AN ENHANCED BLOWFISH ALGORITHM BASED ON CYLINDRICAL COORDINATE SYSTEM AND DYNAMIC PERMUTATION BOX

ASHWAK MAHMOOD ALABAICHI

DOCTOR OF PHILOSOPHY
UNIVERSITI UTARA MALAYSIA
2014

Permission to Use

Abstrak

Algoritma Blowfish (BA) adalah sifer blok simetri yang menggunakan rangkaian *Feistel* untuk melakukan fungsi penyulitan dan penyahsulitan yang mudah. Kunci BA adalah pelbagai dari bit 32 ke 448 untuk memastikan tahap keselamatan yang tinggi. Walau bagaimanapun, kotak penggantian (Kotak-S) dalam BA mengambil peratus ruang memori yang tinggi dan mempunyai masalah keselamatan, terutamanya dalam kerambangtarikan output bagi teks dan fail imej yang mempunyai rentetan besar dan mempunyai bait yang serupa. Dengan demikian, objektif kajian ini adalah untuk mempertingkatkan BA bagi mengatasi masalah ini. Kajian ini melibatkan tiga fasa; reka bentuk algoritma, pelaksanaan, dan penilaian. Dalam fasa reka bentuk, Kotak-S 3D dinamik, Kotak Pilih Atur (Kotak-P) dinamik, dan Fungsi Feistal (Fungsi-F) direkabentuk. Pembaikan ini melibatkan integrasi sistem koordinat silinder (CCS) dan Kotak-P dinamik. BA yang dipertingkatkan dikenali sebagai algoritma Ramlan Ashwak Faudziah (RAF). Fasa pelaksanaan melibatkan pengembangan kunci, penyulitan data, dan penyahsulitan data. Fasa penilaian meliputi mengukur algoritma dari segi memori dan keselamatan. Dari segi memori, keputusan menunjukkan RAF menggunakan 256 bait, iaitu kurang daripada BA (4096 bait). Dari segi kerambangtarikan pada teks dan fail imej yang mempunyai rentetan besar dan mempunyai bait yang serupa, kadar purata kerambangtarikan untuk 188 ujian statistik memperolehi nilai lebih daripada 96%. Ini bermakna RAF mempunyai kerambangtarikan tinggi yang menunjukkan bahawa ianya lebih terjamin. Dengan demikian, keputusan ini menunjukkan bahawa algoritma RAF yang mengintegrasikan CCS dan dinamik Kotak-P adalah satu pendekatan berkesan yang dapat mengurangkan ingatan dan mengukuhkan keselamatan.

Kata kunci: Sistem Koordinat Silinder, Kotak-S dinamik, Kotak-P dinamik, Algoritma Blowfish

Abstract

The Blowfish Algorithm (BA) is a symmetric block cipher that uses Feistel network to iterate simple encryption and decryption functions. BA key varies from 32 to 448 bits to ensure a high level of security. However, the substitution box (S-Box) in BA occupies a high percentage of memory and has problems in security, specifically in randomness of output with text and image files that have large strings of identical bytes. Thus, the objective of this research is to enhance the BA to overcome these problems. The research involved three phases, algorithm design, implementation, and evaluation. In the design phase, a dynamic 3D S-Box, a dynamic permutation box (P-Box), and a Feistal Function (F-Function) were improved. The improvement involved integrating Cylindrical Coordinate System (CCS) and dynamic P-Box. The enhanced BA is known as Ramlan Ashwak Faudziah (RAF) algorithm. The implementation phase involved performing key expansion, data encryption, and data decryption. The evaluation phase involved measuring the algorithm in terms of memory and security. In terms of memory, the results showed that the RAF occupied 256 bytes, which is less than the BA (4096 bytes). In terms of randomness of text and image files that have large strings of identical bytes, the average rate of randomness for 188 statistical tests obtained values of more than 96%. This means that the RAF has high randomness indicating that it is more secured. Thus, the results showed that the RAF algorithm that integrates the CCS and dynamic P-Box serves as an effective approach that can consume less memory and strengthen security.

Keywords: Cylindrical Coordinate System, Dynamic 3D S-Box, Dynamic P-box, Blowfish Algorithm.

Acknowledgement

First of all I have to express my thanks and gratitude to Allah who gives me the ability to achieve this imperfect work and without his blessing and support nothing can be done.

I would like to thank my supervisors, advisors Prof. Dr. Ramlan Mahmod and Associate Prof. Dr. Faudziah Ahmad for their continuous support during my Ph.D. program.

Special thanks goes to my colleague Mohammed Sahib Mechee for his continuous support and encouragement along the study and research time.

I would also like to offer my deepest gratitude to my family for helping me, and for encouraging me to do my PHD work which enabled me to successfully accomplish my tasks.

I am deeply indebted to my dear friends and staff in UUM especially from the School of Computing for taking so much time and interest in my work, always being there for me through my difficult situations and spending their time in guiding me despite their busy schedule.

Last but not least, thanks to all those who have been directly and indirectly involved in helping me complete this research.

Finally, I apology to those whose names might be missed. But, I am grateful to all of them for their help and on various matters of day-to-day life.

Table of Contents

List of Tables

List of Figures

List of Appendices

List of Publications

ALabiachi, A.., Ahmad, F., & Ku, R.K. (2011). A Conceptual Design of Novel Modern Random Key-Stream Generator for High Immunity Correlation Attack. *2011* UKSim 13th International Conference on Computer Modelling and Simulation, 399–402. doi:10.1109/UKSIM.2011.82.

ALabiachi, A., Ahmad, F., & Ku, R.K. (2011). A Competitive Study of Cryptography Techniques over Block Cipher. *2011* UKSim 13th International Conference on Computer Modelling and Simulation, 415–419. doi:10.1109/UKSIM.2011.85.

ALabaichi, A., Mahmod, R., & Ahmad, F. (2013). Randomness Analysis on Blowfish Block Cipher. AWERProcedia Information Technology & Computer Science: 3rd World Conference on Innovation and Computer Science (pp. 1115-1127), Antalya, Turkey.

ALabaichi, A.., Mahmod, R., Ahmad, F., & Mechee, M. (2013).Randomness Analysis on Blowfish Block Cipher using ECB and CBC Modes. Journal of Applied sciences, *13*(5), 758-789.

ALabaichi, A., Mahmod, R., & Ahmad, F. (2013). Analysis of Some Security Criteria for S-Boxes in Blowfish Algorithm. International Journal of Digital Content Technology and its Applications (JDCTA), *7*(12), 8–20.

ALabaichi, A., Mahmod, R., & Ahmad, F. (2013). Security Analysis of Blowfish Algorithm. Proceding of SDIWC: The Second International Conference on Informatics & Applications on IEEE (pp.12–18).

ALabaichi, A., Mahmod, R., & Ahmad, F. (2013). Randomness Analysis of 128 bits Blowfish Block Cipher on ECB mode. (IJCSIS) International Journal of Computer Science and Information Security, 11 (10), 8-21.

ALabaichi, A., Mahmod, R., & Ahmad, F. (2014). A Cylindrical Coordinate System with Dynamic Permutation Table for Blowfish Algorithm. International Journal of Soft Computing 5(9).

ALabaichi, A., Mahmod, R., & Ahmad, F. (2013). Randomness Analysis of 128 bits Blowfish Block Cipher on ECB and CBC Modes, International Journal of Digital Content Technology and its Applications (JDCTA), 7(15), 77-89.

ALabaichi, A., Mahmod, R, Ahmad, F., (2014). A dynamic 3D S–Box based on Cylindrical Coordinate System for Blowfish Algorithm. The 3rd International Conference on Computer Science & Computational Mathematics (ICCSCM 2014). Langkawi, Malaysia, ISBN: 978-967-11414-6-5.

List of Abbreviations

RSA	Rivest – Shamir - Adleman
AES	Advance Encryption Standard
BA	Blowfish Algorithm
DES	Data Encryption Standard
3DES	Triple Data Encryption Standard
IDEA	International Data Encryption Algorithm
RC5	Rivest Cipher 5
RC4	Rivest Cipher 4
S-Box	Substitution box
P-Box	Permutation box
CCS	Cylindrical Coordinate System
CCSDPB	Cylindrical Coordinate System and Dynamic Permutation Box
RAF	Ramlan – Ashwak - Faudziah
3D	Three Dimensional
2D	Two Dimensional
XOR	Exclusive OR
SPN	Substitution - Permutation Network
NIST	National Institute of Standard and Technology
ECB	Electronic Codebook Mode
CBC	Cipher Block Chaining Mode
CFB	Cipher Feedback Mode
OFB	Output Feedback Mode
CTR	Counter Mode
DSDP	Key-Dependent S-Box and Key-Dependent P-Boxes
VMS-AES	Variable Mapping Substitution - Advance Encryption Standard
SK	Secret Key
LFSR	Linear Feedback Shift Register
PN	Pseudo Number
SKs	Secret Keys
Eks	Encryption keys
P-value	Probability value
AVAL	Avalanche Criterion
SAC	Strict Avalanche Criterion
BIC	Bit Independence Criterion
KP	Known Plaintext
LC	Linear Cryptanalysis
BR	Byte Relocation
BT	Byte Transformation
PRT	Partial Round Test
FRT	Full Round Test
BBS	Blum-Blum-Shub

CHAPTER ONE

INTRUDUCTION

1.1 Background

The advancements in technologies have changed the way people communicate with each other. Technologies have accelerated communications, resulting in an exponential information exchange, especially in digital landscape. Hence, it allows people, regardless of the places they are at and the time zone they are in to communicate and transfer information extensively in a borderless manner. In this kind of situation, the protection of transmitted data is very important. This is because in such landscape, the possibility of data theft is high, and eventually results in data loss. More importantly, the attacked data could be manipulated by the attackers for undesirable purposes (Verma, Agarwal, Dafouti, & Tyagi, 2011).

In order to ensure that transmitted data are safe, cryptography has been popularly used Rolf (2005). Cryptography techniques encrypt and hide information. This means that the original information will not been tampered and the information can only be accessed in pieces and not as a whole (Menezes, Van Oorschot, & Vanstone, 1997).

Existing popular cryptographic algorithms on block cipher include DES, RC2, IDEA, CAST, Rijndael, Twofish, RC6, MARS, Serpent, and Blowfish. The limitations of these algorithms except Blowfish are not highly secured and slow.

As mentioned, one of the popular cryptographic algorithms is the Blowfish Algorithm (BA). BA is a symmetric-key block cipher, designed in 1993 by Bruce Schneier and included in a large number of cipher suites and encryption products. BA provides a good encryption rate in software and it is more suitable and efficient for hardware implementation; no license is required as it is unpatented. The key of the BA is a variable from 32 bits to 448 bits, so it requires 2^{448} combinations to examine all the keys, thus ensuring safety of the data (Milad et al., 2012; Sindhuja, Logeshwari, & Sikamani, 2010; Li n & Lin, 2000). On top of that, a variable-length key would make the process of cryptanalysis more difficult for possible attackers (Milad et al., 2012). Other advantages of BA include no effective cryptanalysis of it has been found to date and an effective cryptanalysis on the full-round version of BA too has not been discovered (Cody, Madigan, Donald, and Hsu, 2007; Meyers and Desoky, 2008; Sindhuja et al., 2010, Kumar and Baskaran, 2010, and Kiran et al., 2013).

However, the algorithm has some limitations. According to Vaudenay (1996) a known-plaintext attack requires 2^{8r+1} known plaintexts to break, where $\boldsymbol{r}$ is the number of rounds. He also found a class of weak keys that can be detected and broken by the same attack with only 2^{4r+1} of known plaintexts. However, this attack cannot be used against the regular Blowfish, because it assumes knowledge of key-dependent S-Boxes.

Other drawbacks are it computes all subkeys and S-Boxes dynamically before the beginning of an encryption; thus causing overhead in the computation. This results in roughly the equivalent of encrypting an additional four kilobytes of data per data file which could be an issue for small devices with limited memory. Hence, BA is more applicable for applications that   rarely change the key value and for applications that encrypt or decrypt large streams of data (Cornwell, n.d.; Wang, Graham, Ajam, &

Jiang, 2011; Zhang & Chen, 2008; Chandrasekaran, Subramanyan & Raman, 2011; Milad et al., 2012).

In short, although the BA is considered as better than other algorithms in terms of securing sensitive information, it still faces limitations. However, the limitation can be overcomes by enhancing the algorithm. In this research the BA will be enhanced to address the limited memory issue and security.

1.2 Problem Statement

Among the existing cryptography algorithms, the BA is one of the earliest algorithms that have been accepted as a strong encryption algorithm (Schneier, 1996; Mahdi, 2009; Hashim et al., 2009; Milad et al., 2012). This is due to several reasons: it is faster in data encryption, suitable for hardware implementation, and has a variable key length that inhibits successful attacks. However, the algorithm has some drawbacks. It utilizes a large computational space from the result of using four S-Boxes. Four S-Boxes occupies a large memory (Schneier, 1994; Cornwell, n.d.; Zhang & Chen, 2008; Wang et al., 2011; Chandrasekara et al., 2011; Milad et al., 2012), thus, is inefficient in terms of memory and cost. Another problem is its incompatibility with image and text files that involve large strings of identical bytes. Specifically, the problem is related to the randomness of encrypted text file outputs and encrypted image file outputs (uncompressed and lossless compressed images). BA was also found to have less resistance against differential and linear attacks when the key dependent of S-Box is known.

There have been various previous attempts to enhance BA in different directions such as performance in terms of complexity time and memory (Krishnamurthy, Ramaswamy & Leela, 2007; Chandrasekaran et al., 2011; Vaidhyanathan et al., 2010; Mahdi, 2009; Hashim et al., 2009; Manikandan, Kamarasan, Rajendiran & Manikandan, 2011b; Manikandan et al., 2012a) and enhance security (Schmidt, 2006; Al-Neaimi & Hassan, 2011a; Manikandan, Sairam, & Kamarasan, 2012b; Halagali, 2013; Mahdi, 2009; Hashim et al., 2009; Manikandan, Kamarasan, Rajendiran, & Manikandan, 2011b; Manikandan et al., 2012a).

In cryptography, confusion and diffusion are the guiding principles in a cipher design (Shannon, 1949). Substitution Box (S-Box) is examples of "confusion" and permutation box (P-Box) are examples of "diffusion". An S-Box is a basic component of symmetric key algorithms, which performs substitution. S-Box plays an important role in block cipher algorithms and several weaknesses in S-Box can therefore lead to a cryptosystem which can be easily broken. Much security of block cipher based on feistel network depends on the properties of S-Box that are used in round function and improvement to an S-Box will in turn enhance to the block cipher system.

Several previous works have focused on improving S-Box. Among these are using single S-Box (Hashim et al., 2009), two S-Boxes (Mahdi, 2009), Chaos theory (Chandrasekaran et al., 2011), and replacing the pi initialization (in the algorithm) with a random number generator (Halagali, 2013).

Hashim et al., (2009) were not able to improve the situation as the single S-Box still requires a large memory (65543 bytes) to improve Blowfish of 128 bits. In addition, randomness and the security of the new S-Box were not conducted. Thus, the methodology cannot be confirmed. Mahdi (2009) suggested reducing four S-Boxes to two S-Boxes with 259 bytes. The total size of the two S-Boxes is 518 bytes. Similarity with Hashim et al. (2009), this study did not test for randomness and security of S-Box. Later, Chandrasekaran et al. (2011) managed to decrease the time complexity over the original algorithm in the key generation of subkeys and S-Boxes. But they were not able to reduce the memory requirement. Their work required a memory space of 2 to the power 32 bytes. Halagali (2013) the proposed modification only change initial values of   subkeys and S-Boxes. But did not address the problem of memory requirements and still used four S-Boxes with large memory.

Another attempt to improve block cipher was the use of 3D block cipher such as cubic (Nakahara, 2008; Suri and Deora, 2011; Ariffin, 2012) and these attempts together with byte permutation gave good diffusion results without compromising on the security of the original algorithm.

On the other hand, the P-Box unlike S-Box is meant to reorder or permute the elements. The fixed P-Box is open to differential and linear attacks. However, the dynamic P-Box in which the content is based on secret keys did not have a fixed structure. As a result, dynamic P-Box is more secure than fixed P-Box. Thus, in this research adapts dynamic P-Box to diffuse the output of S-Box.

Thus, using the idea of 3D S-Box with byte permutation, our research attempts to reduce memory requirement in BA. Specifically, the work used a single 3D S-Box with 256 bytes. Byte permutation requires multi secret keys to design a dynamic 3D S-Box. These keys can be generated by different approach such as taking the last byte from the keys generated from each round, performing XOR between all bytes of keys (Juremi et al., 2012; (Krishnamurthy and Ramaswamy, 2009), taking the remainder using the formula MOD 4 (Stoianov, 2011), and random selection (Suri and Deora, 2011; Mahmoud, Hafez, Elgarf, and Zekry, 2013). Among these, the random selection is better. The method is able to generate unpredictable keys during an encryption process. As results, resistance indirectly, can be increased.

In order to render any 3D space, a Cartesian coordinate system (rectangular coordinates), can be used to represent graphs and indicate the points on a 3D space (*xyz*-space). When Cartesian coordinates are not appropriate for a problem under consideration, another coordinate system is needed. Among these are cylindrical coordinate system, spherical coordinate system, and bipolar coordinate system, However, the cylindrical coordinate system is suitable for this research work because a dynamic S-Box is required to perform byte permutation and byte permutation depends on the rotation around x axis. In addition the Cubic is not appropriate to present S-Box (256 bytes) in 3D array.

In summary, based on the problems related to memory and security on BA, this research presents an enhanced BA cryptography algorithm. The algorithm is named as the RAF. It incorporates the Cylindrical Coordinate System (CCS) in the round function with Dynamic Permutation Box to solve the limitation of BA. It utilizes less memory space, and is more secure in terms of randomness and has a high resistance against attacks.

1.3 Research Questions

The study intends to answer the following research questions (RQ):

RQ1-How can the memory requirement of BA be reduced?

RQ2-Can the use of dynamic 3D S-Box and CCS rectify the memory requirement problem?

RQ3-Can the use of multi secret keys enhance the security of BA?

RQ4-Can the use of dynamic P-Box enhances the security of BA?

RQ5-How can the security of BA be improved?

RQ6-Is the enhanced algorithm computationally efficient?

1.4 Research Objectives

The main objectives of this study are to enhance BA in terms of reducing memory requirements, and increasing security.

Specifically, this study aims to achieve the following sub-objectives:

1. To design a dynamic 3D S-Box based on CCS that aims to reduce memory requirements

2. To generate multi secret keys during the encryption process by using the random function

3. To design a dynamic P-Box so that the security can be enhanced

4. To design a new F-Function.

5. To determine the computational efficiency for the enhanced BA.

1.4 Scope of Research

This research study focuses on enhancing BA in terms memory and security. The range of data used includes Cipher Block Chaining Mode, and Random Plaintext/Random 128-bit keys. Image files used are in different formats (gif, jpeg png, tiff, and bmp), Data in ".txt" format was used for the text document and “.flv" and ".wmv" for the video files.

1.5 Contribution of the Study

The main contribution of this research study is the enhanced BA. This enhanced algorithm occupies smaller memory space and can be applied on any type of data even has large string of bytes such as text, image files, and it has good resistance against the attacks.

Below are specific contributions:

- Design a new round function (CCSDPB).

- Incorporate CCS in a 3D S-Box design.

- Design a dynamic 3D S-Box by employ two procedures of byte permutation based on multi secret keys.

- Design a dynamic P-Box to diffuse the output of 3D S-Box.

- Generate multi secret keys during encryption process from random function that its seed based on variables; one of them is sequence of plaintext to enhance randomness of the algorithm as well as increase resistance to the attack.

1.6 Organization of the Thesis

This whole thesis is organised into seven chapters. Chapter One introduces the research study through the discussion of concepts in cryptography, types of cryptography, advantages and disadvantage of each type, as well as advantages and disadvantages of BA. It also formulates the problem statement, research questions that require urgent answers, objectives that need to be fulfilled, scope, and significance of the study.

Chapter 2 presents the literature review in three parts, basic concepts of cryptography, concepts on coordinate systems and transformations, and past works related to the study.

Next, Chapter 3 describes the research methodology. It details the works to be carried out in order to achieve the research objectives.

Chapter 4 introduces a new cryptography algorithm. It is based on the Blowfish design and adapts an F-Function in CCS. This chapter also covers the definitions, notations and conventions, mathematical models, and algorithm specifications.

Chapter 5, which presents the experimental results of the security 3D S-Box in RAF using Avalanche Criterion, Strict Avalanche Criterion, and Bit Independence Criterion, and correlation coefficient s of 3D S-Box with different keys are also presented.

Chapter 6 presents a detailed discussion of the results and analysis of the security test of the proposed cryptography algorithm, output of randomness, correlation coefficient, avalanche text, and cryptanalysis.

Finally, Chapter 7 provides the conclusion of the whole research study and put forward some recommendations for future work.

CHAPTER TWO

LITERATURE REVIEW

This chapter consists of three parts. The first and second parts discuss the basic concept of cryptography, and presents concepts on coordinate systems and transformations. The third part discusses on past works related to the study.

2.1 Basic Concepts of Cryptography

A block cipher is a type of symmetric key cryptography that transforms a fixed length block of plaintext data into a block of ciphertext data of the same length. In this cipher, the transformation occurs when the user provides the secret key. Decryption is performed using reverse transformation on the ciphertext block with the same secret key. The fixed length block is known as the block size, which is typically 64 bits or more in the block cipher system.

In mathematical terms, a block cipher is described as an equation with two parts: an encryption E and a decryption $D = E^{-1}$. If k is the key, m is the plaintext, and c is the corresponding ciphertext, then

$$c = E_k(m) \text{ and } D_k = E_k^{-1}(c) = E_k^{-1}(E_k(m)) \qquad (2.1)$$

Block ciphers use simple operations such as rotation, shift, XOR, and substitutions (S-Boxes). These operations are continuously applied for n iterations, where n is between 4 and 32. The iteration is called a round, and the simple operations applied in every round are called a round function. The length of the plaintext determines whether the plaintext should be expanded to a multiple of the block size. Different modes are used to encipher multiple blocks in a block cipher (Braeken, 2006; Meiser, 2007; Mollin, 2007; Tamimi, 2008).

Basically, the two different designs are the Feistel structure and Substitution-Permutation Network (SPN) (Preneel et al., 2003). The Feistel structure is a class of SPNs with a special structure invented by Horst Feistel. A Feistel structure modifies only half of the data in every round, whereas an SPN structure modifies the whole data in every round. An interesting characteristic of a Feistel structure is that encryption and decryption algorithms are structurally the same, except for the round keys that work in reverse order. The advantage of SPN is its inherent parallelism, whereas its disadvantage is that it requires two different algorithms; the inverse algorithm is required for the decryption algorithm may vary from that required for the encryption algorithm (Keliher, 1997; Zhang & Chen, 2008; Naganathan, Nandakumar, & Dhenakaran, 2011). Examples of a Feistel cipher include DES (National Institute of Standard and Technology or NIST, 1999) and BA (Schneier, 1994), whereas an example of an SPN cipher is AES (Daemen & Rijmen, 2002).

2.1.1 Mode of Operation

This section explains the two most common modes of operations in block cipher system: Electronic Codebook (ECB) and Cipher Block Chaining (CBC). Schneier (1996) posits that system security can be strengthened by several modes of operations, namely, ECB, CBC, Cipher Feedback (CFB), Output Feedback (OFB), and Counter (CTR) modes. This study is concerned with the ECB and CBC modes of operation. In ECB mode, each block is encrypted on its own without any other addition. Meanwhile, CBC mode uses the cipher block from the previous step of encryption to the current one to form a chain-like encryption process (Tamimi, 2008; Ariffin, 2012).

2.1.1.1 Electronic Codebook (ECB) Mode

In ECB mode, data is divided into blocks. Each block of data is encrypted individually. The different blocks are totally independent of each other, indicating that if data is transmitted over a network or a phone line, transmission errors would only affect the erroneous blocks. Among the modes of operation, ECB is the weakest because it has no additional security measures aside from the basic algorithm. However, ECB is the fastest and easiest to implement (Thakur & Kumar, 2011).

ECB mode is defined as follows:

ECB Encryption:

$$C_j = CIPH_k\,(P_j) \tag{2.2}$$

ECB Decryption:

$$P_j = CIPH_k^{-1}(C_j), \qquad for\ j = 1 \ldots n \tag{2.3}$$

In ECB encryption, the forward cipher function is applied directly and independently to each block of the plaintext. The resulting sequence of the output blocks is the ciphertext.

In ECB decryption, the inverse cipher function is applied directly and independently to each block of the ciphertext. The resulting sequence of output blocks is the plaintext. Figure 2.1 illustrates the ECB mode (Dworkin, 2001).

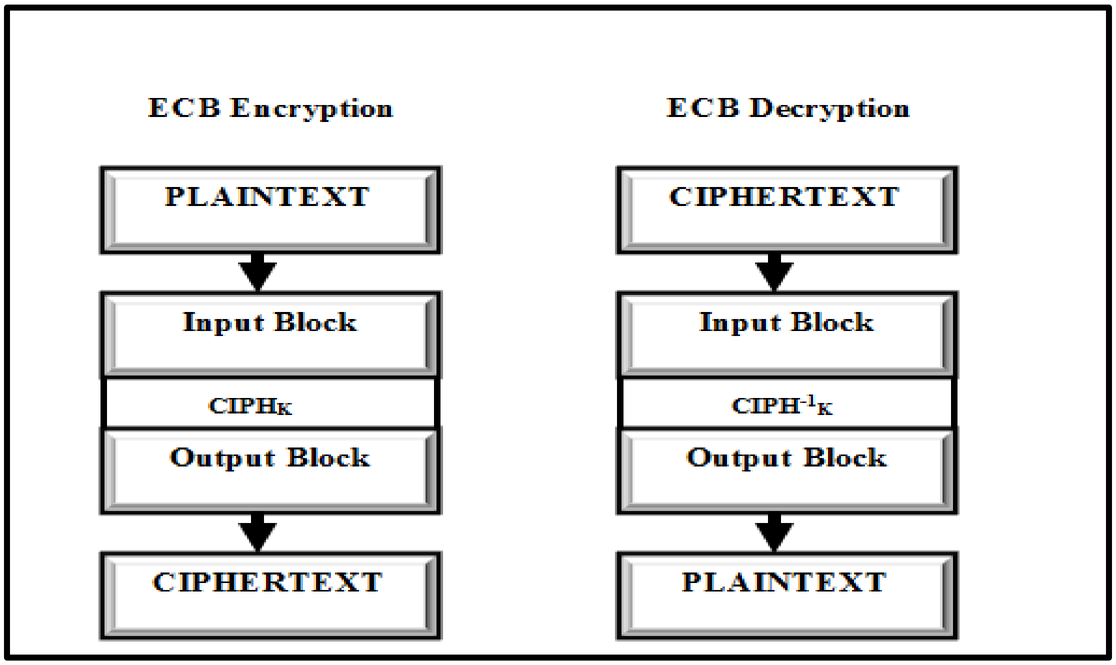


Figure 2.1. ECB mode

2.1.1.2 Cipher Block Chaining (CBC) Mode

In CBC encryption, the first input block is formed by exclusive-ORing (XORing) the first block of the plaintext with the initial vector. Forward cipher function is applied to the first input block, and the resulting output block becomes the first block of the ciphertext. This output block is then XORed with the second plaintext data block to produce the second input block, and forward cipher function is applied to produce the second output block. Afterward, the output block, which is the second ciphertext block, is XORed with the next plaintext block to form the next input block. Each successive plaintext block is XORed with the previous output/ciphertext block to produce a new input block. Forward cipher function is then applied to each input block to produce the ciphertext block. CBC decryption begins with the application of the inverse cipher function to the first ciphertext. The resulting output block is

XORed with the initialized vector to recover the first plaintext block. Inverse cipher function is then applied to the second ciphertext block, and the resulting output block is XORed with the first ciphertext block to recover the second plaintext block. In general, inverse cipher function is applied to the corresponding ciphertext block to recover any plaintext block (except for the first), and the resulting block is XORed with the previous ciphertext block. When data over a network or phone line transmission error (i.e., addition or deletion of bits) occurs, the error would be carried forward to all subsequent blocks because each block depends on the previous block. If the bits are modified in transit, as in most common cases, the error would only affect the bits in the changed block and corresponding bits in the following blocks. The error stops propagating further. Thus, this mode of operation is more secure than ECB because of the extra XOR steps that add another layer to the encryption process. Figure 2.2 illustrates the CBC mode (Dworkin, 2001; Thakur & Kumar, 2011).

The following steps demonstrate the CBC mode:

CBC Encryption:

$$C_1 = CIPH_K (P_1 \oplus IV) \tag{2.4}$$

$$C_j = CIPH_K (P_j \oplus C_{j-1}) \tag{2.5}$$

CBC Decryption:

$$P_1 = CIPH^{-1}_K (C_1) \oplus IV \tag{2.6}$$

$$P_j = CIPH^{-1}_K (C_j) \oplus C_{j-1} \quad for\ j = 2 \ldots n. \tag{2.7}$$

where

- P_j is the jth plaintext block.
- C_j is the jth ciphertext block.

- *CIPH*$_K$ is the forward cipher function of the block cipher algorithm with the key *K* that is applied to the data block.
- *CIPH*$^{-1}_K$ is the inverse cipher function of the block cipher algorithm with the key *K* that is applied to the data block.

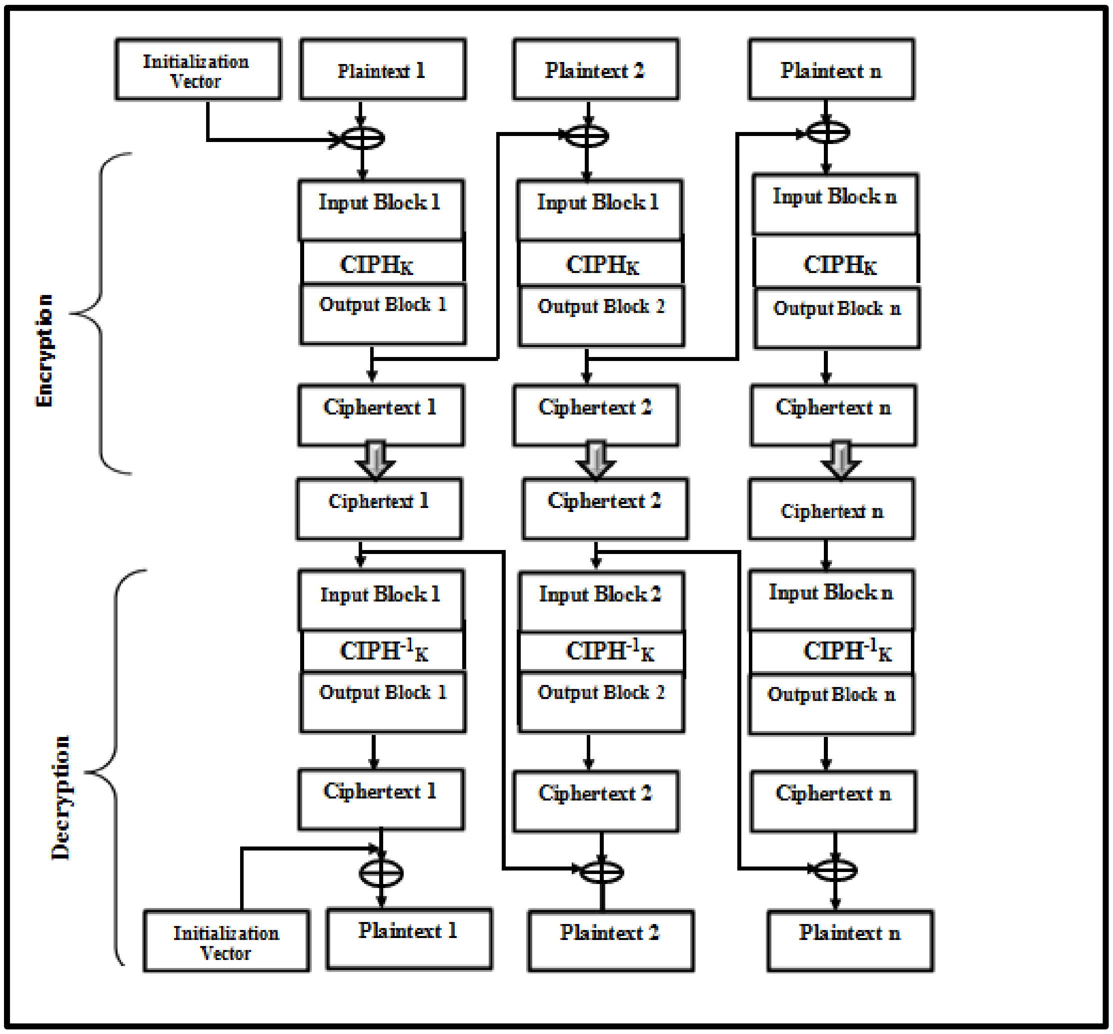


Figure 2.2. CBC mode

2.1.2 Cryptographic Security Requirements

Most of the requirements of contemporary cryptographic security were introduced by Shannon (1916–2001). Shannon (1949) deduced the theoretical principles of confusion and diffusion. He posited that both confusion and diffusion are in a computationally secure cryptosystem. At present, the concepts of confusion and diffusion are still the guiding principles in cipher designs. Confusion is designed to obscure the relationship between the plaintext and the ciphertext, with the objective of frustrating the adversary who uses ciphertext to find the key. S-Boxes are excellent examples of confusion. Diffusion is supposed to spread the statistics of the plaintext through the ciphertext, with the aim of frustrating the adversary who uses the statistics of ciphertext to find the plaintext. If a cipher has a good diffusion property, then flipping one bit of the input changes every bit of the output with an approximate probability of ½. Permutation is a technique used in conducting diffusion (Menezes et al., 1997; Stamp, 2006; Maximov, 2006; Rapeti, 2008; Al-Hazaimeh, 2010; Chandrasekaran et al., 2011).

2.1.2.1 Permutation Box (P-Box)

Classical transposition involves the reordering of elements. A single transposition is the simple exchange in the positions of two elements within a message or vector and is the simplest form of permutation.

Mathematically, a cryptographic permutation process generates a permutation of the input data, that is, the data is simply rearranged. For example, if the block has n different elements, the first element can be positioned in (n) possible places, the second in (n-1), the third in (n-2), and so on, for a total of (n)(n-1)(n-2)…1, or n! possibilities. If a block has 26 different elements, then there exist approximately 26! different ways to permute that block. In the permutation process, every block or message cipher is permuted in exactly the same way. If an opponent acquires multiple messages, a single permutation that makes both ciphertexts readable and identifies the correct permutation may be found (Ritter, 1990).

2.1.2.2 Substitution Box (S-Box)

Classical simple substitution replaces each letter of the alphabet with one fixed substitute. Simple substitution is a very weak cryptographic operation because it cannot obscure the letter-frequency distribution of the plaintext. For a particular language, a statistical analysis of the enciphered data matches the general statistic for that language.

An S-Box can be regarded as a mini substitution cipher. An S-Box is an $m \times n$ substitution unit, where m and n are not necessarily the same. An S-Box may or may not be invertible. In an invertible S-Box, the number of input bits should be the same as the number of output bits.

S-Boxes carry out the confusion concept where non-linear transformation can provide the confusion property. In this form of transformation, the output is not directly proportional to its input. During this process, each input bit is substituted for another output bit (El-Ramly et al., 2001; Maximov, 2006; Ariffin, 2012).

Block ciphers are cascades of diffusion and confusion layers. Confusion layers are usually formalized as the application of S-Boxes defined by lookup tables. In theory, confusion alone is sufficient for security; however, the problem is that a large lookup table, which consumes much memory, is required. In embedded systems, such as smart cards with limited memory resource, conducting confusion on small blocks of 4 to 8 bits is necessary. However, the confusion is considered a weak substitution in this case because of the small block size that permits cryptanalytic cataloguing. Permutation is a form of linear transformation, which makes it weak. Nevertheless, a strong cipher is possible with a combination of substitution and permutation in one round in an appropriate way.

DES is a good example of combination. In the DES algorithm, the expansion permutation and P-Box perform diffusion, whereas the S-Boxes perform confusion. The expansion permutation and P-Box are linear transformations, whereas the S-Boxes are nonlinear transformations. The operations of each of these transformations are simple on their own, but the performance is enhanced once they work together (Ayoub, 1982; Kruppa & Shahy, 1998; Junod & Vaudenay, 2004; Paar, 2005; Al-Hazaimeh, 2010).

A block cipher operates on multiple rounds of similar operations to support the concepts of confusion and diffusion. Examples of such operations include bit-shuffling (P-Boxes), linear mixing (which usually uses the XOR operation), and nonlinear functions (S-Boxes) (Maximov, 2006; Rapeti, 2008; Ariffin, 2012).

A block cipher that mixes confusion with diffusion techniques is preferred to obtain a high level of security in ciphers (Zhang, Sun, & Zhang, 2004; Paar, 2005; Maximov, 2006; Meiser, 2007). For this reason, both the diffusion and confusion concepts are included in this study.

2.1.2.3 Dynamic P-Box and Dynamic S-Box

Dynamic permutation extends to classical transposition, which combines two data sources into complex results; one data source is accumulated into a block, whereas the other is used to rearrange that block (key) (Ritter, 1991).

The fixed P-Box is open to differential and linear attacks. Diffusion alone can usually be broken without much effort (Kruppa & Shahy, 1998; Zhang & Chen, 2008).

In a dynamic P-Box, the content of its base on secret keys does not have a fix map. An example is DSDP (Zhang & Chen, 2008). In dynamic permutation, each block permutes independently. Thus, the dynamic permutation completely voids the previous attack on the fix permutation. The shuffle algorithm is a convenient way of constructing one of the many possible re-arrangements at random (Ritter, 1990; Ritter, 1991).

Dynamic substitution is a type of extended substitution that is similar to simple substitution; however, it has a second data input that rearranges the contents of the S-Box. The S-Box can be changed under the control of a separate data stream, usually originating from a pseudo random sequence generator. This process can be performed after each element is substituted. Random elements exchange of substitution is based on the permutation algorithm (Ritter, 1990). A fixed or static S-Box has no relation with a cipher key, and their contents are not related to the content of the secret key. The role of the secret key is to make changes only on the address of such S-Boxes. Thus, the structure of the key generator is mainly fixed. The only changeable parameter is the secret key. Therefore, a static or fixed S-Box indicates that the same S-Box would be used in every round.

The main problem in implementing any block cipher system is the elements of the fixed structure of S-Boxes. An example of a fixed S-Box is S-Boxes in DES. A fixed S-Box allows attackers to study the properties of the S-Box and locate its weak points. A dynamic S-Box changes every round depending on the key. An S-Box translates each data value into a substituted value; after each substitution, the S-Box is reordered. Compared with fixed S-Boxes, dynamic S-Boxes are more resistant to differential and linear cryptanalysis. Given that the structure of the dynamic S-Box is completely hidden from the cryptanalyst, the attacker faces difficulty in conducting any offline analysis of an attack of a particular set of S-Boxes. A dynamic S-Box is also easier to implement and less susceptible to arguments of "hidden" properties. This S-Box can be created when they are required, thereby decreasing the need for the storage of large data structures within the algorithm. However, the overall performance of S-Boxes in terms of security and speed has not been sufficiently addressed and investigated (Schneier, 1994; El-Ramly et al., 2001; Ali, 2009; Ritter, 1990; Juremi, Mahmod, & Sulaiman, 2012). The most well-known S-Box is BA, which uses the cryptosystem itself to generate an S-Box (Schneier, 1994; Kazlauskas & Kazlauskas, 2009). Elkamchouchi and Makar (2004) stated that ciphers with dynamic S-Box are general and more secure than those with fixed S-Box.

Dynamic S-Box may be considered as a black box with two inputs (i.e., data in and random in) and one output (Ali, 2009; Ritter, 1990). The S-Box starts out as completely unknown. When data is translated through the S-Box, the particular substitution is at least potentially known. Nevertheless, this substitution value is immediately changed, thereby making the S-Box completely unknown again. The S-Box arrangement is complicated and makes cryptanalysis more difficult (Ritter, 1990).

2.2 Basic Concepts of Coordinate Systems and Transformations

This section discusses the concepts of several well-known coordinate systems and the main focus being CCS. The definitions and properties of CCS needed for dynamic 3D S-Box (3D array) construction are also presented in this section.

2.2.1 Coordinate Systems

A coordinate system is meant to determine the unique position of an object or a point in space. “Space,” literally means physical space, but generally refers to “variable-space,” where each dimension corresponds to one variable. For instance, a graph of stock prices has variables of “time” and “value,” such that the graph is in a time-value space. The coordinate system for the time and value of each object or point in any equation has to be clearly specified.

The number of parameters needed to specify a coordinate system is related to the concept of independent and dependent vectors. All points in a 3D object cannot possibly be obtained with only two independent vectors, just as it is impossible to specify data in three variables with only two parameters. However, any three independent variables will span 3D space.

Coordinate systems can broadly be classified into two categories: orthogonal and non-orthogonal coordinate systems. When coordinates are mutually perpendicular, they are said to be orthogonal; otherwise, they are non-orthogonal. Non-orthogonal systems are difficult to handle and have little or no significant use. Thus, orthogonal systems will be the focus of this study. Examples of orthogonal systems include the Cartesian or rectangular, the cylindrical, the spherical, the elliptical cylindrical, the parabolic cylindrical, the conical, the prolate spheroidal, the ellipsoidal, and so on. However, the most common coordinate systems are the Cartesian, the cylindrical, and the spherical.

A Cartesian coordinate system is the most commonly used coordinate system. However, in some applications where rotation is considered, a special form of Cartesian coordinate based on a circle is used. The polar coordinates consist of two parameters: the radial, which is distance between the point and the origin, and $\theta,$ the angle between the point and the positive x-axis.

The cylindrical coordinates (3D) are an extension of the polar coordinate (2D). These coordinates comprise a radial distance r and an angle θ in one plane, similar to polar coordinates, and a distance z perpendicular to this plane. The relationship between cylindrical and Cartesian coordinates is identical to that between polar and Cartesian coordinates, with the addition of $z = z$ (Kalnins, 2009; Collins, 1989; Brannon, 2004; Brougham, n.d.).

The choice of coordinate is often dependent on the physical problem. The Cartesian coordinates are useful for problems with translational invariance, whereas the cylindrical coordinates are useful for problems that are invariant under rotations around a fixed axis. The choice of an inappropriate coordinate system usually results in increased complexity because a hard problem in one coordinate system may be easier in another system, such that a reasonable amount of work and time may be saved by choosing the appropriate coordinate system for a given problem. Hence, when Cartesian coordinates are inappropriate for a problem under consideration, another coordinate system is required. The reason why coordinates are introduced is that a “good choice” of coordinates can substantially simplify a problem. For example, polar coordinates in the plane are very useful in planar problems with rotational symmetry. Certain shapes, including circular ones, cannot even be represented as a function in Cartesian coordinates. These shapes are more easily represented in polar (in plane 2D) and cylindrical (in space 3D) coordinates (Brougham, n.d.; Kalnins, 2009; Lautrup, 2011). The three common coordinate systems are briefly explained in the subsequent section.

2.2.1.1 Cartesian Coordinate System

A Cartesian coordinate system, also called rectangular coordinates, provides a method of rendering graphs and indicating the positions of points on a 2D surface or in 3D space. Each point is uniquely specified in a plane by a pair of numerical coordinates, which are the signed distances from the point to two fixed perpendicular directed lines, measured in the same unit of length. Each line is called a coordinate axis or axis of the system and the point where they meet is its origin, usually at ordered pair (0, 0). The Cartesian plane is also known as the *xy*-plane, whereas a Cartesian three-space, also is called *xyz*-space, has a third axis, oriented at right angles to the *xy*-

plane. This axis, usually called the *z*-axis, passes through the origin of the *xy*-plane (Wrede & Spiegel, 2002; Deakin, 2004; Lambers, 2009). Figures 2.3(a) and 2.3(b) show different Cartesian coordinate systems.

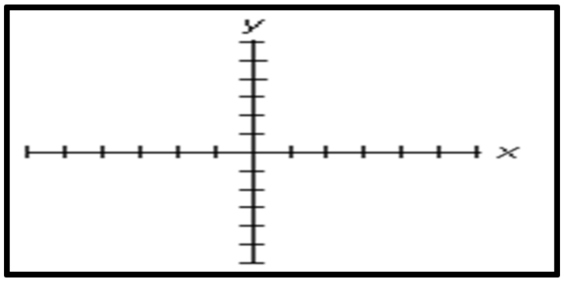


Figure 2.3(a). Cartesian coordinate system for xy-plane

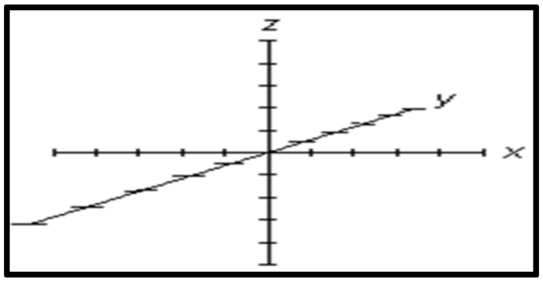


Figure 2.3(b). Cartesian coordinate system for xyz-space

Any point (P) in spherical coordinate systems can be represented as $(r, \theta, \emptyset,)$ (see Figure 2.4), where r is defined as the distance from the origin to point P or the radius of a sphere centered at the origin and passing through P, θ (called the colatitude) is the angle between the z-axis and the position vector of P, and $\emptyset$ is measured from the x-axis (the same azimuthal angle in cylindrical coordinates) (Brougham, n.d.). According to these definitions, the ranges of the variables are as follows:

$$0 \leq r < \infty$$

$$0 \leq \theta \leq \pi$$

$$0 \leq \emptyset < 2\pi$$

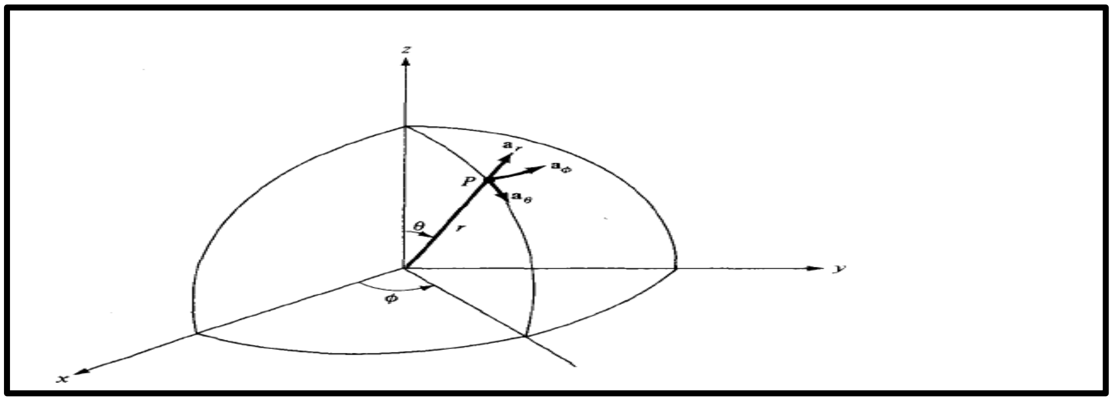


Figure 2.4. Spherical coordinate system

2.2.1.3 Cylinder Coordinate System

To understand the cylinder coordinate system, several definitions are first explained. An overview of the CCS is also further elaborated.

- **Cylinder**

A cylinder is a prism-shaped solid with bases that are closed graphs. A prism with bases that are regular polygons begins to approach being a cylinder when the number of sides is large (Kern & Bland, 1948).

- **Cross Section**

A cross section of a solid is a plane Figure obtained when the solid intersects with a plane.

- **Circular Cylinder**

A circular cylinder is a solid circular cross section in which the centers of the circles all lie on a single line (Brannon, 2004).

- **Right Circular Cylinder**

When the two bases of a cylinder are exactly over each other, and the axis is right angle to the base or when the segment joining the centers of the circles of a cylinder is perpendicular to the planes of the bases, the cylinder is a right circular cylinder; otherwise, the cylinder is said to be oblique (Zwillinger, 2003). Figures 2.5(a) and 2.5(b) show these two types of cylinders.

The unqualified term "cylinder" is also commonly used to refer to a right circular cylinder.

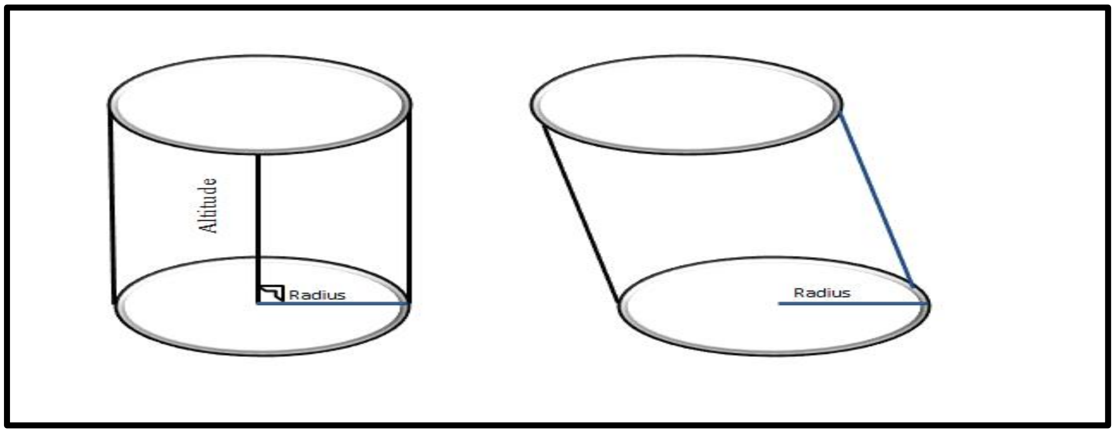


Figure 2.5(a). Right Cylinder *Figure 2.5(b).* Oblique Cylinder

- **Cylindrical Coordinate System**

A cylindrical coordinate system can conveniently deal with problems having cylindrical symmetry. A point (P) in a cylindrical coordinate system is represented as $(\rho, \emptyset, z)$, where that ρ is the radius of the cylinder passing through P or the radial distance from the z-axis, $\emptyset$ is the angle between the x-axis and the projection of the point $(\rho, \emptyset, z)$ onto the xy-plane, and z is the same as that in a Cartesian system (Figure 2.6) (Brougham, n.d.).

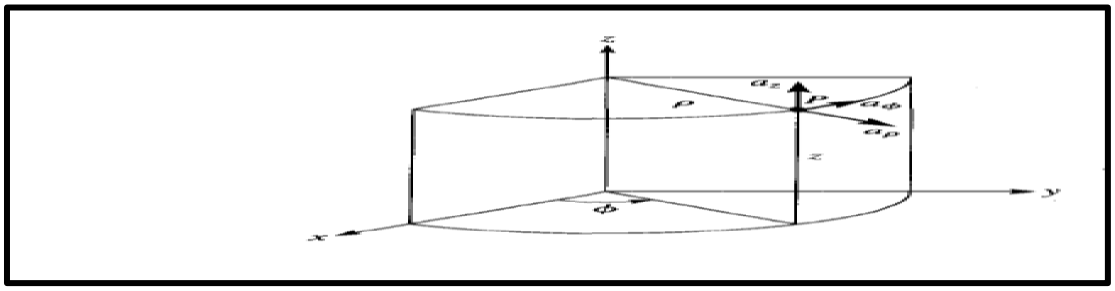


Figure 2.6. Cylindrical Coordinate System

The ranges of the variables are as follows:

$$0 \leq \rho < \infty$$

$$0 \leq \emptyset < 2\pi$$

$$-\infty < z < \infty$$

The level surface of points, such as $z = z_p$, define a plane. A few contours that have constant values of ρ can be drawn. These "level contours" are circles. By contrast, if z were not restricted to $z = z_\rho$, as in Figure 2.7the level surfaces for constant values of ρ would be cylinders coaxial with the z-axis.

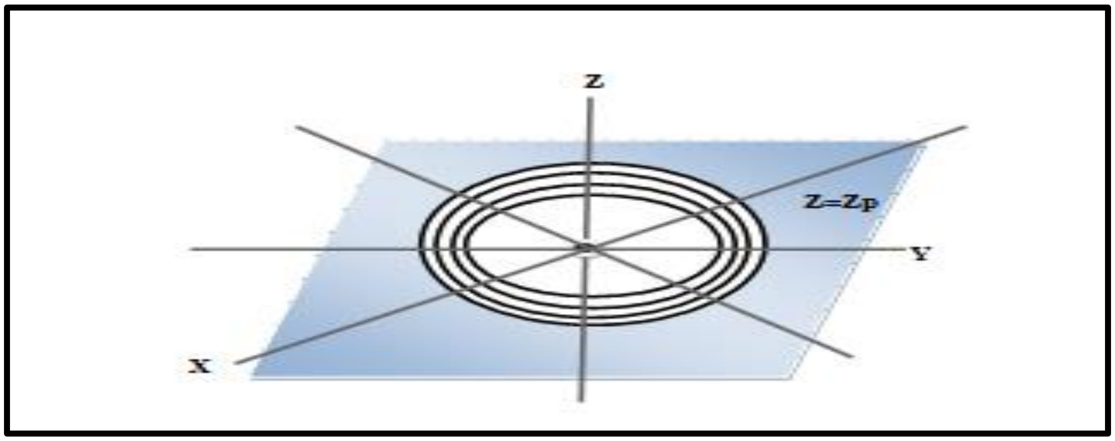


Figure 2.7. Level surfaces for the coordinate ρ

In Figure 2.8, all points that lie on a ray from the origin to infinity passing through P have the same value of $\emptyset$. For any random point, $\emptyset$ can take on values from $0 \leq \emptyset < 2\pi$. In Figure 2.8, "level surfaces" for the angular coordinates are drawn. The coordinates $(\rho, \emptyset)$ in the plane $z = z_\rho$ are called plane polar coordinates (MIT, 2005).

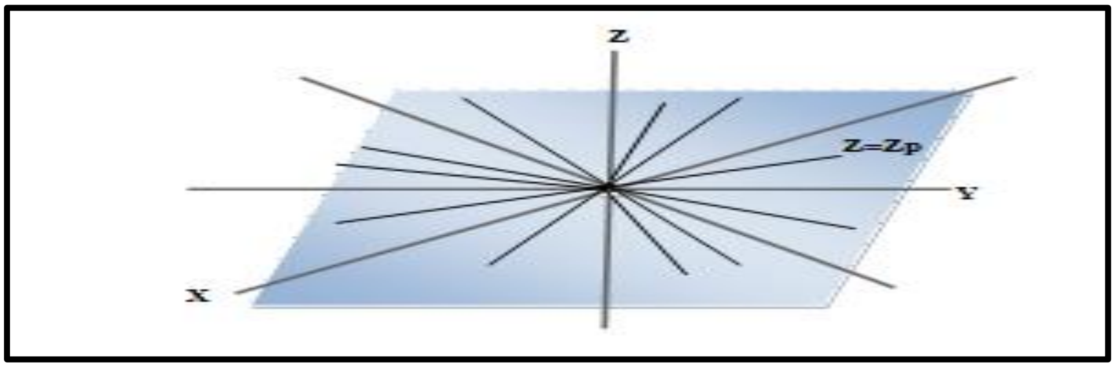


Figure 2.8. Level surfaces for the angle coordinate

2.2.2 Coordinate Transformations

A transformation associates each point (x, y) to a different point or itself in the same coordinate system $(x, y) \rightarrow F(x, y)$.

For example, translating down by a distance d is achieved by $(x, y) \rightarrow (x, y - d)$. Thus, the transformation on a point can be computed immediately.

The equation of the transformed object has the inverse transformation $(x, y) \rightarrow G(x, y)$, which *is* defined by $G\big(F(x, y)\big) = (x, y)$ and $F\big(G(x, y)\big) = (x, y)$. The circle with equation $x^2 + y^2 = 1$ and translating down by a distance d, the inverse transformation is $(x, y) \mapsto (x, y + d)$ (translating up), and the equation of the translated circle is $x^2 + (y + d)^2 = 1$ (Zwillinger, 2003).

2.2.2.1 Transformation of Cartesian coordinate system

To show how a Cartesian coordinate system transforms using a rotation, a running example is given below. Two Cartesian systems(x, y) and (x',y') are related as follows: they have the same origin, and the positive x'-axis is obtained from the positive x-axis by a (counter clockwise) rotation through an angle (Figure 2.9). If a point has coordinates (x, y) in the unprimed system, its coordinates (x', y') in the primed system are the same as those in the unprimed system of a point that undergoes the inverse rotation, that is, a rotation by an angle $\propto= -\theta$.

Accordingly, the transformation is as follows:

$$(x, y) \to \begin{bmatrix} \cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} (x, y) = (xcos\theta + ysin\theta, -xsin\theta + ycos\theta) \tag{2.8}$$

The right-hand side of Equation (3.1) is equivalent to

$$x' = xcos\theta + ysin\theta,$$

$$y' =-xsin\theta + ycos\theta, \tag{2.9}$$

The transformation the primed system to an unprimed system is defined as follows:

$$x = x'cos\ \theta - y'\ sin\theta, \tag{2.10}$$

$$y = x'sin\ \theta + y'\ cos\theta.$$

The x-axis is obtained from the x'-axis by a rotation through an angle θ and y-axis is obtained from the y'-axis.

Similarly, two Cartesian coordinate systems (x, y) and (x', y') differ by a translation x to x' and y to y'. The first system is translated to the second system by the point (x_0, y_0). The coordinates (x, y) and (x', y') are as follows (Zwillinger, 2003; Gill, 2003).

$$x' = x - x_0,\ x = x' + x_0,$$
$$y' = y - y_0,\ y = y' + y_0,$$

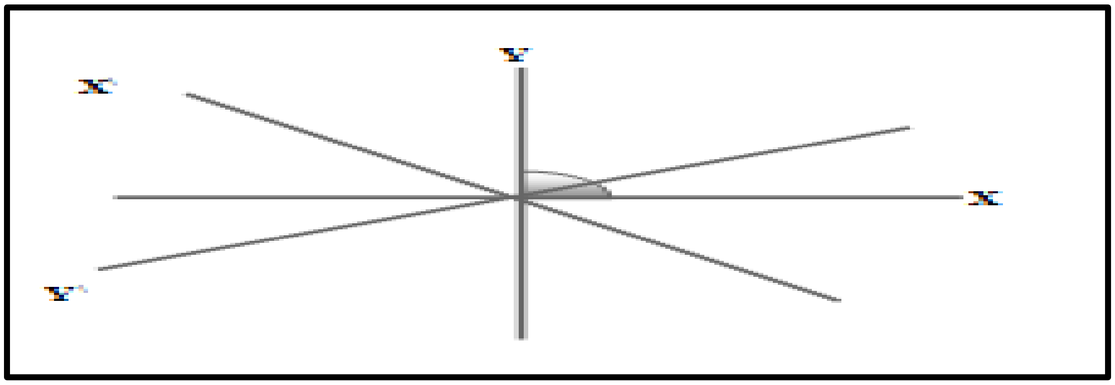


Figure 2.9. Change in coordinates by a rotation

2.2.2.2 Transformation of Polar Coordinates by a Rotation

Cartesian coordinates are versatile, but for some applications including many curves, rotations, and complex numbers, a coordinate system based on the circle is simpler to use.

These systems have polar coordinates (2D), and two parameters *r*, radial distance between the point and the origin, and θ, which is the angle between the point and the positive x-axis. A pole has infinitely many polar coordinates of the form (r, θ), where θ can be any value. A positive θ indicates counter-clockwise rotation, whereas a negative θ indicates clockwise rotation. Thus, the same point can have several polar coordinates, such as $(2, 90)$ and $(2, -270)$ (Kalnins, 2009; Gill, 2003).

2.2.2.3 Transformation of Cylindrical Coordinate System

The relationship between Cartesian and cylindrical coordinate systems is the same as that between Polar and Cartesian coordinate systems. In addition, the transformation used for the polar coordinate system can be used for the cylindrical coordinate system.

2.2.3 Relations among Coordinate Systems

Substitution is another concept that relates a point in one coordinate system to the same point in an entirely different coordinate system. By contrast, transformation relates a point to a different point or itself in the same coordinate system.

Transformation is suitable for problems in the physical space because it can associate a different point to a point in such space.

The relation among the coordinate systems can be better understood through an example. The following example is given to explain various relations (Zwillinger, 2003):

The point in Cartesian (x, y, z), cylindrical (ρ, ϕ, z), and spherical coordinate systems (r, θ, ϕ) are related as follows:

$$Cartesian \leftrightarrow Cylindrical \begin{cases} x = \rho\, cos\, \emptyset \\ y = \rho\, sin\, \emptyset, \\ z = z, \end{cases} \qquad \begin{cases} \rho = \sqrt{x^2 + y^2} \\ \emptyset = \tan^{-1} \frac{y}{x} \\ z = z, \end{cases}$$

$$Cylindrical \leftrightarrow pherical \begin{cases} \rho = r \sin\theta, \\ z = r\cos\theta\ , \\ \emptyset = \emptyset \end{cases} \qquad \begin{cases} r = \sqrt{\rho^2 + z^2} \\ \theta = \tan^{-1} \frac{\rho}{z} \\ \emptyset = \emptyset\ , \end{cases}$$

$$Cartesian \leftrightarrow Spherical \begin{cases} x = r\sin\theta\, cos\,\emptyset \\ y = r\, sin\,\theta\, sin\,\emptyset, \\ z = \text{r}\cos\theta \end{cases} \qquad \begin{cases} r = \sqrt{x^2+y^2+z^2}\ , \\ \emptyset = \tan^{-1}\frac{y}{x}\ , \\ \theta = \cos^{-1}\frac{z}{\sqrt{x^2+y^2+z^2}} \end{cases}$$

The relations among the three coordinate systems are shown in Figure 2.10.

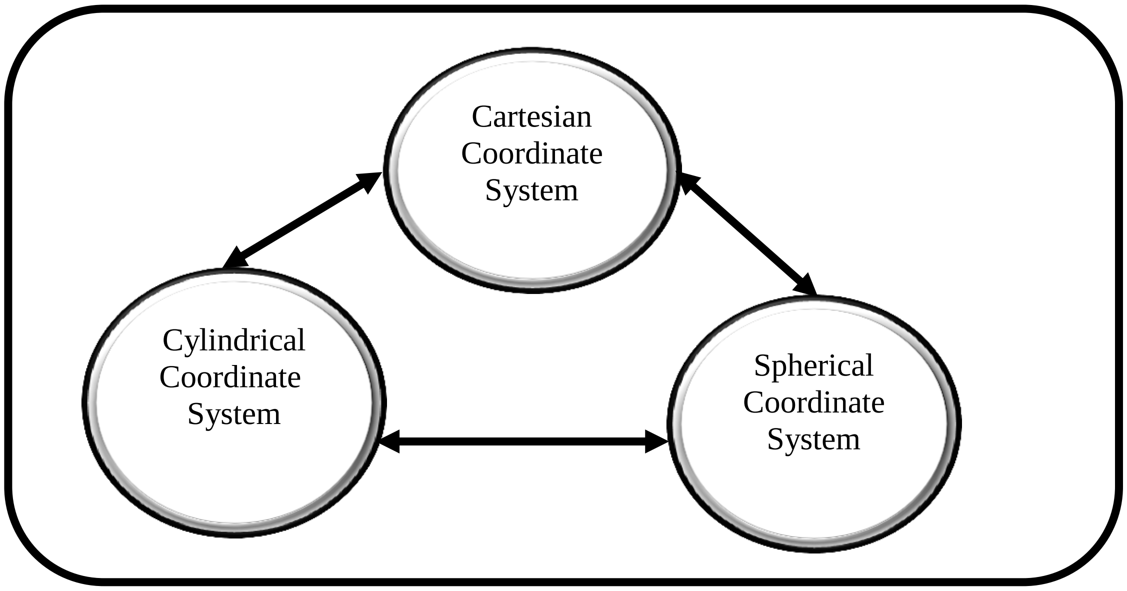

Cartesian Coordinate System
Cylindrical Coordinate System
Spherical Coordinate System

Figure 2.10. Relationship among the three coordinate systems

2.3 Past Related Works

The related works are presented in three parts. The first part includes a brief explanation on stream cipher, and popular pre-eSTREAM cryptography algorithms. The second part presents the pre-AES cryptography algorithms, while, the third part discusses on BA and related works.

