Indian Journal of Science and Technology, *Vol 8(30), DOI: 10.17485/ijst/2015/v8i30/86800, November 2015*

ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

A Dynamic 3D S-Box based on Cylindrical Coordinate System for Blowfish Algorithm

Ashwak Mahmood Alabaichi*

Faculty of Sciences, Kerbala University, Karbala - 56001, Iraq;

Abstract

Blowfish Algorithm (BA) is a symmetric block cipher that uses Feistel network to iterate simple encryption and decryption functions. The BA key varies from 32 to 448 bits to ensure a high level of security. However, S-boxes in the BA have a high percentage of memory. A new cryptography algorithm based on BA is designed to overcome this problem. This algorithm adapts a new function (F-function) into a Cylindrical Coordinate System (CCS). The F-function is known as Cylindrical Coordinate System with Dynamic Permutation Box (CCSDPB). The study involved three phases: design, implementation, and verification. In the first phase, dynamic 3D S-box, dynamic P-box, and F-function were designed. The second phase involved performing key expansion, data encryption, and data decryption. Verification includes evaluating the output of the new design using the National Institute of Standard and Technology (NIST) randomness statistical test and cryptanalysis. Results of the statistical tests show that the new design is suitable with any data stream, including a long string of identical bytes. The combination of a dynamic permutation box with a dynamic 3D S-box is an effective approach that strengthens the design resistance against attacks such as differential, linear, and short attacks, as well as and increase the randomness of outputs.

Keywords: Cylindrical Coordinate System, Dynamic 3D S-Box, Dynamic P-Box, NIST Statistical Tests

1. Introduction

Blowfish Algorithm (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[1-3]. Schneier designed BA in 1994 to replace Data Encryption Standard (DES). The 56-bit key size of DES makes the algorithm vulnerable to brute-force attack, and recent advances in differential cryptanalysis and linear cryptanalysis indicate that DES is vulnerable to other attacks as well[4].

Key expansion of BA starts with the P-array and S-boxes that use numerous 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}) and four 32-bit S-boxes have 256 entries.

A key with a maximum of 448 bits is converted into several subkey arrays up to a total of 4168 bytes.

The steps in calculating these subkeys are explained as follows.

- 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.
- P_1 XOR is conducted on the first 32 bits of the key, whereas P_2 XOR is conducted on the second 32 bits. This condition is repeated up to P_{14}. The cycle is iterated through the key bits until XOR has been performed on the entire P-array with key bits.
- BA is used to encrypt the all-zero strings by employing the described subkeys in step 1.
- P_1 and P_2 are replaced with the output of step 3.
- The output of step 3 is encrypted using modified subkeys.
- P_3 and P_4 are replaced with the output of step 5.

*Author for correspondence

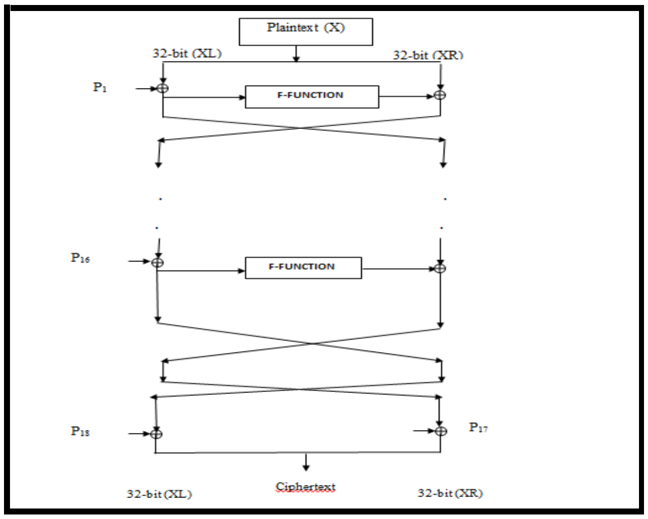


Figure 1. Encryption process in BA.

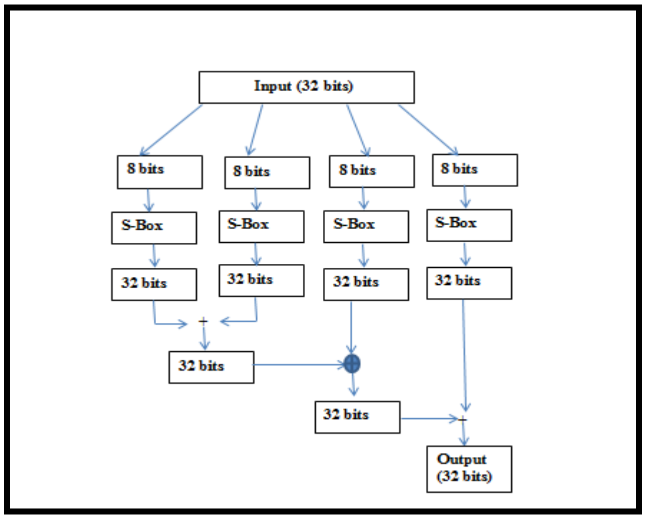


Figure 2. F-function architecture.

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

Schneier[1], Hashim et al[5] and Mahdi[6] stated that 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[7]. The same procedure was used in this study to perform key expansion in cryptographic design.

1.1 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: XL and XR. XOR is then implemented between the first 32-bit block segment (XL) and the first P-array (P1). The resulting 32-bit data are moved to the F-function that permutes the data to form a 32-bit block segment, which is XORed with the second 32-bit segment (XR). Then, 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 1 illustrates the encryption process in BA with 16 rounds.

The F-function of BA is the most complex part of the algorithm, which is the only part that employs the S-boxes. The input of the F-function is 32 bits and its output is 32 bits. The input splits into four equal quarters. Every quarter (8 bits) is substituted into 32 bits in a corresponding S-box. These 32 bits are then combined (XOR or addition modulo 2^{32}). Figure 2 describes the architecture of the F-function[3,8-10].

Decryption is similar to encryption, but P_1, P_2... P_{18} are used in reverse order. BA is significantly faster than DES when implemented on 32-bit microprocessors with large data caches, such as Pentium and powered PC[4]. However, BA does not fulfill all the requirements for a new cryptographic standard and is suitable only for applications in which the key is not often changed, such as in a communication link or an automatic file encryptor. BA is inappropriate for devices that have small memory[3,8,11-14]. Four S-boxes require large memory space (4096 bytes); thus, they are not feasible for small devices. 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 build relations between rounds[15,16]. Thus, 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 includein[5,6,13]. Hashim et al[5] 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 65543 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. The 65543 bytes (128-bits) still have a large memory requirement. Meanwhile, Mahdi[6] 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. Chandrasekaran et al[13] 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 reduced computational overhead in key generation. Despite the decreased time complexity of the original algorithm, the memory requirement increases where the modified design requires a memory of 17179869184 bytes to tabulate 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 been unsuccessful. Several studies are related to the 3D designs of a new block cipher, instead of the common 2D approach to design a new block cipher that does not compromise security.

Nakahara[17] and Ariffin[18] designed a 3D block cipher based on the AES algorithm. Nakahara[17] improved the security of the 3D block cipher by providing good diffusion in three rounds, but speed was not improved. The AES algorithm still required 22 rounds to encrypt one block of plaintext, which 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.

Considering these limitations Nakahara[17], Ariffin[18] successfully designed a 3D block cipher with byte permutation. This block cipher required only 10 rounds and provided good diffusion in three rounds. The author also tested for randomness and attacks, and obtained good results. Based on the aforementioned 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[19] successfully designed a 3D block cipher with good diffusion, but encountered a problem with speed. The method required 64 rounds to encrypt one block of plaintext. The reliability of this method was questionable because the researchers did not conduct a comprehensive test, but conducted only four NIST statistical tests for randomness. These studies Ariffin[18] and Suri and Deora[19] presented a 3D block cipher in Cartesian Coordinates System (X, Y and Z) and conducted byte permutation using a rotation 3D array (cubic). This 3D array can be rotated by $\frac{\pi}{2}, \pi$ and $\frac{3\pi}{2}$ only to perform byte permutation. The rotation in the CCS can be conducted by $\frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}, \pi, \frac{5\pi}{4}, \frac{3\pi}{2}$, only to perform byte permutation in this study. The cubic is not suitable to present the S-box (256 bytes) in 3D array. These conditions indicate that CCS is more suitable than Cartesian Coordinates System in this study. The idea of a 3D array with byte permutation has been incorporated in the present study to design a dynamic 3D S-box that can decrease the BA memory requirement. In this study, a 3D array was used to design a 3D S-box. Byte permutation was employed to permute the values of the 3D S-box. A 3D array was constructed using the CCS structure, and byte permutation was performed using the CCS transformation.

