4-bit Boolean functions in generation and cryptanalysis of secure 4-bit crypto S-boxes.

Sankhanil Dey[1], Amlan Chakrabarti[2] and Ranjan Ghosh[3],
Department of Radio Physics and Electronics[1,3], University of Calcutta,
92 A P C Road, Kolkata-700009,
A K Choudhury School of Information Technology[2], University of Calcutta,
Sector III, JD-2 block, Salt-lake City, Kolkata-700098,
sdrpe_rs@caluniv.ac.in[1], achakra12@yahoo.com[2], rghosh47@yahoo.co.in[3].

Abstract. In modern ciphers of commercial computer cryptography 4-bit crypto substitution boxes or 4-bit crypto S-boxes are of utmost importance since the late sixties. Since then the 4 bit Boolean functions (BFs) are proved to be the best tool to generate the said 4-bit crypto S-boxes. In this paper the crypto related properties of the 4-bit BFs such as the algebraic normal form (ANF) of the 4-bit BFs, the balancedness, the linearity, the nonlinearity, the affinity and the non-affinity of the 4-bit BFs and the strict avalanche criterion (SAC) of 4-bit BFs are studied in detail. An exhaustive study of 4-bit BFs with some new observations and algorithms on SAC of 4-bit BFs is also reported in this paper. A bit later in the end of nineties the Galois field polynomials over Galois field GF(2^8) are in use to generate the 8-bit crypto S-box of the Advance Encryption Standard (AES). A detailed study on generation of the 4-bit crypto S-boxes with such Galois field polynomials over the binary as well as non-binary extended Galois fields is also given in this paper. The generated 4-bit crypto S-boxes are analyzed with four cryptanalysis techniques and the well-defined SAC algorithms of 4-bit crypto S-boxes to search for the best possible 4-bit crypto S-boxes. Some existing 4-bit crypto S-boxes like the 32 4-bit crypto S-boxes of the Data Encryption Standard (DES) and the four 4-bit crypto S-boxes of the two variants of the Lucifer are analyzed to report the weakness of such S-boxes. A comparative study of the ancient as well as the modern 4-bit crypto S-boxes with the generated 4-bit crypto S-boxes proves the said generated 4-bit crypto S-boxes to be the best possible one.

1. Introduction and Scope. The four bit Boolean functions (4-bit BFs) contain 16 bits with bit values 0 or 1 [1]. The 16 bit long 4-bit BF can be considered as a 16 bit binary number with position 0 as MSB and position f as LSB and the decimal equivalent of the binary number is considered as the decimal equivalent (DE) of the 4-bit BF [2]. The positions of each bit within the 4-bit BFs vary from 0 to f in hex. The bits in each position from 1 to 4 of the 4-bit binary equivalents of the 16 positions increases sequentially from 0 to f in hex constitute four 4-bit input bit vectors (IPVs) with decimal equivalents of four 16 bit long IPVs are 255, 3855, 13107 and 21845 respectively [3]. The 4-bit BFs with balanced number of bits with same bit values i.e. 8 bits with bit value 0 and 8 bits with bit value 1 are termed as balanced 4-bit BFs [4]. This property is reviewed or described in section 2. The general equation to derive each bit of a 4-bit BF can be termed as the Algebraic Normal Form (ANF) of the 4-bit BF. Coefficients of the 16 terms are termed as ANF coefficients of the 4-bit BFs. The terms of the ANF equation contain either a constant or one variable or product of two, three or four variables. They are called as constant term, linear term and product or nonlinear terms of the ANF equation respectively. The 4-bit BFs with ANF equations contain only constant term with coefficient 0 and only linear terms are called as linear BFs. The 4-bit BFs with ANF equations contain constant term with coefficient 0 and at least one product or nonlinear terms are called as non-linear 4-bit BFs. The 4-bit BFs with constant term with coefficient 1 and only linear terms are called as Affine BFs. The 4-bit BFs with ANF equations contain constant term with coefficient 1 and at least one product or nonlinear terms are called as Non-Affine 4-bit BFs [5-6]. The ANF, Linearity, Nonlinearity, Affinity and Non-Affinity are reviewed in sec.2. If four IPVs of the 4-bit BFs are complemented one at a time and the distance between the 4-bit BFs before and after complement operation is a balanced 4-bit BF then the 4-bit BFs are said to satisfy the strict avalanche criterion (SAC) of 4-bit BFs [7-8]. The property is illustrated in section 2 in this paper. The old algorithms of SAC of 4-bit BFs with new observations and algorithms are described in section 3.

The 4-bit crypto S-box contains 16 unique and distinct elements vary from 0 to f in hex. The positions of each bit within the 4-bit BFs vary from 0 to f in hex. The bits in each position from 1 to 4 of the 4-bit binary equivalents of the 16 positions of the S-box elements increases sequentially from 0 to f in hex constitute four 4-bit input bit vectors (IPVs) with decimal equivalents of four 16 bit long IPVs are 255, 3855, 13107 and 21845 respectively [9]. The bits in each position from 1 to 4 of the 4-bit binary equivalents of the 16 elements of the S-box constitute four 4-bit output bit vectors (OBVs) [9]. 4 IPVs and 4 OBVs of a crypto S-box are 8 distinct and unique and balanced 4-bit BFs [9]. If four 4-bit BFs of a crypto S-box satisfy SAC of the 4-bit BFs together then the crypto S-box is said to satisfy SAC of 4-bit crypto S-boxes [10]. The said criterion with new observations and algorithms is described in section.3. The generation of a 4-bit crypto S-box with four 4-bit BFs is shown in section.4. If the resultant ${}^4C_2$ (= 6) 4-bit BFs of the bitwise xor operation between all possible combination of the two 4-bit BFs of a crypto S-box are balanced then the crypto S-box is said to satisfy the (output) bit independence criterion or BIC of the 4-bit S-boxes. The BIC criterion for the 4-bit crypto S-boxes is described in section 4.

The generated 4-bit S-boxes are analyzed with cryptanalysis techniques of 4-bit crypto S-boxes such as linear cryptanalysis of 4-bit crypto S-boxes [11], linear approximation analysis [12], differential cryptanalysis of 4-bit crypto S-boxes [13] and differential cryptanalysis of 4-bit crypto S-boxes with 4-bit BFs [14] and SAC algorithms of 4-bit S-boxes [15]. The results are then compared with the said analysis on the existing 32 and four 4-bit crypto S-boxes of Data Encryption Standard (DES) and Lucifer respectively to show the weakness of the existing crypto S-boxes and to prove the generated S-boxes are the best possible ones. The detail discussion is included in section.5. The conclusion and the acknowledgement are given in section 6 and 7 respectively.

2. 4-bit BFs: Its Features and Properties.

The BFs are usually represented by input-output binary bits in a Truth Table. Its features are expressed with following three formalisms which are explained in detail in Sec.2.1. The general equation to generate 16 linear, 16 affine, 32752 nonlinear and 32752 non-affine 4 bit equations is called as Algebraic Normal Form (ANF) of 4-bit BFs. It is reviewed in Sec. 2.2. Of the 65536 4-bit BFs, 32 are linear and 65504 are nonlinear – the linear BFs and the nonlinear BFs are well reviewed in Sec.2.3. Again of the 65536 4-bit BFs, 12870 (= ${}^{12}C_8$) are balanced and the rest 52666 are unbalanced – this is detailed in Sec.2.4. A 4-bit BF is said to satisfy Strict Avalanche Criterion (SAC) if, on flipping all bits of one of the four 16-bit input vectors, 50% of its output bits gets flipped and the changed 16-bit output vector may be balanced or unbalanced. This property of BFs can also be named as the First Order (FO)-SAC which is explained in detail in Sec.2.5. On successively flipping two of the four input bit vectors, if a particular BF successively satisfies two respective FO-SACs then the BF is said to satisfy Two Successive First Order (SFO) SACs. In the event three or four input bit vectors are successively flipped and it is observed that if a particular BF successively satisfies three or four FO-SACs, the concerned BF is said to satisfy three or four SFO-SACs, The Strict Avalanche Criterion (SAC) of 4-bit BFs is reviewed from SFO-SAC angles in Sec.2.6. Besides SFO-SACs, one can also consider another type of SAC, namely Higher Order (HO) SAC. If two or more Input Vectors (IPVs) are simultaneously flipped, the bits in the BF before and after flip is changed in 8 positions and in rest 8 positions remains the same then the BF is said to satisfy Higher Order (HO) SACs – for two IPVs, it is said as Second Order HO-SAC, for three, Third Order HO-SAC and for four, fourth Order HO-SAC. The Strict Avalanche Criterion (SAC) of 4-bit BFs is also reviewed from Higher Order (HO) SAC angles in Sec.2.7.