2.3.1 Pre-eSTREAM Cryptography Algorithms

A stream cipher is a symmetric key cryptography that encrypts an individual character (i.e., one bit at a time) and consists of two major components: a keystream generator and a mixing function. The keystream generator is dynamic, that is it varies with time, and the mixing function is usually an XOR function. The keystream generator is the main component in a stream cipher. If the keystream generator produces a series of zeroes, the output (cipher) would be identical to the original plaintext. All stream ciphers aim to achieve the properties of a cipher similar to the One-Time Pad cipher.

Figure 2.11 shows the diagram of a stream cipher. The One-Time Pad (or Vernam scheme), which is the simplest and most secure keystream, uses an addition of modulo two keys with the plaintext to produce the ciphertext. The main weakness of this scheme is its key length. Its minimum length is as long as the length of the plaintext, thus making the schema an unpractical one. Nevertheless, many of the recently proposed stream ciphers promote the use of the One-Time Pad that uses a short key to generate a random keystream.

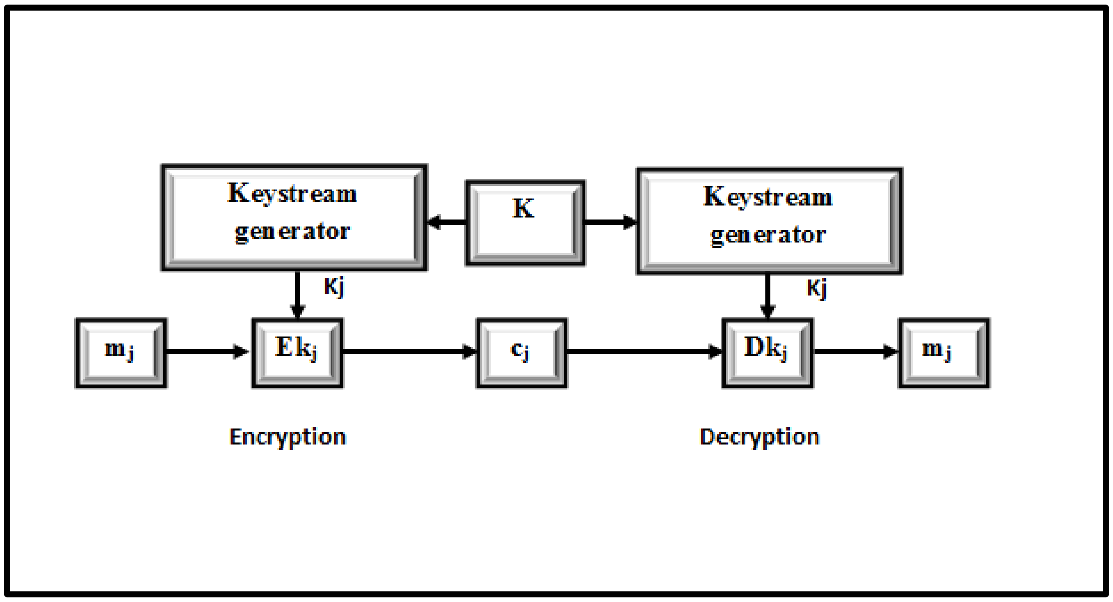


Figure 2.11. Stream Cipher

One of the earlier stream ciphers is RC4. RC4 is a popular stream cipher generating a small size (8 bit) keystream block to XOR with 8 bits of plaintext. RC4 is a software stream cipher and is currently, the standard encryption used in SSL and WEP wireless applications. However, RC4 is still vulnerable to attack due to its random nature and thus, is not recommended for the use in new applications which require higher levels of security (Al-Hazaimeh, 2010).

As a response to the lack of efficient ciphers, two recent European projects that have had influence in this direction are the NESSIE and eSTREAM projects. NESSIE was a project within the Information Society Technologies Programme of the European Commission from 2000–2003. Its main objective was to put forward a portfolio of strong cryptographic primitives, including stream ciphers. However, weaknesses were found in all stream cipher submissions, and therefore no stream cipher made it to the final portfolio. After NESSIE came to an end, the eSTREAM project was initiated by the European Network of Excellence for Cryptology, ECRYPT, to identify new stream ciphers that might be suitable for widespread adoption. ECRYPT is a European research initiative and was launched on February 1st, 2004. The project was completed in four years. One of the projects of ECRYPT is called eSTREAM, and consists of two groups (better known as profiles):

Profile 1: ‘Stream ciphers for software applications with high throughput requirements.’

Profile 2: ‘Stream ciphers for hardware applications with restricted resources such as limited storage, gate count, or power consumption.’

The timeline of the project is divided into three main parts called phases. Phase 1 started immediately after the deadline for submission in April 2005. Phase 1 was aimed on general analysis of all (34) submissions with the goal of determining a subset of interesting candidates to build the Phase 2 cipher list. Phase 1 ended in February 2006 and 5 months later Phase 2 began. The candidates of Phase 2 were scrutinized using criteria like performance, resistance against improved attacks and, of course, deeper cryptanalysis. Phase 3 ended 15 April 2008, with the announcement of the candidates that had been selected for the final eSTREAM portfolio. The selected Profile 1 algorithms were: HC-128, Rabbit, Salsa20/12, and SOSEMANUK. The selected Profile 2 algorithms were: F-FCSR-H v2, Grain v1, Mickey v2, and Trivium (Meiser, 2007; Mattsson, 2006; Hakala, 2007).

Below are summaries of the final eSTREAM portfolio for Profile 1:

- **SOSEMANUK** is a stream cipher that was developed by Berbain, Billet, Canteaut, Courtois, and Gilbert in 2008. It is asynchronous software-oriented stream cipher that corresponds to Profile 1 of the ECRYPT. Its key length varies between 128 and 256 bits. Any key length is claimed to achieve 128-bit security. SOSEMANUK cipher uses some basic design principles from the stream cipher SNOW 2.0 and some transformations derived from the block cipher SERPENT. SOSEMANUK aims at improving SNOW 2.0 both from the security and from the efficiency points of view. Most notably, it uses a faster IV-setup procedure. It also requires a reduced amount of static data, yielding better performance on several architectures (Berbain et al., 2008; Babbage et al., 2008).

Salsa 20 is a family of 256-bit stream ciphers designed by Bernstein in 2008. The 20-round stream cipher, Salsa 20/20 is consistently faster than AES and is recommended by the designer for typical cryptographic applications. It is built on a pseudorandom function based on 32-bit addition, bitwise addition (XOR) and rotation operations, which maps a 256-bit key, a 64-bit nonce, and a 64-bit stream position to a 512-bit output (a version with a 128-bit key also exists). This gives Salsa 20 the unusual advantage that the user can efficiently seek to any position in the output stream in constant time. It offers speeds of around 4–14 cycles per byte in software on modern x86 processors, and reasonable hardware performance. Salsa 20/12 is cipher offers a simple, clean, and scalable design. As well as supporting 128-bit and 256-bit keys in a very natural way, the simplicity and scalability of the algorithm has undoubtedly contributed to it receiving much cryptanalytic attention. Eight and twenty round versions were also considered during the eSTREAM, but Salsa 20/12 offers the best balance, combining a very nice performance profile with what appears to be a comfortable margin for security. The fastest known attacks use approximately 2153 simple operations against Salsa 20/7, approximately 2249 simple operations against Salsa 20/8, and approximately 2255 simple operations against Salsa 20/9, Salsa 20/10, etc. (Bernstein, 2008; Bernstein, 2006; Babbage et al., 2008).

- **Rabbit** was designed by Boesgaard, Vesterager, Pedersen, Christiansen, and Scavenius in 2003. Rabbit was first presented at the Fast Software Encryption workshop in 2003. The Rabbit algorithm can briefly be described as follows. It takes a 128-bit secret key and a 64-bit IV (if desired) as input and generates for each iteration an output block of 128 pseudo-random bits from a combination of the internal state bits. Encryption/decryption is done by XOR'ing the pseudo-random data with the plaintext/ciphertext. The size of the internal state is 513 bits divided between eight 32-bit state, eight 32-bit counters and one counter carry bit. The eight states are updated by eight coupled non-linear functions. The counters ensure a lower bound on the period length for the states. Rabbit was designed to be faster than commonly used ciphers and to justify a key size of 128 bits for encrypting up to 2^{64} bytes of plaintext. This means that for an attacker who does not know the key, it should not be possible to distinguish up to 2^{64} bytes of cipher output from the output of a truly random generator, using less steps than would be required for an exhaustive key search over 2^{128} keys (Boesgaard et al., 2003; Boesgaard, Pedersen, Vesterager, & Zenner, 2004; Babbage et al., 2008).

- **HC-256** is a stream cipher designed by Wu in 2004 to provide bulk encryption in software at high speeds while permitting strong confidence in its security. A 128-bit variant was submitted as an eSTREAM and has been selected as one of the four final contestants in the software profile. HC-256 is considered as reliable software for stream ciphers. The conceptual process of this software aims to generate keystream with 256-bit secret key and the initialization vector of 256-bit. It includes of two secret tables, each one with a fundamental elements of 1024 32-bit. The process establishes during the update of the fundamental elements for each table with non-linear function. With 2048 steps, it leads to update all the tables' elements. From the other hand, each step in the HC-256 produces one 32-bit output based on the utilization of the 32-bit-to-32-bit mapping, which involves two phases initialization process and generation process (Wu, 2004).

2.3.2 Pre-AES Cryptography Algorithms

DES, Triple DES, RC2, Blowfish, IDEA, CAST, Seal, and AES are other examples of symmetric key cryptography. Among these, the DES has been widely deployed by the U.S. Government and banking industry.

Data Encryption Standard (DES) was developed by IBM during 1970's. The DES employs a short 56-bits key length K and 64-bits block size as mentioned previously and is a simple Feistel block network cipher. It was used as the standard by the United States and other parts of the world, before it was cracked in less than 24 hours with little force. For this reason DES is no longer appropriate in this security-conscious generation since it can be easily hacked (Elminaam, Kader, & Hadhoud, 2009).

The same company, IBM, then developed 3DES as an improvement over DES in the late 1970's. The 3DES varies from the typical DES in the sense that it runs three times the succession in the latter. 3DES caters for the problem arising from the shortness in the length of DES (56-bits key length) with its 168-bits. This improves the complexity of the algorithm and in turn makes it difficult for attackers to break. 3DES is still widely used by financial institutions for various financial transactions for security purpose (Kellerman, 2008). 3DES uses 48 rounds of transposition and substitution functions and such makes it highly resistant to differential cryptanalysis, this notwithstanding, the extra effort required of 3DES in terms of long computation time 3DES makes it too slower than DES and unsuitable for real-time applications (Schneier, 1994; Al-Hazaimeh, 2010).

In the late 1980’s, the "Rivest Cipher 2," or RC2 encryption algorithm was developed and named after its developer, Ron Rivest. The algorithm uses a 64-bits block size and variable key length with a source-heavy Feistel network with 16 rounds. As an early cipher it was good for its time and remained a secret for a few years before it became publicly available via the internet. RC2 is vulnerable to attack using 2^{34} chosen plaintexts. RC2 is seen as a fairly easily cracked cipher and not an optimal solution for today's encryption needs (Kellerman, 2008; Elminaam et al., 2009).

IDEA is developed by Lai and Massey in 1991. It utilizes key of a 128-bit and block size is 64-bit. IDEA uses both confusion and diffusion. The design philosophy behind the algorithm is one of “mixing operations from different algebraic groups”. Three algebraic groups are being mixed, and they are all easily implemented in both hardware and software. The key can recovered with a computation complexity of 2^{126} using narrow Bicliques. This attack is computationally faster than full brute force attack (Schneier, 1996; Khovratovich, Leurent, Rechberger, 2012).

In 1993, Bruce Schneier developed the BA, which is a symmetric key block cipher with a 64-bits block size and variable key length. The key in BA varies from 32-bits to 448-bits in length. BA has four dynamic S-Boxes. BA is a highly secured one of the fastest block ciphers developed in the recent time.

An effective cryptanalysis on the full-round version of BA has not yet been discovered as of year 2013 (Cody, Madigan, Donald, & Hsu, 2007; Meyers & Desoky, 2008; Sindhuja et al., 2010; Kumar & Baskaran, 2010; Kiran et al., 2013). According to Vaudenay (1995), a chosen-plaintext attack requires 2^{48} chosen plaintexts to break the key against a number of rounds reduced to eight. However, this attack cannot be used against the full 16-round of BA as it assumes knowledge of key-dependent S-Boxes. BA has several advantages. First, no license is required as BA is unpatented. The BA key is a variable ranging from 32 bits to 448 bits; therefore, BA requires 2^{448} combinations to examine all the keys, ensuring data safety (Lin & Lin, 2000; Sindhuja, Logeshwari, & Sikamani, 2010; Milad et al., 2012). A variable-length key would make the process of cryptanalysis more difficult for possible attackers (Milad et al., 2012). However BA has significant problem. It has four S-Boxes required large memory (4096 bytes).

Based on its high secure property, BA is chosen in this study. However, because of the limitation in terms of consuming a large memory space, the study attempted to decrease the memory consumption by introducing a new design. Past works related to improving memory consumption were found to be unsuccessful.

CAST (Adams and Tavares, 1996) was similar to BA, which consists of key-dependent S-Boxes, a non-invertible F-Function, and a Feistel network. It uses a 64-bit block size and a 64-bit key. The algorithm uses six S-Boxes with an 8-bit input and a 32-bit output. Construction of these S-Boxes is implementation-dependent and complicated based on bent functions, key-dependent rotations, modular addition and subtraction, and XOR operations. There are three alternating types of round function, but they are similar in structure and differ only in the choice of the exact operation (addition, subtraction or XOR) at various points. CAST used six S-boxes which mean that these S-Boxes consume a large memory space (6144 bytes). Wagner, Kelsey, and Schneier in 1997 have discovered a related-key attack on the 64-bit version of

CAST that requires approximately 2^{17} chosen plaintexts, one related query, and 2^{48} offline computations. Other members of the CAST family are CAST-128 and CAST-256 (Schneier, 1996; Kelsey, Schneier, Wagner, 1997).

In 1997, the National Institute of Standard and Technology (NIST) US (an agency of the U.S Department of Commerce's Technology Administration) initiated a process to select a symmetric key encryption algorithm to be used to protect sensitive Federal information and adopted as the Advance Encryption Standard. This new algorithm will replace previous and outdated standard DES. In 1998, NIST announced the acceptance of fifteen candidate algorithms and requested assistance of the cryptographic research community in analyzing the candidates. NIST reviewed the results of this preliminary research and selected MARS, RC6^{TM}, Rijndael, Serpent and Twofish as finalists (Nechvatal, Barker, Bassham, Burr, & Dworkin, 2000; Ali, 2009). Having reviewed further public analysis of the finalists, NIST has decided to propose Rijndael (Daemen et.al., 1999) as the as Advance Encryption Standard (AES) (Nechvatal et al., 2000).

Below is a summary of each of the final candidates:

- Rijndael uses a block size of 128-bits with a variable key length of 128-bit to 256-bit. Rijndael employs a substitution-permutation network that accounts for its fast speed in both software and hardware applications. Rijndael can be used for either classified or non-classified government information today as being practically crack-proof. While the algorithm is considered to be theoretically attackable, it is not a realistic threat with today's level of technology. Brute force attacks against

Rijndael have proven ineffective to date. Side channel attacks, which try to attack the taking care of the cipher implementations rather than the cipher itself, this has proven that an attack of Rijndael is possible but not of a Practical concern unless the attack is taking place on the same server as the encryption is happening on (Pub, 2001; Kellerman, 2008).

- Twofish (Schneier et al., 1998) is a Feistle network with 16 rounds. The Feistel structure is slightly modified using 1-bit rotations. The round function acts on 32-bit words with four key- dependent 8x8 S-Boxes, followed by a fixed maximum distance separable matrix over GF (2^8), a pseudo-Hadamard transform, and the key addition (Carter, Kassin, & Magoc, 2007).

- RC6 (Rivest et al., 1998) is a Feistel network. The round function of RC6 uses variable rotations that are regulated by a quadratic function of the data. Each round includes 32-bit modular multiplication, addition, XOR. And key addition. Key addition is also used for pre- and post-whitening (Rivest, Robshaw, Sidney, & Yin, 1998).

- MARS (Burwick, 1999) has several layers: key addition as pre whitening, 8 rounds of unkeyed forward mixing, eight rounds of keyed forward transformation, 8 rounds of keyed backward transformation, eight rounds of unkeyed backwards mixing, and key subtraction as post-whitening. The 16 keyed transformations are called the cryptographic core the unkeyed round uses two 8×32 bit S-Boxes, and the XOR operation. In addition to those elements, the keyed rounds use 32-bit key multiplication, data-dependent rotations, and key addition. Both the mixing and the core rounds are modified Feistel rounds in which one fourth of the data block is used to alter the other three fourths of the data block. MARS is said to be vulnerable to the some new techniques for the attacks that is developed by (Kelsey et al., 1997).

- Serpent (Anderson et al, 1998) is a substitution-linear transformation network consisting of 32 rounds. Serpent has a block size of 128 bits and supports a key size of 128, 192 or 256 bits. The round function consists of three layers: the key XOR operation, 32 parallel applications of one of the eight specified 4×4 S-Boxes, and a linear transformation. All known attacks are computationally infeasible. However linear cryptanalysis attack breaks 11 round Serpent (all key sizes) with 2^{116} known plaintexts. $2^{107.5}$ time and 2^{104} memory (Nguyen, Wu, & Wang, 2011)

Table 2.1 shows the comparison made on the popular block cipher algorithms.

Table 2.1

Comparison the basic information of the most popular block cipher algorithms

Algorithm	Created by	year	Key Size	Block Size	Algorithm Structure	Rounds	Security Is attacks?	Security (Existing Attacks)	Memory Requirement (S-Box)
DES	IBM	1975	56-bit	64-bit	Feistel Network	16	Yes	Differential and linear cryptanalysis, Davies' attack	8 S-Boxes totally 256 bytes
3 DES	IBM	1978	128-bit or 168-bit	64-bit	Feistel Network	48	No	Theoretically Possible	8 S-Boxes totally 256 bytes
RC2	Rivest	1987	8-128 bit in steps of 8-bit, 64-bit by default	64-bit	Source Heavy Feistel Network	16 Mixing 2 Mashing	Yes	Related key attack	No has S-Box
IDEA	Lai and Massey	1991	128-bit	64-bit	Lai-Massey Scheme	8.5	Yes	Bicliques	No has S-Box
Blowfish	Schneier	1993	32-448 bit	64-bit	Feistel Network	16	No	Differential and linear cryptanalysis	4 S-Boxes totally 24096 bytes
CAST	Adams and Tavares	1996	64-bit	64-bit	Feistel Network	8	Yes	Related key attack	6 S-Boxes 8×32 bit totally 6144 bytes
Rindeal	Daemen and Rijmen	1998	128-bit 192-bit 256-bit	128-bit	Substitution Permutation Network	10 12 14	No	Side channel attack	1 S-Box
Twofish	Schneier	1998	128-bit 192-bit 256-bit	128-bit	Feistel Network	16	No	Truncated differential cryptanalysis	4 S-Boxes totally 1024 bytes
RC6	Rivest et al.	1998	128-bit 192-bit 256-bit	128-bit	Feistel Network	20	No	Linear attack	No has S-box
MARS	Burwick	1999	128-bit 192-bit 256-bit	128-bit	Type 3 Feistel Network	32	Yes	Attacks that is developed by	2 S-Boxes 8×32 bit totally 2048 bytes

								(Kelsey et al., 1997)	
Serpent		1998	128-bit 192-bit 256-bit	128-bit	Substitution Permutation Network	32	Yes	All know attacks are computational infeasible. Linear cryptanalysis breaks 11 rounds (all key sizes) with 2^{116} know plaintexts. And $2^{107.5}$ time and 2^{104} memory	8 S-Boxes 4×4 bit totally 64 bytes

Significant effort has been exerted on analyzing block ciphers. Block ciphers are generally more applicable to a wide range of applications than stream ciphers. The vast majority of network-based conventional cryptographic applications use block ciphers (Meiser, 2007; Rapeti, 2008).

As a result, the breaking of a stream cipher is easier than the breaking of a block cipher with a similar key length (Schmidt, 2006; Nie & Zhang, 2009; Vaidhyanathan, Manikandan, & Krishnan, 2010; Nie, Song, & Zhi, 2010; Manikandan, Manikandan, Rajendiran, Krishnan, & Sundarganesh, 2011a; Pandey, Manoria, & Jain, 2012; Manikandan et al., 2012).

Block ciphers can be used to design stream ciphers with a variety of synchronization and error extension properties, one way hash functions, message authentication codes, and pseudo-random number generators. Because of this flexibility, they are the workhorse of modern cryptography (Schneier et al., 1998).

The large size of the blocks in block cipher provides more security and reduces the success rate of any type of man-in-the middle and differential attacks (Menezes et al., 1997). Recently, the PC1 Security Standards Council has announced the insecurity of WEP in Wi-Fi devices. It seems there are demands for a stream cipher algorithm, which acquires its security by combining it with block cipher algorithms (Lashkari, Danesh, & Samadi, 2009). Therefore this study focuses on block cipher algorithms.

2.3.3 Blowfish Algorithm

The Blowfish Algorithm was invented to replace DES. The cryptographic community needs to provide a new encryption standard because DES, which has been the workhorse encryption algorithm for the past 35 years, is nearing the end of its useful life. Its 56-bit key size is already vulnerable to brute-force attack, and recent advances in differential cryptanalysis and linear cryptanalysis indicate that DES is vulnerable to other attacks as well (Halagali, 2013).

Schneier designed BA in 1994 to replace DES. BA is a symmetric block cipher that uses a Feistel network and iterates simple encryption and decryption 16 times. BA can be divided into key expansion and data encryption (Schneier, 1994; Kumar, Pradeep, Naveen, & Gunasekaran, 2010; Cornwell, n.d.).

The following sections explain the key expansion and data encryption of BA.

Key Expansion

Key expansion starts with the P-array and S-Boxes utilizing many subkeys, which have to be precomputed before data encryption or decryption. The P-array consists of 18 32-bit subkeys (i.e., P_1, P_2… P_{18}).

This section describes how key expansion is conducted. A key with a maximum of 448 bits is converted into several subkey arrays up to a total of 4168 bytes.

Each of the four 32-bit S-Boxes has 256 entries.

$S_{1,0}, S_{1,1},..., S_{1,255}$
$S_{2,0}, S_{2,1},..., S_{2,255}$
$S_{3,0}, S_{3,1},..., S_{3,255}$
$S_{4,0}, S_{4,1},..., S_{4,255}$

The calculation of these subkeys is explained below.

1. The P-array is first initialized, followed by initializing the four S-Boxes with a fixed string that has the hexadecimal digits of P_i.
2. XOR P_1 has the first 32 bits of the key, whereas XOR P_2 has the second 32 bits. This condition is repeated up to P_{14}. The cycle is iterated through the key bits until the entire P-array has been XORed with key bits.
3. BA is then used to encrypt the all-zero strings, employing the described subkeys in steps 1 and 2.
4. P_1 and P_2 are replaced with the output of step 3.
5. The output of step 3 is then encrypted with BA with the use of modified subkeys.
6. P_3 and P_4 are replaced with the output of step 5.

7. The procedure is continued until all the elements of the P-array are replaced, followed by all four S-Boxes being replaced with the output continuously changing.

Schneier (1994), Hashim et al. (2009), and Mahdi (2009) stated that a key expansion procedure preserves the entire entropy of the keys and distributes the entropy uniformly throughout the subkeys. This procedure is designed to distribute the set of subkeys randomly throughout the domain of possible subkeys. In BA, the algorithm itself generates the S-Boxes without additional requirements or algorithms (Kazlauskas & Kazlauskas, 2009). In this study, the same procedure was used to perform key expansion in cryptographic design (RAF).

Data Encryption

Data encryption begins with a 64-bit block element of plaintext transforming into a 64-bit ciphertext. The input is a 64-bit (*X*) divided into two 32-bit halves: *X*L and *X*R. XOR is then implemented between the first 32-bit block segment (*XL*) and the first P-array (*P1*). The obtained 32-bit data is moved to the F-Function that permutes the data to form a 32-bit block segment, which is XOR'ed with the second 32-bit segment (*XR*). Segments *XL* and *XR* are then swapped. This process is repeated for 16 rounds. Segments *XL* and *XR* are then swapped. *XR* is next XORed with *P17*, whereas *XL* is XORed with *P18*. Figure 2.12 illustrates the encryption process in BA with 16 rounds.

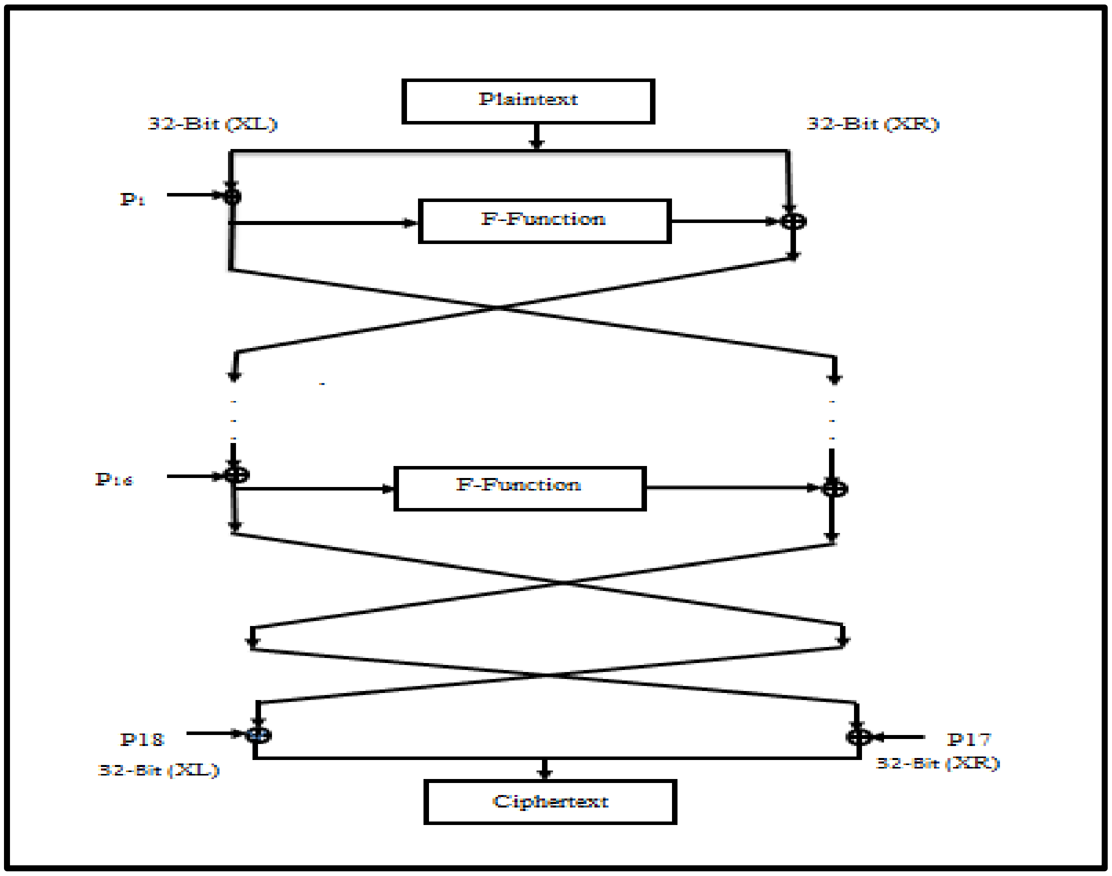


Figure 2.12. Encryption process in BA

The F-Function of BA is the most complex part of the algorithm, which is also the only part that utilizes the S-Boxes. The input of the F-Function is 32-bit, and its output is 32-bit. The input splits into four equal quarters. Every quarter (8-bit) is substituted into a 32-bit in their corresponding S-Boxes. These 32 bits are then combined (XOR or addition modulo 2^{32}). Figure 2.13 describes the architecture of the F-Function (Schneier, 1996; Bagad & Dhotre, 2008; Tilborg & Jajodia, 2005; Cornwell, n.d.).

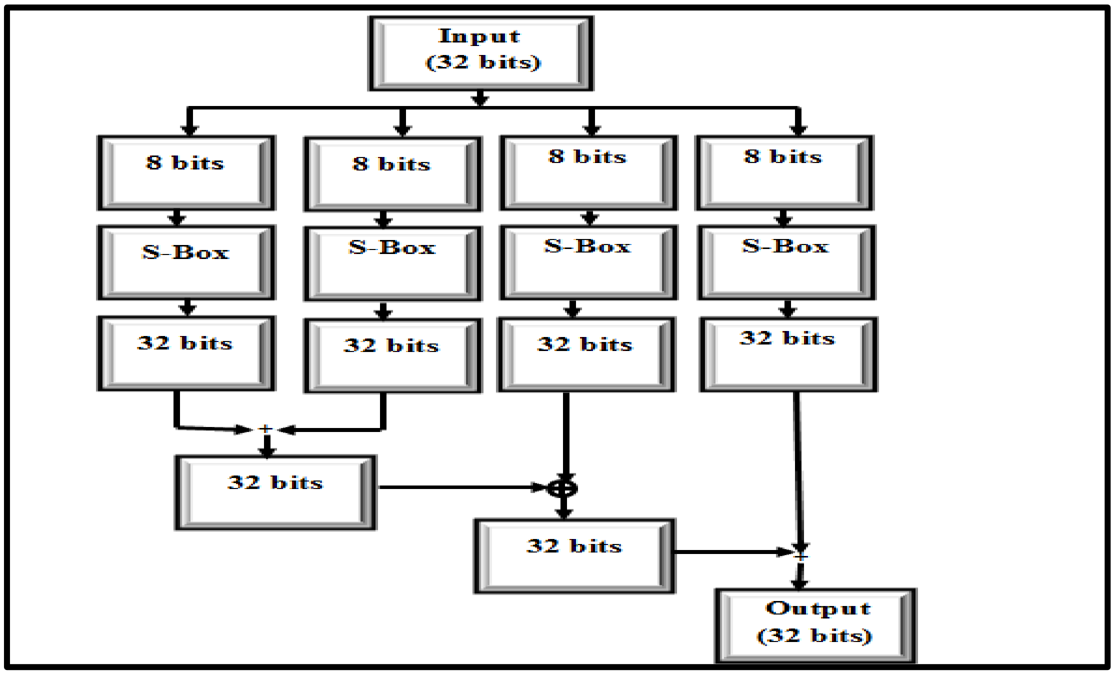


Figure 2.13. F-Function architecture

Data Decryption

Decryption is similar to encryption, but P_1, P_2... P_{18} are used in the reverse order. BA is significantly faster than DES when implemented on 32-bit microprocessors with large data caches, such as Pentium and powered PC (Halagali, 2013). However, BA does not fulfill all the requirements for a new cryptographic standard and is only appropriate for applications where the key is not often changed, such as in a communication link or an automatic file encryptor. BA is inappropriate for smart cards that have small memory. Despite the dynamic structure of the S-Box in BA, the S-Box is not changeable in every round. This condition allows an attacker to try building relations between rounds. For this reason, this study modified the S-Box in BA, such that its dynamic properties are enhanced in terms of security and its memory requirements are decreased.

Previous studies that attempted to modify the S-Box in BA include Hashim et al. (2009), Mahdi (2009), and Chandrasekaran et al. (2011). Hashim et al. (2009) proposed improving BA to encrypt 16 bytes with the use of a variable key length that varies from 8 bytes to 144 bytes. The improved algorithm can decrease the memory requirement by using a single S-Box of 259 bytes (64 bits) and 65,543 bytes (128 bits) instead of four S-Boxes with 4096 bytes (64 bits) and 2097152 bytes (128 bits) without compromising security. However, the results of the randomness test are not presented and S-Box security is not verified. The 65,543 bytes (128-bits) still has a large memory requirement. Mahdi (2009) proposed a 128-bit block cipher (B-R algorithm) that combines BA with the RC6 algorithm to increase security and enhance performance. BR used two S-Boxes, each having a size of 259 bytes, instead of four S-Boxes in BA. However, the results of the randomness test were not presented and S-Box security was not verified.

Chandrasekaran et al. (2011) proposed a new method for the design of S-Boxes based on Chaos Theory to decrease the time complexity of S-Box and P-array generation. The results reveal that the modified design of key generation continued to offer the same level of security as the original BA, but with a less computational overhead in key generation. Despite the decrease in the original algorithm’s time complexity, the memory requirement increases where the modified design requires memory 17179869184 bytes for the tabulation of all key possibilities. The results of the randomness test are also not presented.

Based on these previous studies, the attempts to improve security and decrease memory have not been successful.

Table 2.2 shows the comparison between previous studies on S-Box in BA.

Table 2.2

Comparison made between previous studies on S-Box in BA

Authors	Year	Techniques	Objectives	Block Size	Key length	No of Rounds	Size of memory of S-Box	Weakness
Hashim et al.	2009	-Single S-Box With overlapping -g entries New F-Function	-Reduce memory -Increase security	64bits or 128 bits	32 bits-576 bits or 64 bits-1152 bit	16	-259 in case 64 bit block size. -65,543 bytes in case 128 bits block size	-Large memory in case 128 bits. -S-Box not test -Not test Randomness of whole algorithm
Mahdi et al.	2009	-Two Single S-Box With overlapping -g entries New F-Function	-Reduce memory -Increase security	128 bits	64 bits to 1024 bits	16	518 bytes	- S-Box Not test -Randomness of whole algorithm
Chandrasekaran et al.	2011	Chose theory	Reduction time complexity	64 bit	32bit -448 bits	16	No has S-Boxes	-Large memory (17179869184 bytes). -Not Randomness test

2.3.3.1 Related Works on BA

Many studies related to BA focused on security and performance. The following section presents these studies.

2.3.3.1.1 Security Enhancement

Several methods that were proposed to enhance the BA security include extending the BA architecture (Schmidt, 2006), replacing the old XOR in both sides by a new operation, generating multiple keys by Cellular Automata (Al-Neaimi & Hassan, 2011a, 2011b), using an iterative model to encrypt and decrypt different types of data (Manikandan et al., 2012b), modifying the sub-keys generated by replacing Pi initialization with a linear congruential generator (Halagali, 2013), and combining the good features of BA and the CAST-128 algorithm (Krishnamurthy et al., 2008).

In Schmidt (2006), the performance speed slows down. The number of rounds is 32. It has also been untested for randomness and the S-Box. The memory requirement has been increased from 4096 bytes (4 kB) to 8192 bytes (8 kB).

According to Manikandan et al. (2012b), the performance speed slows down depending on the number of iterations. It has not been tested for randomness and the S-Box, and the memory requirement is the same.

Halagali (2013), Al-Neaimi and Hassan (2011a, 2011b), Krishnamurthy et al. (2008). It has not been tested for randomness and the S-Box, and the memory requirement is the same. From the previous studies, the methods have not been tested and could not verify the methods. Table 2.3 shows comparisons of previous studies on BA.

Table 2.3

Comparisons of Pervious Studies on BA

Authors	Year	Techniques	Objectives	Block Size	Key length	No of Rounds	Size of memory of S-Box	Weakness
Hashim et al.	2009	-Single S-Box With overlapping -g entries New F-Function	-Reduce memory -Increase security	64bits or 128 bits	32 bits-576 bits or 64 bits-1152 bit	16	-259 in case 64 bit block size. -65,543 bytes in case 128 bits block size	-Large memory in case 128 bits. -S-Box not test -Not test Randomness of whole algorithm
Mahdi et al.	2009	-Two Single S-Box With overlapping -g entries New F-Function	-Reduce memory -Increase security	128 bits	64 bits to 1024 bits	16	518 bytes	- S-Box Not test -Randomness of whole algorithm
Chandrasekaran et al.	2011	Chose theory	Reduction time complexity	64 bit	32bit -448 bits	16	No has S-Boxes	-Large memory (17179869184 bytes). -Not Randomness test

2.3.3.1.2 Performance Enhancement

Previous studies related to performance concentrated on two aspects: methods on performance enhancement and comparisons of performance methods.

Several methods were proposed to enhance BA performance. These include modifying the F-Function of the BA by executing two addition operations in parallel using threads (Kishnamurthy, Ramaswamy, and Leela, 2007), decreasing the BA execution time by modifying the order of executing the F-Function (2-XOR gates and 1-ADDERS), implementing this process using multithreading (Vaidhyanathan et al., 2010), and developing a software tool to encipher and decipher the modified BA with file splitting and merging mechanisms (Manikandan et al., 2012a).

All of the research studies discussed above faced the following drawbacks: the algorithm is not appropriate for applications with limited memory such as smart cards, most of the studies did not conduct randomness tests, and most of these studies did not verify S-Box security.

Other studies related to BA include analyzing BA performance with other algorithms. Among the studies are Kofahi et al. (2004), Nadeem and Javed (2005), Mousa (2005), Tamimi (2008), Nie and Zhang (2009), Nie et al. (2010), Elminaam, Kader, and Hadhoud (2010), Thakur and Kumar (2011), and Singh et al. (2011), Verma et al. (2011), Mandal (2012), Milad et al. (2012).

Kofahi et al. (2004), as well as Thakur and Kumar (2011), compared 3 encryption algorithms. Kofahi et al. (2004) compared DES, 3DES, and BA, whereas Thakur and Kumar (2011) compared DES, AES, and BA. Nadeem and Javed (2005), Singh et al. (2011), Verma et al. (2011), Tamimi (2008), and Mandal (2012) compared the performances of 4 encryption algorithms (i.e., DES, 3DES, BA, and AES). Nie and Zhang (2009), Nie et al. (2010) compared on 2 popular encryption algorithms (i.e., DES and BA), whereas Elminaam et al., (2010) compared 6 algorithms (i.e., AES, DES, 3DES, RC2, BA, and RC6).

In all these studies, their results show that BA outperformed other algorithms in terms of speed.

Mousa (2005) evaluated the BA execution time under different types and sizes of data files (i.e., texts, sound, and image), as well as different key lengths in each encryption and decryption process. The results show that changing the key length has no effect on the encryption or decryption time, whereas changing the plaintext file size has a direct effect on the processing time.

Milad et al. (2012) evaluated BA and Skipjack performance. A comparison was made between the two with different input file types and sizes, namely, .txt, .doc, and .jpg. The algorithms were implemented using the C# Programming language. From the results, BA was found to work faster than Skipjack.

From these studies, the BA performance is superior to other algorithms. However, the evaluation of these studies did not include the adaptive analysis of BA security in terms of output randomness, which is one of the most important factors in the evaluation process. For this reason, this research presents four studies (Alabaichi et al., 2013a; Alabaichi et al., 2013b; Alabaich et al., 2013e; Alabaichi et al., 2013f) related to the analysis of BA security in terms of randomness. Table 2.4 illustrates comparison of speeds of the popular algorithms

Table 2.4

Comparison of Speeds of the Popular Algorithms

Name of algorithm	Speed					
	First	Second	Third	Fourth	Fifth	Sixth
BA	✓					
RC6		✓				
AES			✓			
DES				✓		
3DES					✓	
RC2						✓

2.3.4 3D Block Cipher

Several studies are related to the 3D designs of a new block cipher to design a new block cipher that does not compromise security.

Nakahara (2008) and Ariffin (2012) designed a 3D block cipher based on the AES algorithm. Nakahara (2008) improved the 3D block cipher in terms of security by providing good diffusion in three rounds, but was not improved in terms of speed and time. It still required 22 rounds to encrypt one block of plaintext, where it increased the rounds of the AES block cipher by 57%. This condition decreases the speed performance of the block cipher, and has not been tested for randomness.

Based on the limitation of the study of Nakahara (2008), Ariffin (2012) successfully designed a 3D block cipher with byte permutation. It required only 10 rounds and provided good diffusion in three rounds. He also tested for randomness and attacks that produced good results.

From the above studies, the 3D array can be used to generalize a block size of plaintext up to 512 bits with the AES algorithm. The 3D array with byte permutation also provided good diffusion in three rounds without compromising the security of the original algorithm.

In another study, Suri and Deora (2011) successfully designed a 3D block cipher with good diffusion, but had problem in terms of speed. Their method required 64 rounds to encrypt one block of plaintext. The reliability of their method was questionable as they did not conduct a comprehensive test. They conducted only four NIST statistical tests for randomness.

These studies presented a 3D block cipher in Cartesian Coordinates System (X, Y, and Z). Ariffin (2012), and Suri and Deora (2011) conducted byte permutation using a Rotation 3D array. This 3D array can be rotated by $\frac{\pi}{2}$, π, and $\frac{3\pi}{2}$ only to perform byte permutation. The rotation in the CCS can be conducted by $\frac{\pi}{4}$, $\frac{\pi}{2}$, $\frac{3\pi}{4}$, π, $\frac{5\pi}{4}$, $\frac{3\pi}{2}$, and $\frac{7\pi}{4}$ to perform byte permutation in this study. This condition means that CCS is more suitable than Cartesian Coordinates System in this study.

In this study, a 3D array was used to design a 3D S-Box. Byte permutation was used to permute the values of the 3D S-Box. A 3D array was constructed using CCS, whereas byte permutation was constructed using the transformation of CCS.

2.3.5 Dynamic S-Box

Many studies have attempted to modify the AES S-Box to make it dynamic instead of fixed to increase algorithm security. This section reviews some related studies in this field.

Krishnamurthy and Ramaswamy (2009), Mohammad, Rohiem, and Elbayoumy (2009), Stoianov (2011), Hosseinkhani and Javadi (2012), Juremi el al. (2012), and Mahmoud, Hafez, Elgarf, and Zekry (2013) were among the studies designed a Dynamic S-Box in AES.

Krishnamurthy and Ramaswamy (2009) proposed a dynamic S-Box in every round without changing the basic AES operations. They used four cases with different levels of security requirement. The first case used the last byte from the round keys and rotated the S-Box dependent on it. The second case XORed all bytes of the round keys and rotated the S-Box dependent on the resulted value. The third case used another set of round keys generated using a key expansion algorithm similar to that of the AES key expansion algorithm. The last byte of the round keys was then used to rotate the S-Box. The fourth case is similar to the third case, except XORing the values of all the bytes in round keys instead of using the last byte. The Rotate S-Box depended on this value.

Mohammad et al. (2009) proposed AES with Variable Mapping (VMS-AES) S-Box. It uses the key to generate a subkey to randomly shift (remapping) the substitution of the S-Box to another location.

Stoianov (2011) proposed two new S-Boxes (i.e., S-BOXLeft and S-BOXright) using the left and right diagonal as the axis of symmetry. An algorithm using these S-Boxes was based on a pre-selected byte of the secret key that was divided by 4. One of the four S-Boxes (i.e., S-BOX, Inv S-BOX, S-BOXLeft, and S-BOXRight) is selected based on the reminder.

Hosseinkhani and Javadi (2012) introduced a new algorithm that dynamically generates S-Box from cipher key in only two steps. The first step is to generate a primary S-Box using the same procedure used in the previous algorithm (AES). The second step is to swap the values in the rows with the values in the columns of primary S-Box. This routine uses cipher key as input and then dynamically generates S-Box from the cipher key by using shift columns, shift row, shift account.

Juremi el al. (2012) proposed another new key dependent S-Box and uses S-Box rotation to make the S-Box dynamic. The round key was used to identify a value for use in the rotation of S-Box. All the bytes of the round key are XORed. The results are then used to rotate the S-Box. The rotation value depends on the entire round key.

Mahmoud et al. (2013) also proposed a dynamic S-Box based on byte permutation of the standard S-Box under the control of the AES SK. Linear Feedback Shift Register (LFSR) was used to generate random sequences. The AES SK is used to generate an initial state of LFSR by dividing it into two parts and placing an XOR between these parts. The results can be used as the initial value of LFSR. The output of the PN generator is XORed with an SK. The result is converted to a hexadecimal. The repeated values are discarded, and then the missing numbers are added to the sequence to ensure that all S-Box indexes are mapped. These numbers are used to rearrange columns and rows on the standard S-Box.

The above studies show that most attempts are based on the application of different byte permutation mechanisms that are applied by rearranging the location of the elements in S-Box using SKs to achieve a dynamic S-Box, which increases the security of the original algorithm. Dynamic S-Box is protected against differential and LC because the structure of the S-Box becomes completely hidden from the cryptanalyst (Schneier, 1994; El-Ramly et al., 2001; Ali, 2009; Juremi et al., 2012). In this study, dynamic S-Box in BA has been improved by using byte permutation based on SKs. Dynamic S-Box is achieved by rearranging the location of the elements in the 3D S-Box after each two bytes substituted at each round with every block of plaintext. Thus any trails from cryptanalysis to build relations between rounds leads to fail. In addition randomness of the algorithm will be enhanced. Random SKs with byte permutation are proven to be a good mechanism. Therefore the method is adopted in this study.

2.3.6 Secret Key Generation

Secret keys can be generated using different methods. Krishnamurthy and Ramaswamy (2009) generated SKs using four methods. The first method used the last byte from the round keys. The second method used XOR among the values of the all bytes in round keys. The third method used another set of round keys, which are generated using a key expansion algorithm similar to the AES key expansion algorithm, and then takes the last byte of the round keys. The fourth method is similar to the third, except that it performs XOR between the values of the all bytes in round keys. Mohammad et al. (2009) generated SKs for relocation by dividing SK mod 256. The result is added to the index to get new location of the element. Stoianov (2011) generated keys by dividing pre-selected bytes of the SK and dividing them by four. Juremi el al. (2012) generated keys by applying XOR to all bytes of the round keys. Mahmoud et al. (2013) generated SKs using LFSR. Suri and Deora (2010, 2011) used the random number generator of Turbo C to generate random numbers. The random number generator is a pseudo-random generator that can return a pseudo-random integer between zero and the maximum value.

Both methods, LFSR and random number generator, are used to generate different random multi SKs. LFSR requires additional time (overhead time), making random number generator faster and easier to implement than LFSR.

In this study, random numbers between 0 and 3 in five sets are required, making the random number generator more appropriate than LFSR to generate multi random numbers without adding overhead time. These multi random numbers can be used as random SKs for the permutation of the values of 3D S-Box.

2.3.7 Dynamic P-Box

Most previous studies used fixed P-Box, such as initial and final P-Box in DES (NIST, 1999), as well as P-Box in a RAINBOW algorithm proposed by (Zhang et al., 2004).

The fixed P-Box is open to differential and linear attacks. In addition, diffusion alone can usually be broken without significant effort (Kruppa & Shahy, 1998; Zhang & Chen, 2008). By contrast, dynamic P-Box places its base content on SKs and does not have a fixed map, such as in the DSDP structure (Zhang & Chen, 2008).

Various methods were proposed for the design of dynamic P-Box. For example, Zhang and Chen (2008) proposed a 128-bit Feistel block cipher that involved both the dynamic S-Box and dynamic P-Box simultaneously. The internal structure of this cipher algorithm is secured with two key-dependent transformations. In other words, the cipher can resist linear and differential cryptanalysis in a few rounds of encryptions. A fast permutation algorithm (key scheduling of RC4) is used to generate both the dynamic S-Box and dynamic P-Boxes.

Ritter (1990; 1991) used the shuffle algorithms by Durstenfeld for dynamic transposition or permutation. Ritter (1990) used this dynamic transposition as a cryptographic combiner to replace the Vernam XOR combiner in the stream cipher. The shuffle algorithm is used to rearrange the contents and is among the many possible strategies for permutation. Ritter (1991) used this algorithm to permute the plaintext into ciphertext by swapping every element with another element that is selected pseudo-randomly.

The pseudo code of this algorithm as follows:

```
Function p = GRPdurG(p)
%   n is the number of blocks
n = length (p)
For k = n:-1:2
         r = 1+floor (rand*k);     % random integer between 1 and k
         t    = p (k)
        p (k) = p(r)               % Swap(p(r),p(k)).
        p(r) = t
End
```

A close examination of dynamic P-Box design shows that it is based on fast permutation under the control of SKs. Many techniques can be used for fast permutation based on SKs, one of which is the shuffle algorithm by Durstenfeld. An advantage of the shuffle algorithm by Durstenfeld is that it is a simple algorithm with no additional requirements, complexity, and overhead in time. This algorithm can shuffle any array in place and can permute any object. This algorithm is a member of a family of transposition algorithms used to generate combinatorial objects. Thus in this study the Durstenfeld's Shuffle algorithm in design dynamic P-Box is adapted, which swaps every element with another selected element based on random number.

2.3.8 Evaluation of Block Cipher

The most important factor in the evaluation of cryptographic algorithms is the element of security. The evaluation criteria are divided into three major categories, namely, security, cost, and the algorithm implementation and characteristics. The first category, security, comprises such features as randomness of algorithm output, avalanche effect, resistance of the algorithm to cryptanalysis, and relative security over other candidates. The second important category is cost, which encompasses licensing requirements, computational efficiency or speed on various platforms, and memory requirements. The third category is implementation, and its evaluation is based on algorithm characteristics, such as flexibility, hardware and software suitability, and simplicity (Ali, 2005; Ali, 2009; Ariffin, 2012).

Security and cost were evaluated in this study. Security evaluation was based on a randomness test, correlation coefficient, avalanche effect, S-Box properties, and cryptanalysis whereas that on cost was based on memory requirements and computational efficiency.

2.3.8.1 Randomness Test

Randomness is an important criterion for an efficient block cipher. Thus, the block cipher should be statistically analyzed to determine whether the tested algorithm fulfills this requirement; if a block cipher appears to be non-random, then it becomes vulnerable to all types of attack (Isa & Z'aba, 2012). The output should also be random for a block cipher. According to Ali (2005; 2009) and Ariffin (2012), a good algorithm should be efficient in producing single random bits.

Table 2.5

NIST statistical test

Statistical Test	No. of P-values	Test ID
Frequency	1	1
Block Frequency	1	2
Cumulative Sum(Cusums)	2	3-4
Runs	1	5
Longest Run	1	6
Rank	1	7
FFT	1	8
Non Overlapping Template	148	9–156
Overlapping Template	1	157
Universal	1	158
Approximate Entropy	1	159
Random Excursions	8	160–167
Random Excursions Variant	18	168–185
Serial	2	186–187
Linear Complexity	1	188

To test for randomness, the NIST Test Suite was used. The NIST Test Suite is a statistical package comprising tests on 15 different aspects. The tests focus on various types of non-randomness that could exist in a sequence (Rukhin et al., 2010). Table

2.5 shows the different statistical tests and the number of tests to be conducted for each core test (Soto & Bassham, 2000).

Table 2.6 shows the minimum requirements for each test, where *n* is the length of the bit string, M and L are the lengths of each block, Q is the number of blocks, and N is the number of independent blocks (Ariffin, 2012).

Table 2.6

Minimum requirements of NIST statistical test

No.	NIST Statistical Test	Minimum Requirement
1	Frequency	$n \geq 100$
2	Frequency within a Block	$n \geq 100$
3	Runs	$n \geq 100$
4	Longest-Run-of-Ones in a Block	$n \geq 128$, $M = 8$
5	Binary Matrix Rank	$n \geq 38912$
6	Discrete Fourier Transform (Spectral)	$n \geq 1000$
7	Non-overlapping Template Matching	$n \geq 1000000$, $M = 13072$, $M > 0{:}01n$
8	Overlapping Template Matching	$n \geq 1000000$, $n \geq MN$
9	Maurer's Universal Statistical	$6 \leq L \leq 16, Q = 10(2L)$; $n \geq (Q + K)L$, $n \geq 387840$
10	Linear Complexity	$n \geq 1000000$; $500 \leq M \leq 5{,}000$
11	Serial	$M < (\log_2 n) - 2$
12	Approximate Entropy	$M < (\log_2 n) - 2$
13	Cusums	$n \geq 100$
14	Random Excursions	$n \geq 1000000$
15	Random Excursions Variant	$n \geq 1000000$

The following researchers conducted these statistical tests to measure the strength of the randomness of their algorithms: Soto and Bassham (2000), Fahmy, Shaarawy, El-

Hadad, Salama, and Hassanain (2005), Katos (2005), Ali (2005), Alsultanny and Jarrar (2006), Limin, Dengguo, and Yongbin (2008), Doroshenko et al. (2008), Mohammad et al. (2009), Ali (2009), Zhou, Liao, Wong, Hu, and Xiao (2009), Patidar, Sud, and Pareek (2009), Abd-ElGhafar et al. (2009), Sulak, Doganaksoy, Ege, and Koak (2010), Alani (2010), Suri and Deora (2010, 2011), Ariffin (2012), Isa and Z'aba (2012), Sulaiman, Muda, and Juremi (2012a), Sulaiman, Muda, Juremi, Mahmod, and Yasin (2012b), ALabaichi et al.(2013a, 2013b, 2013e, 2013f).

This research adopted these tests (NIST) to evaluate RAF.

2.3.8.1.1 NIST Framework

NIST is based on hypothesis testing. A hypothesis test determines whether an assertion about a particular characteristic of a population is reasonable. In this case, the test involves the assessment of a specific sequence of zeroes and ones to determine if the sequence is random. Table 2.7 illustrates the step-by-step process for the evaluation of a single binary sequence (Soto, 1999a).

Table 2.7
Evaluation Procedure for a Single Binary Sequence

State your null hypothesis	Assume that the binary sequence is random
Compute a sequence test statistic	Testing is conducted at the bit level
Compute the P-value	P-value ϵ [0, 1]
Compare the P-value to α	Fix α, where $\alpha \epsilon$ (0.001, 0.01]. *Success* is declared whenever P-value $\geq$ α; otherwise, *failure* is declared

The significance level α was fixed at 0.01 for each experiment. The maximum number of binary sequences expected to be rejected at the chosen significance level was computed using the formula by Soto (1999b). For example, the rejection rate of a sample with 128 sequences should not be more 4.657.

$$s\left(\alpha + 3\left(\sqrt{\frac{\alpha\,(1-\alpha)}{s}}\right)\right) \tag{2.11}$$

where s is the number of samples and α is the significance level.

The proportion of sequence that passes statistical tests is computed by using the following formula:

$$p\alpha = (1-\alpha) - 3\sqrt{\frac{\alpha(1-\alpha)}{s}} \tag{2.12}$$

The proportion of sequences that passes a specific statistical test in the analysis should be greater than $p\alpha$ (Rukhin et al., 2010).

2.3.8.1.2 Test Package

According to Rukhin et al. (2010), the statistical package has 15 tests, a namely, Frequency Test, Block Frequency Test, Cumulative Sums Forward (Reverse) Test, Runs Test, Long Runs of one's Test, Rank Test, Spectral (Discrete Fourier Transform) Test, Non-periodic Templates Test, Overlapping Template Test, Serial Test, Universal Statistical Test, Approximate Entropy Test, Random Excursion Test, Random Excursion Variant Test, and Linear Complexity Test.

The descriptions of the 15 tests and their corresponding purposes are shown in Appendix A.

2.3.8.2 Correlation Coefficient

Correlation coefficient refers to a number between -1 and 1 that measures the degree of linear relation between two variables. The correlation is 1 in case of an increasing linear relationship and -1 in case of a decreasing linear relationship. In all other cases, the values vary depending on the degree of linear dependence between variables. If the variables are independent, the correlation is 0. Fahmy et al. (2005), Mohammad et al. (2009), Ariffin et al.(2012), Mahmoud et al. (2013), and Alabaichi et al. (2013d) conducted this test. The values below provide a description of the linear relationship between two variables.

- 0 indicates a non-linear relationship.
- +1 indicates a perfect positive linear relationship.
- -1 indicates a perfect negative linear relationship.
- Values between 0 and 0.3 (0 and -0.3) indicate a weak positive (negative) linear relationship.
- Values between 0.3 and 0.7 (-0.3 and -0.7) indicate a moderate positive (negative) linear relationship.
- Values between 0.7 and 1.0 (-0.7 and -1.0) indicate a strong positive (negative) linear relationship.

This research study used this test to evaluate 3D S-Box and RAF.

2.3.8.3 Security of S-Box

Most previous works on S-Box focused on the design or analysis of S-Boxes because S-Box brings nonlinearity to cryptosystems and strengthens cryptographic security. A weaknesses in S-Boxes causing cryptography to fail. Before the cryptographic uses secure S-Box, two important aspects have to be considered. The first is how to design a good S-Box, and the second is how to verify whether S-Box is good. Thus, the quantitative values of the desired properties for an S-Box must be obtained to confirm its secureness (Adams & Tavares, 1990; Mar & Latt, 2008; Hussain, Shah, Afzal, & Mahmood, 2010; Ahmed, n.d.).

Most previous works on S-Box have attempted to design good S-Boxes by generating them randomly and then evaluating them to reject those that fail to meet the criteria (Adams & Tavares, 1990).

Several properties, such as Avalanche Criterion (AVAL), Strict Avalanche Criterion (SAC), and Bit Independence Criterion (BIC), guarantee S-Box randomness. These properties are cryptographically desirable in S-Boxes and are used as a guide in the design of S-Boxes. Among the researchers who used the evaluation criteria were Adams & Tavares (1990), Vergili & Yücel (2000, 2001), Kavut & Yücel (2001), Abd-ElGhafar et al. (2009), and Stoianov (2011).

Vergili and Yücel (2000, 2001) investigated the criteria of AVAL, SIC, and BIC for a randomly chosen S-Box. The results show that these properties can be achieved randomly in chosen S-Boxes with values of relative absolute errors. These properties of large S-Boxes are probably satisfied within a low error range. The correlations among the test criteria are evaluated to determine the extent to which such criteria can measure different cryptographic aspects of S-Boxes. The results indicate that AVAL and BIC are uncorrelated. Although observing the AVAL characteristic of a small-sized S-Box may provide some information on SAC, an advisable method is to test S-Boxes for the AVAL, SAC, and BIC criteria separately.

Kavut and Yücel (2001) investigated the characteristics of Rijndael's S-Box for the criteria of AVAL, SAC, BIC, nonlinearity, and XOR table distribution. The results are compared with those of Safer K-64. Experimental results show that the S-Boxes of Rijndael satisfy these criteria with very small values of relative absolute errors. Moreover, the parameters of Rijndael's S-Box show better results than those of Safer K-64.

Abd-ElGhafar et al. (2009) proposed a novel method for the construction of a cryptographically variable dynamic S-Box (AES-RC4) algorithm. This method was tested using AVAL, BIC. The results prove the security of the proposed S-Box (AES-RC4) because it passed the AVAL, BIC.

Stoianov **(**2011**)** proposed the dynamic S-Box of AES and tested it by using the following tests: Balancing, Nonlinearity, Completeness, SAC, Low XOR Table, Diffusion Order, invertability, Static Criteria (Independence between the input and output data, Independence between the output and input data, Independence between the output and output data), Dynamic Criteria (Dynamic Independence between the input and output data, Dynamic Independence between the output and input data, Dynamic Independence between the output and output data), and Private Criteria (Completeness of S-BOX and Non-contradiction).

However, the evaluation of these studies did not include the analysis of the security of S-Boxes in BA in terms of AVAL, SAC, and BIC. Thus, the current research conducted two case studies and published findings on the analysis of the security of S-Boxes in BA and the output of BA. The first case study is related to the analysis of the security of S-Box in BA in terms of AVAL, SAC, and BIC. The second study is related to the avalanche text and correlation coefficient of BA.

Some studies used the avalanche effect to measure the strength of algorithm outputs. Among these studies were Dawson, Gustafson, & Pettitt (1992), Castro, Sierra, Seznec, Izquierdo, & Ribagorda (2005), Mohammed et al.(2009), Doganaksoy, Ege, Koçak, & Sulak (2010), Agrawal & Sharma (2010), Mohan & Reddy (2011), Ramanujam & Karuppiah (2011), Juremi et al. (2012), Ariffin (2012), Sulaiman et al. (2012b).

The following sections describe each criterion.

2.3.8.3.1 Avalanche Criterion

Feistel (1973) identified AVAL as a property of S-Boxes. AVAL is a crucial cryptographic property of block ciphers whereby a small number of bit differences in the input plaintext or key leads to an “avalanche” of changes that causes a large number of differences in ciphertext bit.

Mathematically, a function f: $\{0, 1\}^n \rightarrow \{0, 1\}^n$ satisfies AVAL whenever one input bit is changed. On average, half of the output bits change, where i and j $\{(i, j)\ 1 \leq i, j \leq n\}$ are the input and output bits, respectively.

An $n \times n$ S-Box is said to satisfy the *AVAL* criterion for all $i = 1, 2, n$

$$\frac{1}{2^n}\sum_{j=1}^{n} w(a_j^{ei}) = \frac{n}{2} \tag{2.13}$$

where

$$w(a_j^{ei}) = \sum_{\substack{for X \in whole \\ input alphabet}} a_j^{ei} \tag{2.14}$$

where e_i is the unit vector with bit i equal to 1 and all other bits equal to 0. A^{ei} exclusive-or sums will be referred to as avalanche vectors, each of which contains n bits or avalanche variables, with changes only to the *ith* bit in the input string.

A^{ei} is defined as:

$$A^{ei} = f(X) \oplus f(X \oplus e_i) = \left[a_1^{ei}\, a_2^{ei} \ldots a_n^{ei}\right], \tag{2.15}$$

where $a_j^{ei} \in \{0, 1\}$.

The total change in the j*th* avalanche variable, a_j^{ei}, is computed over the whole input alphabet of size 2^n $\quad 0 < W(a_j^{ei}) < 2^n$.

Equation (2.15) can be manipulated to define an AVAL parameter $k_{AVAL}(i)$ as

$$k_{AVAL}((i)) = \frac{1}{n2^n}\sum_{j=1}^{n} w\left(a_j^{ei}\right) = \frac{1}{2} \tag{2.16}$$

k_{AVAL}(i) can take values in the range [0,1]. This parameter should be interpreted as the probability of change in the overall output bits with changes only on the i-*th* bit in the input string. If k_{AVAL} (i) is not 1/2 for any value of i, then S-Box does not satisfy AVAL.

If k_{AVAL} (i) is approximately 1/2 for all i, then the S-Box satisfies AVAL with a small error region (Vergili &Yücel, 2000; Kavut & Yücel, 2001; Vergili & Yücel, 2001; Ahmed, n.d.).

- **Relative Error for the Avalanche Criterion**

Vergili and Yücel (2001) concluded that S-Box can satisfy Equation (2.16) for small values of n, but when the values of $n \geq 6$, satisfying AVAL becomes difficult. Therefore, we should expect the criterion in Equation (2.16) to be satisfied within an error range of $\pm \in A$. This error range is known as the relative error interval for AVAL. An S-Box satisfies AVAL with $\pm \in A$ for all *i* when

$$\frac{1}{2}(1 - \in_A) \leq k_{AVAL}(i) \leq \frac{1}{2}(1 + \in_A) \tag{2.17}$$

is true. Given an S-Box, the corresponding relative error ϵ*A* can be calculated using Equation (2.18)

$$\in_A = \max_{1 \leq i \leq n} \left| 2K_{AVAL}(i) - 1 \right| \tag{2.18}$$

2.3.8.3.2 Strict Avalanche criterion

Webster and Tavares (1986) combined the completeness and avalanche properties into the SAC. An S-Box satisfies SAC if the probability of change in any output bit approximates 1/2 with changes in any input bit. In other words, an adversary A selects two values, *i* and *j,* where i is for the input of S, and j is for the output of S, $1 \leq i, j \leq n$ with the assumption that A does not know any value of input except the *i-*th bit, and all the other input bits (except *i-th*) are selected randomly. The probability is that the *j-th* bit will change when the *i-th* bit that is complemented approximates 1/2 for A.

