The division of number of rounds in a cipher structure in the algorithm is discussed by an example. Suppose a 10 round SPN cipher structure is available. As per the previous knowledge, a linear attack requires a 9 round approximation to mount an attack successfully. Hence nine rounds are divided in four 2 rounds plus the last 1 round. The

last round is separate because of its different nature in decision making when compared to the previous rounds. On the other hand, if the number of rounds in the cipher is odd then in the last two rounds the decision criterion is different from the previous rounds.

The algorithm is illustrated in the context of Cipher-A. The four input bits of an S-box gives 4 output bits in Cipher-A. So the S-box has Hamming weight of output bits and input bits from 0 to 4. In round one, the maximum bias value or maximum differential probability from the bias-table or difference-table is found for a given masking or a difference output-input pair for all possible combinations of Hamming weight 0 to 4. The purpose is basically to run the algorithm from round two and minimize the searching for masking or difference input-output pairs for round one as discussed in Section 3.3. If the above work is not done then it will involve 2^{16} possible masking or difference inputs before making a decision for the good result in round one. A good masking or difference output value of round one is used as masking or difference input for round two after the permutation. The bias values or differential probabilities are not affected by the permutation block. The searching is done intelligently using these values in round two. A sorted array is stored that contains good masking or difference inputs and their corresponding biases or probabilities after every two rounds. The sorted array size should be large enough to make a good decision for the next two rounds. Assuming odd n, the same procedure is repeated upto n-1 rounds with good linear biases or differential probabilities are stored in a sorted array. However, in the last round n the maximum bias or differential probability is calculated for a given masking or difference input-output pair. A good bias or probability obtained quickly in round n then becomes the decision

criteria for the rest of the ΓX_n or ΔX_n values. After all ΓX_n or ΔX_n values are exhausted, a good approximation of B_n is made.

The idea of an array is to include good solutions for the sub problem. The array is sorted so that the good biases or differential probabilities reside at the lower index in the array. The resulting sorted array helps to facilitate the searching for a good approximation / characteristic after the next 2 rounds.. The sorted array should be large enough so that optimal solution has a good chance of remaining in the set.

The pseudo code for the TRI algorithm with respect to linear and differential cryptanalysis is given in Figure 3.7 and Figure 3.8, respectively. The greedy approach and the intelligent pruning used in TRI algorithm are same as the IPM algorithm. However, number of active S-boxes is constraint to maximum three S-boxes per round. This is done to get the good result in practical time as not many S-boxes are involved per round during the linear and differential cryptanalysis. The inputs to the algorithms are the network structure and the S-box properties. The outputs of the algorithms are the bias / differential probability and the corresponding linear approximation / differential characteristic.

Two-Round Iterative Algorithm:

BEGIN
i=0, $\overline{B_n}$ =0
FOR $1 \le k \le n$ **DO**
  FOR $1 \le j \le L$ **DO**
    $C_k[j] = 0$
  END FOR
END FOR
WHILE $i < n$
  IF $(i == 0)$ **THEN** call Procedure FirstRound ()
  IF $(i+2) > n$ **THEN** call Procedure Round (n)
  IF $(i+2) == n$ **THEN** call Procedure Round-even (n-1)
  ELSE call Procedure Round (i+1)
  $i = i+2$
END WHILE
EXIT

Procedure FirstRound ()

FOR each candidate for ΓY_1 **DO**
  $p_1 = \max_{\Gamma X}(\Gamma X, \Gamma Y_1)$
  $\Gamma X_1 = \Gamma X \mid p_1 = (\Gamma X, \Gamma Y_1)$
  IF $[p_1, Z_{n-1}] \ge \overline{B_n}$ **THEN**
    $\Gamma X_2 = perm(\Gamma Y_1)$
    FOR each candidate for ΓY_2 **DO**
      $p_2 = (\Gamma X_2, \Gamma Y_2)$
      IF $[p_1, p_2] \ge$ smallest bias in C_2 **THEN**
        INSERT $[p_1, p_2]$ in C_2
        INSERT ΓX_1 in D_1', ΓY_1 in D_1"
        INSERT ΓX_2 in D_2', ΓY_2 in D_2"
      END IF
    END FOR
  END IF
END FOR
RETURN

Procedure Round (i)

FOR $1 \le j \le L$ **DO**
  $\delta = C_{i-1}[j]$, $\Gamma X_i = perm(D_{i-1}"[j])$
  FOR each candidate for ΓY_i **DO**
    $p_i = (\Gamma X_i, \Gamma Y_i)$
    IF $[\delta, p_i, Z_{n-i}] \ge \overline{B_n}$ **THEN**
      $\Gamma X_{i+1} = perm(\Gamma Y_i)$
      FOR each candidate for ΓY_{i+1} **DO**
        $p_{i+1} = (\Gamma X_{i+1}, \Gamma Y_{i+1})$
        IF $[\delta, p_i, p_{i+1}] \ge$ smallest bias in C_{i+1} **THEN**
          INSERT $[\delta, p_i, p_{i+1}]$ in C_{i+1}
          INSERT ΓX_i in D_i', ΓY_i in D_i"
          INSERT ΓX_{i+1} in D_{i+1}', ΓY_{i+1} in D_{i+1}"
        END IF
      END FOR
    END IF
  END FOR
END FOR
RETURN

Procedure Round (n)

FOR $1 \le j \le L$ **DO**
  $\delta = C_{n-1}[j]$, $\Gamma X_n = perm(D_{n-1}"[j])$
  $p_n = \max_{\Gamma Y}(\Gamma X_n, \Gamma Y)$
  $\Gamma Y_n = \Gamma Y \mid p_n = (\Gamma X_n, \Gamma Y)$
  IF $[\delta, p_n] \ge \overline{B_n}$ **THEN** $\overline{B_n} = [\delta, p_n]$
  IF $[\delta, p_n] \ge$ smallest bias in C_n **THEN**
    INSERT $[\delta, p_n]$ in C_n
    INSERT ΓX_n in D_n', ΓY_n in $D_n"$
  END IF
END FOR
RETURN

Procedure Round-even (i)

FOR $1 \le j \le L$ **DO**
  $\delta = C_{i-1}[j]$, $\Gamma X_i = D_{i-1}"[j]$
  FOR each candidate for ΓY_i **DO**
    $p_i = (\Gamma X_i, \Gamma Y_i)$
    IF $[\delta, p_i, Z_1] \ge \overline{B_n}$ **THEN**
      $\Gamma X_{i+1} = perm(\Gamma Y_i)$
      $p_{i+1} = \max_{\Gamma Y}(\Gamma X_{i+1}, \Gamma Y)$
      $\Gamma Y_{i+1} = \Gamma Y \mid p_{i+1} = (\Gamma X_{i+1}, \Gamma Y)$
      IF $[\delta, p_i, p_{i+1}] \ge \overline{B_n}$ **THEN** $\overline{B_n} = [\delta, p_i, p_{i+1}]$
      IF $[\delta, p_i, p_{i+1}] \ge$ smallest bias in C_n **THEN**
        INSERT $[\delta, p_i, p_{i+1}]$ in C_n
        INSERT ΓX_i in D_{n-1}', ΓY_i in $D_{n-1}"$
        INSERT ΓX_{i+1} in D_n', ΓY_{i+1} in $D_n"$
      END IF
    END IF
  END FOR
END FOR
RETURN

Figure 3.7: Pseudocode for TRI Algorithm for Linear Cryptanalysis

Two-Round Iterative Algorithm:

BEGIN
i=0, $\overline{B_n}$ =0
FOR $1 \le k \le n$ **DO**
  FOR $1 \le j \le L$ **DO**
    $C_k[j] = 0$
  END FOR
END FOR
WHILE $i < n$
  IF $(i == 0)$ **THEN** call Procedure FirstRound ()
  IF $(i+2) > n$ **THEN** call Procedure Round (n)
  IF $(i+2) == n$ **THEN** call Procedure Round-even (n-1)
  ELSE call Procedure Round (i+1)
  $i = i + 2$
END WHILE
EXIT

Procedure FirstRound ()

FOR each candidate for ΔY_1 **DO**
  $p_1 = \max_{\Delta X}(\Delta X, \Delta Y_1)$
  $\Delta X_1 = \Delta X \mid p_1 = (\Delta X, \Delta Y_1)$
  IF $[p_1, Z_{n-1}] \ge \overline{B_n}$ **THEN**
    $\Delta X_2 = perm(\Delta Y_1)$
    FOR each candidate for ΔY_2 **DO**
      $p_2 = (\Delta X_2, \Delta Y_2)$
      IF $[p_1, p_2] \ge$ smallest bias in C_2 **THEN**
        INSERT $[p_1, p_2]$ in C_2
        INSERT ΔX_1 in D_1', ΔY_1 in D_1"
        INSERT ΔX_2 in D_2', ΔY_2 in D_2"
      END IF
    END FOR
  END IF
END FOR
RETURN

Procedure Round (i)

FOR $1 \le j \le L$ **DO**
  $\delta = C_{i-1}[j]$, $\Delta X_i = perm(D_{i-1}"[j])$
  FOR each candidate for ΔY_i **DO**
    $p_i = (\Delta X_i, \Delta Y_i)$
    IF $[\delta, p_i, Z_{n-i}] \ge \overline{B_n}$ **THEN**
      $\Delta X_{i+1} = perm(\Delta Y_i)$
      FOR each candidate for ΔY_{i+1} **DO**
        $p_{i+1} = (\Delta X_{i+1}, \Delta Y_{i+1})$
        IF $[\delta, p_i, p_{i+1}] \ge$ smallest bias in C_{i+1} **THEN**
          INSERT $[\delta, p_i, p_{i+1}]$ in C_{i+1}
          INSERT ΔX_i in D_i', ΔY_i in D_i"
          INSERT ΔX_{i+1} in D_{i+1}', ΔY_{i+1} in D_{i+1}"
        END IF
      END FOR
    END IF
  END FOR
END FOR
RETURN

```
Procedure Round (n)

FOR 1≤ j≤ L DO
   δ = C_{n-1}[j],  ΔX_n = perm(D_{n-1}"[j])
   p_n = max_ΔY (ΔX_n, ΔY)
   ΔY_n = ΔY | p_n = (ΔX_n, ΔY)
   IF [δ, p_n] ≥ B_n THEN B_n = [δ, p_n]
   IF [δ, p_n] ≥ smallest bias in C_n THEN
     INSERT [δ, p_n] in C_n
     INSERT ΔX_n in D_n', ΔY_n in D_n"
   END IF
 END FOR
 RETURN

Procedure Round-even (i)

 FOR 1≤ j≤ L DO
    δ = C_{i-1}[j],  ΔX_i = D_{i-1}"[j]
   FOR each candidate for ΔY_i DO
     p_i = (ΔX_i, ΔY_i)
     IF [δ, p_i, Z_i] ≥ B_n THEN
       ΔX_{i+1} = perm(ΔY_i)
       p_{i+1} = max_ΔY (ΔX_{i+1}, ΔY)
       ΔY_{i+1} = ΔY | p_{i+1} = (ΔX_{i+1}, ΔY)
       IF [δ, p_i, p_{i+1}] ≥ B_n THEN B_n = [δ, p_i, p_{i+1}]
       IF [δ, p_i, p_{i+1}] ≥ smallest bias in C_n THEN
         INSERT [δ, p_i, p_{i+1}] in C_n
         INSERT ΔX_i in D_{n-1}', ΔY_i in D_{n-1}"
         INSERT ΔX_{i+1} in D_n', ΔY_{i+1} in D_n"
       END IF
     END IF
   END FOR
 END FOR
 RETURN
```

Figure 3.8: Pseudocode for TRI Algorithm for Differential Cryptanalysis

Most of the notations used in the above pseudocode have the same meaning as cited in Section 3.3. Some new notations that are used explicitly for the above pseudo code are discussed below.