2.1. Features: Bit level, Bit Vector level and Galois Field Level presentations.

The Truth Table of a 4-bit BF is presented in Table 3.1 in such a fashion that it is possible to view it from three angles, (a) Bit Level, (b) Vector Level and (c) Galois Field Level. The first column with sub-columns 1 to 3 is the Bit Level presentation of the Truth Table, while the Vector Level and the Galois Field Level presentations are made together in the second column within sub-columns 4 to 9.

(a) Bit Level presentation

The 16 rows of 3 columns (1 to 3) of Table 2.1 represent the Bit Level Truth Table of a 4-bit BF [13][14]. The 16 rows of col.1 indicate sequentially and monotonically increasing 4 input bits whose left-most bit is the MSB and right-most bit is the LSB. Considering the LSB-MSB issue, the Decimal Equivalent (DE) of the 16 set of 4 bits input is given in the respective row of col.2. The 1-bit output corresponding to 4-bit input is also put in the respective row of col.3. The functional relation of the bit level presentation of a 4-bit BF between a single output bit (y) and four input bits (x_1,x_2,x_3,x_4) can be expressed as,

$$y = \text{BL-BF}(x_1,x_2,x_3,x_4) \quad \ldots \quad \ldots \quad (2.1.a)$$

(b) Vector Level Presentation

The 17 rows of 5 columns (4 – 7 & 9) of Table 2.1 represent the Vector Level Truth Table of a 4-bit BF. The columns 4 to 7 are the four 16-bit Input Vectors $\{x_1, x_2, x_3, x_4\}$ of the same input shown in 16 rows of col.1 of the Bit Level presentation and a 16-bit Output Vector {y} of the same output shown in 16 rows of col.3 is shown in col.9. Of the 16 set of 4 input bits (x_1,x_2,x_3,x_4), input vector $\{x_1\}$ is formed by 16 (x_1) bits, $\{x_2\}$ by 16 (x_2) bits, $\{x_3\}$ by 16 (x_3) bits, $\{x_4\}$ by 16 (x_4) bits and output vector {y} by using 16 (y) bits. The decimal equivalents of the four input vectors, $\{x_1, x_2, x_3, x_4\}$ and that of the output vector, {y} are given in respective column of the 17^{th} row. While computing the decimal equivalents of the 16-bit four input vectors, the respective bit in row 1 of Table 3.1 is considered as the MSB and the respective bit in row 16, the LSB. The input-output functional relation for vector level presentation of Truth Table of a 4-bit BF is expressed between four input x-vectors $\{x_1,x_2,x_3,x_4\}$ and output {y}-vector as follows,

$$\{y\} = \text{VL-BF}\{x_1, x_2, x_3, x_4\} \quad \ldots \quad \ldots \quad (2.1.b_1)$$

It can also be expressed between $\{y\}_{DE}$ and DEs of four IVs $\{x_1,x_2,x_3,x_4\}$ as follows,

$$\{y\}_{DE} = \text{VL-BF}(255, 3855, 13107, 21845) \quad \ldots \quad \ldots \quad (2.1.b_2)$$

(c) **Presentation using Galois Field Polynomials**

The 16 rows of columns 8 and 9 of Table 2.1 represent the Galois Field Level Truth Table of a 4-bit BF. The col.8 is the 16-character Input Vector in Hex {h} of the same input shown in 16 rows of col.1 of the Bit Level presentation and the col.9 is the 16-bit Output Bit Vector {y}. The decimal Equivalents (DEs) of Galois Field Polynomial of {y} over Finite Field 2^{15} is designated as y, while the decimal Equivalents (DEs) of Galois Field Polynomial of {h} over Finite Field 16^{15} is designated as h, as given below.

$$\{h\}_{DE} = 0z^{15}+1z^{14}+2z^{13}+3z^{12}+4z^{11}+5z^{10}+6z^{9}+7z^{8}+8z^{7}+9z^{6}+az^{5}+bz^{4}+cz^{3}+dz^{2}+ez+fz^{0}, (z=16)$$

$$\{y\}_{DE} = 0z^{15}+1z^{14}+1z^{13}+0z^{12}+1z^{11}+0z^{10}+1z^{9}+1z^{8}+1z^{7}+0z^{6}+0z^{5}+1z^{4}+1z^{3}+0z^{2}+0z+0z^{0}, (z=2)$$

It may be noted that the decimal equivalent of {h} turns out to be **81985529216486895**.
The input-output functional relation between $\{y\}_{DE}$ and $\{h\}_{DE}$ can be expressed as,

$$\{y\}_{DE} = \text{GFL-BF}(81985529216486895) \quad \ldots \quad \ldots \quad (2.1.c)$$

2.2 The Algebraic Normal Form (ANF) of a 4-bit BF

The 4-bit BF is a mapping from $(0,1)^4$ to $(0,1)^1$ which means 4-bit binary input given to a digital system provides 1-bit output. The 4 input bits to a 4-bit Boolean Function (F) are algebraically designated as ($x_1x_2x_3x_4$). Following the Bit level presentation of its Truth Table shown in columns 1 to 3 of Table 3.1, the 16 set of inputs are shown in col.1, the corresponding decimal values between 0 and 15 are respectively shown in col.2 and each set of input providing 1-bit output is shown in col.3 expressed by y. Its functional relation, $y=F(x)=F(x_1x_2x_3x_4)$ can be expressed in Algebraic Normal Form (ANF) with 16 coefficients as given in eq. (2.2) below,

$$\begin{aligned} y &= F(x_1x_2x_3x_4) \\ &= a_0 + (a_1.x_1+a_2.x_2+a_3.x_3+a_4.x_4) + (a_5.x_1.x_2+a_6.x_1.x_3+a_7.x_1.x_4+a_8.x_2.x_3+a_9.x_2.x_4+a_{10}.x_3.x_4) + \\ &\quad + (a_{11}.x_1.x_2.x_3+a_{12}.x_1.x_2.x_4+a_{13}.x_1.x_3.x_4+a_{14}.x_2.x_3.x_4) + a_{15}.x_1.x_2.x_3.x_4 \quad \ldots \quad \ldots \quad (2.2) \end{aligned}$$

where y assumes 1-bit output, x represents the decimal or hex value of 4 input bits represented by $\{x_1x_2x_3x_4\}$, The two operators, '**.**' and '+' represent AND and XOR operations respectively. Here a_0 is a constant coefficient, (a_1 to a_4) are 4 linear coefficients, and (a_5 to a_{15}) are 11 nonlinear coefficients of which (a_5 to a_{10}) are 6 nonlinear coefficients associated with 6 terms having AND-operated-2-input-bits, (a_{11} to a_{14}) are 4 nonlinear coefficients associated with 4 terms having AND-operated-3-input-bits and a_{15} is a non-linear coefficient associated with one term having AND-operated-4-input-bits. The 16 binary ANF coefficients, from a_0 to a_{15} are marked respectively as anf.bit0 to anf.bit15 in ANF representation and are evaluated from the 16-bit output vector of a BF designated as bf.bit0 to bf.bit15 using the following relations as given in eq.(3.3),

anf.bit0 = bf.bit0;
anf.bit1 = anf.bit0 + bf.bit8;
anf.bit2 = anf.bit0 + bf.bit4;
anf.bit3 = anf.bit0 + bf.bit2;
anf.bit4 = anf.bit0 + bf.bit1;
anf.bit5 = anf.bit0 + anf.bit1 + anf.bit2 + bf.bit12;
anf.bit6 = anf.bit0 + anf.bit1 + anf.bit3 + bf.bit10;
anf.bit7 = anf.bit0 + anf.bit1 + anf.bit4 + bf.bit9;
anf.bit8 = anf.bit0 + anf.bit2 + anf.bit3 + bf.bit6;
anf.bit9 = anf.bit0 + anf.bit2 + anf.bit4 + bf.bit5;
anf.bit10 = anf.bit0 + anf.bit3 + anf.bit4 + bf.bit3;
anf.bit11 = anf.bit0 + anf.bit1 + anf.bit2 + anf.bit3 + anf.bit5 + anf.bit6 + anf.bit8 + bf.bit14;
anf.bit12 = anf.bit0 + anf.bit1 + anf.bit2 + anf.bit4 + anf.bit5 + anf.bit7 + anf.bit9 + bf.bit13;
anf.bit13 = anf.bit0 + anf.bit1 + anf.bit3 + anf.bit4 + anf.bit6 + anf.bit7 + anf.bit10 + bf.bit11;
anf.bit14 = anf.bit0 + anf.bit2 + anf.bit3 + anf.bit4 + anf.bit8 + anf.bit9 + anf.bit10 + bf.bit7;
anf.bit15 = anf.bit0 + anf.bit1 + anf.bit2 + anf.bit3 + anf.bit4 + anf.bit5 + anf.bit6 + anf.bit7
+ anf.bit8 + anf.bit9 + anf.bit10 + anf.bit11 + anf.bit12 + anf.bit13 + anf.bit14 + bf.bit15 … (2.3)