Mathematically, SAC can be described as follows:

A function $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$ satisfies SAC for all $i, j \in (1, 2, ..., n)$. Flipping input bit *i* changes the output bit *j* with the probability of exactly 1/2. Thus, an S-Box satisfies the *SAC* if

$$\frac{1}{2^n} w(a_j^{ei}) = \frac{1}{2^n} \tag{2.19}$$

For all *i, j* can be modified to define an SAC parameter $k_{SAC}(i, j)$ as

$$k_{SAC}(i, j) = \frac{1}{2^n} w(a_j^i) = \frac{1}{2} \tag{2.20}$$

kSAC *(i, j*) can take values in the range [0,1]. Therefore, the interpretation is that the probability of change in the *j-th* output bit is based on the changes in the *i-th* bit in the input string. If kSAC (*i, j*) is not 1/2 for any *(i, j*) pair, then the S-Box is does not satisfy the SAC. However, the satisfaction of Equation (2.20) for all values of *i* and *j* is not a realistic expectation; hence, Equation (2.20) should be interprets within an error interval of $\{\pm \epsilon S\}$. This condition means that given that kSAC (*i*, *j*) approximates 1/2 for all (*i*, *j*) pairs, then the S-Box satisfies SAC within a small error region (Vergili & Yücel, 2000; Kavut & Yücel, 2001; Vergili & Yücel, 2001; Ahmed, n.d.).

- **Relative Error for the Strict Avalanche criterion**

SAC is a more specialized form of the AVAL criterion, such that the number of S-Boxes that satisfy SAC is smaller than the number of S-Boxes that satisfy AVAL. These criteria with large S-Box size ($n \geq 6$) will be satisfied for SAC with a low error range. Therefore, by modifying Equation (2.21), an S-Box satisfies the *SAC* within $\pm \in_{SAC}$ for all *i* and *j*.

The following equation satisfies:

$$\frac{1}{2}(1 - \in_{SAC}) \leq k_{SAC}(i, j) \leq \frac{1}{2}(1 + \in_{SAC}) \tag{2.21}$$

Using (2.21) for a given S-Box, the relative error the ϵ_{SAC} for *SAC* can be calculated as

$$\in_{SAC} = \max_{1 \leq i, j \leq n} \left| 2k_{SAC}(i, j) - 1 \right| \tag{2.22}$$

2.3.8.3.3 Bit Independence Criterion

Webster and Tavares (1986) identified BIC as another property for any cryptographic transformation. For a given set of avalanche vectors generated by complementing a single plaintext bit, all the avalanche variables should be pairwise independent. To measure the degree of independence between a pair of avalanche variables, their correlation coefficient should be calculated.

Independent of the output bits is an element that ensures any two output bits *i* and *j* are "independent" of each other. That is, bits *i* and *j* are not equal to each other either significantly more or significantly less than half the time (over all possible input vectors). This condition is necessary for the space can be reduced in certain attacks if the correlation between two output bits is significantly other than zero.

When either the SAC or AVAL variable independence requirement is not satisfied, then a cryptanalyst can obtain some information about the statistical properties of the system, whereby he could conceivably use the information to his advantage in an attack on the system. Mathematically, BIC is defined as follows:

A function *f*: $\{0, 1\}^n \rightarrow \{0, 1\}^n$ satisfies the *BIC* for all $i, j, k \in \{1, 2, \ldots, n\}$, with $j \neq k$. Inverting the input bit *i* causes output bits *j* and *k* to change independently. The correlation coefficient computed between the *j-th* and *k-th* components of the output difference string is known as the avalanche vector A^{ei}. BIC corresponds to the effect of the *ith* input bit change on the *j-th* and *k-th* bits of A^{e}, which is defined as

$$BIC^{ei}(a_j, a_k) = \left| corr(a_j^{ei}, a_k^{ei}) \right| \qquad (2.23)$$

Overall, the BIC for the S-Box is calculated as

$$BIC(f) = \max_{\substack{1\leq i \leq n \\ 1 \leq j,k \leq n \\ j \neq k}} BIC^{ei}\left(a_j, a_k\right) \tag{2.24}$$

BIC (f) takes values in the range [0, 1]. In the case of binary variables, a correlation coefficient of 0 indicates that the variables are independent. In addition, the variables will always be identical if the correlation coefficient is equal to 1, and a value of -1 means that they always complement one another. In brief, BIC (f) is ideally equal to zero, and in the worst case, it is equal to one (Adams & Tavares, 1990; Vergili & Yücel, 2000; Vergili & Yücel, 2001; Kavut & Yücel, 2001).

- **Relative Error for Bit Independence Criterion**

The relative error for BIC can be defined as (Vergili & Yücel, 2000; Vergili &Yücel, 2001; Kavut & Yücel, 2001):

$$\in_{BIC} = BIC(f) \tag{2.25}$$

2.3.8.4 Cryptanalysis

Cryptanalysis serves an essential function in cipher design. Cryptanalysis uses a mathematical formula to search for the vulnerabilities of an algorithm to break it when the key is unknown. Numerous cryptanalysis theories and practices are relevant to block ciphers. Among them, two very powerful cryptanalysis techniques applied to symmetric–key block ciphers are differential cryptanalysis and LC (Schneier, 1996; Heys, 2002).

2.3.8.4.1 Linear Cryptanalysis

Matsui (1994) discovered the LC technique that has been applied successfully against the DES and FEAL block ciphers. This technique is one of the most widely known attacks on block ciphers and has become a benchmark technique for evaluating the security of any new modern cipher. LC is a Known Plaintext (KP) attack, whereby the cryptanalysis has a set of plaintext and corresponding ciphertexts.

The basic tool of a linear attack is a linear distinguisher, which comprises a linear relationship between bits of plaintext, ciphertext, and key, holding a non-uniform probability (different from 1/2). This discrepancy between the associated probability of a cipher and that of random behavior is known as the bias. The number of known plaintexts needed for a highly successful attack is inversely proportionate to the bias. Thus, a larger bias requires less plaintext for a highly successful attack. Linearity refers to a mod-2 bit-wise operation or XOR that is denoted by $\oplus$. An expression of linearity can be of the form

$$X_{i1} \oplus X_{i2} \oplus \ldots X_{iu} \oplus Y_{j1} \oplus Y_{j2} \oplus \ldots\ Y_{jv} = 0 \qquad (2.26)$$

where X_i represents the i-th bit of the input $X = [X_1, X_2 \ldots, X_n]$, and Y_j represents the j-th bit of the output $Y = [Y_1, Y_2 \ldots, Y_n]$. This equation represents the XOR of u input bits and v output bits, which determine the high or low probability of occurrence. If a block cipher displays a tendency for linear equations to hold a probability higher or lower than $\frac{1}{2}$, the ciphers exhibit poor randomization capabilities (Heys, 2002).

2.3.8.4.2 Differential Cryptanalysis

Biham and Shamir (1993) introduced differential cryptanalysis, which remains one of the most influential techniques in block cipher cryptanalysis. This technique is also known as chosen plaintext attack, where the attacker must obtain encrypted ciphertexts from a set of plaintexts of his choice. Differential cryptanalysis is the study of how differences in an input can affect the resultant difference at the output. In the case of a block cipher, differential cryptanalysis refers to a set of techniques for tracing differences through the network of transformation, locating where the cipher exhibits nonrandom behavior, and exploiting such properties to recover the SK. By using this method, Biham and Shamir discovered a chosen-plaintext attack against DES that was more efficient than brute force. Differential cryptanalysis looks specifically at pairs of ciphertext and plaintext with particular differences (Heys, 2002). Consider a system with input $X = [X_1, X_2 ..., X_n]$ and output $Y = [Y_1, Y_2 ..., Y_n]$. We select two inputs from the system as X' and X'' with the corresponding outputs Y' and Y''. The input difference is given by $\Delta X = X' \oplus X''$, where $\oplus$ represents a bit-wise XOR of the n-bit vectors, such that

$$\Delta X = [\Delta X_1\ \Delta X_2 ... ,\ \Delta X_n] \tag{2.27}$$

where $\Delta X_i = X_i' \oplus X_i''$, with X_i' and X_i'' representing the i*th* bit of X' and X'', respectively.

Similarly, $\Delta Y = Y' \oplus Y''$ is the output difference and

$$\Delta Y = [\Delta Y_1\ \Delta Y_2 ...,\ \Delta Y_n] \tag{2.28}$$

where $\Delta Y_i = Y_i' \oplus Y_i''$.

In an ideal randomizing cipher, the probability of occurrence of a particular output difference ΔY given a particular input difference ΔX is $1/2^n$, where n is the number of bits of X. Differential cryptanalysis exploits a situation where a particular ΔY occurs given a particular input difference ΔX with a very high probability (much greater than $1/2^n$). The pair $(\Delta X, \Delta Y)$ is referred to as a differential. It is a chosen plaintext attack, which indicates that the attacker can select an input and examine the output when trying to derive the key. This attack uses different propagation properties of a cipher to deduce the key bits. The attacker will select pairs of inputs X' and X'' to satisfy a particular ΔX, with the knowledge of a high probability of a particular ΔY.

2.3.8.5 Computational Efficiency

An algorithm's complexity or computational efficiency is determined by the computational power needed to execute it. Generally, the computational complexity of an algorithm is expressed in what is called "big *O*" notation: the order of magnitude of the computational complexity. It's just the term of the complexity function which grows the fastest as *n* gets larger; all lower-order terms are ignored. For example, if the time complexity of a given algorithm is $4n^2 + 7n + 12$, then the computational complexity is on the order of n^2, expressed $O(n^2)$. Where *n* is the size of the input.

Generally, algorithms are classified according to their time or space complexities. An algorithm is constant if its complexity is independent of n: *O(1)*. An algorithm is linear, if its time complexity is *O(n)*. Algorithms can also be quadratic, cubic, and so on. All these algorithms are polynomial; their complexity is *$O(n^m)$*, when m is a constant. The classes of algorithms that have a polynomial time complexity are called polynomial-time algorithms. Table 2.8 illustrates class of the algorithms (Schneier, 1996; Denning, 1982).

Table 2.8

Class of the algorithms and number of operations

Number of operations			
Class	**Complexity**	**For n=10^6**	**Real Time**
Constant	*$O(1)$*	1	1μsec
Linear	*$O(n)$*	10^6	1second
Quadratic	*$O(n^2)$*	10^{12}	10 days
Cubic	*$O(n^3)$*	10^{18}	27397 years

2.4 Summary

This chapter presents the literature review. The discussion include basic concepts of cryptography, coordinate systems, and past work related to the BA, dynamic S-Box, and 3D block cipher.

CHAPTER THREE

RESEARCH METHODOLOGY

3.1 Introduction

This research was conducted in three phases: Phase 1, RAF Design; Phase 2, RAF Implementation; and Phase 3, RAF Verification. Figure 3.1 shows the overview of the research process. The details of each phase are presented in Section 3.2 to 3.4.

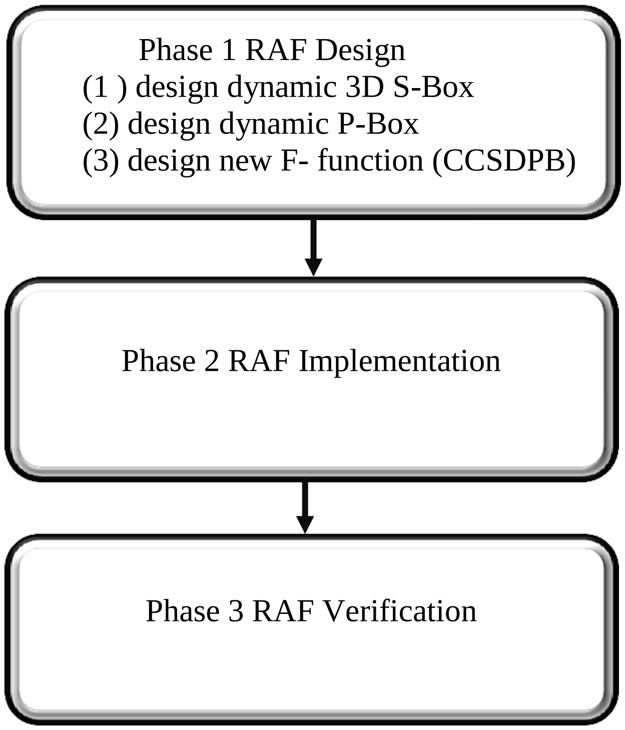

Phase 1 RAF Design
(1) design dynamic 3D S-Box
(2) design dynamic P-Box
(3) design new F- function (CCSDPB)
Phase 2 RAF Implementation
Phase 3 RAF Verification

Figure 3.1. Overview of the research process

3.2 Phase 1 RAF Design

This phase consists of three steps: (1) design of the dynamic 3D S-Box, (2) design of the dynamic P-Box, and (3) design of a new F-Function. The following sections explain the steps.

3.2.1 Dynamic 3D S-Box

The process of designing 3D S-Box consists of three parts: (1) generation of random SKs, (2) define transformation of the right cylinder, and (3) conduct byte permutation (byte relocation and byte transformation). However, before performing Phase 1, the 3D S-Box structure is initially prepared by converting the right cylinder (Figure 3.2) into 3D S-Box using the following matrix:

A=$[a_{ijk}]_{884}$ where a_{ijk} related with the Point $P(\rho_i, \phi_j, z_k)$

Such that

$$\rho_i \in \{1,2,3,\ldots 8\},\ \ i = 0,1,2,\ldots 7.$$

$$\phi_j \in \left\{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}, \pi, \frac{5\pi}{4}, \frac{3\pi}{2}, \frac{7\pi}{4}\right\}, j = 0,1,2,\ldots 7.$$

$$z_k \in \{1,2,\ldots 4\}, k = 0, 1,2,3.$$

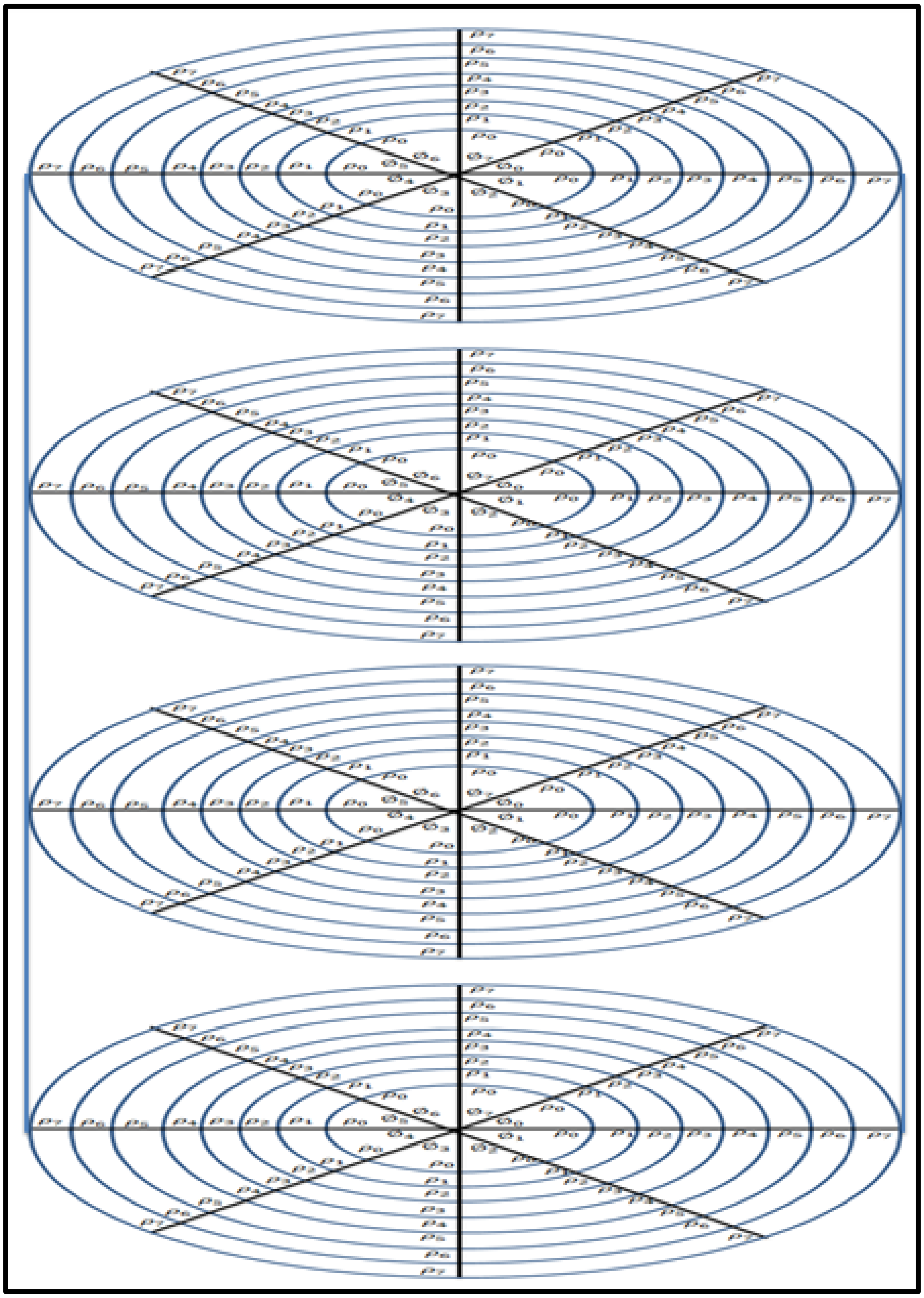


Figure 3.2. Right cylinder

Figure 3.3 shows the Cross-Section of the right cylinder

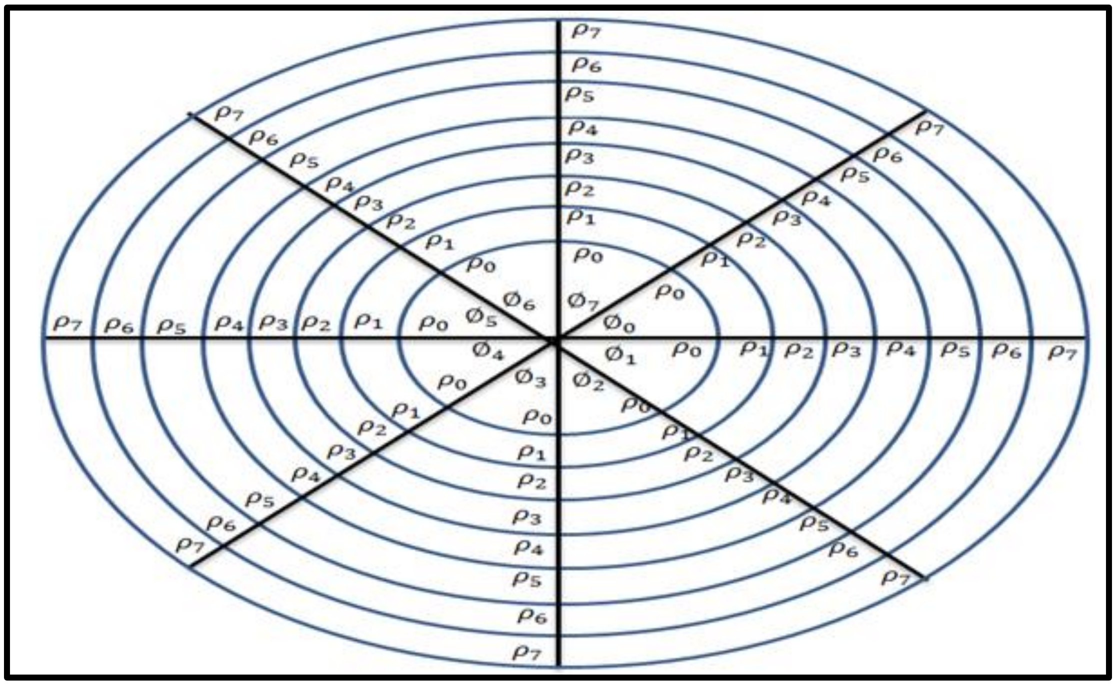


Figure 3.3. Cross-Section of the right cylinder

The output of this step is presented in Figure 3.4. The 3D array has 8 bit input and 8 bit output.

Section$_0$ (8×8) bytes

	ϕ_0	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_7
ρ_0	a_{000}	a_{010}	a_{020}	a_{030}	a_{040}	a_{050}	a_{060}	a_{070}
ρ_1	a_{100}	a_{110}	a_{120}	a_{130}	a_{140}	a_{150}	a_{160}	a_{170}
ρ_2	a_{200}	a_{210}	a_{220}	a_{230}	a_{240}	a_{250}	a_{260}	a_{270}
ρ_3	a_{300}	a_{310}	a_{320}	a_{330}	a_{340}	a_{350}	a_{360}	a_{370}
ρ_4	a_{400}	a_{410}	a_{420}	a_{430}	a_{440}	a_{450}	a_{460}	a_{470}
ρ_5	a_{500}	a_{510}	a_{520}	a_{530}	a_{540}	a_{550}	a_{560}	a_{570}
ρ_6	a_{600}	a_{610}	a_{620}	a_{630}	a_{640}	a_{650}	a_{660}	a_{670}
ρ_7	a_{700}	a_{710}	a_{720}	a_{730}	a_{740}	a_{750}	a_{760}	a_{770}

Section$_1$ (8×8) bytes

	ϕ_0	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_7
ρ_0	a_{001}	a_{011}	a_{021}	a_{031}	a_{041}	a_{051}	a_{061}	a_{071}
ρ_1	a_{101}	a_{111}	a_{121}	a_{131}	a_{141}	a_{151}	a_{161}	a_{171}
ρ_2	a_{201}	a_{211}	a_{221}	a_{231}	a_{241}	a_{251}	a_{261}	a_{271}
ρ_3	a_{301}	a_{311}	a_{321}	a_{331}	a_{341}	a_{351}	a_{361}	a_{371}
ρ_4	a_{401}	a_{411}	a_{421}	a_{431}	a_{441}	a_{451}	a_{461}	a_{471}
ρ_5	a_{501}	a_{511}	a_{521}	a_{531}	a_{541}	a_{551}	a_{561}	a_{571}
ρ_6	a_{601}	a_{611}	a_{621}	a_{631}	a_{641}	a_{651}	a_{661}	a_{671}
ρ_7	a_{701}	a_{711}	a_{721}	a_{731}	a_{741}	a_{751}	a_{761}	a_{771}

Section$_2$ (8×8) bytes

	ϕ_0	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_7
ρ_0	a_{002}	a_{012}	a_{022}	a_{032}	a_{042}	a_{052}	a_{062}	a_{072}
ρ_1	a_{102}	a_{112}	a_{122}	a_{132}	a_{142}	a_{152}	a_{162}	a_{172}
ρ_2	a_{202}	a_{212}	a_{222}	a_{232}	a_{242}	a_{252}	a_{262}	a_{272}
ρ_3	a_{302}	a_{312}	a_{322}	a_{332}	a_{342}	a_{352}	a_{362}	a_{372}
ρ_4	a_{402}	a_{412}	a_{422}	a_{432}	a_{442}	a_{452}	a_{462}	a_{472}
ρ_5	a_{502}	a_{512}	a_{522}	a_{532}	a_{542}	a_{552}	a_{562}	a_{572}
ρ_6	a_{602}	a_{612}	a_{622}	a_{632}	a_{642}	a_{652}	a_{662}	a_{672}
ρ_7	a_{702}	a_{712}	a_{722}	a_{732}	a_{742}	a_{752}	a_{762}	a_{772}

Section$_3$ (8×8) bytes

	ϕ_0	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_7
ρ_0	a_{003}	a_{013}	a_{023}	a_{033}	a_{043}	a_{053}	a_{063}	a_{073}
ρ_1	a_{103}	a_{113}	a_{123}	a_{133}	a_{143}	a_{153}	a_{163}	a_{173}
ρ_2	a_{203}	a_{213}	a_{223}	a_{233}	a_{243}	a_{253}	a_{263}	a_{273}
ρ_3	a_{303}	a_{313}	a_{323}	a_{333}	a_{343}	a_{353}	a_{363}	a_{373}
ρ_4	a_{403}	a_{413}	a_{423}	a_{433}	a_{443}	a_{453}	a_{463}	a_{473}
ρ_5	a_{503}	a_{513}	a_{523}	a_{533}	a_{543}	a_{553}	a_{563}	a_{573}
ρ_6	a_{603}	a_{613}	a_{623}	a_{633}	a_{643}	a_{653}	a_{663}	a_{673}
ρ_7	a_{703}	a_{713}	a_{723}	a_{733}	a_{743}	a_{753}	a_{763}	a_{773}

Figure 3.4. Representation of the right cylinder in 3D array

Figure 3.4 shows the representation of the right cylinder in 3D array. Each square (array 8×8) is a set of 64 bytes representing a section of the cylindrical coordinate system for the right cylinder. Each row in the array represents one circle from eight nested circles in the section of the right cylinder. Each individual byte in the section consists of three indices: the first index acts as a row number (ρ), the second index represents a column number (ϕ), whereas the third index represents a section number (z). Therefore, any point in the right cylindrical coordinate is referred to as a_{ijk}.

In BA, each quarter (8-bit) is used as an entry to one of the S-Boxes, thus requiring four S-Boxes. In RAF, all four quarters (every quarter is 16-bit) are used as entry to the same 3D S-Box. Two procedures of byte permutation were applied to permute the elements of 3D S-Box after each entry.

Every section in the 3D S-Box (right cylinder) can be divided into four sets of elements. The sets of elements are called quarters. These quarters represent circles, halves circles, tracks, and set of random points in right cylinder.

Once the structure of 3D S-Box is prepared, Phase 1 is performed. The activities conducted in part 1 (generation of random SKs), part 2 (define transformation of the right cylinder), and part 3 (conduct byte permutation) are described in Sections 3.2.1.1 to 3.2.1.3, consecutively.

3.2.1.1 Generation of Random SKs

The random function is used to generate a random SK. The seed of the random function is computed as follows:

Seed = (*XL* XOR round subkey) + block sequence of the plaintext (3.1)

Based on Equation (3.1), the seed of the random function in RAF is the left side of the round input (XL) that was XORed with a round subkey. The result is added to the block sequence of the plaintext. Thus every block has different seed in encryption process. The pseudo code is used to generate random SKs as follows:-

```
static unsigned __int64 next = 1;

    /* RAND_MAX assumed to be 32767 */
    int myrand(void)
              {
                next = next * 1103515245 + 12345;
                return((unsigned)(next/65536) % 32768);
               }
   void mysrand(unsigned __int64 seed)
              {
      next = seed;
             }
```

Five sets of SKs are generated using the random function in. The random SKs are in the interval [0,3] and are given below.

$SK_i = k_{ij}$ *where* $i=0, 1, 2, 3, 4$ $j=0, 1, 2, 3$

$SK_0 = \{ k_{00}, k_{01}, k_{02}, k_{03} \}$
$SK_1 = \{ k_{10}, k_{11}, k_{12}, k_{13} \}$
$SK_2 = \{ k_{20}, k_{21}, k_{22}, k_{23} \}$
$SK_3 = \{ k_{30}, k_{31}, k_{32}, k_{33} \}$
$SK_4 = \{ k_{40}, k_{41}, k_{42}, k_{43} \}$

In every set, the SKs were generated without repetition to ensure that all sections will be chosen and that all the elements of the quarters in 3D S-Box will be swapped.

The numbers of the first set of SKs represent the four sections of the 3D S-Box and are used to choose every two sections. For example, the numbers: 1, 2, 0, 3 in the first set (SK_0), the second and the third sections, as well as the first and fourth sections, are selected together. The last four sets of SKs (SK_1 to SK_4) represent the four quarters of sections. Table 3.1 illustrates an example of five sets of SKs.

Table 3.1

Five Sets of SKs

No of Sets	Sets	Representations
SK_0	0 2 1 3	section number
SK_1	1 2 0 3	quarter numbers of the first section
SK_2	3 1 0 2	quarter numbers of the second section
SK_3	1 3 2 0	quarter numbers of the third section
SK_4	1 0 3 2	quarter numbers of the fourth section

The SKs in SK_1 represent the quarter number in the first section, whereas the SKs in SK_2 represent the quarter numbers in the second section. The SKs in SK_3 represent the quarter numbers in the third section. Finally, the SKs in SK_4 represent the quarter numbers in the fourth section.

3.2.1.2 Define Transformations of the Right Cylinder

The right cylinder which is used in the design defined in the cylindrical coordinate as follow

$$S = \{ (\rho, \phi, z) \colon 1 \leq \rho \leq 8,\ 0 \leq \phi < 2\pi,\ 1 \leq z \leq 4\}$$

The transformation on the right cylinder it can be defined as follow:

$$f\ (\rho, \emptyset, z) = (\rho \mp \rho_0, \emptyset \mp \emptyset_0, z \mp z_0) \tag{3.2}$$

where

$\rho_0 \in \{0, 1, \ldots, 7\}$,

$\emptyset_0 \in \{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{8}, \pi, \frac{5\pi}{4}, \frac{3\pi}{2}, \frac{7\pi}{4}\}$,

$z_0 \in \{0, 1, 2, 3\}$

This transformation converts the point $P(\rho, \emptyset, z)$ ϵ S to point $P'(\rho, \emptyset, z)$, which also belongs to S. The eight types of transformations on the right cylinder S in Table 3.2 are conducted as follows:

1. In this study, the right cylinder is divided into four sections, as shown in Figure 3.2, which is indexed to $k_0,\ k_1,\ k_2,\ k_3 \epsilon K = \{0, 1, 2, 3\}$.
2. In every section of the right cylinder, we have eight nested circles (contours), as shown in Figure 3.3.

In mathematical notation, consider

$S_k = \{Q_{jk} : Q_{jk}\ is\ circle\ or\ half\ circle\ or\ track\ or\ set\ of\ random\ points\ j = 1, 2, 3, 4\}\ for\ k \in K = \{0, 1, 2, 3\}$.

Table 3.2

Eight Transformations of the Right Cylinder

Cases	Transformation	Kind
Case z_0 =0	$T: S_{k_n} \to S_{k_n}$	
ρ_0=0, $\emptyset_0 = 0$	$T_1(\rho, \emptyset, z) = (\rho, \emptyset, z)$	1
ρ_0=0, $\emptyset_0 \neq 0$	$T_2(\rho, \emptyset, z) = (\rho, \emptyset \pm \emptyset_0, z)$	2
$\rho_0 \neq$0, $\emptyset_0 = 0$	$T_3(\rho, \emptyset, z) = (\rho \pm \rho_0, \emptyset, z)$	3
$\rho_0 \neq$0, $\emptyset_0 \neq 0$	$T_4(\rho, \emptyset, z) = (\rho \pm \rho_0, \emptyset \pm \emptyset_0, z)$	4
Case $z_0 \neq 0$	T: $S_{k_n} \to S_{k_m}$	
ρ_0=0, $\emptyset_0 = 0$	$T_5(\rho, \emptyset, z) = (\rho, \emptyset, z \pm z_0)$	5
ρ_0=0, $\emptyset_0 \neq 0$	$T_6(\rho, \emptyset, z) = (\rho, \emptyset \pm \emptyset_0, z \pm z_0)$	6
$\rho_0 \neq$0, $\emptyset_0 = 0$	$T_7(\rho, \emptyset, z) = (\rho \pm \rho_0, \emptyset, z \pm z_0)$	7
$\rho_0 \neq$0, $\emptyset_0 \neq 0$	$T_8(\rho, \emptyset, z) = (\rho \pm \rho_0, \emptyset \pm \emptyset_0, z \pm z_0)$	8

1-Transformation of the first kind

Let $T_1: S_{k_n} \to S_{k_n},\ k_n \in K$,

T_1 (ρ, $\emptyset$, z)= (ρ,$\emptyset$, z).

This transformation is an identity transformation, that's mean it preserved all points of the domain.

2-Transformation of the second kind

Let $T_2: S_{k_n} \to S_{k_n}, k_n \in K$,

T_2 (ρ, $\emptyset$, z)= (ρ, $\emptyset \pm \emptyset_0$, z).

This transformation rotates every point in Q_{jk_n} of section k_n or points in the section k_n by the angles$\pm\emptyset_0$.

3-Transformation of the third kind

Let $T_3: S_{k_n} \to S_{k_n},\ k_n \in K,$

$T_3\ (\rho, \emptyset, z)$= $(\rho \pm \rho_0, \emptyset, z)$.

This transformation translates every point in Q_{jk_n} of the section k_n or points in the section $k_n$with increment of radius$\pm\ \rho_0$.

4-Transformation of the fourth kind

Let $T_4: S_{k_n} \to S_{k_n\,,}\ k_n \in K,$

$T_4\ (\rho, \emptyset, z)$= $(\rho\ \pm\rho_0, \emptyset \pm \emptyset_0, z)$.

This transformation rotates every point in Q_{jk_n} of the section k_n or points in the section k_n by the angle$s \pm \emptyset_0$ and translates by increment of the radius $\pm\ \rho_0$.

5-Transformation of the fifth kind

Let $T_5: S_{k_n} \to S_{k_m},\ k_n \neq k_m \in K,$

$T_5\ (\rho, \emptyset, z)$= $(\rho, \emptyset, z \pm z_0)$.

This transformation translates every point in Q_{jk_n} of section $k_n$or points in the section k_n to another section $k_m$with increment$\pm z_0$.

6-Transformation of the sixth kind

Let $T_6: S_{k_n} \rightarrow S_{k_m}, k_n \neq k_m \in K,$

T_6 $(\rho, \emptyset, z)$= $(\rho, \emptyset \pm \emptyset_0, z \pm z_0)$.

This transformation rotates every point in $Q_{jk_n}$of the section $k_n$or points in the section k_n to another section k_m by the angle$s \pm \emptyset_0$ and translates with increment $\pm z_0$.

7-Transformation of the seventh kind

Let $T_7: S_{k_n} \rightarrow S_{k_m},\ k_n \neq k_m \in K,$

T_7 $(\rho, \emptyset$, z)= $(\rho \pm \rho_0, \emptyset, z \pm z_0)$.

This transformation translates every point in $Q_{jk_n}$of the section k_n or points in the section k_n to another section k_m with increment $\pm z_0$ and radius $\pm \rho_0$.

8-Transformation of the eighth kind

Let $T_8: S_{k_n} \rightarrow S_{k_m}$, $k_n \neq k_m \in K,$

T_8 $(\rho, \emptyset, z)$= $(\rho \pm \rho_0$, $\emptyset \pm \emptyset_0, z \pm z_0)$.

This transformation translates every point in $Q_{jk_n}$of the section $k_n$or points in the section k_n to another section $k_m$with increment $\pm z_0$, radius $\pm\rho_0$, and rotates by angle$s \pm \emptyset_0$.

3.2.1.3 Byte Permutation

Two procedures of byte permutation based on SKs are conducted in every round to permute the elements of 3D S-Box. The first procedure is known as Byte Relocation (BR). BR is used to generate one dynamic 3D S-Box. The second procedure of byte permutation is called Byte Transformation (BT). BT is used to generate three dynamic 3D S-Boxes.

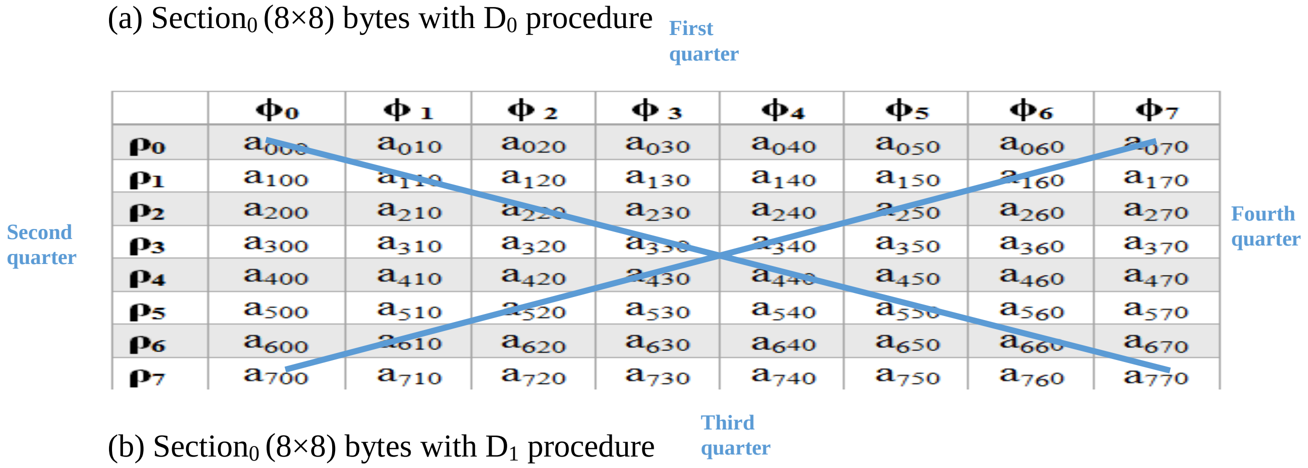
3.2.1.3.1 Byte Relocation

BR is used in four procedures, namely, D_0, D_1, D_2, and D_3. BR is used to swap between the elements of the quarters in the sections. Every number from the last four sets of random SKs is used to determine one quarter of the section of 3D S-Box. D_0, D_1, D_2, and D_3 are conducted on the elements of the quarters of the sections in the first, second, third, and fourth rounds, respectively. This process is repeated in a cyclical manner from $D_{0,}$…,D_3 until 10 rounds are completed. $D_{0,}$…, D_3 are conducted on the quarters of 3D S-Box in the key expansion part.

BR is conducted in two steps. The first step is choosing two sections from four sections depending on the first set of random *SKs*. The second step is swapping between the elements of the quarters in the selected sections depending on the last four sets of SKs. The four procedures D_0, D_1, D_2, and D_3 are conducted corresponds to T_5 if the swapping the elements of the quarter in the selected section with the elements of the quarter in another selected section correspondingly; otherwise, the swapping corresponds to T_6 to T_8.

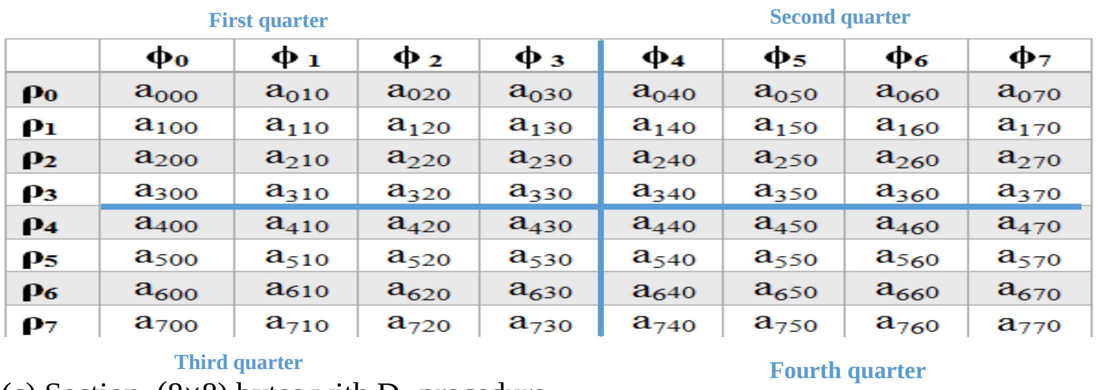
The division process into quarters differs for D_0, D_1, D_2, and D_3. Figure 3.5 illustrates the division process of the sections with each BR on only the first section ($section_0$). Figure 3.6 illustrates the flowchart of BR.

(a) $Section_0$ (8×8) bytes with D_0 procedure


First quarter
Second quarter
Fourth quarter
Third quarter

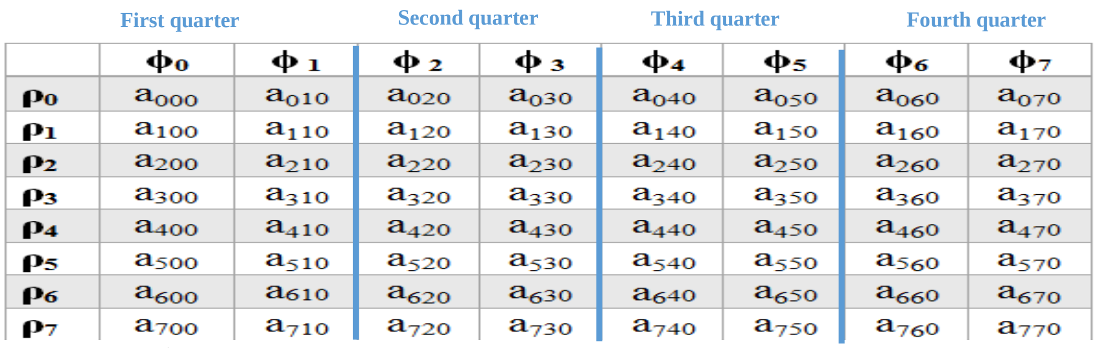
	ϕ_0	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_7
ρ_0	a_{000}	a_{010}	a_{020}	a_{030}	a_{040}	a_{050}	a_{060}	a_{070}
ρ_1	a_{100}	a_{110}	a_{120}	a_{130}	a_{140}	a_{150}	a_{160}	a_{170}
ρ_2	a_{200}	a_{210}	a_{220}	a_{230}	a_{240}	a_{250}	a_{260}	a_{270}
ρ_3	a_{300}	a_{310}	a_{320}	a_{330}	a_{340}	a_{350}	a_{360}	a_{370}
ρ_4	a_{400}	a_{410}	a_{420}	a_{430}	a_{440}	a_{450}	a_{460}	a_{470}
ρ_5	a_{500}	a_{510}	a_{520}	a_{530}	a_{540}	a_{550}	a_{560}	a_{570}
ρ_6	a_{600}	a_{610}	a_{620}	a_{630}	a_{640}	a_{650}	a_{660}	a_{670}
ρ_7	a_{700}	a_{710}	a_{720}	a_{730}	a_{740}	a_{750}	a_{760}	a_{770}

(b) $Section_0$ (8×8) bytes with D_1 procedure


First quarter
Second quarter
Third quarter
Fourth quarter

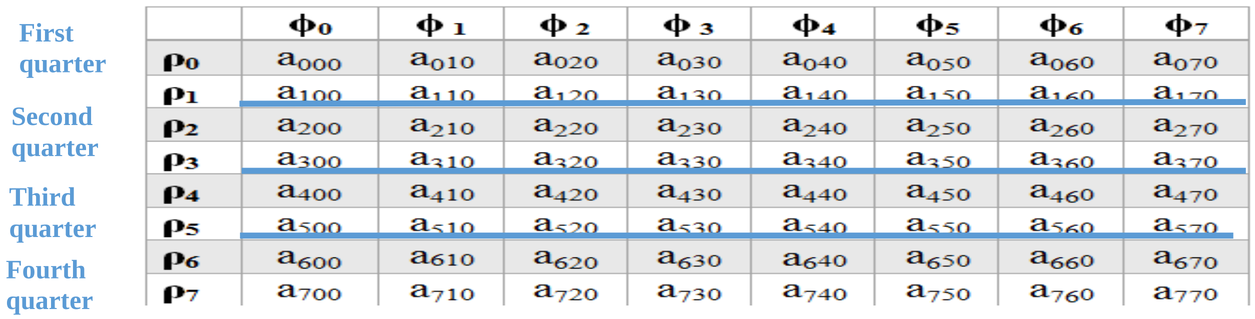
	ϕ_0	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_7
ρ_0	a_{000}	a_{010}	a_{020}	a_{030}	a_{040}	a_{050}	a_{060}	a_{070}
ρ_1	a_{100}	a_{110}	a_{120}	a_{130}	a_{140}	a_{150}	a_{160}	a_{170}
ρ_2	a_{200}	a_{210}	a_{220}	a_{230}	a_{240}	a_{250}	a_{260}	a_{270}
ρ_3	a_{300}	a_{310}	a_{320}	a_{330}	a_{340}	a_{350}	a_{360}	a_{370}
ρ_4	a_{400}	a_{410}	a_{420}	a_{430}	a_{440}	a_{450}	a_{460}	a_{470}
ρ_5	a_{500}	a_{510}	a_{520}	a_{530}	a_{540}	a_{550}	a_{560}	a_{570}
ρ_6	a_{600}	a_{610}	a_{620}	a_{630}	a_{640}	a_{650}	a_{660}	a_{670}
ρ_7	a_{700}	a_{710}	a_{720}	a_{730}	a_{740}	a_{750}	a_{760}	a_{770}

(c) $Section_0$ (8×8) bytes with D_2 procedure


First quarter
Second quarter
Third quarter
Fourth quarter

	ϕ_0	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_7
ρ_0	a_{000}	a_{010}	a_{020}	a_{030}	a_{040}	a_{050}	a_{060}	a_{070}
ρ_1	a_{100}	a_{110}	a_{120}	a_{130}	a_{140}	a_{150}	a_{160}	a_{170}
ρ_2	a_{200}	a_{210}	a_{220}	a_{230}	a_{240}	a_{250}	a_{260}	a_{270}
ρ_3	a_{300}	a_{310}	a_{320}	a_{330}	a_{340}	a_{350}	a_{360}	a_{370}
ρ_4	a_{400}	a_{410}	a_{420}	a_{430}	a_{440}	a_{450}	a_{460}	a_{470}
ρ_5	a_{500}	a_{510}	a_{520}	a_{530}	a_{540}	a_{550}	a_{560}	a_{570}
ρ_6	a_{600}	a_{610}	a_{620}	a_{630}	a_{640}	a_{650}	a_{660}	a_{670}
ρ_7	a_{700}	a_{710}	a_{720}	a_{730}	a_{740}	a_{750}	a_{760}	a_{770}

(d) $Section_0$ (8×8) bytes with D_3 procedure



		ϕ_0	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_7
First quarter	ρ_0	a_{000}	a_{010}	a_{020}	a_{030}	a_{040}	a_{050}	a_{060}	a_{070}
	ρ_1	a_{100}	a_{110}	a_{120}	a_{130}	a_{140}	a_{150}	a_{160}	a_{170}
Second quarter	ρ_2	a_{200}	a_{210}	a_{220}	a_{230}	a_{240}	a_{250}	a_{260}	a_{270}
	ρ_3	a_{300}	a_{310}	a_{320}	a_{330}	a_{340}	a_{350}	a_{360}	a_{370}
Third quarter	ρ_4	a_{400}	a_{410}	a_{420}	a_{430}	a_{440}	a_{450}	a_{460}	a_{470}
	ρ_5	a_{500}	a_{510}	a_{520}	a_{530}	a_{540}	a_{550}	a_{560}	a_{570}
Fourth quarter	ρ_6	a_{600}	a_{610}	a_{620}	a_{630}	a_{640}	a_{650}	a_{660}	a_{670}
	ρ_7	a_{700}	a_{710}	a_{720}	a_{730}	a_{740}	a_{750}	a_{760}	a_{770}

Figure 3.5. (a-d) Quarters in the first section with Byte Relocation for (a) with D_0,(b) with D_1, (c) with D_2, and(d) with D_3

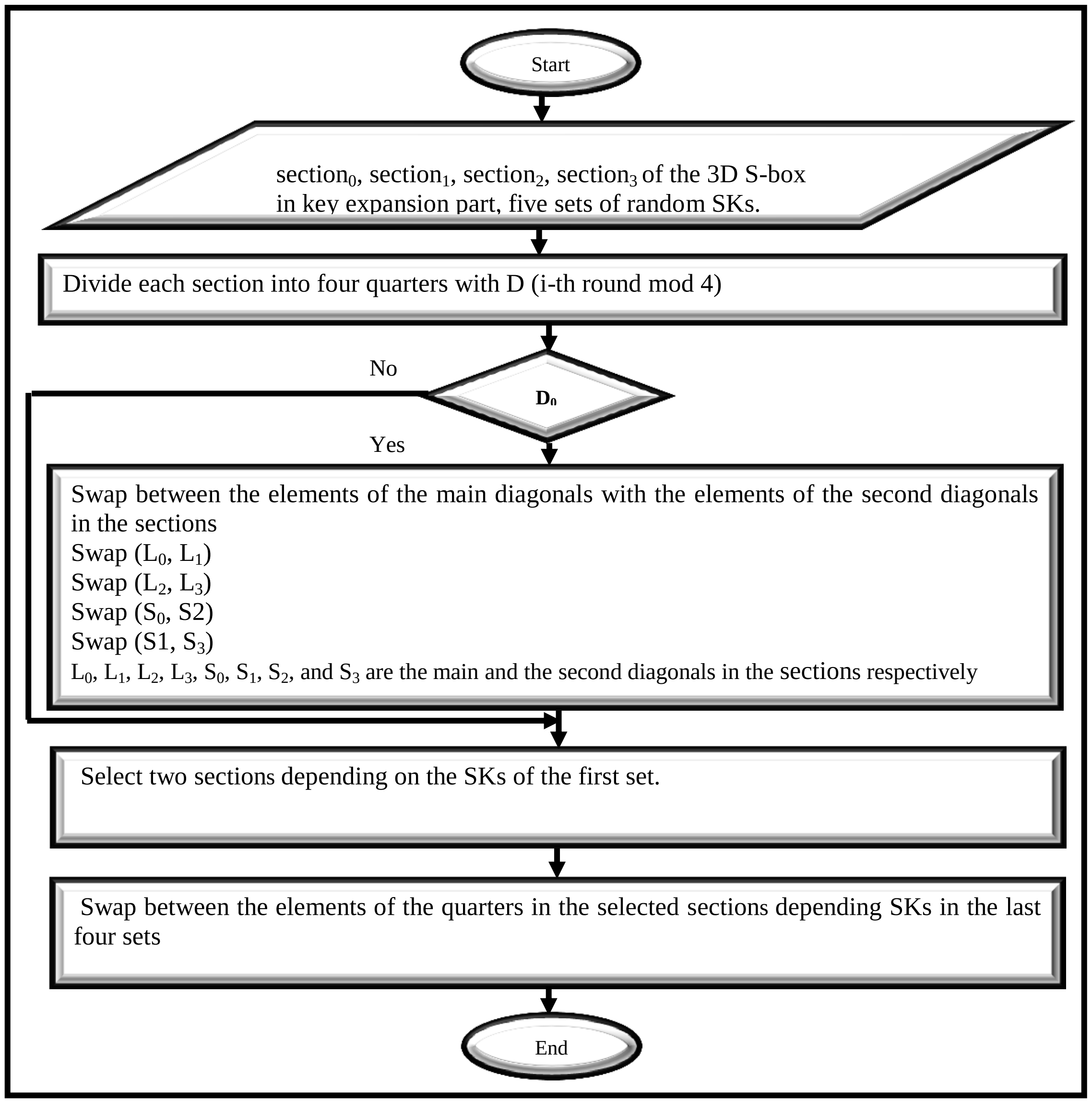
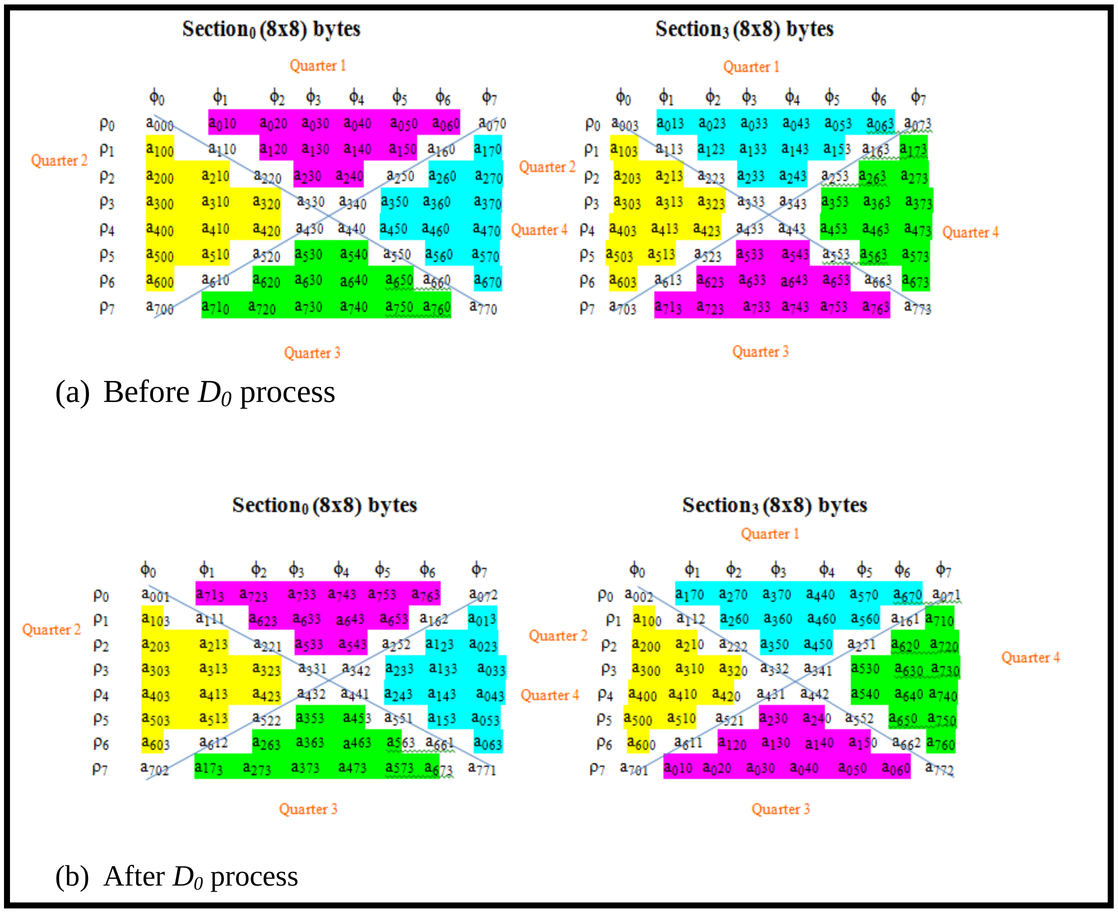

Start
section$_0$, section$_1$, section$_2$, section$_3$ of the 3D S-box in key expansion part, five sets of random SKs.
Divide each section into four quarters with D (i-th round mod 4)
No
D_0
Yes
Swap between the elements of the main diagonals with the elements of the second diagonals in the sections
Swap (L_0, L_1)
Swap (L_2, L_3)
Swap (S_0, S2)
Swap (S1, S_3)
L_0, L_1, L_2, L_3, S_0, S_1, S_2, and S_3 are the main and the second diagonals in the sections respectively
Select two sections depending on the SKs of the first set.
Swap between the elements of the quarters in the selected sections depending SKs in the last four sets
End

Figure 3.6. Flowchart of BR

An example of how BR is conducted is explained below.

Using SKs in Table 3.1, the elements of the second quarter of the first section are swapped with the elements of the second quarter of the third section, whereas the elements of the third quarter of the first section are swapped with the elements of the fourth quarter of the third section. The elements of the first quarter of the first section are swapped with the elements of the third quarter of the third section, and the elements of the fourth quarter of the first section are swapped with the elements of the first quarter of the third section. The same procedure is followed by the second and the fourth sections depending on SKs. Figure 3.7 explains the D_0 process for the first section only using the SKs in Table 3.1



(a) Before D_0 process

(b) After D_0 process

Figure 3.7. D_0 process for the first section (a before D_0 process, b after D_0 process)

3.2.1.3.2 Byte Transformation

BT was conducted on elements of the sections of the 3D S-Box (right cylinder). The BT was conducted after every two bytes (16 bit) were substituted. T_8 was conducted after the first two bytes were substituted from the 3D S-Box. T_4 was conducted after the second two bytes were substituted. Finally, T_6 was conducted after the third two bytes were substituted. The T_6 was conducted on the elements of the sections of the 3D S-Box from T_4, whereas T_4 was conducted on the elements of the sections of the 3D S-Box from T_8. Meanwhile, T_8 was conducted on the elements of the sections of the 3D S-Box from the first procedure (BR). BT (BT were used to permute the elements of sections of 3D S-Box after each two bytes were substituted. T_8, T_4, and T_6 are secret.

Figure 3.8 illustrates the rotation of a single circle of any section in the cylinder, where $\phi_0 = \pi/4$.

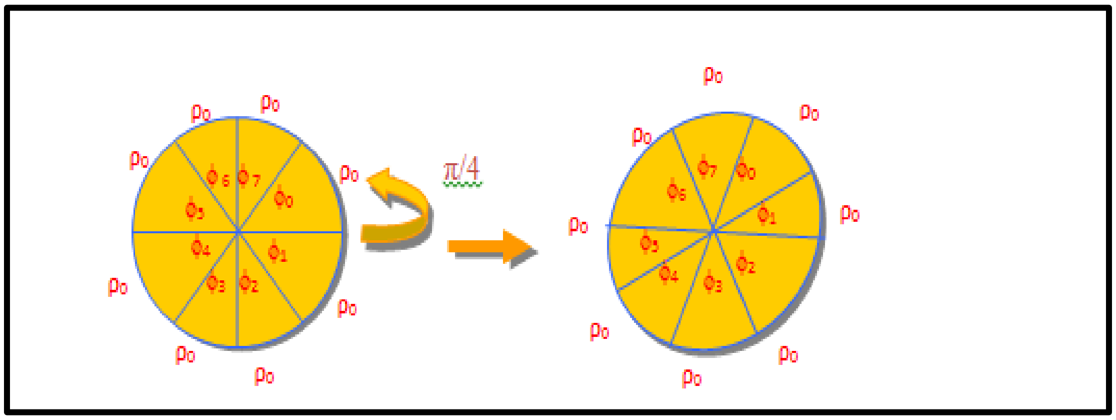


Figure 3.8. Rotation of a circle ($\phi_0 = \pi/4$)

Figure 3.9 explains the rotation by (ϕ_0) on the elements of the first section where $\phi_0 = \pi/4$.

Section$_0$ (8x8) bytes

	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_7	ϕ_0
ρ_0	a_{010}	a_{020}	a_{030}	a_{040}	a_{050}	a_{060}	a_{070}	a_{000}
ρ_1	a_{110}	a_{120}	a_{130}	a_{140}	a_{150}	a_{160}	a_{170}	a_{100}
ρ_2	a_{210}	a_{220}	a_{230}	a_{240}	a_{250}	a_{260}	a_{270}	a_{200}
ρ_3	a_{310}	a_{320}	a_{330}	a_{340}	a_{350}	a_{360}	a_{370}	a_{300}
ρ_4	a_{410}	a_{420}	a_{430}	a_{440}	a_{450}	a_{460}	a_{470}	a_{400}
ρ_5	a_{510}	a_{520}	a_{530}	a_{540}	a_{550}	a_{560}	a_{570}	a_{500}
ρ_6	a_{610}	a_{620}	a_{630}	a_{640}	a_{650}	a_{660}	a_{670}	a_{600}
ρ_7	a_{710}	a_{720}	a_{730}	a_{740}	a_{750}	a_{760}	a_{770}	a_{700}

Figure 3.9. Rotation of the first section ($\phi_0 = \pi/4$)

Figure 3.10 explains the translation by ρ_0 on the elements of the first section where ρ_0 =2.

Section$_0$ (8x8) bytes

	ϕ_0	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_7
ρ_6	a_{600}	a_{610}	a_{620}	a_{630}	a_{640}	a_{650}	a_{660}	a_{670}
ρ_7	a_{700}	a_{710}	a_{720}	a_{730}	a_{740}	a_{750}	a_{760}	a_{770}
ρ_0	a_{000}	a_{010}	a_{020}	a_{030}	a_{040}	a_{050}	a_{060}	a_{070}
ρ_1	a_{100}	a_{110}	a_{120}	a_{130}	a_{140}	a_{150}	a_{160}	a_{170}
ρ_2	a_{200}	a_{210}	a_{220}	a_{230}	a_{240}	a_{250}	a_{260}	a_{270}
ρ_3	a_{300}	a_{310}	a_{320}	a_{330}	a_{340}	a_{350}	a_{360}	a_{370}
ρ_4	a_{400}	a_{410}	a_{420}	a_{430}	a_{440}	a_{450}	a_{460}	a_{470}
ρ_5	a_{500}	a_{510}	a_{520}	a_{530}	a_{540}	a_{550}	a_{560}	a_{570}

Figure 3.10. Translation of the first section (ρ_0 = 2)

In the translation by z_0 on the elements of the section where z_0= 2, the elements of the first and the second sections were translated into the third and the fourth sections, respectively, whereas the elements of the third and the fourth sections was translated into the first and the second sections, respectively.

3.2.2 Dynamic P-Box

The following steps were used to design the dynamic P-Box:

1. Input seed of the mysrand ( ) using the output of the 3D S-Box (8 bytes).
2. Generate 64 random numbers between 0 and 63 from the myrand ( ) and storing these numbers in vector n
3. Initialize the P-Box (vector) with values from 0 to 63.
4. Swap the values of the index P-Box[i] with P-Box[n[i]]

   where i=0...63

Figure 3.11 shows the flowchart of generating the dynamic P-Box. As well as Dynamic P-Box used inside F-Function in RAF as in Figure 3.12.

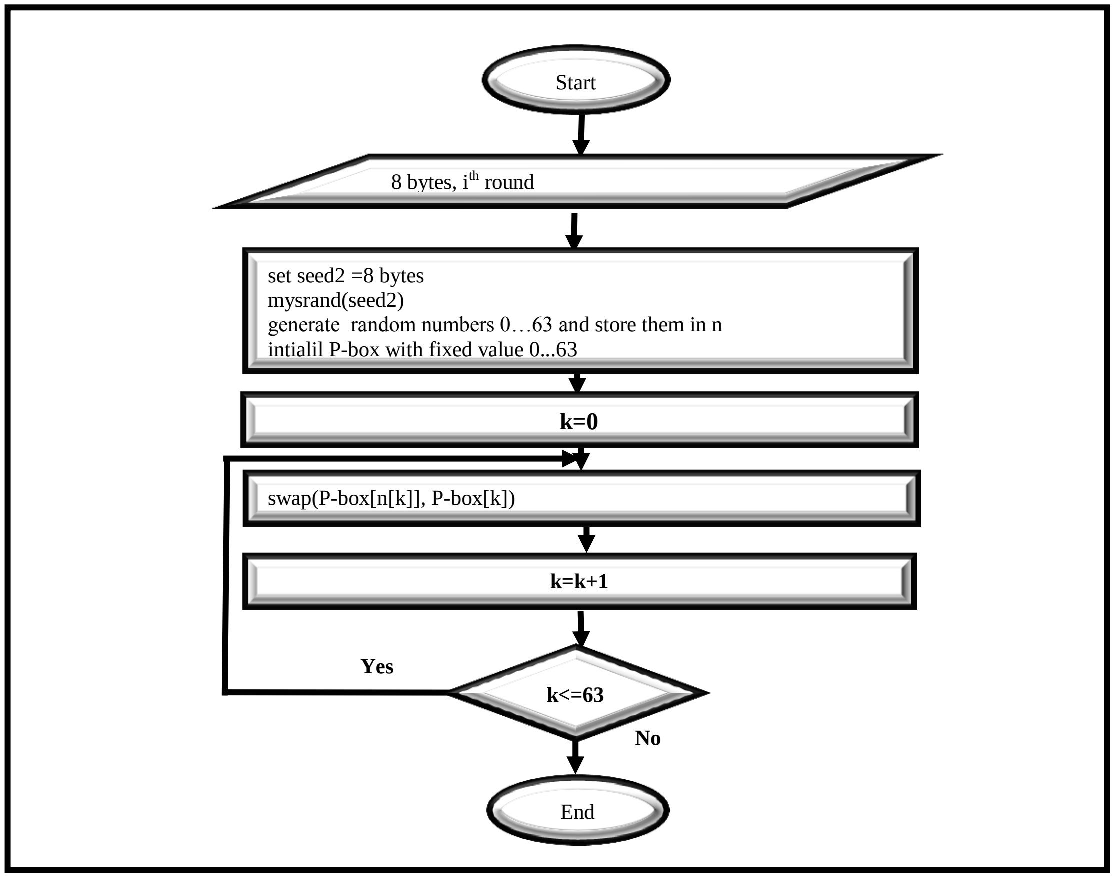

Start
8 bytes, ith round
set seed2 =8 bytes
mysrand(seed2)
generate random numbers 0…63 and store them in n
intialil P-box with fixed value 0...63
k=0
swap(P-box[n[k]], P-box[k])
k=k+1
Yes
k<=63
No
End

Figure 3.11. Flowchart of the dynamic P-Box

3.2.3 Designing a New F-Function

The most complex part is designing the F-Function, known as Coordinate Cylindrical System with Dynamic Permutation box (CCSDPB) in RAF, and which includes a dynamic 3D S-Box and a dynamic P-Box in every round of plaintext block.