This paper is organized as follows. The first section presents the background. The second section describes the dynamic S-box. The third section includes coordinate systems, and the fourth section presents a detailed explanation of the methodology. The fifth section presents the results, the sixth section concludes this paper, and the seventh section presents suggestions for future research.

2. Dynamics S-Box

S-boxes are lookup tables that map n bits to m bits. Several ways can be employed to construct and test good S-boxes for ciphers. Numerous block ciphers depend on the traditional Shannon idea of confusion and diffusion. Typically, confusion results from certain forms of substitute “S-boxes”[39].

A fixed or static S-box has no relation with a cipher key, and their contents are not related to that of the secret key. The function 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 concerns the elements of the fixed structure of S-boxes. An example of a fixed S-Box is that used in DES. A fixed S-box allows attackers to study the S-box properties and locate its weak points.

Dynamic substitution is a type of extended substitution that is similar to simple substitution but 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, typically originating from a pseudo random sequence generator. This process can be performed after each element is substituted. The substitution of random elements is based on the permutation algorithm[20].

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 is less susceptible to arguments of "hidden" properties. However, the overall performance of S-boxes in terms of security and speed has not been sufficiently investigated[1,20-23].

According to the Elkamchouchi and Makar[24], ciphers with dynamic S-boxes are more secure than those with fixed S-Boxes. A dynamic S-box may be considered as a block box with two inputs (i.e., data in and random in) and one output[20,22]. The S-box starts out as completely unknown. When data are translated through the S-box, the particular substitution is potentially known. However, 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[20,27]. Numerous studies[23,25,26,28-30] have attempted to modify the AES S-box to make it dynamic instead of fixed to increase algorithm security.

These studies show that most attempts are based on the application of different byte permutation mechanisms by rearranging the location of the elements in the S-box using Secret Keys (SKs) to achieve a dynamic S-box, which increases the security of the original algorithm. A dynamic S-box is protected against differential and linear attacks because its structure is completely hidden from the cryptanalyst[1,21-23]. In this study, a dynamic S-box in BA has been improved by using byte permutation based on SKs. A dynamic S-box is achieved by rearranging the location of the elements in the 3D S-box after each substitution at each round with every block of plaintext. Thus, any trail from cryptanalysis to build relations between rounds leads to failure. In addition, the randomness of the algorithm is enhanced. Random SKs with byte permutation are proven to be good mechanisms; thus, this method is adopted in our study.

3. Secret Key Generation

SKs can be generated through various methods. Krishnamurthy and Ramaswamy[25] generated SKs using four methods. The first method used the last byte from the round keys. The second used XOR among the values of all bytes in round keys. The third 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 is similar to the third, except that it performs XOR between the values of all bytes in round keys. Mohammad, Rohiem and Elbayoumy[26] generated SKs for relocation by dividing SK mod 256, and the result was added to the index to obtain the new location of the element. Stoianov[28] generated keys by dividing pre-selected bytes of the SK and dividing them by four. Juremi et al[23] generated keys by applying XOR to all bytes of the round keys. Mahmoud et al[30] generated SKs using Linear Feed Back Shift Register (LFSR). Suri and Deora[19,31] 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. Methods such as LFSR and random number generator are used to generate various random multiple SKs. LFSR requires additional time (overhead time), which makes 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 suitable than LFSR in generating multiple random numbers without adding overhead time. These multiple random numbers can be used as random SKs for the permutation of the values of the 3D S-box.

4. Coordinate Systems

Coordinate systems can 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 limited or no significant use. Thus, this study focuses on orthogonal systems. Examples of orthogonal systems include the Cartesian or rectangular, cylindrical, spherical, elliptical cylindrical, parabolic cylindrical, conical, prolate spheroidal, and ellipsoidal. However, the most common coordinate systems are the Cartesian, cylindrical, and spherical systems.

The Cartesian coordinate system is the most commonly used coordinate system. However, in applications where rotation is considered, a special form of Cartesian coordinate based on a circle is used[32-34].

4.1 Cylindrical Coordinate System

A cylindrical coordinate system can conveniently solve problems related to cylindrical symmetry. A Point (P) in a cylindrical coordinate system is represented as (ρ, Ø z), where ρ is the radius of the cylinder passing through P or the radial distance from the z-axis, Ø is the angle between the x-axis, and the projection of the point (ρ, Ø z) onto the xy-plane, and z is the same as that in a Cartesian system (Figure 3)[34].

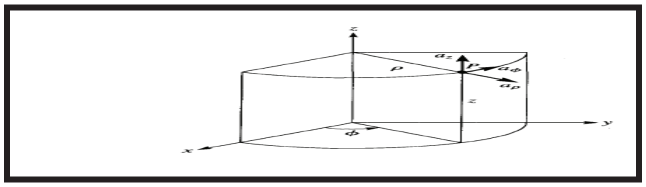


Figure 3. Cylindrical coordinate system.

The ranges of the variables are as follows:

$0 \le \rho < \infty$

$0 \le Ø < 2\pi$

$-\infty < z\ \infty$

The level surface of points, such as $z = z_p$, defines a plane. A few contours that have constant values of ρ can be drawn. These "level contours" are circles. By contrast, if z is not restricted to $z = z_\rho$, as shown in Figure 4, then the level surfaces for constant values of ρ would be cylinders that are coaxial with the z-axis.

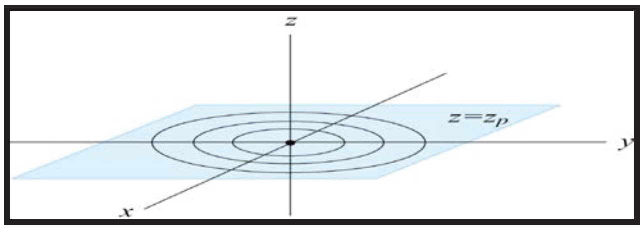


Figure 4. Level surfaces for the coordinate.

In Figure 4, all points that lie on a ray from the origin to infinity passing through P have the same value of Ø. For any random point, Ø can take on values from 0 < Ø < 2π 2. In Figure 5, "level surfaces" for the angular coordinates are drawn. The coordinates (ρ, Ø) in the plane $z = z_\rho$ are called plane polar coordinates[35].

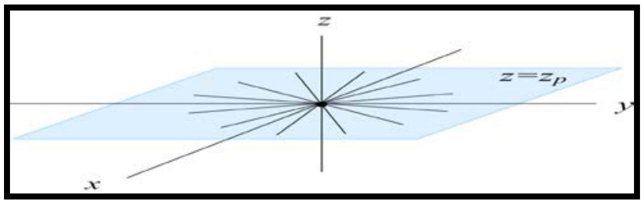


Figure 5. Level surfaces for the angle coordinate.

5. Methodology

This study was conducted in three phases: design, implementation, and verification.

5.1 Phase 1: Design

Phase 1 was conducted in 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 these steps.

5.1.1 Dynamic 3D S-Box

The process of designing a 3D S-box consists of three parts: 1. Generating random SKs, 2. Defining the transformation of the right cylinder and 3. Conducting byte permutation (byte relocation and byte transformation). The 3D S-box structure is initially prepared by converting the right cylinder (Figure 6.) into a 3D S-box using the following matrix:

A=$[a_{ijk}]_{884}$ where a_{ijk} is related with point $P(\rho_i, Ø_j, z_k)$ such that

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

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

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

Figure 7 shows the cross-section of the right cylinder.

