Rochester Institute of Technology
Department of Computer Science

Differential Cryptanalysis of Substitution Permutation Networks and Rijndael-like Ciphers

Master's Project Report

Gnanasekaran Sakthivel
`http://www.cs.rit.edu/~ gxs5626`
gxs5626@cs.rit.edu

Chairman: Prof. Stanisław P. Radziszowski	Date
Reader: Prof. Christopher M. Homan	Date
Observer: Prof. Rajendra K. Raj	Date

Contents

Abstract

Generally, in block ciphers that follow the strategy of Substitution Permutation Networks (SPN), a round transformation includes three components: a nonlinear substitution, a linear diffusion and a key addition. In order to strengthen the cipher against linear and differential cryptanalysis, most of the researches has been focused on the efficient design of substitution component. This project focuses on the role of permutation component in SPN against Differential Cryptanalysis. Another goal of this project is to design tools to configure and perform differential cryptanalysis on SPN and Rijndael-like ciphers.

Chapter 1

Background

1.1 Introduction

Both Substitution Permutation Networks (SPN) and Rijndael are block ciphers. In general, block ciphers are formed as a product of nonlinear (confusion) and linear (diffusion) functions.

1.2 Block ciphers

In this report, the term cipher refers to a cryptographic algorithm, mostly the encryption algorithm. In general, block cipher algorithms consists of repetitions of a weak block cipher, known as a round transformation [33]. Round transformations used in block ciphers are formed by combining two cryptosystems that have same plaintext and ciphertext space. For example, round transformations in SPN and Rijndael are products of a nonlinear substitution and a linear diffusion component.

Block cipher also can be defined as a set of boolean transformations operating on n_b-bit vectors (which are called blocks) [15]. It transforms plaintext blocks of a fixed length n_b to ciphertext blocks of the same length under the influence of a cipher key k. Usually the boolean transformation of block ciphers is divided into three layers, namely substitution, diffusion and key mixing. This transformation is key-dependent. The size of the key may determine the number of transformations.

As we said earlier, security of this kind of systems is achieved by repeatedly applying the transformation. In order to encrypt, the input message will be

divided into plaintext blocks whose size is equal to the block size of the cipher. In Data Encryption Standard (DES) [35], the predecessor to AES [10, 28], the block size is 64 bits. In Rijndael, we have 3 variable block sizes namely 128, 192 and 256, whereas in case of AES (a standard chosen and published by NIST), the block size is fixed to 128 bits [10].

1.3 DES and AES

Data Encryption Standard was the first known encryption standard available for public and corporate use. The original idea behind DES was developed by IBM in 1960s. Then it was adopted as a standard by National Institute of Standards and Technology in 1977.

DES is a special type of iterated cipher called a Feistel cipher [35]. It is a 16 round cipher having a block length of 64 bits. It uses a 56-bit key.

This standard was secure until mid 1990. Then it was found insecure mostly because of its short key size. In 1998, Electronic Frontier Foundation has designed a DES cracker hardware which can exhaustively search and find the key in less than 3 days.

Other than exhaustive key search, linear and differential cryptanalysis are the two most important attacks against DES. These attacks are not practically efficient, because they require large number of plaintext-ciphertext pairs to mount the attack.

In 1997, NIST has planned to replace DES by AES. AES requires a cipher of block length 128 bits. The cipher should support variable key lengths: 128, 192 and 256 bits. Out of the 21 algorithms submitted from worldwide, after two rounds of the competition, five algorithms were chosen for the final round. MARS, RC6, Rijndael, Serpent and Twofish were the five finalists [35].

In the final round, Rijndael, which is developed by Joan Daemen and Vincent Rijmen, was chosen as AES [9]. AES was adopted as a standard in November 2000. All five final algorithms were found to be secure. Rijndael is found to be superior in its combined performance of all the required aspects: security, performance, efficiency, implementability and flexibility.

1.4 Key alternating and Iterated ciphers

In iterated block ciphers, the transformations are iterated many times. Each iteration is called a round, and the corresponding transformation is called the round transformation. The round transformations are key-dependent. Each round transformation may or may not be unique. One round transformation may be different from the cipher's other round transformations. Each round will have a different key, and the round keys are computed from the cipher key [9]. The algorithm used to derive the individual round keys from the cipher key is called the key schedule algorithm. This block cipher has same round transformation in all of its rounds (except for the first and the last rounds which may be slightly different for security reasons) [9]. If the round transformation of the iterated block cipher is not dependent on the round key, then it is a key alternating cipher. The cipher is considered as an alternated application of the round transformation and the key addition. Usually, the key addition component is a simple XOR operation. [9].

1.5 Cryptanalysis

Cryptanalysis is a process of finding some weakness in the cipher and extract partial or whole key bits used in the cipher. All the cryptanalysis techniques assume that the cipher algorithm is known and public. The only thing that is not known is the key used in the cipher. The cipher is considered broken or not secure if one can be able to extract the key with a time complexity less than the time taken by exhaustive search. There are many cryptanalytic attacks. Some attacks are applicable to only one particular encryption algorithm. Some of the attacks are applicable to a class of encryption algorithms. In general, cryptanalytic attacks can be categorized based on the information available to the analyst with respect to the attack.

ciphertext only attack In this type, the analyst has a ciphertext from the encryption system. The plaintext corresponding to the ciphertext does not really matter in this context. The analyst will be trying to attack the cipher only with this information.

known plaintext attack In this type of attack, the analyst has a plaintext and its corresponding ciphertext.

chosen plaintext attack This type of attack assumes that the analyst has a temporary access to the encryption system, so that he or she

can choose the plaintext to be encrypted. The analyst chooses the plaintexts and gets the corresponding ciphertexts.

chosen ciphertext attack This type of attack is the same as the chosen plaintext attack, except that the analyst has access to the decryption system and he or she can choose the ciphertexts and can get the corresponding plaintexts.

1.6 Differential cryptanalysis

Differential cryptanalysis is a chosen plaintext attack. It first targets to get the partial key bits of the last round key. In order to perform the analysis, we choose a set of plaintext pairs, where all the pairs have a given xor-difference. We feed these special pairs into the encryption system and tracks the flow of the xor-difference of the (intermediate) texts at each round until the input of the last round. Eventually, given an xor-difference d_1 at the first round input, we expect an xor-difference d_2 at the last round input with some probability p. For each pair of ciphertexts corresponding to the chosen plaintext pair, we then do partial decryption (decryption until the last round input) with all possible combinations of chosen key bits. There is a count associated with each possible key. The count corresponding to a key is equal to the number of ciphertext pairs resulted in the expected xor-difference at the last round input after partial decryption. The key with the maximum count will be likely the correct key. The number of plaintext pairs required to extract the targeted partial key bits is the complexity of the attack.

We know that the pair of texts pass through three components of the cipher at each of the rounds. We will discuss the xor-difference of the texts that goes in and out of these components. Since the substitution component is nonlinear, there is a probability associated with each pair (D_{x_1}, D_{y_1}), where D_{x_1} is the xor-difference of the pair of input texts and D_{y_1} is the xor-difference of the corresponding pair of output texts. Xor-difference across the linear diffusion layer has no effect since xor of diffusion of two texts is equal to the diffusion of xor of two texts. Viewing the xor-difference of the texts at the key addition cancels the key bits involved since same key bits are used for both texts of the pair and xor-ed. Further details about the differential cryptanalysis can be found in chapter 5.

1.7 This project

Differential cryptanalysis is proven successful in cryptanalyzing SPN. It is theoretically efficient in attacking DES. Rijndael is secure against this attack as it is designed taking this attack into consideration. In chapter 5, we have illustrated how to perform differential cryptanalysis on SPN.

In this project, we developed tools to create SPN and Rijndael-like ciphers and to perform differential analysis on them. In case of SPN, we found the differential cryptanalysis ineffective if the round transformation has no or blockwise permutation. The case of no or blockwise permutation introduces certain uniformity in the cipher which makes the differential analysis ineffective. We also described the importance of permutation against differential cryptanalysis (refer Chapter 6). In case of Rijndael-like ciphers, we found the attack ineffective if the diffusion component of the round transformation follows the Rijndael strategy (refer Chapter 7).

Chapter 2

Substitution Permutation Networks

2.1 Introduction

Substitution Permutation Network is a primitive model for all block ciphers. It is an iterated and key-alternating block cipher. SPN can be used to learn various aspects of differential analysis, because differential analysis on SPN is practically feasible. Each round transformation has three components: Substitution, Permutation and Key addition (Figure 2.1). Substitution is a nonlinear component which performs confusion of bits at sub-block level and permutation is a linear component which performs diffusion at bit level. Substitution in the cipher is accomplished using an s-box. So s-box means nothing but substitution. We define SPN as described in the references [35, 15]. Here we give a brief definition of the SPN cipher.

Figure 2.1 is an SPN with 4 rounds, 16-bit blocksize and a 4-bit s-box. In Figure 2.1, K^i refers to the key at round i, S^i_j refers to the j^{th} s-box at round i, x refers to the plaintext block, y refers to the corresponding ciphertext block, u^i refers to the substitution input text at round i (4 s-boxes at one round together constitutes the whole substitution at that round) , v^i refers to the output of substitution at round i or to the input of permutation at round i and w^i refers to the output of round i.

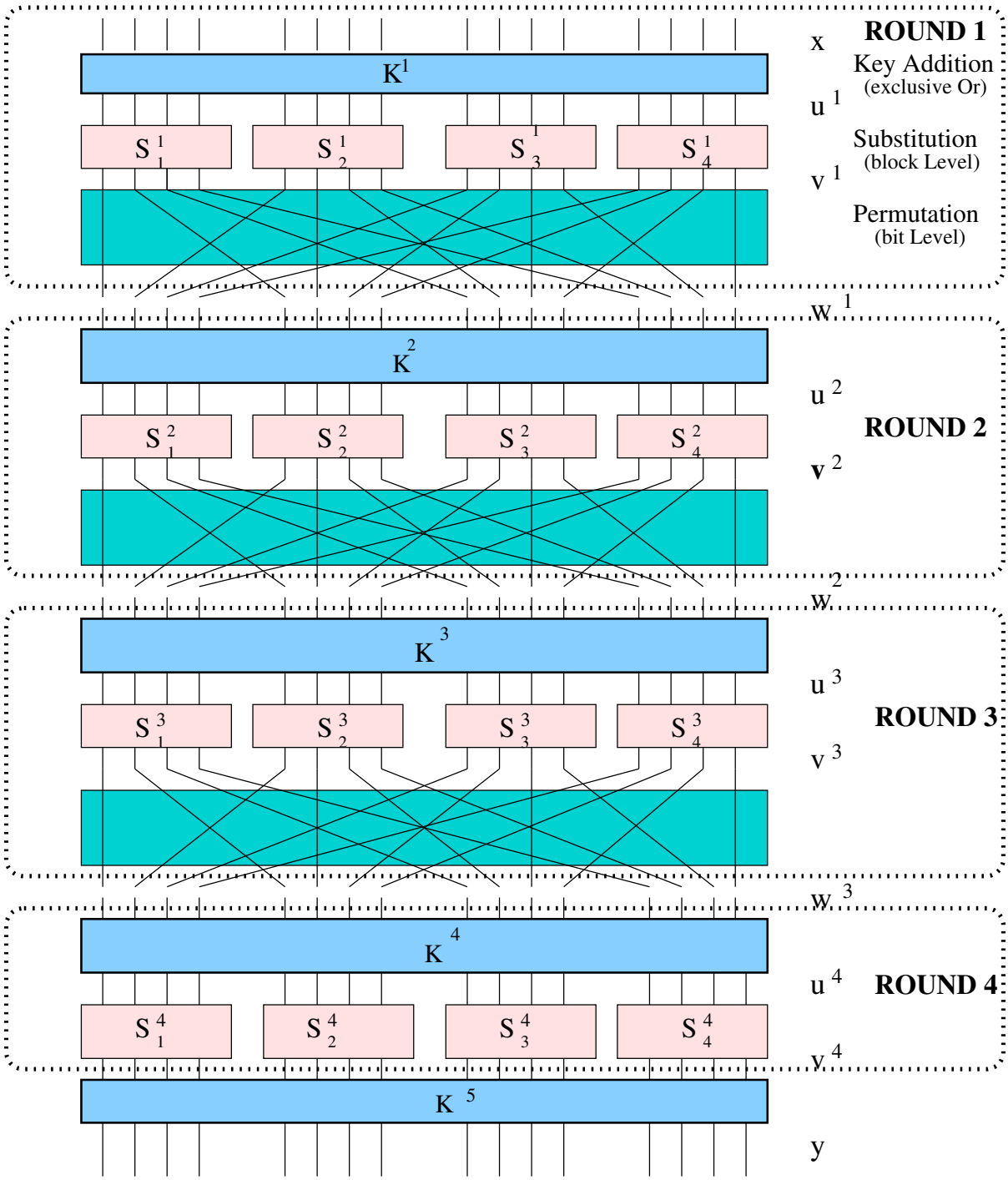

Figure 2.1: SPN: Nr = 4, s-box block size = 4 bits, Block size = 16 bits

2.2 Definition and description

Let l, m and N_r be positive integers, and $\pi_s : \{0,1\}^l \rightarrow \{0,1\}^l$ be a substitution and $\pi_p : \{1,...,lm\} \rightarrow \{1,...,lm\}$ be a permutation. Let $\mathcal{P}(plaintextspace) = \mathcal{C}(ciphertextspace) = \{0,1\}^{lm}$, and $\mathcal{K} \subseteq (\{0,1\}^{lm})^{N_r+1}$ consist of all possible key schedules that could be derived from an initial key K using the key scheduling algorithm. For a key schedule $(K^1,...,K^{N_r+1})$, we encrypt the plaintext x using the following algorithm. The first and the last operation are key addition in order to prevent anyone to start the encryption and decryption without knowing the key.

The SPN shown in Figure 2.1 has the following values: l (s-box block size) is 4 bits, m (number of s-boxes at a round) is 4 bits, lm (block size) is 16 bits and N_r (number of rounds) is 4.

The SPN algorithm shown in Fig. 2.2 can be used for both encryption and decryption, because key addition and permutation are linear and are interchangeable. That is $\pi_p(x) \oplus key = \pi_p(x \oplus key)$. In order to have the same algorithm for both encryption and decryption, we do not have permutation in the last round. So SPN with N_r rounds needs $N_r + 1$ keys. As we discussed earlier, round keys are generated using a public key-scheduling algorithm. As some of the attacks can extract one of the round keys easily, the key-scheduling algorithm should be strong enough to prevent getting other round keys or the complete cipher key once one key is known.

SPN $(x, \pi_s, \pi_p, (K^1, \ldots, K^{N_{r+1}}))$

$w^0 \leftarrow x$

for $r \leftarrow 1$ to $N_r - 1$
begin
  $u^r \leftarrow w^{r-1} \oplus K^r$
  for $i \leftarrow 1$ to m
    do $v^r_{(i)} \leftarrow \pi_s(u^r_{(i)})$
  $w^r \leftarrow (v^r_{\pi p(1)}, \ldots, v^r_{\pi p(lm)})$
end

$u^{N_r} \leftarrow w^{N_r-1} \oplus K^{N_r}$

for $i \leftarrow 1$ to m
    do $v^{N_r}_{(i)} \leftarrow \pi_s(u^{N_r}_{(i)})$
$y \leftarrow v^{N_r} \oplus K^{N_r+1}$

output (y)

Figure 2.2: SPN Algorithm

Chapter 3

Rijndael

3.1 A short definition

Rijndael is a key-alternating iterated block cipher. Rijndael can be defined by the function:

$$B[K] = f(K_r) \bullet p(r) \bullet f(K_{r-1}) \bullet p(r-1) \bullet ... \bullet f(K_1) \bullet p(1) \bullet f(K_0),$$

where $f(K_i)$ is the i^{th} round key addition, $p(i)$ is the i^{th} round transformation, $\bullet$ operator denotes function composition.

Figure 3.1 represents the basic structure of the encryption process in Rijndael. Decryption can be achieved by reversing the steps of the algorithm and inverting each of the individual components. The block size (note that the binary message to be encrypted will be divided into plaintext blocks), the key size and the number of rounds are suggested as follows. Block and key sizes can be 128, 192 and 256 bits. Number of rounds respective to the key sizes are 10, 12 and 14 [27]. S-box of Rijndael operates on 8 bits. In the following sections, we briefly explain every functional component of Rijndael.

For the experimental purpose of this project, using the tool developed in this project, we build Rijndael-like ciphers with block and key sizes: 64, 80, 96, 112 or 128 bits. Corresponding s-box block sizes would be 4, 5, 6, 7 or 8 bits. This gives 16 s-boxes at each round. Mostly we experiment ciphers of 3 to 7 rounds.