- C_{i+1} is a sorted array of size L. The value of L should be large enough to collect good bias / probability for i+1 rounds and is ideally large enough to include the

partial optimal solution. C_{i+1} is always sorted in ascending order. If the array is full and the new value is larger than smallest value in the array, then the new value is inserted into the appropriate position within the array.

- D_{i+1}' is an array of size L. The array collects the masking / difference input values for active S-boxes of round i+1 corresponding to biases / probabilities in array C_{i+1}.
- D_i' is an array of size L. It collects the masking / difference input values for active S-boxes of round i. These values help in backtracking the masking inputs and the active S-boxes when the n-round bias / probability is found and the corresponding linear approximation / differential characteristic is to be determined.
- D_i'' and D_{i+1}'' represents output masking / difference value corresponding to D_i' and D_{i+1}' respectively.
- INSERT operation inserts the value, if appropriate, into the appropriate position in the array.

In Chapter 4, we will study the effects of varying the array size L.

3.7 Advantages of TRI Algorithm

In this section, the advantages of the TRI algorithm over the IPM and MM algorithm are discussed. The complexity of algorithm increases linearly with the increase in the number of rounds. This is a significant achievement with this algorithm. As a result, we can get good results for the number of rounds greater than 8. In fact, we can analyze highly secure ciphers that include a large number of rounds. The search is

reduced drastically because the tree search is performed only on maximum two rounds; hence number of nodes contributing in the tree structure is reduced. The searching for masking or difference input-output pair in round one is reduced as the algorithm is executed from round two by exploiting the properties of S-box from the bias-table or difference-table. As a result, the algorithm is scalable to large networks like Cipher-B and Cipher-C. If the algorithm's sub procedures are implemented efficiently then the execution time of the algorithm can be modest, resulting in an algorithm that gives a result in practical time. However, this result, while good is not guaranteed to be optimal.

In Table 3.3, a sample result for the TRI algorithm is given. The result shown in the table is for the linear cryptanalysis of a 10 round Cipher-A. The S-box used in the simulation is the DES S-box is shown in Table 2.1. Table 3.3 shows active S-boxes along with the input values in the boxes for each round for the solution derived by the algorithm.

0	0	6	0
0	0	2	2
3	3	3	0
E	0	0	E
9	9	0	0
C	0	0	0
0	0	8	0
2	0	0	0
8	8	8	0

Table 3.3: Tabular Representation of 10 Round Linear Approximation for Cipher-A using TRI Algorithm.

The maximum bias value B_9 is found to be 1.466e-4. There are total 16 active S-boxes out of maximum 36 S-boxes which is approximately 44.4%. The algorithm gives a result in approximately 20 minutes. Table 3.4 shown below reflects on bias value after every two rounds.

Round No.	Bias Value
2	.14062
4	.01483
6	.00185
8	3.47e-4

Table 3.4: Tabular Representation of Intermediate Solution for Cipher-A.

This approach is employed on some realistic SPN cipher structures and results are investigated and discussed later in the thesis in Chapter 4. By using the TRI algorithm, good results are achieved in practical time which may be the best result in most of the cases for the SPN architecture. However a drawback with TRI algorithm is that there is no way mathematically or practically to guarantee that the result attained is the optimal. Nevertheless, the algorithm is a useful tool to look into properties of S-boxes and cipher structures that are necessary for good cryptographic resistance to linear and differential cryptanalysis.

3.8 Conclusion

In this chapter, we discussed about the three algorithms: the IPM algorithm, the MM algorithm and the TRI algorithm. All algorithms were implemented for SPN architectures by us. The limitations and advantages of the three algorithms were discussed and preliminary results shown where necessary to strengthen our viewpoints. In particular, the TRI algorithm was noted to be efficient and practical for large networks and is conjectured to be capable of determining good linear biases and differential probabilities and the corresponding linear approximations and differential characteristics.

We can also state the worst case complexities for the IPM, MM and our TRI algorithms. The worst case execution time of exhaustively searching all nodes in case of IPM and MM algorithm is given by,

$$t(N,n) \; \varepsilon \; O(2^{N \times n}) \tag{3.3}$$

where $t(N,n)$ represents the worst case execution time of exhaustively searching nodes in IPM or MM algorithm, 2^N is the maximum number of nodes in each round and n is the number of rounds in the approximation. On the other hand the worst case execution time of exhaustively searching all nodes in case of TRI algorithm is given by,

$$t(N,n,L) \; \varepsilon \; O(2^{2 \times N} + L \times n) \tag{3.4}$$

where $t(N,n,L)$ represents the worst case execution time of exhaustively searching nodes in TRI algorithm and L is a sorted list after every two rounds. Hence from Equation (3.3) and (3.4) we can deduce that worst case execution time complexity for the TRI algorithm is better than IPM and MM algorithm.

This leads to the next chapter where results are discussed in more detail. It will be shown that efficiency of the TRI algorithm is indeed favorable.

Chapter 4

Algorithm Results for Different Ciphers

In this chapter, we will look at the results obtained by running the IPM algorithm, MM algorithm and the TRI algorithm. Discussion is also given by analyzing the results for Cipher-A, Cipher-B and Cipher-C.

In Section 4.1, different networks of Cipher-A, distinguished by different S-boxes, are considered and a discussion is involved by running the optimal algorithm (IPM and MM) and our non-optimal algorithm (TRI). Some conclusions are drawn at the end of the section that strengthens our intuition to use our TRI algorithm on larger networks to investigate the characteristics of the ciphers with reference to linear and differential cryptanalysis. In section 4.2, a study is made by using different networks (i.e., different S-boxes) of Cipher-B and some comparisons are drawn by analyzing the results. Finally, in Section 4.3, we will test our algorithm on a number of different networks of Cipher-C.

All the tables were constructed by running the algorithm on Intel Pentium Centrino 1.6 GHz processor with 512 MB RAM (Random Access Memory) except for tables obtained for Results for Cipher-C. Results for Cipher-C where obtained by running the algorithm on Intel Pentium4 3.0 GHz processor with 2GB RAM. The algorithms were implemented in Java and the environment selected was Eclipse.

4.1 Results for Cipher-A

In this section, we examine the results for the 16-bit network based on 4×4 S-boxes. The intent of studying this network is that we will analyze the efficiency and effectiveness of our TRI algorithm as compared to the optimal IPM and MM algorithms.

4.1.1 Results for Randomly Selected S-boxes

Table 4.1 indicates the maximum values from the bias-table and the difference-table. It is obtained for 20 different S-boxes, each corresponding to one of the Cipher-A networks. The S-boxes were selected by changing the input-output mapping randomly. The actual S-box mapping for all 20 S-boxes is given in the Appendix (Table A.1).

As discussed earlier in Chapter 2, all the values in the tables are even. In 90% of the ciphers, the highest value from the bias-table is 6 when ΓX and ΓY are not equal to zero. Therefore the maximum bias value is equal to 6/16. The higher the bias value for the linear approximation, the better will be the effectiveness of linear cryptanalysis [16]. A similar analogy holds true for the differential cryptanalysis.

Cipher	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Max. value (bias-table)	6	6	6	6	6	6	6	6	6	6	6	4	4	6	6	6	6	6	6	6
Max. value (difference-table)	6	4	6	6	6	6	8	10	6	6	6	4	6	6	6	6	6	6	6	8

Table 4.1: Maximum Value in the Bias-table and Difference-table for 20 Different Cipher-A Networks using Random S-boxes.

Table 4.2 shows the bias for a 6 round linear approximation of 20 different Cipher-A networks using the randomly selected S-boxes. The bias values are shown for the optimal algorithm (IPM/MM) and for non-optimal algorithm (TRI) with different buffer lengths, L. As discussed in Chapter 3, the optimal algorithms do not give a result in practical time for large number of rounds for Cipher-A. The results shown in Table 4.2 are for a small 7 round Cipher-A. From the Table 4.2, we see that for 80% of the cases, the result of our TRI algorithm when L=4000 is equal to the optimal result. The results also show that the TRI algorithm matches the bias value of the optimal algorithm even for L=100 for 60% of the time. In most cases, when the TRI algorithm does not give an optimal result, it still finds a good non-optimal result. Since the complexity of the linear cryptanalysis is inversely proportional to the square of the bias, a non-optimal result that is in the same order as the optimal result is still indicative of cipher's resistance to the attack. For example, for cipher 1 the bias value is equal to .00741 for L=4000, 1000 and 100, which is approximately 1.05 times less than the optimal solution. Even for L=50 the result is not bad and is approximately 1.18 times less than the optimal solution. Similarly, for cipher 15 the factor is 1.18, which is not that significant in linear cryptanalysis. Overall the very low value of L=50, the TRI algorithm result equals the optimal solution for 45% of the cases and for rest of the cases the result is close to optimal.

From Table 4.2, we can see that cipher 9 is a very weak cipher as the resulting bias is equal to the upper bound for the bias of a 6-round approximation with the corresponding S-box. That is the result that involves one S-box in each round during the approximation and is based on the maximum value in the bias-table. Contrary to cipher 9, cipher 5 is a strong cipher because it has very low bias value. From these 20 random

networks, we can also see the significant variation of bias value from .08898 to .00247, a factor of 36 which indicates a large variation in the resistance level of the ciphers.