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

Figure 8 illustrates the representation of the right cylinder in 3D array. Each square (array 8×8) is a set of 64 bytes that represent a section of the cylindrical coordinate system for the right cylinder. Each row in the array

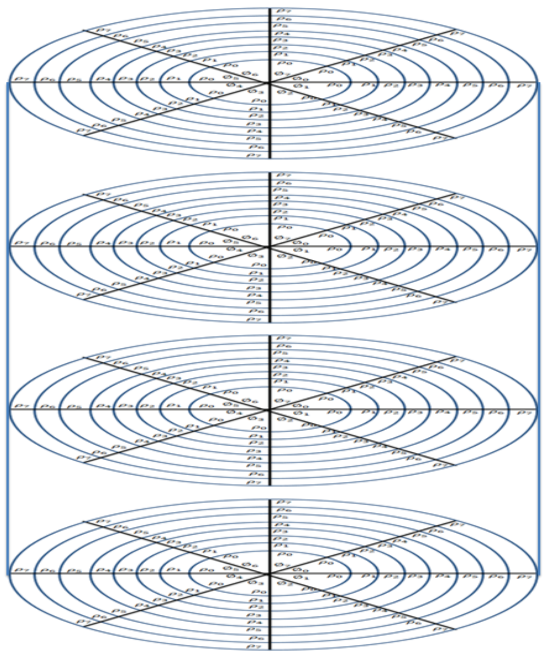


Figure 6. Right cylinder.

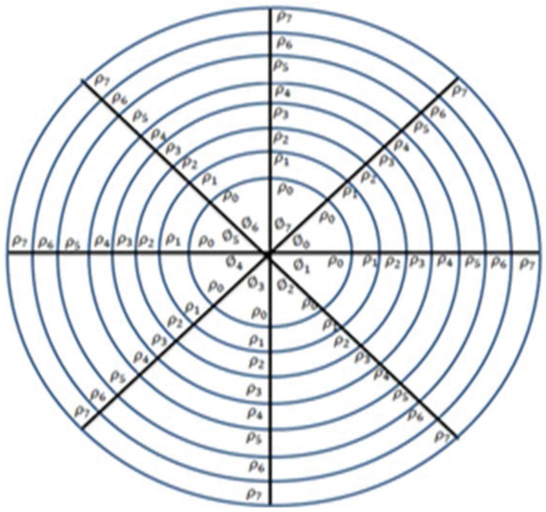


Figure 7. Cross-section of the right cylinder.

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 represents a row number (ρ), the second represents a column number (ϕ), and the third represents a section number (z). Therefore, any point in the right cylindrical coordinate is referred to as a_{ijk}.

In BA, each quarter (8 bits) is used as an entry point to one of the S-boxes, thereby requiring four S-boxes. In the new design, all four quarters (each one consisting of 16bits) are used as entry to the same 3D S-box. Two procedures of byte permutation were applied to permute the elements of the 3D S-box after each entry. Every section in the 3D S-box (right cylinder) can be divided into four sets of elements called quarters (Q_0, Q_1, Q_2, and Q3). These quarters represent circles, half circles, tracks, and set of random points in the right cylinder.

Once the structure of the 3D S-box is prepared, the following activities are conducted:

5.1.1.1 Generating Random SKs

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

Seed = (XL XOR round subkey) + block sequence of the plaintext (1)

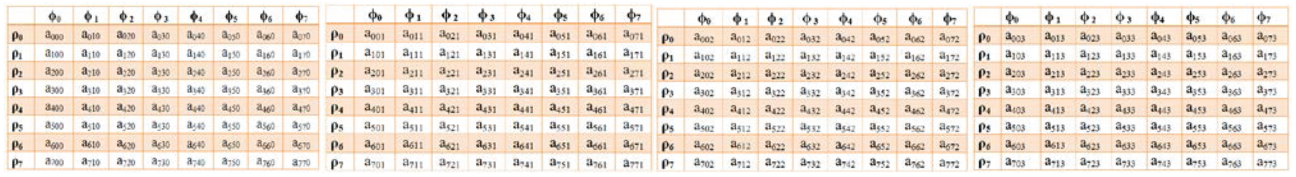
Based on Equation (1), the seed of the rand function is the left side of the round input (XL) that was XORed with a round subkey. The result is added to the block sequence of the plaintext. Thus, every block has a different seed in the encryption process. Five sets of SKs are generated using the rand function in C++. The random SKs are in the interval [0,3] and are given as follows.

$SK_i=k_{ij}$, where i=0, 1, 2, 3, 4 and 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 would be chosen and that

Section$_0$ (8×8) bytesSection$_1$ (8×8) bytes Section$_2$ (8×8) bytesSection$_3$ (8×8) bytes



	ϕ0	ϕ1	ϕ2	ϕ3	ϕ4	ϕ5	ϕ6	ϕ7
ρ0	a_{000}	a_{010}	a_{020}	a_{030}	a_{040}	a_{050}	a_{060}	a_{070}
ρ1	a_{100}	a_{110}	a_{120}	a_{130}	a_{140}	a_{150}	a_{160}	a_{170}
ρ2	a_{200}	a_{210}	a_{220}	a_{230}	a_{240}	a_{250}	a_{260}	a_{270}
ρ3	a_{300}	a_{310}	a_{320}	a_{330}	a_{340}	a_{350}	a_{360}	a_{370}
ρ4	a_{400}	a_{410}	a_{420}	a_{430}	a_{440}	a_{450}	a_{460}	a_{470}
ρ5	a_{500}	a_{510}	a_{520}	a_{530}	a_{540}	a_{550}	a_{560}	a_{570}
ρ6	a_{600}	a_{610}	a_{620}	a_{630}	a_{640}	a_{650}	a_{660}	a_{670}
ρ7	a_{700}	a_{710}	a_{720}	a_{730}	a_{740}	a_{750}	a_{760}	a_{770}

	ϕ0	ϕ1	ϕ2	ϕ3	ϕ4	ϕ5	ϕ6	ϕ7
ρ0	a_{001}	a_{011}	a_{021}	a_{031}	a_{041}	a_{051}	a_{061}	a_{071}
ρ1	a_{101}	a_{111}	a_{121}	a_{131}	a_{141}	a_{151}	a_{161}	a_{171}
ρ2	a_{201}	a_{211}	a_{221}	a_{231}	a_{241}	a_{251}	a_{261}	a_{271}
ρ3	a_{301}	a_{311}	a_{321}	a_{331}	a_{341}	a_{351}	a_{361}	a_{371}
ρ4	a_{401}	a_{411}	a_{421}	a_{431}	a_{441}	a_{451}	a_{461}	a_{471}
ρ5	a_{501}	a_{511}	a_{521}	a_{531}	a_{541}	a_{551}	a_{561}	a_{571}
ρ6	a_{601}	a_{611}	a_{621}	a_{631}	a_{641}	a_{651}	a_{661}	a_{671}
ρ7	a_{701}	a_{711}	a_{721}	a_{731}	a_{741}	a_{751}	a_{761}	a_{771}

	ϕ0	ϕ1	ϕ2	ϕ3	ϕ4	ϕ5	ϕ6	ϕ7
ρ0	a_{002}	a_{012}	a_{022}	a_{032}	a_{042}	a_{052}	a_{062}	a_{072}
ρ1	a_{102}	a_{112}	a_{122}	a_{132}	a_{142}	a_{152}	a_{162}	a_{172}
ρ2	a_{202}	a_{212}	a_{222}	a_{232}	a_{242}	a_{252}	a_{262}	a_{272}
ρ3	a_{302}	a_{312}	a_{322}	a_{332}	a_{342}	a_{352}	a_{362}	a_{372}
ρ4	a_{402}	a_{412}	a_{422}	a_{432}	a_{442}	a_{452}	a_{462}	a_{472}
ρ5	a_{502}	a_{512}	a_{522}	a_{532}	a_{542}	a_{552}	a_{562}	a_{572}
ρ6	a_{602}	a_{612}	a_{622}	a_{632}	a_{642}	a_{652}	a_{662}	a_{672}
ρ7	a_{702}	a_{712}	a_{722}	a_{732}	a_{742}	a_{752}	a_{762}	a_{772}

	ϕ0	ϕ1	ϕ2	ϕ3	ϕ4	ϕ5	ϕ6	ϕ7
ρ0	a_{003}	a_{013}	a_{023}	a_{033}	a_{043}	a_{053}	a_{063}	a_{073}
ρ1	a_{103}	a_{113}	a_{123}	a_{133}	a_{143}	a_{153}	a_{163}	a_{173}
ρ2	a_{203}	a_{213}	a_{223}	a_{233}	a_{243}	a_{253}	a_{263}	a_{273}
ρ3	a_{303}	a_{313}	a_{323}	a_{333}	a_{343}	a_{353}	a_{363}	a_{373}
ρ4	a_{403}	a_{413}	a_{423}	a_{433}	a_{443}	a_{453}	a_{463}	a_{473}
ρ5	a_{503}	a_{513}	a_{523}	a_{533}	a_{543}	a_{553}	a_{563}	a_{573}
ρ6	a_{603}	a_{613}	a_{623}	a_{633}	a_{643}	a_{653}	a_{663}	a_{673}
ρ7	a_{703}	a_{713}	a_{723}	a_{733}	a_{743}	a_{753}	a_{763}	a_{773}

Figure 8. Representation of the right cylinder in 3D array.