In CCSDPB, the XL was divided into four 16-bit quarters with each quarter split into two 8-bit parts. Each 8-bit part, in turn, is further subdivided into three parts which were used as indices to the a_{ijk} , where the first subpart, used as an index to the row number of a_{ijk},is the first three even bits of the byte; the second subpart, used as an index to the column number of a_{ijk}, is the first three odd bits of the byte; and the third subpart, used as an index to the section number, was the last two bits of the byte.

The output of each quarter from 3D S-Box was multiplied by the previous quarter after rotation them, except for the first quarter, and the result was added to the round subkey part. Then the result is modulo 2^{16} which mean that the round subkey part is 16 bits. The multiplication result for the second quarter with first quarter was added to the first 16 bits from the round subkey, whereas that for the multiplication result for third quarter with second quarter was added to the second 16 bits from the round subkey. This procedure was the same for the multiplication result of the fourth quarter with third quarter. The number of rotations (right and left) is differed for each 3D S-Box output (output of quarter) because of the truncated four bits from different positions. If two or more different inputs (quarters) of the same output in the 3D S-Box are present, the resulting output will not be the same. Then the outputs of four quarters are combined together based on the values from dynamic P-Box. Figures

3.12 and 3.13 illustrate the diagram and flowchart of the CCSDPB function respectively. Steps taken for CCSDPB function is describe in algorithm 1.

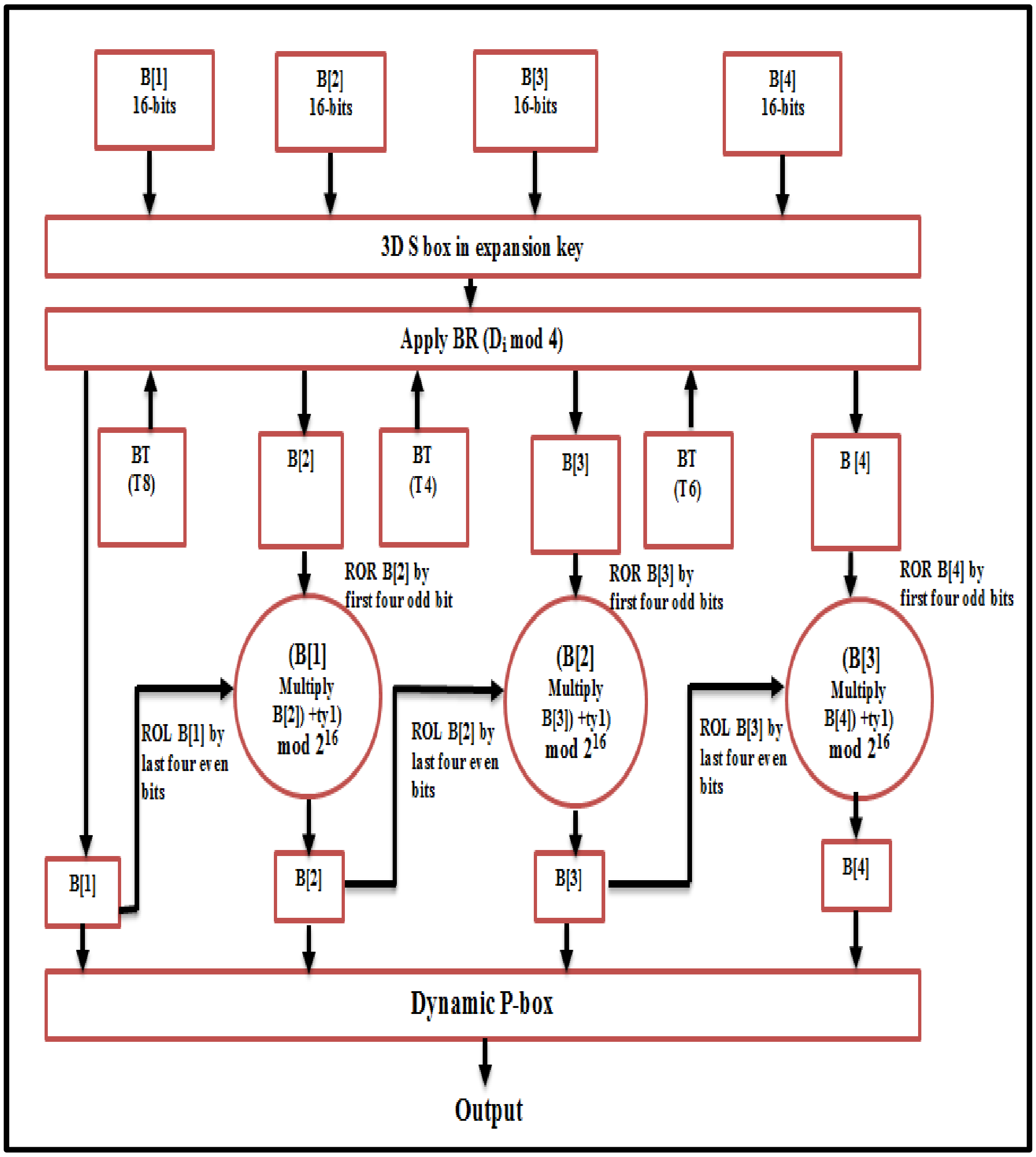

B[1] 16-bits
B[2] 16-bits
B[3] 16-bits
B[4] 16-bits
3D S box in expansion key
Apply BR (D_i mod 4)
BT (T8)
B[2]
BT (T4)
B[3]
BT (T6)
B [4]
ROR B[2] by first four odd bit
ROR B[3] by first four odd bits
ROR B[4] by first four odd bits
(B[1] Multiply B[2]) +ty1) mod 2^16
(B[2] Multiply B[3]) +ty1) mod 2^16
(B[3] Multiply B[4]) +ty1) mod 2^16
ROL B[1] by last four even bits
ROL B[2] by last four even bits
ROL B[3] by last four even bits
B[1]
B[2]
B[3]
B[4]
Dynamic P-box
Output

Figure 3.12. F-Function (CCSDPB)

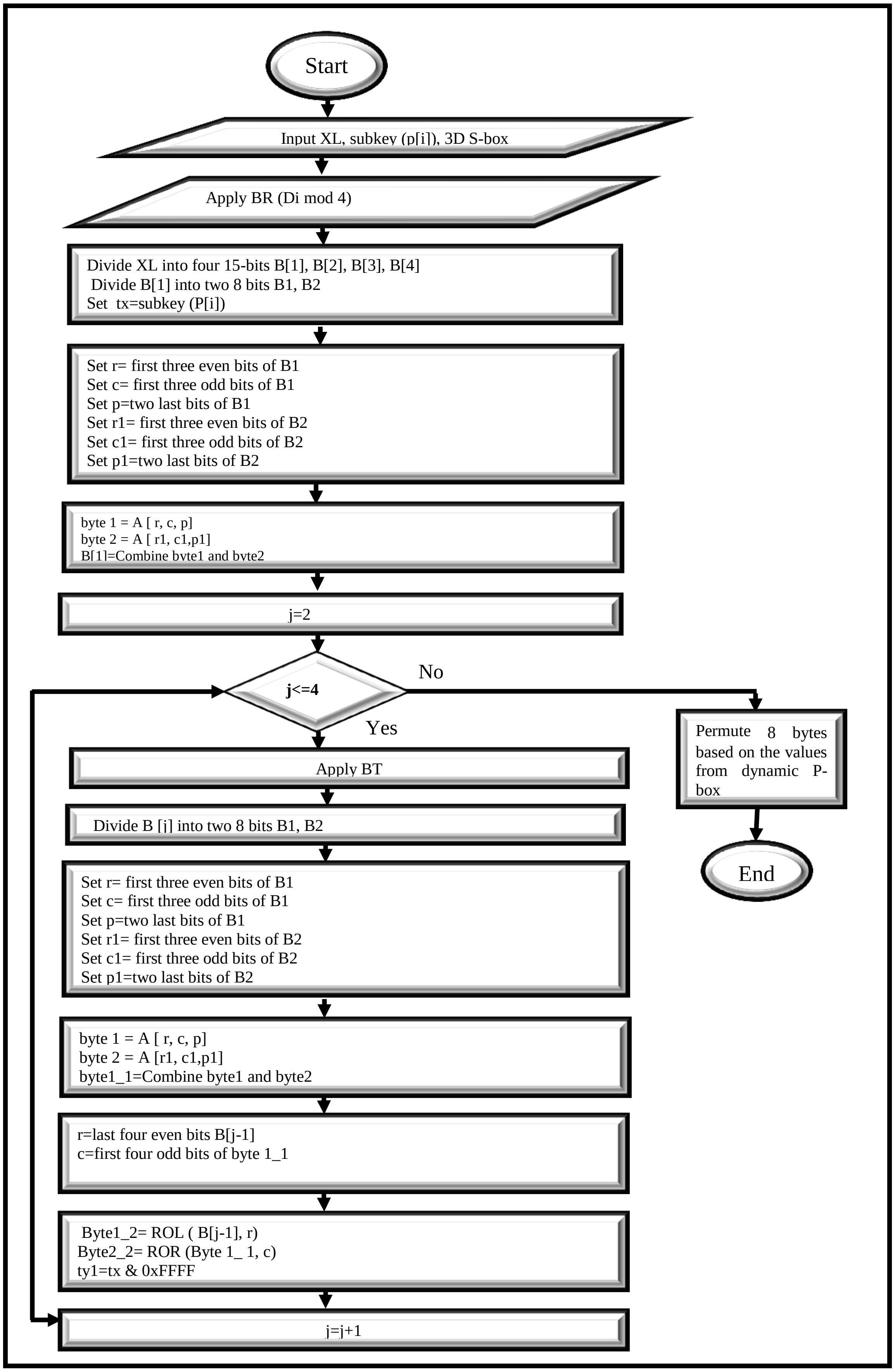

Start
Input XL, subkey (p[i]), 3D S-box
Apply BR (Di mod 4)
Divide XL into four 15-bits B[1], B[2], B[3], B[4]
Divide B[1] into two 8 bits B1, B2
Set tx=subkey (P[i])
Set r= first three even bits of B1
Set c= first three odd bits of B1
Set p=two last bits of B1
Set r1= first three even bits of B2
Set c1= first three odd bits of B2
Set p1=two last bits of B2
byte 1 = A [r, c, p]
byte 2 = A [r1, c1,p1]
B[1]=Combine byte1 and byte2
j=2
j<=4
No
Yes
Permute 8 bytes based on the values from dynamic P-box
End
Apply BT
Divide B [j] into two 8 bits B1, B2
Set r= first three even bits of B1
Set c= first three odd bits of B1
Set p=two last bits of B1
Set r1= first three even bits of B2
Set c1= first three odd bits of B2
Set p1=two last bits of B2
byte 1 = A [r, c, p]
byte 2 = A [r1, c1,p1]
byte1_1=Combine byte1 and byte2
r=last four even bits B[j-1]
c=first four odd bits of byte 1_1
Byte1_2= ROL (B[j-1], r)
Byte2_2= ROR (Byte 1_ 1, c)
ty1=tx & 0xFFFF
j=j+1

Figure 3.13. Flowchart of F-Function (CCSDPB)

Algorithm 1: CCSDPB function

```
Input: XL, subkey round P[i], 3D S-Box   //  XL is the left side of the 64-bit plaintext
Output: $z_1$                                  //   64-bit
    Apply BR (i-th mod 4)
    Divide XL into four 16-bit quarters B [1], B [2], B [3], and B [4], respectively
    Divide B[1] into two 8-bit B1 and B2, respectively
     Set  tx=subkey (P[i])
    Set r=first three even bits of B1
    Set c= first three odd bits of B1
    Set p=two last bits of B1
    Set r1=first three even bits of B2
    Set c1= first three odd bits of B2
    Set p1=two last bits of B2
    byte 1 = A [r, c, p]
    byte 2 = A [r1, c1, p1]
    B[1]= Combine byte 1 and byte 2
    Set j=2
    While j<= 4 do {the number of quarters}
      {
        Apply BT
        Divide B [j] into two 8-bit B1 and B2, respectively
        r=first three even bits of B1
        c= first three odd bits of B1
        p=two last bits of B1
        r1=first three even bits of byte B2
        c1= first three odd bits of byte B2
        p1=two last bits of byte B2
        byte1= A [r, c, p]
        byte2= A [r1, c1, p1]
        Combine byte 1 and byte 2 into byte1_1
        r=last four even bits of B [j-1]
        c= first four odd bits of byte1_1
          Byte1_2= ROL (Byte [j-1], r)   // ROL Rotate Left
          Byte2_2= ROR (Byte1_1, c)    // ROR Rotate Right
          ty1=tx & 0xFFFF                  // 16-bit from P[I]
          tx =tx>>16                         // >>  Shift Right
         B[j] =((Byte1_2*Byte2 _2)+ ty1) mod $2^{16}$
        j=j+1
       }
    Permute the eight bytes based on the values from dynamic P-Box.
```

3.2.4 Comparison the basic information between BA and RAF

This section presents similarity and difference between BA and RAF. Table 3.3 illustrates this compassion.

Table 3.3

Similarity and difference between BA and RAF

Name of the Algorithm	No. of Round	Structure	No. of S-Box	Length of key	Dynamic of S-Box	Block Size	Initialization (key expansion)
BA	16	Feistel network.	Four Dynamic S-Boxes with memory requirement is 4096 bytes	Variable key length from 32 bits to 448 bits	-The values of S-Box are generated in key expansion. -the same S-Box used in every round	64	Used the algorithm itself to generate values to the subkeys and S-Boxes as explained in section 2.2
RAF	10	Feistel network.	One Dynamic 3D S-Box with memory requirement is 256 bytes	Variable key length from 64 bits to 640 bits	-The values of subkeys and dynamic 3D S-Box are generated in the same steps of key expansion in BA. -the values of 3D S-Box are changed in every round with every block(high dynamically)	128	Used the same steps as in BA to generate values of subkeys and dynamic 3D S-Box

3.3 Phase 2 RAF Implementation

The RAF implementation involved two stages: key expansion, and data encryption and decryption. In the key expansion stage, values for the 3D S-Box and the P-array were generated whereas in the data encryption plaintext was encrypted to ciphertext and vice versa for data decryption.

3.3.1 Key Expansion

The key expansion of RAF must be started before the data encryption or data decryption. The key expansion consists of 12 64-bit sub-keys (P [0]…P [11]) and one 3D S-Box values. A variable-length key (640 bits) is converted into several subkey arrays in the key expansion, totaling 352 bytes. The sub-keys are as follows:

1. P-array of 12 64-bit subkeys: P [0], P [1] … P [11].
2. One 3D S-Box (a_{ijk}) with 8-bit entries: i=0...7, j= 0…7, k=0…3.

The same procedure used to generate the subkey values (P-array and S-Box) in the BA is conducted with RAF because this procedure is designed to distribute the set of subkeys randomly throughout the domain of possible subkeys, thereby preserving the entropy of the keys and distributing the entropy uniformly throughout the subkeys (Schneier, 1994; Hashim et al., 2009; Mahdi, 2009).

3.3.2 Perform Data Encryption and Data Decryption

The plaintext was encrypted to ciphertext in the data encryption stage, whereas the ciphertext was decrypted to plaintext in the data decryption stage. The encryption algorithm consists of 128-bit data of plaintext and gives an output of 128-bit ciphertext with variable key up to 640 bits. The external structure of the RAF is the same as that of the BA. The input 128-bit data (X) is divided into two 64-bit halves: XL and XR. The CCSDPB function is the main RAF component. The encryption architecture is shown in Figure 3.14 and the encryption steps are enumerated below:

```
Algorithm 2:   RAF Data Encryption

Input:     plaintext 128 bits (X)
Output:  ciphertext 128 bits
           For i = 0 to 9
               {
                  XL= XL XOR P[i]
                  XR= CCSDPB (XL) XOR XR
                  Swap XL and XR
                }
           Swap XL and XR (Undo the last swap)
           XR = XR XOR P [10]
           XL = XL XOR P [11]
           Recombine XL and XR
```

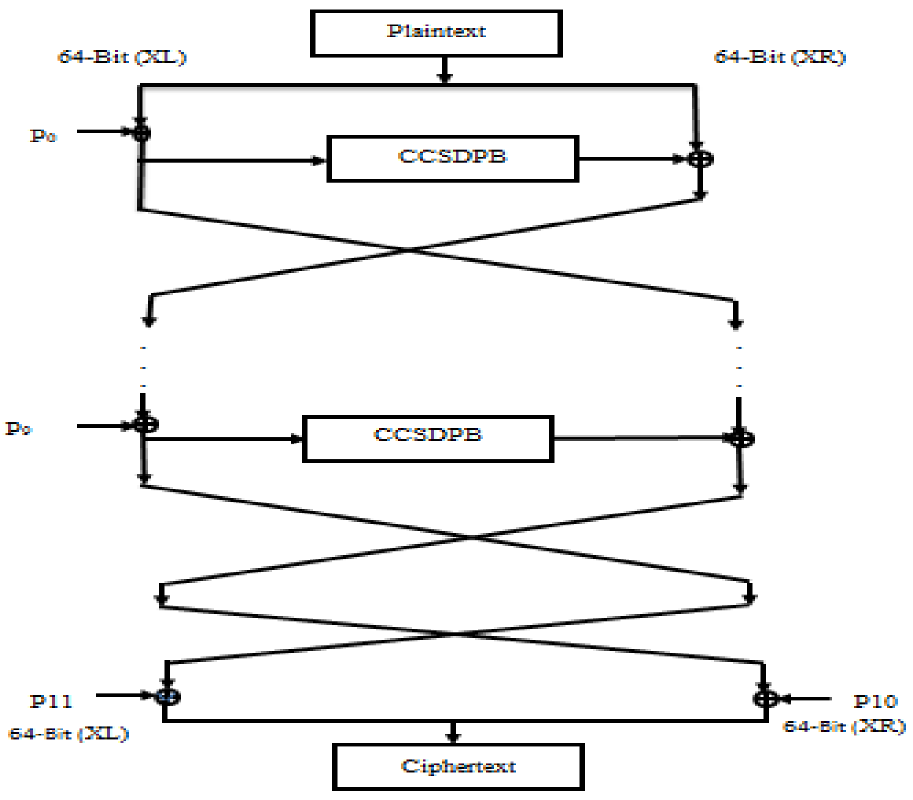


Figure 3.14. Data Encryption RAF

Encryption and decryption are structurally identical in a Feistel cipher. The subkeys used during encryption in every round are reused during decryption but in reverse order. Hence, the steps taken to perform the RAF decryption of are similar to that of the RAF encryption. Figure 3.15 illustrates the architecture of the 10-round RAF data decryption. The input is a 128-bit ciphertext divided into two halves, XL and XR, each with 64 bits. Steps taken for RAF data decryption of RAF are described below.

```
Algorithm 3:   Data Decryption RAF
Input:     ciphertext 128 bits (X)
Output:    plaintext 128 bits
           XL = XL XOR P[11]
            XR = XR XOR P[10]
            For i= 9 to 0
            {
              XR= CCSDPB(XL) XOR XR
               XL= XL XOR P[i]
               Swap XL and XR
             }
           Swap  XL and XR
           Recombine XL and XR to get plaintext
```

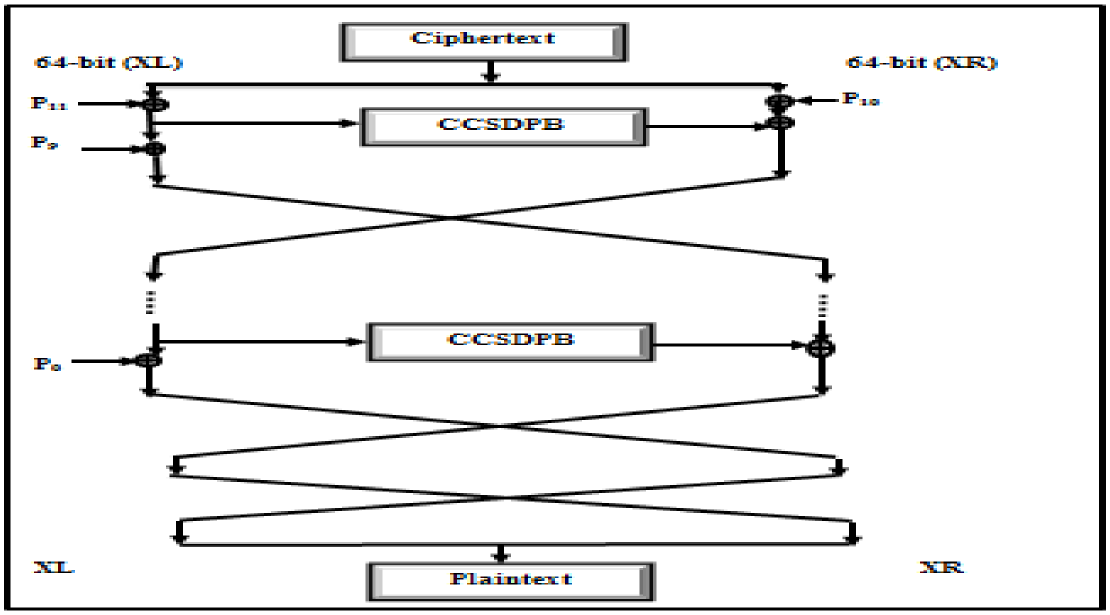


Figure 3.15. Data decryption RAF Architecture

C++ was used in this study to implement the proposed RAF because of its popularity as a programming language and its wide application in different software platforms (Verma et al., 2011; Sulong, 2008). The complete program was run under Windows 7 operating system.

3.4 Phase 3 RAF Verification

The verification takes place in three stages: Stage 1 Verification of the 3D S-Box, Stage 2 Verification of the RAF output, and Stage 3 Comparison of RAF with other cryptographic algorithms.

3.4.1 Stage 1 Verification of 3D S-Box

The first stage involves the 3D S-Box verification to ensure that all the components work correctly. 210 experiments were conducted in the first stage.

First, 180 experiments were conducted under two different key types (random encryption key, nonrandom encryption keys that includes low entropy (ones and zeroes) using the three S-Box criteria (AVAL, SAC, and BIC) in both BA and RAF algorithms. Experiments were carried out in each of the four BA S-Boxes. Figure 3.16 below shows the 3D S-Box verification flow.

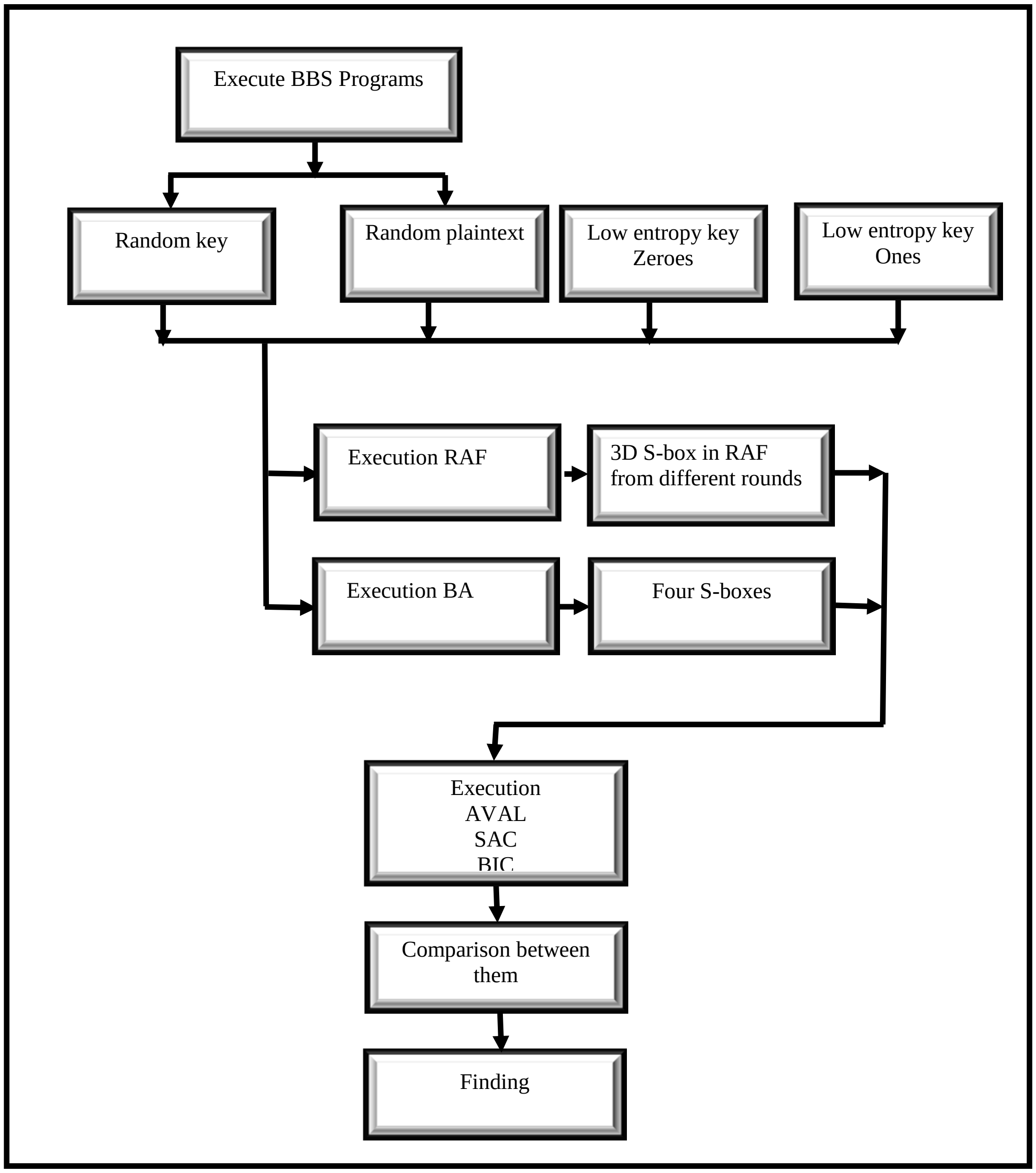


Figure 3.16. 3D S-Box verification flow

Second, 30 experiments were conducted on the correlation coefficient between 3D S-Boxes under different key types (correlated and uncorrelated) with two types of plaintext (random plaintext and nonrandom plaintext, the latter both low-entropy ones and zeroes).

The first 15 experiments depended on uncorrelated random Eks divided into three of five experiments. The first five experiments were conducted using random plaintext, the second five experiments using nonrandom high-entropy plaintext, and the final five experiments using nonrandom low-entropy plaintext.

The second set of 15 experiments was the same as the first but conducted with correlated Eks. All the experiments for this stage were implemented using MATLAB (Mathworks R2012a). Figure 3.17 describes the execution flow of the experiments on the correlation coefficient.

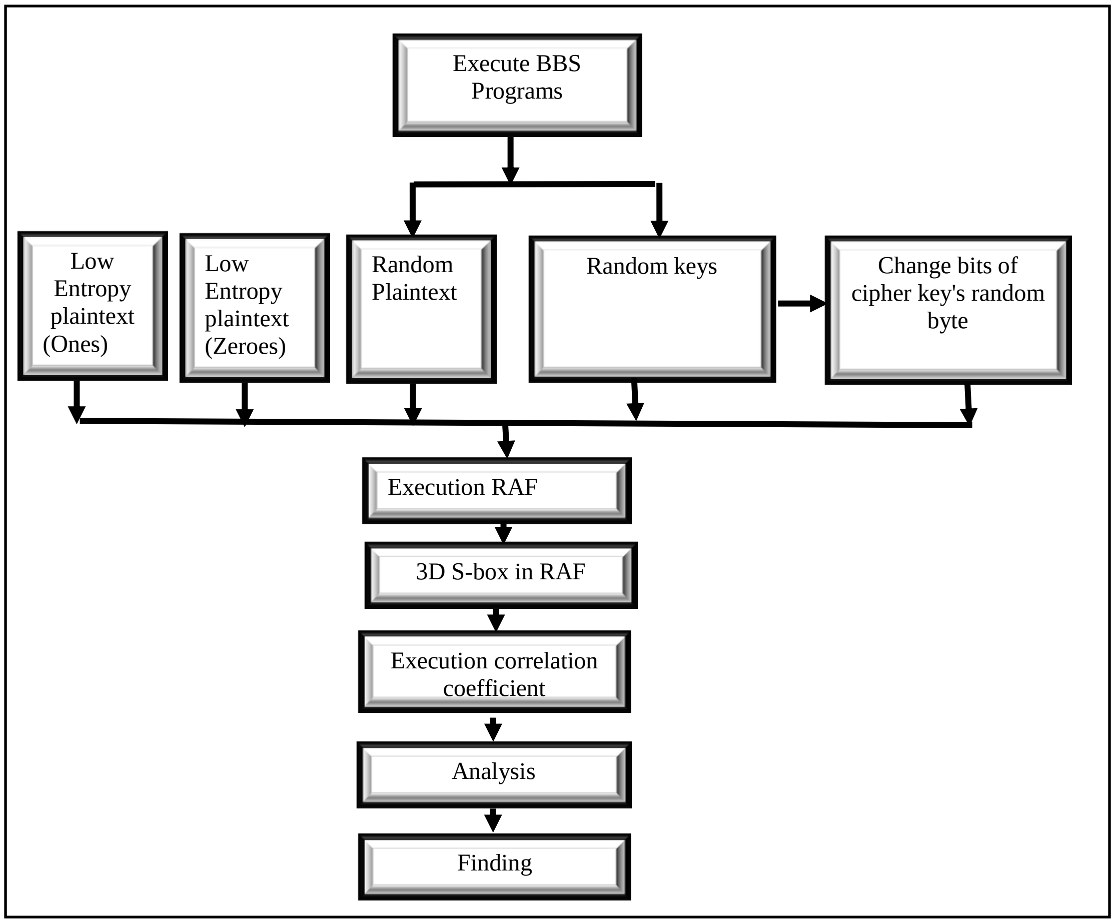

Execute BBS Programs
Low Entropy plaintext (Ones)
Low Entropy plaintext (Zeroes)
Random Plaintext
Random keys
Change bits of cipher key's random byte
Execution RAF
3D S-box in RAF
Execution correlation coefficient
Analysis
Finding

Figure 3.17. Flow analysis of correlation coefficient of 3D S-Box in RAF

The test data was generated using the BBS generator which generates random 128-bit and 256-bit Eks as well as random 128-bit plaintext.

3.4.2 Stage 2   Verification of RAF Output

The second stage includes the verification of the RAF output. This stage consists of three parts: Part 1 which evaluates the RAF output using NIST statistical tests on five data types; Part 2 which evaluates the RAF output using avalanche text and correlation coefficient; Part 3 which evaluates the RAF resistance to cryptanalysis; and part 4 evaluation of the computation efficiency of RAF. Details of each part are presented in the following sections.

3.4.2.1 Part 1 Evaluation of RAF Output Using NIST Statistical Tests

For the first part of the experiment, the output of RAF rounds were verified using 15 NIST statistical tests on different data types and samples. The data types are as follows: cipher block chaining mode; random plaintext/random 128-bit keys; image in different formats; text; and video files. Each type of data included a sample size of 128 sequences except for the cipher block chaining mode where the sample size was 300 sequences. Each sequence was stored in one file.

The cipher block chaining mode and the random plaintext/random 128-bit keys have been used in selecting the finalists for the AES block cipher (Soto, 1999b; Soto & Bassham, 2000; Isa & Z'aba, 2012; Ariffin, 2012). The same data and sample size as used previously were used in this study. Moreover, the other three types of testing data (image, text, and video) have been used in BA evaluation, as reported by Mousa (2005), Meyers and Desoky (2008), Elminaam et al. (2010), Singh et al. (2011), Chandrasekaran et al. (2011), Mandal (2012), and Alabaichi et al. (2013a, 2013b, 2013e, 2013f). The same data was likewise used in this study.

The evaluation in this part includes Partial Round Testing (PRT) and Full Round Testing (FRT). Soto and Bassham (2000) tested Twofish rounds in pairs. Because Twofish is a Feistel network, each round leaves some of the data bits unchanged, thereby making the Twofish appear nonrandom under test conditions after one round. However, all data bits are affected after two rounds, hence the evaluation of paired Twofish rounds, i.e., even numbered rounds from two to fourteen. Therefore, in this study, the PRT with all five testing data types were conducted in pairs: from two to fourteen rounds for BA and from two to eight rounds for RAF algorithm. In a FRT, the output (ciphertext) is tested of all types of data in both algorithms (Ali, 2005; Ali, 2009; Isa & Z'aba, 2012).

Description of data types

The five data types used are further explained below.

• Cipher Block Chaining Mode

A binary sequence of 1,048,576 bits was constructed using ciphertext computed in the cipher block chaining (CBC) given a random 128-bit key, a 128-bit initialization vector (*IV*) of all zeroes, and a 128-bit plaintext block (*PT*) of all zeroes. This binary sequence consisted of 8,192 concatenated 128-bit ciphertext blocks. The first ciphertext block (CT_1) was defined by $CT_1 = Ek\ (IV \oplus PT)$ whereas subsequent ciphertext blocks were defined by $CT_{i+1} = Ek\ (CT_i \oplus PT)$ for $1 \le i \le 8{,}191$.

Altogether, the constructed binary sequences totaled to 300, each with a different random 128-bit key. The described data were used for both RAF and BA algorithms. However, 16,384 64-bit blocks were used in BA algorithm instead of 8,192. The proportion of sequences that passed a specific statistical test is equal or greater than $p\alpha$, as defined in Equation 2.12. In this type of data, the proportion value is

$$p\alpha = (1-0.01) - 3\sqrt{\frac{0.01(1-0.01)}{300}} = 0.972766312$$

while the proportion value of reminder of four types of testing data is

$$p\alpha(1-0.01) - 3\sqrt{\frac{0.01(1-0.01)}{128}} = 0.963616$$

• **Random Plaintext/Random 128-bit keys**

The Blum-Blum-Shub (BBS) pseudorandom bit generator is a cryptographically secure generator for random plaintext/random 128-bit keys (Menezes et al., 1997). 128 sequences were constructed to examine the randomness of the ciphertext based on random plaintext and random 128-bit keys. Each sequence resulted from the concatenation of 8,128 128-bit ciphertext blocks (1,040,384 bits) using 8,128 random plaintext blocks of the same length and a random 128-bit key. The data in this sequence were used for both RAF and BA algorithms as testing data. For BA, however, 16,256 64-bit blocks were used instead of 8,128.

• **Image file**

Image compression addresses the problem of reduction the amount of data required to represent a digital image. The underlying basis of the reduction process is the removal of redundant data. The compressed image can be broadly classified in two categories lossless and lossy (Acharya & Ray, 2005).

Lossless compression methods find repetitive patterns of values in the file and replace them with codes. Which further more occur patterns that are often repeated in the file. These methods produce compressed files that can be used to exactly reconstruct the original.

Lossy compression method is to achieve significantly greater amounts of compression by discarding some of the information. The intent is always to discard those details that are less important to the human visual interpretation and recognition of the image contents. Lossy compression must not be applied if the images are to be subsequently used for any measurements or other legal or medical. As well as in digital radiography, where loss of information can compromise diagnostic accuracy.in these cases and other cases, the need for lossless compression is motivated. The amount of data reduction possible using lossy compression can often be much more substantial than what is possible with lossless data compression techniques (Russ & Russ, 2007; Gonzales & Woods, 2002).

There are a number of file formats. These are known as image file format standards. The most popular image file format standard is:

- **JPEG (Joint Photographic Expert Group)**

JPEG file format was developed by a professional photographer. JPEG compression is built into the firmware of many cameras to reduce storage requirements. It is a commonly used method of lossy compression for digital images. JPEG typically achieves 10:1 compression with little perceptible loss in image quality (Russ & Russ, 2007).

• **GIF (Graphics Interchange Format)**

This format supports 8-bit color palette images and is not very popular among the image processing researchers. It used lossless compression.

• **BMP (Windows bitmap)**

This format handles graphics files within the Microsoft Windows OS. Typically, BMP files are uncompressed.

• **Png (Portable Network Graphics)**

This is an extensible file format that provides lossless compression. This is simple format covers the major functionalities of .tiff. Gray scale, color palette, and true color images are support by this file format.

• **Tiff (Tagged Image Format)**

The format is very broad format, it has two types of compression: the first it which can handle anything from bitmap to compressed color palette image. While the second type of compression includes lossless compression that depends on DPCM and Run Length (Acharya & Ray, 2005).

In this study used were 128 image files in different file formats (jpeg, gif, bmp, png, tiff). It includes uncompressed and compressed files. The compressed files consist of lossless and lossy compress files were .jpeg, .gif, .bmp, .png, and .tiff. Each file contained one sequence resulted from the concatenation of 12,290 (1,573,120 bits) 128-bit ciphertext blocks using 12,290 128-bit plaintext blocks and a random 256-bit key. The data in this sequence were used for both RAF and BA algorithms as testing data. However for BA, 24,580 64-bit blocks were used instead of 12,290.

• **Text files**

The data set consist of 128 text files (.txt). Each file contained one sequence resulted from the concatenation of 8,128 (1,040,384 bits) 128-bit ciphertext blocks using 8,128 128-bit plaintext blocks and a random 256-bit key. The data in this sequence were used for both RAF and BA algorithms as testing data. For BA, however, 16,256 64-bit blocks were used instead of instead of 8,128.

• **Video files**

The data set consist of 128 sequences of video files (.flv, .wmv). Each file contained one sequence resulted from the concatenation of 8,128 (1,040,384 bits) 128-bit ciphertext blocks using 8,128 128-bit plaintext blocks and a random 256-bit key. The data in this sequence were used for both RAF and BA algorithms as testing data. As with other files, 16,256 64-bit blocks were used for BA instead of 8,128. The experimental flow is shown in Figure 3.18 below.

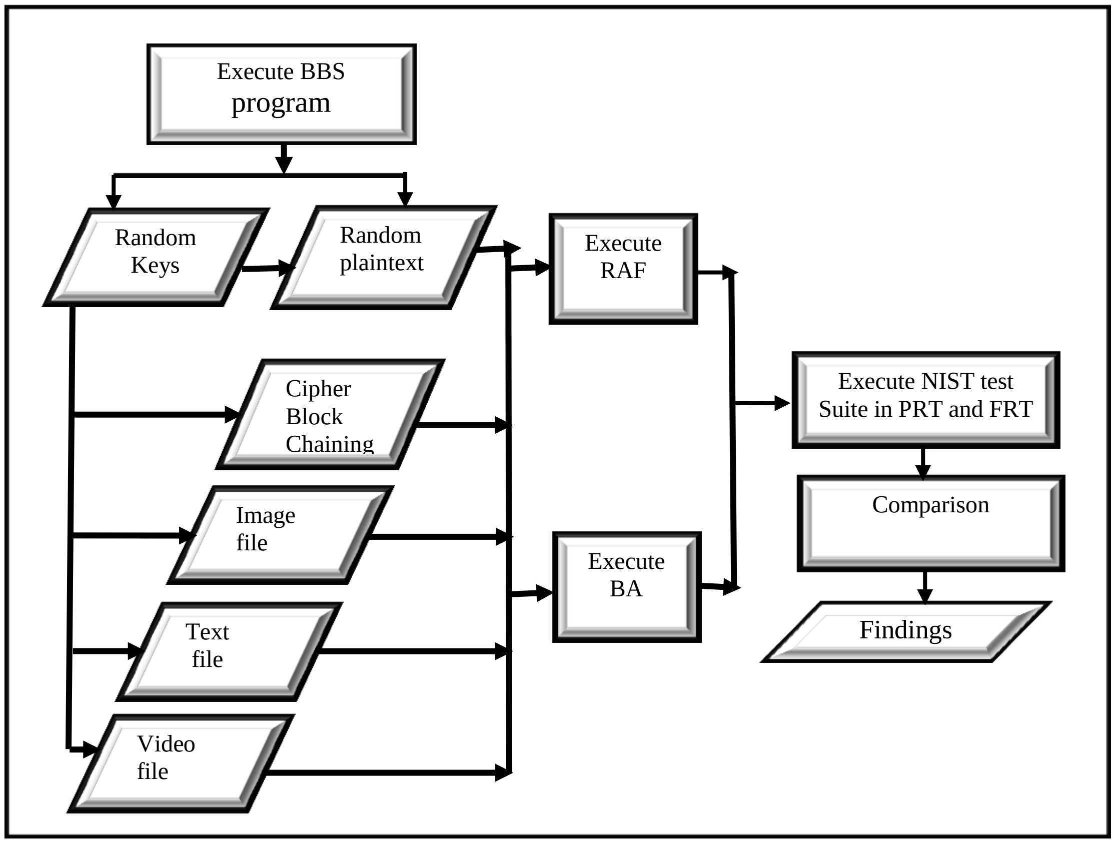


Figure 3.18. NIST experimental flow

3.4.2.2 Part 2 Evaluation of RAF Output Using Avalanche Text and Correlation Coefficient

The second part of the experiment included verifying the RAF output using the avalanche text and the correlation coefficient. The Blum-Blum-Shub (BBS) generator produced 128 files of random data using Java programming language (NetBeans IDE 7.2). The data consisted of 128 128-bit sequences with a128-bit random keys. Every sequence is stored in one file. The random 128 64-bit sequences were used to test the BA. Figure 3.19 illustrates the experimental flow on avalanche text.

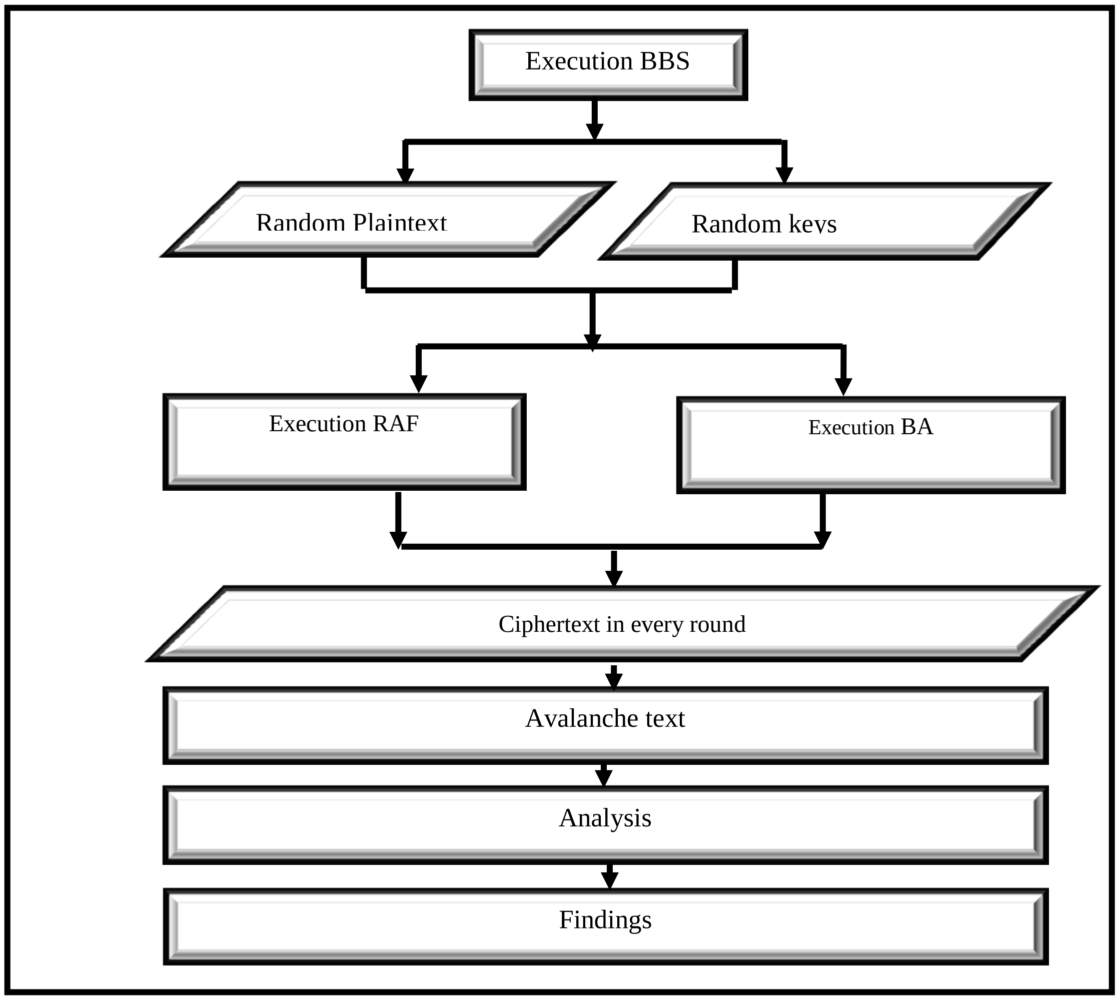


Figure 3.19. Experimental flow on avalanche text

3.4.2.3 Part 3 Evalution of RAF Resistance to Crypytanalysis

The third part evaluated the RAF resistance to 3 cryptanalysis; differential, linear, and short attacks.

3.4.2.4 Part 4 Evaluation of Computational Efficiency of RAF

The fourth part evaluated the computational efficiency RAF.

3.4.3 Stage 3 Comparison of RAF with other cryptographic algorithms

The third stage includes comparing RAF with Mars, RC6, Rijndael, Serpent, and Twofish in terms of randomness. These algorithms were selected because they are strong and have been finalists for the Advance Encryption Standard (AES) in terms of randomness. One dataset, Low Density Plaintext (Soto and Bassham, 20000) was used for the comparison. The descripttion of the dataset is given below.

- **Low Density Plaintext**

The dataset contains 128 Low Density Plaintext sequences. Each sequence consists of 8257 ciphertext blocks. With each sequence, a random distinct 256-bit key was used. The first ciphertext block was calculated using an all zero plaintext block. Ciphertext blocks 2-129 were calculated using plaintext blocks consisting of a single one and 127 zeroes, the one appearing in each of the 128 bits positions of the plaintext block. Ciphertext blocks 130-8257 were calculated using plaintext blocks consisting of two ones and 126 zeroes, the ones appearing in each combination of two bit positions of the plaintext block.

3.5 Summary

This chapter explains the methodology for the study. The methodology consists of three phases: RAF design, RAF implementation, and RAF evaluation. RAF design consists of designing dynamic 3D S-Box, design dynamic P-Box, and design new F-Function (CCSDPB). RAF implementation includes two stages key expansion, and data encryption and decryption. While RAF evaluation takes place in three stages, verification of the 3D S-box, verification of the RAF output, and comparison of RAF with other cryptographic algorithms.

CHAPTER FOUR

RESULTS

4.1 Introduction

This chapter presents the study results according to phases described in Chapter Three. The results of Phases 1 and 2 are shown in Chapter Four, whereas the results of Phase 3 are shown in Chapters Five and Six.

4.2 Phase 1 RAF Design

This section shows the results of the dynamic 3D S-Box, the dynamic P-Box, and the CCSDPB.

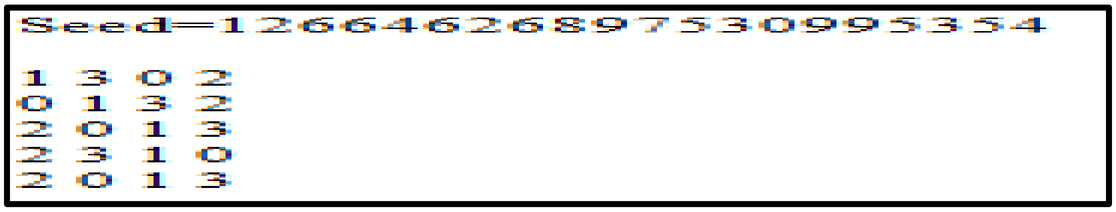
4.2.1 Dynamic 3D S-Box

The deliverables for this step are the generated random secret keys and the algorithms used to perform BR and BT.

4.2.1.1 Random Secret Keys

In this section, only the random secret keys for Round 0 are shown. The generated random secret keys are shown in Figure 4.1.

Round 0



```
Seed=12664626897530995354

1 3 0 2
0 1 3 2
2 0 1 3
2 3 1 0
2 0 1 3
```

Figure 4.1. Random secret keys in round 0

Figure 4.1 illustrates five sets of Random secret keys that are generated from myrand ( ) function. The seed of myrand ( ) function in this round is 12664626897530995354.

4.2.1.2 Algorithms of Byte Relocation and Byte Transformation

The algorithms conducted to perform BR and BT are Algorithms 4 and 5, respectively.

Algorithm 4: Byte Relocation

```
Input: section$_0$, sectionn$_1$, section$_2$, section$_3$ of the key expansion part of the
algorithm, five sets of random (SKs).
Output: Section$_0$, Section$_1$, Section$_2$, Section$_3$ after applying BR.
        Divide each section into four quarters with D (i-th round mod 4).
        Select two sections depending on the first SKs set.
        Swap between the elements of the quarters in the selected section depending
         on the last four SKs sets.
        Swap between the elements of the main diagonals with the elements of the
         second diagonals in the sections with D$_0$ only.
            Swap (L$_0$, L$_1$)
            Swap (L$_2$, L$_3$)
            Swap (S$_0$, S$_2$)
            Swap (S$_1$, S$_3$)
// L$_0$, L$_1$, L$_2$, L$_3$, S$_0$, S$_1$, S$_2$, and S$_3$ are the main and the second diagonals of the
  sections, respectively//
```

Algorithm 5: Byte Transformation

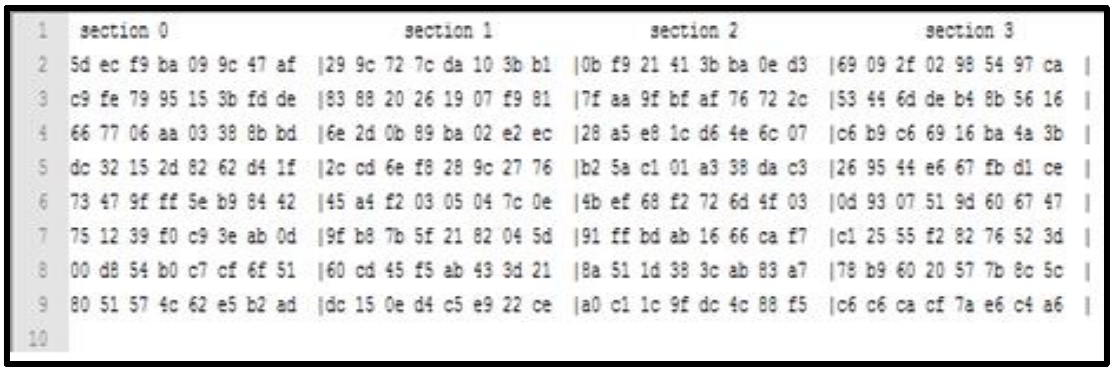
```
Input: dynamic 3D S-Box from BR, $\rho_0, z_0, \emptyset_0$
Output: dynamic 3D S-Box after apply BT
         Apply T8 on the sections of dynamic 3D S-Box
         Apply T4 on the sections of dynamic 3D S-Box
         Apply T6 on the sections of dynamic 3D S-Box
```

The results of executing both the BR and the BT algorithms are shown below:

The input and the output are shown for the first round as follows:-

- **BR**

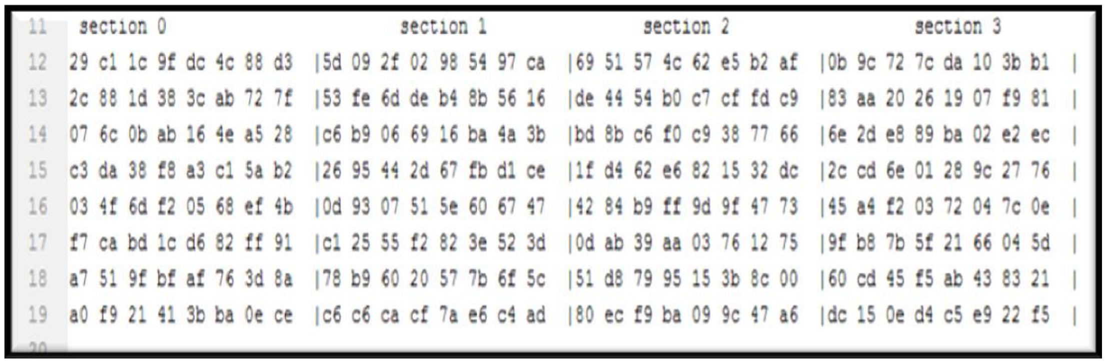
Input: dynamic 3D S-Box from key expansion part.



```
section 0                                section 1                    section 2                    section 3
5d ec f9 ba 09 9c 47 af  |29 9c 72 7c da 10 3b b1  |0b f9 21 41 3b ba 0e d3  |69 09 2f 02 98 54 97 ca  |
c9 fe 79 95 15 3b fd de  |83 88 20 26 19 07 f9 81  |7f aa 9f bf af 76 72 2c  |53 44 6d de b4 8b 56 16  |
66 77 06 aa 03 38 8b bd  |6e 2d 0b 89 ba 02 e2 ec  |28 a5 e8 1c d6 4e 6c 07  |c6 b9 c6 69 16 ba 4a 3b  |
dc 32 15 2d 82 62 d4 1f  |2c cd 6e f8 28 9c 27 76  |b2 5a c1 01 a3 38 da c3  |26 95 44 e6 67 fb d1 ce  |
73 47 9f ff 5e b9 84 42  |45 a4 f2 03 05 04 7c 0e  |4b ef 68 f2 72 6d 4f 03  |0d 93 07 51 9d 60 67 47  |
75 12 39 f0 c9 3e ab 0d  |9f b8 7b 5f 21 82 04 5d  |91 ff bd ab 16 66 ca f7  |c1 25 55 f2 82 76 52 3d  |
00 d8 54 b0 c7 cf 6f 51  |60 cd 45 f5 ab 43 3d 21  |8a 51 1d 38 3c ab 83 a7  |78 b9 60 20 57 7b 8c 5c  |
80 51 57 4c 62 e5 b2 ad  |dc 15 0e d4 c5 e9 22 ce  |a0 c1 1c 9f dc 4c 88 f5  |c6 c6 ca cf 7a e6 c4 a6  |
```

Figure 4.2. Dynamic 3D S-Box from key expansion part

Output: dynamic 3D S-Box after apply BR (D0)



```
section 0                                section 1                    section 2                    section 3
29 c1 1c 9f dc 4c 88 d3  |5d 09 2f 02 98 54 97 ca  |69 51 57 4c 62 e5 b2 af  |0b 9c 72 7c da 10 3b b1  |
2c 88 1d 38 3c ab 72 7f  |53 fe 6d de b4 8b 56 16  |de 44 54 b0 c7 cf fd c9  |83 aa 20 26 19 07 f9 81  |
07 6c 0b ab 16 4e a5 28  |c6 b9 06 69 16 ba 4a 3b  |bd 8b c6 f0 c9 38 77 66  |6e 2d e8 89 ba 02 e2 ec  |
c3 da 38 f8 a3 c1 5a b2  |26 95 44 2d 67 fb d1 ce  |1f d4 62 e6 82 15 32 dc  |2c cd 6e 01 28 9c 27 76  |
03 4f 6d f2 05 68 ef 4b  |0d 93 07 51 5e 60 67 47  |42 84 b9 ff 9d 9f 47 73  |45 a4 f2 03 72 04 7c 0e  |
f7 ca bd 1c d6 82 ff 91  |c1 25 55 f2 82 3e 52 3d  |0d ab 39 aa 03 76 12 75  |9f b8 7b 5f 21 66 04 5d  |
a7 51 9f bf af 76 3d 8a  |78 b9 60 20 57 7b 6f 5c  |51 d8 79 95 15 3b 8c 00  |60 cd 45 f5 ab 43 83 21  |
a0 f9 21 41 3b ba 0e ce  |c6 c6 ca cf 7a e6 c4 ad  |80 ec f9 ba 09 9c 47 a6  |dc 15 0e d4 c5 e9 22 f5  |
```

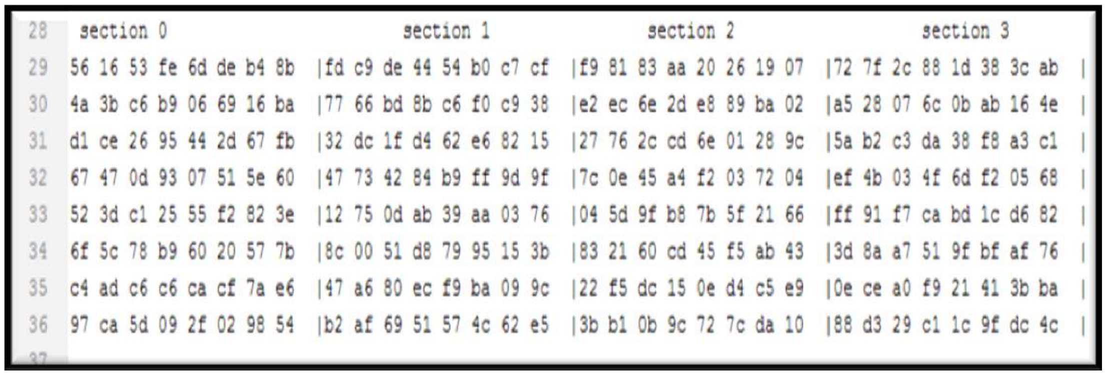
Figure 4.3. Dynamic 3D S-Box after BR (D0) in Round 0

The above Figure 4.2 and Figure 4.3 illustrate the input and output from BR (D0) in Round 0. Four sections are presented Section 0, Section 1, Section 2, and Section 3 respectively. Every position in Sections contains one byte in hexadecimal.

- **BT**

The results from BT are as follows:

--------------------------------T8 => $\rho_0 \neq 0$, $\phi_0 \neq 0$, $z_{0\neq}0$ => $\rho_0=7$, $\phi_0=2$, $z_0=3$-------------------------------------

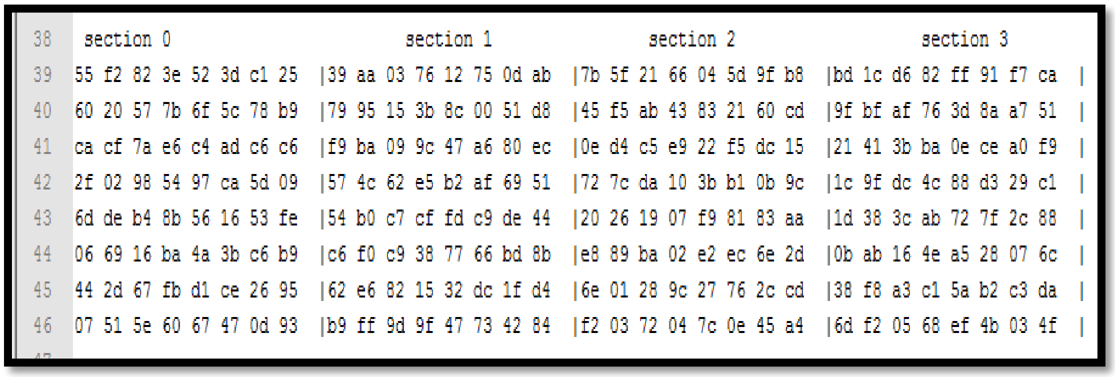


```
 section 0                        section 1                section 2                section 3
56 16 53 fe 6d de b4 8b  |fd c9 de 44 54 b0 c7 cf  |f9 81 83 aa 20 26 19 07  |72 7f 2c 88 1d 38 3c ab  |
4a 3b c6 b9 06 69 16 ba  |77 66 bd 8b c6 f0 c9 38  |e2 ec 6e 2d e8 89 ba 02  |a5 28 07 6c 0b ab 16 4e  |
d1 ce 26 95 44 2d 67 fb  |32 dc 1f d4 62 e6 82 15  |27 76 2c cd 6e 01 28 9c  |5a b2 c3 da 38 f8 a3 c1  |
67 47 0d 93 07 51 5e 60  |47 73 42 84 b9 ff 9d 9f  |7c 0e 45 a4 f2 03 72 04  |ef 4b 03 4f 6d f2 05 68  |
52 3d c1 25 55 f2 82 3e  |12 75 0d ab 39 aa 03 76  |04 5d 9f b8 7b 5f 21 66  |ff 91 f7 ca bd 1c d6 82  |
6f 5c 78 b9 60 20 57 7b  |8c 00 51 d8 79 95 15 3b  |83 21 60 cd 45 f5 ab 43  |3d 8a a7 51 9f bf af 76  |
c4 ad c6 c6 ca cf 7a e6  |47 a6 80 ec f9 ba 09 9c  |22 f5 dc 15 0e d4 c5 e9  |0e ce a0 f9 21 41 3b ba  |
97 ca 5d 09 2f 02 98 54  |b2 af 69 51 57 4c 62 e5  |3b b1 0b 9c 72 7c da 10  |88 d3 29 c1 1c 9f dc 4c  |
```

Figure 4.4. Dynamic 3D S-Box from BT (T8) in Round 0

The above Figure 4.4 illustrates the output from BT (T8) in Round 0. In this step the input to BT is the output from previous step (3D in Figure 4.3). Every rows in dynamic 3D S-Box in Figure 4.3 is translated by increment $\rho_0=7$, and every section is translated by increment $z_0=3$ as well as every column is rotated by increment $\phi_0=2$ which is means rotate by $\phi_0=90$.