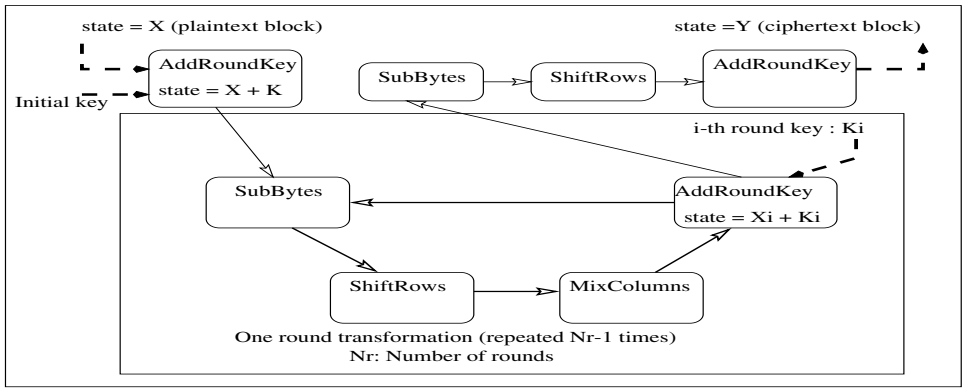

Figure 3.1: Encryption

3.2 General terminologies and concepts

State, Plaintext Block and the Ciphertext Block State is an intermediate result of the cipher. It is a two dimensional $(4 \times N_b)$ matrix of bytes, where N_b is the number of columns of the *state*. Number of rows is fixed to four in order to gain good performance in hardware implementations, especially in 32 bit architectures. N_b varies according to the input block size and will be 4 if the block size is 128 bits(16 bytes), 5 if the block size is 192 bits(20 bytes) and 6 if the block size is 256 bits(24 bytes). Individual round key is also represented as a *state*. The bytes of the plaintext are mapped to a *state* (which is called the initial *state*) and the bytes of the ciphertext are mapped to the result *state*.

Bytes of the plaintext block are denoted by p_0, p_1, p_2, ..., $p_{(4.N_b-1)}$. Similarly, bytes of the ciphertext block are denoted by c_0, c_1, c_2, ..., $c_{(4.N_b-1)}$. The *state* is denoted by $a_{i,j}$, $0 < i < 4$, $0 < j < N_b$, where the element $a_{i,j}$ denotes the byte in row i and column j. The input bytes p_0, p_1, p_2, ..., $p_{4.N_b-1}$ are mapped to the bytes of the *state* in the order $a_{0,0}$, $a_{1,0}$, $a_{2,0}$, $a_{3,0}$, $a_{0,1}$, $a_{1,1}$, $a_{2,1}$, $a_{3,1}$, ... a_{0,N_b}, a_{1,N_b}, a_{2,N_b}, a_{3,N_b}.

Algebraic Nature of Rijndael All components of Rijndael are defined in finite fields $GF(2)$ and $GF(2^8)$ [22, 5]. GF refers to Galois or finite field. This is not true in the case of other block ciphers. In Rijndael, each byte of the input *state* is considered as a polynomial in the field $GF(2^8)$. Polynomials in $GF(2^8)$ have two operations, addition and multiplication.

Addition can be done by simply adding the corresponding coefficients modulo 2 [35, 20]. Since every byte of the *state* is in $GF(2^8)$, multiplication results have to be reduced to a polynomial of degree less than 8. For this purpose, an irreducible polynomial of degree 8 has to be chosen. Rijndael uses $x^8 + x^4 + x^3 + \mathrm{x} + 1$ as its irreducible polynomial [9]. The coefficients of the polynomials in $GF(2^8)$ are in $GF(2)$. So the values of the coefficients can only be either 0 or 1. Addition and multiplication of polynomials in $GF(2^8)$ are illustrated below.

Addition:

$$(x^7 + x^5 + x^4 + x + 1) + (x^6 + x^5 + x^4 + x^3 + 1) = (x^7 + x^3 + x)$$

Multiplication:

$$\begin{aligned}(x^5 + x^3 + 1) * (x^6 + x^4 + 1) = \\ (x^{11} + x^9 + x^5 + x^9 + x^7 + x^3 + x^6 + x^4 + 1) = \\ (x^{11} + x^7 + x^6 + x^5 + x^4 + 1) \pmod{x^8 + x^4 + x^3 + x + 1} = \\ (x^5 + x^3 + 1)\end{aligned}$$

Using Rijndael analysis tool, we can generate Rijndael-like ciphers that use irreducible polynomials of degree 4, 5, 6, 7 or 8. The degree of the cipher's irreducible polynomial is the block size of the s-box. Since each element of the *state* of Rijndael is a byte, which is 8 bits, we have an irreducible polynomial of degree 8. If we have to construct a 64-bit cipher with a *state* of 4×4 matrix and elements of size 4 bits, then we would have an irreducible polynomial of degree 4 for the cipher.

Security margin of a cipher and other security measures A n round cipher has an absolute security margin of n-k rounds or a relative security margin of $\frac{(n-k)}{n}$ if there exists a successful cryptanalytic attack (successful in the sense, the attack algorithm should be faster than the exhaustive key search) against a reduced-round version of the cipher with k rounds [9]. There are other quantitative measures that may also be taken to determine the strength of a cipher. Finding lower bounds with respect to the complexity of specific attacks and finding the maximum input-output correlation in the context of the linear cryptanalysis are few other security paradigms [9, 29].

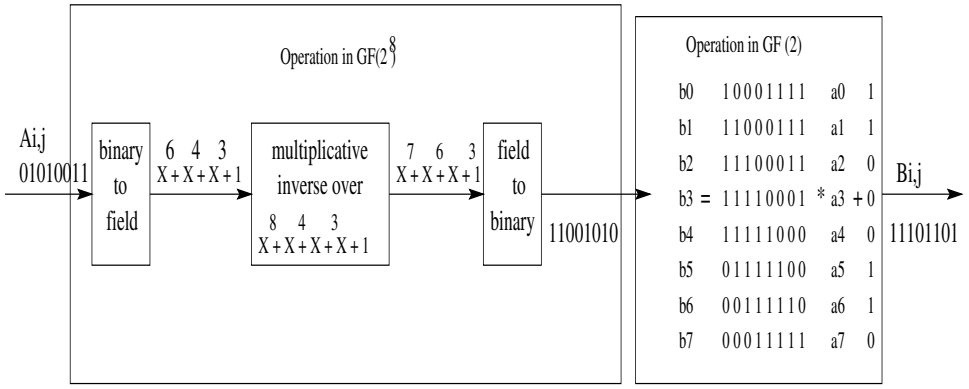

Figure 3.2: Rijndael Substitution

3.3 Components of Rijndael

AddRoundKey $[S' = S + K]$ This component takes a *state* as input and XORs it with the key. All rounds of Rijndael can use different keys or the same key. The decision to use different keys or same key is discussed in the reference [9]. AddRoundKey operation can also be represented as addition of matrices.

SubBytes This component is similar to the substitution in SPN [15]. However substitutions in SPN do not have any algebraic definition, whereas substitution in Rijndael can be defined algebraically [9]. Substitutions in SPN and Rijndael are nonlinear.

A function f is linear if and only if $f(A+B) = f(A)+f(B)$. The non-linearity property is proved to be a strong cryptographic primitive against some of the known attacks [27]. Other desirable properties are invertibility, minimization of the largest non-trivial correlation between linear combinations of input bits and linear combination of output bits, minimization of the largest non-trivial value in the X-OR table, and the complexity of its algebraic expression in $GF(2^8)$ and simplicity of description [15]. Most of these criteria are now necessary for any block cipher to be secure against the already known attacks, especially linear and differential cryptanalysis.

The functionality of s-box includes operations in $GF(2^8)$ and operations in $GF(2)$. Rijndael's s-box takes a byte as input and it has two sub components. The first component finds the multiplicative inverse of the field

representation of the input byte. Since it is a simple algebraic expression by itself, it is possible to mount algebraic attacks. Hence it is followed by an affine transformation. The affine transformation is properly chosen in order to make the SubBytes a complex algebraic expression while preserving the nonlinearity property. The functionality of Rijndael's s-box is illustrated in Figure 3.2.

Linear and differential cryptanalysis exploit the input-output correlation and the difference propagations of the cipher in order to extract partial or whole bits of the keys. To secure the cipher against these attacks, the nonlinearity of the s-box should satisfy few properties. The maximum input output correlation and the difference propagation probability should be minimum. The substitution component of Rijndael has these properties at a maximum optimum level. The sub-bytes component can be implemented as a table look-up operation. Table look-up implementation has two advantages: one is speed and the other is constant processing time. Constant processing time of the components precludes timing and differential power attacks [14, 30].

ShiftRows [Permutation of bytes at row level] We have seen that the permutation of SPN, is at bit level. Rijndael's shift-rows function is blockwise permutation or it is sometimes called transposition. Blockwise permutation can also be viewed as a bitwise permutation. The main function of this layer is to diffuse the changes made at the SubBytes layer. The diffusion should be well optimized to give resistance against linear and differential cryptanalysis. The ShiftRows layer operates on row level of the *state*. Permutation is achieved by shifting the rows cyclically over different offsets. Each row should have a different offset [10]. The choice of the shift offsets makes a difference to the algorithm's strength against differential and saturation attacks [21]. So in Rijndael, the simplest and the strongest options are chosen as the offsets. For a target block of 128 bits, first row is not shifted, second is shifted left by one, third by two and fourth row by three. Using our tool, we can construct a cipher with shift-rows of different offsets. Subsequent differential analysis will show the consequences.

MixColumns [Permutation at column level] The sub-bytes component changes the bytes as it substitutes one byte for another. This local byte change is diffused further with the help of mix-columns operation. Insufficient diffusion may reveal some information that could eventually be used for a successful attack. Diffusion is one of the three desired crypto-

graphic primitives [22]. As we have already discussed, the mix-columns and the shift-rows layers together contribute to Rijndael's diffusion. Rijndael's diffusion is completely algebraic.

Each column of the input *state* is considered as a cubic polynomial over $GF(2^8)$ and it is multiplied modulo $x^4 + 1$ with a fixed polynomial $C(X)$

$$C(X) =' 03' x^3 +' 01' x^2 +' 01' x +' 02'$$

Although polynomial $x^4 + 1$ is not irreducible, the operation is invertible to perform decryption since polynomial $C(X)$ is co-prime to $x^4 + 1$[36]. Another interesting fact is that the coefficients of the polynomial $C(x)$ and the polynomial of a column of the *state* are by themselves polynomials in $GF(2^8)$. So if we take the 0^{th} column of the *state*, it is:

$$a_{30}x_3 + a_{20}x_2 + a_{10}x + a_{00}$$

This is a polynomial in $GF(2^8)$. In order to perform mix-columns, this is multiplied by the polynomial $C(x)$ *modulo* $x_4 + 1$ giving the result

$$b_{30}x_3 + b_{20}x_2 + b_{10}x + b_{00}$$

For example, the coefficient 03 in $C(x)$ is actually the hexadecimal representation of a byte, and is nothing but the polynomial $x + 1$.

The multiplication by $C(x)$ modulo $x^4 + 1$ can also be represented as a matrix transformation. For example if we take the 0^{th} column, which has four bytes (a_{00}, a_{10}, a_{20}, a_{30}), the mix-columns layer does the following to produce the 0^{th} column of the result $(b_{00}, b_{10}, b_{20}, b_{30})$. Similarly the operation is repeated for other columns.

$$\begin{pmatrix} b_{00} \\ b_{10} \\ b_{20} \\ b_{30} \end{pmatrix} = \begin{pmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x \end{pmatrix} * \begin{pmatrix} a_{00} \\ a_{10} \\ a_{20} \\ a_{30} \end{pmatrix}$$

The choice of the coefficients of $C(x)$ has two advantages. Choosing 03, 02 and 01 as coefficients allows for easy implementation in hardware (Since multiplication by 02, which corresponds to polynomial x, can be achieved by a left shift modulo the Rijndael polynomial. Multiplication by 01 has no processing at all). They are chosen to give optimal diffusion in order to preclude differential and linear cryptanalysis [9]. In order to avoid timing and power attacks, the number of cycles per execution in the implementation

is a constant. If any of the shift operations (in order to multiply by 02) leads to an overflow resulting in an additional operation, then the cycle time will vary [30]. This could result in a successful power attack. There are many implementation techniques of Rijndael that have been suggested to avoid these attacks.

Using the tools developed in this project, we can carry out differential cryptanalysis either by modifying the constants or by completely removing this layer.

Key Scheduling Every round has a different key. Using key scheduling algorithm, each round key is derived from the cipher's original key. The key schedule algorithm has two components. First is key expansion, which derives the expanded key (the concatenation of individual round keys) from the cipher key. The second component is round key selection. Rijndael has a simple key selection procedure. Nonlinearity, diffusion and symmetry elimination is incorporated to avoid attacks against key scheduling [9]. Nonlinearity is achieved by using the same sub-bytes component, diffusion is used to efficiently spread the cipher key differences into the expanded key and the symmetry elimination is achieved by using a different constant in generating each round key.

3.4 Other important aspects

Following are some of the aspects that are important in the context of the design of Rijndael.

Simplicity and Symmetry Simplicity is maintained in the design of every component of the algorithm, which is one of the characteristic imposed by the NIST. Rijndael designers have simplified both the specification and the analysis. It should not be misunderstood that simplicity would contribute to weakness. It is working in the opposite way in Rijndael. In the absence of successful cryptanalysis, simplicity of the algorithm helps to appeal to its credibility. Analysis simplicity demonstrates the strength of the cipher against known and possible attacks. Simplicity can be achieved in many ways and Rijndael has achieved it by choosing the symmetry in its components and the choice of their operations.

The algorithm has significant symmetry. Symmetry means each bits of the plaintext is treated similarly in the algorithm. Symmetry is accomplished across the rounds, within the round transformation and in the steps of the round. One of the attacks called the Slide Attack [3] exploits the symmetry of a block cipher. The weakness of this attack is: it demands significant symmetry even in the key schedule [9]. In Rijndael, round constants are used in getting the key for each round. This is one of the simplest ways to eliminate the symmetry in a key schedule. There exists a good bundle alignment (where a bundle is a group of bits that are together processed as a unit in any component of the layer) to achieve the symmetry. The alignment property of Rijndael can be exploited only up to six rounds by saturation (or square) attack. For the complete Rijndael, this attack is of no practical value because of its high complexity [21].

Symmetry and alignment properties of Rijndael's round transformation have made the following tasks easy and clear:

- Computing the lower bounds of the complexity of the attacks
- Finding the maximum difference propagation probability
- Finding the maximum input-output correlation

For ciphers that have sophisticated and complicated round transformation, these tasks are not easy [9].

Global and Local optimizations Local optimization is concerned with the optimization and security measures taken in the round transformation. Global optimization is about getting good properties like diffusion at the overall cipher level. The latter deals with the sequence of rounds rather than a single round. Local optimization obtains nonlinearity and diffusion at the local level. Maximum input-output correlation is minimized to the extent possible in the context of linear cryptanalysis. Maximum difference propagation probability is minimized in the context of differential cryptanalysis. Most of the available block ciphers are designed with good local optimization. Global optimization is considered significantly in the design of Rijndael. Wide Trail Strategy is a unique and an efficient approach taken along the line of global optimization [9]. The important benefits of targeting global optimization are inexpensive nonlinear boolean transformations such as small s-boxes [9].

Linear transformation Although nonlinear substitution layer contributes to the strength of the cipher against differential analysis, linear transformation of the algorithm increases the resistance to differential analysis [16]. From the research done on the design of s-boxes against known attacks, it was found that large s-boxes should be chosen. Choosing large s-boxes will take more units of time and storage resources. So in the design of Rijndael and other related ciphers which follow the wide-trail strategy [9], large resources are spent in the linear transformation to provide high multiple-round diffusion [9].

Because nonlinearity and diffusion are optimized locally and globally, Rijndael is proved to be secure against these analysis. In pursuit of finding the role of diffusion in Rijndael, this project has tools to minimize diffusion by the techniques that we have already discussed in the earlier sections. The reason we focus mainly on diffusion is due to the fact that, the strength of Rijndael against differential analysis is unusually gained more through linear transformation rather than nonlinear.

Chapter 4

Project description

4.1 Goals of the project

- Develop tools to configure SPN ciphers
- Perform differential analysis to understand the pros and cons of s-boxes
- Describe the importance of permutations in relation to differential cryptanalysis
- Develop tools to configure Rijndael-like ciphers and to perform differential cryptanalysis on them
- Explain the tools and their usage

4.2 Main discussion of the project

The discussion can be divided into two parts. First part is on the programs developed to perform differential cryptanalysis on SPN and Rijndael-like ciphers. Second part is on the differential cryptanalysis of these ciphers.