Cipher	IPM/MM (Optimal)	TRI (L=4000)	TRI (L=1000)	TRI (L=100)	TRI (L=50)
1	.00781	.00741	.00741	.00741	.00659
2	.00439	.00439	.00195	.00195	.00195
3	.01178	.01178	.01178	.01178	.01178
4	.00989	.00988	.00988	.00988	.00988
5	.00247	.00247	.00247	.00247	.00195
6	.00781	.00781	.00781	.00781	.00781
7	.00781	.00781	.00781	.00781	.00390
8	.01977	.01757	.01757	.01757	.01757
9	.08898	.08898	.08898	.08898	.08898
10	.00781	.00781	.00781	.00781	.00781
11	.02966	.02966	.02966	.02636	.02636
12	.00781	.00781	.00781	.00781	.00390
13	.00781	.00781	.00781	.00781	.00781
14	.00494	.00494	.00494	.00247	.00247
15	.00439	.00370	.00370	.00370	.00370
16	.00781	.00781	.00781	.00781	.00781
17	.00781	.00781	.00586	.00439	.00439
18	.00781	.00781	.00781	.00781	.00781
19	.00313	.00219	.00219	.00219	.00195
20	.00781	.00781	.00781	.00781	.00781

Table 4.2: Maximum Bias for 6 Round Approximation of 20 Different Cipher-A Networks using Random S-boxes.

Table 4.3 shows the maximum number of active S-boxes involved during the 6 round linear approximation of Cipher-A. From this table, we can see that for 90% of the cases, the total number of active S-boxes involved in 6 round approximations is less than or equal to 8. Hence, the average number of S-boxes per round is small and is less than 2, even equal to 1 for 60% of the cases. Table 4.3 gives an intuition that an optimal solution is not likely to involve a large number of active S-boxes in each round.

Cipher	IPM/MM (Optimal)	TRI (L=4000)	TRI (L=1000)	TRI (L=100)	TRI (L=50)
1	6	9	9	9	8
2	7	7	8	8	8
3	6	6	6	6	6
4	8	8	8	8	8
5	9	9	9	9	8
6	6	6	6	6	6
7	6	6	6	6	7
8	7	6	6	6	6
9	6	6	6	6	6
10	6	6	6	6	6
11	7	7	7	6	6
12	6	6	6	6	6
13	6	6	6	6	6
14	8	8	8	10	10
15	7	10	10	10	10
16	6	6	6	6	6
17	6	6	7	8	8
18	6	6	6	6	6
19	12	8	8	8	8
20	6	6	6	6	6

Table 4.3: Number of Active S-boxes Involved in 6 Round Approximation of 20 Different Cipher-A Networks using Random S-boxes

We can see an anomaly in cipher 19, where an optimal result involves a larger number of active S-boxes as compared to the non-optimal solution found by the TRI algorithm. The reason is that the TRI algorithm makes use of the greedy approach and intelligent pruning while making a decision after every two rounds as indicated in the details about the algorithm given in Chapter 3.

Table 4.4 characterizes the execution time in finding the maximum bias by an optimal IPM and the non-optimal TRI algorithm. There is a large variation in time in finding the optimal solution with the time ranging from 1 minute to 48 minutes. If the

solution is found very early during the search operation then the algorithm takes much less time.

Cipher	IPM(Optimal) Time(min:sec)	TRI (L=4000) Time(min:sec)	TRI (L=1000) Time(min:sec)	TRI (L=100) Time(min:sec)	TRI (L=50) Time(min:sec)
1	4:00	7:00	1:20	0:35	0:23
2	25:00	10:00	2:00	0:30	0:30
3	2:00	4:00	1:00	0:20	0:15
4	2:00	10:00	2:30	0:30	0:30
5	48:00	14:00	2:00	0:40	0:30
6	3:00	4:00	1:20	0:40	0:30
7	3:00	10:00	1:20	0:20	0:12
8	1:00	14:00	1:00	0:25	0:15
9	1:00	4:00	1:00	0:10	0:06
10	1:00	9:00	1:30	0:30	0:08
11	1:00	5:00	1:20	0:08	0:05
12	2:00	3:00	1:30	0:30	0:30
13	2:00	9:00	1:20	0:30	0:30
14	10:00	5:00	1:30	0:15	0:13
15	13:00	8:00	1:20	0:30	0:30
16	4:00	7:00	2:00	0:15	0:10
17	4:00	8:00	1:20	0:30	0:25
18	3:00	6:00	1:30	0:30	0:30
19	40:00	10:00	2:00	0:30	0:40
20	3:00	8:00	1:30	0:30	0:25

Table 4.4: Execution Time Required in 6 Round Approximation of 20 Different Cipher-A Networks using Random S-boxes

However if the algorithm doesn't find the optimal solution during the early phase of the search operation, then it has to consider many branches and nodes, which will significantly increase the execution time. This is the reason that the IPM algorithm ceases to give results in practical time when the number of rounds are increased in Cipher-A.

The average execution time for the TRI algorithm when L=4000 is approximately equal to 8 minutes, and when L=100, it is approximately 28 seconds. Hence the execution time for TRI algorithm increases with the increase in the L value. But increasing L also

increases the probability of finding an optimal result or a close to optimal result. There is not much variation in time in finding the result using the TRI algorithm for a given L, because TRI tries to find good result after every two rounds; hence its complexity is linear with respect to number of rounds.

Table 4.5 represents the differential probability for a 6 round characteristic of 20 different Cipher-A networks using random S-boxes. The TRI algorithm gives optimal results in 85%, 70%, 60% and 40% of the time when L is equal to 4000, 1000, 100 and 50 respectively. Even when the result is not optimal the value is close to optimal for L=4000. Since the complexity of the differential cryptanalysis is inversely proportional to the differential probability, a non-optimal result that is in the same order as the optimal result is still indicative of cipher's resistance to the attack. For example, in cipher 10, the TRI algorithm gives approximately 1.5 times lesser value as compared to the optimal solution. Similarly, in cipher 18, the factor is reduced to 1.33. Considering the case when L=100 and cipher 10, the solutions are just 1.5 times less than the optimal. Hence, the result for the differential probability for different random networks also strengthens the argument for the viability of the TRI algorithm. That is, although TRI does not guarantee an optimal result, it effectively finds good, often optimal solutions.

Table 4.6 strengthens our claim that not many active S-boxes are required in each round during linear and differential cryptanalysis of SPNs. From the table, 85% of the cases involved only one active S-box per round in finding the optimal 6 round differential characteristics of a cipher. The result also shows that 100% of the time the average number of S-boxes in each round is less than 2.

Cipher	IPM/MM (Optimal)	TRI (L=4000)	TRI (L=1000)	TRI (L=100)	TRI (L=50)
1	1.37e-4	1.37e-4	1.03e-4	1.03e-4	1.03e-4
2	6.10e-5	6.10e-5	6.10e-5	7.62e-6	3.81e-6
3	9.15e-5	9.15e-5	9.15e-5	9.15e-5	6.86e-5
4	1.54e-4	1.54e-4	1.54e-4	7.72e-5	7.72e-5
5	1.52e-5	1.52e-5	1.52e-5	1.52e-5	1.52e-5
6	.00278	.00278	.00278	.00278	.00278
7	2.44e-4	2.44e-4	2.44e-4	2.44e-4	2.44e-4
8	9.31e-4	9.31e-4	8.94e-4	8.94e-4	8.94e-4
9	5.49e-4	5.49e-4	5.49e-4	5.49e-4	1.83e-4
10	3.09e-4	2.05e-4	1.54e-4	1.37e-4	1.37e-4
11	9.15e-5	9.15e-5	9.15e-5	9.15e-5	6.10.e-5
12	6.10e-5	6.10e-5	6.10e-5	6.10e-5	6.10e-5
13	1.52e-5	1.52e-5	1.52e-5	1.52e-5	1.52e-5
14	1.52e-5	1.52e-5	1.52e-5	1.52e-5	1.52e-5
15	1.16e-4	1.16e-4	1.16e-4	1.16e-4	2.44e-5
16	5.15e-5	5.15e-5	3.86e-5	2.29e-5	2.29e-5
17	2.29e-5	2.29e-5	2.29e-5	2.29e-5	2.29e-5
18	9.15e-5	6.86e-5	6.86e-5	6.10e-5	6.10e-5
19	.00278	.00278	.00278	.00278	.00278
20	2.44e-4	1.22e-4	1.22e-4	1.22e-4	1.22e-4

Table 4.5: Maximum Differential Probability of 6 Round for 20 Different Cipher-A Networks using Random S-boxes.

The time variation for the optimal algorithm is dramatic in finding the 6 round differential characteristic of Cipher-A. The results are shown in Table 4.7. The time ranges from 20 seconds to 43 minutes for a small number of rounds in a cipher. However using TRI algorithm the variation in time is not that significant. For example, when L=4000, the average time for the cipher is approximately 8 minutes, and it ranges from 4 minutes to 13 minutes. Similarly, when L is less than or equal to 100, the time variation is in magnitude of seconds.

Cipher	IPM/MM (Optimal)	TRI (L=4000)	TRI (L=1000)	TRI (L=100)	TRI (L=50)
1	6	6	7	7	7
2	6	6	6	7	7
3	6	6	6	6	6
4	6	6	6	8	8
5	6	6	6	6	6
6	6	6	6	6	6
7	6	6	6	6	6
8	9	9	8	8	8
9	6	6	6	6	6
10	6	6	7	6	6
11	6	6	6	6	6
12	6	6	6	6	6
13	6	6	6	6	6
14	6	6	6	6	6
15	8	8	8	8	10
16	8	8	8	6	6
17	6	6	6	6	6
18	6	7	7	6	6
19	6	6	6	6	6
20	6	6	6	6	6

Table 4.6: Number of Active S-boxes Involved in 6 Round Differential Characteristics of 20 Different Cipher-A Networks using Random S-boxes

4.1.2 Results Using Good S-boxes in Cipher-A Network

In this section, we will consider good S-boxes that are selected to satisfy some cryptographic properties, like the DES S-boxes. The DES S-boxes are 4×4 S-boxes and the mapping of the S-boxes is shown in the Appendix (Table A.2). The maximum value in the bias-table and in the difference-table is shown in Table 4.8. When the 6 round linear approximation is made using the optimal IPM algorithm and the TRI algorithm, there is a good amount of difference in the result obtained from the two algorithms in the cipher 1. However in other four ciphers the result achieved using the TRI algorithm is optimal or close to optimal as depicted in Table 4.9.