-----------------------------T4 => $\rho_0 \neq 0$, $\phi_{0\neq}0$, $z_{0=}0$ => $\rho_0=4$, $\phi_0=4$--

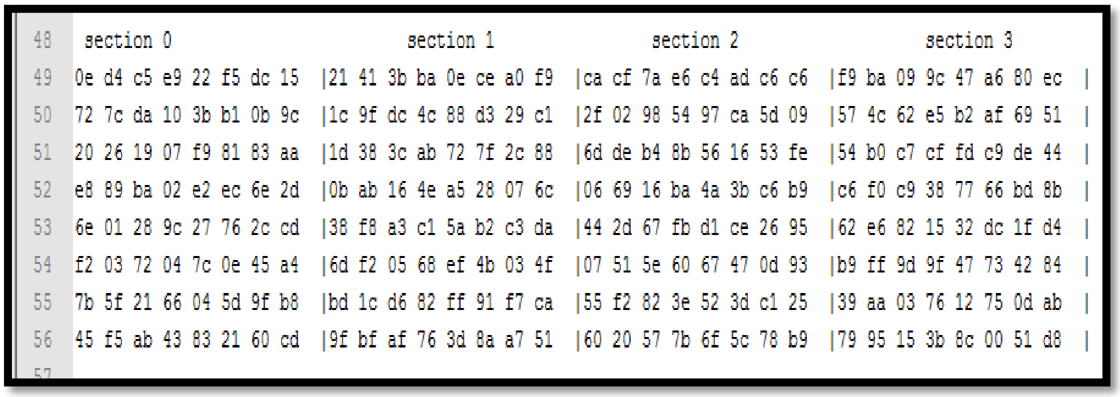


```
 section 0                        section 1                section 2                section 3
55 f2 82 3e 52 3d c1 25  |39 aa 03 76 12 75 0d ab  |7b 5f 21 66 04 5d 9f b8  |bd 1c d6 82 ff 91 f7 ca  |
60 20 57 7b 6f 5c 78 b9  |79 95 15 3b 8c 00 51 d8  |45 f5 ab 43 83 21 60 cd  |9f bf af 76 3d 8a a7 51  |
ca cf 7a e6 c4 ad c6 c6  |f9 ba 09 9c 47 a6 80 ec  |0e d4 c5 e9 22 f5 dc 15  |21 41 3b ba 0e ce a0 f9  |
2f 02 98 54 97 ca 5d 09  |57 4c 62 e5 b2 af 69 51  |72 7c da 10 3b b1 0b 9c  |1c 9f dc 4c 88 d3 29 c1  |
6d de b4 8b 56 16 53 fe  |54 b0 c7 cf fd c9 de 44  |20 26 19 07 f9 81 83 aa  |1d 38 3c ab 72 7f 2c 88  |
06 69 16 ba 4a 3b c6 b9  |c6 f0 c9 38 77 66 bd 8b  |e8 89 ba 02 e2 ec 6e 2d  |0b ab 16 4e a5 28 07 6c  |
44 2d 67 fb d1 ce 26 95  |62 e6 82 15 32 dc 1f d4  |6e 01 28 9c 27 76 2c cd  |38 f8 a3 c1 5a b2 c3 da  |
07 51 5e 60 67 47 0d 93  |b9 ff 9d 9f 47 73 42 84  |f2 03 72 04 7c 0e 45 a4  |6d f2 05 68 ef 4b 03 4f  |
```

Figure 4.5. Dynamic 3D S-Box from BT (T4) in Round 0

The above Figure 4.5 illustrates the output from BT (T4) in round 0. In this step the input to BT is the output from previous step (3D in Figure 4.4). Every rows in dynamic 3D S-Box in Figure 4.4 is translated by increment ρ_0=4, and every column is rotated by increment ϕ_0 =4 which is means rotate by ϕ_0 =180.

------------------------------------- T6 => $\rho_0 \neq 0$, $\phi_0=0$, $z_0 \neq 0$ =>$\rho_0=6$, $\phi_0=0$, $z_0=-2$----------------------



```
 section 0                        section 1                 section 2                  section 3
0e d4 c5 e9 22 f5 dc 15  |21 41 3b ba 0e ce a0 f9  |ca cf 7a e6 c4 ad c6 c6  |f9 ba 09 9c 47 a6 80 ec  |
72 7c da 10 3b b1 0b 9c  |1c 9f dc 4c 88 d3 29 c1  |2f 02 98 54 97 ca 5d 09  |57 4c 62 e5 b2 af 69 51  |
20 26 19 07 f9 81 83 aa  |1d 38 3c ab 72 7f 2c 88  |6d de b4 8b 56 16 53 fe  |54 b0 c7 cf fd c9 de 44  |
e8 89 ba 02 e2 ec 6e 2d  |0b ab 16 4e a5 28 07 6c  |06 69 16 ba 4a 3b c6 b9  |c6 f0 c9 38 77 66 bd 8b  |
6e 01 28 9c 27 76 2c cd  |38 f8 a3 c1 5a b2 c3 da  |44 2d 67 fb d1 ce 26 95  |62 e6 82 15 32 dc 1f d4  |
f2 03 72 04 7c 0e 45 a4  |6d f2 05 68 ef 4b 03 4f  |07 51 5e 60 67 47 0d 93  |b9 ff 9d 9f 47 73 42 84  |
7b 5f 21 66 04 5d 9f b8  |bd 1c d6 82 ff 91 f7 ca  |55 f2 82 3e 52 3d c1 25  |39 aa 03 76 12 75 0d ab  |
45 f5 ab 43 83 21 60 cd  |9f bf af 76 3d 8a a7 51  |60 20 57 7b 6f 5c 78 b9  |79 95 15 3b 8c 00 51 d8  |
```

Figure 4.6. Dynamic 3D S-Box from BT (T6) in Round 0

The above Figure 4.6 illustrates the output from BT (T6). In this step the input to BT is the output from previous step (3D in Figure 4.5). Every rows in dynamic 3D S-Box in Figure 4.5 is translated by increment ρ_0=6, and every section is translated by increment z_0=-2

4.2.2 Dynamic P-Box

The deliverables for this step are an algorithm and dynamic P-Box values.

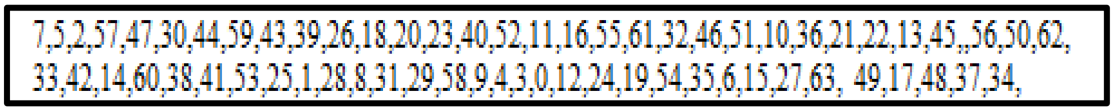
4.2.2.1 Algorithm of Dynamic P-Box

The algorithm developed to perform this step is as follows:

```
Algorithm 6: Dynamic P-Box

Input:   8 bytes, i-th round
Output: 8 bytes after premutation
            Seed2=8 bytes
            mysrand(seed2)
            for k= 0 to 63
              {
                P-Box[k]=k        // intial P-Box with fixed value of 0...63
                n[k]=myrand%64   //generate random numbers 0…63  from myrand ( )
                                   function and  store them in n
              }
            for k=0 to 63
            swap(P-Box[n[k]], P-Box[k])
```

4.2.2.2 Dynamic P-Box Values

The values produced are shown in Figure 4.7.

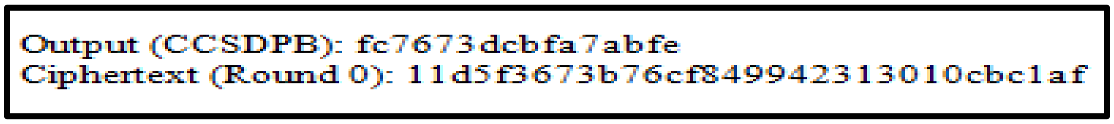


```
7,5,2,57,47,30,44,59,43,39,26,18,20,23,40,52,11,16,55,61,32,46,51,10,36,21,22,13,45,,56,50,62,
33,42,14,60,38,41,53,25,1,28,8,31,29,58,9,4,3,0,12,24,19,54,35,6,15,27,63,  49,17,48,37,34,
```

Figure 4.7. Dynamic P-Box in Round 0

4.2.3 Cylindrical Coordinate System with Dynamic Permutation Box Function

The output of CCSDPB and ciphertext in Round 0 are shown in Figure 4.8.



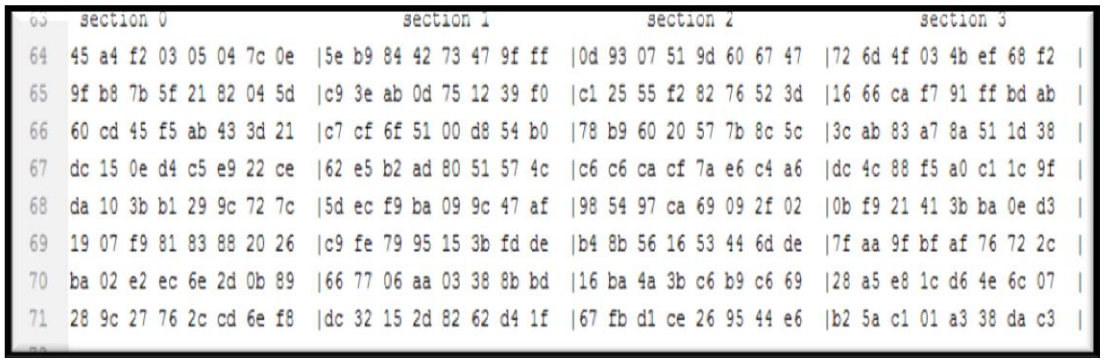
```
Output (CCSDPB): fc7673dcbfa7abfe
Ciphertext (Round 0): 11d5f3673b76cf8499423130l0cbc1af
```

Figure 4.8. Output of CCSDPB Function and Ciphertext in Round 0

It is shown in Figure 4.8 the output of CCSDPB function and Ciphertext in hexadecimal where the CCSDPB function is 64 bits while the outputs of the ciphertext is 128 bits. The results of the other three rounds are as follows:

Round 1

Output: dynamic 3D S-BOX of after apply Relocate BR (D1)

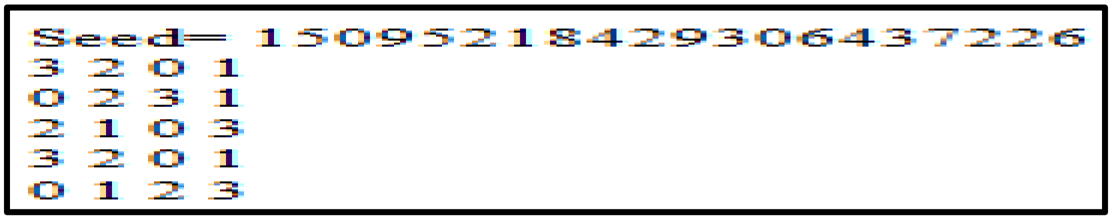


```
 section 0                       section 1                  section 2                  section 3
45 a4 f2 03 05 04 7c 0e   |5e b9 84 42 73 47 9f ff   |0d 93 07 51 9d 60 67 47   |72 6d 4f 03 4b ef 68 f2   |
9f b8 7b 5f 21 82 04 5d   |c9 3e ab 0d 75 12 39 f0   |c1 25 55 f2 82 76 52 3d   |16 66 ca f7 91 ff bd ab   |
60 cd 45 f5 ab 43 3d 21   |c7 cf 6f 51 00 d8 54 b0   |78 b9 60 20 57 7b 8c 5c   |3c ab 83 a7 8a 51 1d 38   |
dc 15 0e d4 c5 e9 22 ce   |62 e5 b2 ad 80 51 57 4c   |c6 c6 ca cf 7a e6 c4 a6   |dc 4c 88 f5 a0 c1 1c 9f   |
da 10 3b b1 29 9c 72 7c   |5d ec f9 ba 09 9c 47 af   |98 54 97 ca 69 09 2f 02   |0b f9 21 41 3b ba 0e d3   |
19 07 f9 81 83 88 20 26   |c9 fe 79 95 15 3b fd de   |b4 8b 56 16 53 44 6d de   |7f aa 9f bf af 76 72 2c   |
ba 02 e2 ec 6e 2d 0b 89   |66 77 06 aa 03 38 8b bd   |16 ba 4a 3b c6 b9 c6 69   |28 a5 e8 1c d6 4e 6c 07   |
28 9c 27 76 2c cd 6e f8   |dc 32 15 2d 82 62 d4 1f   |67 fb d1 ce 26 95 44 e6   |b2 5a c1 01 a3 38 da c3   |
```

Figure 4.9. Dynamic 3D S-BOX of after apply Relocate BR (D1) in Round 1

Figure 4.9 illustrates the output from BR (D1) in Round 1. The input in this step is dynamic 3D S–Box from key expansion part in Figure 4.2. Four sections are presented Section 0, Section 1, Section 2, and Section 3 respectively. Every position in Sections contains one byte in hexadecimal.

Random secret keys

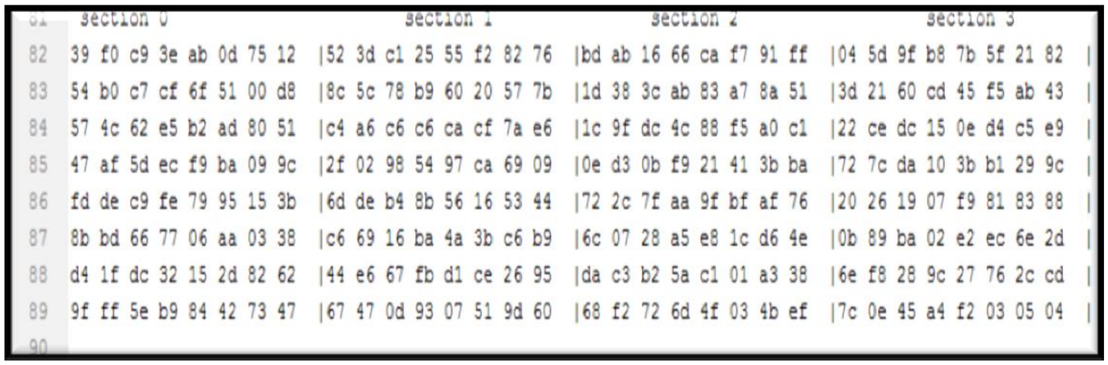


```
Seed= 1509521842930643 7226
3 2 0 1
0 2 3 1
2 1 0 3
3 2 0 1
0 1 2 3
```

Figure 4.10. Random secret keys in Round 1

Figure 4.10 illustrates five sets of Random secret keys that is generated from my rand function. The seed of myrand ( ) function in this round is 1509521842930643 7226.

------------------------------- T8 => $\rho_0 \neq 0$, $\phi_0 \neq 0$, $z_{0\neq}0$ => $\rho_0=7$, $\phi_0=2$, $z_0=3$----------------

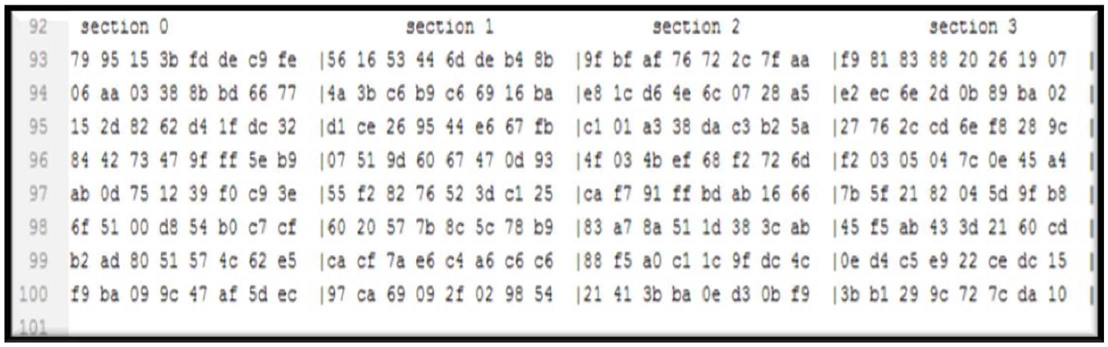


```
 section 0                  section 1                 section 2                  section 3
39 f0 c9 3e ab 0d 75 12  |52 3d c1 25 55 f2 82 76  |bd ab 16 66 ca f7 91 ff  |04 5d 9f b8 7b 5f 21 82  |
54 b0 c7 cf 6f 51 00 d8  |8c 5c 78 b9 60 20 57 7b  |1d 38 3c ab 83 a7 8a 51  |3d 21 60 cd 45 f5 ab 43  |
57 4c 62 e5 b2 ad 80 51  |c4 a6 c6 c6 ca cf 7a e6  |1c 9f dc 4c 88 f5 a0 c1  |22 ce dc 15 0e d4 c5 e9  |
47 af 5d ec f9 ba 09 9c  |2f 02 98 54 97 ca 69 09  |0e d3 0b f9 21 41 3b ba  |72 7c da 10 3b b1 29 9c  |
fd de c9 fe 79 95 15 3b  |6d de b4 8b 56 16 53 44  |72 2c 7f aa 9f bf af 76  |20 26 19 07 f9 81 83 88  |
8b bd 66 77 06 aa 03 38  |c6 69 16 ba 4a 3b c6 b9  |6c 07 28 a5 e8 1c d6 4e  |0b 89 ba 02 e2 ec 6e 2d  |
d4 1f dc 32 15 2d 82 62  |44 e6 67 fb d1 ce 26 95  |da c3 b2 5a c1 01 a3 38  |6e f8 28 9c 27 76 2c cd  |
9f ff 5e b9 84 42 73 47  |67 47 0d 93 07 51 9d 60  |68 f2 72 6d 4f 03 4b ef  |7c 0e 45 a4 f2 03 05 04  |
```

Figure 4.11. Dynamic 3D S-Box from BT (T8) in Round 1

The above Figure 4.11 illustrates the output from BT (T8) in Round 1. In this step the input to BT is the output from previous step (3D in Figure 4.9). Every rows in dynamic 3D S-Box in Figure 4.9 is translated by increment ρ_0=7, and every section is translated by increment z_0=3 as well as every column is rotated by increment ϕ_0 =2 which is means rotate by ϕ_0 =90.

----------------------------------T4 => $\rho_0 \neq 0$, $\phi_{0\neq}0$, $z_{0=}0$ => $\rho_0=4$, $\phi_0=4$---------------------



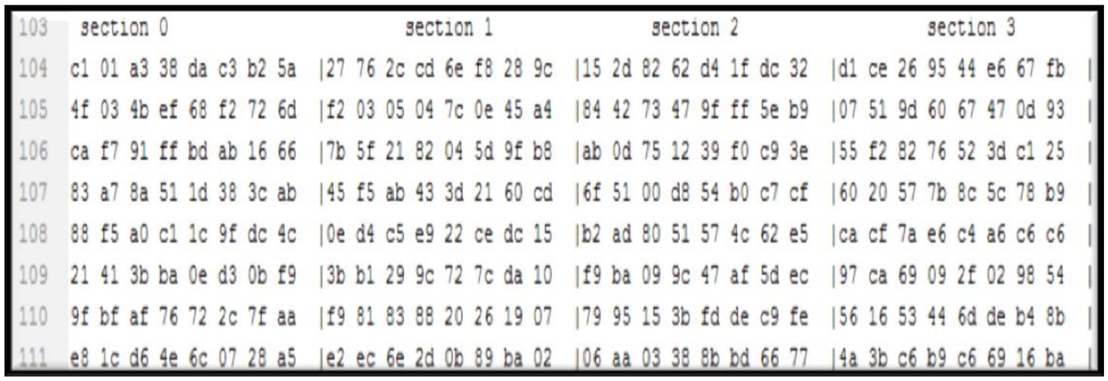
```
 section 0                  section 1                 section 2                  section 3
79 95 15 3b fd de c9 fe  |56 16 53 44 6d de b4 8b  |9f bf af 76 72 2c 7f aa  |f9 81 83 88 20 26 19 07  |
06 aa 03 38 8b bd 66 77  |4a 3b c6 b9 c6 69 16 ba  |e8 1c d6 4e 6c 07 28 a5  |e2 ec 6e 2d 0b 89 ba 02  |
15 2d 82 62 d4 1f dc 32  |d1 ce 26 95 44 e6 67 fb  |c1 01 a3 38 da c3 b2 5a  |27 76 2c cd 6e f8 28 9c  |
84 42 73 47 9f ff 5e b9  |07 51 9d 60 67 47 0d 93  |4f 03 4b ef 68 f2 72 6d  |f2 03 05 04 7c 0e 45 a4  |
ab 0d 75 12 39 f0 c9 3e  |55 f2 82 76 52 3d c1 25  |ca f7 91 ff bd ab 16 66  |7b 5f 21 82 04 5d 9f b8  |
6f 51 00 d8 54 b0 c7 cf  |60 20 57 7b 8c 5c 78 b9  |83 a7 8a 51 1d 38 3c ab  |45 f5 ab 43 3d 21 60 cd  |
b2 ad 80 51 57 4c 62 e5  |ca cf 7a e6 c4 a6 c6 c6  |88 f5 a0 c1 1c 9f dc 4c  |0e d4 c5 e9 22 ce dc 15  |
f9 ba 09 9c 47 af 5d ec  |97 ca 69 09 2f 02 98 54  |21 41 3b ba 0e d3 0b f9  |3b b1 29 9c 72 7c da 10  |
```

Figure 4.12. Dynamic 3D S-Box from BT (T4) in Round 1

The above Figure 4.12 illustrates the output from BT (T4) in Round 1. In this step the input to BT is the output from previous step (3D in Figure 4.11). Every rows in

dynamic 3D S-Box in Figure 4.11 is translated by increment ρ_0=4, and every column is rotated by increment ϕ_0 =4 which is means rotate by ϕ_0 =180.

-------------------------------- T6 => $\rho_0 \neq 0$, ϕ_0=0, $z_0 \neq 0$ => ρ_0=6, ϕ_0=0, z_0=-2---------------

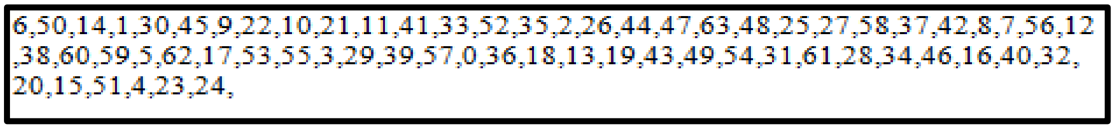


```
 section 0                        section 1                  section 2                    section 3
c1 01 a3 38 da c3 b2 5a  |27 76 2c cd 6e f8 28 9c  |15 2d 82 62 d4 1f dc 32  |d1 ce 26 95 44 e6 67 fb  |
4f 03 4b ef 68 f2 72 6d  |f2 03 05 04 7c 0e 45 a4  |84 42 73 47 9f ff 5e b9  |07 51 9d 60 67 47 0d 93  |
ca f7 91 ff bd ab 16 66  |7b 5f 21 82 04 5d 9f b8  |ab 0d 75 12 39 f0 c9 3e  |55 f2 82 76 52 3d c1 25  |
83 a7 8a 51 1d 38 3c ab  |45 f5 ab 43 3d 21 60 cd  |6f 51 00 d8 54 b0 c7 cf  |60 20 57 7b 8c 5c 78 b9  |
88 f5 a0 c1 1c 9f dc 4c  |0e d4 c5 e9 22 ce dc 15  |b2 ad 80 51 57 4c 62 e5  |ca cf 7a e6 c4 a6 c6 c6  |
21 41 3b ba 0e d3 0b f9  |3b b1 29 9c 72 7c da 10  |f9 ba 09 9c 47 af 5d ec  |97 ca 69 09 2f 02 98 54  |
9f bf af 76 72 2c 7f aa  |f9 81 83 88 20 26 19 07  |79 95 15 3b fd de c9 fe  |56 16 53 44 6d de b4 8b  |
e8 1c d6 4e 6c 07 28 a5  |e2 ec 6e 2d 0b 89 ba 02  |06 aa 03 38 8b bd 66 77  |4a 3b c6 b9 c6 69 16 ba  |
```

Figure 4.13. Dynamic 3D S-Box from BT (T6) in Round 1

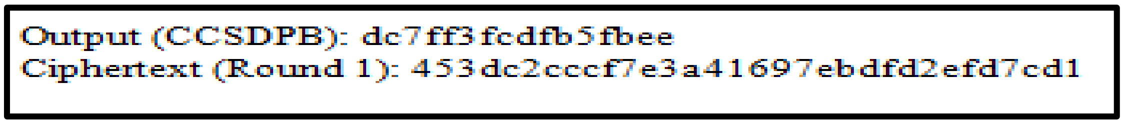
The above Figure 4.13 illustrates the output from BT (T6). In this step the input to BT is the output from previous step (3D in Figure 4.12). Every rows in dynamic 3D S-Box in Figure 4.8 is translated by increment ρ_0=6, and every section is translated by increment z_0=-2.

Dynamic P-Box



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

Figure 4.14. Dynamic P-Box in Round 1



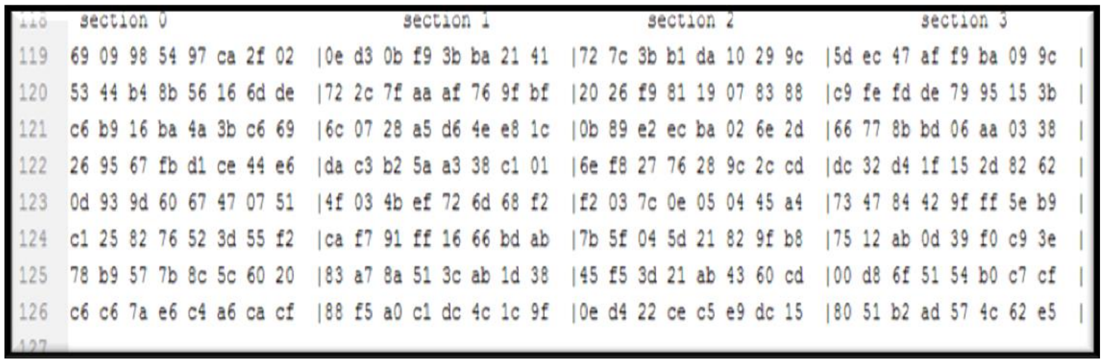
```
Output (CCSDPB): dc7ff3fcdfb5fbee
Ciphertext (Round 1): 453dc2cccf7e3a41697ebdfd2efd7cd1
```

Figure 4.15. Output of CCSDPB function and ciphertext in Round 1

It is shown in Figure 4.15 the output of CCSDPB function and Ciphertext in hexadecimal where the CCSDPB function is 64 bits while the outputs of the ciphertext is 128 bits.

Round 2

Output: dynamic 3D S-Box Relocate after apply BR (D_2)

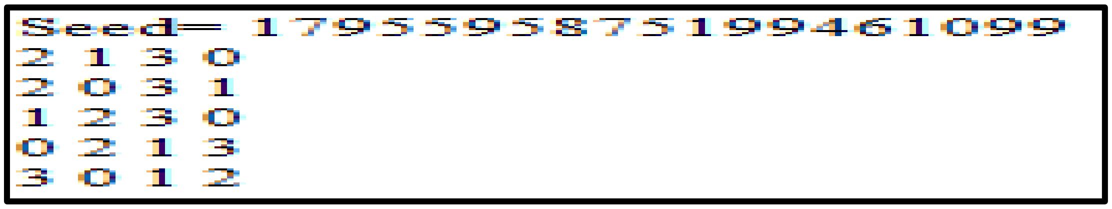


```
 section 0                section 1                section 2                section 3
69 09 98 54 97 ca 2f 02  |0e d3 0b f9 3b ba 21 41  |72 7c 3b b1 da 10 29 9c  |5d ec 47 af f9 ba 09 9c  |
53 44 b4 8b 56 16 6d de  |72 2c 7f aa af 76 9f bf  |20 26 f9 81 19 07 83 88  |c9 fe fd de 79 95 15 3b  |
c6 b9 16 ba 4a 3b c6 69  |6c 07 28 a5 d6 4e e8 1c  |0b 89 e2 ec ba 02 6e 2d  |66 77 8b bd 06 aa 03 38  |
26 95 67 fb d1 ce 44 e6  |da c3 b2 5a a3 38 c1 01  |6e f8 27 76 28 9c 2c cd  |dc 32 d4 1f 15 2d 82 62  |
0d 93 9d 60 67 47 07 51  |4f 03 4b ef 72 6d 68 f2  |f2 03 7c 0e 05 04 45 a4  |73 47 84 42 9f ff 5e b9  |
c1 25 82 76 52 3d 55 f2  |ca f7 91 ff 16 66 bd ab  |7b 5f 04 5d 21 82 9f b8  |75 12 ab 0d 39 f0 c9 3e  |
78 b9 57 7b 8c 5c 60 20  |83 a7 8a 51 3c ab 1d 38  |45 f5 3d 21 ab 43 60 cd  |00 d8 6f 51 54 b0 c7 cf  |
c6 c6 7a e6 c4 a6 ca cf  |88 f5 a0 c1 dc 4c 1c 9f  |0e d4 22 ce c5 e9 dc 15  |80 51 b2 ad 57 4c 62 e5  |
```

Figure 4.16. Dynamic 3D S-Box of after apply Relocate BR (D2) in Round 2

Figure 4.16 illustrates the output from BR (D2) in Round 2. The input in this step is dynamic 3D S–Box from key expansion part in Figure 4.2. Four Sections are presented Section 0, Section 1, Section 2, and Section 3 respectively. Every position in sections contains one byte in hexadecimal.

Random Secret Keys

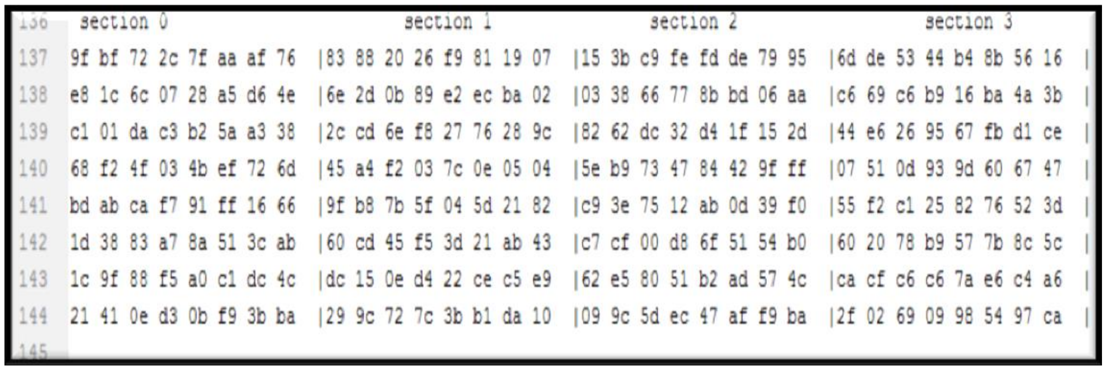


```
Seed= 1795595875199461099
2 1 3 0
2 0 3 1
1 2 3 0
0 2 1 3
3 0 1 2
```

Figure 4.17. Random secret keys in Round 2

Figure 4.17 illustrates five sets of Random secret keys that is generated from my rand function. The seed of myrand ( ) function in this round is 1795595875199461099.

--T8 => $\rho_0 \neq 0$, $\phi_0 \neq 0$, $z_{0\neq}0$ => $\rho_0=7$, $\phi_0=2$, $z_0=3$-----------------

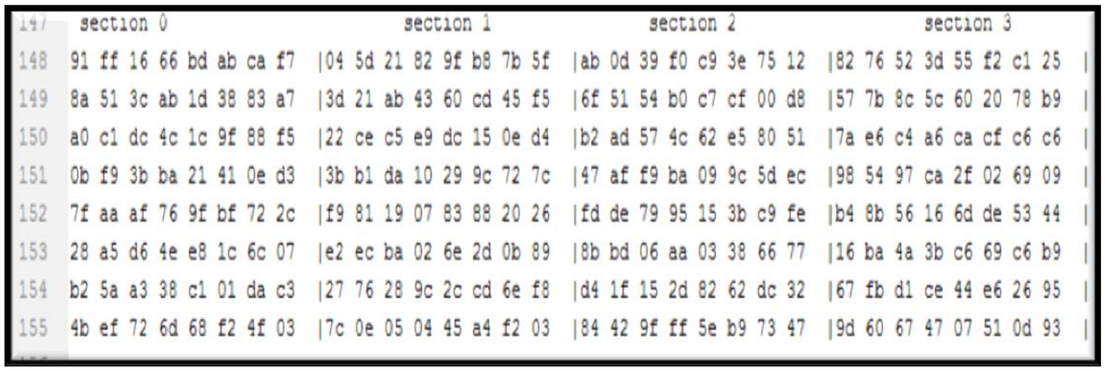


```
 section 0                   section 1                 section 2                  section 3
9f bf 72 2c 7f aa af 76  |83 88 20 26 f9 81 19 07  |15 3b c9 fe fd de 79 95  |6d de 53 44 b4 8b 56 16  |
e8 1c 6c 07 28 a5 d6 4e  |6e 2d 0b 89 e2 ec ba 02  |03 38 66 77 8b bd 06 aa  |c6 69 c6 b9 16 ba 4a 3b  |
c1 01 da c3 b2 5a a3 38  |2c cd 6e f8 27 76 28 9c  |82 62 dc 32 d4 1f 15 2d  |44 e6 26 95 67 fb d1 ce  |
68 f2 4f 03 4b ef 72 6d  |45 a4 f2 03 7c 0e 05 04  |5e b9 73 47 84 42 9f ff  |07 51 0d 93 9d 60 67 47  |
bd ab ca f7 91 ff 16 66  |9f b8 7b 5f 04 5d 21 82  |c9 3e 75 12 ab 0d 39 f0  |55 f2 c1 25 82 76 52 3d  |
1d 38 83 a7 8a 51 3c ab  |60 cd 45 f5 3d 21 ab 43  |c7 cf 00 d8 6f 51 54 b0  |60 20 78 b9 57 7b 8c 5c  |
1c 9f 88 f5 a0 c1 dc 4c  |dc 15 0e d4 22 ce c5 e9  |62 e5 80 51 b2 ad 57 4c  |ca cf c6 c6 7a e6 c4 a6  |
21 41 0e d3 0b f9 3b ba  |29 9c 72 7c 3b b1 da 10  |09 9c 5d ec 47 af f9 ba  |2f 02 69 09 98 54 97 ca  |
```

Figure 4.18. Dynamic 3D S-Box from BT (T8) in Round 2

The above Figure 4.18 illustrates the output from BT (T8) in round 2. In this step the input to BT is the output from previous step (3D in Figure 4.17). Every rows in dynamic 3D S-Box in Figure 4.17 is translated by increment $\rho_0=7$, and every section is translated by increment $z_0=3$ as well as every column is rotated by increment $\phi_0=2$ which is means rotate by $\phi_0=90$.

----------------------------------T4 => $\rho_0 \neq 0$, $\phi_{0\neq}0$, $z_{0=}0$ => $\rho_0=4$, $\phi_0=4$----------------------



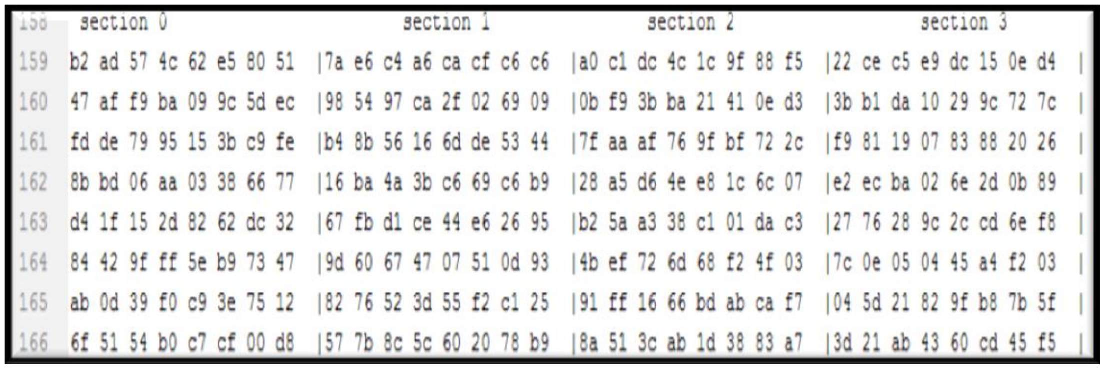
```
 section 0                   section 1                 section 2                  section 3
91 ff 16 66 bd ab ca f7  |04 5d 21 82 9f b8 7b 5f  |ab 0d 39 f0 c9 3e 75 12  |82 76 52 3d 55 f2 c1 25  |
8a 51 3c ab 1d 38 83 a7  |3d 21 ab 43 60 cd 45 f5  |6f 51 54 b0 c7 cf 00 d8  |57 7b 8c 5c 60 20 78 b9  |
a0 c1 dc 4c 1c 9f 88 f5  |22 ce c5 e9 dc 15 0e d4  |b2 ad 57 4c 62 e5 80 51  |7a e6 c4 a6 ca cf c6 c6  |
0b f9 3b ba 21 41 0e d3  |3b b1 da 10 29 9c 72 7c  |47 af f9 ba 09 9c 5d ec  |98 54 97 ca 2f 02 69 09  |
7f aa af 76 9f bf 72 2c  |f9 81 19 07 83 88 20 26  |fd de 79 95 15 3b c9 fe  |b4 8b 56 16 6d de 53 44  |
28 a5 d6 4e e8 1c 6c 07  |e2 ec ba 02 6e 2d 0b 89  |8b bd 06 aa 03 38 66 77  |16 ba 4a 3b c6 69 c6 b9  |
b2 5a a3 38 c1 01 da c3  |27 76 28 9c 2c cd 6e f8  |d4 1f 15 2d 82 62 dc 32  |67 fb d1 ce 44 e6 26 95  |
4b ef 72 6d 68 f2 4f 03  |7c 0e 05 04 45 a4 f2 03  |84 42 9f ff 5e b9 73 47  |9d 60 67 47 07 51 0d 93  |
```

Figure 4.19 . Dynamic 3D S-Box from BT (T4) in Round 2

The above Figure 4.19 illustrates the output from BT (T4) in Round 2. In this step the input to BT is the output from previous step (3D in Figure 4.18). Every rows in

dynamic 3D S-Box in Figure 4.18 is translated by increment ρ_0=4, and every column is rotated by increment ϕ_0 =4 which is means rotate by ϕ_0 =180.

----------------------------------T6 => $\rho_0 \neq 0$, ϕ_0=0, $z_0 \neq 0$ => ρ_0=6, ϕ_0=0, z_0=-2-------------------

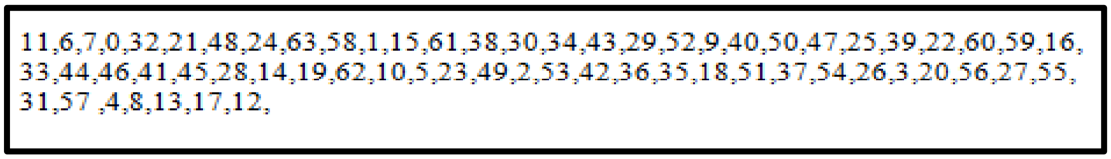


```
     section 0                  section 1                  section 2                  section 3
b2 ad 57 4c 62 e5 80 51  |7a e6 c4 a6 ca cf c6 c6  |a0 c1 dc 4c 1c 9f 88 f5  |22 ce c5 e9 dc 15 0e d4  |
47 af f9 ba 09 9c 5d ec  |98 54 97 ca 2f 02 69 09  |0b f9 3b ba 21 41 0e d3  |3b b1 da 10 29 9c 72 7c  |
fd de 79 95 15 3b c9 fe  |b4 8b 56 16 6d de 53 44  |7f aa af 76 9f bf 72 2c  |f9 81 19 07 83 88 20 26  |
8b bd 06 aa 03 38 66 77  |16 ba 4a 3b c6 69 c6 b9  |28 a5 d6 4e e8 1c 6c 07  |e2 ec ba 02 6e 2d 0b 89  |
d4 1f 15 2d 82 62 dc 32  |67 fb d1 ce 44 e6 26 95  |b2 5a a3 38 c1 01 da c3  |27 76 28 9c 2c cd 6e f8  |
84 42 9f ff 5e b9 73 47  |9d 60 67 47 07 51 0d 93  |4b ef 72 6d 68 f2 4f 03  |7c 0e 05 04 45 a4 f2 03  |
ab 0d 39 f0 c9 3e 75 12  |82 76 52 3d 55 f2 c1 25  |91 ff 16 66 bd ab ca f7  |04 5d 21 82 9f b8 7b 5f  |
6f 51 54 b0 c7 cf 00 d8  |57 7b 8c 5c 60 20 78 b9  |8a 51 3c ab 1d 38 83 a7  |3d 21 ab 43 60 cd 45 f5  |
```

Figure 4.20. Dynamic 3D S-Box from BT (T6) in Round 2

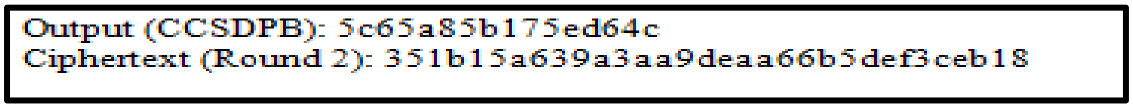
The above Figure 4.20 illustrates the output from BT (T6). In this step the input to BT is the output from previous step (3D in Figure 4.19). Every rows in dynamic 3D S-Box in Figure 4.19 is translated by increment ρ_0=6, and every section is translated by increment z_0=-2.

Dynamic P-Box



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

Figure 4.21. Dynamic P-Box in Round 2



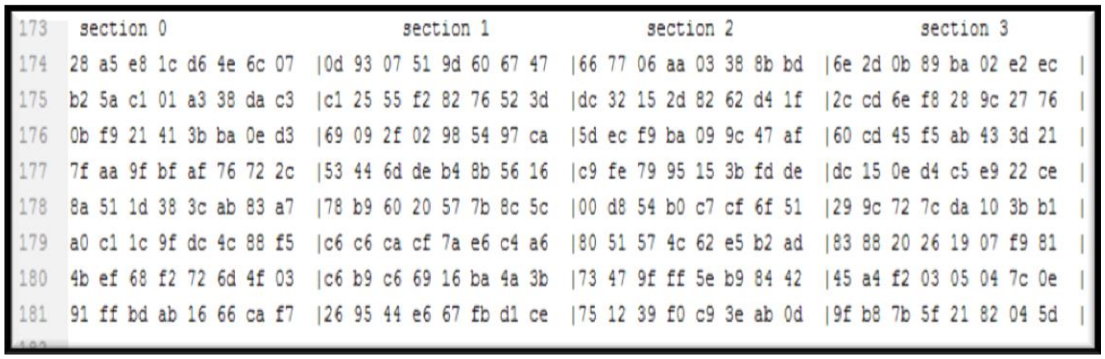
```
Output (CCSDPB): 5c65a85b175ed64c
Ciphertext (Round 2): 351b15a639a3aa9deaa66b5def3ceb18
```

Figure 4.22. Output of CCSDPB function and ciphertext in Round 2

Figure 4.22 illustrates the output of CCSDPB function and ciphertext in Round 2 in hexadecimal where the CCSDPB function is 64 bits while the outputs of the ciphertext in Round 2 is 128 bits.

Round 3

Output: dynamic 3D S-Box, Relocate after apply BR (D_3)

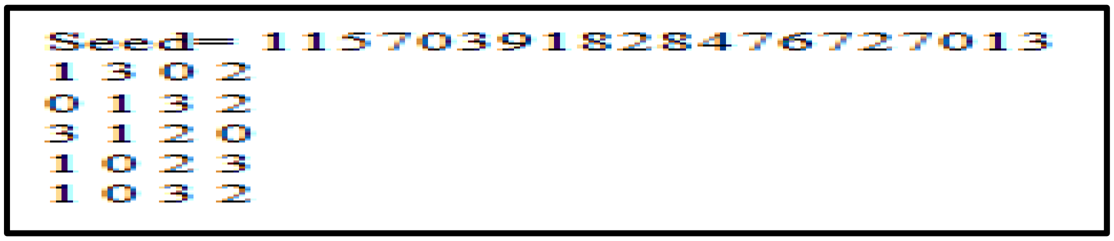


```
 section 0                         section 1                 section 2                   section 3
28 a5 e8 1c d6 4e 6c 07  |0d 93 07 51 9d 60 67 47  |66 77 06 aa 03 38 8b bd  |6e 2d 0b 89 ba 02 e2 ec  |
b2 5a c1 01 a3 38 da c3  |c1 25 55 f2 82 76 52 3d  |dc 32 15 2d 82 62 d4 1f  |2c cd 6e f8 28 9c 27 76  |
0b f9 21 41 3b ba 0e d3  |69 09 2f 02 98 54 97 ca  |5d ec f9 ba 09 9c 47 af  |60 cd 45 f5 ab 43 3d 21  |
7f aa 9f bf af 76 72 2c  |53 44 6d de b4 8b 56 16  |c9 fe 79 95 15 3b fd de  |dc 15 0e d4 c5 e9 22 ce  |
8a 51 1d 38 3c ab 83 a7  |78 b9 60 20 57 7b 8c 5c  |00 d8 54 b0 c7 cf 6f 51  |29 9c 72 7c da 10 3b b1  |
a0 c1 1c 9f dc 4c 88 f5  |c6 c6 ca cf 7a e6 c4 a6  |80 51 57 4c 62 e5 b2 ad  |83 88 20 26 19 07 f9 81  |
4b ef 68 f2 72 6d 4f 03  |c6 b9 c6 69 16 ba 4a 3b  |73 47 9f ff 5e b9 84 42  |45 a4 f2 03 05 04 7c 0e  |
91 ff bd ab 16 66 ca f7  |26 95 44 e6 67 fb d1 ce  |75 12 39 f0 c9 3e ab 0d  |9f b8 7b 5f 21 82 04 5d  |
```

Figure 4.23. Dynamic 3D S-Box of after apply Relocate BR (D3) in Round 3

Figure 4.23 illustrates the output from BR (D3) in round 3. The input in this step is dynamic 3D S–Box from key expansion part in Figure 4.2. Four Sections are presented Section 0, Section 1, Section 2, and Section 3 respectively. Every position in Sections contains one byte in hexadecimal.

Random Secret Keys

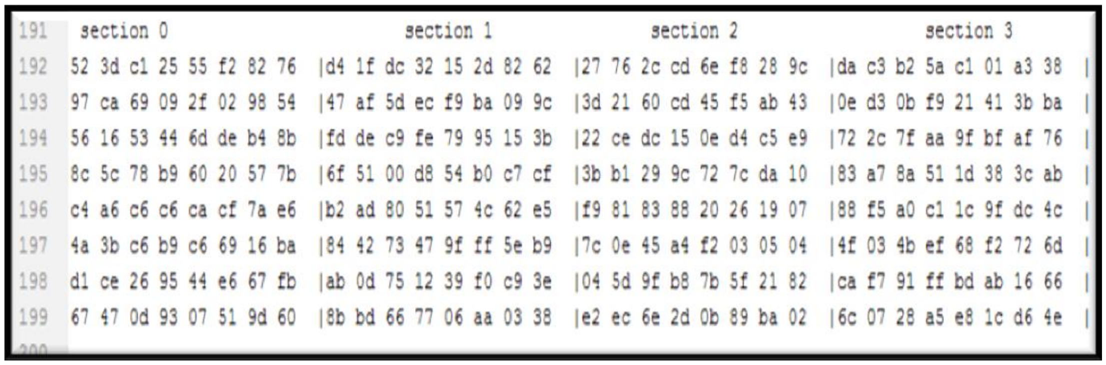


```
Seed= 115703918284767227013
1 3 0 2
0 1 3 2
3 1 2 0
1 0 2 3
1 0 3 2
```

Figure 4.24. Random secret keys in Round 3

Figure 4.24 illustrates five sets of Random secret keys that is generated from my rand ( ) function. The seed of myrand ( ) function in this round is 11570391828476727013.

---------------------T8 => $\rho_0 \neq 0$, $\phi_0 \neq 0$, $z_{0\neq}0$ => $\rho_0=7$, $\phi_0=2$, $z_0=3$-------------------------

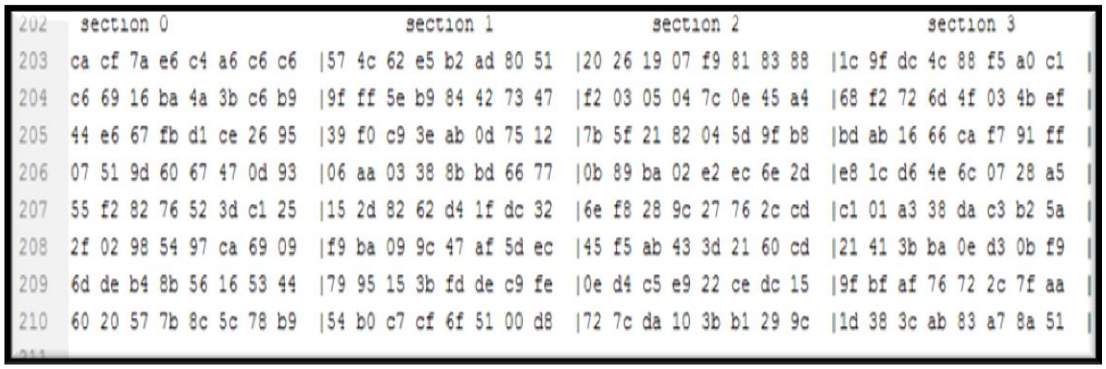


```
 section 0                 section 1                 section 2                 section 3
52 3d c1 25 55 f2 82 76  |d4 1f dc 32 15 2d 82 62  |27 76 2c cd 6e f8 28 9c  |da c3 b2 5a c1 01 a3 38  |
97 ca 69 09 2f 02 98 54  |47 af 5d ec f9 ba 09 9c  |3d 21 60 cd 45 f5 ab 43  |0e d3 0b f9 21 41 3b ba  |
56 16 53 44 6d de b4 8b  |fd de c9 fe 79 95 15 3b  |22 ce dc 15 0e d4 c5 e9  |72 2c 7f aa 9f bf af 76  |
8c 5c 78 b9 60 20 57 7b  |6f 51 00 d8 54 b0 c7 cf  |3b b1 29 9c 72 7c da 10  |83 a7 8a 51 1d 38 3c ab  |
c4 a6 c6 c6 ca cf 7a e6  |b2 ad 80 51 57 4c 62 e5  |f9 81 83 88 20 26 19 07  |88 f5 a0 c1 1c 9f dc 4c  |
4a 3b c6 b9 c6 69 16 ba  |84 42 73 47 9f ff 5e b9  |7c 0e 45 a4 f2 03 05 04  |4f 03 4b ef 68 f2 72 6d  |
d1 ce 26 95 44 e6 67 fb  |ab 0d 75 12 39 f0 c9 3e  |04 5d 9f b8 7b 5f 21 82  |ca f7 91 ff bd ab 16 66  |
67 47 0d 93 07 51 9d 60  |8b bd 66 77 06 aa 03 38  |e2 ec 6e 2d 0b 89 ba 02  |6c 07 28 a5 e8 1c d6 4e  |
```

Figure 4.25. Dynamic 3D S-Box from BT (T8) in Round 3

The above Figure 4.25 illustrates the output from BT (T8) in round 3. In this step the input to BT is the output from previous step (3D in Figure 4.23). Every rows in dynamic 3D S-Box in Figure 4.23 is translated by increment $\rho_0=7$, and every section is translated by increment $z_0=3$ as well as every column is rotated by increment $\phi_0=2$ which is means rotate by $\phi_0=90$.

----------------------T4 => $\rho_0 \neq 0$, $\phi_0 \neq 0$, $z_{0=}0$ => $\rho_0=4$, $\phi_0=4$---------------------------------



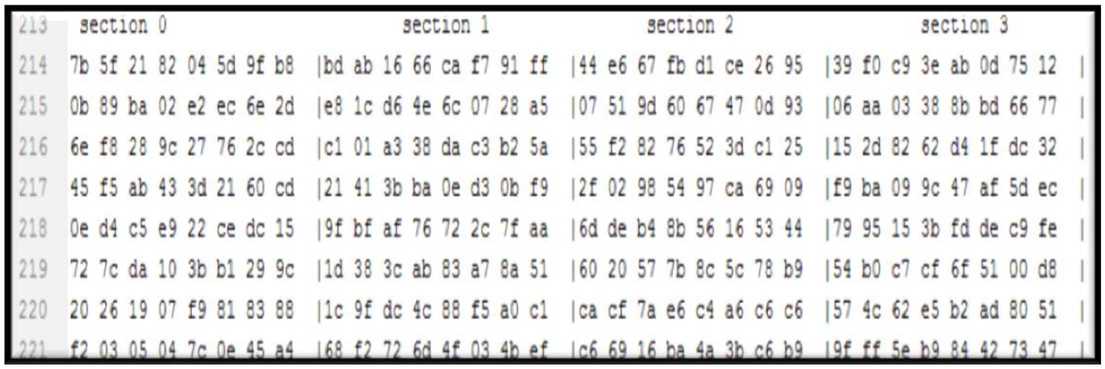
```
 section 0                 section 1                 section 2                 section 3
ca cf 7a e6 c4 a6 c6 c6  |57 4c 62 e5 b2 ad 80 51  |20 26 19 07 f9 81 83 88  |1c 9f dc 4c 88 f5 a0 c1  |
c6 69 16 ba 4a 3b c6 b9  |9f ff 5e b9 84 42 73 47  |f2 03 05 04 7c 0e 45 a4  |68 f2 72 6d 4f 03 4b ef  |
44 e6 67 fb d1 ce 26 95  |39 f0 c9 3e ab 0d 75 12  |7b 5f 21 82 04 5d 9f b8  |bd ab 16 66 ca f7 91 ff  |
07 51 9d 60 67 47 0d 93  |06 aa 03 38 8b bd 66 77  |0b 89 ba 02 e2 ec 6e 2d  |e8 1c d6 4e 6c 07 28 a5  |
55 f2 82 76 52 3d c1 25  |15 2d 82 62 d4 1f dc 32  |6e f8 28 9c 27 76 2c cd  |c1 01 a3 38 da c3 b2 5a  |
2f 02 98 54 97 ca 69 09  |f9 ba 09 9c 47 af 5d ec  |45 f5 ab 43 3d 21 60 cd  |21 41 3b ba 0e d3 0b f9  |
6d de b4 8b 56 16 53 44  |79 95 15 3b fd de c9 fe  |0e d4 c5 e9 22 ce dc 15  |9f bf af 76 72 2c 7f aa  |
60 20 57 7b 8c 5c 78 b9  |54 b0 c7 cf 6f 51 00 d8  |72 7c da 10 3b b1 29 9c  |1d 38 3c ab 83 a7 8a 51  |
```

Figure 4.26. Dynamic 3D S-Box from BT (T4) in Round 3

The above Figure 4.26 illustrates the output from BT (T4) in Round 3. In this step the input to BT is the output from previous step (3D in Figure 4.25). Every rows in

dynamic 3D S-Box in Figure 4.21 is translated by increment ρ_0=4, and every column is rotated by increment ϕ_0 =4 which is means rotate by ϕ_0 =180.

------------------------T6 => $\rho_0 \neq 0$, ϕ_0=0, $z_0 \neq 0$ => ρ_0=6, ϕ_0=0, z_0=-2-------------------------

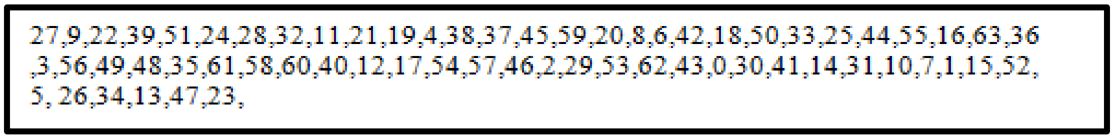


```
 section 0                       section 1                 section 2                 section 3
7b 5f 21 82 04 5d 9f b8  |bd ab 16 66 ca f7 91 ff  |44 e6 67 fb d1 ce 26 95  |39 f0 c9 3e ab 0d 75 12  |
0b 89 ba 02 e2 ec 6e 2d  |e8 1c d6 4e 6c 07 28 a5  |07 51 9d 60 67 47 0d 93  |06 aa 03 38 8b bd 66 77  |
6e f8 28 9c 27 76 2c cd  |c1 01 a3 38 da c3 b2 5a  |55 f2 82 76 52 3d c1 25  |15 2d 82 62 d4 1f dc 32  |
45 f5 ab 43 3d 21 60 cd  |21 41 3b ba 0e d3 0b f9  |2f 02 98 54 97 ca 69 09  |f9 ba 09 9c 47 af 5d ec  |
0e d4 c5 e9 22 ce dc 15  |9f bf af 76 72 2c 7f aa  |6d de b4 8b 56 16 53 44  |79 95 15 3b fd de c9 fe  |
72 7c da 10 3b b1 29 9c  |1d 38 3c ab 83 a7 8a 51  |60 20 57 7b 8c 5c 78 b9  |54 b0 c7 cf 6f 51 00 d8  |
20 26 19 07 f9 81 83 88  |1c 9f dc 4c 88 f5 a0 c1  |ca cf 7a e6 c4 a6 c6 c6  |57 4c 62 e5 b2 ad 80 51  |
f2 03 05 04 7c 0e 45 a4  |68 f2 72 6d 4f 03 4b ef  |c6 69 16 ba 4a 3b c6 b9  |9f ff 5e b9 84 42 73 47  |
```

Figure 4.27. Dynamic 3D S-Box from BT (T6) in Round 3

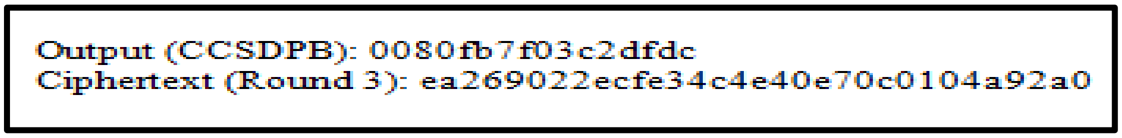
The above Figure 4.27 illustrates the output from BT (T6). In this step the input to BT is the output from previous step (3D in Figure 4.26). Every rows in dynamic 3D S-Box in Figure 4.26 is translated by increment ρ_0=6, and every section is translated by increment z_0=-2.

Dynamic P-Box



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

Figure 4.28. Dynamic P-Box in Round 3



```
Output (CCSDPB): 0080fb7f03c2dfdc
Ciphertext (Round 3): ea269022ecfe34c4e40e70c0104a92a0
```

Figure 4.29. Output of CCSDPB function and ciphertext in Round 3

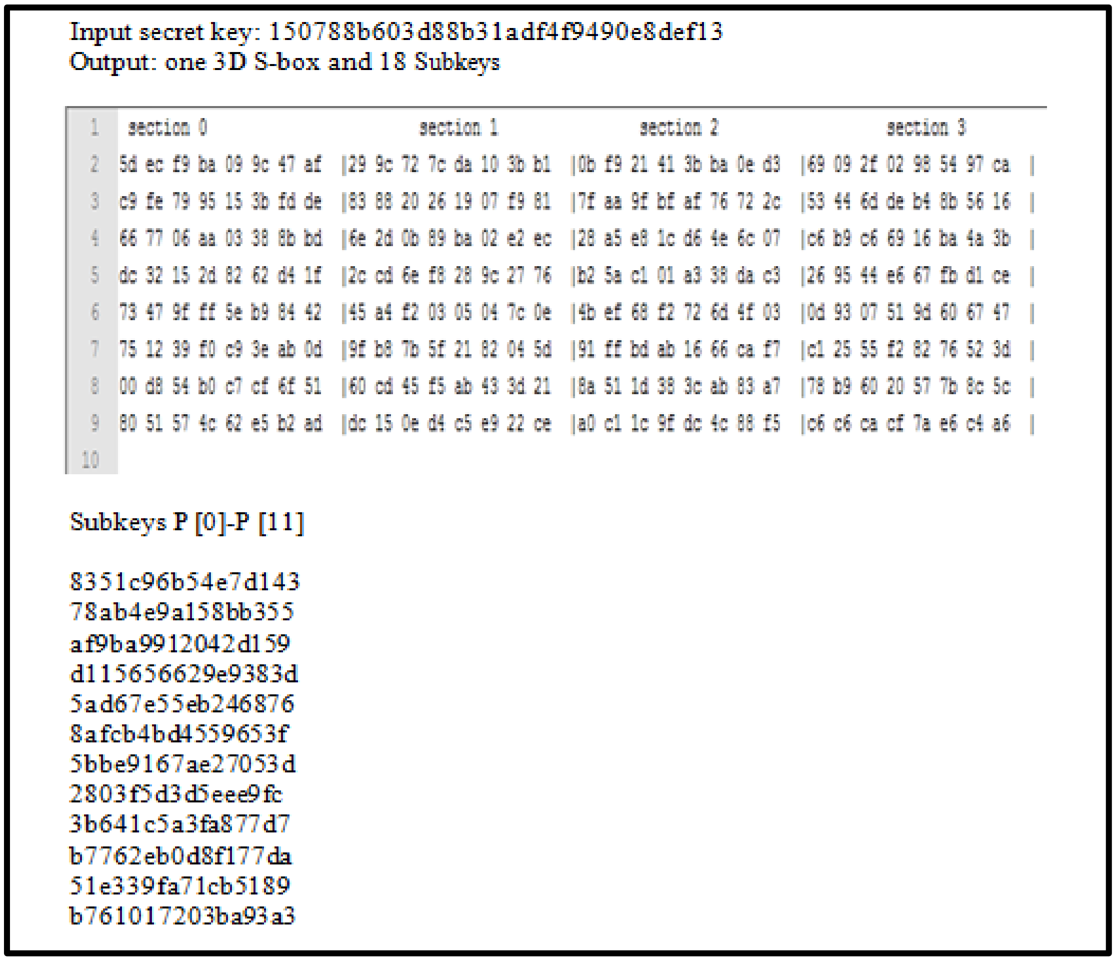
Figure 4.29 illustrates the output of CCSDPB function and ciphertext in round 3 in hexadecimal where the CCSDPB function is 64 bits while the outputs of the ciphertext in Round 3 is 128 bits. The results on the others rounds are shown in Appendix B.

4.3 Phase 2 RAF Implementation

The deliverables of this phase comprise two parts: (1) key expansion, and (2) data encryption and data decryption.

4.3.1 Key Expansion

The key expansion generated the P-array and dynamic 3D S-Box values. The results from the key expansion are shown below.



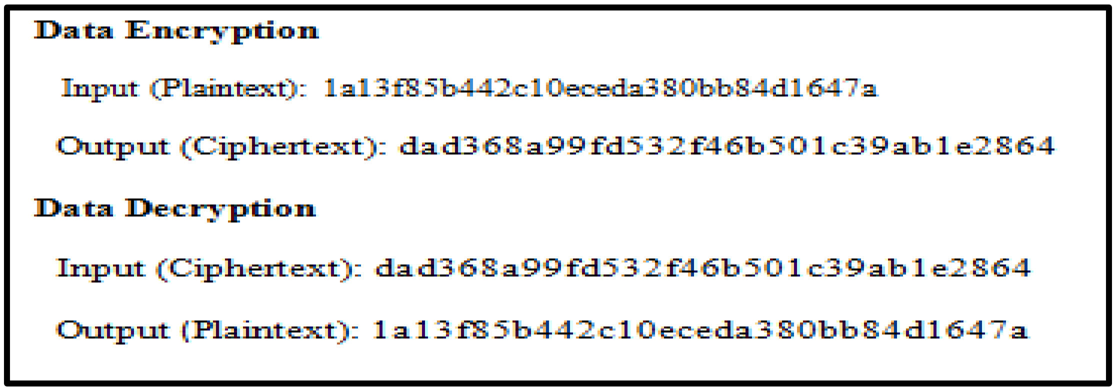
```
Input secret key: 150788b603d88b31adf4f9490e8def13
Output: one 3D S-box and 18 Subkeys

 section 0                     section 1                 section 2                  section 3
5d ec f9 ba 09 9c 47 af  |29 9c 72 7c da 10 3b b1  |0b f9 21 41 3b ba 0e d3  |69 09 2f 02 98 54 97 ca  |
c9 fe 79 95 15 3b fd de  |83 88 20 26 19 07 f9 81  |7f aa 9f bf af 76 72 2c  |53 44 6d de b4 8b 56 16  |
66 77 06 aa 03 38 8b bd  |6e 2d 0b 89 ba 02 e2 ec  |28 a5 e8 1c d6 4e 6c 07  |c6 b9 c6 69 16 ba 4a 3b  |
dc 32 15 2d 82 62 d4 1f  |2c cd 6e f8 28 9c 27 76  |b2 5a c1 01 a3 38 da c3  |26 95 44 e6 67 fb d1 ce  |
73 47 9f ff 5e b9 84 42  |45 a4 f2 03 05 04 7c 0e  |4b ef 68 f2 72 6d 4f 03  |0d 93 07 51 9d 60 67 47  |
75 12 39 f0 c9 3e ab 0d  |9f b8 7b 5f 21 82 04 5d  |91 ff bd ab 16 66 ca f7  |c1 25 55 f2 82 76 52 3d  |
00 d8 54 b0 c7 cf 6f 51  |60 cd 45 f5 ab 43 3d 21  |8a 51 1d 38 3c ab 83 a7  |78 b9 60 20 57 7b 8c 5c  |
80 51 57 4c 62 e5 b2 ad  |dc 15 0e d4 c5 e9 22 ce  |a0 c1 1c 9f dc 4c 88 f5  |c6 c6 ca cf 7a e6 c4 a6  |

Subkeys P [0]-P [11]

8351c96b54e7d143
78ab4e9a158bb355
af9ba9912042d159
d115656629e9383d
5ad67e55eb246876
8afcb4bd4559653f
5bbe9167ae27053d
2803f5d3d5eee9fc
3b641c5a3fa877d7
b7762eb0d8f177da
51e339fa71cb5189
b761017203ba93a3
```

Figure 4.30. Output of key expansion part

Figure 4.30 illustrates the output of key expansion part that is dynamic 3D S-Box with four sections and 18 subkeys in hexadecimal. The input in this step is secret key (150788b603d88b31adf4f9490e8def13 in hexadecimal) while the outputs are one 3DS-Box and 18 subkeys from P0-P17 of 64 bits (16 digits in hexadecimal).