4.3 Programs developed

Programs are developed in C language. They are explained in Tables 4.1 and 4.2. Besides these tools, in order to interpret the results, there are some UNIX shell scripts developed to compile the outputs of the analysis. Compiled output helped in interpreting the results in various ways.

Name	Purpose	Parameters
Tools developed for SPN		
spnanalysis.c	Performing Differential Cryptanalysis on SPN	Cipher key, Permutation S-box and Number of rounds
s-box-create.c	Creating random or manual s-box	Input/output size of the s-box in bits. In case of manual s-box, individual s-box values has to be given
permute.c	Creating a random or manual permutation	Number of bits to permute. In case of manual permutation, individual values has to be given

Table 4.1: Tools developed for this project - Part 1

A brief description of the usage of programs is given below.

- We can configure any variation of SPN cipher. For example, the user can vary one or many of the following: block size of the cipher, block size of the s-box, s-box, permutation, key schedule, number of rounds, etc.
- We can perform cryptanalysis on the configured SPN. Differential cryptanalysis using the programs is user-friendly and interactive. We can set up various differential trails interactively, execute the analysis and extract the partial key bits.
- In the context of Rijndael-like ciphers, we can configure the ciphers and perform differential cryptanalysis. We can configure the cipher with or without shift-rows, with or without mix-columns, etc. We can choose to use random permutation instead of Rijndael-like diffusion. We can choose to use a different substitution instead of Rijndael-like s-box. We can vary the offsets used in shift-rows or vary the irreducible polynomial or vary the polynomial used in the mix-columns, etc.

4.4 Differential cryptanalysis

Differential cryptanalysis is performed on both SPN and Rijndael-like ciphers. While analyzing SPN, we focused on the role of permutation. While

Name	Purpose	Parameters
Tools developed for Rijndael		
rijndael_analysis.c	Performs Differential cryptanalysis on Rijndael-like ciphers. Given a specific input, this program generates the diffusion component internally.	Blocksize of the cipher in bits, cipher key of size equal to block size, s-box (either Rijndael type or SPN type), diffusion component (can be just a permutation or just shiftrows or shiftrows with mixcolumns or permutation with mixcolumns and number of rounds
generate_sbox.c	Generates Rijndael type s-box	A text file that contains four inputs in sequence delimited by newline character: degree n of $GF(2^n)$, a number (decimal representation of the cipher polynomial), and matrices A and B of the affine transformation $y = Ax + B$
matrix.c	Includes modules to read, print, add and multiply matrices. Matrices represent polynomials. Addition operation follows $GF(2)$.	Matrices to be processed
find_irreducible.c	Lists all the polynomials in $GF(2^n)$ and indicates for every polynomial if it is reducible	Degree n of $GF(n)$

Table 4.2: Tools developed for this project - Part II

analyzing Rijndael-like ciphers, we varied the configuration of the diffusion component. However, much effort is given to experiments on SPN rather than on Rijndael. But we can use the programs to analyze Rijndael-like ciphers further.

Chapter 5 contains a general introduction of differential cryptanalysis and an illustration of differential cryptanalysis on SPN. In chapter 6, we presented the results and conclusions of differential cryptanalysis on SPNs. Experiments and results are divided based on the various possibilities of the permutation component. As the role of permutation is affected by the choice of s-boxes, part of the experiments are performed focusing the combined effect of s-box and permutation.

In chapter 7, we discuss the ways to configure different Rijndael-like ciphers. An illustration of differential cryptanalysis on a Rijndael-like cipher is presented. While doing differential cryptanalysis on Rijndael-like ciphers, whenever we have a Rijndael-like diffusion component, it is impossible to extract the partial keys. Rijndael-like diffusion components introduce high degree of uniformity in the cipher texts. Differential cryptanalysis is ineffective if the target cipher contains uniformity.

Chapter 5

Differential analysis and SPN

5.1 General description and the algorithm

General information Differential cryptanalysis [35, 15] is a well-known chosen plaintext attack [22] and it is successful against many block ciphers. Differential analysis exploits the following properties of a block cipher: lack of optimal nonlinearity and lack of good diffusion. Non-linearity is associated with the substitution component and diffusion is associated with the permutation component.

Principles behind differential analysis There are four important concepts in the context of SPN and differential analysis.

- Permutation is linear. It implies $permutation(a) \oplus permutation(b)$ is equal to $permutation(a \oplus b)$.
- Ideal randomness of s-box cannot be achieved. If we achieve an ideal random s-box, all elements in the xor-difference table (refer Table 5.4 in section 5.3.2) would be 1. The probability that an output difference occurs given an input difference would be $\frac{1}{2^n}$ for all elements, where n is the number of bits of the s-box input. However, ideal randomness is not mathematically possible for the following reasons: the value of all elements in the s-box difference table should be even, since $a \oplus b = b \oplus a$. Since the s-box is bijective, the input difference of 0 will lead to an output difference of 0. So the element corresponding to (row=0, column=0) at the difference table will be 2^n and all other elements in row 0 and column 0 will be 0.

- It can be assumed that the probability of a differential at one round is independent of the differential probability at other rounds.
- Key bits are cancelled when we deal with xor-difference of input texts.

We refer to Figure 5.1 to illustrate briefly the principles behind differential analysis on SPN. In part 1 of Figure 5.1, we have given a simple one round SPN cipher which accepts plaintext blocks of size 3-bits. One round constitutes three components: key addition, substitution and permutation. In part 2 of Figure 5.1, we show the encryption of two plaintexts $X1$ and $X2$. $X1$ is encrypted to give $Z1$ and $X2$ is encrypted to give $Z2$.

In part 3 of the Figure 5.1, we consider the xor-difference of the two inputs $X1$ and $X2$. It is important to note that we are not feeding the difference $X1 \oplus X2$ into the SPN, rather we just consider the xor-difference between the two plaintexts $X1$ and $X2$ that are fed. Let the xor-difference of two inputs $X1$ and $X2$ be $D1$. Since the same key K is xor-ed with itself, the xor-difference of the two texts after key addition will still be the same $D1$. In other words, key can be ignored when we trace xor-difference. Now, back to part 2, the keyed plaintexts $(X1 \oplus K)$ and $(X2 \oplus K)$ enter the s-box, with the xor-difference $D1$, yielding $Y1$ and $Y2$. Now on part 3, the xor-difference of the outputs $Y1$ and $Y2$ is $D2$.

If we fix the xor-difference of the input texts to some constant $D1$ (we do not fix any values to $X1$ and $X2$) and take pairs $(X1, X2)$ such that $X1 \oplus X2 = D1$, then there are 2^n such pairs of $(X1, X2)$. For each such pair $(X1, X2)$, there will be a corresponding output pair $(Y1, Y2)$. Suppose if we want the value of $(Y1 \oplus Y2)$, irrespective of their individual values, to be specifically some constant $D2$, it will occur with a certain probability $P1$.

In other words, if we input any two plaintexts into the s-box with a certain xor-difference $D1$, the corresponding output difference will be $D2$ with a probability of $P1$. Now we will look into the permutation, which is next in sequence to the substitution. $Y1$ is permuted to give $Z1$ and $Y2$ is permuted to give $Z2$. We know $Y1 \oplus Y2$ will be $D2$ with probability $P1$. Linearity states that the permutation$(Y1)$ $\oplus$ permutation $(Y2)$ is equal to the permutation $(Y1 \oplus Y2)$. Let the permutation of $(Y1 \oplus Y2)$ be $D3$. Following the linearity of permutation, if the xor-difference of $X1$ and $X2$

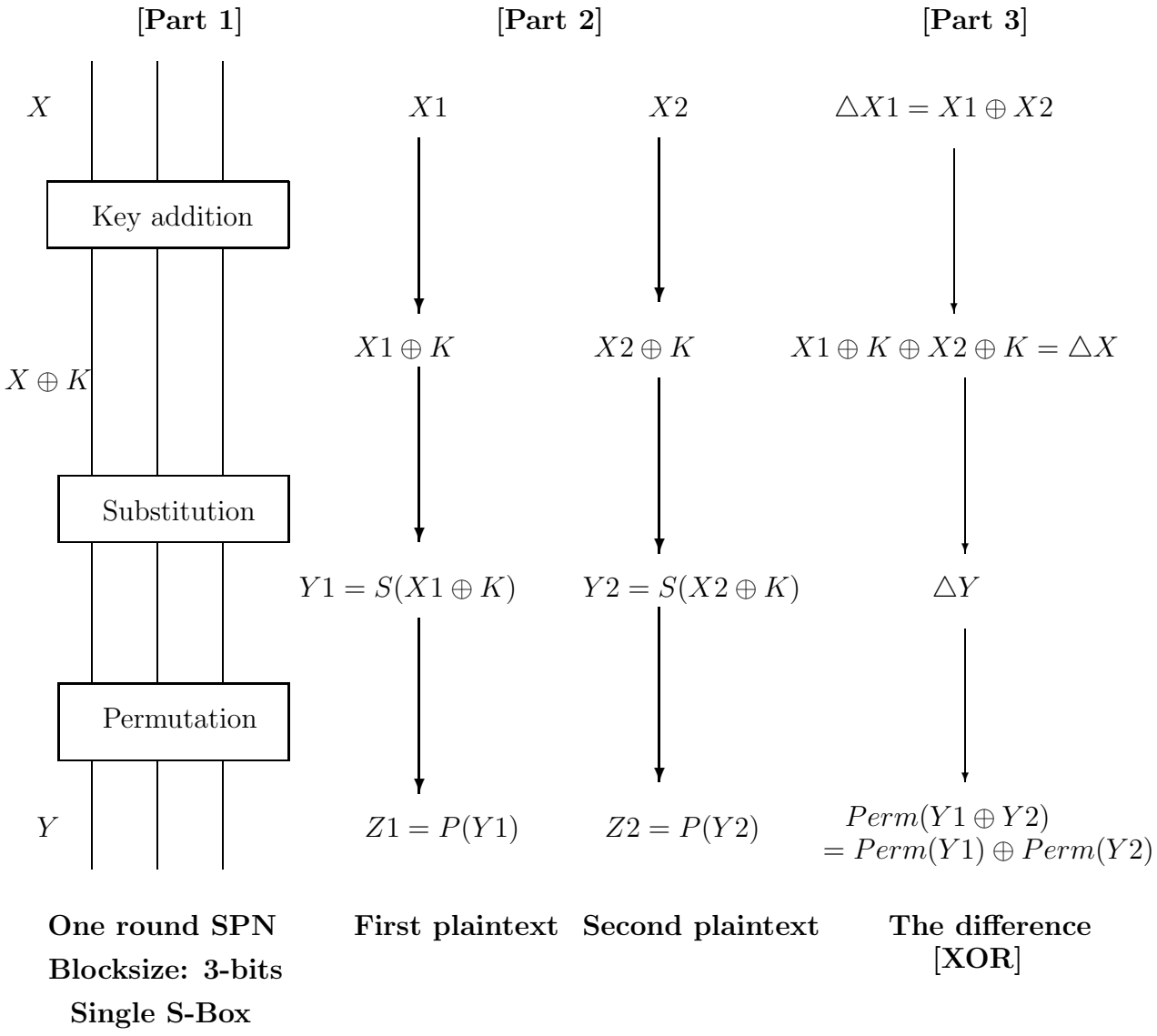

Figure 5.1: Principles of Differential Analysis on SPN

is $D1$, then the probability that the permutation of $(Y1 \oplus Y2) = D3$ is the same $P1$.

From the above facts, we can discuss differential analysis on a complete SPN. In the actual differential analysis, one round may have many (same or different) s-boxes at a row (for example we can take the 16 bit SPN in Figure 2.1 which has four 4-bit s-boxes at each round, where all s-boxes in the cipher are same). In the context of differential cryptanalysis, we ignore the presence of key addition component and just assume one round of SPN contains only substitution and permutation components.

5.2 Analysis

5.2.1 S-box differential table calculation

We calculate the difference table (refer Figure 5.4 for an example) for each of the s-boxes involved in the SPN [35]. In the SPN shown at Figure 2.1, we used the same s-box all over the cipher. We calculated the difference table only for this single s-box. The rows of the table are indexed by the input xor-difference ($\triangle X$) and the columns are indexed by the output xor-difference ($\triangle Y$). An element in the difference table at the intersection of a $\triangle X$ and a $\triangle Y$ is the probability that the output difference will be $\triangle Y$ if the input difference is $\triangle X$.

5.2.2 Setting the differential trail

First round To set a differential trail, we start with first round. Let $\{ S_1, S_2, \ldots, S_k\}$ be the sequence of s-boxes in first round, where k is the number of s-boxes per round. We choose one or more s-boxes from the number of s-boxes at first round. For simplicity, we can assume we choose only one s-box at first round. For each of the chosen s-boxes, we refer the corresponding difference table and select a $\langle \triangle X, \triangle Y \rangle$ pair that has a relatively higher probability. In first round, all s-boxes that are not chosen should have $\langle 0, 0 \rangle$ as their $\langle \triangle X, \triangle Y \rangle$.

The s-boxes chosen in any round are called the active s-boxes. Usually we choose not more than one or two s-boxes, becuase more the number of active s-boxes, wider is the differential trail and so lesser is the probability of success. Let

$$\{\langle \triangle X_1, \triangle Y_1, \mathcal{P}_1 \rangle, \langle \triangle X_2, \triangle Y_2, \mathcal{P}_2 \rangle, \cdots, \langle \triangle X_1, \triangle Y_1, \mathcal{P}_{N_s} \rangle\}$$

be the sequence of xor-difference pairs and their probabilities at the first round, where $\triangle X_i$ represent the input difference of S_i, where S_i denotes the i^{th} s-box, $\triangle Y_i$ represents the output difference of the s-box S_i and $\mathcal{P}_i$ represents the probability of the xor-difference pair: $\langle \triangle X_i, \triangle Y_i \rangle$ at s-box S_i.

We might wonder what these xor-differences have to do with differential analysis. We will find out the reason after reading through the end of this section. After setting up a complete trail, we perform differential analysis. We choose a set of plaintext pairs, where each pair $\langle P_a, P_b \rangle$ have an xor-difference $\triangle X$, where

$$\triangle X = \langle X_1 X_2 \cdots X_n \rangle,$$

X_i represents the bits of $P_a \oplus P_b$ that corresponds to the s-box S_i.

Proceeding to the subsequent rounds to build the trail In the first round, we have $\triangle Y$ as the output difference of the substitution where

$$\triangle Y = \langle Y_1 Y_2 \cdots Y_n \rangle,$$

Y_i represents the bits of the xor-difference of the output that corresponds to s-box S_i. We permute this difference $\triangle Y$ to get the input difference $\triangle X$ of the next round. At the second round, the s-boxes that have a non zero input xor-difference are active. So the number of active s-boxes and their input xor-differences are determined by the first round permutation. Then, corresponding to the input xor-difference of each active s-box, we refer the difference table to select the best output xor-difference. Mostly, the best output xor-difference will be the one that has the highest probability and minimum number of "1" bits.

In the first round, we chose the active s-boxes and their input and output difference pairs. But in the second round, we only choose the output differences of the active s-boxes becuase the active s-boxes and their input differences were determined by the first round.

At every round (other than the first round), we have the active s-boxes and their input differences. We then refer the difference table to select the best output differences. We permute the output difference to determine the input difference of the next round. We repeat this process until we reach the input difference of the last round. The input difference at the last round is the expected input difference $\triangle X_e$.

For any plaintext pair P_a, P_b, such that $P_a \oplus P_b = \triangle X^1$ (where $\triangle X^1$ is the input difference in the first round as set by the differential trail), we expect the input difference in the last round to be $\triangle X_e$ with the probability $\mathcal{P}$, where $\mathcal{P}$, the propagation ratio, is the product of individual probabilities of all the active s-boxes [35].

5.2.3 Extraction of the last round key

After setting a high probable differential trail, we choose certain number of plaintext pairs (Refer to [35] to learn about how many plaintext pairs we have to choose) that have an xor-difference $\triangle X^1$. For example, taking the SPN at Figure 2.1, we have four s-boxes at a round. In the first round, suppose we have chosen the second s-box and the input difference to be 0011, then all input text pairs for our analysis should be: (xxxx aaaa xxxx xxxx) and (xxxx aaāā xxxx xxxx). It means, the first, the third and the fourth 4-bits of each of the input text pairs should be same. The xor-difference of the second 4 bits of the input texts should be 0011.