Cipher	IPM(Optimal) Time(min:sec)	TRI (L=4000) Time(min:sec)	TRI (L=1000) Time(min:sec)	TRI (L=100) Time(min:sec)	TRI (L=50) Time(min:sec)
1	4:30	4:14	1:53	0:30	0:26
2	6:13	4:58	1:18	0:35	0:21
3	8:36	4:28	1:04	0:21	0:10
4	4:39	7:15	1:23	0:32	0:16
5	38:32	7:41	1:11	0:36	0:30
6	0:23	6:20	1:13	0:14	0:13
7	1:50	5:42	1:15	0:18	0:10
8	1:14	11:28	1:13	0:30	0:26
9	0:50	7:01	1:03	0:25	0:27
10	2:03	12:32	1:29	0:33	0:07
11	7:36	9:00	1:16	0:27	0:17
12	4:21	8:14	1:26	0:35	0:24
13	31:27	10:40	1:40	0:25	0:25
14	42:47	11:02	1:31	0:36	0:27
15	5:19	7:11	1:01	0:16	0:12
16	12:38	7:37	1:27	0:38	0:16
17	30:19	11:26	1:24	0:33	0:22
18	5:31	9:38	1:21	0:26	0:21
19	0:21	5:24	0:49	0:13	0:12
20	1:01	5:18	1:03	0:25	0:19

Table 4.7: Execution Time Required in 6 Round Differential Characteristic of 20 Different Cipher-A Networks using Random S-boxes

.

Cipher using DES S-boxes	1	2	3	4	5
Max.bias (bias-table)	6	6	6	6	6
Max. difference (difference-table)	8	6	8	8	8

Table 4.8: Maximum Value in Bias-table and Difference-table using DES S-boxes in Cipher-A Network

The TRI results from the ciphers 1, 3 and 5 give the impression that with the increase in the length L, the results obtained are better and somewhat close to optimal. Hence increasing the value of L appears to be a practical way to get close to the optimal result.

Cipher (DES S-boxes)	IPM/MM (Optimal)	TRI (L=4000)	TRI (L=1000)	TRI (L=100)	TRI (L=50)
1	.00469	.00247	.00208	.00110	.00110
2	.00293	.00293	.00293	.00293	.00293
3	.00235	.00195	.00124	.00110	.00110
4	.01583	.01583	.01583	.01583	.01583
5	.00989	.00989	.00989	.00742	.00626

Table 4.9: Maximum Bias of 6 Round Linear Approximation for 5 Different Cipher-A Networks using DES S-boxes

An interesting result can be viewed in Table 4.10, which shows the number of active S-boxes involved during the approximation of Cipher-A using DES S-boxes. The results for ciphers 1 and 3 shows that the number of active S-boxes involved in the approximation is more when the optimal result is found as compared to the TRI algorithm which involves fewer active S-boxes during the approximation. The execution time is shown in the Appendix (Table A.3). These results clearly suggest that average execution time per cipher for the optimal solution is factor of 3 more than the TRI algorithm for L=4000. Hence TRI is definitely faster as compared to the optimal IPM algorithm.

Cipher (DES S-boxes)	IPM/MM (Optimal)	TRI (L=4000)	TRI (L=1000)	TRI (L=100)	TRI (L=50)
1	12	10	12	7	7
2	8	8	8	8	8
3	11	8	9	9	9
4	12	12	12	12	12
5	8	8	8	9	11

Table 4.10: Number of Active S-boxes Involved in 6 Round Linear Approximation of 5 Different Cipher-A Networks using DES S-boxes

The results for TRI algorithm are also encouraging during the differential cryptanalysis of Cipher-A using DES S-boxes. The TRI algorithm gives the optimal result for 40% of the cases even for low value of L as shown in the Table 4.11. The solutions which are not optimal are still close to optimal in all the cases.

Cipher (DES S-boxes)	IPM/MM (Optimal)	TRI (L=4000)	TRI (L=1000)	TRI (L=100)	TRI (L=50)
1	1.54e-4	1.54e-4	1.54e-4	1.54e-4	1.54e-4
2	2.44e-5	1.53e-5	1.53e-5	8.58e-6	3.22e-6
3	1.53e-5	1.53e-5	1.53e-5	1.53e-5	1.53e-5
4	2.28e-5	1.72e-5	1.72e-5	1.72e-5	1.14e-5
5	1.73e-4	1.46e-4	1.46e-4	1.30e-4	1.30e-4

Table 4.11: Maximum Differential Probability for 6 Round Differential Characteristic of 5 Different Cipher-A Networks using DES S-boxes.

From the Table 4.12, we can deduce that the average execution time for the IPM algorithm is approximately 75 minutes as compared to TRI algorithm for L=4000 which gives a good result in approximately 8 minutes. Hence TRI is much faster as compared to

the IPM algorithm and is scalable to more number of rounds in Cipher-A as well as being applicable to 64-bit networks like Cipher-B and Cipher-C.

Cipher using DES S-box	IPM(Optimal) Time(min:sec)	TRI (L=4000) Time(min:sec)	TRI (L=1000) Time(min:sec)	TRI (L=100) Time(min:sec)	TRI (L=50) Time(min:sec)
1	80:10	8:27	1:30	0:34	0:17
2	73:23	7:53	1:56	0:27	0:13
3	114:11	8:32	1:25	0:37	0:29
4	91:43	9:18	1:37	0:33	0:40
5	17:14	6:01	1:28	0:36	0:19

Table 4.12: Execution Time Required in 6 Round Differential Characteristic of 5 Different Cipher-A Network using DES S-boxes

As discussed earlier in the section, the IPM and MM algorithms fail to give result in practical time when the number of rounds is greater than 8 for the Cipher-A network. However, the TRI algorithm can be used for a large number of rounds for the Cipher-A network. Results for a sample 15 round linear approximation are shown in Table 4.13. Since the optimal result of the network is not available, the results are compared with the loose upper bounds. The loose upper bound is calculated by using the maximum value from the bias-table and assuming that only one S-box will be involved during the approximation. This upper bound is very loose and in most cases even the optimal result will deviate from it because the optimal result will not always consist of one S-box and the maximum value in the table. This can be seen in the results in Tables 4.2, 4.3, 4.9 and 4.10. The results from the TRI algorithm give an insight look into the difficulty of the attack and also show the change in the linear approximation with the change in the value

of L. From the Table 4.13, we can see that the cipher 9 is a very weak cipher as its bias value is equal to the upper bound and is an order of magnitude 3 higher than that of other bias values found in the table.

Cipher	Upper Bound (15 rounds)	TRI (L=4000)	TRI (L=1000)	TRI (L=100)	TRI (L=50)
1	.00668	1.833e-5	1.031e-5	1.031e-5	1.031e-5
2	.00668	1.788e-7	1.341e-7	1.341e-7	1.005e-7
3	.00668	1.609e-6	1.430e-6	1.430e-6	1.430e-6
4	.00668	2.749e-5	2.749e-5	2.749e-5	2.749e-5
5	.00668	9.536e-7	9.536e-7	9.536e-7	9.536e-7
6	.00668	1.526e-7	1.526e-7	1.526e-7	1.526e-7
7	.00668	1.430e-6	1.073e-6	7.152e-7	2.384e-7
8	.00668	1.738e-4	1.158e-4	1.158e-4	1.158e-4
9	.00668	.00668	.00668	.00668	.00668
10	.00668	1.526e-5	1.526e-5	1.526e-5	1.526e-5
11	.00668	3.666e-5	3.666e-5	3.258e-5	3.258e-5
12	1.526e-5	1.526e-5	1.526e-5	1.526e-5	7.629e-7
13	1.526e-5	1.907e-6	9.536e-7	9.536e-7	9.536e-7
14	.00668	3.576e-7	3.576e-7	1.192e-7	7.54e-8
15	.00668	3.576e-7	2.682e-7	2.682e-7	2.263e-7
16	.00668	1.526e-5	1.526e-5	1.526e-5	1.526e-5
17	.00668	1.526e-5	1.526e-5	7.629e-6	3.814e-6
18	.00668	1.526e-5	3.814e-6	3.814e-6	3.814e-6
19	.00668	9.536e-7	9.536e-7	9.536e-7	9.536e-7
20	.00668	1.526e-5	1.526e-5	1.907e-6	3.576e-7

Table 4.13: Maximum bias of 15 Round Approximation for 20 Different Cipher-A Networks using Random S-boxes

Similarly, cipher 12 has a bias value that is equal to the upper bound, although the bias is not as poor as cipher 9. We can also see that cipher 5 and cipher 19 are good ciphers because they have low bias values among the 20 ciphers given in the table for L=4000. In approximately 90% of the cases, we can see that even for large value of L the result is not close to the upper bound and we can conjecture that the upper bound is significantly greater than the actual largest bias. We conjecture that a good way of

checking the performance of the TRI algorithm is to run the algorithm for different values of L. If for large L the result remains the same, then we can assume that the solution might be close to optimal, if not optimal. However, there is no way to prove whether the solution is optimal or not.

The execution times and the numbers of active S-boxes involved during the approximation are shown in Table 4.14 and in Table 4.15 respectively. The general trend in the execution time clearly suggests that increasing the value of L will increase the execution time but will also increase the probability of finding a good bias for the approximation. Table 4.15 indicates that average number of S-boxes required per round is less than or equal to 2. For L=4000, it also shows that for 55% of the cases the TRI finds a good solution by involving only one S-box per round.

Cipher	TRI (L=4000) Time(min:sec)	TRI (L=1000) Time(min:sec)	TRI (L=100) Time(min:sec)	TRI (L=50) Time(min:sec)
1	14:24	3:14	0:58	0:33
2	30:22	6:57	1:24	1:06
3	29:01	4:33	1:01	0:49
4	34:30	4:08	1:06	0:57
5	30:48	4:16	1:06	0:49
6	4:28	2:25	0:37	0:34
7	22:56	4:38	1:12	0:41
8	5:25	1:49	0:33	0:20
9	4:47	1:08	0:13	0:07
10	8:51	2:44	0:43	0:17
11	24:51	3:40	0:25	0:17
12	5:58	1:36	0:43	0:44
13	23:14	4:50	0:58	0:34
14	50:30	8:10	1:46	0:44
15	26:15	3:33	1:02	0:37
16	12:27	2:57	0:23	0:15
17	8:08	1:56	0:51	1:04
18	9:06	3:30	0:58	0:48
19	20:25	8:35	1:07	0:50
20	19:20	1:41	0:53	0:39

Table 4.14: Execution Time Required in 15 Round Linear Approximation for 20 Different Cipher-A Networks using Random S-boxes

Cipher	TRI (L=4000)	TRI (L=1000)	TRI (L=100)	TRI (L=50)
1	20	21	21	21
2	16	20	20	21
3	17	15	15	15
4	20	20	20	20
5	19	19	19	19
6	15	15	15	15
7	17	17	17	19
8	15	15	15	15
9	15	15	15	15
10	15	15	15	15
11	17	17	16	16
12	15	15	15	15
13	15	15	15	15
14	15	15	17	25
15	17	16	16	18
16	15	15	15	15
17	15	15	15	16
18	15	17	17	17
19	19	19	19	19
20	15	15	16	18

Table 4.15: Number of Active S-boxes Involved in 15 Round Linear Approximation for 20 Different Cipher-A Networks using Random S-boxes

For the 16-bit SPN network or Cipher-A, we can find that the TRI algorithm gives good results and in many of the cases, the result is optimal (i.e., the largest bias/differential probability is found) when L=4000. The length 4000 was arbitrarily chosen to ensure that sufficient number of results is available after every two rounds to make a good decision for the next two rounds. However this length can be increased or decreased depending how well the solution is achieved for low values of L. With smaller values of L, the algorithm will run faster; with larger values of L, the algorithm is more likely to find the optimal, i.e. largest, bias or differential probability. We also found that the TRI algorithm is much faster in giving the result as compared to IPM or MM because

of its linear complexity in terms of the number of rounds. Due to its efficiency, the TRI algorithm can be scalable to larger networks like the 64-bit Cipher-B and Cipher-C. In the next section, we will apply the TRI algorithm to Cipher-B and study the properties of the network by changing the S-boxes.