The DE (Decimal Equivalent) of the output vector {y} of BFs varies from 0 through 65535 and each decimal value is converted to a 16-bit binary output of the Boolean function from bf.bit0 through bf.bit15. Based on the binary outputs of a BF, the ANF coefficients from anf.bit0 through anf.bit15 are calculated sequentially for all BFs using eq. (3.3).

2.3 Linear-Nonlinear and Affine-Non-affine groups of 4-bit BFs

All the 65536 16-bit Output Vectors of 4-bit BFs can be divided in two equal groups each having 32768 BFs, one is the linear-nonlinear group having binary bit '0' as MSB and the other one is the affine-non-affine group having binary bit '1' as MSB. The decimal equivalent of output vectors of the linear-nonlinear group monotonically increases from 0 to 32767. The 16 of it are linear, while the other 32752 ones are nonlinear. The affine-non-affine group has also 32768 BFs, each of its decimal equivalents monotonically increases from 32768 to 65535 and becomes decimal-wise complement of a concerned BF belonging to the linear-nonlinear group whose entire 16-bit BF is binary complement to the affine-non-affine BF. The 16 of it are Affine which are bit-wise as well as decimal-wise complementary to respective linear ones. The same is true for other 32752 are non-affine BFs in relation to nonlinear ones also. The features and properties of linear-affine and nonlinear-non-affine BFs are discussed in detail in Sec.2.3.1 and Sec.2.3.2 respectively.

2.3.1. Linear and Affine 4-bit BFs

The four 16-bit Input Vectors $\{x_1x_2x_3x_4\}$ shown in Table 3.1 are fixed for all BFs and they are members of the family of 65536 BFs. If one of the four input vectors becomes an output BF, the coefficient a_0 of ANF of the BF becomes zero for all the four cases and the concerned ANF of the BFs assumes forms as follows,

Case (i) : if output BF = $\{x_1\}$, then $\{y\} = \{x_1\}$, means $a_1 = 1$ and all other coefficients are zero.
Case (ii) : if output BF = $\{x_2\}$, then $\{y\} = \{x_2\}$, means $a_2 = 1$ and all other coefficients are zero.
Case (iii) : if output BF = $\{x_3\}$, then $\{y\} = \{x_3\}$, means $a_3 = 1$ and all other coefficients are zero.
Case (iv) : if output BF = $\{x_4\}$, then $\{y\} = \{x_4\}$, means $a_4 = 1$ and all other coefficients are zero.

For all the above four cases, the nonlinear coefficients a_5 to a_{15} are zero indicating the four input vectors $\{x_1\}$, $\{x_2\}$, $\{x_3\}$ and $\{x_4\}$ are the four Basic Linear BFs. In Table 3.2 such four Basic Linear BFs and their corresponding ANFs are mentioned in columns 2 and 7 respectively with C=0 along the "relation number" rows of 1, 2, 5 and 9 respectively. It may now be mentioned that 'xor' operation being a linear operation, its successive applications involving two, three or four basic linear BFs are expected to provide linear BFs. It may be noted that xor operation involving two of the four basic linear BFs gives rise to six linear BFs shown in 6 rows of relation number 4, 6, 7, 10, 11 and 13. Four more linear BFs shown in rows of relation number 8, 12, 14 and 15 get evolved following successive two xor operations involving three of the four basic linear BFs. Successive three xor operations involving all the four basic linear BFs provide one linear BF shown in row of relation number 16. All these can be seen in Table 3.2. The number of linear BFs evolved out of the four basic linear BFs turns out to be 11. There are one constant linear BFs having 16 '0's as its output (vide row of relation number = 1. Altogether there are 16 linear BFs. The ANF coefficients mentioned in column 7 of Table 2 indicate that all the 11 nonlinear coefficients of all these linear BFs are zero. The 16 affine BFs shown in column 5 are obtained by bit-wise complementing all the 16 bits of the respective linear BFs. It may also be noted that considering decimal equivalents the linear and the corresponding affine BFs shown in the same row of Table 2.2 are complementary to each other. The discussed matter is elaborated as follows,

- **Four 16-bit fixed input vectors of 4-bit BFs from its sixteen set of four input bits:** The input pattern of 4-bit BFs ($x_1x_2x_3x_4$) has 16 sequential values from '0' to 'f' in hex corresponding to binary values from {0000} to {1111}. These are always considered fixed at the inputs of all BFs. Each column vector of four {x_1, x_2, x_3, x_4} vectors is a 16-bit Input Vector (IPV) shown in column 1 of Table 3.2 under its headings 'x1', 'x2', 'x3'or 'x4' respectively and is designated as IV1, IV2, IV3 or IV4 respectively and is termed as four fixed 16-bit Input Vectors (IPVs) of all 4-bit BFs.

- **Constant Linear BF:** There is one constant linear BF shown in column 2 of relation no. 1 of Table 3.2 with C=0. Corresponding to each of 16 set of 4-bit inputs shown in column 1, the 16 output bits of the BF with C=0 are also '0' as shown in column 3 of relation no.1. Its 16 ANF coefficients also turn out to be zero as shown in column 7 of the same relation no.

- **Four Basic Linear BFs:** If one of {x_1, x_2, x_3, x_4} input vectors depicted under the column heading '[x1]', '[x2] ', '[x3] 'or '[x4]' of Table 3.2 becomes the 16-bit output of a BF, the four BFs can be defined as $F_1(x) = C+x_1$, $F_2(x) = C+x_2$, $F_3(x) = C+x_3$ or $F_4(x) = C+x_4$, all with C = 0, as shown in relation no., 3, 5 or 9 of column 2 and their respective output is given in the corresponding relation no. of column 3. Following the ANF formalism of a BF given in eq. (3.2) of Sec.3.2.2, one can conclude that for each of the four BFs defined above, only one of the 4 linear ANF coefficients has a binary value 1 and all other coefficients are zero, indicating $a_1 = 1$ for $F_1(x)$, $a_2 = 1$ for $F_2(x)$, $a_3 = 1$ for $F_3(x)$ and $a_4 = 1$ for $F_4(x)$ as shown in relation no. 2, 3, 5 and 9 of column 7 respectively. This indicates that $F_1(x) = x_1$, $F_2(x) = x_2$, $F_3(x) = x_3$ and $F_4(x) = x_4$ are the four Basic Linear BFs (BLBFs). The same ANF coefficients are also obtained if these are computed using respective BF outputs given in relation no. 2, 3, 5 or 9 of column 3. All other Linear BFs are obtained using the four Basic Linear BFs. It may be noted that the 16 linear BFs and 16 affine BFs including the constant ones are so organized in Table 3.2 that the Decimal Equivalent of Boolean Function (DEBF) of all of them appear in an ascending order for linear BFs and that of the affine BFs, in a descending order as a decimal-wise complement of 65535 of the respective linear BFs.

- **Eleven Other Linear BFs derived from the four Basic Linear BFs:** It may be noted that the XOR operation, being a linear operator, would give rise to other eleven linear BFs if XOR operations of four Basic Linear BFs are properly undertaken. The XOR operation of any two of the four Basic Linear BFs gives rise to six linear BFs which are shown in relation no. 4, 6, 7, 10, 11 and 13; the XOR operation of any three of the four Basic Linear BFs gives rise to four linear BFs as shown in relation no. 8, 12, 14 and 15 and the last one is the XOR operation of the four Basic Linear BFs shown in relation no. 16. It may be noted that their non-zero linear ANF coefficients correspond to those which are related to the Basic Linear BFs involved in the XOR operations. The non-linear coefficients are obviously zero. The same observation would also be made if the ANF coefficients are computed using eq. (3.2) of Sec. 3.2.2 based on respective BF outputs.