4.3.2 Data Encryption and Data Decryption

The results for this step are shown in Figure 4.30


Data Encryption

Input (Plaintext): 1a13f85b442c10eceda380bb84d1647a

Output (Ciphertext): dad368a99fd532f46b501c39ab1e2864

Data Decryption

Input (Ciphertext): dad368a99fd532f46b501c39ab1e2864

Output (Plaintext): 1a13f85b442c10eceda380bb84d1647a

Figure 4.31. Data encryption and data decryption

The data encryption in Figure 4.31 was encrypted to produce the ciphertext ' dad368a99fd532f46b501c39ab1e2864' whereas in data decryption the same ciphetext 'dad368a99fd532f46b501c39ab1e2864' was decrypted and produced the original input (plaintext) which is '1a13f85b442c10eceda380bb84d1647a '.

4.3 Summary

The chapter presents the output for the phase 1 and phase 2 of the methodology. These include dynamic 3D S-Box, dynamic P-Box, different secret random keys, F-function (CCSDPB) in the first fourth rounds, subkeys for p1-p17, dynamic 3D S-box in key expansion part, and data encryption and decryption for RAF.

CHAPTER FIVE

EXPERIMENTAL RESULTS OF DYNAMIC 3D S-BOX

5.1 Introduction

This chapter presents the results on the dynamic 3D S-Box, including the verification of the dynamic 3D S-Box using three criteria (AVAL, SAC, and BIC), the correlation coefficient between dynamic 3D S-Boxes in RAF under different 256-bit Encryption keys (*Eks*), and the different random and nonrandom plaintext, the latter both low-entropy and high-entropy. The aim is to examine the effect of randomness of the Eks and the plaintext on the security of the dynamic 3D S-Box in RAF. The verification process has been mentioned earlier in Section 3.4.1.

5.2 Results of Dynamic 3D S-Box Evaluation with 3 Criteria

This section presents the empirical results of the dynamic 3D S-Box using three criteria AVAL, SAC, and BIC with different Eks. The results are compared with S-Boxes in BA.

5.2.1 AVAL Empirical Results

Table 5.1 shows the AVAL results. The values of k_{AVAL} *(i)* satisfies Equation (2.13) and values of k_{AVAL} correspond to the changed input bits (e_i, i= 1 …8) where e_1 represents the first changed input bit, e_2 represents the second changed input bit, and subsequently the other parameters whereby e_i (i=3 …8). The third column in Table 5.1 indicates the random Eks in hexadecimal, the fourth column shows the changed i-th input bit, and the last column contains the average change of the output bits from the changed the i-th input bit. The results of the first five experiments are discussed here as follows:

Table 5.1

k_{AVAL} (i) values for the S-boxes (first random128–bit Ek)

Name of algorithm	No. of experiment	Random 128-bit Ek in Hexadecimal	S-Box sequence	i-th Avalanche	value of i-th Avalanche (kAVAL(i))
BA	1	5a22cf8f5c8b190447fe784 467b2e538	1	$k_{AVAL\ (1)}$	0.5029
				$k_{AVAL\ (2)}$	0.4995
				$k_{AVAL\ (3)}$	0.4961
				$k_{AVAL\ (4)}$	0.5122
				$k_{AVAL\ (5)}$	0.4922
				$k_{AVAL\ (6)}$	0.4941
				$k_{AVAL\ (7)}$	0.4873
				$k_{AVAL\ (8)}$	0.5190
	2		2	$k_{AVAL\ (1)}$	0.5071
				$k_{AVAL\ (2)}$	0.5105
				$k_{AVAL\ (3)}$	0.5037
				$k_{AVAL\ (4)}$	0.4939
				$k_{AVAL\ (5)}$	0.4963
				$k_{AVAL\ (6)}$	0.4988
				$k_{AVAL\ (7)}$	0.5027
				$k_{AVAL\ (8)}$	0.4988
	3		3	$k_{AVAL\ (1)}$	0.4944
				$k_{AVAL\ (2)}$	0.5071
				$k_{AVAL\ (3)}$	0.4929
				$k_{AVAL\ (4)}$	0.4905
				$k_{AVAL\ (5)}$	0.4983
				$k_{AVAL\ (6)}$	0.5002
				$k_{AVAL\ (7)}$	0.5066
				$k_{AVAL\ (8)}$	0.5002
	4		4	$k_{AVAL\ (1)}$	0.4946
				$k_{AVAL\ (2)}$	0.4985
				$k_{AVAL\ (3)}$	0.5039
				$k_{AVAL\ (4)}$	0.5122
				$k_{AVAL\ (5)}$	0.4893
				$k_{AVAL\ (6)}$	0.4956
				$k_{AVAL\ (7)}$	0.5020
				$k_{AVAL\ (8)}$	0.5063
RAF	5		dynamic 3D S-Box	$k_{AVAL\ (1)}$	0.5068
				$k_{AVAL\ (2)}$	0.5010
				$k_{AVAL\ (3)}$	0.5088
				$k_{AVAL\ (4)}$	0.5088

				$k_{AVAL\ (5)}$	0.5205
				$k_{AVAL\ (6)}$	0.4971
				$k_{AVAL\ (7)}$	0.4834
				$k_{AVAL\ (8)}$	0.5186

The results in Table 5.1 indicate that the values of $k_{AVAL}(i)$ approximates to half. This means that the dynamic 3D S-Box in RAF and the S-Boxes in BA do not satisfy the exact AVAL criterion, i.e., these S-Boxes satisfy AVAL only within a range of error. Other experiments have similar results.

Table 5.2 summarize the values of ϵA, the maximum (Max) and the minimum (Min) values of the kAVAL which correspond to the changed input bits *ei* where *i=1...8* with ten random 128-bit Eks and random plaintext (a24a52153c3ede6735e0865e8d99bfbc).

Table 5.2

ϵ_A, Max, and Min values of k_{AVAL} (ten random 128-bit Eks*)*

Name of algorithm	No of experiment	Random 128-bit Eks in Hexadecimal	S-Boxes Sequence	ϵ_A	Max value of $K_{AVA}L$	Min value of k_{AVAL}
BA	1	5a22cf8f5c8b190447fe784467b 2e538	1	0.0381	0.5190	0.4810
	2		2	0.0210	0.5105	0.4895
	3		3	0.0190	0.5095	0.4905
	4		4	0.0244	0.5122	0.4878
RAF	5		dynamic 3D S-Box	0.0410	0.5205	0.4795
BA	6	6ba36e2fe0a4c7840de1537e13 c20ec	1	0.0332	0.5166	0.4834
	7		2	0.0298	0.5149	0.4851
	8		3	0.0361	0.5181	0.4819
	9		4	0.0234	0.5117	0.4883
RAF	10		dynamic 3D S-Box	0.0488	0.5244	0.4756
BA	11	ab4c050208e34cccbae675df09 4ae619	1	0.0156	0.5078	0.4922
	12		2	0.0278	0.5139	0.4861
	13		3	0.0298	0.5149	0.4851
	14		4	0.0137	0.5068	0.4932
RAF	15		dynamic 3D S-Box	0.0321	0.51605	0.48395

BA	16	d48e31d6dec336ff5f34c98bf8ff 088d	1	0.0269	0.5134	0.4866
	17		2	0.0337	0.5168	0.4832
	18		3	0.0464	0.5232	0.4768
	19		4	0.0254	0.5127	0.4873
RAF	20		dynamic 3D S-Box	0.0356	0.5178	0.4822
BA	21	923233d1aafe9e47ee94ba07dc 68bdbd	1	0.0352	0.5176	0.4824
	22		2	0.0278	0.5139	0.4861
	23		3	0.0269	0.5134	0.4866
	24		4	0.0317	0.5159	0.4841
RAF	25		dynamic 3D S-Box	0.0391	0.5195	0.4805
BA	26	7458aa85d6c3c9ef77d07170bb a24fbb	1	0.0239	0.5120	0.4880
	27		2	0.0366	0.5183	0.4817
	28		3	0.0332	0.5166	0.4834
	29		4	0.0269	0.5134	0.4866
RAF	30		dynamic 3D S-Box	0.0566	0.5283	0.4717
BA	31	05605ab55f5cf2eca8781dac2e1 bed6b	1	0.0161	0.5081	0.4919
	32		2	0.0435	0.5217	0.4783
	33		3	0.0215	0.5107	0.4893
	34		4	0.0371	0.5186	0.4814
RAF	35		dynamic 3D S-Box	0.0352	0.5176	0.4824
BA	36	7223 49c1b517cc13292c0b561 08c46	1	0.0283	0.5142	0.4858
	37		2	0.0356	0.5178	0.4822
	38		3	0.0195	0.5098	0.4902
	39		4	0.0347	0.5173	0.4827
RAF	40		dynamic 3D S-Box	0.0261	0.5131	0.4869
BA	41	c49df5e51f2b99736adba91325 33896b	1	0.0239	0.5120	0.4880
	42		2	0.0195	0.5098	0.4902
	43		3	0.0220	0.5110	0.4890
	44		4	0.0273	0.5137	0.4863
RAF	45		dynamic 3D S-Box	0.0366	0.5183	0.4817
BA	46	cc38bd5bacd5eff2f32cfa50519 3c2bf	1	0.0308	0.5154	0.4846
	47		2	0.0264	0.5132	0.4868
	48		3	0.0361	0.5181	0.4819
	49		4	0.0518	0.5259	0.4741
RAF	50		dynamic 3D S-Box	0.0488	0.5244	0.4756

The same previous experiments in Table 5.2 are repeated in Table 5.3. Different low entropy ones 128 bits Ek in hexadecimal (11111111111111111111111111111111) is used.

Table 5.3

ϵ_A, Max, and Min values of k_{AVAL} (Low entropy ones Ek*)*

Name of algorithm	No of experiment	low entropy 128- bit E_k in hexadecimals	S-Boxes sequence	ϵ_A	Max value of k_{AVAL}	Min value of k_{AVAL}
BA	51	111111111111111111111111 11111111	1	0.0278	0.5139	0.4861
	52		2	0.0229	0.5115	0.4885
	53		3	0.0200	0.5100	0.4900
	54		4	0.0200	0.5100	0.4900
RAF	55		dynamic 3D S-Box	0.0264	0.5132	0.4868

The same previous experiments in Table 5.3 are repeated in Table 5.4. Different low entropy zeroes 128 bits Ek in hexadecimal (00000000000000000000000000000000) is used.

Table 5.4

ϵ_A, Max, and Min values of k_{AVAL} (Low entropy zeroes Ek)

Name of algorithm	No of experiment	Low entropy 128 bit E_k in hexadecimals	S-Boxes sequence	ϵ_A	Max value of k_{AVAL}	Min value of k_{AVAL}
BA	56	0000000000000000000000 0000000000	1	0.0195	0.5098	0.4902
	57		2	0.0205	0.5103	0.4897
	58		3	0.0303	0.5151	0.4849
	59		4	0.0249	0.5125	0.4875
RAF	60		dynamic 3D S-Box	0.0229	0.5115	0.4885

Results in Tables 5.2 to 5.4 show that the S-Boxes in BA and in RAF satisfy AVAL with maximum error values (ϵ_{AVAL}) of 0.0518 and 0.0566, respectively. In addition, entropy of Eks bears no effects on the AVAL results.

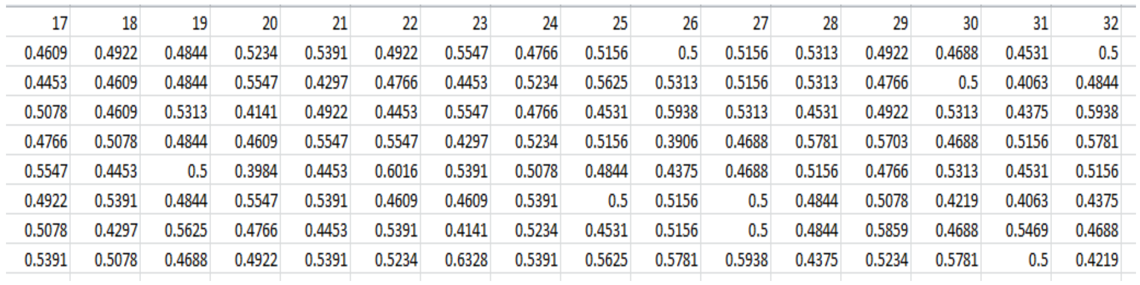
5.2.2 Empirical Results of SAC

Tables 5.5 summarize the values of k_{SAC} *(i, j)* which satisfy Equation (2.17) in BA. The values of k_{SAC} *(i, j)* correspond to the changed input bits (e_i, i= 1…8) where e_1 represents the first changed input bit, e_2 represents the second changed input bit, and subsequently the other parameters e_i (i=3 …8).

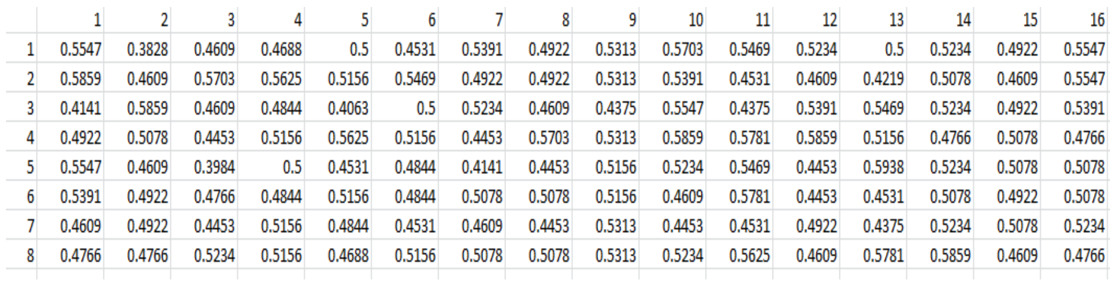
The results of the first S-Box from the first experiment are discussed as follows. This experiment includes SAC values with 8-bit input *(i)* and 32-bit output *(j)* with the first random Ek. The first row indicates the average change in every output bit when the first input bit is changed; the second row shows the average change in every output bit when the second input bit is changed, and so on until the eighth row.

Table 5.5

k_{SAC} *(i, j) with random Ek of the first S-box in BA*



17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
0.4609	0.4922	0.4844	0.5234	0.5391	0.4922	0.5547	0.4766	0.5156	0.5	0.5156	0.5313	0.4922	0.4688	0.4531	0.5
0.4453	0.4609	0.4844	0.5547	0.4297	0.4766	0.4453	0.5234	0.5625	0.5313	0.5156	0.5313	0.4766	0.5	0.4063	0.4844
0.5078	0.4609	0.5313	0.4141	0.4922	0.4453	0.5547	0.4766	0.4531	0.5938	0.5313	0.4531	0.4922	0.5313	0.4375	0.5938
0.4766	0.5078	0.4844	0.4609	0.5547	0.5547	0.4297	0.5234	0.5156	0.3906	0.4688	0.5781	0.5703	0.4688	0.5156	0.5781
0.5547	0.4453	0.5	0.3984	0.4453	0.6016	0.5391	0.5078	0.4844	0.4375	0.4688	0.5156	0.4766	0.5313	0.4531	0.5156
0.4922	0.5391	0.4844	0.5547	0.5391	0.4609	0.4609	0.5391	0.5	0.5156	0.5	0.4844	0.5078	0.4219	0.4063	0.4375
0.5078	0.4297	0.5625	0.4766	0.4453	0.5391	0.4141	0.5234	0.4531	0.5156	0.5	0.4844	0.5859	0.4688	0.5469	0.4688
0.5391	0.5078	0.4688	0.4922	0.5391	0.5234	0.6328	0.5391	0.5625	0.5781	0.5938	0.4375	0.5234	0.5781	0.5	0.4219



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0.5547	0.3828	0.4609	0.4688	0.5	0.4531	0.5391	0.4922	0.5313	0.5703	0.5469	0.5234	0.5	0.5234	0.4922	0.5547
2	0.5859	0.4609	0.5703	0.5625	0.5156	0.5469	0.4922	0.4922	0.5313	0.5391	0.4531	0.4609	0.4219	0.5078	0.4609	0.5547
3	0.4141	0.5859	0.4609	0.4844	0.4063	0.5	0.5234	0.4609	0.4375	0.5547	0.4375	0.5391	0.5469	0.5234	0.4922	0.5391
4	0.4922	0.5078	0.4453	0.5156	0.5625	0.5156	0.4453	0.5703	0.5313	0.5859	0.5781	0.5859	0.5156	0.4766	0.5078	0.4766
5	0.5547	0.4609	0.3984	0.5	0.4531	0.4844	0.4141	0.4453	0.5156	0.5234	0.5469	0.4453	0.5938	0.5234	0.5078	0.5078
6	0.5391	0.4922	0.4766	0.4844	0.5156	0.4844	0.5078	0.5078	0.5156	0.4609	0.5781	0.4453	0.4531	0.5078	0.4922	0.5078
7	0.4609	0.4922	0.4453	0.5156	0.4844	0.4531	0.4609	0.4453	0.5313	0.4453	0.4531	0.4922	0.4375	0.5234	0.5078	0.5234
8	0.4766	0.4766	0.5234	0.5156	0.4688	0.5156	0.5078	0.5078	0.5313	0.5234	0.5625	0.4609	0.5781	0.5859	0.4609	0.4766

Tables 5.6 summarize the values of k_{SAC} *(i, j)* which satisfy Equation (2.17) in RAF. The values of k_{SAC} *(i, j)* correspond to the changed input bits (e_i, *i= 1…8*) where e_1 represents the first changed input bit, e_2 represents the second changed input bit, and subsequently the other parameters e_i *(i=3 …8)*.

The results of the first S-Box from the first experiment are discussed as follows. This experiment includes *SAC* values with 8-bit input *(i)* and 8-bit output *(j)* with the first random Ek. The first row indicates the average change in every output bit when the first input bit is changed; the second row shows the average change in every output bit when the second input bit is changed, and so on until the eighth row.

Table 5.6

SAC of dynamic 3D S-Box in RAF

$K_{SAC(1,j=1..8)}$	0.5078	0.5547	0.4375	0.5156	0.5625	0.4922	0.5000	0.4844
$K_{SAC(2,j=1..8)}$	0.6016	0.4297	0.5000	0.5313	0.5156	0.4922	0.4688	0.4688
$K_{SAC(3,j=1..8)}$	0.5703	0.4609	0.4844	0.5313	0.5000	0.4922	0.4844	0.5469
$K_{SAC(4,j=1..8)}$	0.5078	0.5391	0.4375	0.5313	0.5781	0.5234	0.4688	0.4844
$K_{SAC(5,j=1..8)}$	0.4922	0.4766	0.5781	0.5000	0.5000	0.5391	0.5625	0.5156
$K_{SAC(6,j=1..8)}$	0.5547	0.4766	0.5625	0.4844	0.3906	0.4766	0.4688	0.5625
$K_{SAC(7,j=1..8)}$	0.4766	0.4453	0.5000	0.5313	0.4375	0.4922	0.4688	0.5156
$K_{SAC(8,j=1..8)}$	0.5078	0.5078	0.4063	0.6406	0.5313	0.5078	0.5625	0.4844

Tables 5.5 and 5.6 show that the values of k_{SAC} *(i, j)* with the same random Eks are approximate to one half. This means that the S-Boxes in BA and RAF do not exactly satisfy *SAC*, i.e., the S-Boxes in BA and RAF satisfy *SAC* within an error range.

Table 5.7 summarize the values of ϵ_{SAC}, the maximum (*Max*) and the minimum (*Min*) values of the k_{SAC} which correspond to the changed input bits e_i where *i=1...8* with ten random 128-bit Eks and random plaintext (a24a52153c3ede6735e0865e8d99bfbc), where the fourth to the sixth column summarize the values of ϵ_S, the maximum (*Max*), and the minimum (*Min*) values of k_{SAC}..

Table 5.7

ϵ_S, max, and min of ks_{AC} with random 128-bit Eks

Name of algorithm	No of experiment	S-Boxes sequence	ϵ_S	Max value of k_{SAC}	Min value of k_{SAC}
BA	61	1	0.2656	0.6328	0.3672
	62	2	0.3125	0.6563	0.3438
	63	3	0.2344	0.6172	0.3828
	64	4	0.2188	0.6094	0.3906
RAF	65	dynamic 3D S-Box	0.2813	0.6406	0.3594
BA	66	1	0.2969	0.6484	0.3516
	67	2	0.2656	0.6328	0.3672
	68	3	0.2969	0.6484	0.3516
	69	4	0.2500	0.6250	0.3750
RAF	70	dynamic 3D S-Box	0.2344	0.6172	0.3828
BA	71	1	0.2656	0.6328	0.3672
	72	2	0.2813	0.6406	0.3594
	73	3	0.2656	0.6328	0.3672
	74	4	0.2813	0.6406	0.3594
RAF	75	dynamic 3D S-Box	0.2031	0.6016	0.3984
BA	76	1	0.2656	0.6328	0.3672
	77	2	0.2656	0.6328	0.3672
	78	3	0.2656	0.6328	0.3672
	79	4	0.2656	0.6328	0.3672
RAF	80	dynamic 3D S-Box	0.2656	0.6328	0.3672
BA	81	1	0.2500	0.6250	0.3750
	82	2	0.3125	0.6563	0.3438
	83	3	0.2656	0.6328	0.3672
	84	4	0.2813	0.6406	0.3594
RAF	85	dynamic 3D S-Box	0.2344	0.6172	0.3828
BA	86	1	0.2969	0.6484	0.3516
	87	2	0.2656	0.6328	0.3672
	88	3	0.2656	0.6328	0.3672

	89	4	0.218	0.6094	0.3906
RAF	90	dynamic 3D S-Box	0.2656	0.6328	0.3672
BA	91	1	0.2656	0.6328	0.3672
	92	2	0.2344	0.6172	0.3828
	93	3	0.2656	0.6328	0.3672
	94	4	0.3594	0.6797	0.3203
RAF	95	dynamic 3D S-Box	0.2031	0.6016	0.3984
BA	96	1	0.3125	0.6563	0.3438
	97	2	0.3281	0.6641	0.3359
	98	3	0.2656	0.6328	0.3672
	99	4	0.2500	0.6250	0.3750
RAF	100	dynamic 3D S-Box	0.1875	0.5938	0.4063
BA	101	1	0.2344	0.6172	0.3828
	102	2	0.2500	0.6250	0.3750
	103	3	0.3125	0.6563	0.3438
	104	4	0.2344	0.6172	0.3828
RAF	105	dynamic 3D S-Box	0.2656	0.6328	0.3672
BA	106	1	0.2813	0.6406	0.3594
	107	2	0.2813	0.6406	0.3594
	108	3	0.2500	0.6250	0.3750
	109	4	0.2500	0.6250	0.3750
RAF	110	dynamic 3D S-Box	0.2344	0.6172	0.3828

The same previous experiments in Table 5.7 are repeated in Table 5.8. Different low entropy ones 128 bits E_k in hexadecimal (11111111111111111111111111111111) is used.

Table 5.8

ϵ_S, Max, and Min values of k_{SAC} with Low entropy ones E_k

Name of algorithm	No of experiment	S-Boxes sequence	ϵ_S	Max value of k_{SAC}	Min value of k_{SAC}
BA	116	1	0.2813	0.6406	0.3594
	117	2	0.3281	0.6641	0.3359
	118	3	0.2813	0.6406	0.3594
	119	4	0.2344	0.6172	0.3828
RAF	120	dynamic 3D S-Box	0.2031	0.6016	0.3984

The same previous experiments in Table 5.7 are repeated in Table 5.9. Different low entropy zeroes 128 bits Ek in hexadecimal (00000000000000000000000000000000) is used.

Table 5.9

ϵ_S, Max, and Min values of k_{SAC} with Low entropy zeroes Ek

Name of algorithm	No of experiment	S-Boxes sequence	ϵ_S	Max value of k_{SAC}	Min value of k_{SAC}
BA	111	1	0.2656	0.6328	0.3672
	112	2	0.2188	0.6094	0.3906
	113	3	0.2500	0.6250	0.3750
	114	4	0.2344	0.6172	0.3828
RAF	115	dynamic 3D S-Box	0.1875	0.5938	0.4063

The S-Boxes in BA and in RAF satisfy SAC with a maximum error value (ϵSAC) of 0.3594 and 0.2813, respectively, as shows in Tables 5.7 to 5.9. In addition, the entropy of Eks bears no effect on the SAC results.

5.2.3 Empirical Results of BIC

Table 5.10 summarizes the values of *BIC (i)* which satisfy Equations (2.20) and (2.21), and which correspond to the changed input bits (e_i, $i= 1 \ldots 8$) with ten random 128-bit Eks and random plaintext in hexadecimal (a24a52153c3ede6735e0865e8d99bfbc) in both algorithms. The fourth column indicates to BIC when i-th input bit is changed.

Table 5.10

BIC values with random 128-bit Eks

Name of algorithm	No of experiment	S-Boxes sequence	BIC
BA	121	1	0.3135
	122	2	0.3152
	123	3	0.3285
	124	4	0.3423
RAF	125	dynamic 3D S-Box	0.2698
BA	126	1	0.3598
	127	2	0.3135
	128	3	0.3278
	129	4	0.2972
RAF	130	dynamic 3D S-Box	0.2690
BA	131	1	0.4191
	132	2	0.3774
	133	3	0.3083
	134	4	0.3272
RAF	135	dynamic 3D S-Box	0.2197
BA	136	1	0.3121
	137	2	0.2993
	138	3	0.3219
	139	4	0.3272
RAF	140	dynamic 3D S-Box	0.2646
BA	141	1	0.3111
	142	2	0.3965
	143	3	0.3311
	144	4	0.3777
RAF	145	dynamic 3D S-Box	0.2672
BA	146	1	0.3127
	147	2	0.3591
	148	3	0.3061
	149	4	0.4725
RAF	150	dynamic 3D S-Box	0.2401
BA	151	1	0.3614
	152	2	0.3170
	153	3	0.3450
	154	4	0.3239
RAF	155	dynamic 3D S-Box	0.2694
BA	156	1	0.3311

	157	2	0.3401
	158	3	0.3379
	159	4	0.3298
RAF	160	dynamic 3D S-Box	0.2437
BA	161	1	0.3065
	162	2	0.3007
	163	3	0.3012
	164	4	0.3255
RAF	165	dynamic 3D S-Box	0.2509
BA	166	1	0.3489
	167	2	0.3471
	168	3	0.2975
	169	4	0.3561
RAF	170	dynamic 3D S-Box	0.2437

The same previous experiments in Table 5.10 are repeated in Table 5.11. Different low entropy ones 128 bits Ek in hexadecimal (11111111111111111111111111111111) is used.

Table 5.11
BIC values with Low entropy ones E_k

Name of algorithm	No of experiment	S-Boxes Sequence	BIC
BA	171	1	0.3125
	172	2	0.3909
	173	3	0.3918
	174	4	0.3451
RAF	175	dynamic 3D S-Box	0.2595

The same previous experiments in Table 5.11 are repeated in Table 5.12. Different low entropy zeroes 128 bits Ek in hexadecimal (00000000000000000000000000000000) is used.

Table 5.12

BIC values with Low entropy ones E_k

Name of algorithm	No of experiment	S-Boxes Sequence	BIC
BA	176	1	0.3379
	177	2	0.3127
	178	3	0.3490
	179	4	0.3611
RAF	180	dynamic 3D S-Box	0.2649

From the results in Tables 5.10, 5.11, and 5.12, it can be inferred that the S-Boxes in BA and in RAF satisfy BIC with a maximum error value (ϵBIC) of 0.4725 and 0.2698, respectively. In addition, the entropy of Eks did not affect the BIC results.

Table 5.13

ϵ_{AVAL}, ϵ_{SAC}, and ϵ_{BIC} values

Algorithm & S-Box	ϵ_{AVAL}	ϵ_{SAC}	ϵ_{BIC}
dynamic 3D S-Box in RAF	0.0566	0.2813	0.2698
S-Boxes in BA	0.0518	0.3594	0.4725

In conclusion, all the above results show that the dynamic 3D S-Box in RAF satisfies AVAL, SAC, and BIC with maximum error values of 0.0566, 0.2813, and 0.2698, respectively, whereas the S-Box in BA satisfies AVAL, SAC, and BIC with maximum error values of 0.0518, 0.3594, and 0.4725, respectively. The dynamic 3D S-Box in RAF satisfies AVAL in almost the same that one in BA does, whereas the dynamic 3D S-Box in RAF satisfies SAC and BIC more effectively than the S-Box in BA. This means that the RAF is more secure than the BA. On the other hand, the entropy of the *Eks* does not affect the security of S-Boxes in both algorithms. The above Table 5.13 summarizes ϵ_{AVAL}, ϵ_{SAC}, and ϵ_{BIC} values in both algorithms.

5.3 Results of Correlation Coefficient on dynamic 3D S-Box in RAF

This section presents the empirical results of the correlation coefficient on dynamic 3D S-Box in RAF using uncorrelated random Eks and correlated Eks with three types of plaintext random, low entropy ones and zeroes.

5.3.1 Empirical Results of Uncorrelated Random Eks

Table 5.14 summarizes ten sets of 256-bit random Eks in hexadecimal. The correlation coefficient is computed from each pair of the random Eks after which the correlation coefficient is computed from the resulting dynamic 3D S-Boxes in RAF with each pair of random Eks.

In Table 5.14, the second and third columns indicate the five pairs of the random 256-bit Eks whereas the fourth column indicates the correlation coefficient computed between the random Eks in the second and third columns. Finally, the fifth column indicates the correlation coefficient computed from dynamic 3D S-Boxes resulting from RAF with random plaintext 128-bit in hexadecimal is ‘a24a52153c3ede6735e0865e8d99bfbc’.

Table 5.14

Correlation Coefficient of 3D S-boxes (Random plaintext & uncorrelated Eks*)*

No of experiment	Random Eks (256-bit) in Hexadecimal	Random Eks (256-bit) in Hexadecimal	Correlation Coefficient / random Eks	Correlation Coefficient / dynamic 3D S-Boxes
181	339a59d318b7b357356d26 3acf75e749f8d58d5cf97302 217ba71877898f7719	69d9e968b92366ea79de2a3 408b9e14a58bd0309ecdcaf 436732a14f14ed6444	0.0648	0.0474
182	2bf82a006c928088f71b84c 5caf656a3066b54e844ca85 9f604f6f34e3f18fdc	A10a79cebd3f8715bf483dd f9a0620545eb7781dbea6e6 b4474b1dbdda2305c4	0.0179	-0.0168
183	6d9c74091c2e40b4ae18cbb 4471d87dc3a30880a88829 5204e47808afce8eaba	9e54deb4f9e9384115a11e7 593149ff6385e2460f9cc28 56152424f3bc5365d2	-0.0504	-0.0720
184	A39d8a38bfa7d0ade47f3ad d6b7f7534d1b85f0d3828ad a8ab2cd30a2b9853ae	C7658019f4e4bce3088bb9a 26001f0150f77145fff4355e 71020e038379ee720	0.0050	-0.0023
185	0c11318f33d51b66f7ea168 2dde4760270a1dd844dab6 4634e496e5628f8322d	0c11318f33d51b66f7ea168 2dde4760270a1dd844dab6 4634e496e5628f8322d	-0.0731	-0.0772

The same previous experiments in Table 5.14 are repeated in Table 5.15. Different nonrandom plaintext Low entropy ones (11111111111111111111111111111111) in hexadecimal is used.

Table 5.15

Correlation Coefficient of dynamic 3D S-Boxes (Low entropy zeroes & uncorrelated Eks)

No of experiment	Random Eks (256-bit) in Hexadecimal	Random Eks (256-bit) in Hexadecimal	Correlation Coefficient / random Eks	Correlation Coefficient / dynamic 3D S-Boxes
186	339a59d318b7b357356d263acf75e749f8d58d5cf97302217ba71877898f7719	69d9e968b92366ea79de2a3408b9e14a58bd0309ecdcaf436732a14f14ed6444	0.0648	0.0261
187	2bf82a006c928088f71b84c5caf656a3066b54e844ca859f604f6f34e3f18fdc	A10a79cebd3f8715bf483ddf9a0620545eb7781dbea6e6b4474b1dbdda2305c4	0.0179	0.0067
188	6d9c74091c2e40b4ae18cbb4471d87dc3a30880a888295204e47808afce8eaba	9e54deb4f9e9384115a11e7593149ff6385e2460f9cc2856152424f3bc5365d2	-0.0504	-0.0373
189	A39d8a38bfa7d0ade47f3add6b7f7534d1b85f0d3828ada8ab2cd30a2b9853ae	C7658019f4e4bce3088bb9a26001f0150f77145fff4355e71020e038379ee720	0.0050	0.1245
190	0c11318f33d51b66f7ea1682dde4760270a1dd844dab64634e496e5628f8322d	0c11318f33d51b66f7ea1682dde4760270a1dd844dab64634e496e5628f8322d	-0.0731	0.0443

The same previous experiments in Table 5.14 are repeated in Table 5.16. Different nonrandom plaintext low entropy zeroes (00000000000000000000000000000000) in hexadecimal is used.

Table 5.16

Correlation Coefficient of dynamic 3D S-Boxes (Low entropy ones & uncorrelated Eks*)*

No of experiment	Random Eks (256-bit) in Hexadecimal	Random Eks (256-bit) in Hexadecimal	Correlation Coefficient / random Eks	Correlation Coefficient / dynamic 3D S-Boxes
191	339a59d318b7b357356d263acf75e749f8d58d5cf97302217ba71877898f7719	69d9e968b92366ea79de2a3408b9e14a58bd0309ecdcaf436732a14f14ed6444	0.0648	-0.0263
192	2bf82a006c928088f71b84c5caf656a3066b54e844ca859f604f6f34e3f18fdc	A10a79cebd3f8715bf483ddf9a0620545eb7781dbea6e6b4474b1dbdda2305c4	0.0179	0.0084
193	6d9c74091c2e40b4ae18cbb4471d87dc3a30880a888295204e47808afce8eaba	9e54deb4f9e9384115a11e7593149ff6385e2460f9cc2856152424f3bc5365d2	-0.0504	0.1031
194	A39d8a38bfa7d0ade47f3add6b7f7534d1b85f0d3828ada8ab2cd30a2b9853ae	C7658019f4e4bce3088bb9a26001f0150f77145fff4355e71020e038379ee720	0.0050	0.0380

195	0c11318f33d51b66f7ea1682dde4760270a1dd844dab64634e496e5628f8322d	0c11318f33d51b66f7ea1682dde4760270a1dd844dab64634e496e5628f8322d	-0.0731	0.0044

From the results in Table 5.14, it is noted that all the values of the correlation coefficient of dynamic 3D S-Box are approximately equal to zero that is an indication as of prefect random. In addition the values of the correlation coefficient of dynamic 3D S-Boxes are of the similar values as the correlation coefficients of the random Eks that are generated from a strong generator BBS.

The results in Tables 5.15 and 5.16 show that almost the values are close to zero, which is an indication of prefect random except for two values: 0.1245 (Table 5.15) and 0.1031 (Table 5.16) that indicate weak linearity.

5.3.2 Empirical Results of Correlated Eks

After investigating the effects of uncorrelated random Eks on the security of the dynamic 3D S-Box in RAF with three types of plaintext (random, Low ones and zeroes), the same experiments were repeated with high Correlated Eks to examine the effects of correlated Eks on the security of the dynamic 3D S-Box in RAF with the same types of plaintext (Random, and non–Random (Low entropy ones and zeroes)).

Table 5.17 is a summary of five set of 256-bit random Eks in hexadecimal and changed bits from random bytes that are indicated in yellow. The Correlation Coefficient is computed between each pair of Eks (random Ek with corresponding changed Ek); after which the Correlation Coefficient is computed from the resultant of dynamic 3D S-Boxes with each two pair of Eks (random Ek with corresponding changed Ek).

In Table 5.17 the second column indicates the random Eks of 256-bit; while the third column indicates the changed Eks of 256-bit. Meanwhile, the fourth column indicates the Correlation Coefficient between Eks of the second and third columns. Finally, the fifth column indicates the Correlation Coefficient computed from dynamic 3D S-Boxes i.e result from dynamic 3D S-Box in RAF (random Eks with corresponding changed Eks).

Table 5.17

Correlation Coefficient of dynamic 3D S-Boxes (Random plaintext & correlated Eks*)*

No of experiment	Random Eks (256-bit) in Hexadecimal	Changed Eks (256-bit) in Hexadecimal	Correlation Coefficient / Eks	Correlation Coefficient / dynamic 3D S-Boxes
196	339a59d318b7b357356d263a cf75e749f8d58d5cf97302217 ba71877898f7719	73ba59db18b7b357356d363a cf75e749f8d58d5cf97302217 ba7197789af7718	0.9799	0.0473
197	69d9e968b92366ea79de2a34 08b9e14a58bd0309ecdcaf43 6732a14f14ed6444	eb91e968b92366ea79de2a34 08b9e14a58bd0309ecdcaf4b 6732a14f14ed6445	0.9532	-0.0240
198	2bf82a006c928088f71b84c5 caf656a3066b54e844ca859f6 04f6f34e3f18fdc	2bba2a806c928088f71b84c5 caf656a3066b54e844ca859f6 04f6f34e3f18fdd	0.9688	-0.0285
199	a10a79cebd3f8715bf483ddf9 a0620545eb7781dbea6e6b44 74b1dbdda2305c4	210a79cebd3f8715bf483ddf9 a0620545eb7781dbea6e6b44 f4b1dbdda2305c6	0.9765	-0.0189
200	6d9c74091c2e40b4ae18cbb4 471d87dc3a30880a88829520 4e47808afce8eaba	ad9c74091c2e40b4ae18cbb4 471d87dc3a30880a88829520 4e47808afce8eaba	0.9841	-0.0687

The results in Table 5.17 show that all values of the correlation coefficient are approximately zero, which indicates that these are prefect random in spite of high correlation values (0.9799, 0.9532, 0.9688, 0.9765, and 0.9841) among encryptiys are approximately one that indicate to prefect linearity.

The same previous experiments in Table 5.17 are repeated in Table 5.18. Different nonrandom plaintext low entropy zeroes (11111111111111111111111111111111) in hexadecimal is used.

Table 5.18

Correlation Coefficient of dynamic 3D S-Boxes (Low entropy zeroes & correlated Eks)

No of experiment	Random E_{ks} (256-bit) in Hexadecimal	Changed Eks (256-bit) in Hexadecimal	Correlation Coefficient / Eks	Correlation Coefficient / dynamic 3D S-Boxes
201	339a59d318b7b357356d26a cf75e749f8d58d5cf97302217 7ba71877898f7719	73ba59db18b7b357356d363a cf75e749f8d58d5cf97302217 ba7197789af7718	0.9799	-0.1277
202	69d9e968b92366ea79de2a34 08b9e14a58bd0309ecdcaf43 6732a14f14ed6444	eb91e968b92366ea79de2a34 08b9e14a58bd0309ecdcaf4b 6732a14f14ed6445	0.9532	0.0515
203	2bf82a006c928088f71b84c5 caf656a3066b54e844ca859f6 04f6f34e3f18fdc	2bba2a806c928088f71b84c5 caf656a3066b54e844ca859f6 04f6f34e3f18fdd	0.9688	0.0159
204	a10a79cebd3f8715bf483ddf9 a0620545eb7781dbea6e6b44 74b1dbdda2305c4	210a79cebd3f8715bf483ddf9 a0620545eb7781dbea6e6b44 f4b1dbdda2305c6	0.9765	-0.1321
205	6d9c74091c2e40b4ae18cbb44 71d87dc3a30880a88829520 4e47808afce8eaba	ad9c74091c2e40b4ae18cbb44 71d87dc3a30880a88829520 4e47808afce8eaba	0.9841	-0.0616

The same previous experiments in Table 5.17 are repeated in Table 5.19. Different nonrandom plaintext low entropy zeroes (00000000000000000000000000000000) in hexadecimal is used.

Table 5.19

Correlation Coefficient of dynamic 3D S-Boxes (Low entropy ones & correlated Eks)

No of experiment	Random Eks (256-bit) in Hexadecimal	Changed Eks (256-bit) in Hexadecimal	Correlation Coefficient / Eks	Correlation Coefficient / dynamic 3D S-Boxes
206	339a59d318b7b357356d36 3acf75e749f8d58d5cf973022 17ba71877898f7719	73ba59db18b7b357356d363 acf75e749f8d58d5cf973022 17ba7197789af7718	0.9799	-0.0361
207	69d9e968b92366ea79de2a34 08b9e14a58bd0309ecdcaf4 36732a14f14ed6444	eb91e968b92366ea79de2a34 08b9e14a58bd0309ecdcaf4 b6732a14f14ed6445	0.9532	0.0682
208	2bf82a006c928088f71b84c5 caf656a3066b54e844ca859f 604f6f34e3f18fdc	2bba2a806c928088f71b84c5 caf656a3066b54e844ca859f 604f6f34e3f18fdd	0.9688	0.0328
209	a10a79cebd3f8715bf483ddf 9a0620545eb7781dbea6e6b4 474b1dbdda2305c4	210a79cebd3f8715bf483ddf 9a0620545eb7781dbea6e6b4 4f4b1dbdda2305c6	0.9765	-0.0447
210	6d9c74091c2e40b4ae18cbb4 471d87dc3a30880a8882952 04e47808afce8eaba	ad9c74091c2e40b4ae18cbb4 471d87dc3a30880a8882952 04e47808afce8eaba	0.9841	-0.0157

The results in Tables 5.18 and 5.19 show that all the values of the correlation coefficients are approximate zero that indicates to prefect non-linearity. However, two values (-0.1277,-0.1321) in Table 5.18 indicates weak linearity in spite of high values of the correlation coefficient among the Eks and the plaintext being nonrandom.

Finally, from all the 30 experiments, it can be concluded that the dynamic 3D S-Box shows high non-linearity, and that the randomness of the plaintext and the Eks does not have any effect on the security of the dynamic 3D S-Box.

5.4 Summary

This chapter presents the output of the stage one of the phase 3 from the methodology. This include AVAL, SAC, BIC of the dynamic 3D S-Box and dynamic S-box in BA. As well as the correlation coefficient between the dynamic 3D S-Boxes in RAF.

CHAPTER SIX

EXPERIMENTALRESULTS OF RAF

6.1 Introduction

This chapter, divided into three parts, presents the evaluation results of the RAF output. The first part discusses the results of NIST Test Suit on five types of data (cipher block chaining mode, random plaintext/random 128-bit keys, image files, text files, and video files) on RAF and BA. The second part discusses the results of the avalanche effect and correlation coefficient on both algorithms. Finally, the third part discusses the results of the cryptanalysis.

6.2 Results of RAF Outputs Using NIST

This section presents the empirical results of the RAF outputs using NIST statistical tests on five data set types. The five data set types are explained in details in Section 3.4.2.1.

6.2.1 Empirical Results on Cipher Block Chaining Mode

Figures 6.1 to 6.4 illustrate the results on cipher block chaining mode data set for two pairs of rounds in BA and RAF respectively. In each Figure, the dashed line depicts the smallest proportion that satisfies the 0.01 acceptance criterion based on equation 2.9 is 0.972766312 as mentioned in Section 3.4.2.1, whereas the solid line depicts the expected proportion that is 0.99%. In addition in every Figure the numbers on the horizontal axis represents test ID of 188 NIST statistical test that illustrated in Table 2.5.

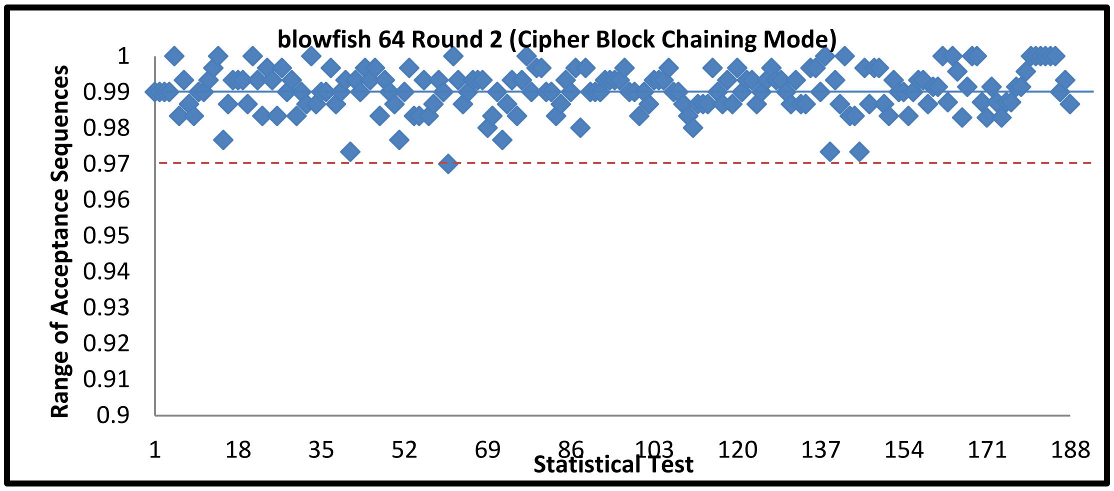


Figure 6.1. Results of Cipher Block Chaining Mode for Round 2 in BA

The result in Figure 6.1 shows that the output of blowfish is random at the end of the second round (the first round pair) on this type of data because majority of the 188 statistical tests show percentage values greater than 97%.

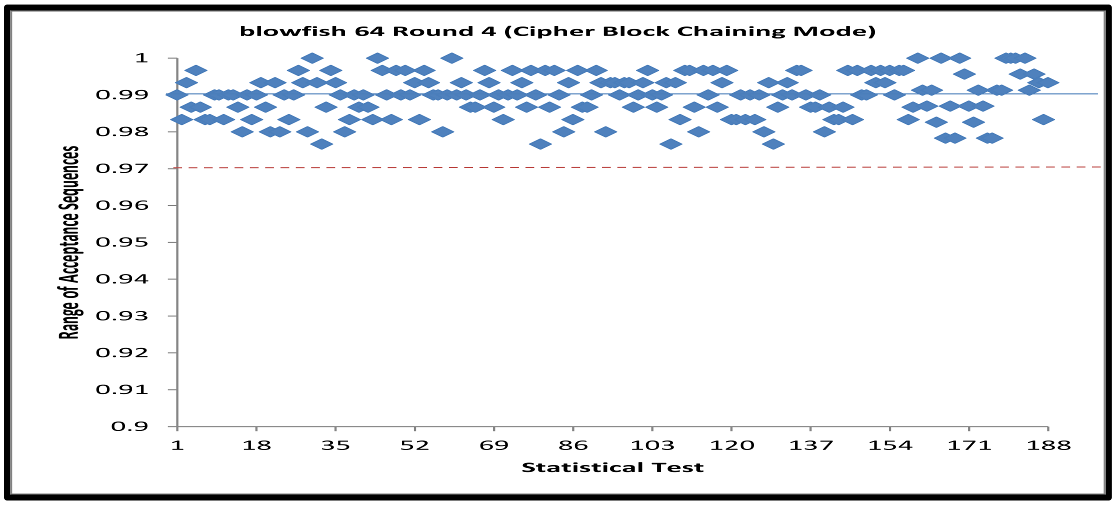


Figure 6.2. Results of Cipher Block Chaining Mode for Round 4 in BA

The result in Figure 6.2 shows that the output of blowfish is random at the end of the fourth round (the second round pair) on this type of data because majority of the 188 statistical tests show percentage values greater than 97%.

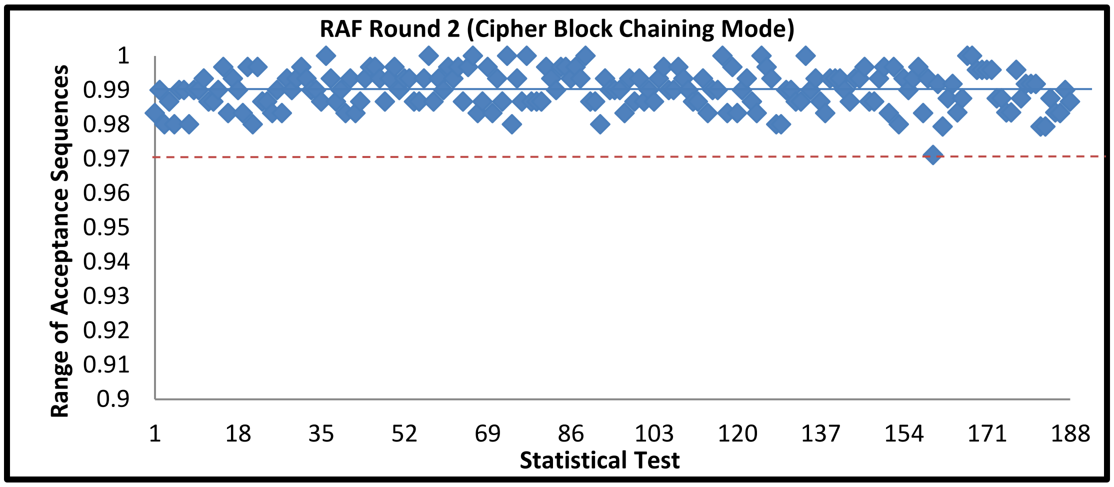


Figure 6.3. Results of Cipher Block Chaining Mode for Round 2 in RAF

The result in Figure 6.3 shows that the output of RAF is random at the end of the second round (the first round pair) on this type of data because majority of the 188 statistical tests show percentage values greater than 97 %.

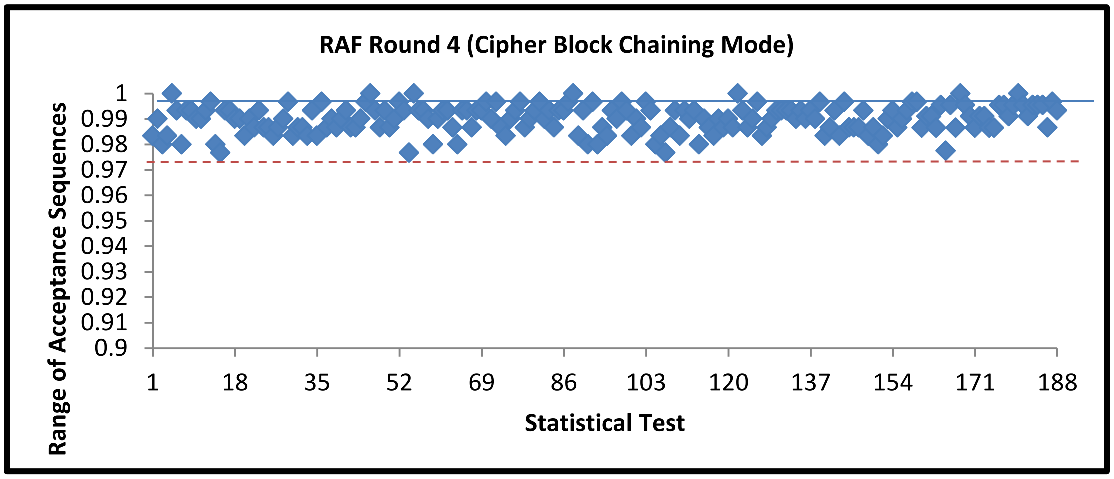


Figure 6.4. Results of Cipher Block Chaining Mode for Round 4 in RAF

The result in Figure 6.4 shows that the output of RAF is random at the end of the fourth round (the second round pair) on this type of data because majority of the 188 statistical tests show percentage values greater than 97%.

From the above results can be concluded the outputs from both algorithms are random at the end of the second round (the first round pair). The subsequent rounds produced similar results (Appendix C).

6.2.2 Empirical Results on Random Plaintext/Random 128-bit keys

Figures 6.5 to 6.8 illustrate the results on the random plaintext/random 128-bit keys data set for two pairs of rounds in BA and RAF respectively.

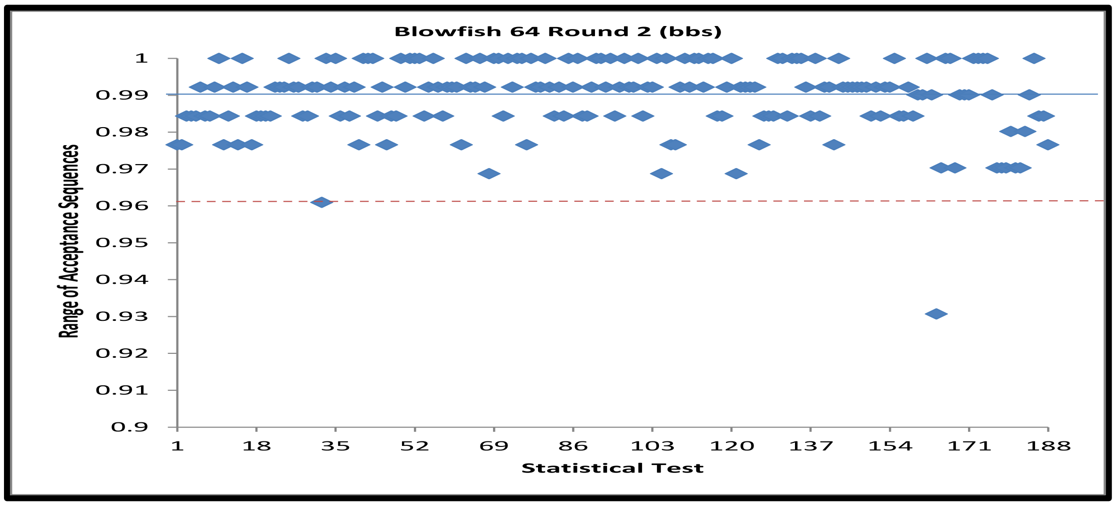


Figure 6.5. Results of Random Plaintext/Random128-bit keys for Round 2 in BA

The result in Figure 6.5 shows that the output of BA is random at the end of the second round (the first round pair) on this type of data because majority of the 188 statistical tests show percentage values greater than 96%.

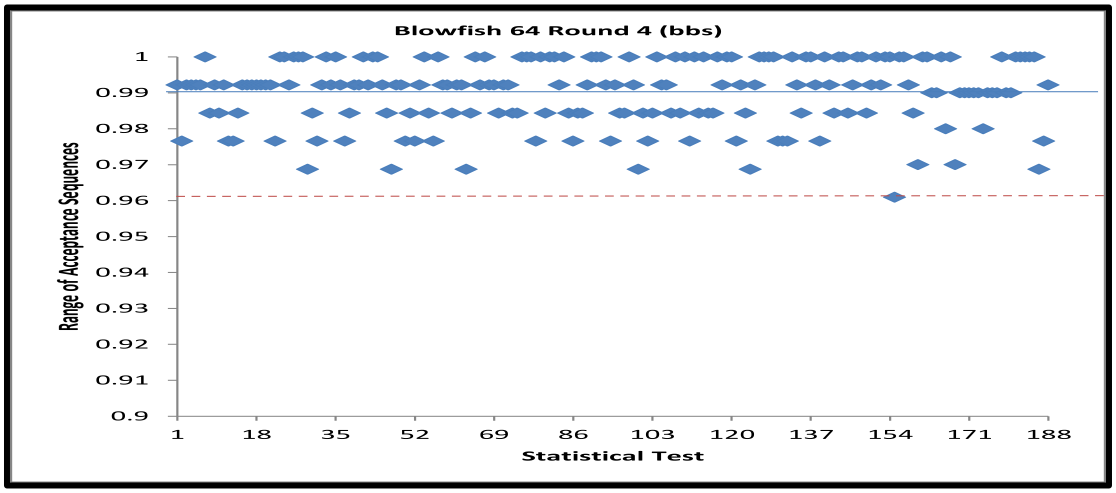


Figure 6.6. Results of Random Plaintext/Random128-bit keys for Round 4 in BA

The result in Figure 6.6 shows that the output of BA is random at the end of the second round (the first round pair) on this type of data because majority of the 188 statistical tests show percentage values greater than 96%.

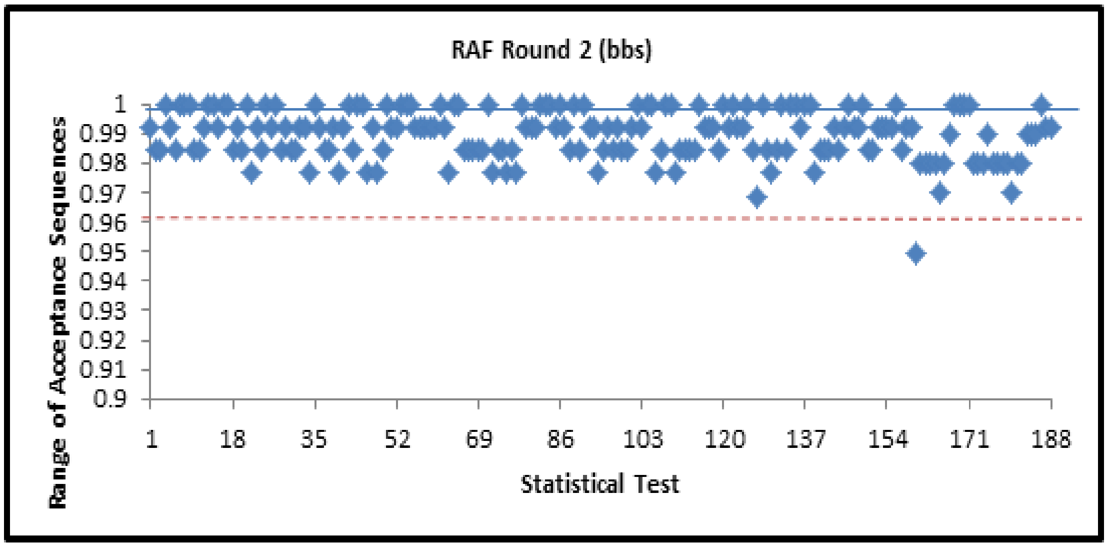


Figure 6.7. Results of Random Plaintext/Random128-bit keys for Round 2 in RAF

The result in Figure 6.7 shows that the output of RAF is random at the end of the second round (the first round pair) on this type of data because majority of the 188 statistical tests show percentage values greater than 96%.

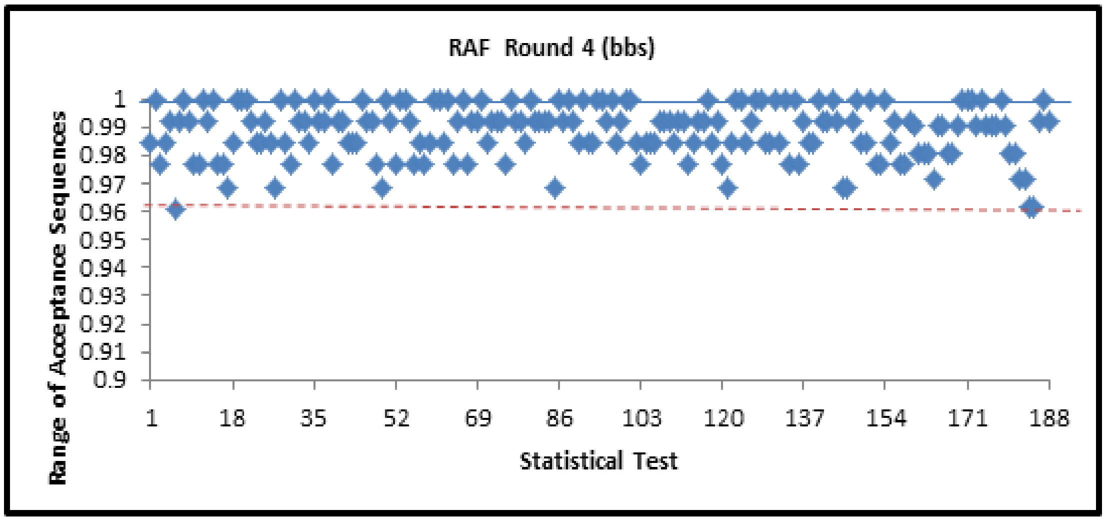


Figure 6.8. Results of Random Plaintext/Random128-bit keys for round 4 in RAF

The result in Figure 6.8 shows that the output of RAF is random at the end of the second round (the first round pair) on this type of data because majority of the 188 statistical tests show percentage values greater than 96%.

From the above results, it can be concluded the outputs from both algorithms are random at the end of the second round (the first round pair). The subsequent rounds produced similar results (Appendix C).

6.2.3 Empirical Results on Image Files

Figures 6.9 to 6.12 illustrate the results on image data set type for two pairs of rounds in the BA and the RAF.

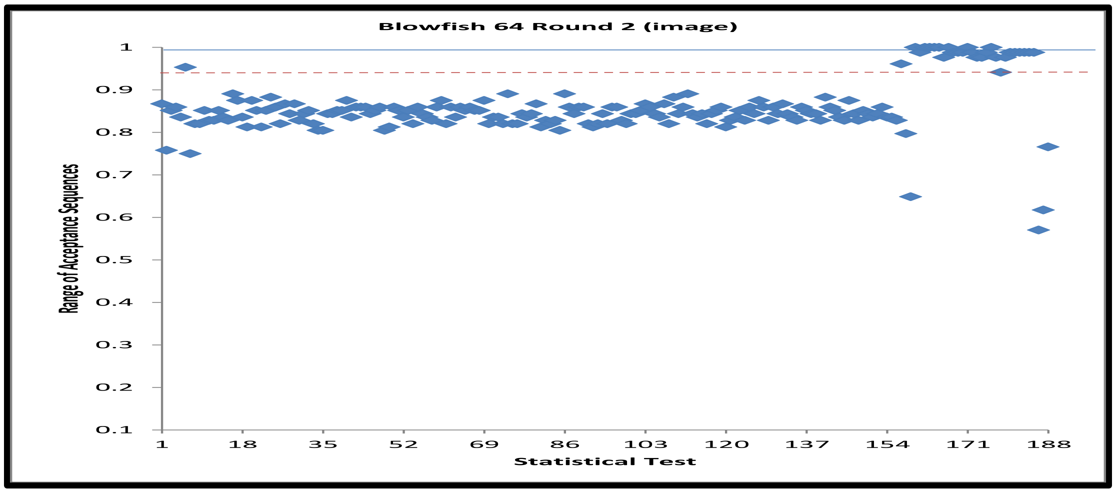


Figure 6.9. Results of image files for Round 2 in BA

The result in Figure 6.9 shows that the output of BA is non-random at the end of the second round (the first round pair) on this type of data because majority of the 188 statistical tests record percentage values less than 96%.