4.2 Results for Cipher-B

In this section we will look into the results obtained by running the TRI algorithm on Cipher-B networks to examine linear and differential cryptanalysis. However, first we will study the bias-table and the difference-table of an 8×8 S-box. From the bias-table and difference-table we will analyze the spread of the maximum value in the table. In general, 8×8 S-box will have 256×256 entries in the table. The maximum value in the bias-table can range from 2 to 126 and from 2 to 254 for the difference-table (excluding the (0,0) entry which will always be 128 for the bias-table and 256 for the difference-table). The higher the values in the table, the more effective the linear and differential cryptanalysis can be. Hence, a cipher designer wishes these values to be generally as low as possible.

The maximum bias value and difference value for 10000 randomly selected 8×8 S-boxes were used to plot a histogram corresponding to the maximum value in the bias-table versus the frequency of occurrence (count). This is shown in Figure 4.1. From the histogram, it can be seen that approximately 94% of the bias values are concentrated around 32 to 38. The most likely maximum value is found to be for 34 which constitutes about 39% of the total and is closely followed by value 36 which comprise approximately 32% of the total. Similarly, the histogram plotted in Figure 4.2 represents the maximum

value in the difference-table versus the frequency of occurrence (count). The plot shows that approximately 94% of the maximum difference probability values are shared between 10 and 12.

Based on the results from Section 4.1, we conjecture that typical number of active S-boxes in each round of the best linear approximation of a cipher is 1 or 2. Hence, while running the TRI algorithm for a number of different Cipher-B networks, we make the assumption that not more than 3 S-boxes are involved in each round in the best linear approximation and differential characteristic of a cipher. This constraint is necessary to make the TRI algorithm run in a practical amount of time on large networks. Later on, in the section, we will see in most cases only one S-box is involved in the good solution found by the TRI algorithm.

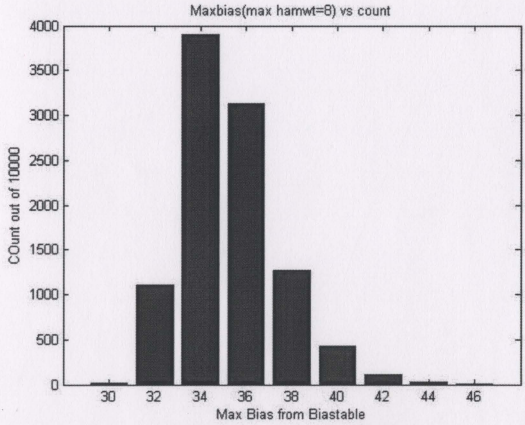

Figure 4.1: Histogram Showing Maximum Value in Bias-table Versus Count

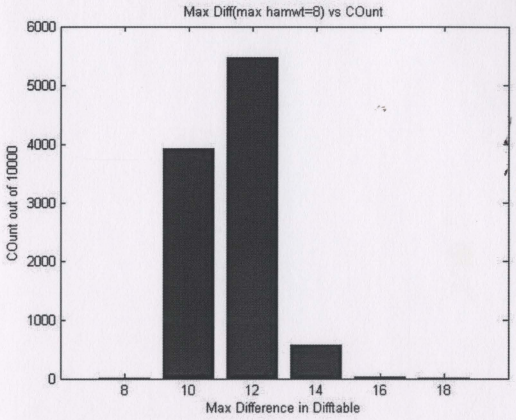

Figure 4.2: Histogram Showing Maximum Value in Difference-table Versus Count

We have also looked into the maximum value from the bias-table and difference-table when only one S-box is involved during the approximation, i.e., when the Hamming weights of the input/output mask/difference is one. Figure 4.3 and Figure 4.4 represent the maximum values when the Hamming weight is equal to one from the bias-table and difference-table, respectively. The plots in Figure 4.3 signifies that for Hamming weight equal to one, 65% of the masking values are concentrated around 18 to 22. Similarly, from the plot in Figure 4.4, it can be viewed that approximately 89% of the difference values are shared between 4 and 6. The plot for the maximum value for Hamming weight equal to two and three from the bias-table and difference-table are shown in the Appendix (Figures A.1 to A.4).

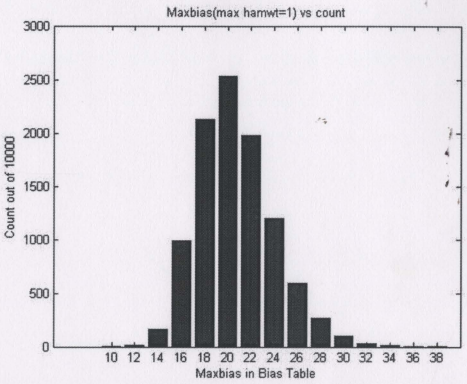

Figure 4.3: Histogram Showing Maximum Value in Bias-table Versus Count When Hamming Weight =1.

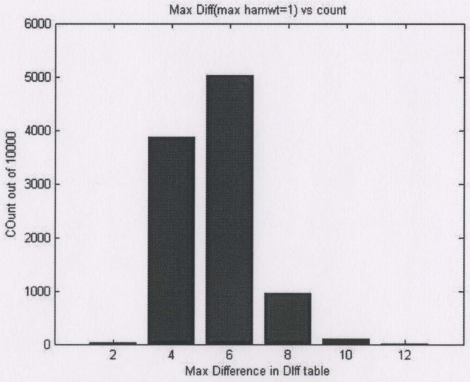

Figure 4.4: Histogram Showing Maximum Value in Difference-table Versus Count When Hamming Weight =1.

The TRI algorithm was run for 10 ciphers based on random S-boxes number R-1 to R-10 as shown in Table 4.16. This table also includes the results for mathematically structured S-boxes like AES and Camellia S-boxes numbered CAM-1 to CAM-4 and also good S-boxes that have low maximum values in bias-table and difference-table for Hamming weight equal to one. The 4 good S-boxes are numbered from GR-1 to GR-4. From the table we can see that for the 10 random S-boxes the maximum bias value in the table is in thirties for 90% of the time as expected from Figure 4.1. Similarly, for 100% of the time, the maximum value in the difference-table is between 10 and 12, similar to what is expected from Figure 4.2. We can see that for mathematically structured S-boxes like AES and Camellia, the maximum values in the bias-table and difference-table are much less, and this is to be expected since these S-boxes were constructed to have good linear and differential properties. The good S-boxes have a value comparable to AES and Camellia for Hamming weight equal to one; however they differ significantly for overall maximum value in bias-table and difference-table. This is because AES and Camellia have good values spread out consistently in the tables, while the good S-box is a randomly selected S-box to have low values for only Hamming weight equal to one.

A seven round approximation is determined using the TRI algorithm and the resulting biases and differential probabilities are shown in Table 4.17. The TRI algorithm was run for fixed L=8000 for all the ciphers. From this table we can see that the good S-boxes give the best results in terms of bias and even outperform the AES and Camellia S-box based networks. However, the difference is very small. Similarly, we can also see that in 50% of the cases a random S-box based network gives a comparable result to the good S-boxes and the mathematically structured S-boxes based network.

In the case of differential cryptanalysis, AES gives the best differential probability. Nevertheless, the results of the random S-boxes and the good S-boxes are also comparable for Cipher-B. We can also see that for mathematically structured S-boxes the solution is very close to the loose upper bound. This is because the maximum values for the AES and Camellia S-boxes are evenly spread out in the bias-table and the difference-table. However for the other S-boxes, the upper bound is several factors higher than the linear and differential solutions. This is because the maximum value in the tables is not actually used during the approximation in many cases.

Cipher	Max. value in bias-table	Max. value (hamming wt =1,2 or 3)	Max. value (hamming wt=1)	Max. difference in difference-table	Max. difference (hamming wt==1,2 or 3)	Max. difference (hamming wt=1)
R-1	38	32	22	12	10	6
R-2	38	30	22	12	10	4
R-3	34	30	20	10	10	6
R-4	36	28	26	10	10	4
R-5	34	32	22	12	12	6
R-6	32	32	18	10	10	4
R-7	42	32	22	12	12	6
R-8	38	36	18	10	8	6
R-9	36	34	20	12	12	4
R-10	34	32	22	12	10	6
AES	16	16	16	4	4	2
CAM-1	16	16	14	4	4	4
CAM-2	16	16	14	4	4	4
CAM-3	16	16	14	4	4	4
CAM-4	16	16	14	4	4	4
GR-1	32	30	14	10	10	4
GR-2	32	32	12	10	8	4
GR-3	34	32	12	12	10	4
GR-4	36	34	14	14	14	2

Table 4.16: Maximum Value in Bias-table and Difference-table of Different Hamming Weight for Random and Good S-boxes for Cipher-B

Cipher	Max. bias (7 round)	Theoretical Upper Bound bias (7 round)	Max. differential probability(7 round)	Theoretical Upper Bound differential probability(7 round)
R-1	1.337e-6	1.016e-4	6.821e-13	1.387e-10
R-2	4.754e-7	1.016e-4	6.821e-13	1.387e-10
R-3	3.830e-7	4.665e-5	1.151e-12	1.387e-10
R-4	1.344e-6	6.960e-5	9.094e-13	1.387e-10
R-5	3.781e-7	4.665e-5	1.534e-12	4.972e-10
R-6	4.516e-7	3.052e-5	6.821e-13	1.387e-10
R-7	9.104e-7	2.047e-4	9.094e-13	4.972e-10
R-8	1.432e-6	1.016e-4	9.592e-13	2.910e-11
R-9	1.230e-6	6.960e-5	8.526e-14	4.972e-10
R-10	1.208e-6	4.665e-5	3.453e-12	1.387e-10
AES	1.788e-7	2.384e-7	7.105e-15	2.273e-13
CAM-1	1.223e-7	2.384e-7	5.684e-14	2.273e-13
CAM-2	1.223e-7	2.384e-7	2.842e-14	2.273e-13
CAM-3	1.223e-7	2.384e-7	5.684e-14	2.273e-13
CAM-4	1.223e-7	2.384e-7	2.842e-14	2.273e-13
GR-1	2.737e-7	3.052e-5	9.094e-13	2.910e-11
GR-2	4.638e-8	3.052e-5	6.812e-13	2.910e-11
GR-3	1.349e-8	4.665e-5	5.116e-13	4.972e-10
GR-4	1.720e-7	6.960e-5	2.664e-14	1.462e-9

Table 4.17: Maximum Bias and Differential Probability for 7 Round Approximation using TRI for fixed L=8000 on a Cipher-B Network.