- **Sixteen Affine BFs:** The affine BFs are obtained by complementing all output bits of the linear BFs. The sixteen affine BFs are their corresponding linear BFs depicted in column 2 of relation no. 1 through 16 with C=1. Each of their 16-bit output is obtained by complementing the corresponding 16 linear BFs and is shown in column 5 of 16 relation nos. The 16 ANF coefficients of each of the affine BFs are identical to the corresponding linear BFs except the one under the heading ‘0’ of column 7 of 16 relation no.s which assumes the binary value of C for affine BFs it is always ‘1’.

2.3.2 Non Linear and Non Affine 4-bit BFs:

The 4-bit BFs with constant term C = ‘0’ and with at least one ‘1’ present in the subheadings 5 through f in the column 7 of table 3.2 with or without 1s in the subheadings 1 through 4 of the table 3.2. So the nonlinearity is judged on the basis of the presence of nonlinear ANF coefficients or ANF product terms in the concerned 16 bit ANF coefficient vector or the concerned ANF equation derived from equation 3.2. Here in equation 3.2 a_5 to a_{15} are 11 nonlinear coefficients of which (a_5 to a_{10}) are 6 nonlinear coefficients associated with 6 terms having AND-operated-2-input-bits. If at most these nonlinear terms are present in the concerned nonlinear ANF equation then the algebraic nonlinearity is counted to be 2^{nd} order algebraic nonlinearity. If at most a_{11} to a_{14} or 4 terms having AND-operated-3-input-bits are present then the 3^{rd} order algebraic nonlinearity and if at most a_{15} or one term having AND-operated-4-input-bits is present then the then the 4^{th} order algebraic nonlinearity is observed. The same is for non-affine BFs with C = ‘1’. The only difference of 2^{nd} , 3^{rd} and 4^{th} order algebraic non-affinity [15] is the presence of the constant term or C =’1’. The number of nonlinear and non-affine 4-bit BFs satisfied some of the categories of algebraic nonlinearity are presented in table. 2.3.

The maximum nonlinearity or in colloquial term the nonlinearity (NL) of a 4-bit nonlinear or non-affine BF is the number of 1s in the 16 bit hamming distance vector (HDV) with the minimum number of 1s among 32 HDVs generated from the bitwise xor operation of the 4-bit nonlinear or non-affine BF to the 16 linear 4-bit BFs and 16 affine 4-bit BFs [16]. For 4-bit BFs the maximum value of maximum nonlinearity is 6 and the minimum value of maximum nonlinearity is 1 [17]. The numbers of nonlinear and non-affine 4-bit BFs with NL equal to 1 through 6 are noted in table 2.3. The tables are discussed below,

2.4 Balanced and Unbalanced 4-bit BFs.

A 4-bit BF contains 16 bits. If there are balanced number of 0s and 1s i.e. there are 8 1s and 8 0s are present in the binary equivalent of the 4-bit BF then it is called as Balanced 4-bit BF. Other 4-bit BFs are called as Unbalanced 4-bit BFs. There are ${}^{16}C_8$ or 12870 balanced 4-bit BFs exist and rest of the 4-bit BFs out of 65536 is unbalanced. The maximum nonlinearity of the balanced BFs are either four or 2 [18]. The balancedness for the same balanced and unbalanced 4-bit BFs are noted in table.2.3.

2.5 First Order Strict Avalanche Criterion (FO-SAC) of the 4-bit BFs.

A 4-bit BF is said to satisfy Strict Avalanche Criterion (SAC) if, on flipping all bits of one of the four 16-bit input vectors, 50% of its output bits gets flipped and the changed 16-bit output vector may be balanced or unbalanced. This property of BFs can also be named as the First Order (FO)-SAC. The FO-SAC of some balanced and unbalanced 4-bit BFs are noted in table.2.3[19].

2.6 Successive First Order Strict Avalanche Criterion (SFO-SAC) of the 4-bit BFs [20].

On successively flipping two of the four input bit vectors, if a particular BF successively satisfies two respective FO-SACs then the BF is said to satisfy Two Successive First Order (SFO) SACs. In the event three or four input bit vectors are successively flipped and it is observed that if a particular BF successively satisfies three or four FO-SACs, the concerned BF is said to satisfy three or four SFO-SACs, The SFO-SAC for the same balanced and unbalanced 4-bit BFs are noted in table.2.3.

2.7 **Multiple Higher Order Strict Avalanche Criterion (MHO-SAC) of the 4-bit BFs.**

Besides SFO-SACs, one can also consider another type of SAC, namely Higher Order (HO) SAC. If two or more Input Vectors (IPVs) are simultaneously flipped, the bits in the BF before and after flip is changed in 8 positions and in rest 8 positions remains the same then the BF is said to satisfy Higher Order (HO) SACs – for two IPVs, it is said as Second Order HO-SAC, for three, Third Order HO-SAC and for four, fourth Order HO-SAC. The MHO-SAC for the same balanced and unbalanced 4-bit BFs are noted in table.2.3.

3 Strict Avalanche Criterion for 4-bit BFs and 4-bit S-boxes: The Strict Avalanche Criterion is introduced by Webster and Tavares [1] in late eighties of the previous century. If four IPVs of a 4-bit OPBF are complemented one at a time and the hamming distance between the said OPBF and complemented OPBFs are 8 or the difference BFs are balanced then the 4-bit OPBF is said to satisfy the FO-SAC of the 4-bit BFs. If four OPBFs of an S-box satisfy the FO-SAC of the 4-bit BFs individually then the S-box is said to satisfy the FO-SAC of the 4-bit S-boxes. Now If four IPVs of a 4-bit OPBF are complemented 2 or 3 together at a time and the hamming distance between the said OPBF and complemented OPBFs are 8 or the difference BFs are balanced then the 4-bit OPBF is said to satisfy the HO-SAC of the 4-bit BFs and if four IPVs of a 4-bit OPBF are complemented four together at a time and the hamming distance between the said OPBF and complemented OPBFs are 8 or the difference BFs are balanced then the 4-bit OPBF is said to satisfy the Extended HO-SAC of the 4-bit BFs [24]. FO-SAC, HO-SAC and Extended HO-SAC of the 4-bit BFs together is called as MHO-SAC of the 4-bit BFs. If four OPBFs of an S-box satisfy the MHO-SAC of the 4-bit BFs individually the S-box is said to satisfy the MHO-SAC of the 4-bit S-boxes. The procedure of the DC is very similar to MHO-SAC of the 4-bit BFs. The S-box is properly defined in section 3.1. In this chapter the concept of the SAC of the 4-bit BF is reviewed in section 3.2 with two new algorithms. The section 3.3 is devoted to establish the analogy of the MHO-SAC of the 4-bit S-boxes with the DC.

3.1 S-box:

A 4-bit S-box can be written as follows in table 3.1.1, where the each element of the first row of table 3.1.1, entitled as index, are the position of each element of the S-box within the given S-box and the elements of the 2nd row, entitled as S-box, are the elements of the given S-box. It can be concluded that the 1st row is fixed for all possible S-boxes. The values of each element of the 1st row are distinct, unique and vary between 0 to F in hex. The values of the each element of the 2nd row of the S-box are also distinct and unique and also vary between 0 to F in hex. The values of the elements of the fixed 1st row are sequential and monotonically increase where for the 2nd row they can be sequential or partly sequential or non-sequential. Here the given substitution box is the 1st S-box of the 1st given S-box out of 8 of the Data Encryption Standard [11].

Table 3.1.1: S-box.