Table 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

Table 2. Eight transformations of the right cylinder

Cases	Transformation	Kind
Case $z_0 = 0$	$T: S_{k_n} \to S_{k_n}$	
$\rho_0 = 0, \emptyset_0 = 0$	$T_1(\rho, \emptyset, z) = (\rho, \emptyset, z)$	1
$\rho_0 = 0, \emptyset_0 \neq 0$	$T_2(\rho, \emptyset, z) = (\rho, \emptyset \pm \emptyset_0, z)$	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 $z_0 \neq 0$	$T: S_{k_n} \to S_{k_m}$	
$\rho_0 = 0, \emptyset_0 = 0$	$T_5(\rho, \emptyset, z) = (\rho, \emptyset, z \pm z_0)$	5
$\rho_0 = 0, \emptyset_0 \neq 0$	$T_6(\rho, \emptyset, z) = (\rho, \emptyset \pm \emptyset_0, z \pm z_0)$	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

all elements of the quarters in the 3D S-box would 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 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 1 presents an example of five sets of SKs. The SKs in SK_1 represent the quarter number in the first section and 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.

5.1.1.2 *Defining the Transformations of the Right Cylinder*

The right cylinder is used in the design defined in the cylindrical coordinate as follows:

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

The transformation of the right cylinder can be expressed as follows:

$$f(\rho, \emptyset, z) = (\rho \mp \rho_0, \emptyset \mp \emptyset_0, z \mp z_0)$$

where

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

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

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

This transformation converts point $P(\rho, \emptyset\ z) \in S$ to point $P'(\rho, \emptyset\ z)$, which also belongs to S. The eight types of transformations on the right cylinder S in Table 2 are conducted as follows:

1. In this study, the right cylinder is divided into four sections, as shown in Figure 7, which is indexed to $k_0, k_1, k_2, k_3, \in K = \{0, 1, 2, 3\}$.
2. In every section of the right cylinder, we have eight nested circles (contours), as shown in Figure 6.

In mathematical notation, we consider $S_k = \{Q_{jk}$: Q_{jk} as a circle or half circle or track or a set of random points $j = 1,2,3,4\}$ for $k \in K$.

5.1.1.3 *Byte Permutation*

Two procedures of byte permutation based on SKs are conducted in every round to permute the elements of the 3D S-box. The first procedure is known as Byte Relocation

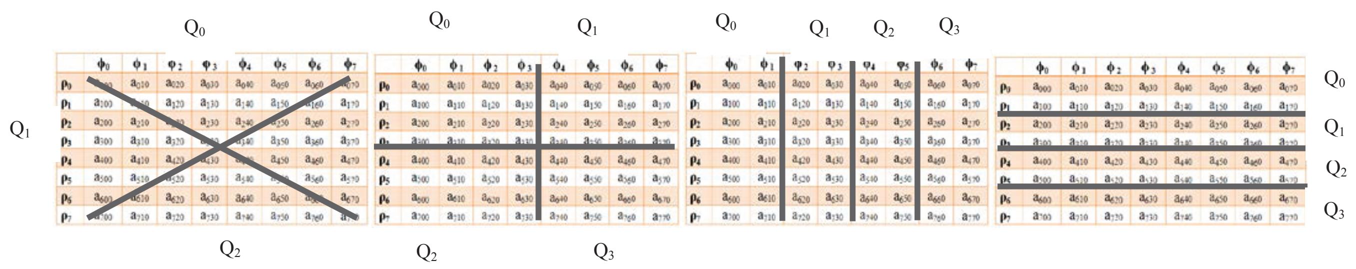


Figure 9. Quarters in the first section with D_0, D_1, D_2 and D_3.

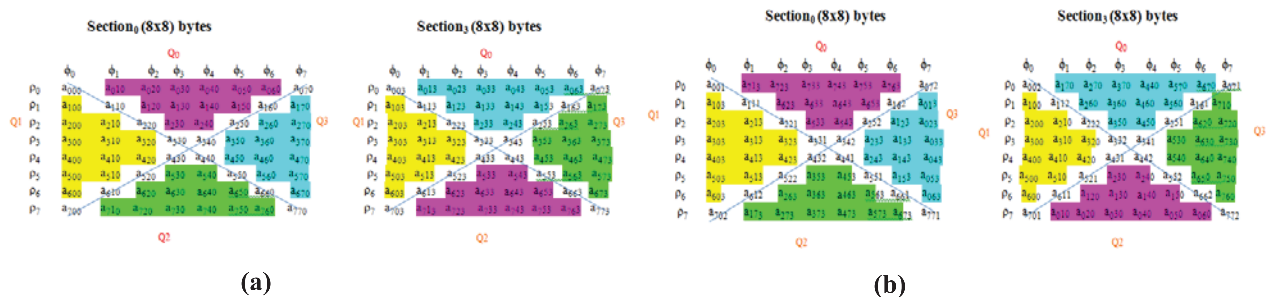


Figure 10. D_0 process for the first section. **(a)** Before D_0 process. (b) After D_0 process.

(BR), which is used to generate one dynamic 3D S-box. The second procedure of byte permutation is called Byte Transformation (BT), which is used to generate three dynamic 3D S-boxes.

5.1.1.4 Byte Relocation

BR is used in four procedures, namely, D_0, D_1, D_2, and D_3, 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 $D_0 \ldots D_3$ until 10 rounds are completed. $D_0 \ldots D_3$ are conducted on the quarters of the 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 in a manner that corresponds to T_5 when 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 process of dividing into quarters differs for D0, D1, D2 and D3. Figure 9 illustrates the division process of the sections with each BR on the first section (section0) only. The following algorithms are conducted to perform BR:

```
Algorithm 1: Byte relocation
Input: Section$_0$, Section$_1$, Section$_2$, and Section$_3$ of the key expansion part of the algorithm; five sets of random SKs.
Output: Section$_0$, Section$_1$, Section$_2$, and Section$_3$ after application of BR.
Each section is divided into four quarters with D (i-thround mod 4).
Two sections are selected depending on the first SK set.
Swapping between the elements of the quarters in the selected section depends on the last four SK sets.
Swapping between the elements of the main diagonals with the elements of the second diagonals is performed in the sections with $D_0$ only.
Swap ($L_0$, $L_1$)
Swap ($L_2$, $L_3$)
Swap ($S_0$, $S_2$)
($S_1$, $S_3$)
// $L_0$, $L_1$, $L_2$, $L_3$, $S_0$, $S_1$, $S_2$, and $S_3$ are the main and second diagonals of the sections, respectively.
```

An Example of the BR Process is presented as Follows; Using the SKs shown in Table 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 the SKs. Figure 10 illustrate the D_0 process for the first section using only the SKs in Table 1.

5.1.1.5 Byte Transformation

BT was conducted on elements of the sections of the 3D S-box (right cylinder) 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 performed 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 and 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 was used to permute the elements of sections of the 3D S-box after every two bytes were substituted. T_8, T_4 and T_6 are secret. Figure 11 illustrates the rotation of a single circle of any section in the cylinder, where $\phi_0 = \pi/4$.

Figure 12 explains the rotation by ϕ_0 on the elements of the first section, where$\phi_0 = \pi/4$.

5.1.2 Dynamic P-Box

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

- Inputting the seed of the rand function using the output of the 3D S-box (8 bytes).
- Generating 64 random numbers between 0 and 63 from the rand function and storing these numbers in vector n.
- Initializing the P-box (vector) with values from 0 to 63.
- Swapping the values of the index P-box [i] with P-box[n[i]],where i=0...63.

5.1.3 Designing a New F-Function

The most complex part is designing the F-function, known as Coordinate Cylindrical System with Dynamic Permutation Box (CCSDPB), and which includes a dynamic 3D S-box as well as 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 that are used as indices to the $a_{ijk,}$ where the first subpart, used as an index to the row number of $a_{ijk,}$ represents the first three even bits of the byte; the second subpart, used as an index to the column number of a_{ijk}, represents the first three odd bits of the byte; and the third subpart, used as an index to the section number, represents the last two bits of the byte.