Further results for linear approximation are shown in Table 4.18 based on the TRI algorithm by varying L for 20 random S-boxes. The linear approximation is of 7 rounds. From the table we can interpret that by increasing the value of L, the bias value also increased and is somewhat close to the upper bound in many cases. The results also indicate that in many cases only one S-box is involved during the approximation. Although we cannot know for sure, we conjecture that the results for larger L are close to the optimal result. An important conclusion can also be drawn that there is little difference in bias value of a random selected S-boxes and the mathematically structured S-boxes like AES and Camellia. From Table 4.17, we can see that the bias value of the

randomly selected S-boxes have almost the same order of magnitude as AES and Camellia.

Cipher	Upper Bound (7 rounds)	TRI (L=1000)	TRI (L=100)	TRI (L=50)
1	6.96e-5	1.394e-6	8.580e-7	8.044e-7
2	6.96e-5	1.655e-6	1.655e-6	1.655e-6
3	1.01e-4	6.930e-7	5.720e-7	3.782e-7
4	4.66e-5	8.380e-7	3.860e-7	2.200e-7
5	3.05e-5	7.039e-7	7.039e-7	2.693e-7
6	3.05e-5	3.094e-6	3.094e-6	3.094e-6
7	4.66e-5	4.585e-7	2.292e-7	2.062e-7
8	6.96e-5	7.603e-7	7.603e-7	5.865e-7
9	6.96e-5	9.547e-7	9.547e-7	9.547e-7
10	6.96e-5	2.819e-6	1.245e-6	1.245e-6
11	3.05e-5	1.773e-6	1.300e-6	1.300e-6
12	4.66e-5	1.121e-6	1.035e-6	9.613e-7
13	4.66e-5	1.047e-6	8.145e-7	8.145e-7
14	4.66e-5	4.061e-7	7.459e-8	1.131e-8
15	4.66e-5	1.616e-6	6.384e-7	6.188e-7
16	3.05e-5	4.073e-6	2.036e-6	2.036e-6
17	6.96e-5	6.630e-7	1.519e-7	1.519e-7
18	4.66e-5	6.223e-7	4.243e-7	3.889e-7
19	4.66e-5	2.071e-6	1.164e-6	9.415e-7
20	3.05e-5	1.591e-6	1.591e-6	1.591e-6

Table 4.18: Maximum Bias of 7 Round Approximations for Cipher-B Network

It is interesting to note the execution time for different values of L. Table 4.19, indicates that there is not much difference in the execution time for L=1000 and L=50. If we compare the results with the results obtained for Cipher-A networks using the TRI algorithm, there was significant amount of time difference for the different L values. The reason is that in Cipher-B networks most of the time is spent in deriving the list value after the initial two rounds. The algorithm has to look into almost 42 million inputs in round one and select the best two round inputs of length L after the first two rounds. For

the rest of the two round steps the algorithm will take much less time because it considers only L input values to find the next L input values for the next two rounds. Hence, by increasing the number of rounds in the cipher for any given L, the change in the execution time will not be of significant factor. Similarly by varying L, for a fixed number of rounds in the cipher, the change in the execution time is not that prominent as shown in Table 4.19. For example for L=50, the average execution time is approximately 28 minutes while for L=1000, the average execution time is approximately 40 minutes. Similar results were achieved for differential cryptanalysis. The results are shown in Appendix (Table A.4 and A.5).

Cipher	TRI (L=1000) Time(min:sec)	TRI (L=100) Time(min:sec)	TRI (L=50) Time(min:sec)
1	44:01	31:44	30:27
2	40:29	27:41	26:51
3	45:47	28:30	31:15
4	40:08	28:30	25:43
5	38:46	28:07	25:25
6	38:24	27:28	27:33
7	43:46	26:37	25:53
8	39:08	27:55	26:44
9	44:44	35:00	29:04
10	44:22	29:20	27:32
11	46:08	27:17	26:00
12	43:53	27:10	26:20
13	43:46	29:56	26:00
14	52:35	29:49	27:08
15	42:36	27:43	26:17
16	39:11	28:18	26:56
17	44:53	30:00	27:11
18	44:53	27:32	26:43
19	39:53	27:59	27:58
20	44:32	27:20	27:04

Table 4.19: Execution Time in 7 Round Linear Approximation for 20 Different Cipher-B Networks using Random S-boxes

4.3 Results for Cipher-C

In Section 4.2, we discussed the results obtained for a 64-bit SPN structure using 8×8 S-boxes. In this section we will again look into the results obtained for a 64-bit SPN structure but using 4×4 S-boxes. The results in this section show the applicability of our algorithm on different SPN structures. The random S-boxes used in Section 4.1 are used in this section too. We have assumed that the maximum 3 active S-boxes are involved in each round to find the maximum bias or differential probability.

Cipher	Upper Bound (7 rounds)	TRI (L=1000)	TRI (L=100)	TRI (L=50)
1	.06674	.00330	.00330	.00330
2	.06674	3.662e-4	3.662e-4	1.144e-5
3	.06674	.00585	.00585	.00585
4	.06674	.00164	.00164	5.493e-4
5	.06674	9.765e-4	9.765e-4	1.220e-4
6	.06674	.00390	.00390	.00390
7	.06674	.00146	7.324e-4	7.324e-4
8	.06674	.01318	.01318	.01318
9	.06674	.06674	.0222	.01407
10	.06674	.00390	.00390	.00198
11	.06674	.0222	.0197	.01318
12	.00390	.00195	.00195	4.882e-4
13	.00390	.00390	4.882e-4	4.882e-4
14	.06674	.00123	.00123	.00123
15	.06674	.00219	.00219	9.269e-4
16	.06674	.00146	.00146	.00146
17	.06674	.00390	9.765e-4	9.765e-4
18	.06674	.00390	9.765e-4	2.441e-4
19	.06674	4.882e-4	2.441e-4	2.441e-4
20	.06674	.00390	.00390	.00390

Table 4.20: Maximum Bias of 7 Round Linear Approximation for Cipher-C using Random S-boxes

Table 4.20 illustrates the bias determined by applying the TRI algorithm to 20 different Cipher-C networks using random S-boxes. From the Table 4.20, we can see that

in 65% of the cases the result obtained for L=1000 and L=100 are the same. The result shown in column two of the table refers to the weak upper bound that involves just one S-box per round and the maximum bias of the S-box. For cipher 9 we can see that for L=1000, the result is equal to the upper bound which is the exceptional case and indicates that the cipher is weak. The results in Table 4.21 suggest that by increasing the length of the buffers to store the good solutions the execution time also increases and in some cases by a good margin. If we consider cipher 8 in Table 4.21, we can see that execution time for L=1000 is 45 minutes, while the execution time for L=50 is 3 minutes to get the same result.

Cipher	TRI (L=1000) Time(min:sec)	TRI (L=100) Time(min:sec)	TRI (L=50) Time(min:sec)
1	110:48	83:36	72:37
2	20:23	19:34	20:02
3	16:13	15:32	13:33
4	18:04	17:34	16:46
5	95:18	78:24	63:29
6	55:06	50:23	39:06
7	44:17	16:33	16:39
8	45:29	5:00	3:06
9	20:55	3:20	2:25
10	90:17	30:06	39:05
11	33:48	2:31	2:08
12	25:29	20:17	5:39
13	70:39	43:04	35:22
14	52:52	31:12	18:56
15	74:32	38:48	37:49
16	315:19	11:48	5:09
17	64:59	36:38	36:36
18	70:46	45:28	43:21
19	78:21	57:42	50:36
20	62:20	37:11	38:48

Table 4.21: Execution Time in 7 Round Linear Approximation for Cipher-C Networks using Random S-boxes

Similarly for differential cryptanalysis, if we analyze Table 4.22, we can see that the results are the same 70% of the cases for L=1000 and L=100. We can also see that for cipher 6 the result is equal to the upper bound. This result also suggests that there might be some ciphers for which the optimal result will be equal to the upper bound. However, if we consider Table 4.23, we can see sharp difference in the execution time for L=1000 and L=100. For example, cipher 13 takes almost 5 hours to get the result with L=1000, but just 45 minutes with L=100 and 26 minutes with L=50 to get the same result. Similarly, we can view that for cipher 16 TRI algorithm for L=1000 has execution time 5 times more as compared to L=100 to get the same result.