Row	**Column**	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G
1	**Index**	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	**S-Box**	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

3.2. Strict Avalanche Criterion (SAC) of the 4-bit BFs and 4-bit S-boxes [12][13]:

The SAC of the 4-bit BFs with pseudo code is reviewed in section 3.2.1 and a new technique entitled “Shift Method” to find SAC of 4-bit BFs with pseudo code is noted in brief in section 3.2.2. Another new technique “Flip Method” of the SAC of 4-bit BFs and the SAC of 4-bit S-boxes with pseudo code is also reviewed in section 3.2.3

3.2.1. A brief review on SAC of 4-bit BFs: A 4-bit BF is said to satisfy the SAC of the 4-bit BFs if the distant BFs or bitwise hamming distances are balanced that are generated due to the bitwise xor operations of the OPBF with the complemented OPBFs (COPBFs) that are also generated due to the complementation of the four IPVs individually. In the SAC of the 4-bit BFs IPV4, IPV3, IPV2 and IPV1 respectively that are shown in column 1 through G of the row 2, 3, 4 and 5 in table 3.2.1.1 are complemented individually one at a time. If due to the said operation on OPBF the numbers of bits changed in COPBFs are 8 or half of the number of bits in a 4-bit BF then the OPBF is said to satisfy SAC of 4-bit BFs.

IPV4, CIPV4, IPV3, CIPV3, IPV2, CIPV2, IPV1, CIPV1 are shown in column 2 thorough H of row 1, 3, 7, 9 , D, F, J, L respectively of the table 3.2.1.2. The OPBFs and the COPBFs and CIPV4, CIPV3, CIPV2 and CIPV1 are noted in column 2 thorough H of the row 2, 4, 8, A, E, G and K, M respectively. The difference BFs or DBFs more specifically, DBF4, DBF3, DBF2, DBF1 are shown in column 2 thorough H of row 5, B, H, N respectively.

Now the changes in numbers of bits in COPBFs from OPBF are 12, 8, 4, 12 respectively. So the given OPBF does not satisfy the SAC of the 4-bit BFs. To satisfy SAC of the 4-bit BFs changes in numbers of bits in four COPBFs from OPBFs must be 8, 8, 8, 8.

Note: If four OPBFs of a particular S-box satisfy SAC of 4-bit BFs individually then the said S-box is said to satisfy the SAC of the 4-bit S-boxes.

Pseudo Code:

```
Let BF[16].bit0 is a bit level array of 16 bits of a 4-bit BF and BF[16] is an
array of 16 bits of a 4-bit BF. CV[16].bit0 is a bit level array of 16 bits to
store either 00FF, 0F0F, 3333, 5555 in hex. CVC[16].bit0 is a bit level array of 16
bits to store either FF00, F0F0, CCCC, AAAA in hex. Here ^ represents bitwise Xor
operation. NL represents Numbers of bits changed in lower halves and NU represents
numbers of bits changed in upper halves.

Start.

Step 0A: For 1:16 BF[16].bit0 = BF[16].
Step 0B: For 1:16 CV[16].bit0 = 00FF, 0F0F, 3333, 5555.
Step 0C: For 1:16 CVC[16].bit0 = FF00, F0F0, CCCC, AAAA.


// Next five steps demonstrates the algorithm.
Step 01: wt{(BF[16].bit0 & 00FF)^(BF[16].bit0>>8&00FF)}+
WT{(BF[16].bit0&FF00)^(BF[16].bit0>>8&FF00)}= N= NL3 + NU3.
Step 02: wt{(BF[16].bit0 & 0F0F)^(BF[16].bit0>>4&0F0F)}+
WT{(BF[16].bit0&F0F0)^(BF[16].bit0>>4&F0F0)}= N = NL2 +NU2.
Step 03: wt{(BF[16].bit0 & 3333)^(BF[16].bit0>>2&3333)}+
WT{(BF[16].bit0&CCCC)^(BF[16].bit0>>2&CCCC)}=N= NL1 +NU1.
Step 04: wt{(BF[16].bit0 & 5555)^(BF[16].bit0>>1&5555)}+
WT{(BF[16].bit0&AAAA)^(BF[16].bit0>>1&AAAA)}=N= NL0 + NU0.
Step 05: If N=8 for Step 01, Step 02, Step 03, Step 04.
      then BF[16].bit0 Satisfies SAC.
      else BF[16].bit0 Does not Satisfies SAC.
Stop.
```

Note: Time complexity of the algorithm has been O(n).

Note: This algorithm is also called as FO-SAC algorithm. Now if 2, 3 or 4 IPVs are complemented together at a time respectively then the said algorithm is called as MHO-SAC algorithm and the last case of four IPVs together is called as Extended Higher Order SAC of the 4-bit BFs [24].

3.2.2. Shift method for the SAC of the 4-bit BFs: Here in COPBF complement of 4^{th} IPV means interchange of the each distinct 8 bit halves of the 16 bit long 4^{th} IPV so the 2, 8 bit halves of the OPBF is interchanged due to complement of the 4^{th} IPV or the CIPV4. Next to it in COPBF complement of 3^{rd} IPV means interchanges of the each distinct 4 bit halves of the each distinct 8 bit halves of the IPV3 in CIPV3 so the each distinct 4 bit halves of the each distinct 8 bit halves of the OPBF are interchanged due to the complement of the IPV3. Now in COPBF the complement of the 2^{nd} IPV means interchange of the each distinct 2 bit halves of the each distinct 4 bit halves of the each distinct 8 bit halves of the OPBF in COPBF and the complement of 1^{st} IPV means interchange of the each bit of the each distinct 2 bit halves of the 16 bit long OPBF.

IPV4, CIPV4, IPV3, CIPV3, IPV2, CIPV2, IPV1, CIPV1 are shown in column 2 thorough H of row 1, 3, 7, 9 , D, F, J, L respectively of table 3.2.1.2. The OPBFs and COPBFs are noted in column 2 thorough H of row 2, 4, 8, A, E, G and K, M respectively. The difference BFs or DBFs more specifically, DBF4, DBF3, DBF2, DBF1 are shown in column 2 thorough H of row 5, B, H, N respectively. Now changes in numbers of bits in COPBFs from OPBF are

12, 8, 4, 12 for this example. So the given OPBF does not satisfy the SAC of the 4-bit BFs. To satisfy the SAC of the 4-bit BFs changes in numbers of bits in COPBFs from OPBFs must be 8, 8, 8, 8.

Note: If four OPBFs of a particular S-box satisfy SAC of 4-bit BFs individually then the said S-box is said to satisfy the SAC of the 4-bit S-boxes.

Pseudo Code:

```
Start.
// bits of the 16 bit long OPBF are relocated to bit level array BF[16].bit0.
Step 00:  For 1:16 BF[16].bit0 = BF[16].
// OPBF is circularly shifted by 8 bits and complemented BF or COPBF is located to
bit level array CBF[16].bit0.
Step 1A: CBF[16].bit0 = (BF[16].bit0>>8);
// Difference BF is obtained by xor of each bit of OPBF and COPBF.
Step 1B: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0;
// Numbers of 1s in DBF are counted.
Step 1C: Count = IF(DBF[16].bit0==1);
// Each distinct 8 bit halves of OPBF is circularly shifted by 4 bits and
complemented BF or COPBF is located to bit level array CBF[16].bit0.
Step 2A: CBF[16].bit0 = (BF[8A].bit0>>4)&& (BF[8B].bit0>>4);
// Difference BF is obtained by xor of each bit of OPBF and COPBF.
Step 2B: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0;
// Numbers of 1s in DBF are counted.
Step 2C: Count = IF(DBF[16].bit0==1);
// In next step Each distinct 4 bit halves of each distinct 8 bit halves of OPBF is
circularly shifted by 2 bits and complemented BF or COPBF is located to bit level
array CBF[16].bit0.
Step 3A: CBF[16].bit0 = (BF[4A].bit0>>2)&& (BF[4B].bit0>>2)&& (BF[4C].bit0>>2)&&
(BF[4D].bit0>>2);
// Difference BF is obtained by xor of each bit of OPBF and COPBF.
Step 3B: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0;
// Numbers of 1s in DBF are counted.
Step 3C: Count = IF(DBF[16].bit0==1);
// In next step each bit of each distinct 2 bit halves are circularly shifted by 1
bits and complemented BF or COPBF is located to bit level array CBF[16].bit0.
Step 4A: CBF[16].bit0 = (BF[2A].bit0>>1)&&(BF[2B].bit0>>1)
&&(BF[2C].bit0>>1)&&(BF[2D].bit0>>1)&&(BF[2E].bit0>>1)
&&(BF[2F].bit0>>1)&&(BF[2G].bit0>>1)&&(BF[2H].bit0>>1);
// Difference BF is obtained by xor of each bit of OPBF and COPBF.
Step 4B: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0;
// Numbers of 1s in DBF are counted.
Step 4C: Count = IF(DBF[16].bit0==1);
// Test of SAC criterion.

Step 05 : IF Count = 8 for Step 1C, Step 2C, Step 3C, Step 4C. BF[16] Satisfies SAC
of 4-bit BFs.
ELSE BF[16] does not Satisfy SAC of 4-bit BFs.
Stop.
```

Time complexity of the given pseudo code: Time complexity of the algorithm is O(n) since the body contains no nested loops.