We encrypt each plaintext pair (P_1, P_2) and find their corresponding ciphertext pair (C_1, C_2). For each such ciphertext pair and for every possible partial key (last key used in the encryption), we perform partial decryption until the input of the last round. Let the result of the partial decryption be (C_1', C_2'). If $C1' \oplus C_2'$ is equal to the expected input difference $\triangle X_e$ of the differential trail, then we increment the count of the corresponding partial key. We repeat this process for all possible partial keys and for all chosen plaintext pairs. Finally, the partial key which has the largest count is expected to be the actual partial key.

In the last round, if the number of active s-boxes is smaller than the actual number of s-boxes, then the number of possible partial-keys to apply will be far lesser than the exhaustive number of full-keys. This is true if the differential trail is narrow.

Once we are done with all chosen plaintext pairs, the partial key which has the largest count is expected to be the correct one. We can set up a different good differential trail which may yield rest of the key bits. If the number of bits in the rest of the key is small enough, we can exhaustively search to find them. Once we find the last round key, we can apply partial decryption technique to the next round with more confidence because the probability is more than the earlier case.

Round	1	2	3	4	5
Key	3a94	a94d	94d4	4d43	d438

Table 5.1: Round keys

5.2.4 Differential cryptanalysis algorithm

We give the complete differential analysis algorithm in this section (refer Figures 5.2 and 5.3). This algorithm is a generalized version of the one given in the textbook [35]. In this Figure, L_1, L_2, ..., L_a are the last round key bits corresponding to the active s-boxes S_{A_1}, S_{A_2}, ..., S_{A_a} in the last round. The symbol τ denotes a set of 4-tuples (x, y, x^*, y^*), where (x, x^*) is a chosen plaintext pair with a chosen xor difference and (y, y^*) is the ciphertext pair corresponding to the chosen plaintext pair. $y(A_i)$ is the ciphertext bits that are the output of s-box S_{A_i}. E_i is the expected difference bits of the textpair that is input to the last round s-box S_{A_i}. This algorithm can be understood by reading sections 5.2.1, 5.2.2 and 5.2.3.

5.3 A complete example of differential analysis

This example illustration is different from the examples we normally find in the publications. Other relative publications seem to use always the same example [35, 15].

5.3.1 Analysis - Sample Illustration

Step 1: Initial configuration Before analysis, we configure SPN using the program. The example cipher we took has the following configuration: number of rounds = 4, block size = 16 bits and the s-box size is 4 bits. The cipher key is 32 bits long and the individual round keys are 16 bits long. The individual round keys are generated using an internal key scheduling module which has no significance in the context of differential cryptanalysis. In our example, the cipher key is *3a94d438* (given in hexadecimal format) and the individual round keys are given in Table 5.1

The 4-bit s-box used in our cipher is given in Table 5.2. u represents the input in hexadecimal format and similarly v represents the output in hexadecimal format. For example, if the input is *0010* then the output will be *0001*.

Let N_r be the # of rounds of the cipher

Let a be the # of active $s-$boxes at the last but one round
Let $\{A_1, A_2, ..., A_a\}$ are the position of the active $s-$boxes at the last but one round

for $(L1, L2, ..., L_a) \leftarrow (0, 0, ..., 0)$ **to** $(F, F, ..., F)$
do $Count[L1, L2, ..., L_a] \leftarrow 0$

for each $(x, y, x^*, y^*) \in \mathcal{T}$
begin

$if\ (\ y_{(A_1)} = (y_{(A_1)})^\star\ ) \bigwedge (\ y_{(A_2)} = (y_{(A_2)})^\star) \bigwedge ... \bigwedge (\ y_{(A_a)} = (y_{(A_a)})^\star)$

begin

for $(L1, L2, ..., La) \leftarrow (0, 0, ..., 0)$ *to* $(F, F, ..., F)$

begin

for $i \leftarrow (1, 2,$ *to* $a)$

begin

$v_{(A_i)}^{N_r} \leftarrow L_i \oplus y_{(A_i)}$ $\qquad$ $u_{(A_i)}^{N_r} \leftarrow \pi_S^{-1}(v_{(A_i)}^{N_r})$

$(v_{(A_i)}^{N_r})^\star \leftarrow L_i \oplus (y_{(A_i)})^\star$ $\qquad$ $(u_{(A_i)}^{N_r})' \leftarrow u_{(A_i)}^{N_r} \oplus (u_{(A_i)}^{N_r})^\star$

$(u_{(A_i)}^{N_r})^\star \leftarrow \pi_S^{-1}((v_{(A_i)}^{N_r})^\star)$

end

if $((u_{(A_i)}^{N_r})' = E_i)$ *and* $((u_{(A_i)}^{N_r})' = E_i)$ **for** *all i in* $(0, 1, ..., a)$

then $Count[L1, L2, ..., La] \leftarrow Count[L1, L2, ..., La] + 1$

end

end

end

Figure 5.2: Differential cryptanalysis algorithm

$max \leftarrow -1$

$for\ (L1, L2, ..., La) \leftarrow (0, 0, ..., 0)\ to\ (F, F, ..., F)$
,

$begin$

$if\ Count[L1, L2, ..., La] > max$

$max \leftarrow Count[L1, L2, ..., La], maxkey \leftarrow (L1, L2, ..., La)$

end

$output(maxkey)$

Figure 5.3: Differential cryptanalysis algorithm, part 2

u	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
v	4	15	1	11	8	9	10	5	3	2	12	14	6	7	13	0

Table 5.2: 4 bit s-box

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

Table 5.3: Permutation

The permutation we used in the cipher example is given in Table 5.3. This table specifies a bitwise permutation for the cipher with a block size of 16 bits. Substitution and permutation components are random in nature. They are generated using tools *s-box-create.c* and *permute.c* respectively (refer Table 4.1 for the description of tools). The differential table corresponding to the above s-box is calculated using our tool.

5.3.2 Step 2: Difference distribution table

Table 5.4 corresponds to the s-box we have discussed above. Each row represents the xor-difference of an input pair ($X1$, $X2$) of this s-box and similarly each column represents the xor-difference of an output pair of this s-box. An element in the table indicates the frequency of a pair (x-difference, y-difference). For example, the element at (row 4, column 5) indicates the fact that if the difference between two inputs to this s-box is 4, then the number of times the difference between the corresponding outputs is 5, would be 4 out of 16. The probability is $\frac{4}{16}$, that is 0.25. Notice that we have a frequency 8 at row 12 and column 11. We can find many frequencies with value 6. This is significant in our analysis, because most of the times, larger frequency values give better chance of getting higher probability of the trail. This kind of s-boxes is weak. A significant amount of research has been done in building good s-boxes [**?**, 1, 34], especially against differential cryptanalysis [2, 16]. We will discuss some of these concepts in the following chapters in the context of our examples. If the frequency values are evenly distributed, then the s-box would be strong.

5.3.3 Step 3. Setting up the differential trail with high total probability

After several trails were tested, we finalized a differential trail that has a total probability of 4.69e-2. Note that this is not the only trail which is better, there might be other trails also. The trails are given in Table 5.5. In the first row, we selected the first s-box as active. The input difference we chose was *12 (1100)* and the corresponding output difference we chose was *11 (1011)*. The main reason for this input/output choice is becuase of its high probability. After the permutation in the first round, the first s-box in the second round is the only one that has a non-zero input, *13 (1101)*. Hence for this s-box, we chose *10 (1010)* as the output difference. Here we could have chosen *9 (1011)* as the output difference becuase it also has an equal probability as *10*, but there are no restrictions in choosing. We repeated this

$x\triangle/y\triangle$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	6	2	0	0	0	0	0	0	0	2	2	0	2	0	2
2	0	0	2	0	2	2	0	2	0	0	0	2	4	0	0	2
3	0	0	0	2	0	0	2	0	0	0	2	0	0	4	4	2
4	0	2	0	0	0	4	2	0	0	0	0	2	2	0	4	0
5	0	2	0	2	6	0	0	2	0	0	0	0	2	2	0	0
6	0	0	4	0	0	0	0	0	0	4	4	0	0	0	4	0
7	0	2	0	4	0	2	0	0	4	0	0	2	0	0	0	2
8	0	0	0	0	0	4	0	4	0	0	0	0	0	4	4	0
9	0	0	0	0	0	0	2	2	2	0	2	0	2	0	0	6
10	0	2	4	0	0	2	0	0	2	4	0	0	2	0	0	0
11	0	0	0	6	2	0	0	0	4	0	2	0	0	2	0	0
12	0	0	2	0	0	0	2	0	2	0	0	**8**	2	0	0	0
13	0	2	0	2	2	0	2	0	0	4	4	0	0	0	0	0
14	0	0	0	0	2	0	0	6	0	4	0	0	2	0	0	2
15	0	0	2	0	2	2	6	0	2	0	0	0	0	2	0	0

Table 5.4: Differential table

Round	s_1	s_2	s_3	s_4	Individual Prob. ρ	Total Prob σ
1	12,11	0,0	0,0	0,0	0.500	5.00e-1
2	13,10	0,0	0,0	0,0	0.250	1.25e-1
3	5,4	0,0	0,0	0,0	0.125	4.69e-2
4	0,0	2,0	0,0	0,0		**4.69e-2**

Table 5.5: Differential trail

Key	0000	0100	0300	0400	0600	0800	0F00
Count	3	1	1	6	1	1	1

Table 5.6: Round keys

procedure until the input of the last round. The resulting total differential probability is *4.69e-2*.

5.3.4 Step 4. Performing the analysis

We performed the analysis on the differential trail shown in Figure 5.4 and in Table 5.5. The total differential probability is

$$\sigma = (\frac{1}{2} \times \frac{1}{4} \times \frac{3}{8}) = \frac{3}{64} = 0.046875.$$

In other words, given a random set of plaintext pairs with $\triangle X$ equal to [c 0 0 0] (where each value in the 4-tuple corresponds to 4 bits and is represented in hexadecimal format), we expect the $\triangle X$ at the s-box S_2^4 to be 2 (0010) with probability σ.

The complexity of the cryptanalysis, which is the total number of plaintext pairs we need to extract the partial key, is given as $z \times \frac{1}{\sigma}$, where z is a small positive constant. In our example, $\frac{1}{\sigma} \simeq 21$. We have assumed $c = 2$. So we took 42 plaintext pairs and we extracted the key. Table 5.6 shows the counts of the real and other keys. The real key is 4 which is shown as [0400].

We successfully extracted the partial key (second 4 bits) used in the last round. Now, we can extract the rest of the key bits either by setting up a different trail that leads to other s-boxes in the last round or we can exhaustively compute the rest of the key bits. For example, to analyze a 64 bit cipher, if we can extract 32 bits of the last round key using the

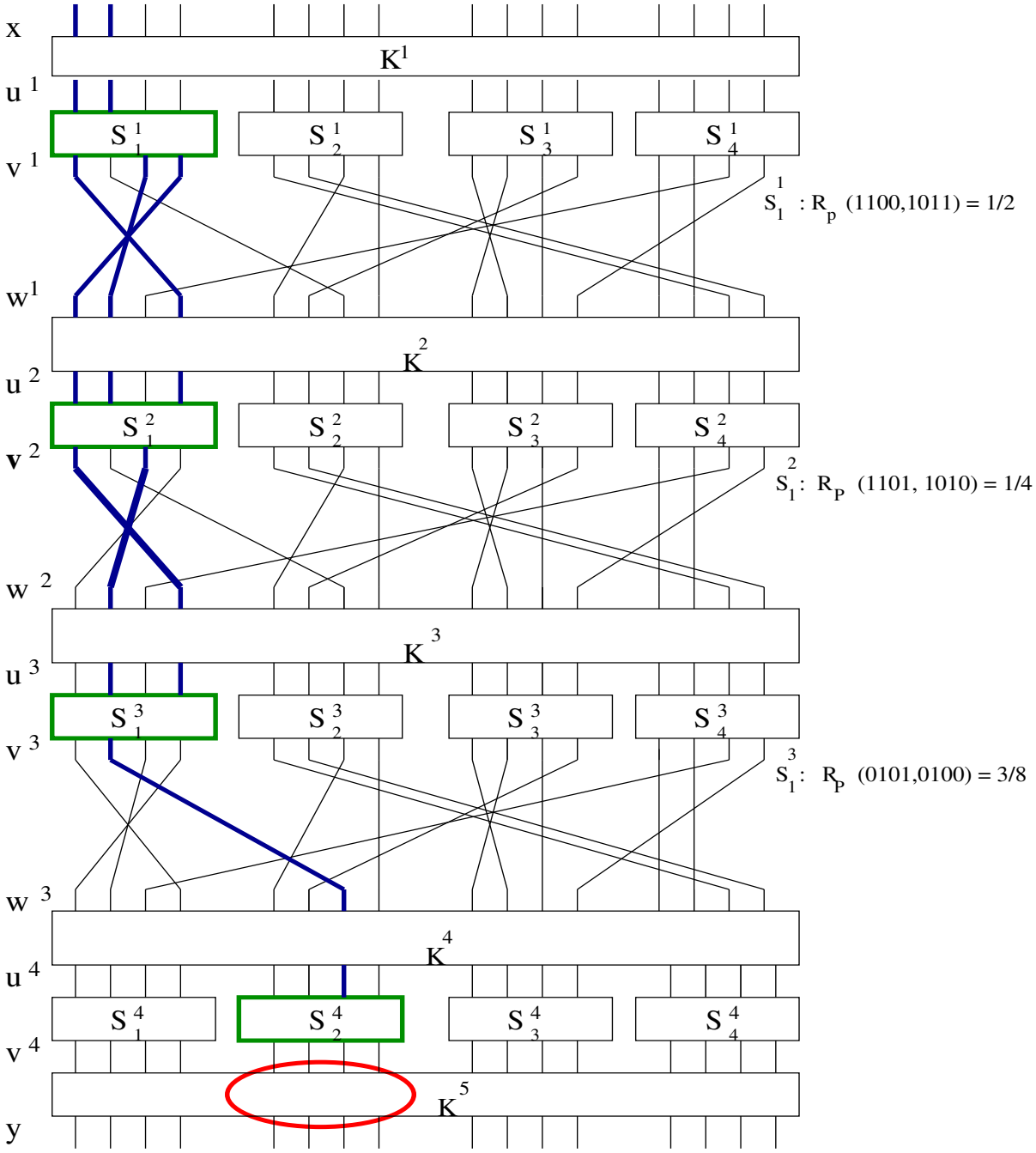

Figure 5.4: SPN analysis: An example

differential analysis, then it is easy to exhaustively search the remaining 32 bits. But getting 32 bits of the last round key depends upon one or more of the following: s-box used, permutation used, number of rounds of the cipher taken for attack, differential trail used, etc.

Chapter 6

SPN permutation and Differential analysis

6.1 Program description

As discussed earlier, three programs are developed in this project with respect to SPN analysis. One to generate the s-box and to calculate its differential table. The second one is to generate permutation. The third program, which corresponds to the main analysis, performs the following steps,

- Generate an SPN cipher. This step uses one of the s-boxes and the permutation tables we created earlier. The cipher key is given as an input and the round keys are generated using a simple key scheduling algorithm (refer [35]).
- Setting up a differential trail for differential analysis. This process is interactive. A differential trail is set by
  - Choosing the input and output difference at the first round for all the s-boxes we want to make active.
  - Choosing the output differences corresponding to the permuted input differences at each round until the last but one round. For each choice, the individual and the total probability is calculated and will be shown in order to decide the optimality of the choices made.

  In this trail set up, we can go back and forth across the rounds of the cipher to discover a better trail. A better differential trail is the one

that has a propagation ratio greater than all other trails. In this step, we use the s-box differential table as a significant aid in choosing the input and output differences. For this, we can run the s-box generator program in parallel and see the corresponding differential table either completely or partially. If the s-box is too large to be displayed on the screen, the program has the facility to see a row for a given $\triangle X$. The program also has a facility to see all the pairs ($\triangle X$, $\triangle Y$) that has a particular probability. Automatic setting of better differential trail can be implemented. Many differential trails have to be compared to determine a better one. The strength of s-box, the strength of permutation and the number of rounds determine the number of different trails to compare.

- Performing differential analysis to find the key. In this step, we can choose the number of plaintext pairs to be fed into the analysis. This number is mostly determined by the total probability of the differential trail. Lesser the total differential probability, greater is the number of plaintext pairs needed. Sometimes we may not be able to extract the partial key even if the total differential trail probability is high. In such cases, we can again go back to set up a different differential trail to analyze again.

6.2 Criteria for successful cryptanalysis

The success of the analysis depends on two factors:

Strength of the cipher If the cipher is weak, a simple analysis may give a chance to break the cipher. But in advanced ciphers like Rijndael, the weakness is yet to be found. In the case of SPN, bigger s-boxes (may be of size 8 bits minimum), bigger block size, a good permutation, and more number of rounds will give a cipher that is as strong as Rijndael. In our analysis, we will show how the nonlinearity of the s-box and how the permutation matters to the strength of the cipher against differential analysis.