Cipher	Upper Bound (7 rounds)	TRI (L=1000)	TRI (L=100)	TRI (L=50)
1	.00104	6.437e-6	1.072e-6	5.364e-7
2	6.103e-5	3.814e-6	1.907e-6	9.536e-7
3	.00104	1.716e-5	1.716e-5	1.287e-5
4	.00104	3.862e-5	3.862e-5	1.716e-5
5	.00104	1.907e-6	1.907e-6	1.907e-6
6	.00104	.00104	.00104	.00104
7	.00781	6.103e-5	6.103e-5	6.103e-5
8	.03725	2.384e-5	2.384e-5	2.384e-5
9	.00104	4.577e-5	4.577e-5	4.577e-5
10	.00104	5.149e-5	5.149e-5	2.574e-5
11	.00104	1.525e-5	7.629e-6	3.814e-6
12	6.103e-5	1.525e-5	7.629e-6	3.814e-6
13	.00104	1.907e-6	1.907e-6	1.907e-6
14	.00104	1.907e-6	1.907e-6	1.907e-6
15	.00104	2.575e-5	3.620e-6	3.620e-6
16	.00104	2.861e-6	2.861e-6	1.430e-6
17	.00104	2.861e-6	2.861e-6	2.861e-6
18	.00104	1.525e-5	1.144e-5	5.722e-6
19	.00104	.00104	.00104	.00104
20	.00781	6.103e-5	6.103e-5	6.103e-5

Table 4.22: Maximum Differential Probability of 7 Round Approximation for Cipher-C using Random S-boxes

Cipher	TRI (L=1000) Time(min:sec)	TRI (L=100) Time(min:sec)	TRI (L=50) Time(min:sec)
1	79:15	48:42	30:06
2	175:13	54:51	14:18
3	93:48	24:06	8:17
4	156:53	13:53	9:34
5	243:05	65:42	60:35
6	50:09	6:13	3:33
7	66:40	8:43	13:15
8	75:28	38:27	29:25
9	155:17	121:13	122:05
10	138:41	9:13	4:39
11	54:00	25:32	19:07
12	147:06	14:40	13:39
13	315:00	41:06	26:44
14	231:10	62:54	58:11
15	62:04	54:09	13:51
16	108:35	21:42	11:37
17	155:12	26:12	25:05
18	77:16	48:04	43:28
19	28:53	9:34	7:43
20	59:06	27:14	25:43

Table 4.23: Execution Time in 7 Round Differential Characteristic for Cipher-C Networks using Random S-boxes

Another interesting aspect is to note that the execution time for the Cipher-C network varies dramatically with the increase in the value of L but the execution time for Cipher-B network doesn't vary much with the increase in the value of L. The reason is that Cipher-B makes use of the 8×8 S-boxes and it has large bias and differential probability variation due to the values present in the bias-table and the difference-table. For example, the values in the bias-table can vary from 0 to 40 as seen in Figure 4.1. Contrary to that, the values in the bias-table for the 4×4 S-box will only vary from 0 to 6. Hence, the greedy approach does not help much in focusing on best approximation /

characteristic. Due to this reason TRI shows different behavior for Cipher-B and Cipher-C when it is executed. In the case of Cipher-B, TRI spends the maximum time in order to find the best approximation for the first two rounds; hence the execution time for different L does not vary sharply. On the other hand for Cipher-C, TRI approximately even time to execute for every two rounds and for increasing L value the execution time varies dramatically. This is because the number of combinations involved to make approximation in first two rounds in Cipher-C is much less compared to Cipher-B.

4.4 Summary

In this chapter we showed the performance of our TRI algorithm for various cipher networks. The cipher networks were changed with the S-boxes for the fixed cipher structures of Cipher-A, Cipher-B and Cipher-C. All the cipher structures we studied are practically realizable ciphers and had good cryptographic properties. We have shown with tables and figures that our TRI algorithm is often successful in finding the optimal solution in practical time for realistically-sized networks.

The TRI algorithm can be used to analyze SPN structures resistance with respect to linear and differential cryptanalysis. With respect to Cipher-A, the TRI algorithm finds the optimal result (i.e. best linear approximation and differential probability) with high likelihood or finds a good non-optimal result. The results from Cipher-A suggest that optimal results tend to involve small number of S-boxes per round. With respect to the 64-bit network, it can be seen that the random S-boxes can provide good properties with low bias and differential probability. From the results it can be concluded TRI is a practical tool to give results on 64-bit realistically sized networks (using either 8×8 or

4×4 S-boxes). The results also suggest that S-boxes can be selected to give results as good as the AES and Camellia S-boxes. Lastly, it can be concluded that TRI is much more efficient than other algorithms (IPM/MM) that are guaranteed to give optimal results. This leads to the next chapter where we will summarize our work and draw conclusions relevant to our work and suggest future directions for the research.

Chapter 5

Conclusions and Future Work

In this chapter, we will summarize our thesis. We will draw conclusions of our work and also discuss the limitations related to our work. The limitations of our work can be further used as a basis for future research work.

In Chapter 2, we discussed a basic SPN architecture. We have analyzed SPNs because they are still widely used in today's modern cipher design in ciphers such as DES and AES. We studied three types of SPNs: a 16-bit SPN, based on 4-bit S-boxes (Cipher-A), and two 64-bit SPNs, one based on 8-bit S-boxes (Cipher-B) and one based on 4-bit S-boxes (Cipher-C). We discussed in detail about the applicability of the two most fundamental attacks on block ciphers, referred to as linear cryptanalysis and differential cryptanalysis. This chapter also provided an overview of the linear attack and the differential attack using the Cipher-A network. (A sample Cipher-A network was used to show the linear approximation and differential characteristic of the cipher.) A good linear attack tries to find the largest bias in the network, while a good differential attack tries to find the maximum differential probability in order to deduce the minimum number of plaintext / ciphertext pairs required to mount the attacks successfully.

In Chapter 3, a number of algorithms have been studied and developed by us related to linear cryptanalysis and differential cryptanalysis. The algorithms find the best bias or differential probability for a block cipher and a corresponding linear approximation or differential characteristic.

Original algorithms developed related to the SPN structure were introduced in this chapter. Two optimal algorithms (IPM and MM), guaranteed to find the largest bias/differential probability, were developed along with the non-optimal TRI algorithm that tries to find the optimal or close to optimal result. Some preliminary investigation showed that IPM and MM algorithm have there own limitations when applied to the SPN architecture. The complexity of both the algorithms increases considerably with the increase of the number of rounds and the number of active S-boxes in each round. Hence, both the IPM and MM algorithm were only successful for 16-bit SPN and cannot be applied to the larger sized networks (like a 64-bit SPN).

In order to overcome the limitations of the optimal algorithms, the heuristic TRI algorithm was developed. The TRI algorithm tries to find the optimal or close to optimal results. A significant achievement of the TRI algorithm is that the complexity of algorithm increases linearly with the increase in the number of rounds. Thus, we can analyze practically sized ciphers that include a large number of rounds. The search is reduced drastically because the tree search is performed only on a maximumof two rounds; hence the number of nodes contributing in the tree structure is reduced. The TRI algorithm is shown to be scalable to large networks like Cipher-B and Cipher-C. Hence, the TRI algorithm is a useful tool to look into properties of S-boxes and cipher structures that are necessary for good cryptographic resistance to linear and differential cryptanalysis.

In Chapter 4, a discussion was involved by analyzing the results for Cipher-A, Cipher-B and Cipher-C by running the IPM, MM and TRI algorithms. The performance of the TRI algorithm was also evaluated by running it on various cipher networks. The

cipher networks were changed by modifying the S-boxes for the fixed cipher structures of Cipher-A, Cipher-B and Cipher-C. All the cipher structures we studied are practically realizable ciphers and had good cryptographic properties. We have shown with tables and figures that our TRI algorithm is often successful in finding the optimal solution in practical time on realistically sized networks.

TRI can be used to analyze SPN structures resistance with respect to linear and differential cryptanalysis. With respect to the 16-bit network, the TRI algorithm finds the optimal result (i.e. best linear approximation and differential probability) with high likelihood or finds a good non-optimal result. The results from Cipher-A suggest that optimal results tend to involve small number of S-boxes per round. With respect to the 64-bit network, it can be seen that the random S-boxes can provide good properties with low bias and differential probability. From the results it can be concluded that TRI is a practical tool to give results on realistically sized 64-bit networks (using either 8×8 or 4×4 S-boxes). The results also suggest that S-boxes can be selected to give results as good as the AES and Camellia S-boxes. Lastly, it can be concluded that TRI is much more efficient than other algorithms (IPM/MM) that are guaranteed to give the optimal result.

During the implementation of the TRI algorithm on realistically sized ciphers (e.g. 64-bit), we have assumed that not more than 3 active S-boxes are involved in each round in order to find the best linear approximation and differential characteristic of the cipher. This assumption is made in order to reduce the search operations in the tree structure and to get the result in practical time. Even though this assumption is valid for most of the cipher cases, it cannot be generalized. Hence, one of the future works is to

implement an algorithm that considers all the possible combination of S-boxes in each round and still gives result in practical time. In our work we just considered two basic realistically-sized ciphers (Cipher-B and Cipher-C) and tested the effectiveness of the TRI algorithm on them. Another future research can be testing the effectiveness of TRI algorithm on different kinds of practically realizable networks (e.g. 128-bit SPN or other 64-bit ciphers). Also, an important future research can be the application of TRI to AES and other actually proposed ciphers.

References:

[1] J. Garms, D. Somerfield, *Professional Java Security*, 1st ed., Wrox Press, 2001.

[2] A. J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

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

[4] W. Stallings, *Cryptography and Network Security: Principles and Practices*, 3rd ed., Prentice Hall, 2005.

[5] L. Knudsen, "Block Ciphers: A Survey", *State of the Art in Applied Cryptography: Course on Computer Security and Industrial Cryptography (Lecture Notes in Computer Science no. 1528)*, Springer-Verlag, pp. 18-48, 1998.

[6] A. Youssef, S.E. Tavares, and H.M. Heys, "A New Class of Substitution-Permutation Networks", *Workshop on Selected Areas in Cryptography (SAC '96)*, Queen's University, Kingston, Ontario, Aug. 1996.

[7] M. Matsui, "Linear Cryptanalysis Method for DES Cipher", *Advances in Cryptology- EUROCRYPT '93 (Lecture Notes in Computer Science no. 765)*, Springer-Verlag, pp. 386-397, 1994.

[8] M. Matsui, "The First Experimental Cryptanalysis of the Data Encryption Standard", *Advances in Cryptology - CRYPTO '94 (Lecture Notes in Computer Science no. 839)*, Springer-Verlag, pp. 1-11, 1994.

[9] E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", *Journal of Cryptology*, vol. 4, no. 1, pp. 3-72, 1991.

[10] H. Feistel, "Cryptography and Computer Privacy", *Scientific American*, vol. 228, no. 5, pp. 15-23, 1973.

[11] C.E. Shannon, "Communication Theory of Secrecy Systems", *Bell System Technical Journal*, vol. 28, no. 4, pp 656-715, 1949.

[12] A.M. Youssef, S. Mister, and S.E. Tavares, "On the Design of Linear Transformations for Substitution Permutation Encryption Networks", *Workshop on Selected Areas of Cryptography (SAC '96): Workshop Record*, pp. 40-48,1997.

[13] L. Keliher, H. Meijer, S. Tavares, "Modeling Linear Characteristics of Substitution-Permutation Networks", *Selected Areas in Cryptography (Lecture Notes in Computer Science no. 1758)*, Springer-Verlag, pp. 78-91, 1999.

[14] H.M. Heys, "The Design of Substitution-Permutation Network Cipher Resistant to Cryptanalysis", *Ph.D. Thesis*, Queen's University, Kingston, Canada, 1994.

[15] H.M. Heys and S.E. Tavares, "Substitution-Permutation Networks Resistant to Differential and Linear Cryptanalysis", *Journal of Cryptology*, vol. 9, no. 1, pp. 1-19, 1996 (also presented at *2nd ACM Conference on Computer and Communications Security*, Fairfax, Virginia, Nov. 1994).