Note: This algorithm is also called as FO-SAC algorithm. Now if 2, 3 or 4 IPVs are complemented together at a time respectively then the said algorithm is called as MHO-SAC algorithm and the last case of four IPVs together is called as Extended SAC of the 4-bit BFs.

3.2.3. Flip method of the SAC of the 4-bit BFs and 4-bit S-boxes [12][13]: row 2 through 5 and row 7 through A of each column of column 1 through G in table 3.2.1.1 constitutes 16 4-bit input binary numbers and 16 4-bit output binary numbers respectively. Here for each OPBF each input binary number is flipped in one fixed position and the corresponding bit values of the OPBF before and after are xored to obtain the flipped BF (FB). If four flipped BF for four fixed positions are balanced then the 4-bit OPBF is said to satisfy SAC of the 4-bit BF.

All the elements of the given S-box in hex, index of each element of the given S-box in hex (INH) and 4 bit binary form (INB) are given in column 2 through H of row 3, 1, 2 of the table 3.2.3.1 respectively. OPBFs are shown in column 2 through H of row 4, 5, 6, 7 of the table 4.2.3.1 respectively.

Now 16 INBs before flip and 16 INBs after flip in one bit particularly in fixed bit positions 1, 2, 3, 4 are shown in row 2 through H of column 1, 2, 6, 7, B, C, G, H respectively of table 3.2.3.2. Each corresponding bits of the concerned OPBF duly before and after flip are noted in row 2 through H of column 3, 4, 8, 9, D, E, I, J respectively in the same table. 1 in any position of the flipped BF in row 2 through H of column 5, A, F, K illustrate dissimilarity in bits in the corresponding positions of the concerned OPBF duly before and after flip in one bit in fixed bit positions 1, 2, 3 and 4 respectively.

If out of 16 positions in each row from 2 through H column of column 5, A, F, K there are 8 1s and 8 0s then the given BF is said to satisfy the SAC of the 4-bit BFs. If all the four OPBFs of an S-box satisfy the SAC of the 4-bit BFs then the S-box is said to satisfy the SAC of the 4-bit S-boxes.

Here in table 3.2.3.2. row I shows the numbers of bits changed in OPBF1, OPBF2, OPBF3, OPBF4 before and after flip in pos. 1, pos. 2, pos. 3 and pos. 4 respectively. Since the value is not equal to 8 all positions for the given OPBF so the concerned OPBF and the given S-box does not satisfy the SAC of the 4-bit BFs and the SAC of the 4-bit S-boxes respectively.

Note: This algorithm is also called as Flip FO-SAC algorithm. Now if 2, 3 or 4 bits of the IPVs are flipped together at a time respectively then the said algorithm is called as MHO-SAC algorithm and the last case of the four bit flip together is called as Extended SAC of the 4-bit BFs.

Table 3.2.1.1: IPVs and OPBFs of the 1st S-box of the DES.

Row	Column	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	**H.**
1	**Index**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**	Decimal Equivalent
2	**IPV4**	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	00255
3	**IPV3**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	03855
4	**IPV2**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	13107
5	**IPV1**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	21845
6	**S-box**	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7	
7	**OPBF4**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0	42836
8	**OPBF3**	1	1	1	0	0	1	0	0	0	0	1	1	1	0	0	1	58425
9	**OPBF2**	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	1	36577
A	**OPBF1**	0	0	1	1	0	1	1	0	1	0	0	0	1	1	0	1	13965

Table 3.2.1.2: SAC Criterion for 4-bit BFs.

R|C	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
1	**IPV4**	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
2	**OPBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
3	**CIPV4**	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
4	**COPBF**	0	1	0	1	0	1	0	0	1	0	1	0	0	1	1	1
5	**DBF**	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1
6	Number of bits changed in COPBF												12				
R|C	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
7	**IPV3**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
8	**OPBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0

9	**CIPV3**	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
A	**COPBF**	0	1	1	1	1	0	1	0	0	1	0	0	0	1	0	1
B	**DBF**	1	1	0	1	1	1	0	1	0	0	0	1	0	0	0	1
C	Number of bits changed in COPBF												8				
R|C	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
D	**IPV2**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
E	**OPBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
F	**CIPV2**	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
G	**COPBF**	1	0	1	0	1	1	0	1	0	1	0	1	0	0	0	1
H	**DBF**	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	1
I	Number of bits changed in COPBF												4				
R|C	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
J	**IPV1**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
K	**OPBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
L	**CIPV1**	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
M	**COPBF**	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	0
N	**DBF**	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0
O	Number of bits changed in COPBF												12				

Table 3.2.3.1: S-box and OPBFs for SAC test of 4-bit BFs as well as the S-boxes.

R|C	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
1	**Hex Index** **Pos INB**	0 4321	1 4321	2 4321	3 4321	4 4321	5 4321	6 4321	7 4321	8 4321	9 4321	A 4321	B 4321	C 4321	D 4321	E 4321	F 4321
2	**INB**	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
3	**S-box**	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7
4	**OBF1**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
5	**OBF2**	1	1	1	0	0	1	0	0	0	0	1	1	1	0	0	1
6	**OBF3**	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	1
7	**OBF4**	0	0	1	1	0	1	1	0	1	0	0	0	1	1	0	1

Table 3.2.3.2: SAC Test with Flip method of 4-bit BFs and the S-boxes.

Col | Row	**Flip of 1 bit of Index at Pos. 1**					**Flip of 1 bit of Index at Pos. 2**					**Flip of 1 bit of Index at Pos. 3**					**Flip of 1 bit of Index at Pos. 4**				
	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H	I	J	K
1	**Bef Flip**	**AfterFlip**	**1**	**1’**	**FB**	**Bef Flip**	**After Flip**	**2**	**2’**	**FB**	**Bef Flip**	**AfterFlip**	**3**	**3’**	**FB**	**Bef Flip**	**AfterFlip**	**4**	**4’**	**FB**
2	0000	0001	1	0	1	0000	0010	1	1	0	0000	0100	1	0	1	0000	1000	1	0	1
3	0001	0000	0	1	1	0001	0011	0	0	0	0001	0101	0	1	1	0001	1001	0	1	1
4	0010	0011	1	0	1	0010	0000	1	1	0	0010	0110	1	1	0	0010	1010	1	0	1
5	0011	0010	0	1	1	0011	0001	0	0	0	0011	0111	0	1	1	0011	1011	0	1	1
6	0100	0101	0	1	1	0100	0110	0	1	1	0100	0000	0	1	1	0100	1100	0	0	0
7	0101	0100	1	0	1	0101	0111	1	1	0	0101	0001	1	0	1	0101	1101	1	1	0
8	0110	0111	1	1	0	0110	0100	1	0	1	0110	0010	1	1	0	0110	1110	1	0	1
9	0111	0110	1	1	0	0111	0101	1	1	0	0111	0011	1	0	1	0111	1111	1	0	1
A	1000	1001	0	1	1	1000	1010	0	0	0	1000	1100	0	0	0	1000	0000	0	1	1
B	1001	1000	1	0	1	1001	1011	1	1	0	1001	1101	1	1	0	1001	0001	1	0	1
C	1010	1011	0	1	1	1010	1000	0	0	0	1010	1110	0	0	0	1010	0010	0	1	1
D	1011	1010	1	0	1	1011	1001	1	1	0	1011	1111	1	0	1	1011	0011	1	0	1
E	1100	1101	0	1	1	1100	1110	0	0	0	1100	1000	0	0	0	1100	0100	0	0	0
F	1101	1100	1	0	1	1101	1111	1	0	1	1101	1001	1	1	0	1101	0101	1	1	0
G	1110	1111	0	0	0	1110	1100	0	0	0	1110	1010	0	0	0	1110	0110	0	1	1
H	1111	1110	0	0	0	1111	1101	0	1	1	1111	1011	0	1	1	1111	0111	0	1	1
I	**No of Bits Changed due to Flip** 12					**No of Bits Changed due to Flip** 4					**No of Bits Changed due to Flip** 8					**No of Bits Changed due to Flip** 12				

Description of table 3.2.3.2: Here in the table 4.2.3.2 16 4-bit long input bit patterns before and after flip of 1 bit in position 1, position 2, position 3 and position 4 respectively are shown in row 2 through H of the column 1, 2, 6, 7, B, C and G, H respectively. The OPBF before and after flip of 1 bit in 1 bit in position 1, position 2, position 3 and position 4 respectively are shown in row 2 through H of the column 3, 4, 8, 9, D, E and I, J respectively. The four flipped BF after flip of 1 bit in position 1, position 2, position 3 and position 4 respectively are shown in row 2 through H of the column 5, A, F and K respectively.