Table 6.1 shows the configuration of the SPNs which are taken for our analysis.

Strength of the cryptanalysis Sometimes the way we analyze matters very much. For example, setting a high probable trail and using a good set of random chosen plaintexts, etc in differential cryptanalysis. In the next section, we discuss this in detail.

Block size	16 to 30 bits
Number of rounds	2 to 6
s-box size	4 to 6 bits
Permutation	random, blockwise and no permutation

Table 6.1: SPN Configurations taken for differential cryptanalysis

u	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
v	14	5	13	2	12	9	0	1	10	15	8	7	6	11	3	4

Table 6.2: 16-bit s-box

6.3 The importance of the combined effect of s-box and permutation towards the strength of SPN

Although it can be generally said that the strength of the cipher can be improved by increasing the number of rounds and/or the size of the s-box, it is not always true. The strength also depends on permutation components. Here is an illustration of how we can use their weakness to strengthen the attack. Following is a sample of an s-box and a permutation used to illustrate this point. We assumed a 24-bit cipher, with 4-bit s-box and with 7 rounds. Tables 6.2 and 6.3 shows the 16-bit s-box and its differential table respectively. Tables 6.4 and 6.5 shows the permutation.

In Tables that represent s-box, u is the s-box input and v is the output. In permutation Tables, a value i in first row denotes i^{th} bit of the input and a value j in second row denotes j^{th} bit of the output. The i^{th} bit in the input is shifted to j^{th} bit in the output.

In a differential trail, lesser number of active s-boxes results in a higher probability, which means lesser number of plaintext pairs have to be analyzed. This also means that the probability of success of the attack is high. We discuss an efficient way to set up a good differential trail. The higher values in the differential table are highlighted. There are 6 s-boxes at each round of the cipher. The preferred s-box values at the third row of the permutation table highlights the fact that, if we choose these values as the output difference of the s-boxes while setting up a differential trail, it will

$x\triangle/y\triangle$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	2	0	0	0	**4**	0	2	0	0	0	2	0	2	0	**4**
2	0	0	2	2	0	2	0	2	**4**	0	0	0	2	0	0	2
3	0	0	2	0	0	0	0	2	**4**	2	0	0	2	**4**	0	0
4	0	0	2	**4**	2	0	0	0	0	0	0	2	**4**	2	0	0
5	0	2	2	0	2	0	0	2	0	**4**	0	0	**4**	0	0	0
6	0	2	0	0	2	0	0	0	0	2	0	**4**	2	0	**4**	0
7	0	2	0	2	2	2	0	0	0	0	0	0	2	0	**4**	2
8	0	0	2	2	2	**6**	0	0	0	0	**4**	0	0	0	0	0
9	0	2	2	0	2	0	0	2	0	0	**4**	0	0	0	0	**4**
10	0	0	2	0	0	0	**4**	2	0	0	2	0	0	**4**	0	2
11	0	0	2	0	0	0	0	2	**4**	2	2	2	0	2	0	0
12	0	0	0	0	0	0	**8**	0	**4**	0	0	0	0	0	**4**	0
13	0	2	0	**6**	0	2	0	2	0	**4**	0	0	0	0	0	0
14	0	2	0	0	2	0	0	0	0	2	2	2	0	2	**4**	0
15	0	2	0	0	2	0	**4**	0	0	0	2	**4**	0	0	0	2

Table 6.3: Difference frequency table

i	1	2	3	4	5	6	7	8	9	10	11	12
j	8	22	2	23	7	1	3	6	12	20	15	18
Preferred $\triangle Y$ of the corresponding s-box	1,2,4,8,5				1,2,4,8,6,9				1,2,4,8,5			

Table 6.4: Permutation part I

i	13	14	15	16	17	18	19	20	21	22	23	24
j	9	14	19	21	17	10	4	11	13	5	24	16
Preferred $\triangle Y$ of the corresponding s-box	1,2,4,8				1,2,4,8,5				1,2,4,8,9			

Table 6.5: Permutation part II

result in a narrow trail with single active s-box at each round. For example, the 6^{th} and the 7^{th} bit of one round will be permuted to the 1^{st} and the 3^{rd} bit of the next round. The 1^{st} and the 3^{rd} bit belong to the same s-box in the next round. This means choosing 6 as the output difference at the second s-box of any round will make only the first s-box of the following round active. So the value 6 at the second column of the third row of the permutation table corresponds to the 6^{th} and the 7^{th} bit.

It is obvious to see the values 1,2,4 and 8 in every column of the preferred s-box output row, becuase they have a single "1" in their bits. Choosing one of them at a round will end up in a single active s-box in the next round. More number of "1" bits at the output difference of the s-box at any round will lead to many active s-boxes in the following round after permutation. In all the rounds, we need not follow this method. In the last but one round, we can choose any output-difference that has a higher probability. Even though this may lead to more than one active s-boxes in the last row, the number of active s-boxes in the last row will not count towards the total probability of the trail. But, it may count towards the search space of the partial key bits.

Another important point to be noted is, if the s-box differential table has more higher values on the columns of 1,2,4 and 8 than other columns, then it shows a possible weakness in the design of the s-box. At the same time, having higher values at columns other than 1,2,4 and 8 and having a permutation with these values as the preferred output difference is also a weakness. We should note that it is mathematically not possible to design an ideal s-box.

As a conclusion, permutation is used to compensate the unavoidable design issues of the s-box. Assuming we have higher differential values on columns other than 1,2,4 and 8, we tend to choose the other values as the output-difference of the s-box in every round to get a higher probability. But, if the permutation is well designed and it does not have the chosen values as the preferred output, then the trail will not be narrow and eventually will lead to lesser probability with more active s-boxes in most of the rounds. Once we reach more active s-boxes at any of the intermediate rounds, then it will spread further in the later rounds leading to many more active s-boxes. The resulting trail will not be efficient. In other words, we need more plaintext pairs to extract the partial key bits of the last round.

6.4 Sample trails to illustrate our claim

Tables 6.6, 6.7 and 6.8 give a few efficient trails that are set based on the techniques discussed in the former section. In trail 1, in the first round, we selected (12,6) as the ($\triangle X,\ \triangle Y$) in the second column becuase the value 6 is available as the preferred output difference in the permutation table. It resulted in 10 as $\triangle X$ in the first s-box of the second round. We do not have any preferred output differences other than 1,2,4 and 8. For 10 as $\triangle X$ (refer the s-box differential table), we can choose 2, which is the best option. If we choose 2, we are certain that we will get only one active s-box at the third round. We got 4 as $\triangle X$ in the first s-box of the third round. Selecting 3 or 12 as $\triangle Y$ at this point will result in two active s-boxes in the next round. So we can choose either 2 or 4. We chose 4. This resulted in 4 as $\triangle X$ in the fourth column of the fourth round. We again chose 4 as $\triangle Y$, which resulted in 8 as $\triangle X$ in the second column of the fifth round. In the second column, we have 1,2,4,8,6 and 9 as preferred outputs. The best choice (refer corresponding s-box difference table) is either 2,3 or 4. We chose 4 as $\triangle Y$. We got 8 as $\triangle X$ again in the first column of the sixth round.

We need not bother choosing the $\triangle Y$s, because the number of active s-boxes in the last round does not affect the total probability of the trail. In our case, we have 5 as the preferred $\triangle Y$ in the first column, which possesses a high probability corresponding to 8 as $\triangle X$. So we chose 5 as $\triangle Y$. The total probability of the complete trail is 4.58e-5. We extracted the key by analyzing a random set of 32767 text pairs. The number of plaintext pairs we used is 2^{15}, out of the total 2^{24}. As a conclusion, it is a reasonable attack.

Similarly the reader can verify other trails in corresponding tables. In most of these trails we reached probabilities with the value $9.10e-5$ and in trail 7 we reached a lowest of $2.44e-4$.

6.5 The importance of permutation

Now we can change our permutation with no preferred output difference values other than 1,2,4 and 8. Tables 6.9 and 6.10 shows this permutation.

With this permutation, we found that it is very difficult to get differential trails with total probability higher than or equal to 9e-5. Tables 6.11 and 6.12 show this situation. We are discussing a cipher with 7 rounds. If number of rounds is more than 7, it is easy to see that it is impossible to get

Round	s_1	s_2	s_3	s_4	s_5	s_6	prob. ρ	Total prob. σ
Trail 1 - 32767 plaintext pairs								
1	0,0	12,6	0,0	0,0	0,0	0,0	0.500	5.00e-1
2	10,2	0,0	0,0	0,0	0,0	0,0	0.125	6.25e-2
3	4,4	0,0	0,0	0,0	0,0	0,0	0.125	7.81e-3
4	0,0	0,0	0,0	0,0	0,0	4,4	0.125	9.77e-4
5	0,0	8,4	0,0	0,0	0,0	0,0	0.125	1.22e-4
6	8,5	0,0	0,0	0,0	0,0	0,0	0.375	4.58e-5
7	0,0	0,0	0,0	0,0	0,0	6,0		4.58e-5
Trail 2 - 29127 plaintext pairs								
1	0,0	12,6	0,0	0,0	0,0	0,0	0.500	5.00e-1
2	10,2	0,0	0,0	0,0	0,0	0,0	0.125	6.25e-2
3	4,4	0,0	0,0	0,0	0,0	0,0	0.125	7.81e-3
4	0,0	0,0	0,0	0,0	0,0	4,4	0.125	9.77e-4
5	0,0	8,5	0,0	0,0	0,0	0,0	0.375	3.67e-4
6	8,5	4,3	0,0	0,0	0,0	0,0	0.375 0.250	3.43e-5
7	2,0	4,0	0,0	0,0	0,0	6,0		3.43e-5
Trail 3 - 16383 plaintext pairs								
1	0,0	0,0	0,0	0,0	0,0	5,9	0.250	2.50e-1
2	0,0	0,0	0,0	9,2	0,0	0,0	0.125	3.13e-2
3	0,0	0,0	0,0	0,0	2,8	0,0	0.250	7.81e-3
4	0,0	0,0	0,0	0,0	8,5	0,0	0.375	2.93e-3
5	0,0	0,0	6,4	0,0	0,0	0,0	0.125	3.62e-4
6	0,0	0,0	0,0	0,0	1,5	0,0	0.250	9.16e-5
7	0,0	0,0	6,0	0,0	0,0	0,0		9.16e-5
Trail 4 - 10922 plaintext pairs								
1	0,0	0,0	0,0	0,0	8,5	0,0	0.375	3.75e-1
2	0,0	0,0	6,4	0,0	0,0	0,0	0.125	4.69e-2
3	0,0	0,0	0,0	0,0	1,5	0,0	0.250	1.17e-2
4	0,0	0,0	6,4	0,0	0,0	0,0	0.125	1.46e-3
5	0,0	0,0	0,0	0,0	1,5	0,0	0.250	3.67e-4
6	0,0	0,0	6,11	0,0	0,0	0,0	0.250	9.16e-5
7	0,0	0,0	1,0	2,0	4,0	0,0		9.16e-5

Table 6.6: Differential trails

Round	s_1	s_2	s_3	s_4	s_5	s_6	prob. ρ	Total prob. σ
Trail 5 - 38836 plaintext pairs								
1	8,5	0,0	0,0	0,0	0,0	0,0	0.375	3.75e-1
2	0,0	0,0	0,0	0,0	0,0	6,4	0.125	4.69e-2
3	0,0	8,4	0,0	0,0	0,0	0,0	0.125	5.86e-3
4	8,5	0,0	0,0	0,0	0,0	0,0	0.375	2.20e-3
5	0,0	0,0	0,0	0,0	0,0	6,4	0.125	2.75e-4
6	0,0	8,5	0,0	0,0	0,0	0,0	0.375	1.03e-4
7	8,0	4,0	0,0	0,0	0,0	0,0		1.03e-4
Trail 6 - 32766 plaintext pairs								
1	0,0	0,0	0,0	0,0	8,5	0,0	0.375	3.75e-1
2	0,0	0,0	6,4	0,0	0,0	0,0	0.125	4.69e-2
3	0,0	0,0	0,0	0,0	1,1	0,0	0.125	5.86e-3
4	0,0	0,0	2,8	0,0	0,0	0,0	0.250	1.47e-3
5	0,0	0,0	1,5	0,0	0,0	0,0	0.250	3.62e-4
6	0,0	0,0	0,0	0,0	5,9	0,0	0.250	9.16e-5
7	0,0	0,0	2,0	0,0	8,0	0,0		9.16e-5
Trail 7 - 12288 plaintext pairs								
1	0,0	12,8	0,0	0,0	0,0	0,0	0.250	2.50e-1
2	0,0	2,8	0,0	0,0	0,0	0,0	0.250	6.25e-2
3	0,0	2,8	0,0	0,0	0,0	0,0	0.250	1.56e-2
4	0,0	2,8	0,0	0,0	0,0	0,0	0.250	3.91e-3
5	0,0	2,8	0,0	0,0	0,0	0,0	0.250	9.77e-4
6	0,0	2,8	0,0	0,0	0,0	0,0	0.250	2.44e-4
7	0,0	2,0	0,0	0,0	0,0	0,0		2.44e-4
Trail 8 - 21844 plaintext pairs								
1	0,0	0,0	0,0	0,0	12,8	0,0	0.250	2.50e-1
2	0,0	0,0	0,0	0,0	8,5	0,0	0.375	9.38e-2
3	0,0	0,0	6,4	0,0	0,0	0,0	0.125	1.17e-2
4	0,0	0,0	0,0	0,0	1,5	0,0	0.250	2.92e-3
5	0,0	0,0	6,4	0,0	0,0	0,0	0.125	3.67e-4
6	0,0	0,0	0,0	0,0	1,5	0,0	0.250	9.16e-5
7	0,0	0,0	6,0	0,0	0,0	0,0		9.16e-5

Table 6.7: Differential trails

Round	s_1	s_2	s_3	s_4	s_5	s_6	Individual prob. ρ	Total prob. σ
Trail 9 - 32766 plaintext pairs								
1	0,0	0,0	0,0	0,0	0,0	12,8	0.250	2.50e-1
2	0,0	0,0	0,0	8,4	0,0	0,0	0.125	3.12e-2
3	0,0	0,0	0,0	4,2	0,0	0,0	0.125	3.91e-3
4	0,0	0,0	0,0	0,0	2,8	0,0	0.250	9.77e-4
5	0,0	0,0	0,0	0,0	8,5	0,0	0.375	3.66e-4
6	0,0	0,0	6,11	0,0	0,0	0,0	0.250	9.16e-5
7	0,0	0,0	1,0	2,0	4,0	0,0		9.16e-5

Table 6.8: Differential trail

i	1	2	3	4	5	6	7	8	9	10	11	12
j	17	8	11	14	18	21	12	15	1	19	22	16
Preferred $\triangle Y$ of the corresponding s-box	1,2,4,8				1,2,4,8				1,2,4,8			

Table 6.9: Permutation part I

i	13	14	15	16	17	18	19	20	21	22	23	24
j	2	5	20	23	3	6	9	24	4	7	10	13
Preferred $\triangle Y$ of the corresponding s-box	1,2,4,8				1,2,4,8				1,2,4,8			

Table 6.10: Permutation part II

the probability compared to that of the earlier case. With the permutation (table 6.4) in the earlier case, we could easily get many probabilities that were nearly equal to $9.10e-5$ and few of them equal to $2.0e-4$. But, with the current permutation it is impossible to achieve probabilities greater than or equal to $9.10e-5$.

This permutation is designed in such a way that no two bits of $\triangle Y$ of any s-box in a round is permuted to the same s-box in the next round. That is why we do not have values other than 1,2,4 and 8 in the preferred $\triangle Y$ row of the permutation table. Refer sample trails in Tables 6.11 and 6.12.

6.6 Other insights into cryptanalysis of SPN

This section includes two more important aspects of the cryptanalysis.

6.6.1 Case 1: The effect of sequencing the plaintext pairs

For cryptanalysis to work well on SPN, we have to make sure that the set of plaintext pairs chosen are random. If we give a sequence of plaintext pairs for analysis, it will have a uniform behavior (any two text pairs of the sequence coming into the SPN are the same in most of their bits) at the transformations of the intermediate rounds. Eventually, cryptanalysis will never follow our probability assumption (individual probability of one round is independent of the individual probability of the next round). Here we take two examples, one is a 16 bit SPN with 4 bit s-box and the other one is a 24 bit SPN with 6 bit s-box.

SPN 1: 16-bit, 4-bit s-box and 5 rounds - Tables 6.13 and 6.14
We analyzed the SPN mentioned above with two trails shown in table 6.15. Table 6.16 that shows the difference between running differential cryptanalysis with sequence of plaintext pairs and with a random set of plaintext pairs.