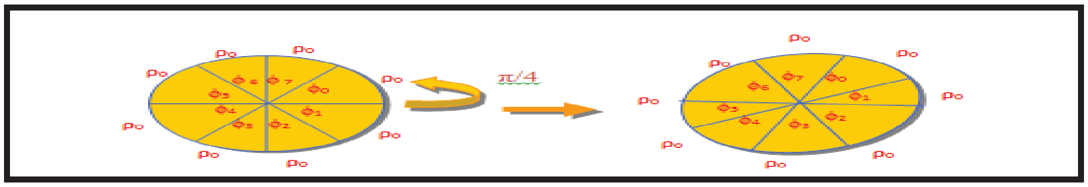
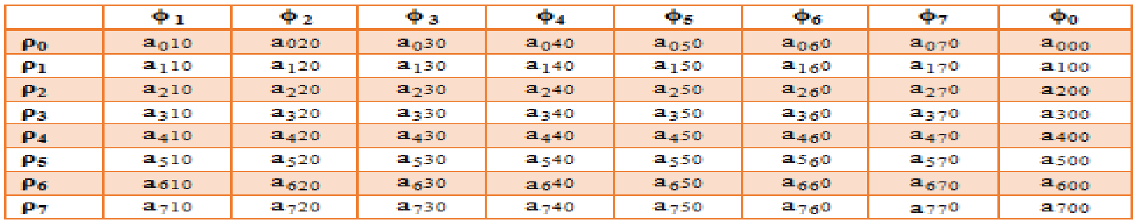


Figure 11. Rotation of a circle ($\phi_0 = \pi/4$).



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

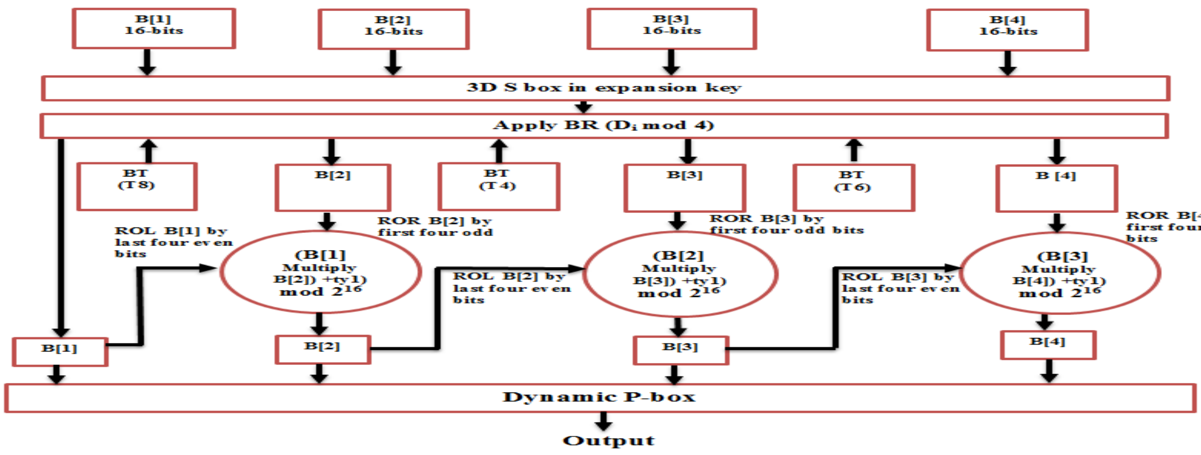

B[1] 16-bits
B[2] 16-bits
B[3] 16-bits
B[4] 16-bits
3D S box in expansion key
Apply BR (Di mod 4)
BT (T 8)
B[2]
BT (T 4)
B[3]
BT (T 6)
B [4]
ROL B[1] by last four even bits
ROR B[2] by first four odd
(B[1] Multiply B[2]) +ty1) mod 2^16
ROL B[2] by last four even bits
ROR B[3] by first four odd bits
(B[2] Multiply B[3]) +ty1) mod 2^16
ROL B[3] by last four even bits
ROR B[4] by first four odd bits
(B[3] Multiply B[4]) +ty1) mod 2^16
B[1]
B[2]
B[3]
B[4]
Dynamic P-box
Output

Figure 13. F-function (CCSDPB).

The output of each quarter from the 3D S-box was multiplied by the previous quarter after rotating them, except for the first quarter, and the result was added to the round subkey part. Then, the result is modulo 2^{16}, which means that the round subkey part is 16 bits. The multiplication result for the second quarter with the first quarter was added to the first 16 bits from the round subkey, whereas the multiplication result for the third quarter with the 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 the third quarter. The number of rotations (right and left) differs 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 is not the same. Then, the outputs of the four quarters are combined based on the values from the dynamic P-box. Figure 13 illustrates the diagram of the CCSDPB function.

The steps taken to perform the CCSDPB are shown as follows:

```
    Algorithm 2: 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 the
dynamic P-box.
```

5.2 Phase 2 Implementation

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

5.2.1 Key Expansion

The key expansion must be started before 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, amounting to a total of352 bytes. The subkeys 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. 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[1,5,6].

5.2.2 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 is called RAF, which consists of 128-bit data of plaintext and provides an output of 128-bit ciphertext with variable key reaching 640 bits. The external structure of the new design is the same as that of the BA. The encryption steps are enumerated as follows:

```
Algorithm 3: 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
```

Encryption and decryption are structurally identical in a Feistel cipher. The subkeys used during encryption in every round are reused during decryption but in a reverse order. Thus, the steps taken to perform the data decryption are similar to that used for data encryption.

```
Algorithm 4: RAF Data Decryption
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
```

5.3 Verification

The verification phase includes verification of the output algorithm by NIST Statistical Tests(40) and cryptanalysis of differential, linear, and short attacks.

5.3.1 NIST Statistical Tests

The output of the new design rounds (RAF) were verified using 15 NIST statistical tests on two data types and samples. The data types are as follows: random plaintext/random 128-bit keys, and video files. Each type of data included a sample size of 128 sequences. The random plaintext/random 128-bit keys were used in selecting the finalists for the AES block cipher (18, 36–38). In this study, we used the same data and sample size used previously.

$$p\alpha = (1 - \alpha) - 3\sqrt{\frac{\alpha(1-\alpha)}{s}} \quad (2) \qquad (2)$$

Where s is the number of samples and a is the significance level. The proportion of sequences that passed a specific statistical test is equal or greater than *pa*, as defined in Equation 2.The proportion value of testing data is as follows:

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

In this part, the evaluation includes Partial Round Testing (PRT) and Full Round Testing (FRT). Twofish rounds are tested in pairs. Twofish is a Feistel network Thus, each round leaves several data bits unchanged, thereby making the Twofish appear nonrandom under test conditions after one round. However, all data bits are affected after two rounds. Thus, paired Twofish rounds are evaluated, i.e., even numbered rounds from 2 to 14[37]. Therefore, in this study, the PRT was conducted in pairs from two to eight rounds for the new algorithm design. In FRT, the output (ciphertext) was tested[22,38].

5.3.1.1 Random Plaintext/Random 128-Bit Keys

The Blum BlumShub (BBS) pseudorandom bit generator is a cryptographically secure generator for random plaintext/random 128-bit keys[41]. Moreover, 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 8128 128-bit ciphertext blocks (1040384 bits) using 8128 random plaintext blocks of the same length and random 128-bit key. Figure 14 illustrates the NIST results with random plaintext/random 128-bit keys. In each Figure, the dashed line depicts the smallest proportion that satisfies the 0.01 acceptance criterion, whereas the solid line depicts the expected proportion.

5.3.1.2 Video Files

The data set consists of 128 sequences of video files (.flv, .wmv). Each file contained one sequence resulting from the concatenation of 12290 (1573120 bits) 128-bit ciphertext blocks using 12290 128-bit plaintext blocks and random 256-bit key. Figure 15 illustrates the NIST results with video files.

At the end of the second round, the output from RAF is random (the first round pair) because majority of the 188 statistical tests show values greater than 96%. Subsequent rounds also produced similar results.