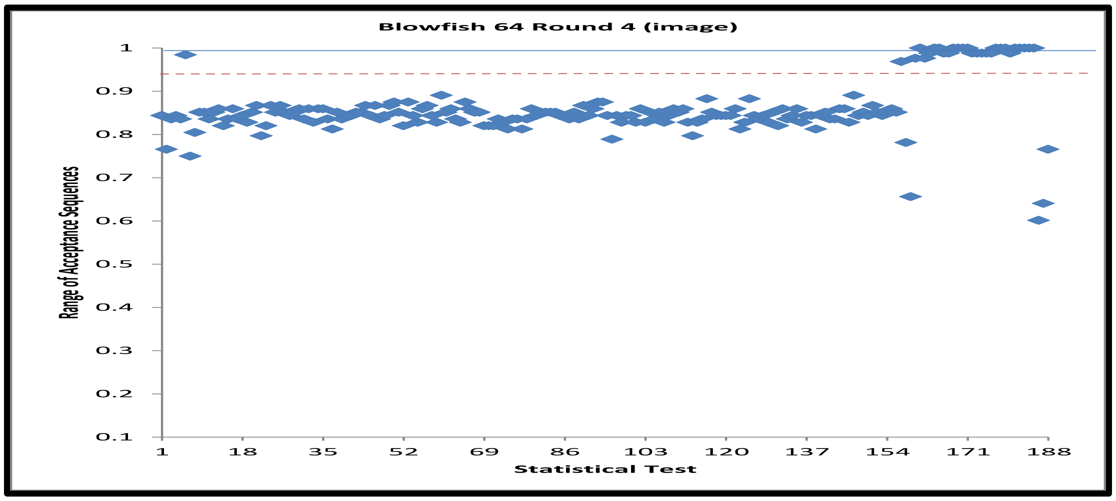


Figure 6.10. Results of image files for Round 4 in BA

The result in Figure 6.10 shows that the output of BA is non-random at the end of the fourth round (the second round pair) on this type of data because majority of the 188 statistical tests record percentage values less than 96%.

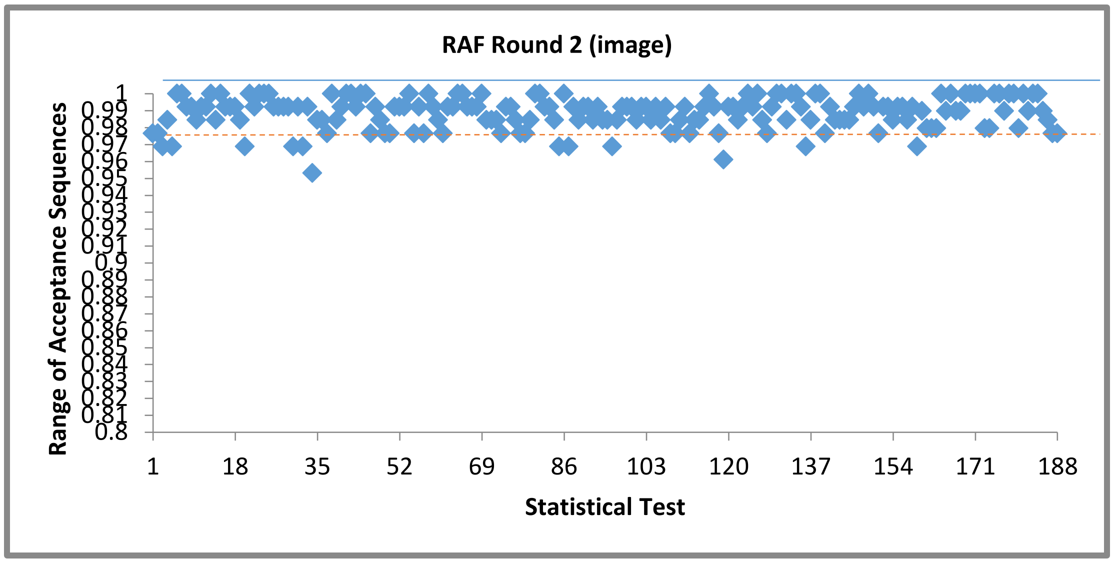


Figure 6.11. Results of image files for Round 2 in RAF

The result in Figure 6.11 shows that the output of RAF is random at the end of the second round (the first round pair) on this type of data this is because majority of the 188 statistical tests show percentage values greater than 96%.

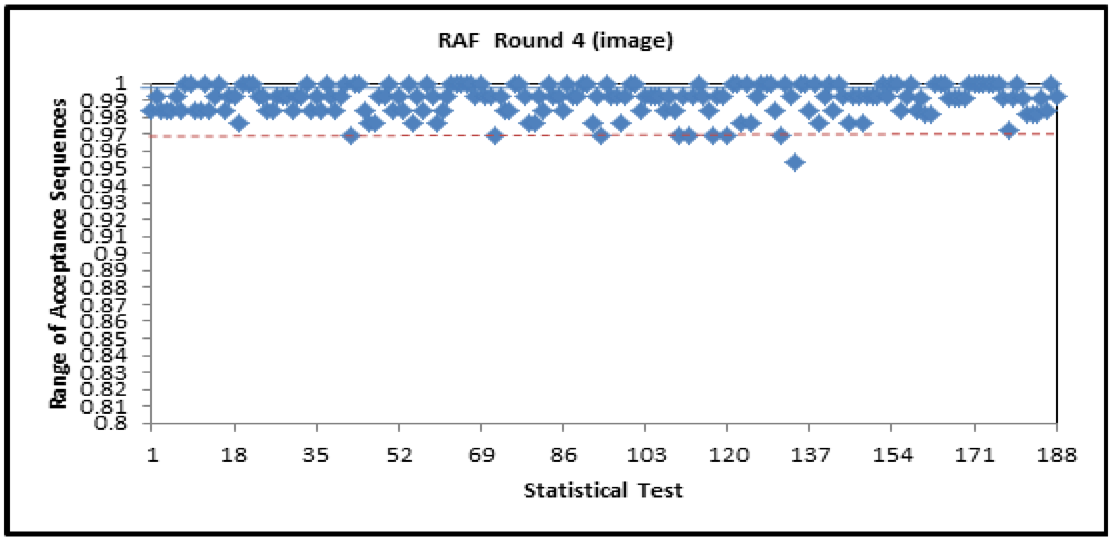


Figure 6.12. *Results of image files for Round 4 in RAF*

The result in Figure 6.12 shows that the output of RAF is random at the end of the fourth round (the second round pair) on this type of data because majority of the 188 statistical tests show percentage values greater than 96%.

From the above results, it can be concluded that the output from BA is non-random at the end of the second round (the first round pair). The subsequent rounds produced similar results (Appendix C). The reason for this failure is some types of image have many identical bytes that lead to the failure of BA. These images are bitmap (uncompressed) and lossless (gif, png, and tif) images. Lossy compression images such as jpeg pass the NIST statistical test.

Whereas the output from RAF is non-random at the end of the second round (the first round pair). However, it becomes random at the end of the fourth round (second round pair). The subsequent rounds produce similar results (Appendix C). RAF presents good randomness with any type of the images even when using uncompressed images. The reason for this success is RAF has dynamic 3D S-Box with every block, thus, leading to a success. RAF is able to encrypt any type of data even those with large identical bytes.

6.2.4 Empirical Results of Text Files

Figures 6.13 to 6.16 illustrate the results on text data set type for two pairs of rounds.

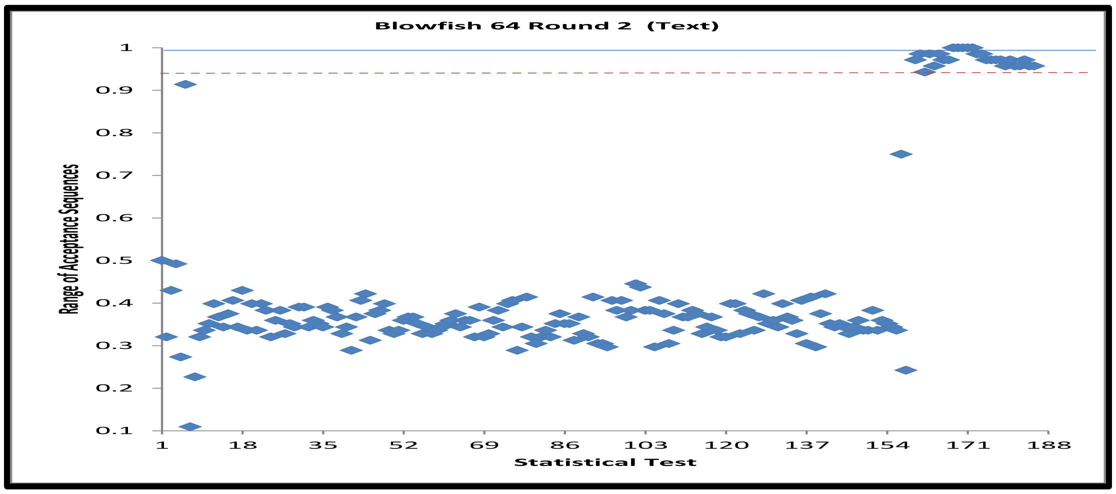


Figure 6.13. Results of text files for Round 2 in BA

The result in Figure 6.13 shows that the output of BA is non-random at the end of the second round (the first round pair) on this type of data because majority of the 188 statistical tests record percentage values less than 96%.

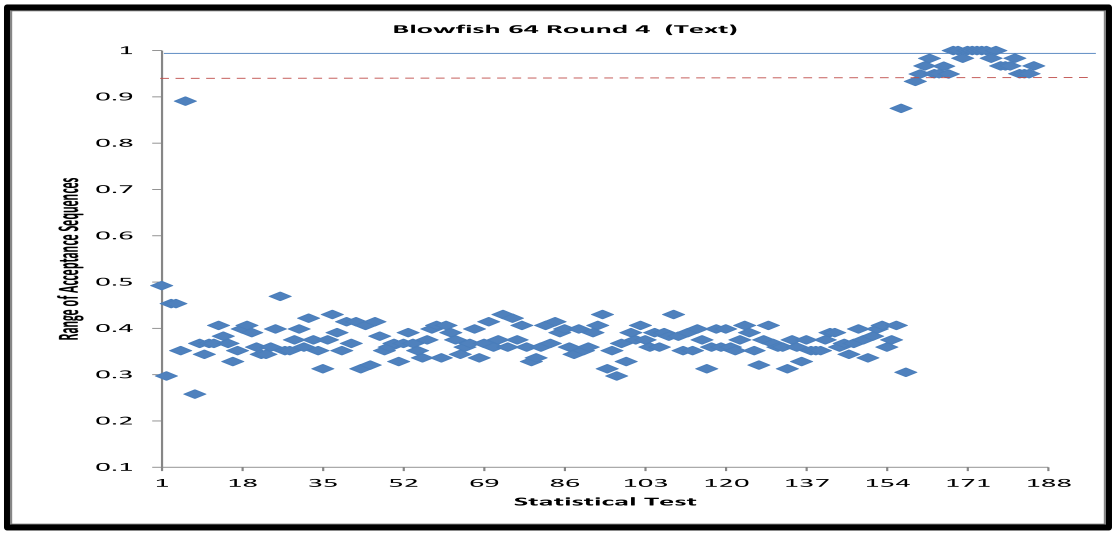


Figure 6.14. Results of text files for Round 4 in BA

The result in Figure 6.14 shows that the output of BA is still non-random at the end of the fourth round (the second round pair) on this type of data because majority of the 188 statistical tests record percentage values less than 96%.

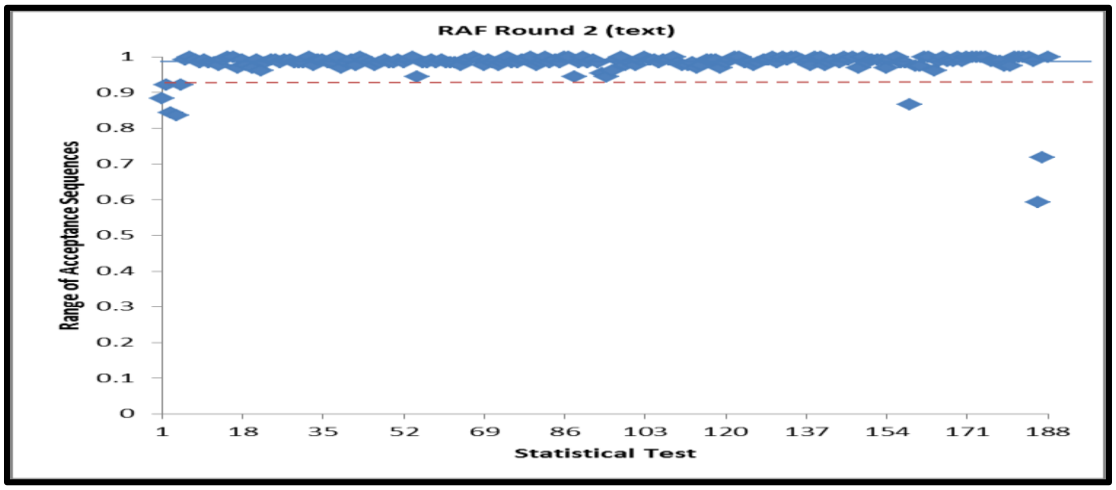


Figure 6.15. Results of text files for Round 2 in RAF

The result in Figure 6.15 shows that the output of RAF is non- random at the end of the second round (the first round pair) on this type of data because some of the 188 statistical tests yield proportion less than 96%.

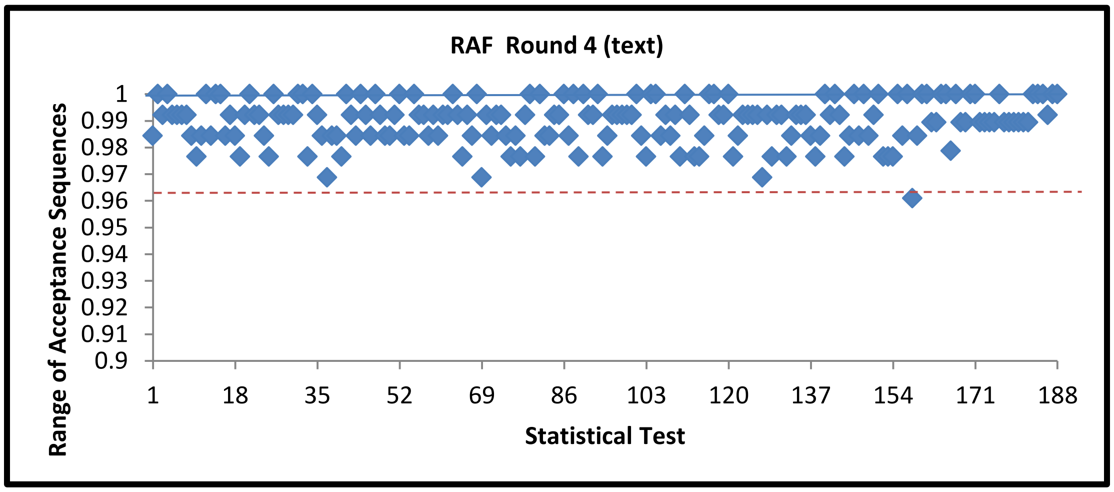


Figure 6.16. Results of text files for Round 4 in RAF

The result in Figure 6.16 shows that the output of RAF is random at the end of the fourth round (the second round pair) on this type of data because majority of the 188 statistical tests show percentage values greater than 96%.

From the above results, it can be concluded the outputs from BA is non-random at the end of the second round (the first round pair). The subsequent rounds produced similar results (Appendix C). The reason of this failure the text data set has many identical bytes leads to fail BA in NIST statistical test. Whereas the output from RAF is non-random at the end of the second round (the first round pair) However, it becomes random at the end of the fourth round (second round pair). The subsequent rounds produce similar results (Appendix C). The reason of this successful in spite of using text data set has many identical bytes leads to RAF has dynamic 3D S-Box with every block. Thus leads to success RAF to encryption any type of data even have large identical bytes.

6.2.5 Empirical Results of Video Files

Figures 6.17 to 6.20 illustrate the results on video data set type for two pairs of rounds.

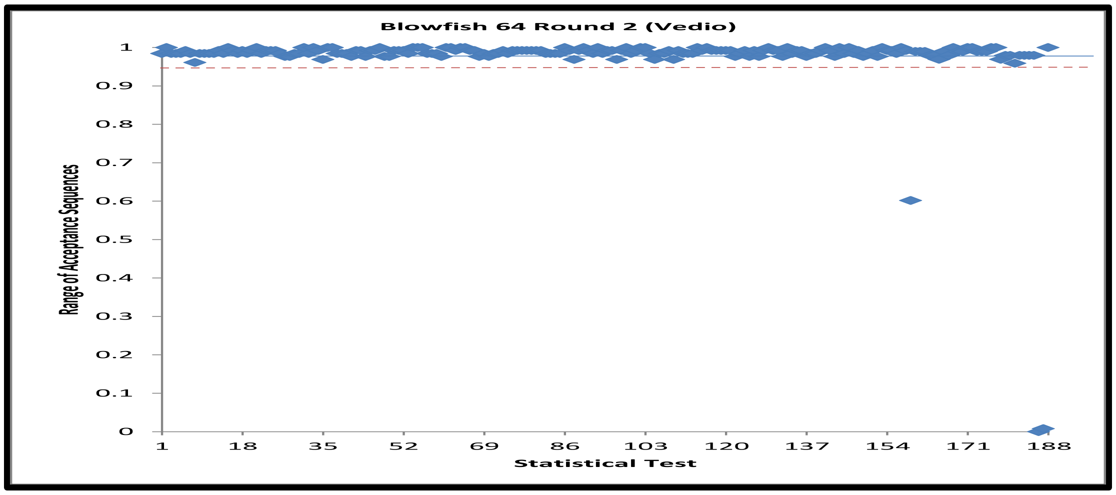


Figure 6.17. Results of video files for Round 2 in BA

From The result in Figure 6.17, It is clear the output from BA on this type of data is random at the end of the second round (the first round pair) because majority of the 188 statistical tests record values greater than 96% except for Approximate Entropy (159) and serial (186, 187) statistical tests which record less than 96%.

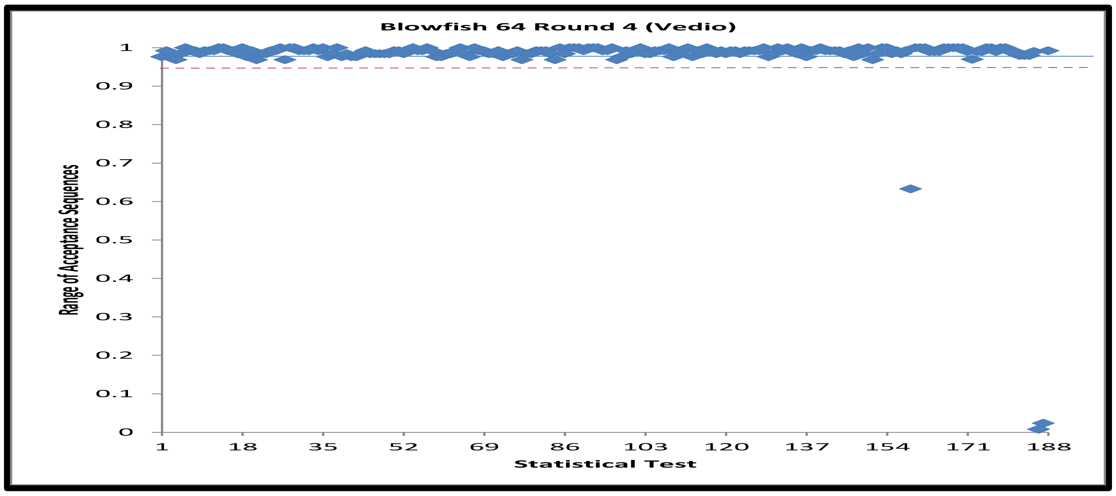


Figure 6.18. Results of video files for Round 4 in BA

From The result in Figure 6.18, It is clear the output from BA on this type of data is still random at the end of the fourth round (the second round pair) because majority of the 188 statistical tests record values greater than 96% except for Approximate Entropy (159) and serial (186, 187) statistical tests which record less than 96%.

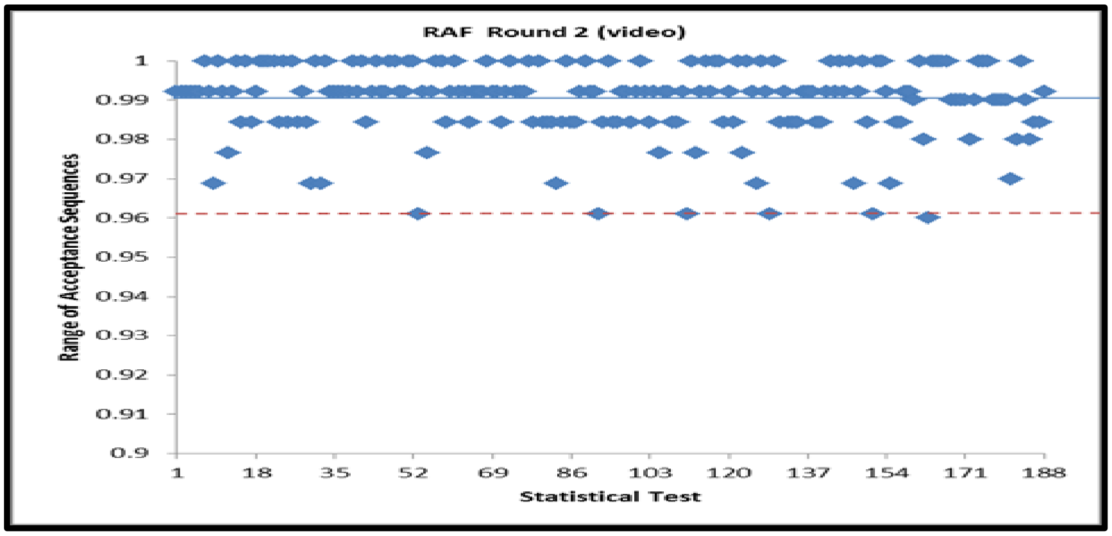


Figure 6.19. Results of video files for Round 2 in RAF

The result in Figure 6.19 shows that the output of RAF is random at the end of the second round (the first round pair) on this type of data because majority of the 188 statistical tests show percentage values greater than 96%.

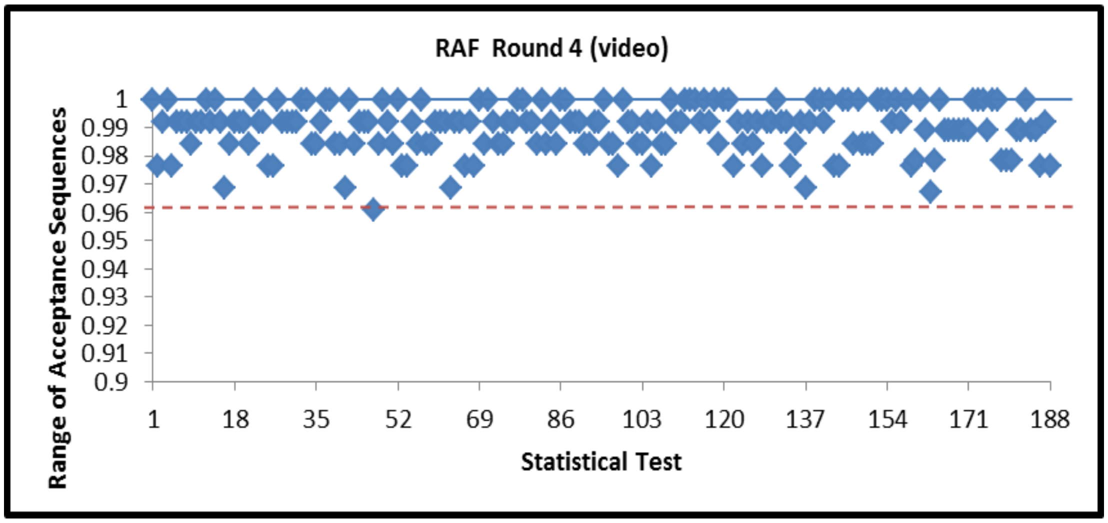


Figure 6.20. Results of video files for Round 4 in RAF

The result in Figure 6.20 shows that the output of RAF is random at the end of the fourth round (the second round pair) on this type of data because majority of the 188 statistical tests show percentage values greater than 96%.

From the above results, it can be concluded the output from BA is random at the end of the second round (the first round pair). The subsequent rounds produced similar results (Appendix C). Whereas the output from RAF is random at the end of the second round (the first round pair). Subsequent rounds also produced similar results (Appendix C).

From all the results above in this section, it can be concluded that BA is not suitable for image and text files with large strings of identical bytes whereas the RAF algorithm is suitable for encrypting all file types despite the large strings of identical bytes.

6.3 Results of RAF Output Using Avalanche Text and Correlation Coefficient

This section presents the empirical results of the RAF outputs using avalanche text and correlation coefficient.

6.3.1 Empirical Results on Avalanche Text

Tables 6.1 to 6.4 summarize the values of the avalanche text for the first to three rounds, as well as for the ciphertext in both RAF and BA, respectively. In every Table, the different bits number (BA) and different bit number (RAF) columns show that the number of bits in output varies when one bit in the plaintext is changed. The ratio bits (BA) and ratio bits (RAF) columns, on the other hand, show the ratios or the different bits number divided by the total number of bits sequence.

Table 6.1

Avalanche text for both algorithms in the first round

No. of Seq.	Different bits number (RAF)	Ratio (RAF)	Different bits number (BA)	Ratio (BA)	No. of Seq.	Different bits number (RAF)	Ratio (RAF)	Different bit number (BA)	Ratio (BA)
1	34	0.2656	13	0.2031	38	34	0.2656	15	0.2344
2	26	0.2031	20	0.3125	39	33	0.2578	16	0.2500
3	40	0.3125	12	0.1875	40	27	0.2109	18	0.2813
4	38	0.2969	18	0.2813	41	27	0.2109	16	0.2500
5	28	0.2188	17	0.2656	42	32	0.2500	19	0.2969
6	21	0.1641	18	0.2813	43	34	0.2656	16	0.2500
7	35	0.2734	15	0.2344	44	29	0.2266	15	0.2344
8	36	0.2813	16	0.2500	45	30	0.2344	18	0.2813
9	38	0.2969	14	0.2188	46	33	0.2578	10	0.1563
10	37	0.2891	12	0.1875	47	36	0.2813	22	0.3438
11	37	0.2891	19	0.2969	48	32	0.2500	22	0.3438
12	31	0.2422	16	0.2500	49	31	0.2422	20	0.3125
13	39	0.3047	21	0.3281	50	26	0.2031	11	0.1719
14	34	0.2656	12	0.1875	51	34	0.2656	14	0.2188
15	37	0.2891	17	0.2656	52	38	0.2969	17	0.2656
16	32	0.2500	20	0.3125	53	39	0.3047	12	0.1875
17	35	0.2734	17	0.2656	54	37	0.2891	21	0.3281
18	34	0.2656	14	0.2188	55	31	0.2422	24	0.3750
19	32	0.2500	16	0.2500	56	35	0.2734	13	0.2031
20	22	0.1719	18	0.2813	57	34	0.2656	19	0.2969
21	27	0.2109	17	0.2656	58	29	0.2266	20	0.3125
22	33	0.2578	14	0.2188	59	36	0.2813	16	0.2500
23	38	0.2969	17	0.2656	60	36	0.2813	10	0.1563
24	29	0.2266	17	0.2656	61	29	0.2266	16	0.2500
25	31	0.2422	21	0.3281	62	37	0.2891	17	0.2656
26	33	0.2578	18	0.2813	63	28	0.2188	17	0.2656
27	33	0.2578	15	0.2344	64	34	0.2656	15	0.2344
28	35	0.2734	18	0.2813	65	39	0.3047	12	0.1875
29	32	0.2500	18	0.2813	66	26	0.2031	20	0.3125
30	37	0.2891	19	0.2969	67	27	0.2109	14	0.2188
31	38	0.2969	20	0.3125	68	41	0.3203	13	0.2031
32	26	0.2031	24	0.3750	69	30	0.2344	22	0.3438
33	29	0.2266	16	0.2500	70	38	0.2969	20	0.3125
34	30	0.2344	12	0.1875	71	29	0.2266	19	0.2969
35	29	0.2266	16	0.2500	72	32	0.2500	16	0.2500
36	34	0.2656	19	0.2969	73	28	0.2188	19	0.2969
37	28	0.2188	17	0.2656	74	34	0.2656	16	0.2500

No. of Seq.	Different bits Number (RAF)	Ratio (RAF)	Different bits Number (BA)	Ratio (BA)
75	31	0.2422	16	0.2500
76	37	0.2891	16	0.2500
77	38	0.2969	21	0.3281
81	38	0.2969	17	0.2656
82	35	0.2734	18	0.2813
83	30	0.2344	21	0.3281
84	38	0.2969	15	0.2344
85	37	0.2891	17	0.2656
86	32	0.2500	15	0.2344
87	31	0.2422	16	0.2500
88	31	0.2422	23	0.3594
89	30	0.2344	15	0.2344
90	32	0.2500	16	0.2500
91	25	0.1953	21	0.3281
92	33	0.2578	18	0.2813
93	32	0.2500	21	0.3281
94	26	0.2031	19	0.2969
95	34	0.2656	19	0.2969
96	40	0.3125	20	0.3125
97	32	0.2500	18	0.2813
98	35	0.2734	17	0.2656
99	28	0.2188	12	0.1875
100	38	0.2969	14	0.2188
101	25	0.1953	16	0.2500
102	27	0.2109	18	0.2813
103	33	0.2578	19	0.2969
104	29	0.2266	16	0.2500
105	35	0.2734	17	0.2656
106	31	0.2422	16	0.2500
107	33	0.2578	18	0.2813
108	29	0.2266	17	0.2656
109	36	0.2813	18	0.2813
110	31	0.2422	18	0.2813
111	31	0.2422	20	0.3125
112	31	0.2422	17	0.2656
113	34	0.2656	15	0.2344
114	27	0.2109	12	0.1875
115	34	0.2656	17	0.2656
116	34	0.2656	15	0.2344
117	38	0.2969	17	0.2656
118	38	0.2969	21	0.3281
119	39	0.3047	17	0.2656
120	31	0.2422	17	0.2656

121	28	0.2188	32	0.5000
122	36	0.2813	23	0.3594
123	33	0.2578	22	0.3438
124	31	0.2422	19	0.2969
125	31	0.2422	18	0.2813
126	41	0.3203	17	0.2656
127	33	0.2578	17	0.2656
128	34	0.2656	13	0.2031

A one-bit change in the input causes a change on an approximately a quarter of the output bits in the first round in both algorithms, as may be noted from the results in Table 6.1. The average change of bits in RAF and BA are 0.2690 and 0.2555, respectively, which means that the avalanche text for both algorithms is rated “not good” for the first round.

Table 6.2

Avalanche text for both algorithms in the second round

No of Seq.	Differen t bits number (RAF)	Ratio (RAF)	Different bits number (BA)	Ratio (BA)	No of Seq.	Different bits number (RAF)	Ratio (RAF)	Different bits number (BA)	Ratio (BA)
1	62	0.4844	26	0.4063	39	66	0.5156	29	0.4531
2	57	0.4453	37	0.5781	40	60	0.4688	33	0.5156
3	68	0.5313	27	0.4219	41	56	0.4375	32	0.5000
4	72	0.5625	33	0.5156	42	61	0.4766	31	0.4844
5	58	0.4531	28	0.4375	43	64	0.5000	30	0.4688
6	49	0.3828	38	0.5938	44	65	0.5078	30	0.4688
7	74	0.5781	32	0.5000	45	66	0.5156	31	0.4844
8	74	0.5781	31	0.4844	46	63	0.4922	25	0.3906
9	71	0.5547	29	0.4531	47	60	0.4688	37	0.5781
10	70	0.5469	24	0.3750	48	58	0.4531	40	0.6250
11	66	0.5156	36	0.5625	49	66	0.5156	33	0.5156
12	60	0.4688	35	0.5469	50	54	0.4219	29	0.4531
13	68	0.5313	42	0.6563	51	64	0.5000	30	0.4688
14	68	0.5313	27	0.4219	52	72	0.5625	36	0.5625
15	62	0.4844	34	0.5313	53	68	0.5313	28	0.4375
16	55	0.4297	38	0.5938	54	66	0.5156	39	0.6094
17	65	0.5078	31	0.4844	55	59	0.4609	43	0.6719
18	64	0.5000	32	0.5000	56	56	0.4375	28	0.4375
19	54	0.4219	29	0.4531	57	63	0.4922	39	0.6094
20	57	0.4453	38	0.5938	58	58	0.4531	33	0.5156
21	57	0.4453	35	0.5469	59	67	0.5234	29	0.4531
22	67	0.5234	29	0.4531	60	66	0.5156	29	0.4531
23	65	0.5078	33	0.5156	61	65	0.5078	30	0.4688
24	64	0.5000	40	0.6250	62	70	0.5469	28	0.4375
25	64	0.5000	41	0.6406	63	54	0.4219	32	0.5000
26	60	0.4688	35	0.5469	64	61	0.4766	29	0.4531
27	65	0.5078	29	0.4531	65	75	0.5859	24	0.3750
28	65	0.5078	40	0.6250	66	58	0.4531	37	0.5781
29	69	0.5391	31	0.4844	67	61	0.4766	31	0.4844
30	63	0.4922	37	0.5781	68	71	0.5547	27	0.4219
31	67	0.5234	34	0.5313	69	61	0.4766	39	0.6094
32	58	0.4531	39	0.6094	70	65	0.5078	31	0.4844
33	71	0.5547	29	0.4531	71	63	0.4922	38	0.5938
34	62	0.4844	26	0.4063	72	65	0.5078	30	0.4688
35	59	0.4609	28	0.4375	73	51	0.3984	34	0.5313
36	60	0.4688	31	0.4844	74	62	0.4844	30	0.4688
37	60	0.4688	36	0.5625	75	63	0.4922	34	0.5313
38	62	0.4844	29	0.4531	76	72	0.5625	34	0.5313

o of Seq.	Different bits number (RAF)	Ratio (RAF)	Different bits number (BA)	Ratio (BA)
77	62	0.4844	32	0.5000
78	64	0.5000	27	0.4219
79	57	0.4453	32	0.5000
80	63	0.4922	36	0.5625
81	65	0.5078	31	0.4844
82	64	0.5000	30	0.4688
83	62	0.4844	33	0.5156
84	59	0.4609	30	0.4688
85	63	0.4922	26	0.4063
86	70	0.5469	32	0.5000
87	66	0.5156	33	0.5156
88	59	0.4609	39	0.6094
89	59	0.4609	29	0.4531
90	62	0.4844	29	0.4531
91	53	0.4141	30	0.4688
92	61	0.4766	36	0.5625
93	70	0.5469	33	0.5156
94	48	0.3750	33	0.5156
95	57	0.4453	34	0.5313
96	74	0.5781	38	0.5938
97	64	0.5000	34	0.5313
98	62	0.4844	32	0.5000
99	54	0.4219	30	0.4688
100	69	0.5391	31	0.4844
101	52	0.4063	31	0.4844
102	53	0.4141	31	0.4844
103	65	0.5078	35	0.5469
104	59	0.4609	34	0.5313
105	64	0.5000	32	0.5000
106	53	0.4141	36	0.5625
107	66	0.5156	32	0.5000
108	59	0.4609	34	0.5313
109	68	0.5313	36	0.5625
110	66	0.5156	31	0.4844
111	66	0.5156	33	0.5156
112	67	0.5234	32	0.5000
113	66	0.5156	33	0.5156
114	60	0.4688	26	0.4063
115	59	0.4609	33	0.5156
116	63	0.4922	30	0.4688
117	62	0.4844	32	0.5000
118	70	0.5469	38	0.5938

119	67	0.5234	38	0.5938
120	64	0.5000	36	0.5625
121	60	0.4688	34	0.5313
122	71	0.5547	45	0.7031
123	66	0.5156	42	0.6563
124	61	0.4766	31	0.4844
125	60	0.4688	32	0.5000
126	69	0.5391	37	0.5781
127	58	0.4531	35	0.5469
128	60	0.4688	26	0.4063

From the results shown in Table 6.2, it can be noted that a one-bit change in the input causes a change on approximately half of the output bits in the second round in both algorithms. The average change of bits in RAF and BA are 0.4912 and 0.5110, respectively, which means that the avalanche text for both algorithms is rated “good” for the second round.

Table 6.3

Avalanche text for both algorithms in the third round

No of Seq.	Different bits number (RAF)	Ratio (RAF)	Different bits number (BA)	Ratio (BA)	No of Seq.	Different bits number (RAF)	Ratio (RAF)	Different bits number (BA)	Ratio (BA)
1	59	0.4609	31	0.4844	39	67	0.5234	30	0.4688
2	61	0.4766	33	0.5156	40	63	0.4922	27	0.4219
3	63	0.4922	33	0.5156	41	65	0.5078	32	0.5000
4	62	0.4844	34	0.5313	42	68	0.5313	32	0.5000
5	65	0.5078	28	0.4375	43	60	0.4688	29	0.4531
6	59	0.4609	35	0.5469	44	71	0.5547	32	0.5000
7	70	0.5469	36	0.5625	45	72	0.5625	31	0.4844
8	71	0.5547	32	0.5000	46	72	0.5625	25	0.3906
9	62	0.4844	32	0.5000	47	58	0.4531	34	0.5313
10	64	0.5000	30	0.4688	48	62	0.4844	35	0.5469
11	66	0.5156	36	0.5625	49	72	0.5625	35	0.5469
12	63	0.4922	35	0.5469	50	58	0.4531	33	0.5156
13	66	0.5156	35	0.5469	51	67	0.5234	36	0.5625
14	70	0.5469	31	0.4844	52	70	0.5469	34	0.5313
15	56	0.4375	35	0.5469	53	63	0.4922	33	0.5156
16	57	0.4453	38	0.5938	54	59	0.4609	38	0.5938
17	62	0.4844	26	0.4063	55	60	0.4688	37	0.5781
18	57	0.4453	29	0.4531	56	53	0.4141	31	0.4844
19	54	0.4219	30	0.4688	57	64	0.5000	36	0.5625
20	63	0.4922	38	0.5938	58	67	0.5234	29	0.4531
21	62	0.4844	38	0.5938	59	59	0.4609	28	0.4375
22	68	0.5313	30	0.4688	60	62	0.4844	36	0.5625
23	60	0.4688	34	0.5313	61	70	0.5469	31	0.4844
24	67	0.5234	43	0.6719	62	73	0.5703	27	0.4219
25	63	0.4922	38	0.5938	63	62	0.4844	36	0.5625
26	59	0.4609	39	0.6094	64	60	0.4688	32	0.5000
27	59	0.4609	30	0.4688	65	68	0.5313	27	0.4219
28	62	0.4844	42	0.6563	66	71	0.5547	36	0.5625
29	70	0.5469	24	0.3750	67	60	0.4688	38	0.5938
30	58	0.4531	34	0.5313	68	66	0.5156	34	0.5313
31	58	0.4531	33	0.5156	69	66	0.5156	36	0.5625
32	67	0.5234	29	0.4531	70	65	0.5078	28	0.4375
33	71	0.5547	25	0.3906	71	67	0.5234	28	0.4375
34	63	0.4922	30	0.4688	72	72	0.5625	30	0.4688
35	64	0.5000	24	0.3750	73	62	0.4844	35	0.5469
36	61	0.4766	34	0.5313	74	56	0.4375	31	0.4844
37	66	0.5156	36	0.5625	75	63	0.4922	38	0.5938
38	52	0.4063	34	0.5313	76	63	0.4922	34	0.5313

No of Seq.	Different bits number (RAF)	Ratio (RAF)	Different bits number (BA)	Ratio (BA)
77	57	0.4453	30	0.4688
78	55	0.4297	31	0.4844
79	61	0.4766	34	0.5313
80	61	0.4766	34	0.5313
81	64	0.5000	36	0.5625
82	65	0.5078	27	0.4219
83	64	0.5000	31	0.4844
84	53	0.4141	32	0.5000
85	54	0.4219	23	0.3594
86	76	0.5938	34	0.5313
87	70	0.5469	37	0.5781
88	69	0.5391	31	0.4844
89	64	0.5000	33	0.5156
90	71	0.5547	33	0.5156
91	60	0.4688	24	0.3750
92	60	0.4688	36	0.5625
93	71	0.5547	32	0.5000
94	53	0.4141	29	0.4531
95	59	0.4609	34	0.5313
96	62	0.4844	34	0.5313
97	65	0.5078	31	0.4844
98	62	0.4844	34	0.5313
99	60	0.4688	32	0.5000
100	61	0.4766	35	0.5469
101	58	0.4531	31	0.4844
102	55	0.4297	28	0.4375
103	58	0.4531	31	0.4844
104	60	0.4688	35	0.5469
105	67	0.5234	29	0.4531
106	50	0.3906	39	0.6094
107	64	0.5000	32	0.5000
108	68	0.5313	34	0.5313
109	64	0.5000	38	0.5938
110	63	0.4922	30	0.4688
111	69	0.5391	29	0.4531
112	68	0.5313	31	0.4844
113	60	0.4688	35	0.5469
114	67	0.5234	33	0.5156
115	58	0.4531	31	0.4844
116	61	0.4766	28	0.4375
117	60	0.4688	25	0.3906
118	62	0.4844	34	0.5313

119	65	0.5078	36	0.5625
120	61	0.4766	36	0.5625
121	64	0.5000	35	0.5469
122	67	0.5234	42	0.6563
123	69	0.5391	36	0.5625
124	59	0.4609	31	0.4844
125	66	0.5156	35	0.5469
126	62	0.4844	33	0.5156
127	56	0.4375	33	0.5156
128	56	0.4375	31	0.4844

From the results shown in Table 6.3, it can be noted that a one-bit change in the input causes a change on approximately half of the output bits in the third round in both algorithms. The average change of bits in RAF and BA are 0.4926 and 0.5098, respectively, which means that the avalanche text for both algorithms is rated “good” for the third round.

Table 6.4

Avalanche text for both algorithms in the ciphertext

No of Seq.	Different bits number (RAF)	Ratio (RAF)	Different bits Number (BA)	Ratio (BA)	No of Seq.	Different bits Number (RAF)	Ratio (RAF)	Different bits number (BA)	Ratio (BA)
1	60	0.4688	27	0.4219	39	66	0.5156	35	0.5469
2	54	0.4219	41	0.6406	40	67	0.5234	30	0.4688
3	68	0.5313	34	0.5313	41	70	0.5469	31	0.4844
4	59	0.4609	31	0.4844	42	75	0.5859	28	0.4375
5	64	0.5000	32	0.5000	43	64	0.5000	34	0.5313
6	64	0.5000	28	0.4375	44	60	0.4688	31	0.4844
7	68	0.5313	33	0.5156	45	63	0.4922	27	0.4219
8	64	0.5000	34	0.5313	46	61	0.4766	31	0.4844
9	57	0.4453	26	0.4063	47	64	0.5000	26	0.4063
10	59	0.4609	39	0.6094	48	65	0.5078	27	0.4219
11	66	0.5156	24	0.3750	49	54	0.4219	37	0.5781
12	55	0.4297	35	0.5469	50	64	0.5000	38	0.5938
13	64	0.5000	32	0.5000	51	60	0.4688	34	0.5313
14	59	0.4609	28	0.4375	52	66	0.5156	27	0.4219
15	62	0.4844	38	0.5938	53	55	0.4297	30	0.4688
16	70	0.5469	27	0.4219	54	62	0.4844	34	0.5313
17	67	0.5234	28	0.4375	55	63	0.4922	29	0.4531
18	55	0.4297	35	0.5469	56	62	0.4844	30	0.4688
19	63	0.4922	32	0.5000	57	62	0.4844	33	0.5156
20	67	0.5234	28	0.4375	58	69	0.5391	31	0.4844
21	67	0.5234	29	0.4531	59	63	0.4922	29	0.4531
22	69	0.5391	38	0.5938	60	65	0.5078	31	0.4844
23	61	0.4766	33	0.5156	61	74	0.5781	41	0.6406
24	67	0.5234	36	0.5625	62	62	0.4844	34	0.5313
25	66	0.5156	31	0.4844	63	67	0.5234	28	0.4375
26	62	0.4844	24	0.3750	64	60	0.4688	33	0.5156
27	65	0.5078	33	0.5156	65	73	0.5703	27	0.4219
28	64	0.5000	29	0.4531	66	65	0.5078	32	0.5000
29	68	0.5313	34	0.5313	67	67	0.5234	28	0.4375
30	63	0.4922	36	0.5625	68	67	0.5234	26	0.4063
31	61	0.4766	35	0.5469	69	58	0.4531	36	0.5625
32	71	0.5547	32	0.5000	70	69	0.5391	35	0.5469
33	61	0.4766	31	0.4844	71	59	0.4609	29	0.4531
34	57	0.4453	33	0.5156	72	56	0.4375	30	0.4688
35	69	0.5391	30	0.4688	73	64	0.5000	27	0.4219
36	72	0.5625	27	0.4219	74	56	0.4375	33	0.5156
37	60	0.4688	28	0.4375	75	58	0.4531	34	0.5313
38	67	0.5234	37	0.5781	76	62	0.4844	30	0.4688

No of Seq.	Different bits Number (RAF)	Ratio (RAF)	Different bits Number (BA)	Ratio (BA)
77	78	0.6094	30	0.4688
78	66	0.5156	33	0.5156
79	67	0.5234	32	0.5000
80	68	0.5313	36	0.5625
81	64	0.5000	36	0.5625
82	68	0.5313	25	0.3906
83	66	0.5156	40	0.6250
84	55	0.4297	31	0.4844
85	54	0.4219	30	0.4688
86	56	0.4375	31	0.4844
87	61	0.4766	32	0.5000
88	65	0.5078	30	0.4688
89	60	0.4688	35	0.5469
90	65	0.5078	33	0.5156
91	71	0.5547	33	0.5156
92	67	0.5234	40	0.6250
93	63	0.4922	23	0.3594
94	70	0.5469	34	0.5313
95	61	0.4766	30	0.4688
96	63	0.4922	31	0.4844
97	54	0.4219	37	0.5781
98	60	0.4688	30	0.4688
99	64	0.5000	29	0.4531
100	67	0.5234	32	0.5000
101	53	0.4141	30	0.4688
102	66	0.5156	36	0.5625
103	73	0.5703	29	0.4531
104	63	0.4922	36	0.5625
105	64	0.5000	30	0.4688
106	63	0.4922	39	0.6094
107	62	0.4844	26	0.4063
108	70	0.5469	34	0.5313
109	71	0.5547	31	0.4844
110	65	0.5078	37	0.5781
111	55	0.4297	36	0.5625
112	49	0.3828	29	0.4531
113	63	0.4922	35	0.5469
114	61	0.4766	26	0.4063
115	57	0.4453	31	0.4844
116	58	0.4531	29	0.4531
117	61	0.4766	34	0.5313
118	56	0.4375	28	0.4375
119	60	0.4688	38	0.5938

120	63	0.4922	26	0.4063
121	65	0.5078	35	0.5469
122	59	0.4609	32	0.5000
123	66	0.5156	35	0.5469
124	65	0.5078	36	0.5625
125	62	0.4844	34	0.5313
126	61	0.4766	30	0.4688
127	63	0.4922	36	0.5625
128	71	0.5547	28	0.4375

From the results shown in Table 6.4, it can be noted that a one-bit change in the input causes a change on approximately half of the output bits in ciphertext in both algorithms. The average change of bits in RAF and BA are 0.4950 and 0.4972, respectively, which means that the avalanche text for both algorithms is rated “good” for the ciphertext.

From all the results presented above, it can be observed that the avalanche text of RAF is almost similar to that of BA. Moreover, the avalanche text for both algorithms is rated as “good” from the second round.

Figures 6.21 to 6.24 illustrate the results of avalanche text in both algorithms from the first to the third rounds, as well as in the ciphertext.

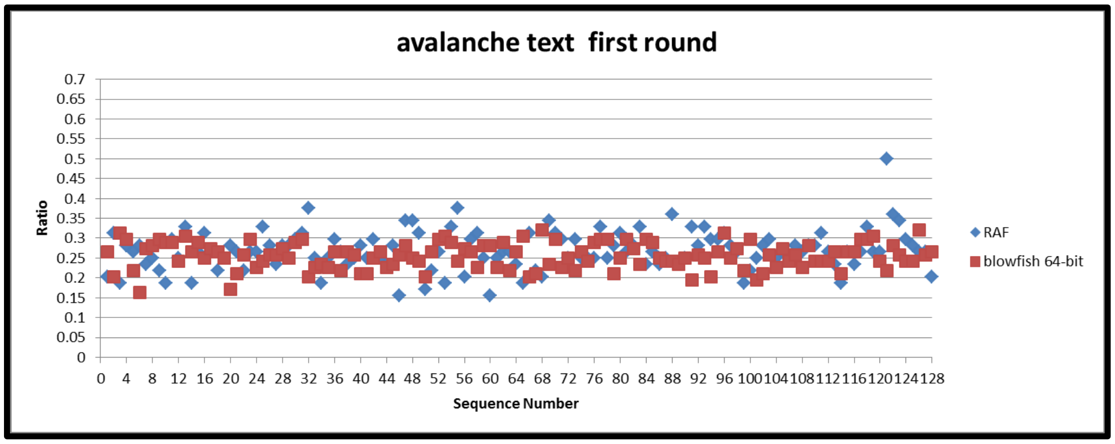


Figure 6.21. Results of the avalanche text of both algorithms for the first round

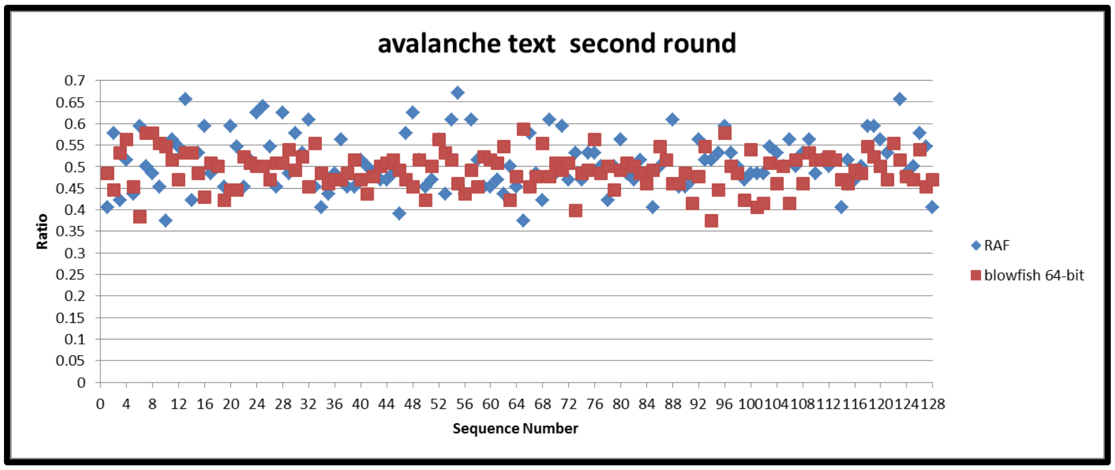


Figure 6.22. Results of the avalanche text of both algorithms for the second round

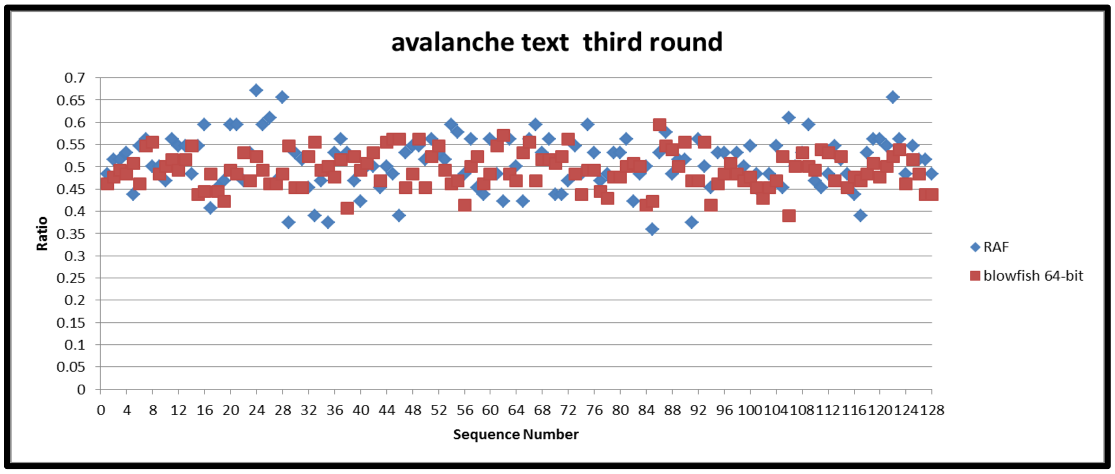


Figure 6.23. Results of the avalanche text of both algorithms for the third round

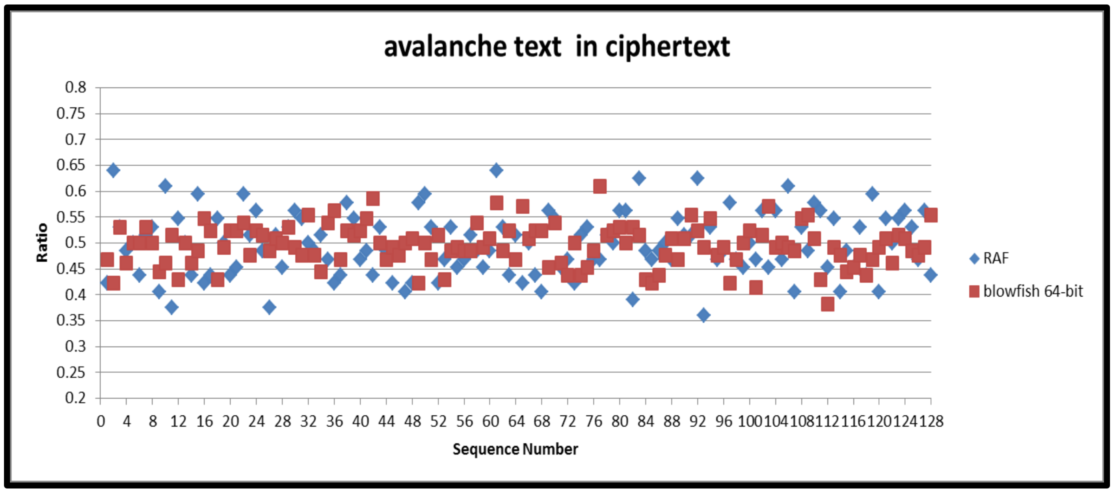


Figure 6.24. Results of the avalanche text of both algorithms for the ciphertext

6.3.2 Empirical Results and Analysis on Correlation Coefficient

From the results in Table 6.5, it can be noted that 87 values of the correlation coefficient in RAF approximates to zero, indicating a prefect non-linear relationship between plaintext and ciphertext. However, 41 values are greater than 0.1 and less than 0.3, as well as less than -0.1 and greater than -0.3, thereby indicating a weak linear positive or linear negative relationship. In BA, 80 values approximate to zero which is an indication of a non-linear relationship between the input and the output. One value (-0.3974) indicates a moderate negative linear relationship. However, 47 values are greater than 0.1 and less than 0.3, as well as less than -0.1 and greater than -0.3. This category indicates a weak positive or weak negative linear relationship. Despite these, both algorithms show good non-linear relationships, although the RAF shows better-impact non-linear relationships than the BA. Figure 6.25 illustrates the correlation results for both algorithms.

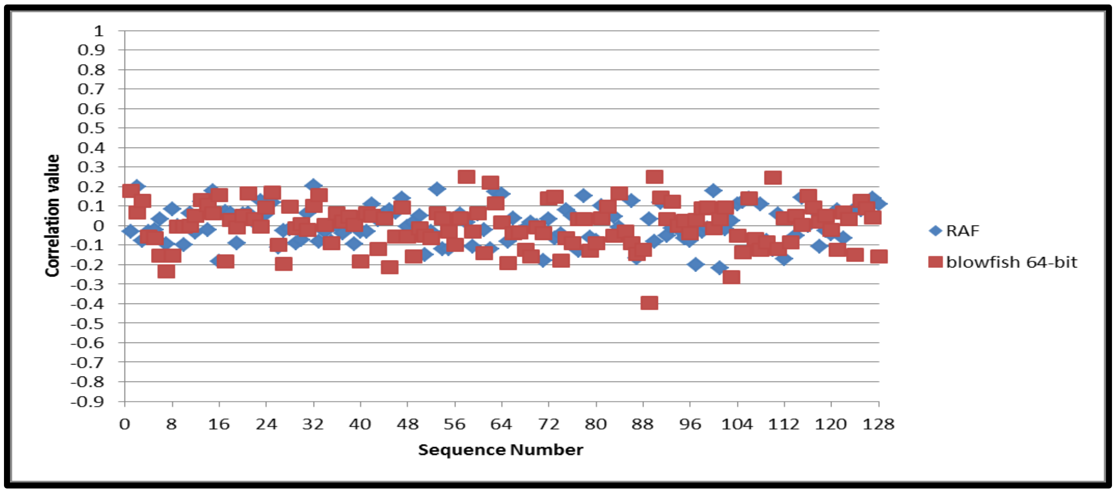


Figure 6.25. Results of correlation of both algorithms

Table 6.5

Correlation Coefficient between plaintext and ciphertext in both algorithms

No of seq.	RAF	BA	Sequence Number	RAF	BA
1	-0.029	0.1758	40	-0.0313	-0.1815
2	0.1998	0.0667	41	-0.0306	0.0636
3	-0.0781	0.1268	42	0.1106	0.0518
4	-0.0297	-0.0552	43	0.0277	-0.1177
5	-0.0201	-0.0658	44	0.0614	0.0355
6	0.0315	-0.1534	45	0.0809	-0.2136
7	-0.0928	-0.2333	46	0.0688	-0.0571
8	0.0854	-0.1536	47	0.1381	0.0949
9	-0.002	-0.0029	48	-0.0029	-0.0542
10	-0.0972	-0.0029	49	0.0156	-0.1569
11	0.0618	-0.0010	50	0.0495	-0.0120
12	-0.035	0.0518	51	-0.1491	-0.0499
13	0.1222	0.1316	52	-0.032	-0.0626
14	-0.0201	0.1068	53	0.1869	0.0616
15	0.1776	0.0625	54	-0.1216	0.0373
16	-0.1851	0.1554	55	-0.1216	-0.0316
17	0.0729	-0.1850	56	0.0285	-0.0977
18	0.0652	0.0294	57	0.0573	0.0390
19	-0.0893	-0.0104	58	0.0181	0.2505
20	0.0573	0.0518	59	-0.1083	-0.0316
21	0.0639	0.1631	60	0.0591	0.0626
22	0.0166	0.0325	61	-0.0237	-0.1420
23	0.1251	-0.0070	62	-0.1209	0.2197
24	0.0609	0.0920	63	0.1738	0.1150
25	0.1171	0.1700	64	0.1588	0.0159
26	-0.1122	-0.1003	65	-0.0834	-0.1914
27	-0.0262	-0.1962	66	0.0373	-0.0373
28	-0.0171	0.0985	67	-0.0366	-0.0333
29	-0.0918	-0.0143	68	-0.0129	-0.1250
30	-0.0739	0.0089	69	0.0157	-0.1563
31	0.0646	-0.0211	70	-0.00098	-0.0079
32	0.202	0.1002	71	-0.1805	-0.0378
33	-0.08	0.1563	72	0.0314	0.1395
34	-0.032	0.0049	73	-0.0628	0.1486
35	-0.0807	-0.0885	74	-0.0417	-0.1794
36	0.0012	0.0635	75	0.0821	-0.0630
37	-0.0388	0.0218	76	0.0507	-0.0882
38	0.0142	0.0479	77	-0.1266	0.0316
39	-0.0929	0.0049	78	0.1513	0.0314

79	-0.0612	-0.1291	104	0.1083	-0.0499
80	-0.0753	-0.0906	105	0.1216	-0.1378
81	0.1034	0.0373	106	0.1408	0.1395
82	0.0499	0.0959	107	-0.1034	-0.0686
83	0.0476	-0.0499	108	0.111	-0.1252
84	-0.0089	0.1651	109	-0.0787	-0.0852
85	-0.047	-0.0313	110	-0.1248	0.2471
86	0.1256	-0.0891	111	0.0606	-0.1183
87	-0.167	-0.1467	112	-0.1692	0.0373
88	-0.1287	-0.1252	113	0.0455	-0.0854
89	0.0315	-0.3974	114	-0.0511	0.0518
90	-0.0797	0.2520	115	0.1423	0.0039
91	0.1196	0.1420	116	0.000244	0.1532
92	-0.0518	0.0333	117	0.0807	0.0949
93	-0.0127	0.1227	118	-0.105	0.0198
94	-0.0156	-0.0010	119	-0.0249	0.0499
95	-0.0628	0.0256	120	-0.0422	-0.0218
96	-0.0752	-0.0404	121	0.0821	-0.1227
97	-0.1985	0.0294	122	-0.0648	0.0683
98	-0.032	0.0888	123	0.0578	0.0313
99	-0.0127	0.0938	124	0.0825	-0.1479
100	0.178	-0.0120	125	0.0784	0.1268
101	-0.2189	0.0333	126	0.1002	0.0885
102	-0.0161	0.0920	127	0.1398	0.0401
103	0.0253	-0.2633	128	0.1086	-0.1569

6.4 Cryptanalysis

This section presents the explanation on the resistance of RAF on differential, linear, and short attacks.

6.4.1 Differential and Linear Attacks

The use of dynamic 3D S-Box and dynamic P-Box is a significant feature of the RAF. The dynamic 3D S-Box and dynamic P-Box protect the algorithm against differential and linear cryptanalysis. The specific properties of the dynamic 3D S-Box, specifically the structure that is completely unknown to the cryptanalyst, assist and support the RAF for it to remain resistant against attacks. The dynamic P-Box, meanwhile, is introduced to provide security and protection to the output of the 3D S-Box. Together, these two components frustrate all linear and differential trails. The 3D S-Box in RAF is not only dynamic but also changeable in every round with every block of plaintext, such that in encryption one block of plaintext, forty dynamic 3D S-Boxes are generated. This means that the 3D S-Box is not a fixed entity, and any attempts to construct iterative linear or differential relations between rounds such as the previous attacks on the original algorithm BA in (Vaudenay, 1995; Nakahara, 2007) lead to frustrations.

6.4.2 Short Attack

Shortcut attack is the trial disclosure of the cryptographic design's internal structure (Hosseinkhani & Javadi, 2012; Ritter, 1991; Zhang & Chen, 2008; Krishnamurthy & Ramaswamy, 2009; Elkamchouchi & Elshafee, 2003; Ritter, 1990). The RAF is resistant to existing shortcut attacks which try to recover the specific internal structure of the cipher. The 3D S-Box in RAF has 4 x 8 x 8 entries with probability of repeating some of the elements. A cryptanalyst wanting to decode the RAF needs to generate all possible 3D S-Box and P-Box values, which is an almost impossible task because the cryptanalysis must attempt all possible cases $(2^n)^{n^2}$ with each section of 3D S-Box. Moreover, four sections are present in 3D S-Box of RAF; therefore, the cryptanalyst must attempt all possible cases $((2^n)^{n^2})^4 = 2^{4n^3}$ where *n* is 8. Each of the ten rounds has four different 3D S-Boxes, thereby creating a total of 40 different 3D S-Box for RAF. This means that the cryptanalyst's must try a total of $(((2^{4n^3})^4)^{10}$ times the first block, and because the 3D S-Box in RAF changes with each encryption process of the plaintext block, the cryptanalyst must try again the same computation

$(((2^{4n^3})^4)^{10}$ that equals to 2^{81920} for every block of plaintext. In other words, if there are five blocks of plaintext, the cryptanalysis must perform five times of 2^{81920}. Moreover, one dynamic P-Box in every round had $2^6!$ possibilities. For 10 rounds, therefore, 10 dynamic P-Boxes with possibility $(2^6!)^{10}$ for just one block of plaintext exist, thereby making the RAF a very secure algorithm. Attackers, therefore, would pay a higher more price. By contrast, S-Boxes of BA with entries 0...255 force the attacker to try $(2^{32})^{256}$ for each S-Box, and because BA has four S-Boxes, the attacker has to try out $((2^{32})^{256})^4$ times that equal 2^{32768}. From this analysis, it can be concluded that RAF is more secure than BA.