For each of the trails, we performed differential cryptanalysis five times with a sequence of plaintext pairs and once with a random set of plaintext pairs. The result is shown in table 6.16. When the plaintext pairs are in a sequence, the correct key could not be extracted. There may be some exceptions if we hit the right set of sequence. But we can see that with a random set of plaintext pairs, we can always extract the partial key bits.

Round	s_1	s_2	s_3	s_4	s_5	s_6	Individual Prob. ρ	Total probσ
Trail 1 - 32768 plaintext pairs								
1	2,8	0,0	0,0	0,0	0,0	0,0	0.250	2.50e-1
2	0,0	0,0	0,0	0,0	8,2	0,0	0.125	3.13e-2
3	0,0	0,0	8,2	0,0	0,0	0,0	0.125	3.91e-3
4	0,0	0,0	0,0	0,0	0,0	4,4	0.125	4.88e-4
5	0,0	2,8	0,0	0,0	0,0	0,0	0.250	1.22e-4
6	0,0	0,0	0,0	0,0	4,3	0,0	0.250	3.05e-5
7	0,0	0,0	8,0	0,0	0,0	1,0		3.05e-5
Trail 2 - 65536 plaintext pairs								
1	2,8	0,0	0,0	0,0	0,0	0,0	0.250	2.50e-1
2	0,0	0,0	0,0	0,0	8,2	0,0	0.125	3.13e-2
3	0,0	0,0	8,2	0,0	0,0	0,0	0.125	3.91e-3
4	0,0	0,0	0,0	0,0	0,0	4,4	0.125	4.89e-4
5	0,0	2,8	0,0	0,0	0,0	0,0	0.250	1.22e-4
6	0,0	0,0	0,0	0,0	4,4	0,0	0.125	1.52e-5
7	0,0	4,0	0,0	0,0	0,0	0,0		1.52e-5
Trail 3 - 65536 plaintext pairs								
1	2,8	0,0	0,0	0,0	0,0	0,0	0.250	2.50e-1
2	0,0	0,0	0,0	0,0	8,2	0,0	0.125	3.13e-2
3	0,0	0,0	8,2	0,0	0,0	0,0	0.125	3.91e-3
4	0,0	0,0	0,0	0,0	0,0	4,4	0.125	4.89e-4
5	0,0	2,8	0,0	0,0	0,0	0,0	0.250	1.22e-4
6	0,0	0,0	0,0	0,0	4,2	0,0	0.125	1.52e-5
7	0,0	0,0	8,0	0,0	0,0	0,0		1.52e-5
Trail 4 - 302144 plaintext pairs								
1	0,0	2,8	0,0	0,0	0,0	0,0	0.250	2.50e-1
2	0,0	0,0	0,0	0,0	4,4	0,0	0.125	3.13e-2
3	0,0	4,2	0,0	0,0	0,0	0,0	0.125	3.91e-3
4	0,0	0,0	1,1	0,0	0,0	0,0	0.125	4.89e-4
5	0,0	0,0	0,0	1,5	0,0	0,0	0.250	1.22e-4
6	0,0	8,2	0,0	0,0	0,0	2,8	0.125 0.250	3.81e-6
7	1,0	0,0	1,0	0,0	0,0	0,0		3.81e-6

Table 6.11: Differential trails - Part I

Round	s_1	s_2	s_3	s_4	s_5	s_6	Individual prob. ρ	Total prob.σ
Trail 5 - 21844 plaintext pairs								
1	0,0	0,0	2,8	0,0	0,0	0,0	0.250	2.50e-1
2	8,2	0,0	0,0	0,0	0,0	0,0	0.125	3.13e-2
3	0,0	0,0	2,8	0,0	0,0	0,0	0.250	7.81e-3
4	8,2	0,0	0,0	0,0	0,0	0,0	0.125	9.77e-4
5	0,0	0,0	2,8	0,0	0,0	0,0	0.250	2.44e-4
6	8,5	0,0	0,0	0,0	0,0	0,0	0.375	9.16e-5
7	0,0	1,0	0,0	4,0	0,0	0,0		9.16e-5
Trail 6 - 32768 plaintext pairs								
1	0,0	0,0	0,0	0,0	2,8	0,0	0.250	2.50e-1
2	2,8	0,0	0,0	0,0	0,0	0,0	0.250	6.25e-2
3	0,0	0,0	0,0	0,0	8,2	0,0	0.125	7.81e-3
4	0,0	0,0	8,4	0,0	0,0	0,0	0.125	9.76e-4
5	0,0	0,0	0,0	0,0	2,8	0,0	0.250	2.44e-4
6	2,8	0,0	0,0	0,0	0,0	0,0	0.250	6.10e-5
7	0,0	0,0	0,0	0,0	8,0	0,0		6.10e-5
Trail 7 - 61166 plaintext pairs								
1	0,0	0,0	0,0	0,0	0,0	2,8	0.250	2.50e-1
2	1,1	0,0	0,0	0,0	0,0	0,0	0.125	3.13e-2
3	0,0	0,0	0,0	4,2	0,0	0,0	0.125	3.90e-3
4	0,0	0,0	0,0	0,0	1,1	0,0	0.125	4.89e-4
5	0,0	0,0	0,0	0,0	0,0	1,1	0.125	6.10e-5
6	0,0	0,0	0,0	8,5	0,0	0,0	0.375	2.29e-5
7	0,0	8,0	0,0	0,0	0,0	2,0		2.29e-5

Table 6.12: Differential trails - part II

4-bit s-box																
u	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
v	e	5	d	2	c	9	0	1	a	f	8	7	6	b	3	4

Table 6.13: 4-bit s-box

16-bit permutation																
i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
j	9	11	5	1	16	15	14	8	3	13	2	6	7	12	10	4

Table 6.14: 16-bit permutation

Trail 1				
Round	s_1	s_2	s_3	s_4
1	2,8	0,0	0,0	0,0
2	0,0	0,0	8,4	0,0
3	0,0	0,0	0,0	8,4
4	0,0	0,0	1,5	0,0
5	0,0	4,0	0,0	8,0
Total Probability σ: 9.76e-4				
Key	4	**3**	8	**3**

Trail 2				
Round	s_1	s_2	s_3	s_4
1	0,0	0,0	0,0	12,6
2	0,0	0,0	5,2	0,0
3	4,12	0,0	0,0	0,0
4	0,0	0,0	10,6	0,0
5	4,0	0,0	0,0	8,0
Total Probability σ: 3.91e-3				
Key	**4**	3	8	**3**

Table 6.15: Trails

Run type	#plaintext pairs/Range	Key and their counts
Results corresponding to the trail 1, for example 28:1 at column 3 is interpreted as key [0 1 0 c], becuase 1*16+12, and it has count 1, Actual key is [0 3 0 3], which in our representation is 51		
Sequence	3072/(0 0 0 0) to (0 b f f)	28:1, 30,68,71,77,78,...:1
	3072/(0 c 0 0) to (1 7 f f)	72:2, 75,120,128,.., 187:1
	4096/(4 0 0 0) to (4 f f f)	67:8, 19,35,42,73,74:5
	5120/(7 0 0 0) to (9 3 f e)	51:64, 57:55, 54:37, 58:28
	5120/(b 0 0 0) to (d 3 f e)	0,3,11,15,17, ..., 205:1
Random	3072	51:18, 48:13, 57:15, 53:7
Results corresponding to the trail 2, Actual key:(4*16+4=67)		
Sequence	1280/(0 0 0 0) to (0 9 f 7)	19:4, 22,35,38,211,214:4
	1280/(3 0 0 0) to (3 9 f 7)	128:9, 3,35:5, 179,176,131,137:9
	1536/(3 9 f 7) to (4 5 f 6)	131:3, 179,211,227:3
	1536/(7 5 f 6) to (8 1 f 5)	67:20, 131:19,179:18
	1792/(b 1 f 5) to (b f f 4)	19:6, 35,26,211:6
Random	1280	67:13, 131:10

Table 6.16: Analysis results

6-bit s-box																
u	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
v	56	2	15	16	37	40	49	60	17	32	18	8	20	29	41	27
u	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
v	3	14	53	6	52	61	10	59	26	4	13	50	47	12	63	7
u	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
v	1	38	11	22	24	58	57	45	55	33	51	9	44	30	42	48
u	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
v	31	5	19	43	54	39	23	46	21	35	0	28	36	34	62	25

Table 6.17: S-box

24-bit permutation												
i(From)	1	2	3	4	5	6	7	8	9	10	11	12
j(To)	8	22	2	23	7	1	3	6	12	20	15	18
i(From)	13	14	15	16	17	18	19	20	21	22	23	24
j(To)	9	14	19	21	17	10	4	11	13	5	24	16

Table 6.18: 24-bit permutation

Trails set				
Round	s_1	s_2	s_3	s_4
1	0,0	0,0	18,18	0,0
2	0,0	0,0	18,18	0,0
3	0,0	0,0	18,13	0,0
4	0,0	4,0	0,0	4,0
Total Probability: 9.76e-4				
Key	14	**3**	42	**20**

Table 6.19: Trail used for analysis

Run	#plaintext pairs/Range	Key and their counts
Results corresponding to this trail, Actual key:(3*64+20)=212		
Random	4096	212:9, 209:4, 202,216:3
Sequence	4096/(0 0 0 0) to (0 1 2f 3f)	192,201,1472:1
	4096/(7 8 8 8) to (7 a 8 7)	107,116,966,971:1
	4096/(4 5 6 4) to (4 7 6 3)	212:8
	4096/(6 8 f 2) to (6 a f 1)	233:2
	4096/(4 0 0 0) to (4 2 f 3f)	212:28

Table 6.20: Analysis results

SPN 2 configuration: 24-bit, 6-bit s-box and 4 rounds - Tables 6.17 and 6.18

6.6.2 Case 2: The effect of no permutation in the SPN cryptanalysis

If we do not have permutation at all in our SPN, then differential cryptanalysis is not a suitable tool to extract the key. In every round, the output is not permuted and it is sent directly to the s-box down to the next round. This strongly affects the assumption: individual probabilities between the rounds are independent. Even though we provide random set of plaintext pairs, differential cryptanalysis does not seem to work well. In this context, we have analyzed many SPNs with different block sizes and s-box sizes. All analysis gave the same behavior with a negligible number of exceptions. What really happens is, the analysis never extracts the partial key which is targeted. In other words, the count corresponding to the correct partial key is smaller than the counts corresponding to the wrong keys. This clearly shows that, without a proper permutation, the probability assumption we made is incorrect.

SPN with no permutation can be seen as an SPN with a simple straight permutation. SPN without permutation is just a sequence of substitution and XOR operations. If the key is fixed, the whole cipher is just a substitution. In the case of permutation, output in each round is permuted and reaches the next round as input. Whereas in SPN without permutation, the output in one round reaches the same column of the next round unaltered. In case of SPN with permutation, every other bit contributes to a change in every round.

24-bit cipher - Tables 6.21, 6.22, 6.23 and 6.24 The above results show that, having no permutation affects the probability assumption. The total expected probability is the complete probability of the trail. It is with this probability that we expect the input difference in the last round. In the case where there is no permutation, the resulted probability is not correct. Refer table 6.24 for details. In the case of SPN with permutation, the different number of plaintext pairs that arrive at the input of the last round is greated than 100. But in the case of the SPN without permutation, this number is significantly less. For example, in one of the above cases, for 5536 plaintext pairs, only 64 different pairs appeared at the input of the last round.

6-bit s-box																
u	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
v	56	2	15	16	37	40	49	60	17	32	18	8	20	29	41	27
u	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
v	3	14	53	6	52	61	10	59	26	4	13	50	47	12	63	7
u	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
v	1	38	11	22	24	58	57	45	55	33	51	9	44	30	42	48
u	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
v	31	5	19	43	54	39	23	46	21	35	0	28	36	34	62	25

Table 6.21: S-box

24-bit permutation, Part 1												
i	1	2	3	4	5	6	7	8	9	10	11	12
j	8	22	2	23	7	1	3	6	12	20	15	18
Preferred $\triangle Y$ of the corresponding s-box	1,2,4,8,16,32, 9,20,24						1,2,4,8,16,32, 3,48					

Table 6.22: Permutation - part I

24-bit permutation, Part 2												
i	13	14	15	16	17	18	19	20	21	22	23	24
j	9	14	19	21	17	10	4	11	13	5	24	16
Preferred $\triangle Y$ of the corresponding s-box	1,2,4,8,16,32, 12,18,33						1,2,4,8,16,32, 9,36					

Table 6.23: Permutation - part II

Run	Total expected probability	Resulted probability	♯ of diff. txtpair inputs at last round	Key found?
With proper permutation				
1 (seed 35)	7.32e-4	$\frac{268}{5536} = 0.048$	$\gg 100$	Yes
2 (seed 58)		$\frac{257}{5460} = 0.047$	$\gg 100$	Yes
3 (seed 192)		$\frac{189}{4095} = 0.046$	$\gg 100$	Yes
Without permutation				
1 (seed 35)	4.89e-4	$\frac{0}{5536}$	64	No
2 (seed 58)		$\frac{0}{4095}$	64	No
3 (seed 192)		$\frac{0}{4095}$	64	No
3 (seed 192)		$\frac{0}{20480}$	64	No

Table 6.24: Analysis results

So, differential cryptanalysis is not the right choice for SPNs without permutation or blockwise permutation.

6.7 SPN analysis with a bad permutation

There are several ways to answer the following questions:

- How can we set up a better trail?
- Is the SPN cipher strong enough (against differential cryptanalysis)?
- Are there many optimal trails available?

In this section, we try to answer the above questions. Referring to Figure 6.1, there are many preferred $\triangle Y$s at every s-box of all rounds. These preferred $\triangle Y$s are determined based on the permutation. In every round, we use the same s-box and the same permutation. These preferred $\triangle Y$s are to make sure that the trail is narrow. Once we choose one of these $\triangle Y$s at an s-box, it may lead to one or more $\triangle X$s activating one or more s-boxes in the next round after permutation. As we choose $\triangle Y$ always from the set of preferred $\triangle Y$s, we will mostly end up with only one active s-box at each round.

Referring to Figure 6.1, we have a set of $\triangle X$s as the input of each s-box in each round. These $\triangle X$s correspond to the result of permutation of all

the preferred $\triangle Y$s in previous round. In general, if we have more number of $\triangle Y$s at the s-boxes and more number of $(\triangle X,\ \triangle Y)$ pairs, formed from the $\triangle X$ set and the $\triangle Y$ set occurring in the s-box differential table having large probabilities, then we have greater chances of finding one or more efficient trails. Hence we have greater chances for successful cryptanalysis. This explains the importance of the permutation clearly.

6.8 Summary

In this project, we classify the permutations into either good, bad, or uniform. Good permutation means a random permutation which have preferred $\triangle Y$ values only in powers of 2 (refer Section 6.3). A good permutation strengthens the cipher with respect to the differential cryptanalysis. Because, it diffuses the bits wide into the cipher. Attacking reduced round versions of the cipher will be restricted to lesser rounds. Bad permutation is a random permutation which have values also not in powers of 2 as its preferred $\triangle Y$. A bad permutation does not diffuse the bits efficiently. There will be many efficient differential trails available. Uniform permutation is either using no permutation or blockwise permutation. Uniform permutation is called so because it gives a certain degree of uniformity into the cipher. Uniformity in the cipher makes the differential cryptanalysis ineffective. Though we choose large number of plaintext pairs, the number of different texts that appear at the last round input would be very less. We summarize the results in the following Table 6.25.