5.3.2 Cryptanalysis

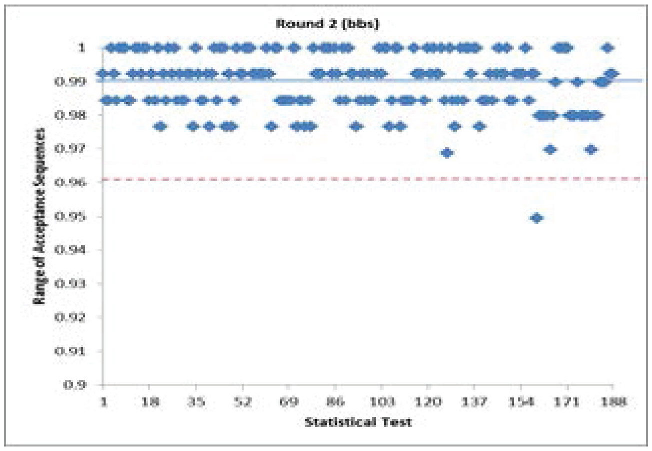
This section presents the explanation on the resistance of RAF on differential, linear, and short attacks.

5.3.2.1 Differential and Linear Attacks

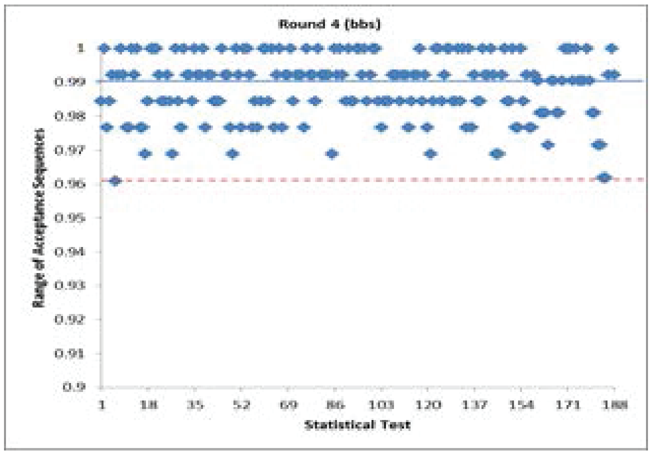
Using the dynamic 3D S-box and dynamic P-box is a significant feature of the RAF. The dynamic 3D S-box and dynamic P-box protect the algorithm against differential and linear cryptanalysis. The specific properties of the dynamic 3D S-box, specifically the structure that is completely unknown to the cryptanalyst, assist and support the RAF to remain resistant against attacks. Meanwhile, the dynamic P-box is introduced to provide security and protection to the 3D S-box output. 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 encrypting one block of plaintext, 40 dynamic 3D S-boxes are generated. Therefore, the 3D S-box is not a fixed entity. Any attempt to construct iterative linear or differential relations between rounds, such as the previous attacks on the original algorithm BA[15,16], leads to failure.

5.3.2.2 Short Attack

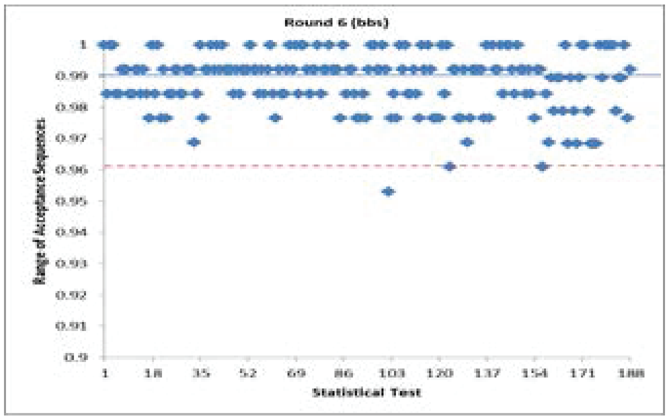
Shortcut attack is the trial disclosure of the internal structure of the cryptographic design[12,20,24,25,27,29]. The RAF is resistant to existing shortcut attacks that attempt to recover the specific internal structure of the cipher. The 3D S-box in RAF has entry 4 x 8 x 8 with probability of repeating several elements. A cryptanalyst aiming to decode the RAF needs to generate all possible 3D S-box and P-box values. This task is almost impossible because the cryptanalysis must attempt all possible cases $(2^n)^n$ with each section of the 3D S-box. Moreover, four sections are present in the 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 10 rounds has 4 different 3D S-boxes, thereby creating a total of 40 different 3D S-boxes for RAF. Therefore, the cryptanalyst must attempt a total of $((2^{4n^3})^4)^{10}$ times the first block. Moreover, the 3D S-box in RAF changes with each encryption process of the plaintext block. Thus, the cryptanalyst must try the same computation $((2^{4n^3})^4)^{10}$ again, which is equal to 2^{81920} for every block of plaintext. In other words, if five blocks of plaintext are present, then the cryptanalysis must perform five times of 2^{81920}. Moreover, one dynamic P-box in every round has $2^6!$ Possibilities. Therefore, 10 rounds require 10 dynamic P-boxes with possibility $(2^6!)^{10}$ for only one block of plaintext, thereby making the RAF a highly secure algorithm. Therefore, attackers would pay a higher price.



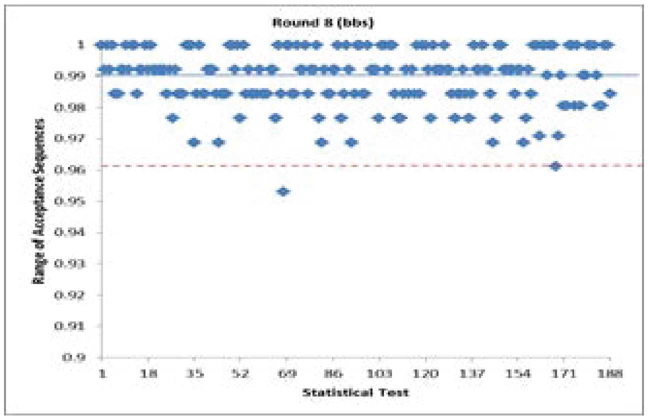
(a)



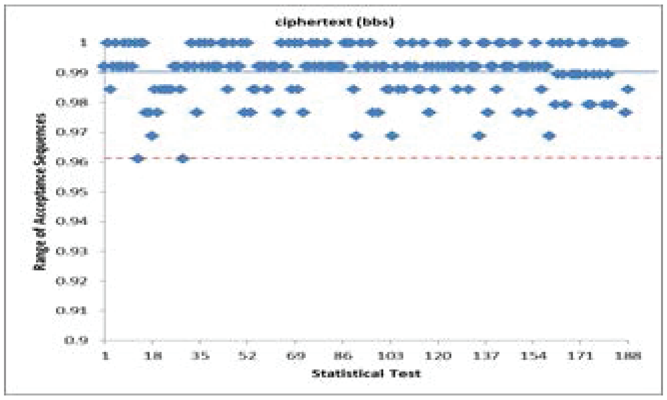
(b)



(c)

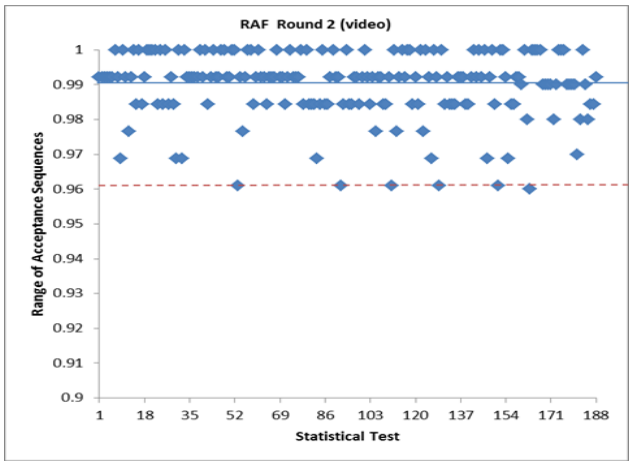


(d)

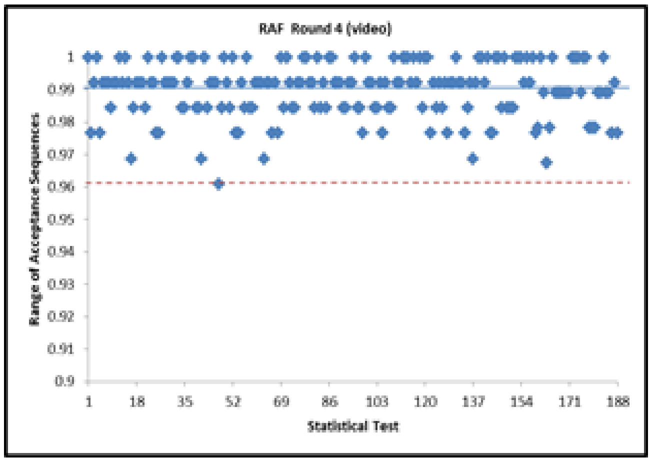


(e)