Pseudo Code:

```
The flipping of bits on particular positions are made by proposing 1-bit in four ev vectors as, e0 {0001}, e1 {0010}, e2 {0100} and e3 {1000}. The Algorithm can be written as,
Start.
Step 0A: For I=0:16 For J=0:16 D[I][J] = 0; // Initializing two dimensional array D[16][16].
Step 0B: ev[4] ={{0,0,0,1},{0,0,1,0},{0,1,0,0},{1,0,0,0}}; // Initializing ev vector
Step 01: For S=0:4 For I=0:16 For J=0:16 t[S][I][J] = 16bt4x[S][I][J] ^ ev[S] // Array of input index after flip.
Step 02: For S=0:4 For I=0:16 For J=0:16 r=16bt4bf[S][I][J] ^ 16bt4bf[t[S][I][J]]; // obtain DBFs by xor operation.
Step 04: if (r==1) D[f][v]++; // Count of 1s in DBFs
// Evaluation of SAC criterion.
Step 05:  IF D[f][v]==8, for All cases 4-bit BF Satisfies SAC of 4bit BFs.
          ELSE 4-bit BF does not Satisfy SAC.
Step 06:  IF all four BFs Satisfy SAC of 4-bit BFs then the given S-Box Satisfies SAC of 4-bit S-Box.
          ELSE the given S-Box does not Satisfy SAC of 4-bit S-Box.
Stop.
```

Time complexity of the given pseudo code: Time complexity of the algorithm has been O(n) since the body contains no nested loops.

3.3 Analogy of DC to MHO-SAC of the S-boxes:

Since complement of a bit is a similar operation of the xor of bit value 1 with the said bit so complement of a 4-bit BF is similar operation to xor operation of the bit value one with the each bit of the said 4-bit BF. In DC for ID 1, 2, 4 and 8 IPV1, IPV2, IPV3 and IPV4 are complemented since BIN ID of the said IDs contain 1 in position 1, 2, 3 and 4 respectively. So the distant S-box for a certain ID in DC contains four complemented OPBFs of the S-box for the MHO-SAC-Bin ID. Now for other IDs the respective complementation of the 2, 3 or 4 IPVs together are shown in table 3.3.1. So the distant S-box for them in DC contains four complemented OPBFs of the S-box for the MHO-SAC-Bin ID. The difference S-box contains four difference 4-bit BFs. If they are balanced for a particular ID then the S-box is said to satisfy MHO-SAC-Bin ID of the 4-bit S-boxes. So it is clear from the table that the procedure of DC for ID 0 to F and MHO-SAC-0000 (Bin ID) to MHO-SAC-1111 (Bin ID) are same.

Table 3.3.1: Analogy of DC to MHO-SAC of S-boxes.

Column → Row ↓	1	2	3
	ID in Hex	Bin ID 4321	Comp. IPVs
1	0	0000	No
2	1	0001	1
3	2	0010	2
4	3	0011	1,2
5	4	0100	3
6	5	0101	1,3
7	6	0110	2,3
8	7	0111	1,2,3
9	8	1000	4
A	9	1001	4,1
B	A	1010	4,2
C	B	1011	4,2,1
D	C	1100	4,3
E	D	1101	4,3,1
F	E	1110	4,3,2
G	F	1111	4,3,2,1

4. A Brief Review of (Output) Bit Independence Criterion (BIC) of 4, 8 bit S-boxes. A short description of a 4-bit crypto S-box has been given in subsec.4.1 of sec 4. The four Input Vectors (IPVs) and four Output Boolean Functions (OPBFs) and the derivation of four IPVs and four OPBFs from elements of Index of 4-bit crypto S-box and elements of 4-bit crypto S-box respectively are illustrated in subsec.4.2.of sec.4. The (Output) Bit Independence Criterion (BIC) of 4-bit S-box is described with example and pseudo code in subsec.4.3. of sec.4.

4.1 4-bit Crypto S-boxes. A 4-bit Crypto S-box can be written as follows in Table 4.1, where the each element of the first row of Table 4.1, entitled as index, are the position of each element of the 4-bit crypto S-box within the given 4-bit crypto S-box and the elements of the 2nd row entitled as S-box are the elements of the given 4-bit crypto S-box. It can be concluded that the 1st row is fixed for all possible 4-bit crypto S-boxes. The values of each element of the 1st row are distinct, unique and vary between 0 to F in hex. The values of the each element of the 2nd row of a crypto S-box are also distinct and unique and also vary between 0 to F in hex. The values of the elements of the fixed 1st row are sequential and monotonically increasing where for the 2nd row they can be sequential or partly sequential or non-sequential. Here the given Substitution box is the 1st 4-bit S-box of the 1st S-box out of 8 of Data Encryption Standard [18][19][20].

4.2 Relation between 4-bit S-boxes and 4-bit Boolean Functions (4-bit BFs). Index of Each element of a 4-bit crypto S-box and the element itself is a hexadecimal number and that can be converted into a 4-bit bit sequence that are given in column 1 through G of row 1 and row 6 under row heading Index and S-box respectively. From row 2 through 5 and row 7 through A of each column from 1 through G of Table 4.2. shows the 4-bit bit sequences of the corresponding hexadecimal numbers of the index of each element of the given crypto S-box and each element of the crypto S-box itself. Each row from 2 through 5 and 7 through A from column 1 through G constitutes a 16 bit, bit sequence that are 16 bit long input vectors (IPVs) and 4-bit output BFs (OPBFs) respectively. column 1 through G of Row 2 is termed as 4th IPV, Row 3 is termed as 3rd IPV, Row 4 is termed as 2nd IPV and Row 5 is termed as 1st IPV whereas column 1 through G of Row 7 is termed as 4th OPBF, Row 8 is termed as 3rd OPBF, Row 9 is termed as 2nd OPBF and Row A is termed as 1st OPBF [21]. The decimal equivalent of the each IPV and the each OPBF is noted at column H of the respective rows.

4.3. (Output) Bit Independence Criterion (BIC) of 4, 8-bit S-boxes. If all possible or total six xored 4-bit BFs or DBFs (Derived BFs) are balanced for a particular 4-bit crypto S-box or 30 xored 8-bit DBFs are balanced for a particular 8-bit crypto-S-box then the said 4-bit or 8-bit S-box is said to satisfy output BIC of S-boxes [22]. The example of BIC of 4-bit S-boxes has been given in Table 4.3. below and Pseudo code with time complexity analysis are given in this section,

In Table 4.3. each column from column 1 through G of row 1 represents each element of 1st 4-bit S-box of Data Encryption Standard or DES. Column 1 through G of each row 2 through 5 is each of four OPBFs, OPBF4, OPBF3, OPBF2, OPBF1 respectively. Column 1 through G of each row 6 through B is each of six DBFs, DBF(4,3), DBF(4,2), DBF(4,1), DBF(3,2), DBF(3,1) and DBF(2,1) respectively. The analysis shows that 6 DBFs are balanced i.e. consists of 8 0s and 8 1s, so at most uncertainty to determine the occurrence of 0 and 1 value in all four OPBFs. So the given 4-bit S-box is said to satisfy (Output) Bit Independence Criterion of the 4-bit crypto S-boxes.