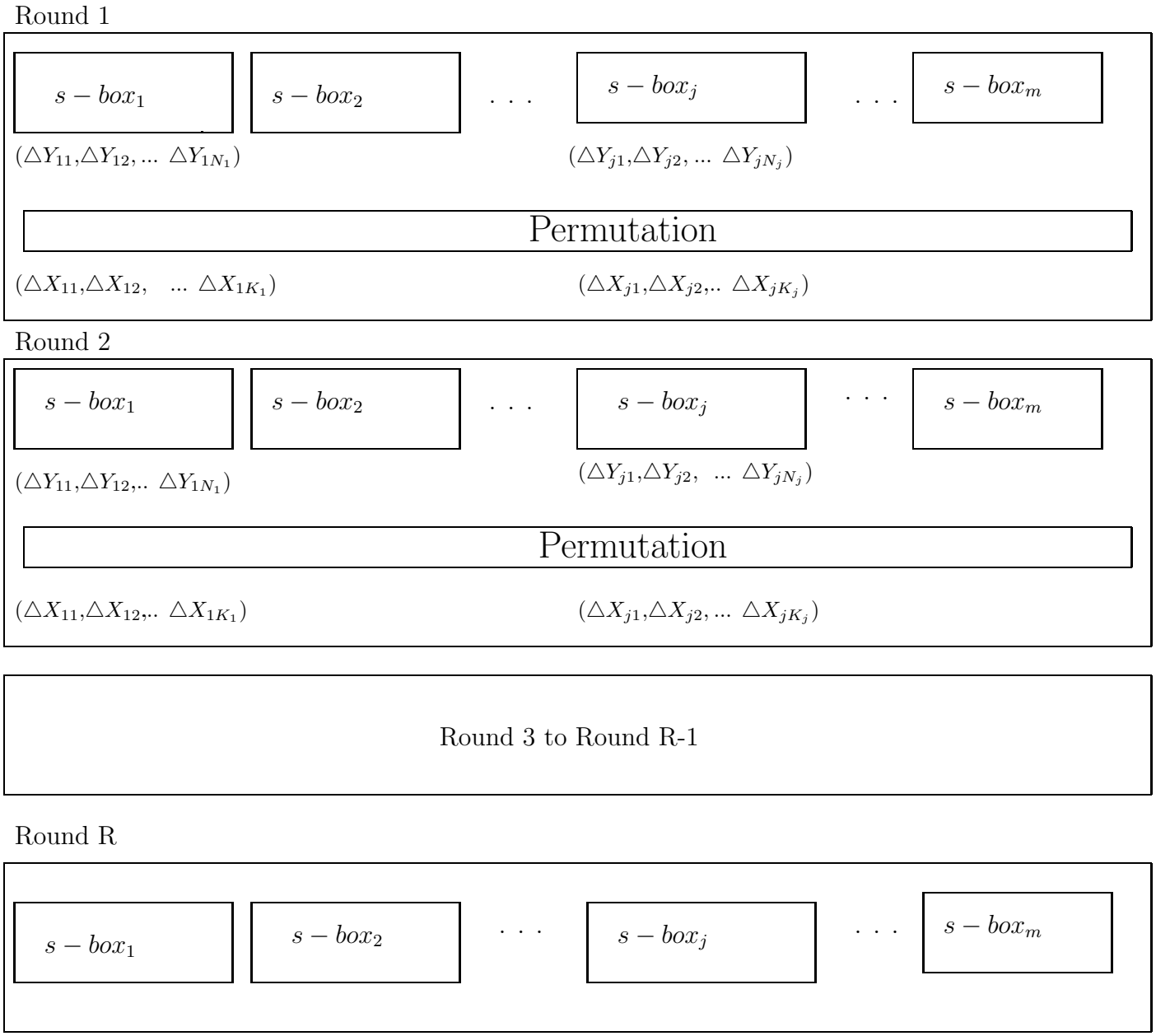

Figure 6.1: SPN Conclusion

No. of ciphers analyzed / No. of trails	Block length	No. of rounds	No. of chosen plaintext pairs required to extract the key	Type and description
10/10	24 bits	7	2^{13} to 2^{15}	Used bad permutations, refer Tables 6.4, 6.5, 6.6, 6.7 and 6.8
10/5	24 bits	7	2^{15} to 2^{18}	Used good permutations, refer Tables 6.9, 6.10, 6.11 and 6.12
15/5	16 bits	5	Irregular behavior 2^{10} to 2^{11}	Uniformity in the cipher is introduced through sequential plaintexts, refer Tables 6.13, 6.14, 6.15, 6.16, 6.17, 6.18, 6.19 and 6.20 Random set of plaintext pairs
	24 bits	4	Irregular behavior 2^{13}	Sequential plaintext pairs, refer Tables 6.17, 6.18, 6.19 and 6.20 Random set of plaintext pairs
5/5	24 bits	6	2^{13} to 2^{15}	Cipher without permutation, refer Tables 6.21, 6.22, 6.23 and 6.24; If we use permutation, we get large number of different text pairs at last round input, otherwise, we get less than 100 number of different text pairs at last round input

Table 6.25: SPN Analysis Summary

Chapter 7

Differential analysis on Rijndael-like ciphers

7.1 Rijndael and SPN similarity

Encryption of Rijndael can be seen as an SPN, except for the MixColumns component. Figure 7.1 illustrates this fact. Figure 7.2, created by John Savard, shows how SubBytes, ShiftRows, MixColumns and AddKey operations are performed.

7.2 Software to work on Rijndael-like ciphers

In addition to the SPN configuration and differential analysis tool which is described in chapter 6, this project includes a software to configure Rijndael-like ciphers and to perform differential analysis on them.

7.2.1 Configuring the cipher

Before we start the differential cryptanalysis, we have to configure the cipher. The analysis program first lets the user to configure the cipher before analysis. The information required to configure the cipher are:

- Block size of the cipher in number of bits
- S-box block size of the cipher in number of bits
- Cipher key: Since the key or the key generating technique has no influence on differential analysis, we are generating the cipher key and

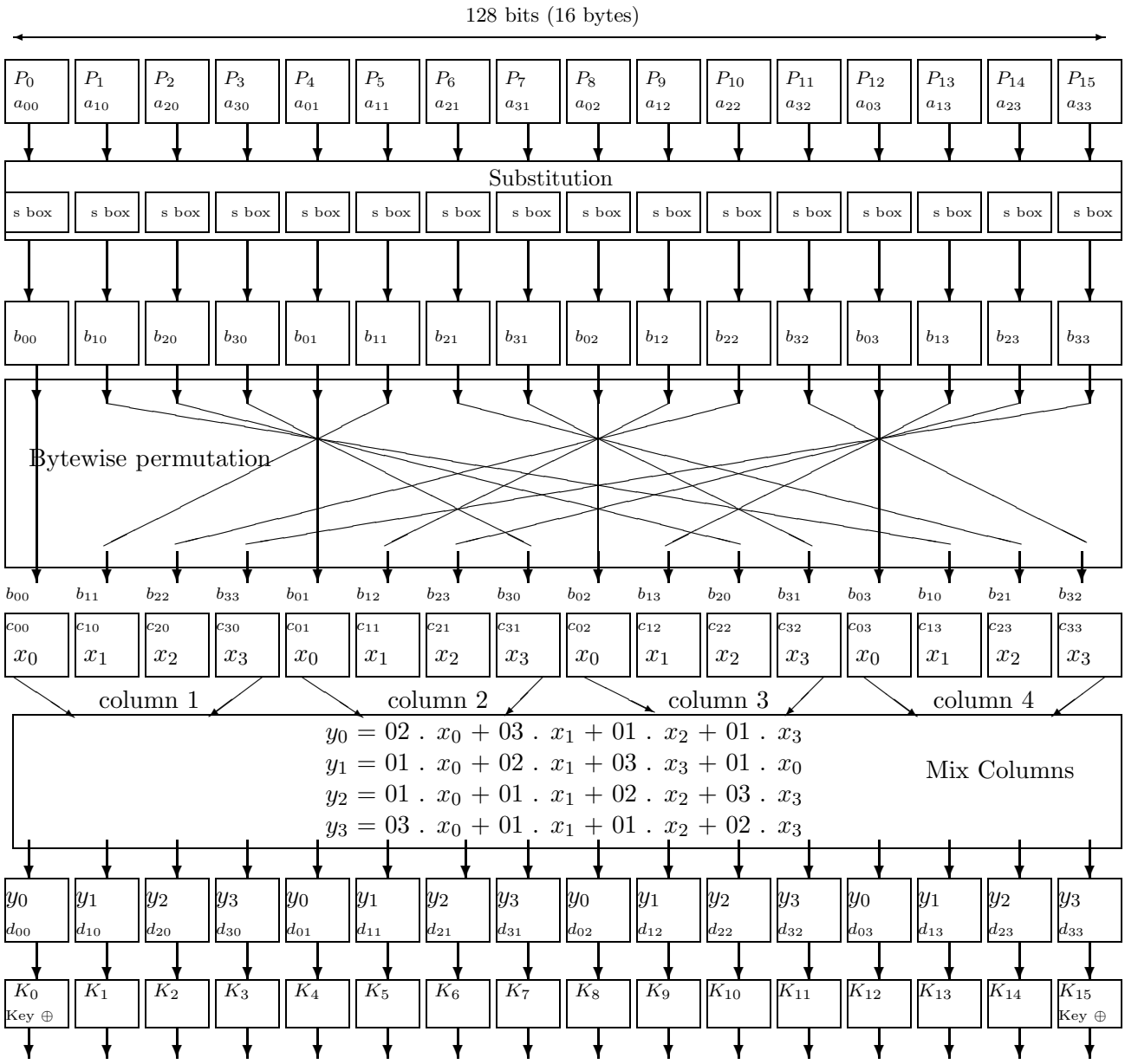

Figure 7.1: One round of 128 bit Rijndael encryption

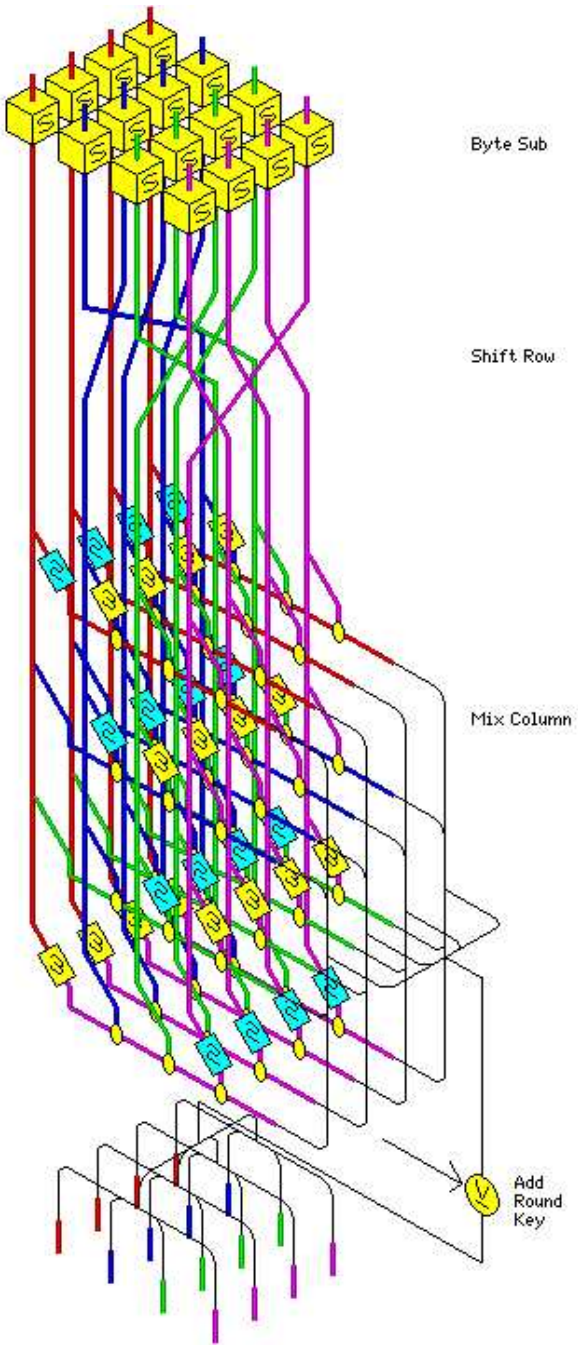

Figure 7.2: Rijndael view by John Savard

individual round keys using a fixed built-in module which is taken from R Stinson's book [35].

- Number of rounds of the cipher: For analysis, we use 4, 5 or 6 round ciphers.
- Diffusion component configuration file: We include the diffusion configuration data in a text file and provide the file to the diffusion configuration step of the analysis tool. The content of the diffusion configuration file is given below. The sequence of the values should be in this order.

```
n n
(n x 1) matrix
(n x n) matrix
```

  First line is the order n of the Mixcolumns matrix. Next is a (n x 1) matrix which represents the shift offsets for the ShiftRows component. Next is a (n x n) MixColumns matrix.

- Substitution (SubBytes) component configuration file: In the SubBytes configuring step of the analysis program, we need to give a s-box as a binary file. We can create this s-box binary file using the s-box generator tool. S-box generator tool requires a configuration file in text format to generate s-box. This text configuration file has the following contents in this sequence:
  - First is the degree n of finite field $GF(2^n)$.
  - Second is the decimal representation of the irreducible polynomial of the cipher. We have a tool to generate a list of irreducible polynomials of a given degree.
  - Third is the $n \times n$ matrix A, used in the affine transformation: $B = Ax + C$
  - Fourth is the $n \times 1$ matrix C, used in the affine transformation

  Once we generate an s-box using this tool, we can provide this s-box in the SubBytes configuration step of the main analysis tool.

A typical configuration Following is the configuration profile for 128-bit Rijndael.

- Block size is 128 bits
- S-box block size is 8 bits
- Number of rounds of the cipher is 10
- Diffusion configuration:

```
4 4
0
1
2
3
02 03 01 01
01 02 03 01
01 01 02 03
03 01 01 02
```

  The order of MixColumns matrix is (4×4). The shift offsets corresponding to rows 1, 2, 3 and 4 are 0, 1, 2 and 3 respectively. Last four lines of the file is the MixColumns matrix [9].

- Substitution configuration: We generate s-box using the s-box generator tool. We provide the following text file to the s-box generator.

```
8
283
1 1 1 1 1 0 0 0
0 1 1 1 1 1 0 0
0 0 1 1 1 1 1 0
0 0 0 1 1 1 1 1
1 0 0 0 1 1 1 1
1 1 0 0 0 1 1 1
1 1 1 0 0 0 1 1
1 1 1 1 0 0 0 1
0
1
1
0
0
0
```

```
1
1
```

The s-box generator tool creates an s-box corresponding to this configuration. Once the s-box is generated using this tool, we can provide it to the SubBytes configuration step of the main analysis tool. Rijndael is built on the finite field $GF(2^8)$, hence the degree 8 is used. The value 283 is the decimal equivalent of the irreducible polynomial $x^8 + x^4 + x^3 + x + 1$. SubBytes has an affine transformation $B = A \times Y + C$ [9]. A is a 8×8 matrix and C is a 8×1 matrix. Matrix A should have odd number of 1s to get a bijective s-box.

7.2.2 Analysis of Rijndael-like ciphers

Once we configure the target cipher, we can choose to perform differential analysis on this cipher. Differential analysis involves two steps:

1. Setting a good differential trail
2. Perform analysis to extract the partial bits of the last round key

We can repeat these two steps any number of times to extract most of the bits of the key which is used in the last round. Detailed information on setting the differential trail and extracting the key bits are available in section 6.1 in chapter 6.

7.2.3 Additional features of the analysis tool

The cipher can be configured in various ways. Possible configurations are listed in table 7.1.

7.3 Summary

There is not much concentration on Differential analysis on Rijndael-like ciphers in this project. However, a few experiments have been done.

7.3.1 Rijndael-like ciphers

We constructed Rijndael-like ciphers using the following strategy:

State State is a representation of a text block or an intermediate result of a round transformation. This project always assume the *state* to be a 4×4 matrix.

Config	Substitution	Diffusion	
		Permutation or Shiftrows	Mixcolumns
1	Random or manual	Random or manual	-
2	Random or manual	Random or manual	Mixcolumns
3	Random or manual	Shiftrows	Mixcolumns
4	Rijndael-like s-box	Random or manual	-
5	Rijndael-like s-box	Random or manual	Mixcolumns
6	Rijndael-like s-box	Shiftrows	Mixcolumns

Table 7.1: Different possible configurations

S-box Regarding s-box construction, unlike Rijndael's 8-bit s-box, we constructed 4 and 6 bit s-boxes. To construct a n bit s-box, following steps are followed.

1. Finding an irreducible polynomial of degree n. This helps in generating $GF(2^n)$.
2. Construct two matrices A and B. These matrices are used to form the affine transformation $y = Ax + B$. Matrix A will be of order n and matrix B will be a $n \times 1$ matrix.
3. With the above two components, s-box is generated.

Diffusion For all the experiments carried out in this project, diffusion component of Rijndael is used without any change. This was possible because we fixed the order of *state* to be 4.

Some of the Rijndael-like ciphers are breakable due to the small size of their s-boxes and less number of rounds. If we increase the number of rounds and/or increase the s-box size, we experience a trend same as Rijndael. We explain the strength of Rijndael-like ciphers with respect to differential cryptanalysis in the following section 7.3.2.

7.3.2 SPN and Rijndael-like ciphers: A comparison

An SPN cipher and a Rijndael-like cipher are taken for our illustration. Differential trail on these two ciphers are built. Both of the ciphers are compared.

Block length Our SPN and Rijndael-like ciphers have a same block length of 64 bits.

0	1	2	3	3	5	6	7	8	9	10	11	12	13	14	15
9	14	0	1	13	11	8	15	6	2	10	3	12	4	5	7

Table 7.2: Rijndael-like s-box

Number of rounds Both of our SPN and Rijndael-like ciphers have 5 rounds.