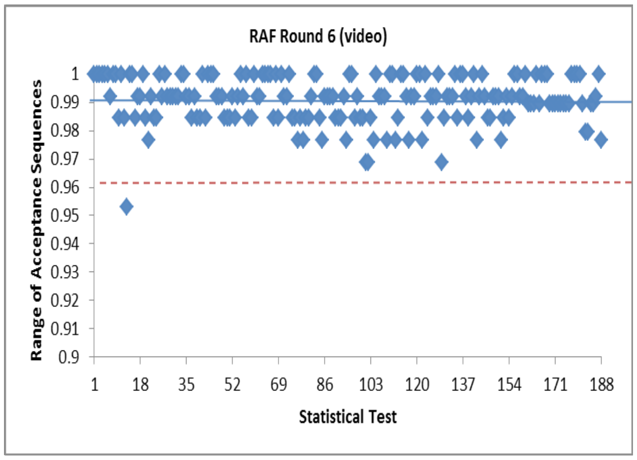
Figure 14. Results of the NIST on random plaintext/random 128-bit keys in the **(a)** Second round, **(b)** Fourth round, **(c)** Sixth round, **(d)** Eighth round and **(e)** Tenth round.



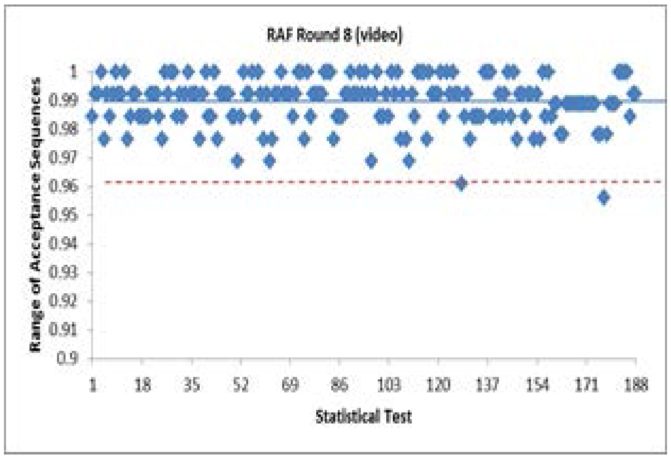
(a)



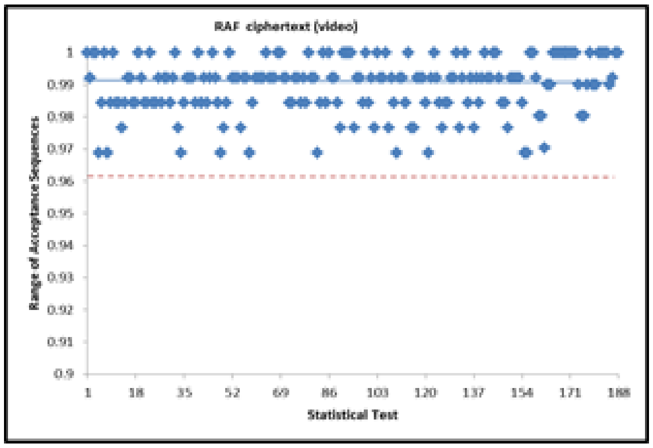
(b)



(c)



(d)



(e)

Figure 15. Results of the NIST on video in the **(a)** Second round, **(b)** Fourth round, **(c)** Sixth round, **(d)** Eighth round and **(e)** Tenth round.

6. Results

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

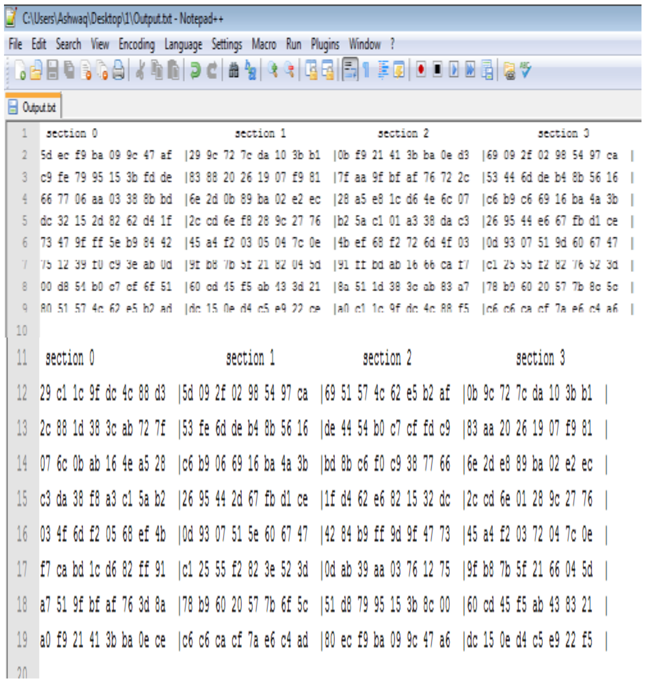
6.1 Dynamic 3D S-Box

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

6.1.1 Random Secret Keys

In this section, only the random SKs for Round 0 are shown. The generated random SKs are as follows:
Round 0
Seed=12664626897530995354
1 3 0 2
0 1 3 2
2 0 1 3
2 3 1 0
2 0 1 3

The input and output for the first round are shown as follows (Figure 16):


C:\Users\Ashwaq\Desktop\1\Output.txt - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
Output.txt

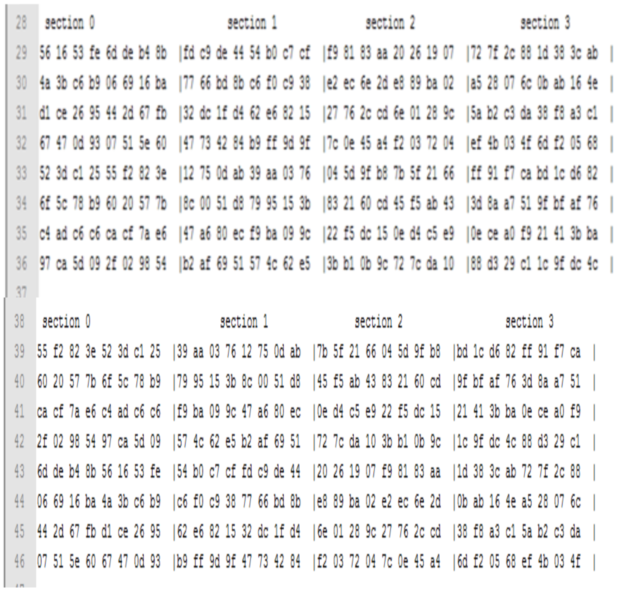
```
 section 0                section 1                section 2                section 3
5d ec f9 ba 09 9c 47 af  |29 9c 72 7c da 10 3b b1  |0b f9 21 41 3b ba 0e d3  |69 09 2f 02 98 54 97 ca  |
c9 fe 79 95 15 3b fd de  |83 88 20 26 19 07 f9 81  |7f aa 9f bf af 76 72 2c  |53 44 6d de b4 8b 56 16  |
66 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  |

 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  |
```

6.1.2 BT

The results from BT are as follows:

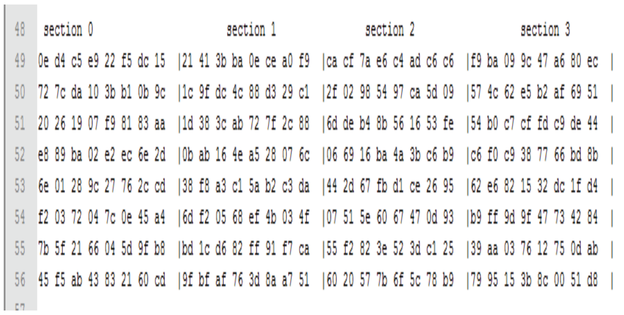
--T8 =>$\rho_0 \neq 0$, $\phi_0 \neq 0$, $z_{0\neq}0$ =>$\rho_0=7$,$\phi_0=2$,$z_0=3$------T4 => $\rho_0 \neq 0$, $\phi_{0\neq}0$, $z_{0=}0$ =>$\rho_0=4$, $\phi_0=4$---



```
 section 0                section 1                section 2                section 3
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  |

 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  |
```

-- 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 16. Input: 3D S-box from key expansion part Output: 3D S-box after applying BR (D_0).