[16] H. M. Heys, "A Tutorial on Linear and Differential Cryptanalysis", *Technical Report CORR 2001-17, Centre for Applied Cryptographic Research, Department of Combinatorics and Optimization*, University of Waterloo, Mar. 2001. (Also appears in *Cryptologia*, vol. XXVI, no. 3, pp. 189-221, 2002.)

[17] J. Daemen and V. Rjimen, "AES Proposal: Rijndael", *First Advanced Encryption Standard (AES) Conference*, California, Aug. 1998.

[18] National Institute of Standards, Advanced Encryption Standard (AES) web site: www.nist.gov/aes.

[19] K. Nyberg, "Linear Approximations of Block Ciphers", *Advances in Cryptology - EUROCRYPT '94 (Lecture Notes in Computer Science no. 950)*, Springer-Verlag, pp. 439-444, 1995.

[20] E. Biham, "On Matsui's Linear Cryptanalysis", *Advances in Cryptology - EUROCRYPT '94 (Lecture Notes in Computer Science no. 950)*, Springer-Verlag, pp. 341-355, 1995.

[21] M. Hellman and S. Langford, "Differential-Linear Cryptanalysis", *Advances in Cryptology - CRYPTO '94 (Lecture Notes in Computer Science no. 839)*, Springer-Verlag, pp. 26-39, 1994.

[22] F. Chabaud and S. Vaudenay, "Links Between Differential and Linear Cryptanalysis", *Advances in Cryptology - EUROCRYPT '94 (Lecture Notes in Computer Science no. 950)*, Springer-Verlag, pp. 356-365, 1995.

[23] L.R. Knudsen, "Truncated and Higher Order Differentials", *Fast Software Encryption (Lecture Notes in Computer Science no. 1008)*, Springer-Verlag, pp. 196-211, 1995.

[24] M. Matsui, "On Correlation Between the Order of S-boxes and the Strength of DES", *Advances in Cryptology - EUROCRYPT '94 (Lecture Notes in Computer Science no. 950)*, Springer-Verlag, pp. 366-375, 1995.

[25] R.L. Kruse, *Data Structure and Program Design*, 3rd ed., Prentice Hall, 1994.

[26] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein, *Introduction to Algorithms*, 2nd ed., MIT Press and McGraw-Hill, 2001.

APPENDIX

Input	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S-1	E	B	1	6	3	5	A	F	2	8	0	7	4	D	C	9
S-2	4	B	0	E	7	D	F	3	8	1	C	6	5	A	2	9
S-3	7	B	8	1	B	9	3	C	A	5	6	2	F	E	0	4
S-4	1	B	E	C	3	A	4	0	7	D	2	9	6	8	5	F
S-5	D	B	2	1	9	7	F	0	6	C	4	5	3	E	A	8
S-6	F	B	5	9	3	C	1	A	8	E	2	0	7	D	6	4
S-7	4	B	A	5	7	E	9	F	1	0	D	C	6	3	8	2
S-8	D	B	F	9	3	2	C	7	8	A	6	E	4	1	5	0
S-9	8	B	4	C	D	9	F	E	2	6	5	0	1	7	A	3
S-10	4	B	7	0	6	A	3	5	D	E	1	2	F	C	9	8
S-11	A	B	6	3	0	E	4	8	7	F	5	2	9	C	D	1
S-12	E	B	4	3	7	F	9	0	6	5	2	D	A	8	C	1
S-13	1	B	5	7	C	2	6	D	0	F	E	9	3	A	4	8
S-14	0	B	8	E	9	5	4	1	2	7	D	A	3	C	F	6
S-15	4	B	C	1	6	E	0	2	A	7	9	8	5	D	3	F
S-16	3	B	8	C	A	7	E	9	D	2	0	5	1	4	F	6
S-17	F	B	C	9	1	6	D	0	3	A	8	7	4	E	2	5
S-18	B	0	F	3	2	5	7	E	C	A	9	1	4	6	8	D
S-19	A	B	4	5	F	E	8	3	1	6	0	C	D	9	2	7
S-20	D	B	9	3	F	C	2	A	4	7	8	1	5	0	6	E

Table A.1: S-box Representation (in hexadecimal) of 20 Random 4×4 S-boxes

Input	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S-1	E	4	D	1	2	F	B	6	3	A	6	C	5	9	0	7
S-2	F	1	8	E	6	B	3	4	9	7	2	D	C	0	5	A
S-3	3	D	4	7	F	2	8	E	C	0	1	A	6	9	B	5
S-4	A	0	9	E	6	5	F	5	1	D	C	7	B	4	2	8
S-5	2	C	4	1	7	A	B	6	8	5	3	F	D	0	E	9

Table A.2: S-box Representation (in hexadecimal) of 5 DES S-boxes

Cipher (DES)	IPM(Optimal) Time(min:sec)	TRI (L=4000) Time(min:sec)	TRI (L=1000) Time(min:sec)	TRI (L=100) Time(min:sec)	TRI (L=50) Time(min:sec)
1	23:00	7:00	2:00	0:30	0:20
2	52:00	7:00	1:30	0:25	0:20
3	67:00	9:00	2:00	1:00	0:40
4	8:00	14:00	2:00	1:00	0:08
5	3:00	8:00	1:30	0:20	0:20

Table A.3: Execution Time Required in 6 round Approximation of 5 Cipher-A Networks using DES S-boxes.

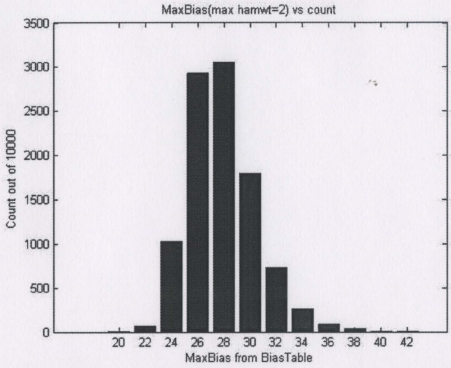

Figure A.1: Histogram Showing Maximum Value in Bias-table Versus Count When Hamming Weight =2

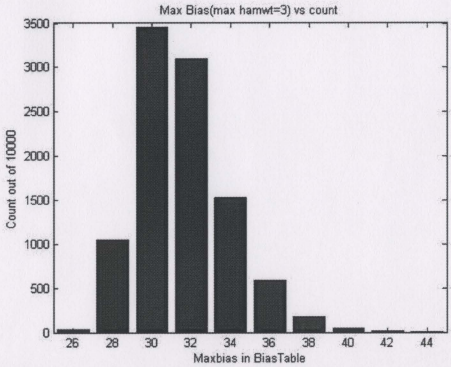

Figure A.2: Histogram Showing Maximum Value in Bias-table Versus Count When Hamming Weight =3

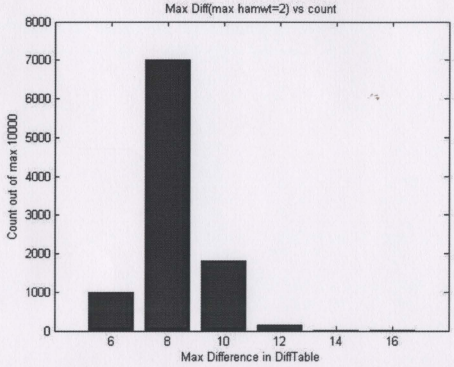

Figure A.3: Histogram Showing Maximum Value in Difference-table Versus Count When Hamming Weight =2

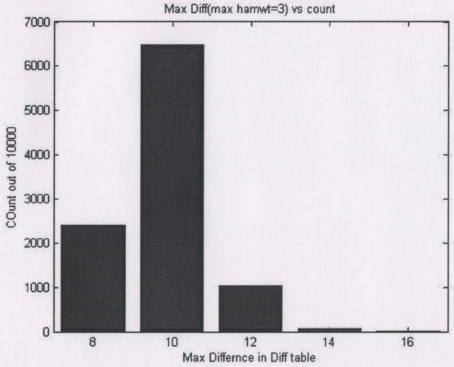

Figure A.4: Histogram Showing Maximum Value in Difference-table Versus Count When Hamming Weight =2

Cipher	Upper Bound (7 rounds)	TRI (L=1000)	TRI (L=100)	TRI (L=50)
1	4.972e-10	1.035e-11	1.705e-12	5.755e-13
2	1.463e-9	6.821e-13	8.526e-14	7.105e-15
3	1.388e-10	2.302e-12	2.302e-12	1.332e-14
4	1.388e-10	3.069e-12	1.364e-12	1.364e-12
5	1.388e-10	3.069e-12	2.046e-12	2.046e-12
6	4.972e-10	6.821e-13	2.131e-14	3.996e-15
7	4.972e-10	1.023e-12	8.526e-13	8.526e-13
8	1.388e-10	1.534e-12	1.534e-12	6.821e-13
9	4.972e-10	7.760e-12	3.830e-12	3.830e-12
10	4.972e-10	2.557e-12	2.557e-12	2.557e-12
11	4.972e-10	1.364e-12	1.364e-12	1.364e-12
12	4.972e-10	3.069e-12	3.069e-12	1.332e-13
13	1.388e-10	1.023e-12	6.821e-13	9.094e-13
14	1.463e-9	1.705e-12	1.705e-12	1.705e-12
15	1.463e-9	3.069e-12	2.301e-12	1.534e-12
16	4.972e-10	3.410e-12	1.534e-12	8.526e-13
17	1.463e-9	2.775e-10	1.110e-10	1.110e-10
18	1.388e-10	1.035e-11	4.604e-12	4.604e-12
19	1.388e-10	1.023e-12	8.526e-14	8.526e-14
20	1.388e-10	1.227e-11	1.091e-11	1.091e-11

Table A.4: Maximum Differential Probability for 7 Round Approximation of Cipher-B Network

Cipher	TRI (L=1000) Time(min:sec)	TRI (L=100) Time(min:sec)	TRI (L=50) Time(min:sec)
1	36:07	31:41	31:27
2	49:07	35:11	30:46
3	45:30	31:30	30:34
4	41:52	30:52	30:10
5	38:31	30:21	30:06
6	38:37	30:44	34:06
7	44:04	29:54	29:11
8	37:41	30:32	29:52
9	43:26	29:37	28:46
10	42:19	30:41	29:51
11	43:21	30:36	30:15
12	41:31	31:56	31:16
13	48:08	31:14	30:49
14	39:51	30:51	29:58
15	41:30	31:43	31:45
16	37:03	33:34	40:22
17	45:42	29:49	29:15
18	39:04	30:31	30:28
19	41:47	31:33	29:56
20	42:24	30:04	29:33

Table A.5: Execution Time Required for 7 Round Differential Characteristic of 20 Different Cipher-B Networks using Random S-boxes