S-box Both ciphers use the same s-box.

- S-box block length is 4 bits
- $x^4 + x + 1$ is used as the polynomial in $GF(2^4)$
- Matrices A and B used for the affine transformation $y = Ax + B$ are

$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

and

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

respectively. S-box constructed using the above configuration is given below in Table 7.2. Difference distribution corresponding to this s-box is given in Table 7.3.

Diffusion or Permutation This is where the ciphers differ from each other. Our SPN cipher uses a random permutation and Rijndael-like cipher uses both shiftrows and mixcolumns component as defined in Rijndael.

A good differential trail is set on both the ciphers for cryptanalysis. Differential trail corresponding to the SPN is given in Table 7.4. Differential trail of the Rijndael-like cipher is given in Table 7.5. In both the Tables, a row represents a round of the cipher and a column represents an s-box. A pair of values at row i and column j denotes a pair $\triangle X, \triangle Y$ at s-box j in round i. A value of 0 indicates a pair $0, 0$. A single non zero value indicates only $\triangle X$. In this case, $\triangle Y$ is ignored since it is not important to our discussion. Similarly, the symbol $-$ indicates the rounds that are not important to our discussion.

$\triangle X/\triangle Y$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	2	2	0	2	0	2	4	2	2	0	0	0	0	0	0
2	0	2	0	2	2	2	0	0	0	4	0	0	2	0	0	2
3	0	2	2	2	0	2	0	0	4	0	0	2	0	0	2	0
4	0	0	0	0	4	2	2	0	2	0	2	0	0	0	2	2
5	0	0	4	2	0	0	2	0	0	2	0	0	0	2	2	2
6	0	4	0	2	0	2	2	2	0	0	2	0	0	2	0	0
7	0	2	0	0	0	0	4	2	0	0	0	2	2	0	2	2
8	0	2	2	0	0	0	0	0	2	0	2	0	2	2	0	4
9	0	0	0	2	0	0	0	2	2	2	2	4	0	0	0	2
10	0	0	0	4	2	0	2	0	2	0	0	2	2	2	0	0
11	0	0	2	2	2	0	0	2	0	0	4	0	2	0	2	0
12	0	0	2	0	0	4	2	0	0	2	2	2	2	0	0	0
13	0	0	2	0	2	2	0	2	0	0	0	2	0	4	0	2
14	0	0	0	0	0	2	0	2	2	2	0	0	4	2	2	0
15	0	2	0	0	2	0	0	0	0	2	2	2	0	2	4	0

Table 7.3: Difference distribution

In the Rijndael-like cipher, all s-boxes are active in the third round. It is to be noted that the diffusion used in this cipher is the same diffusion used in Rijndael. The diffusion is so effective to make all the s-boxes active even in the third round. Each plaintext pair fed into the cipher has a difference of 5 at their 11^{th} 4-bit column. All other 4-bit columns of the pair of plaintexts are same. This small change in first round is having an impact on all the columns of the text pairs in third round. But in SPN, this is not the case (refer Table 7.4). In SPN, the difference of 5 has its impact only on two 4-bit columns of the third round. This clearly shows the strength of Rijndael-like diffusion compared to an ordinary SPN permutation.

Permutation in SPN is a mere shifting of bits. It preserves the number of '1' bits in the xor difference. This is not true in case of Rijndael's diffusion. It has a different impact on the xor difference. Considering the 16 4-bit units as a *state* of 4×4 matrix, shiftrows component moves the xor differences between columns of the *state*. Whereas mixcolumns component, does an

i/j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	0	0	0	0	0	0	0	0	0	**5,2**	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	**2,9**	0	0	0	0
3	0	0	0	**8,15**	0	0	0	0	0	0	0	0	0	**2,9**	0	0
4	**10**	0	0	**2**	0	0	**4**	0	0	**1**	0	0	**2**	0	0	0
5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 7.4: SPN and Rijndael-like ciphers comparison: SPN differential trail

i/j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	0	0	0	0	0	0	0	0	0	**5,2**	0	0	0	0	0
2	**2,9**	**6,1**	**4,4**	**2,9**	0	0	0	0	0	0	0	0	0	0	0	0
3	**1**	**9**	**9**	**8**	**9**	**9**	**8**	**1**	**4**	**12**	**8**	**4**	**3**	**2**	**1**	**1**
4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 7.5: SPN and Rijndael-like ciphers comparison: Rijndael-like cipher differential trail

algebraic operation on a column of the *state*.

We will take a look into our example. Our first round xor difference, represented as a *state*, is:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{5,2} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

In shiftrows process, $\triangle Y$, which is 2, is shifted and the *state* matrix becomes:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \mathbf{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Note that the shiftrows component shifts values from one column to another. Next operation in sequence is mixcolumns. Equation 7.1 shows the

mixcolumns operation in round 1.

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \mathbf{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{2} & 0 & 0 & 0 \\ \mathbf{6} & 0 & 0 & 0 \\ \mathbf{4} & 0 & 0 & 0 \\ \mathbf{2} & 0 & 0 & 0 \end{bmatrix} \tag{7.1}$$

Matrix multiplication in equation 7.1 is algebraic. The columns of the *state* are considered as polynomials over $GF(2^4)$ and multiplied modulo $x^4 + 1$ with $c(x) = 03x^3 + 01x^2 + 01x + 02$. All elements of a *state*'s column are affected by every other element of the same column. This shows the effect of diffusion component as a whole. The *state* shown in the right hand side of equation 7.1 is the result of the first round. The values in this *state* represent $\triangle X$ values for the second round. We chose $\triangle Y$s correspondingly. Following *state* shows both the $\triangle X$s and their corresponding $\triangle Y$s. This is the result after applying substitution in round 2.

$$\begin{pmatrix} \mathbf{2,9} & 0 & 0 & 0 \\ \mathbf{6,1} & 0 & 0 & 0 \\ \mathbf{4,4} & 0 & 0 & 0 \\ \mathbf{2,9} & 0 & 0 & 0 \end{pmatrix}$$

Following is the *state* after applying shiftrows in second round.

$$\begin{pmatrix} \mathbf{9} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & \mathbf{4} & 0 \\ 0 & \mathbf{9} & 0 & 0 \end{pmatrix}$$

Now we can see that the values are shifted such that they impact all other columns of the *state*. Mixcolumns operation in second round and its result *state* is shown in equation 7.2.

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} \mathbf{9} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & \mathbf{4} & 0 \\ 0 & \mathbf{9} & 0 & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{9} & \mathbf{4} & \mathbf{3} \\ \mathbf{9} & \mathbf{9} & \mathbf{12} & \mathbf{2} \\ \mathbf{9} & \mathbf{8} & \mathbf{8} & \mathbf{1} \\ \mathbf{8} & \mathbf{1} & \mathbf{4} & \mathbf{1} \end{bmatrix} \tag{7.2}$$

This illustration shows two facts. An xor difference of value 5 in first round impacts all elements of the *state* that appears as the third round input. As a result, all the s-boxes in round three are active.

This Rijndael-like cipher uses the diffusion of Rijndael. We can also create our own diffusion using the tools of this project. Experiments are not done using non-Rijndael diffusions. Non-Rijndael diffusions may not be that effective. Despite this, we can conclude that a good diffusion will make the differential cryptanalysis ineffective.

7.3.3 Rijndael

We noticed two behaviors with respect to the 128 bit Rijndael. The differential table of the s-box has a maximum frequency of 4. In other words, maximum differential probability at any s-box of any round is $1.56e-1$. During the differential analysis, even if we choose one s-box at each round (or in other words, make one s-box active at each round), for 10 rounds, the total probability would be $1.56e-20$, which needs 2^{60} plaintext pairs minimum. Since the diffusion component is well designed, each byte of the *state* is determined by the other bytes of the other states. So we can never achieve only one active s-box at any round. When we do mixcolumns part of the diffusion, each byte of the column are affected by all bytes of the column. We are forced to choose more than one bytes of the column. Therefore, in most of the rounds, we end up in more than one active s-boxes. Considering two active s-boxes per round, the total probability would be of order $1.56e-80$. The estimated number of plaintext pairs required is higher than the plaintext pairs required for the exhaustive search. This shows that we cannot perform the differential analysis on $GF(2^8)$-Rijndael.

Chapter 8

Conclusion and possible improvements

8.1 Differential analysis and SPN

In the context of SPN, this project focused on two aspects: permutation and uniformity.

8.1.1 Permutation

- If the number of rounds is less than 7, then a good permutation is necessary in addition to the s-boxes.
- If the number of rounds is greater than 10, a good permutation increases the strength of the cipher.

This project focuses permutation in the following ways.

- Good and bad permutations are compared.
- The fact that a good permutation increases the strength of the cipher is illustrated.

8.1.2 Uniformity

Uniformity in the cipher affects differential cryptanalysis. In this project, uniformity is introduced into the cipher in the following ways.

- By taking a set of sequentially chosen plaintext pairs in differential analysis

- By configuring ciphers without permutation

We have explained the importance of permutation in SPN in the context of differential cryptanalysis. Section 6.5 of Chapter 6 describes this in detail.

8.1.3 Tools

The tools that are developed in this project can be used to:

- learn differential cryptanalysis in detail
- learn the strength of substitution and permutation component in the context of differential cryptanalysis (Sections 6.5 and 6.3 in chapter 6)
- test the reduced versions of any block cipher for their strength against differential cryptanalysis

8.2 Rijndael-like ciphers

In most of the cases, differential cryptanalysis does not work in Rijndael-like ciphers. The reasons are

- S-boxes constructed using irreducible polynomials are stronger than random s-boxes.
- Diffusion that have shiftrows and mixcolumns makes all the s-boxes active even at rounds less than five.

8.3 Further improvements

Further improvements can be made in the following directions:

- Graphical interactive environment can be developed to perform differential cryptanalysis.
- Further experiments can be done on Rijndael-like ciphers.
- Analysis tool can be improved to extract the complete key.
- Tools for testing the avalanche effect of the s-boxes can be developed.

Bibliography

[1] Carlisle Adams, Stafford Tavares, *The Structured Design of Cryptographically Good s-boxes*, Journal of Cryptology, 1990, 3:27-41.

[2] Carlisle Adams, Stafford Tavares, *Designing s-boxes for Ciphers Resistant to Differential Cryptanalysis*, Proc. of the 3rd symposium on State and Progress of Research in Cryptography, W. Wolfowicz, Ed., Fondazione Ugo Bordoni, 1993, pp. 181-190.

[3] Alex Biryukov, David Wagner, *Slide Attacks*, Fast Software Encryption 1999, Lecture Notes in Computer Science (LNCS) 1636, Springer Verlag 1999, pp. 245-259.

[4] Lawrence Brown, Jennifer Seberry, *On the design of permutation P in DES type cryptosystems* EUROCRYPT 1989, pp. 696-706.

[5] Jung Hee Cheon, Munju Kim, Kwangjo Kim,Jung-Yeun Lee, and Sung-Woo Kang, *Improved Impossible Differential Cryptanalysis of Rijndael and Crypton*, In K. Kim, editor, Information Security and Cryptology - ICISC 2001, number 2288 in Lecture Notes in Computer Science, pp. 39-49. Springer, 2001.

[6] Nicolas T Courtois and Josef Piepryzk, *Cryptanalysis of block ciphers with overdefined systems of equations*, ASIACRYPT 2002, pp. 267-187.

[7] Nicolas T Courtois, Cryptography, `http://www.minrank.org/aes`.

[8] Evan Damgaar, AES Information, `http://www.cryptomathic.com/company/aes.html`.

[9] Joan Daemen and Vincent Rijmen, *The Design of Rijndael. AES - The Advanced Encryption Standard*, Springer, ISBN 3540425802.

[10] Joan Daemen, Vincent Rijmen, *AES Proposal*, NIST AES Proposal, 1998.

[11] Joan Daemen, Vincent Rijmen, *Answers to new observations on Rijndael*, NIST, August 11, 2000

[12] Neils Ferguson, John Kesley, Stefan Lucks, Bruce Schneier, M.Stay, D.Wagner, David Wagner, and Doug Whiting, *Improved Cryptanalysis of Rijndael*, Proceedings of Fast Software Encryption - FSE'00, number 1978 in Lecture notes in Computer Science. pp. 213-230. Springer-Verlag, 2000

[13] Niels Ferguson, Richard Schroeppel, and Doug Whiting, *A Simple algebraic representation of Rijndael*, Proceedings of Selected areas in Cryptography - SAC'01, number 2259 in Lecture Notes in Computer Science, pp. 103-111. Springer-Verlag, 2001

[14] Helena Handschuh and Howard M Heys, *A Timing Attack on RC5*, Proceedings of the Selected Areas in Cryptography, pp. 306-318, August 17, 1998.

[15] Howard M Heys, *A Tutorial on Linear and differential Cryptanalysis*, Technical report CORR 2001-17, Dept. of Combinatorics and Optimization, University of Waterloo, Waterloo, Canada, 2001.

[16] Howard M Heys, *Substitution-Permutation Networks resistant to differential and linear cryptanalysis*, Journal of cryptology (1996) 9:1-19.

[17] Thomas Jakobsen and Lars R Knudsen, *The Interpolation Attack on Block Ciphers*, Fast Software Encryption, 4th International Workshop Proceedings, Springer-Verlag, 1997, pp. 28-40.

[18] Ju-Sung Kang, Seokhie Hong, Sangjin Lee, Okyeon Yi, Choonsik Park, and Jongin Lim, *Practical and Provable Security against Differential and Linear Cryptanalysis for Substitution-Permutation Networks*, ETRI Journal, Vol. 23, No. 4., Dec 2001, pp. 158-167.

[19] Kwangjo KIM, *A Study on the Construction and Analysis of Substitution s-boxes for Symmetric Cryptosystems*, December 25, 1990, Ph.d. thesis, Electrical and Computer Engineering Department, Yokahama National University

[20] Rudolf Lidl and Harald Neiederreiter, *Introduction to Finite Fields and their Applications*, Cambridge University Press, 1986

[21] S. Lucks, *The saturation attack - a bait for Twofish* Fast Software Encryption 2001, LNCS 2355, M. Matsui, Ed., Springer-Verlag, 2001, pp. 1-15.

[22] Alfred J Menezes, Paul C Van Oorschot and Scott A Vanstone, *Handbook of Applied Cryptography* CRC press, Fifth print, August 2001

[23] Sean Murphy and M J B Robshaw, *Essential Algebraic Structure Within the AES*, In, M. Yung, editor, Advances in Cryptology - CRYPTO 2002, Volume 2442 of Lecture Notes in Computer Science, pp. 1-16, Springer-Verlag, August 2002.

[24] Sean Murphy and Matt Robshaw, *New Observations on Rijndael*, August 7, 2000. Available via `http://www.csrc.nist.gov/aes`.

[25] Modes of AES.
`http://www.csrc.nist.gov/CryptoToolkit/modes/proposedmodes`.

[26] Elisabeth Oswald, Joan Daemen and Vincent Rijmen, *AES - The State of the Art of Rijndael's Security*, Technical notes, October 30, 2002. Available at `http://www.iaik.tugraz.at/aboutus/people/oswald/`.

[27] Bart Preneel, Vincent Rijmen, and Antoon Bosselaers, *Recent Developments in the Design of Conventional Cryptographic Algorithms*, Lecture Notes in Computer Science 1998, Volume 1528, pp. 105-130, Springer-Verlag 1998.

[28] Dhiren R Patel, *The AES winner*, Manuscript, AES third conference.

[29] Sangwoo Park, Soo Hak Sung, Seongtaek Chee, E-Joong yoon, and Jongin Lim, *On the security of Rijndael-Like structures against differential and linear cryptanalysis*, Advances in Cryptology, ASIACRYPT-2002, LNCS 2501, pp. 176-191, Springer-Verlag, 2002.

[30] Jean-Jacques Quisquater and Francois Koeune, *A timing attack against Rijndael*, Technical Report CG-1999/1, Université Catholique de Louvain, 1999.

[31] Rijndael Information, Available at `http://www.rijndael.com`.

[32] Rijndael designer's page,
`http://www.esat.kuleuven.ac.be/rijmen/rijndael`.

[33] Bruce Schneier, Niels Ferguson, *Practical Cryptography*, Wiley publishing, Inc.

[34] Jennifer Seberry, Xian-Mo Zhang, Yuliang Zheng, *Relationship Among Nonlinearity Criteria*, Advances in Cryptography - EUROCRYPT 1994, LNCS 950, pp. 376-388, Springer-Verlag 1995.

[35] Douglas R Stinson, *Cryptography Theory and Practice* 2nd edition, CRC Press

[36] Michael Welschenbach, *Cryptography in C and C++*, Apress.