6.2 Dynamic P-box Values

The values produced are as follows:

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,

6.3 CCSDPB

The CCSDPB and round outputs are shown as follows:

Output (CCSDPB): fc7673dcbfa7abfe
Output (Round 0): 11d5f3673b76cf849942313010cbc1af

7. Conclusion

The new design has been implemented and tested during the design and verification phases. Several conclusions are drawn, of which the most significant are summarized as follows:

- The function of the CCSDPB, which is based on a CCS with a dynamic P-box and multiple SKs, provides high resistance against differential and linear attacks. Thus, this technique is better than the BA because the dynamic 3D S-box and permutation box are not fixed entities. However, this technique is changeable in every round with every block by using dynamic permutation byte based on different SKs. Any attempt from attackers is frustrated, including the construction of iterative linear and differential relations.
    - CCS can reduce memory requirements.
    - These SKs are generated from the random function with numerous variables, one of which is the plaintext sequence. These features not only strengthen the security of the generated 3D S-box but also make the new design compatible with any data type, including long string identical data.
- Based on the results of the NIST tests, this design succeeded in encrypting all file formats without restrictions on the contents of the files. This feature is applicable despite a large string of identical bytes because this approach is based on the sequence of the plaintext as the secret key.

8. Future Research

In accordance with the present study, future research can be conducted on the following topics:

- Analyzing the performance of the new design based on the following factors: avalanche affect and correlation coefficient.
- Analyzing the 3D S-box based on criteria S-box.
- Recommending the development of a new procedure to generate the 3D S-box and p-array in the new design for a new CCSDPB function.

9. References

1. Schneier B. Description of a new variable-length key, 64-bit block cipher (Blowfish). Fast Software Encryption. Berlin Heidelberg: Springer; 1994. p. 191-204.
2. Kumar R, Pradeep E, Naveen K, Gunasekaran R. A novel approach for enciphering data of smaller bytes. International Journal of Computer Theory and Engineering. 2010; 2:654-9.
3. Cornwell J. Blowfish Survey. Department of Computer Science. Columbus: GA Columbus State University; 2012. p. 1-6.
4. Halagali B. Designing the S-boxes of blowfish algorithm using linear congruential generator. ASM's International E-Journal of Ongoing Research in Management and IT; 2013. p. 1-11.
5. Hashim A, Al-Qarrawy S, Mahdi J. Design and implementation of an improvement of blowfish encryption algorithm. IJCCCE. 2009; 9(1):1-15.
6. Mahdi J. Design and implementation of proposed BR encryption algorithm. IJCCCSE. 2009; 9(1):1-17.
7. Kazlauskas K, Kazlauskas J. Key-dependent S-box generation in AES block cipher system. Informatics. 2009; 20(1):23-34.
8. Schneier B. Applied Cryptography. Protocols, Algorithms and Source Code in C. John Wiley and Sons, Inc.; 1996.
9. Bagad V, Dhotre I. Cryptography and Network Security. 2nd ed. Pune, India: Technical Publications; 2008.
10. Tilborg V, Jajodia H. Encyclopedia of Cryptography and Security. 2nd ed. New York, USA: Springer; 2011. p. 1435.
11. Wang Z, Graham J, Ajam N, Jiang H. Design and optimization of hybrid MD5-blowfish encryption on GPUs. Proceedings of 2011 International Conference on Parallel and Distributed Processing Techniques and Applications; Las Vegas, Nevada, USA. 2011. p. 18-21.
12. Zhang R, Chen L. A block cipher using key-dependent S-box and P-boxes. IEEE International Symposium on Industrial Electronics; 2008. p. 1463-8.
13. Chandrasekaran J, Subramanyan B, Raman S. Ensemble of blowfish with chaos based S-box design for text and image encryption. IJNSA. 2011; 3(4):165-73.
14. Milad A, Muda H, Noh Z, Algaet M. Comparative study of performance in cryptography algorithms (Blowfish and Skipjack). Journal of Computer Science. 2012; 8(7):1191-7.
15. Vaudenay S. On the weak keys of Blowfish. Fast Software Encryption. Berlin, Heidelberg: Springer; 1995. p. 27-32.
16. Nakahara J. A linear analysis of Blowfish and Khufu. Information Security Practice and Experience. Berlin, Heidelberg: Springer; 2007. p. 20-32.
17. Nakahara J. 3D: A three-dimensional block cipher. Cryptology and Network Security. Berlin, Heidelberg: Springer; 2008. p. 252-67.
18. Ariffin S. A human immune system inspired byte permutation of block cipher [PhD thesis]. Malaysia: University of Putra; 2012.
19. Suri P, Deora S. 3D array block rotation cipher: An improvement using lateral shift. Global Journal of Computer Science and Technology. 2011; 11(19):1-8.
20. Ritter T. Substitution cipher with pseudo-random shuffling: The dynamic substitution combiner. Cryptologia. 1990; 14(4):289–303.

21. El-Ramly S, El-Garf T, Soliman H. Dynamic generation of S-boxes in block cipher systems. IEEE Proceedings of the 18th National Radio Science Conference; 2011. p. 389-97.
22. Ali F. A New 128-Bit Block Cipher [PhD thesis]. Malaysia: University of Putra; 2009.
23. Juremi J, Mahmod R, Sulaiman S. A proposal for improving AES S-box with rotation and key-dependent. IEEE International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec); 2012. p. 38-42.
24. Elkamchouchi H, Makar M. Kamkar symmetric block cipher. NRSC Proceedings of the 21st National Radio Science Conference; 2004. p. 1-9.
25. Krishnamurthy N, Ramaswamy V. Performance analysis of Blowfish and its modified version using encryption quality, key sensitivity, histogram and correlation coefficientanalysis. International Journal of Recent Trends in Engineering. 2009; 1(2):1-4.
26. Mohammad F, Rohiem A, Elbayoumy A. A novel S-box of AES algorithm using variable mapping technique. Aerospace Sciences and Aviation Technology. 2009:1-9.
27. Ritter T. Transposition cipher with pseudo-random shuffling: the dynamic transposition combiner. Cryptologia. 1991:37–41.
28. Stoianov N. One approach of using key-dependent S-boxes in AES. Multimedia Communications, Services and Security. Berlin, Heidelberg: Springer; 2011. p. 317-23.
29. Hosseinkhani R, Javadi H. Using cipher key to generate dynamic S-box in AES cipher system. International Journal of Computer Science and Security. 2012; 6(19):19-28.
30. Mahmoud E, Hafez A, Elgarf T, Zekry A. Dynamic AES-128 with key-dependent S-box. International Journal of Engineering Research and Applications. 2013; 3(1):1662–70.
31. Suri P, Deora S. A cipher based on 3D array block rotation. IJCSNS. 2010; 10(2):186-91.
32. Kalnins L. Coordinate Systems. 2009:1–5.
33. Collins G. The foundations of celestial mechanics. Tucson, AZ: Pachart Publishing House (Pachart Astronomy and Astrophysics Series); 1989.
34. Brougham H. Coordinate Systems and Transformation. 124–130.
35. MIT. Review B: Coordinate Systems. Department of Physics, Massachusetts Institute of Technology; 2005.
36. Soto J. Randomness testing of the AES candidate algorithms. NIST. Citeseer. 1999:1-10. Available from: csrc. nist. gov.
37. Soto J, Bassham L. Randomness testing of the advanced encryption standard finalist candidates. DTIC Document. 2000:1-14.
38. Isa H, Z'aba M. Randomness analysis on LED block ciphers. ACM Proceedings of the 5th International Conference on Security of Information and Networks; 2012. p. 60-6.
39. Mar P, Latt K. New analysis methods on strict avalanche criterion of S-boxes. World Academy of Science, Engineering and Technology. 2008; 48:150-4.
40. Rukhin A et al. A statistical test suite for random and pseudorandom number generators for cryptographic applications. National Institute of Standards and Technology Special Publication; 2010. Report number: 800-22.
41. Menezes A, Van Oorschot P, Vanstone S. Handbook of Applied Cryptography. CRC Press; 1996.