6.5 Computation Efficiency

This section presents the computation efficiency of RAF and BA.

6.5.1 RAF

RAF consists of one main component (CCSDPB) and two subcomponents (dynamic 3D S-Box and dynamic P-box). Dynamic 3D S-Box has two procedures byte relocations and byte transformation. Byte relocation has four procedures (D_0, D_1, D_2, and D_3). Therefore, the computation efficiency of RAF includes the computation efficiency of byte relocations, byte transformation, dynamic P-Box, and new F-Function. The following shows the computation made:

i. Byte relocation

The computation efficiency for byte relocation is

$O(D_0)$=225 operations
O (D_1)=73 operations

O (D2)=81
O (D3)=69

$$O(Byte\ Relocation)\begin{cases} 1830 & \text{k} = 0 \\ 614 & \text{k} = 1 \\ 677 & \text{k} = 2 \\ 582 & \text{k} = 3 \end{cases}$$

The computation efficiency of byte relocate takes the largest computation efficiency that is 1830. Where i represents an account of the round and D0, D1, D2, D3 is conducted in round 0 to 3 respectively. The details of this computation are presented in Appendix D.

ii. Byte transformation

Byte transformation has three transformations T8, T4, T6. The computation efficiency of these transformations is as follows:

$$O\text{ (Byte Transformation)}=\begin{cases} 1424 & j1 = 1 \ldots. > \text{T8} \\ 588 & j1 = 2 \ldots. > \text{T4} \\ 1183 & j1 = 3 \ldots. > \text{T6} \end{cases}$$

The computation efficiency of Byte Transformation takes the largest computation efficiency that is 1424. The details of this computation are presented in Appendix D.

iii. Dynamic P-BOX

The computation efficiency of Dynamic P-Box is O (dynamic P_Box) = 322. The details of this computation are presented in Appendix D.

iv. New F-Function

The computation efficiency of F-Funcion takes the largest computation efficiency of its parts that is 6934 in Round 0.

$$O(\mathrm{F-Function})\begin{cases} 4708+1830+6(66)\ \ldots\ldots..> & round\ 0 \\ 4708+614+6(66)\ \ldots\ldots\ldots > & round\ 1 \\ 4708+677+6(66)\ \ldots\ldots\ldots > & round\ 2 \\ 4708+583+6(66)\ \ldots\ldots\ldots > & round\ 3 \end{cases}$$

Therefore, the computation effeciency of RAF is *O* (RAF)= 69593. This means that *O* (RAF) is constant time *O*(1). The details of this computation are presented in Appendix D.

6.5.2 Blowfish Algorithm (BA)

BA consists of one main component, F-Function. The computation efficiency of BA includes the computation efficiency of F-Function. The following shows the computation made:

i. F-Function

The computation efficiency of F-Function is O (F-Function) = 9. Therefore, the computation efficiency of BA is O (BA) = 230. This means that the computation efficiency of BA is constant time = O (1).

From the above results, it can be observed RAF requires higher number of operations than BA. However, both algorithms require constant time. The details of this computation are presented in Appendix D. Table 6.6 shows the summary of the computation efficiency of RAF and BA.

Table 6.6

Summary of the Computation Efficiency of RAF and BA.

Name of the algorithms	Components	
RAF	Byte Relocation	
	No. of rounds	Computation Efficiency (no. of operations)
	Round 0	1830
	Round 1	614
	Round 2	677
	Round 3	582
	Byte Transformation	
	Computation Efficiency round(0 to 9)	1424
	Dynamic P-Box	
	Computation Efficiency round(0 to 9)	322
	F-Function(CCSDPB)	
	Computation Efficiency round 0	6934
	RAF	
	Computation Efficiency round(0 to 9)	69593
BA	F-Function	
	Computation Efficiency round 0	9
	BA	
	Computation Efficiency round(0 to 15)	230

6.6 Results of Comparison on RAF with other Cryptographic Algorithms

This section presents results of RAF as compared to Mars, RC6, Rijndael, Serpent, and Twofish in terms of randomness using the Low Density Plaintext dataset.

Figures 6.26 to 6.35 illustrate the results on Low Density Plaintext data set for every round.

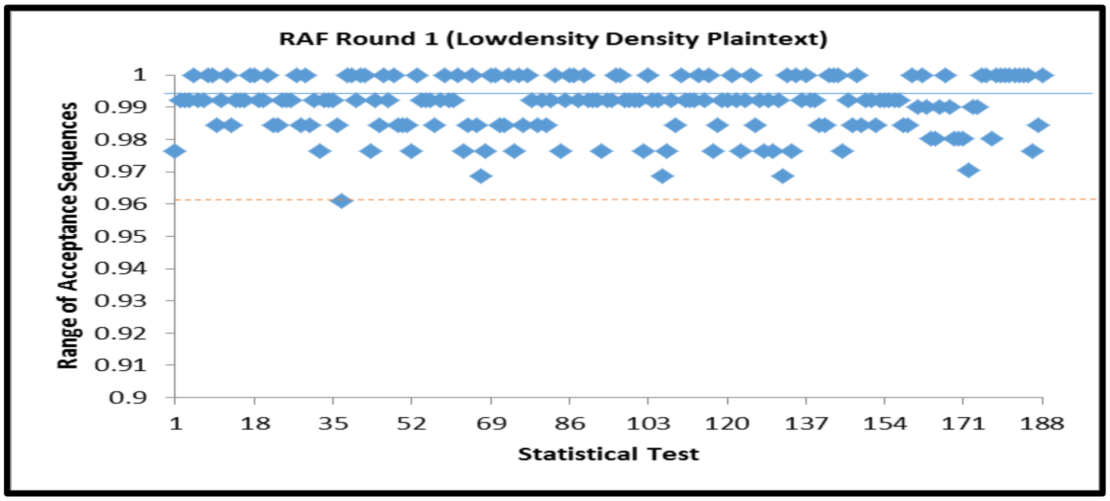


Figure 6.26. Results of Low Density Plaintext for Round 1 in RAF

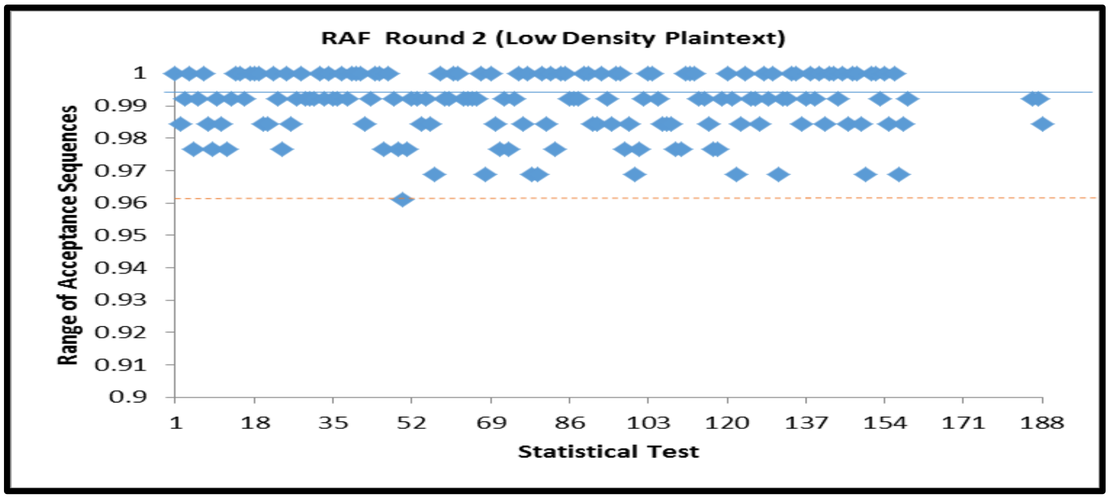


Figure *6.27.* Results of Low Density Plaintext for Round 2 in RAF

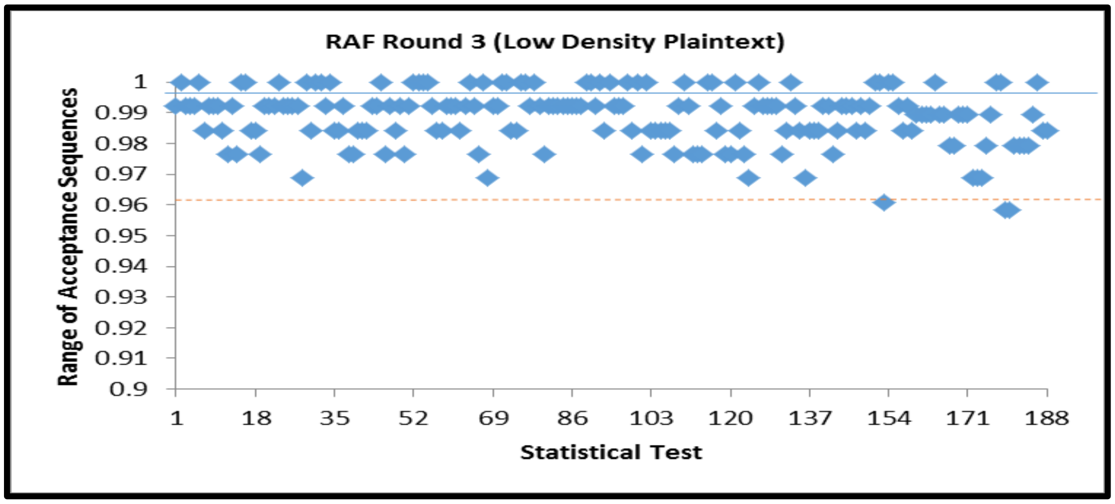


Figure *6.28*. Results of Low Density Plaintext for Round 3 in RAF

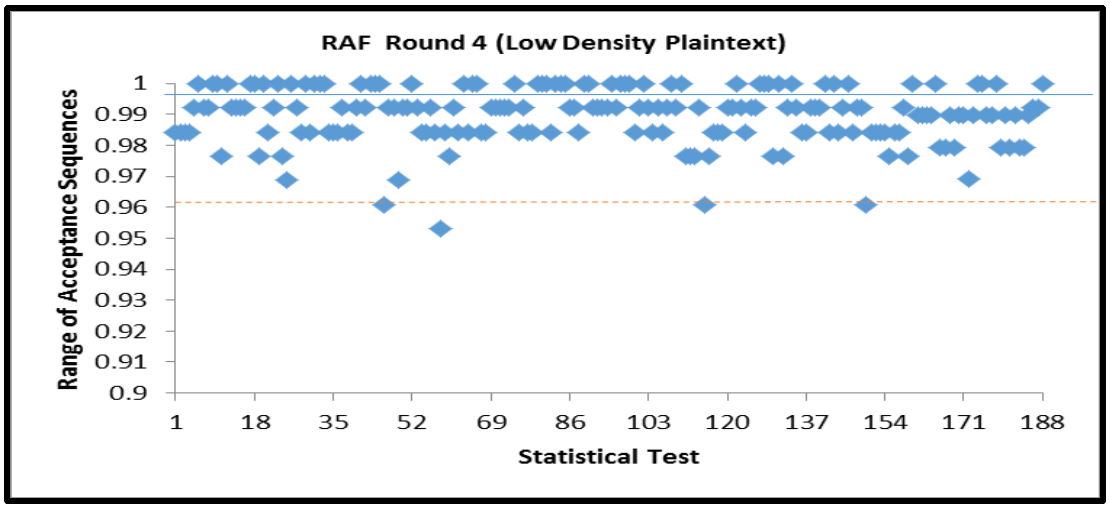


Figure *6.29*. Results of Low Density Plaintext for Round 4 in RAF

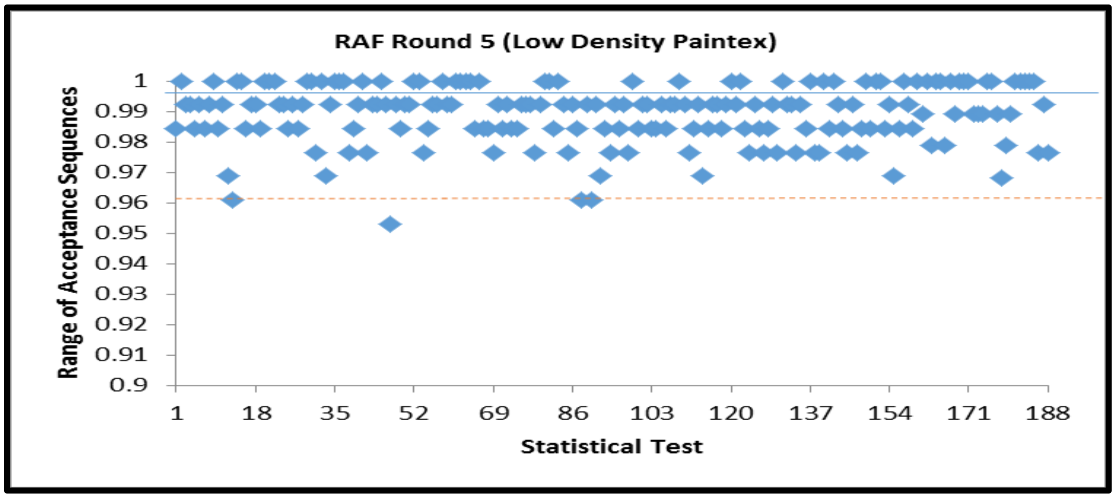


Figure *6.30*. Results of Low Density Plaintext for Round 5 in RAF

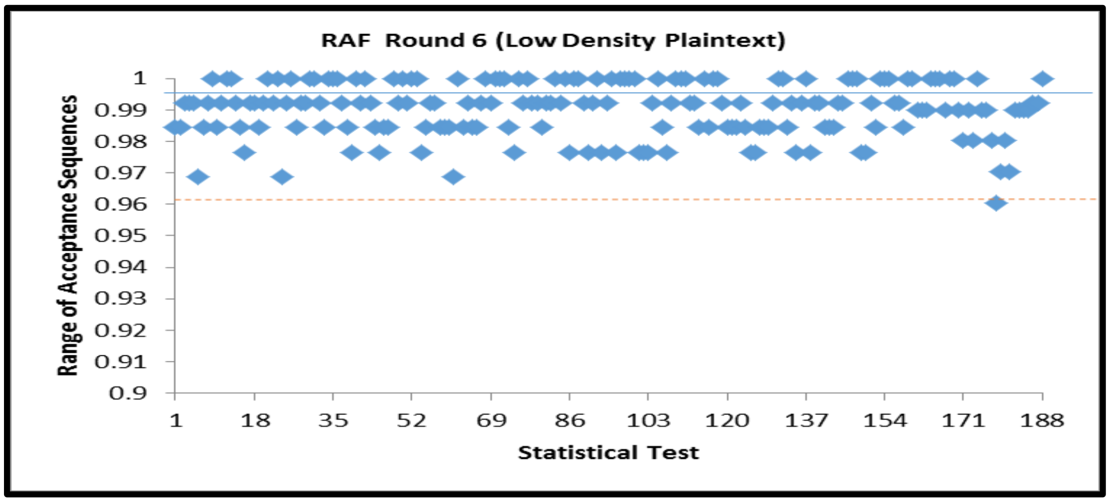


Figure *6.31*. Results of Low Density Plaintext for Round 6 in RAF

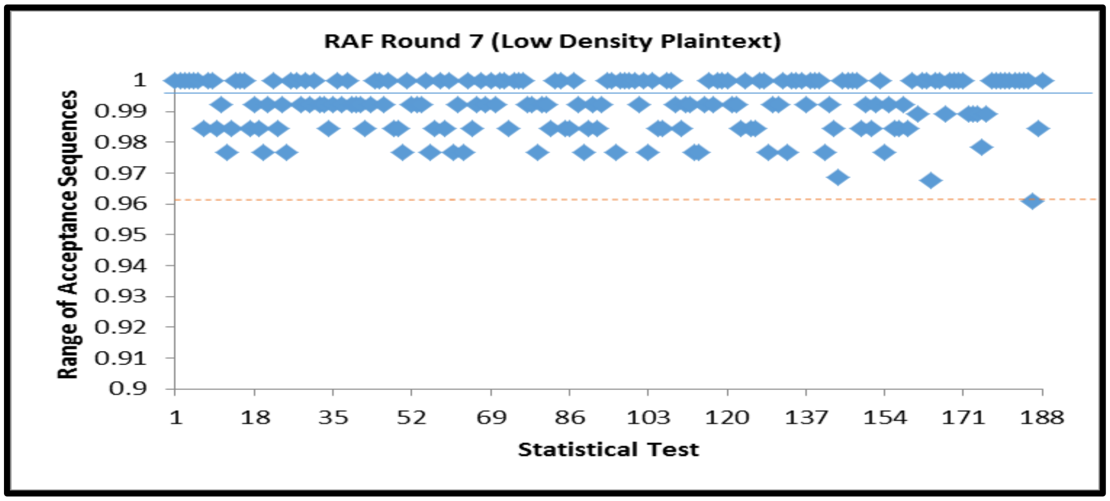


Figure *6.32*. Results of Low Density Plaintext for Round 7 in RAF

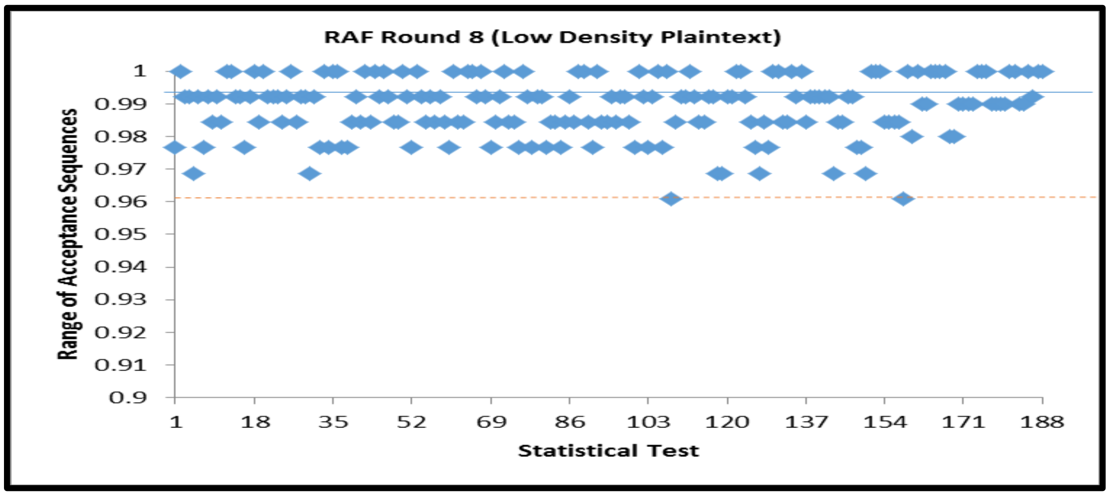


Figure *6.33*. Results of Low Density Plaintext for Round 8 in RAF

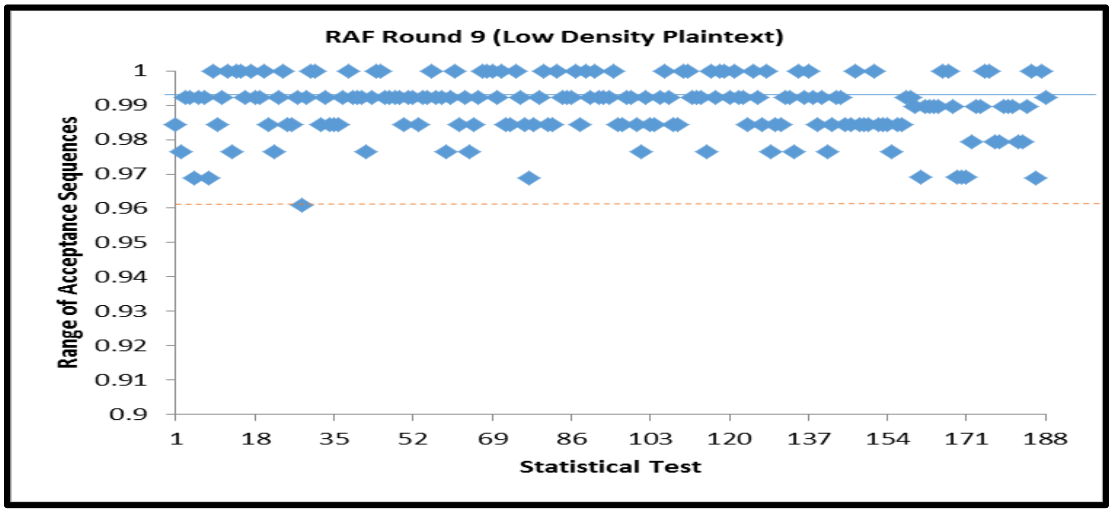


Figure *6.34*. Results of Low Density Plaintext for Round 9 in RAF

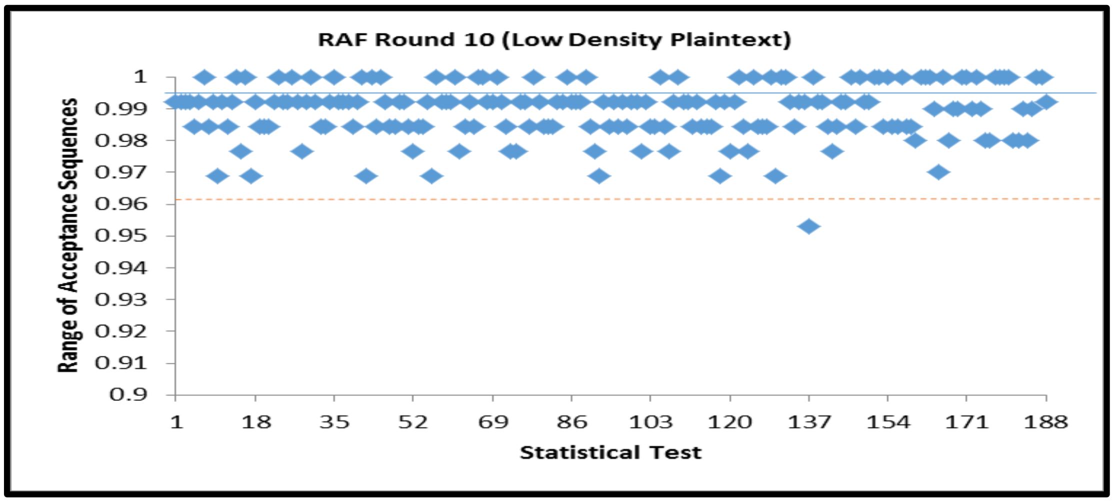


Figure 6.35. Results of Low Density Plaintext for Round 10 in RAF

Results from Figure 6.26 to Figure 6.35 show that the output of RAF is random at the end of the first round on this type of data. This is because the majority of the 188 statistical tests show percentage values greater than 96%.

Table 6.7 shows the comparison of randomness test for RAF with Mars, RC6, Rijndael, Serpent, and Twofish cryptographic algorithms.

Table 6.7

Comparison Randomness of RAF with finalist of AES

Algorithm	Round, where randomness is evident
Mars	1
RC6	4
Rijndael	3
Serpent	4
Twofish	2
RAF	1

The second column in Table 6.7 indicates that to the round number that output of the algorithm is random. It can be seen that Mars achieves random at round 1, RC6 at round 4, Rijndael at round 3, Serpent at round 4, Twofish at round 2, and RAF at round 1. This show that Mars and RAF are secure.

However, RAF is better than Mars as it has a dynamic 3D S-Box that varies in every round with every block. Thus RAF is secured better than others algorithms in table 6.7.

6.7 Summary

This chapter presents results of Phase 3 (Stage 2 and Stage 3) from the methodology. These include results on NIST statistical test, correlation coefficient, avalanche text, and cryptanalysis (linear, differential, and short attacks), and computation efficiency, and comparison of RAF with other cryptographic algorithms.

CHAPTER SEVEN

CONCLUSION

7.1 General Discussion

The research addressed the limitation of the original BA which utilizes a large memory (up to 4,096 bytes), is unsuitable for small devices with limited memory, is incompatible with image and text files having large strings of identical bytes, and with output encrypted files not comprehensively tested for randomness.

In general, the main aim is to design a new cryptography algorithm based on the BA. The new cryptographic algorithm should satisfy these important requirements: reduce memory requirements of S-Boxes and enhance security by increasing randomness of the outputs and the resistance of the algorithm against attacks. To come up with the new algorithm, these specific objectives should be achieved: generating multi secret keys; designing a dynamic 3D S-Box; generating dynamic P-Box; and developing a new F-Function. The strength of the new algorithm was evaluated through verification of dynamic 3D S-Box and RAF output.

The methodology of this research study comprised of three phases: RAF design, RAF implementation, and RAF verification. In the first phase (RAF design), the activities included designing the dynamic 3D S-Box, designing the dynamic P-Box, and developing a new F-Function (CCSDPB). The second phase (RAF implementation) included performing key expansion and performing data encryption and data decryption. Finally, the third phase (RAF verification) verified the RAF strength by performing verifications on the 3D S-Box and the RAF output.

7.2 Research achievement

All the study objectives have been successfully achieved, as summarized below.

- Objective 1: To design a dynamic 3D S-Box

Two algorithms of byte permutation corresponding to the transformation in the right cylinder have been suggested. The execution of this objective produced dynamic 3D S-Box that varied in every round with every block of plaintext; results were presented in Chapter 4. The 3D S-Box was also verified to ensure that all components work well; results for this part were presented in Chapter 5.

- Objective 2: To generate multi secret keys during the encryption process using random function.

Five sets of random number that were used as secret keys to permute the values of 3D S-Box were produced. Results were shown in Chapter 4.

- Objective 3: To design dynamic P-Box.

Dynamic P-Box was successfully developed. The dynamic P-Box is varied in every round with every block of plaintext. The algorithm and the execution results were presented in Chapter 4.

- Objective 4: To design a new F-Function (CCSDPB)

A new round function was developed. The algorithm and the execution results were presented in Chapter 4.

Objective 5: To determine the computational efficiency for the enhanced BA.

Computational efficiency for an enhanced BA is done in Chapter 6 and Appendix D.

7.3 Contributions

The study contributed the following:

- A 3D S-Box design that incorporates the CCS

The research used CCS to render one dynamic 3D S-Box with 256 bytes. In the original BA, four 1D S-Box were used and led to the utilization of large memory space. This was unsuitable for limited-memory applications. However, the dynamic 3D S-Box design in this study is able to cater for the needs of small applications.

- A dynamic 3D S-Box design by employing two procedures of byte permutation based on multi secret keys

Two procedures were used to permute the values of one 3D S-Box so it occupies less memory space, such that every quarter (16-bit) are substituted into 16-bit from the same S-Box (3D S-Box) after applying byte permutation. The original BA occupies larger memory space because the algorithm allocates one S-Box for every quarter (8-bit). Thus, the use of dynamic 3D S-Box reduces the need for a large memory space.

From the evaluation of the S-Boxes in both algorithms, it can be concluded that the 3D S-Box in RAF is more secure than the S-Boxes in BA because 3D S-Box in RAF satisfies AVAL, SAC, and BIC with maximum error values of 0.0566, 0.2813, and 0.2698 respectively. By contrast, the S-Boxes in BA satisfy AVAL, SAC, and BIC with maximum error values of 0.0518, 0.3594, and 0.4725 respectively. Both 3D S-Box in RAF and the S-Boxes in BA are approximately the same in satisfying the AVAL. However, the 3D S-Box in RAF satisfy SAC and BIC more effectively than the S-Boxes in BA, thereby showing that the dynamic 3D S-Box in RAF is more secure than the S-Box in BA. Moreover, the entropy of the encryption keys does not affect the security of S-Boxes in both algorithms.

From the results of correlation coefficient for 3D S-Box in RAF, it can be concluded that the 3D S-Box has high non-linearity and randomness of plaintext and encryption keys that do not affect the security of 3D S-Box.

- A dynamic P-Box design to permute the output of 3D S-Box

The dynamic P-Box developed to break the byte level of 3D S-Box down to bit level thereby giving additional security to the outputs of the 3D S-Box.

- Multi secret keys generation during encryption process using random function

The study was able to generate five different sets of secret keys from the random function. The seed of random function includes some variables, one of which is the sequence of block plaintext which leads to the generation of different seed values and the production of a many random secret keys. This increases the randomness of the algorithm and further increases the level of security.

- A new F- function (CCSDPB)

The CCSDPB function was developed based on the dynamic P-Box and the dynamic 3D S-Box. In previous works, F-Function in BA contained four dynamic S-Boxes which are fixed in every round, thereby leading to some successful trials of attacks. While such attacks fail in RAF because dynamic of S-Box in the original algorithm (BA) is improved. 3D S-Box in RAF varies in every round. Also RAF resists to existing short attacks that try to recover the specific internal structure of the cipher. The cryptanalyst, therefore, needs to try 2^{81920} times for every block because the RAF that has one dynamic P-Box in every round. This means that there is the possibility of $(2^6!)^{10}$ attempts for every block, thereby making the RAF secure such that attackers would have to pay more in terms of costs rather than brute attack. On the other hand, S-Boxes in BA have entries 0...255 and four S-Boxes also, thereby leading the attacker to try as many as 2^{32768} times. In all, it can be concluded that the RAF is more secure than the BA.

The avalanche text of RAF are 0.4912, 0.4926, and 0.4950 for the second, third rounds and ciphertext, respectively, whereas the avalanche texts of BA are 0.5110, 0.5098, and 0.4972 in the second, third rounds and ciphertext, respectively. It is worth noting that the avalanche text of RAF approximates the same avalanche text in BA in these rounds. However, the avalanche text of the first round of both algorithms (RAF and BA) is 0.2555 and 0.2690, respectively. From the results, it can be concluded that the two algorithms gave good avalanche texts from the second round, and that RAF has better impact of the non-linear relations than BA.

In terms of randomness, RAF when compared with MARS, RC6, Rijndael, Serpent, and Twofish shows better results. This means RAF is more secured.

7.4 Limitation

The study has been done to address the memory and security problems. Speed has not been addressed. This will be the future project.

7.5 Recommendations for Future Work

In concluding this study, several recommendations for future work is extended. Even though this research has shown that the proposed algorithm is able to reduce memory space and enhance the security level of BA, further research needs to be done to enhance or support the proposed algorithm. Among the suggestions are the following:

- Analysis of RAF performance based on speed, throughput, and power consumption, after which its performance can be compared with other algorithms of various platforms;
- Implementation and evaluation of the characteristic criteria of RAF, such as flexibility, hardware, software suitability, and algorithm simplicity;
- Development of a new procedure based on key-dependence for the generation of S-Box and P-array;
- Application of another geometric shape (such as hypercube) to design a new 5D S-Box

REFERENCES

Abd-ElGhafar, A. R., Diaa, A., Rohiem, A., & Mohammed, F. (2009). Generation of AES Key Dependent S-Boxes using RC4 Algorithm. *13th International Conference on Aerospace Sciences & Aviation Technology* (pp. 26-28).

Acharya, T., & Ray, A. K. (2005). *Image processing: principles and applications*: John Wiley & Sons.

Adams, C., & Tavares, S. (1990). The structured design of cryptographically good S-Boxes. *journal of Cryptology. Springer,3*(1), 27-41.

Agrawal, H., & Sharma, M. (2010). Implementation and analysis of various symmetric cryptosystems. *Indian Journal of Science and Technology*, *3*(12), 1173-1176.

Ariffin, S. (2012). *A Human Immune System Inspired Byte Permutation Of Block Cipher*. Doctoral dissertation. Universiti Putara Malaysia.

Ahmed, N. (n.d.). Testing an S-Box for Cryptographic Use. *International Journal of Computer and Electrical Engineering*.

ALabaichi, A. M., Mahmod, R., & Ahmad, F. (2013a). Randomness Analysis on Blowfish Block Cipher. *AWERProcedia Iinformation Technology & Computer Science: 3rd World Conference on Innovation and Computer Science* (pp. 1116-1127), Antalya, Turkey.

ALabaichi, A. M., Mahmod, R., Ahmad, F., & Mechee, M.S. (2013b). Randomness Analysis on Blowfish Block Cipher using ECB and CBC Modes, 2013, *Journal of Applied Sciences,13*(6),768-789.

ALabaichi, A., Mahmod, R., & Ahmad, F. (2013c). Analysis of Some Security Criteria for S-Boxes in Blowfish Algorithm, *International Journal of Digital Content Technology and its Applications (JDCTA*), *7*(12), 8–20.

ALabaichi, A. M., Mahmod, R., & Ahmad, F. (2013d). Security Analysis of Blowfish algorithm. *Proceding of SDIWC: The Second International Conference on Informatics &Applications on IEEE* (pp.12–18), lodz university of tehnolgoy .

ALabaichi, A., Mahmod, R., & Ahmad, F. (2013e). Randomness Analysis of 128 bits Blowfish Block Cipher on ECB mode. *(IJCSIS) International Journal of Computer Science and Information Security*, *11* (10), 8-21.

ALabaichi, A., Mahmod, R., & Ahmad, F. (2013f). Randomness Analysis of 128 bits Blowfish Block Cipher on ECB and CBC Modes. *International Journal of Digital Content Technology and its Applications (JDCTA*), 7(15), 77-89.

Alani, M. d M. (2010). Testing randomness in ciphertext of block-ciphers using dieHard tests. *International Journal of Computer Science and Network Security, 10* (4), 53-57.

Al-Hazaimeh, O. M. A. (2010). *New Cryptographic Algorithms for Enhancing Security of Voice Data*. Doctoral dissertation. Universiti Utara Malaysia.

Ali, F.H.M. (2009). *A NEW 128-BIT BLOCK CIPHER*. Doctoral Dissertation. Universiti Putra Malaysia.

Ali, N. B. Z., & Noras, J. M. (2001). Optim al Datapath Design for a Cryptographic Processor: The Blowfish Algorithm. *Malaysian Journal of Computer Science. Faculty of Computer Science and Information Technology, 14*(1), 16-27.

Ali, S.A. (2005*). Improving the Randomness of Output Sequence for the Advanced Encryption Standard Cryptographic Algorithm*. Master thesis. Universiti Putra Malaysia.

Al-Neaimi, A. M. A., & Hassan, R. F. (2011a). New Approach for Modified Blowfish Algorithm Using 4-States Keys. *The 5th International Conference on Information Technology* (pp.1-4).

Al-Neaimi, A. M. A., & Hassan, R. F. (2011b). New Approach for Modifying Blowfish Algorithm by Using Multiple Keys. *International Journal of Computer Science and Network Security (IJCSNS), 11*(3), 21-26.

Alsultanny, Y. A., & Jarrar, H. J. (2006). Generating and testing random key for image encryption using ECB and CBC modes. *Jordan Journal of Applied Science Natural Sciences*. *Deanship Of Scientific Research Applied Science University, 8*(1), 1-11.

Ayoub, F. (1982). Probabilistic completeness of substitution-permutation encryption networks. *IEE Proceedings E (Computers and Digital Techniques), 129*(5), 195-199.

Babbage, S., Canniere, C., Canteaut, A., Cid, C., Gilbert, H., Johansson, T., & Robshaw, M. (2008).

Bagad, V.S. and A.I. Dhotre (2008). *Cryptography and Network Security*. 2nd Revised Edn., Technical Publications, Pune, India.

Berbain, C., Billet, O., Canteaut, A., Courtois, N., & Gilbert, H. (2008). SOSEMANUK , a fast software-oriented stream cipher, 16–18.

Bernstein, D. J. (2008). The Salsa20 family of stream ciphers *New stream cipher designs* (pp. 84-97): Springer.

Bernstein, D. J. (2006). Salsa20/8 and Salsa20/12. *eSTREAM, ECRYPT Stream Cipher Project*.

Biham, E., & Shamir, A. (1993). *Differential cryptanalysis of the full 16-round DES* (pp. 79-88). Springer New York.

Boesgaard, M., Pedersen, T., Vesterager, M., & Zenner, E. (2004). The Rabbit Stream Cipher-Design and Security Analysis. *IACR Cryptology ePrint Archive, 2004,* 291.

Boesgaard, M., Vesterager, M., Pedersen, T., Christiansen, J., & Scavenius, O. (2003). *Rabbit: A new high-performance stream cipher.* Paper presented at the Fast Software Encryption.

Braeken, A. (2006). *Cryptographic properties of Boolean functions and S-Boxes*. Doctoral dissertation. Katholieke Universiteit Leuven.

Brougham, H. P. (n.d.). Coordinate Systems And Transformation, 124–130. Retrieved from http://web.uettaxila.edu.pk/CMS/AUT2012/ectWPAbs/notes%5CLecture%2013%20Notes.pdf

Brannon, R. M. (2004). Curvilinear Analysis in a Euclidean Space. University of New Mexico. Retrieved from http://mech.utah.edu/~brannon/public/curvilinear.pdf

Burwick, C., Coppersmith, D., D'vignon, E., Gennaro, R., Halevi, S., Jutla, C., & Safford, D. (1998). MARS-a candidate cipher for AES. *NIST AES Proposal 268.*

Castro, J. C. H., Sierra, J. M., Seznec, A., Izquierdo, A., & Ribagorda, A. (2005). The strict avalanche criterion randomness test. *Mathematics and Computers in Simulation. Elsevier, 68*(1), 1-7.

Carter, B. A., Kassin, A., & Magoc, T. (2007). Symmetric cryptosystems and symmetric key management. *CiteSeerX, 10*(1.135), 1231.

Chandrasekaran, J., Subramanyan, B., & Raman, G. S. (2011). Ensemble Of Blowfish With Chaos Based S Box Design For Text And Image Encryption. *International Journal of Network Security & Its Applications (IJNSA). Academy & Industry Research Collaboration Center(AIRCC), 3*(4), 165-173.

Cody, B., Madigan, J., Donald, S.M., & Hsu, K. W. (2007). High speed SOC design for blowfish cryptographic algorithm. In *Very Large Scale Integration. IFIP International Conference on* (pp.284-287). IEEE.

Collins, G. W. (1989). The foundations of celestial mechanics. *The Foundations of Celestial Mechanics, by George W. Collins, II. Tucson, AZ, Pachart Publishing*

House (Pachart Astronomy and Astrophysics Series. Volume 16), 1989, 158 p., 1.

Cornwell, J. W. (n.d.). Blowfish Survey. *Department of Computer Science Columbus State University Columbus*, GA. Retrieved from http://cs.columbusstate.edu/cae- ia/StudentPapers/cornwell.jason.pdf

Daemen, J., & Rijmen, V. (2002). *The design of Rijndael: AES-the advanced encryption standard*. Springer.

Dawson, E., Gustafson, H., & Pettitt, A. N. (1992). Strict Key Avalanche Criterion. *Australasian Journal of Combinatorics, 6,* 147-153.

Deakin, R.E. (2004). Coordinate Transformations In Surveying And Mapping, 1–31. Retrieved from http://user.gs.rmit.edu.au/rod/files/publications/COTRAN_1.pdf

Durstenfeld, R. (1964). ACM Algorithm 235: Random Permutation, *Communications of the ACM,* Vol. 7, No 7.

Denning, D. E.R (1982). *Cryptography and data security*. Addison-Wesley Longman Publishing Co., Inc.

Diffie, W., & Hellman, M. (1976). New directions in cryptography. *Information Theory, IEEE Transactions on*, *22*(6), 644-654.

Doganaksoy, A., Ege, B., Koçak, O., & Sulak, F. (2010). Cryptographic Randomness Testing of Block Ciphers and Hash Functions. *IACR Cryptology ePrint Archive*. 564, 1-12.

Doroshenko, S., Fionov, A., Lubkin, A., Monarev, V., Ryabko, B., & Shokin, Y. I. (2008). Experimental Statistical Attacks on Block and Stream Ciphers. *Computational Science and High Performance Computing III* pp.(155-164) Springer Berlin Heidelberg.

Dworkin, M. (2001). *Recommendation for block cipher modes of operation. methods and techniques* (No.Nist-Sp-800-38a). National Institute Of Standards And Technology Gaithersburg Md Computer Security Div.

Elkamchouchi, H. M., & Elshafee, A. M. (2003). Dynamically key-controlled symmetric block cipher KAMFEE. In *Radio Science Conferenc. NRSC 2003. Proceedings of the Twentieth National* (pp. C19-1). IEEE.

Elkamchouchi, H. M., & Makar, M. A. (2004). Kamkar symmetric block cipher. *Radio Science Conference, 2004. NRSC 2004. Proceedings of the Twenty-First National* (pp. C1-1-C1-9). IEEE.

Elminaam, D., Kader, H. M. A., & Hadhoud, M. M. (2010). Evaluating The Performance of Symmetric Encryption Algorithms. *IJ Network Security*.

Elminaam, D., Kader, H. M. A., & Hadhoud, M. M. (2009). Energy efficiency of encryption schemes for wireless devices. *International Journal of Computer Theory and Engineering, 1*, 302-309.

El-Ramly, S. H., El-Garf, T., & Soliman, A. H. (2001). Dynamic generation of S-Boxes in block cipher systems. *Radio Science Conference, 2001. NRSC 2001. Proceedings of the Eighteenth National* Vol. 2, pp. (389-397). IEEE.

Fahmy, A., Shaarawy, M., El-Hadad, K., Salama, G., & Hassanain, K. (2005). A proposal For A key-dependent AES. *3rd International Conference: Sciences of Electronic, Technologies of Information and Telecommunications. Tunisia: SETIT*.

Feistel, H. (1973). Cryptography and computer privacy. *Scientific American , 228*(5), 15-23.

Gan, L. (2001). A New Stream Cipher.for Secure Digital Media Distribution. Thesis Master of Science (Engineering), Queen's University Kingston, Ontario, Canada.

Gonzales, R. C., & Woods, R. E. (2002). Digital Image Processing. *New Jersey: Prentice Hall*, *6*, 681.

Gill, G.S. (2003).*The calculus bible*. Brigham Young University, Mathematics Department.

Halagali, B. P. (2013). Designing The S Boxes Of Blowfish Algorithm Using Linear Congruential Generator, *ASM's international E-journal of ongoing research in management and IT e-ISSN-2320-0065, V S M Institute of Technology*, Nipani, Karnataka.

Hardjono, T., & Dondeti, L. R. (2005). *Security in wireless LANs and MANs*. Artech House.

Hashim, A. T., Al-Qarrawy, S. M., & Mahdi, J. A. (2009). Design and Implementation of an Improvement of Blowfish Encryption Algorithm. IJCCCE , 9(1), 1-15.

Heys, H. M. (2002). A tutorial on linear and differential cryptanalysis. *Cryptologia. Taylor & Francis*, *26*(3), 189-221.

Hosseinkhani, R., & Javadi, H. H. S. (2012). Using Cipher Key to Generate Dynamic S-Box in AES Cipher System. *International Journal of Computer Science and Security (IJCSS)*, *6*(1), 19-28.

Hussain, I., Shah, T., Mahmood, H., & Afzal, M. (2010). Comparative analysis of S-Boxes based on graphical SAC. International Journal of Computer Applications. International Journal of Computer Applications, 2(5),5-8.

Isa, H., & Z'aba, M. R. (2012). Randomness analysis on LED block ciphers. *Proceedings of the Fifth International Conference on Security of Information and Networks(*pp. 60-66).ACM.

Junod, P., & Vaudenay, S. (2004). Perfect diffusion primitives for block ciphers. *Selected Areas in Cryptography* (pp. 84-99). Springer Berlin Heidelberg.

Juremi, J., Mahmod, R., & Sulaiman, S. (2012). A proposal for improving AES S-Box with rotation and key-dependent. In *Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), 2012 International Conference on* (pp. 38-42). IEEE.

Katos, V. (2005). *A randomness test for block ciphers. Applied mathematics and computation. Elsevier,162*(1), 29-35.

Kavut, S., & Yücel, M. D. (2001). On some cryptographic properties of Rijndael. *Information Assurance in Computer Networks* (pp. 300-311).Springer Berlin Heidelberg.

Kalnins, L. M. (2009). Coordinate Systems, (2009), 1–5. Retrieved from http://www.earth.ox.ac.uk/~larak/MMES/CoordinateSystems.pdf

Kazlauskas, K., & Kazlauskas, J. (2009). Key-dependent S-Box generation in AES block cipher system. *Informatics*, *20*(1), 23-34.

Keliher, L. (1997). *Substitution-permutation network cryptosystem using Key-Dependent S-Boxes*. Master thesis. Queen's university, Kingston, Ontario, Canada.

Kellerman Software, "What is The Strongest Encryption Algorithm?" July. 16, 2008 http://www.kellermansofiware.comltArticleStrongestAlgo.aspx[Accessed: June .22, 20101.

Kern, W. F. and Bland, J. R. (1948). Circular Cylinder and Right Circular Cylinder.16-17 in Solid Mensuration with Proofs, 2nd. New York: Wiley.

Kelsey, J., Schneier, B., & Wagner, D. (1997). Related-key cryptanalysis of 3-way, biham-des, cast, des-x, newdes, rc2, and tea. *Information and Communications Security*, 233-246.

Khovratovich, D., Leurent, G., & Rechberger, C. (2012). Narrow-Bicliques: cryptanalysis of full IDEA *Advances in Cryptology–EUROCRYPT 2012* (pp. 392-410): Springer.

Kiran, L. K., Abhilash, J. E. N., & Kumar, P. S. (2013). FPGA Implementation of Blowfish Cryptosystem Using VHDL. *International Journal of Engineering, 2*(1), 1-5.

Kofahi, N. A., Al-Somani, T., & Al-Zamil, K. (2004). Performance evaluation of three encryption/decryption algorithms. *Circuits and Systems, 2004 IEEE 46th Midwest Symposium on 2*(pp. 790-793). IEEE.

Krishnamurthy, G. N., & Ramaswamy, V. (2009). Performance Analysis of Blowfish and its Modified Version using Encryption quality, Key sensitivity, Histogram and Correlation coefficient analysis. *International Journal of Recent Trends in Engineering, 8*(4),1-4.

Krishnamurthy, G. N., Ramaswamy, V., & Leela, M. G. H. (2007). Performance Enhancement of Blowfish algorithm by modifying its function. *Innovative Algorithms and Techniques in Automation, Industrial Electronics and Telecommunications* (pp. 241-244). Springer Netherlands.

Krishnamurthy, G. N., Ramaswamy, V., Leela, G. H., & Ashalatha, M. E. (2008). Blow-CAST-Fish : A New 64-bit Block Cipher, *8*(4), 282–290.

Kruppa, H., & Shahy, S. U. A. (1998). *Differential and Linear Cryptanalysis in Evaluating AES Candidate Algorithms*. ELECTRONIC REPORTING SYSTEM-ERS, National Institute of Standards and Technology.

Kumar, R. S., Pradeep, E., Naveen, K., & Gunasekaran, R. (2010). A Novel Approach for Enciphering Data of Smaller Bytes. *International Journal of Computer Theory and Engineering, 2*(4), 654-659.

Kumar, P.K., & Baskaran, K. (2010). An ASIC implementation of low power and high throughput blowfish crypto algorithm. *Microelectronics Journal*, *Elsevier*, *41*(6), 347-355.

Lai, Y.-K., & Shu, Y.-C. (1999). A novel VLSI architecture for a variable-length key, 64-bit blowfish Block cipher. *Signal Processing Systems, 1999. SiPS 99. 1999 IEEE Workshop on* (pp. 568-577). IEEE.

Lambers, J. (2009) *Three-Dimensional* Coordinate System. Lecture 17 Notes MAT 169 Fall Semester 2009-10. Retrieved from http://www.math.usm.edu/lambers/mat169/fall09/lecture17.pdf

Landge, I., Contractor, B., Patel, A., & Choudhary, R. (2012). Image encryption and decryption using blowfish algorithm. *World Journal of Science and Technology, 2*(3):151-156.

Lashkari, A. H., Danesh, M. M. S., & Samadi, B. (2009). *A survey on wireless security protocols (WEP, WPA and WPA2/802.11 i).* Paper presented at the

Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on.

Lautrup, B. (2011). *Physics of Continuous Matter: Exotic and Everyday Phenomena in the Macroscopic World*. Taylor & Francis

Limin, F., Dengguo, F., & Yongbin, Z. (2008). A fuzzy-based randomness evaluation model for block cipher. *Journal of Computer Research and Development*, *45*(12), 2095-2101.

Lin, M. C.J. & Lin, Y.L. (2000). A VLSI implementation of the blowfish encryption/decryption algorithm. In *Proceedings of the 2000 Asia and South Pacific Design Automation Conference* (pp. 1-2). ACM.

Maenhaut, M. (2010). *Coding and Cryptography Part 1*. Retrieved from www.maths.uq.edu.au/courses/MATH3302/2010/files/cryptonotes.pdf.

Mahdi, J. A. (2009). Design and Implementation of Proposed BR Encryption Algorithm. IJCCCSE, *9*(1), 1-17.

Mahmoud, E. M., Hafez, A.A.E., Elgarf, T. A., & Zekry, A.H. (2013). Dynamic AES-128 with Key-Dependent S-Box. International Journal of Engineering Research and Applications (IJERA), *3*(1), 1662–1670.

Mandal, P. C. (2012). Evaluation of performance of the Symmetric Key Algorithms: DES, 3DES, AES and Blowfish. *Journal of Global Research in Computer Science*, *3*(8), 67-70.

Manikandan, G., Manikandan, R., Rajendiran, P., Krishnan, G., & Sundarganesh, G. (2011a). An integrated block and stream cipher approach for key enhancement. *Journal of Theoretical and applied information Technology*, *28*(2), 83-87.

Manikandan, G., Kamarasan, M., Rajendiran, P., & Manikandan, R. (2011b). A hybrid approach for security enhancement by modified crypto-stegno scheme. *European Journal of Scientific Research*, *60*(2), 224-230.

Manikandan, G., Sairam, N., & Kamarasan, M. (2012b). A New Approach for Improving Data Security using Iterative Blowfish Algorithm. *Research Journal of Applied Sciences*, *4*(6), 603-607.

Manikandan, G., Rajendiran, P., Chakarapani, K., Krishnan, G., & Sundarganesh, G. (2012a). A Modified Crypto Scheme For Enhancing Data Security. *Journal of Theoretical and Applied Information Technology, 35* (2), 149-154.

Mar, P. P., & Latt, K. M. (2008). New analysis methods on strict avalanche criterion of S-Boxes. *World Academy of Science, Engineering and Technology*, 24, 150-1154.

Matsui, M. (1994). Linear cryptanalysis method for DES cipher. In *Advances in Cryptology EUROCRYPT 93*, (pp. 386-397). Springer Berlin Heidelberg.

Mattsson, J. (2006). *Stream Cipher Design*. *Master of Thesis*. Stockholm, Sweden. Citeseer.

Maximov, A. (2006). *Some Words on Cryptanalysis of Stream Ciphers*. Doctoral dissertation, Department of Information Technology Electrical and information technology , *Lund Universit.*

Meiser, G. (2007). *Efficient Implementation of Stream Ciphers on Embedded Processors.* M. Sc. Thesis, Ruhr-University Bochum, Germany.

Menezes, A., Van Oorschot, P., & Vanstone, S. (1997). *Handbook of Applied Cryptography*. CRC Press.

Meyers, R. K., & Desoky, A. H. (2008). An Implementation of the Blowfish Cryptosystem. In *Signal Processing and Information Technology, 2008. ISSPIT 2008. IEEE International Symposium on* (pp. 346-351). IEEE.

Milad, A. A., Muda, H. Z., Noh, Z. A. B. M., & Algaet, M. A. (2012). Comparative Study of Performance in Cryptography Algorithms (Blowfish and Skipjack). *Journal of Computer Science, 8* (7): 1191-1197.

MIT (2005). Review B: Coordinate Systems. Massachusetts Institute of Technology Department of Physics, 8.01.

Mohammad, F. Y., Rohiem, A. E., & Elbayoumy, A. D. (2009). A Novel S-Box of AES Algorithm Using Variable Mapping Technique. *Aerospace Sciences & Aviation Technology*(pp.1-9).

Mohan, H. S., & Red dy, A. R. (2011). Performance Analysis of AES and MARS Encryption Algorithms. *International Journal of Computer Science Issues (IJCSI), 8*(4), 363-368.

Mousa, A. (2005). Data encryption performance based on Blowfish. In *ELMAR, 2005. 47th International Symposium* (pp. 131-134). IEEE.

Mollin, R.A. (2007). *An Introduction to Cryptography*. U.S.A.: Chapman & Hall/CRC.

Mister, S., & Adams, C. (1996). Practical S-Box design. In *Workshop on Selected Areas in Cryptography, SAC* (Vol. 96, pp. 61-76).

Nadeem, A., & Javed, M. Y. (2005). A performance comparison of data encryption algorithms. *Information and communication technologies, 2005. ICICT 2005. First international conference on* (pp. 84-89). IEEE.

Naganathan, E. R., Nandakumar, V., & Dhenakaran, S. S. (2011). Identity Based Encryption Using Feistel Cipher. *European Journal of Scientific Research, 54*(4), 532–537.

Nakahara, J.J. (2008). 3D: A three-dimensional block cipher. In *Cryptology and Network Security*. (pp. 252-267). Springer Berlin Heidelberg.

Nakahara, J., J. (2007). A linear analysis of Blowfish and Khufu. In *Information Security Practice and Experience* (pp. 20-32). Springer Berlin Heidelberg.

National Institute of Standards and Technology (1999). "FIPS-46: Data Encryption Standard (DES)" *DATA Encryption Standard (DES)*. U.S.A:Federal Information Processing Standards Publication. Retrieved from http://csrc.nist .gov/publications/fips/fips46-3/fips46-3.pdf.

Nechvatal, J., Barker, E., Bassham, L., Burr, W., & Dworkin, M. (2000). Report on the development of the Advanced Encryption Standard (AES): DTIC Document.

Nguyen, P. H., Wu, H., & Wang, H. (2011). *Improving the algorithm 2 in multidimensional linear cryptanalysis.* Paper presented at the Information Security and Privacy.

Nie, T., Song, C., & Zhi, X. (2010). Performance Evaluation of DES and Blowfish Algorithms. In *Biomedical Engineering and Computer Science (ICBECS), 2010 International Conference on* (pp. 1-4). IEEE.

Nie, T., & Zhang, T. (2009). A study of DES and Blowfish encryption algorithm. In *TENCON 2009-2009 IEEE Region 10 Conference* (pp. 1-4). IEEE.

Paar, C. (2005). Applied Cryptography and Data Security. *Lecture Notes*. Retrieved from http://imperia.rz.rub.de:9085/imperia/md/content/lectures/notes.pdf

Pandey, U., Manoria, M., & Jain, J. (2012). A Novel Approach for Image Encryption by New M Box Encryption Algorithm using Block based Transformation along with Shuffle Operation. *International Journal of Computer Applications*. *Citeseer*, *42*(1), 9-15.

Patidar, V., Sud, K. K., & Pareek, N. K. (2009). A Pseudo Random Bit Generator Based on Chaotic Logistic Map and its Statistical Testing. *Informatica (Slovenia)*. *Citeseer*, *33*(4), 441-452.

Preneel, B., Biryukov, A., Oswald, E., Rompay, B. V, Granboulan, L., Dottax, E., & Dichtl, M. (2003). *NESSIE security report*. *Deliverable D20, NESSIE Consortium. Feb*.

Pub, N. F. (2001). 197. *Announcing the Advanced Encryption Standard (AES)*. Information Technology Laboratory, Processing Scientists Publication *1* 79, National Institute of Standards am1 Technology (NIST), 2001.

Ramanujam, S., & Karuppiah, M. (2011). Designing an algorithm with high avalanche effect. *International Journal of Computer Science and Network Security (IJCSNS)*, *11*(1), 106-111.

Rapeti, S. A. (2008). *NLFS: A New Non-Linear Feedback Stream Cipher*. Doctoral dissertation, Indian Institute of Technology.

Ritter, T. (1990). Substitution Cipher With Pseudo-Random Shuffling: the Dynamic Substitution Combiner. *Cryptologia*, *14*(4), 289–303. doi:10.1080/0161-119091864986

Ritter, T. (1991). TRANSPOSITION CIPHER WITH PSEUDO-RANDOM SHUFFLING : THE DYNAMIC TRANSPOSITION COMBINER. *Cryptologia*, 37–41.

Rivest, R. L., Robshaw, M., Sidney, R., & Yin, Y. L. (1998). *The RC6TM block cipher*. Paper presented at the First Advanced Encryption Standard (AES) Conference.

Rolf, O. (2005). *Contemporary Cryptography*. Artech House Computer Security Series, Boston-London.

Russ, J. C., & Russ, J. C. (2007). *Introduction to image processing and analysis*: CRC Press, Inc.

Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, M., Vangel, M. ,Banks , D. , Heckert, A., Dray, J. , Vo, S. (2010). *A statistical test suite for random and pseudorandom number generators for cryptographic application*s. National Institute of Standards and Technology Special Publication. Report number: 800-22.

Sha'ameri, A. Z. (2006*). Secured Hf Image Transmission System*. Master thesis. Universiti Teknologi Malaysia.

Schneier, B. (1996*). Applied cryptography. Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc.

Schneier, B. (1994). Description of a new variable-length key, 64-bit block cipher (Blowfish). In *Fast Software Encryption* (pp. 191-204). Springer Berlin Heidelberg.

Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., & Ferguson, N. (1998). Twofish: A 128-bit block cipher. *NIST AES Proposal, 15*.

Shannon, C. E. (1949). Communication theory of secrecy systems. *Bell system technical journal*, *28*(4), 656-715.

Schmidt, D. (2006). Kaweichel, an Extension of Blowfish for 64-Bit Architectures. *International Association for Cryptologic Research*, 1-8.

Sindhuja, A., Logeshwari, R., & Sikamani, K.T. (2010). A secure PMS based on Fingerprint Authentication and Blowfish cryptographic algorithm. In *Signal and Image Processing (ICSIP), 2010 International Conference on* (pp. 424-429). IEEE.

Singh, G., Singla, A. K., & Sandha, K. S. (2011). Performance Evaluation of Symmetric Cryptography Algorithms. *IJCSNS International Journal of Computer Science and Network Security*, *8*(12), 280-286.

Soto, J. (1999a). Statistical testing of random number generators. *Proceedings of the 22nd National Information Systems Security Conference*(pp.1-12). NIST Gaithersburg, MD.

Soto, J. (1999b). Randomness testing of the AES candidate algorithms. *NIST. Available via csrc. nist. gov*. Citeseer.

Soto, J., & Bassham, L. (2000). *Randomness testing of the advanced encryption standard finalist candidates*. DTIC Document.

Stallings, W. (2005). *Cryptography and Network Security*: Principles and Practices,Prentice Hall. *Inc New Jersey*.

Stamp, M. (2006).*Information Security: Principles and Practice*. John Wiley& Sons.

Stoianov, N. (2011). One Approach of Using Key-Dependent S-BOXes in AES. In *Multimedia Communications, Services and Security* (pp. 317-323). Springer Berlin Heidelberg.

Sulaiman, S., Muda, Z., & Juremi, J. (2012a). The new approach of Rijndael key schedule. In *Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), 2012 International Conference on* (pp. 23-27). IEEE.

Sulaiman, S., Muda, Z., Juremi, J., Mahmod, R., & Yasin, S.M. (2012b). A New ShiftColumn Transformation : An Enhancement of Rijndael Key Scheduling. *International Journal of Cyber-Security and Digital Forensics (IJCSDF), 1*(3), 160–166.

Sulak, F., Doganaksoy, A., Ege, B., & Koak, O. (2010). Evaluation of randomness test results for short sequences. In *Sequences and Their Applications–SETA 2010* (pp. 309-319). Springer Berlin Heidelberg.

Sulong, M. R. (2008). *Key Transformation Approach for Rijndael Security*.Master thesis. Universiti Putra Malaysia.

Suri, P R, & Deora, S. S. (2010). A Cipher based on 3D Array Block Rotation. *IJCSNS International Journal of Computer Science and Network Security, 10*(2), 186-191.

Suri, Pushpa R, & Deora, S. S. (2011). 3D array block rotation cipher: An improvement using lateral shift. *Global Journal of Computer Science and Technology*, *11*(19), 1-8.

Tamimi, A. Al. (2008). Performance analysis of data encryption algorithms. *Retrieved from* http://www.cse.wustl.edu/~jain/cse567-06/ftp/encryptionperf/index.html

Thakur, J., & Kumar, N. (2011). DES, AES and Blowfish: Symmetric Key Cryptography Algorithms Simulation Based Performance Analysis. *International Journal of Emerging Technology and Advanced Engineering 1*(2),6-12.

Tilborg, H. C. A. van, & Jajodia, S. (2005). *Encyclopedia of Cryptography and Security*. (H. C. A. van Tilborg & S. Jajodia, Eds.). New York, USA: Springer. doi:10.1007/0-387-23483-7

Vaidhyanathan , V., Manikandan G., & Krishnan G. (2010). A Novel Approach to the Performance and Security Enhancement Using Blowfish Algorithm. *International Journal of advance Research in Computer Science, 1*(4), 451-454.

Vaudenay, S. (1995). On the weak keys of Blowfish. In *Fast Software Encryption* (pp. 27-32). Springer Berlin Heidelberg.

Vergili, I., & Yücel l, M. D. (2001). Avalanche and Bit Independence Properties for the Ensembles of Randomly Chosen S-Boxes. *Turk J Elec Engin*, *9*(2), 137-145.

Vergili, I., & Yücel, M. D. (2000) On Satisfaction of Some Security Criteria for Randomly Chosen S-Boxes. In *Proc. 20th Biennial Symp. on Communications, Kingston* (pp. 64-68).

Verma, O. P., Agarwal, R., Dafouti, D., & Tyagi, S. (2011). Peformance analysis of data encryption algorithms. *Electronics Computer Technology (ICECT), 2011 3rd International Conference on* (Vol. 5, pp. 399-403). IEEE.

Wang, Z., Graham, J., Ajam, N., & Jiang, H. (2011). Design and optimization of hybrid MD5-blowfish encryption on GPUs. *Proceedings of 2011 International Conference on Parallel and Distributed Processing Techniques and Applications (* pp.18-21). Las Vegas, Nevada, USA.

Webster, A. F., & Tavares, S. E. (1986). On the design of S-Boxes. In *Advances in Cryptology—CRYPTO'85 Proceedings* (pp. 523-534). Springer Berlin Heidelberg.

Wrede, R., & Spiegel, M. R. (2002). *Advanced, Theory and Problems of Advanced Calculus*. U.S.A.: McGraw-Hill.

Wu, Y., & Wong, S. (2004). Design Challenges in Security Processing. *15th Annual Workshop on Circuits, Systems and Signal Processing* (pp. 189-194).

Wu, H. (2004, January). A new stream cipher HC-256. In *Fast Software Encryption* (pp. 226-244). Springer Berlin Heidelberg

Zhang, R., & Chen, L. (2008). A block cipher using key-dependent S-Box and P-Boxes. *Industrial Electronics, 2008. ISIE 2008. IEEE International Symposium on* (pp. 1463-1468). IEEE.

Zhou, Q., Liao, X., Wong, K., Hu, Y., & Xiao, D. (2009). True random number generator based on mouse movement and chaotic hash function. *information sciences*. Elsevier, *179*(19), 3442-3450.

Zwillinger, D. (2003). CRC Standard Mathematical Tables and Formulae. Boca Raton, FL: CRC Press.

Zhang, Y. P., Sun, J., & Zhang, X. (2004). A stream cipher algorithm based on conventional encryption techniques. Paper presented at the Canadian Conference on Electrical and Computer Engineering, Canada.