Pseudo Code of BIC with time complexity Analysis.

```
Start.
Step 0:
int BF[4][16], DBF[16]; // The two dimensional array BF[4][16] stores each
             OPBF of a 4-bit crypto S-box in each row and array DBF[16]
             stores Difference BFs.
int i,j;                              // Loop Variables.
int count = 0;                  // Variable to count number of balanced DBFs.
// In step 1. 6 possible two OPBFs have been xored to obtain DBFs.
Step 1:
for i=0:3; // 1st OPBF selection // for Loop 1
        for j = 3: (i+1) // 2nd OPBF selection // for Loop 2
                DBF[16] = BF[i][16]^ BF[j][16]; // Derivation of DBFs
               from two OPBFs
          If (DBF == Balanced). count++; // count number of balanced DBFs.
         End for.// End of for loop 1
 End for. // End of for loop 2
Step 2. If (count ==6) then the crypto 4-bit S-box Satisfies BIC of 4-bit S-
boxes;
      else. does not satisfy BIC of 4-bit S-boxes;
Stop.
```

Time complexity of the given pseudo code.

Time complexity of the algorithm has been $O(n^2)$ since the body contains two nested loops.

5. Generation and analysis of existing and generated 4-bit crypto S-boxes.

The procedure to analyze 4-bit crypto S-boxes with the given analyzing procedures are described in subsection 5.1. The analysis of the existing 4-bit crypto S-boxes of the Data Encryption Standard and two variants of Lucifer are given in subsection 5.2. The generated 16 4-bit crypto S-boxes from 64 distinct nonlinear BFs are also analyzed and proven to be the best possible ones. The analysis is given in section 5.3.

5.1 Cryptanalysis procedure.

'No.elr' shows number of existing linear relations out of 64 possible linear relations in a 4-bit crypto S-box. 'No.8' shows number of 8s in linear approximation table or LAT. 'N0.dif' shows number of 0s in difference distribution table or DDT and 'N8.dat' shows number of 8s in differential approximation table or DAT [21]. The procedures are discussed as follows,

In difference distribution table there are 256 cells, i.e. 16 rows and 16 columns. Each row is for each input difference varies from 0 to f in hex. Each column in each row represents each output difference varies from 0 to f in hex for each input difference. 0 in any cell indicates absence of that output difference for subsequent input difference. Such as 0 in a cell of DDT means for input difference 0 the corresponding output difference is absent. If numbers of 0s are too low or too high it supplies more information regarding concerned output difference. So an S-box is said to be immune to this cryptanalytic attack if number of 0s in DDT is close to 128 or half of total cells or

256. In the said example of 1st DES 4-bit S-box total numbers of 0s in DDT are 168. That is close to 128. So the S-box is said to be almost secure from this attack. [21]

As total number of balanced 4-bit BFs increases in Difference Analysis Table or DAT the security of S-box increases since balanced 4-bit BFs supplies at most uncertainty. Since Number of 0s and 1s in balanced 4-bit BFs are equal i.e. they are same in number means determination of each bit has been at most uncertainty. In the said example of 1st DES 4-bit S-box total numbers of 8s in DAT are 36. That is close to 32 half of total 64 cells. So the S-box has been said to be almost less secure from this attack.[21]

In linear approximation table or LAT there are 256 cells for 256 possible 4-bit linear relations. The count of 16 4-bit binary conditions to satisfy for any given linear relation is put into the concerned cell. 8 in a cell indicate that the particular linear relation is satisfied for 8, 4-bit binary conditions and remain unsatisfied for 8, 4-bit binary conditions. That is at most uncertainty. In the said example of 1st DES 4-bit S-box total numbers of 8s in LAT is 143. That is close to 128. So the S-box is said to be less secure from this attack.

The value of ${}^{n}C_{r}$ is maximum when the value of r is ½ of the value of n (when n is even). Here the maximum number of linear approximations is 64. So if the total satisfaction of linear equation is 32 out of 64 then the number of possible sets of 32 linear equations is the largest. That means if the total satisfaction is 32 out of 64 then the number of possible sets of 32 possible linear equations is ${}^{64}C_{32}$. That is maximum number of possible sets of linear equations. If the value of total number of linear relations is closed to 32 then it is more cryptanalysis immune. Since the number of possible sets of linear equations are too large to calculate. As the value goes close to 0 or 64 it reduces the sets of possible linear equations to search, that reduces the effort to search for the linear equations present in a particular 4-bit crypto S-box. In this example total satisfaction is 21 out of 64. Which means the given 4-bit S-Box is not a good 4 bit crypto S-box or not a good crypt analytically immune 4-bit crypto S-box.

If the value of total number of existing linear relations for a 4-bit crypto S-box is 24 to 32, then the lowest numbers of sets of linear equations are 250649105469666120. This is a very large number to investigate. So the 4-bit crypto S-box is declared as a good 4-bit crypto S-box or 4-bit crypto S-box with good security. If it is between 16 through 23 then the lowest numbers of sets of linear equations are 488526937079580. This not a small number to investigate in today’s computing scenario so the S-boxes are declared as medium 4-bit crypto S-box or 4-bit crypto S-box with medium security. The 4-bit crypto S-boxes having existing linear equations less than 16 are declared as poor 4-bit crypto S-Box or vulnerable to cryptanalytic attack [21].

‘No.sac’, ‘N2sac’, ‘N3sac’ and ‘Nalsac’ gives total number times four 4-bit BFs of the concerned S-box satisfies 4 simple first order SAC, 6, 2nd order HO-SAC, 4, 3rd order HO-SAC and 16, 1st, 2nd, 3rd, and 4th order HO-SAC respectively.

5.2 Discussion on cryptanalysis of 32 4-bit crypto S-boxes of Data Encryption Standard or DES and 4 S-boxes of two variants of Lucifer.

Data Encryption Standard or DES algorithm contains 8 S-boxes with four rows in each S-box. Each row in DES S-box is a 4-bit crypto S-box of DES algorithm. The results of cryptanalysis of 32 DES 4-bit crypto S-box is given in table.5.1 and results are discussed in discussion below,

DES S-boxes	No.elr	No.8	N0.ddt	N8.dat	No.sac	N2sac	N3sac	Nalsac
e4d12fb83a6c5907	21	143	168	36	7	15	11	36
0f74e2d1a6cb9538	29	143	168	36	7	17	9	36
41e8d62bfc973a50	23	138	168	36	8	15	11	36
fc8249175b3ea06d	25	154	166	42	10	20	12	42
f18e6b34972dc05a	24	132	162	30	6	12	9	30
3d47f28ec01a69b5	21	143	166	30	8	12	7	30

0e7ba4d158c6932f	31	143	166	21	4	10	6	21
d8a13f42b67c05e9	20	126	168	36	8	12	12	36
a09e63f51dc7b428	17	133	162	30	7	12	8	30
d709346a285ecbf1	22	133	168	30	7	13	8	30
d6498f30b12c5ae7	23	151	166	21	6	9	4	21
1ad069874fe3b52c	28	158	174	30	6	11	10	30
7de3069a1285bc4f	22	136	168	36	8	16	10	36
d8b56f03472c1ae9	22	136	168	36	8	16	10	36
a690cb7df13e5284	20	136	168	36	8	16	10	36
3f06a1d8945bc72e	22	136	168	36	8	16	10	36
2c417ab6853fd0e9	25	137	162	30	6	14	8	30
eb2c47d150fa3986	20	143	166	36	8	16	9	36
421bad78f9c5630e	30	130	160	27	6	11	7	27
b8c71e2d6f09a453	21	134	166	18	3	7	6	18
c1af92680d34e75b	30	141	159	36	8	16	10	36
af427c9561de0b38	29	127	164	36	7	15	11	36
9ef528c3704a1db6	24	127	168	18	5	7	5	18
432c95fabe17608d	24	130	162	30	6	12	9	30
4b2ef08d3c975a61	26	134	168	30	7	13	8	30
d0b7491ae35c2f86	27	145	166	30	7	14	7	30
14bdc37eaf680592	28	137	168	36	8	16	10	36
6bd814a7950fe23c	25	135	173	0	0	0	0	0
d2846fb1a93e50c7	23	144	161	30	8	14	7	30
1fd8a374c56b0e92	20	147	174	27	9	12	4	27
7b419ce206adf358	27	132	166	18	5	7	5	18
21e74a8dfc90356b	28	138	168	39	8	16	12	39

Table.5.1. Cryptographic analysis of 32 DES 4-bit crypto S-boxes.

Discussion.

In table.5.1. out of 32 DES S-boxes 1 have 17, 3 have 21, 4 have 22, 1 have 23, 3 have 24, 3 have 25, 1 have 26, 2 have 27, 3 have 28, 2 have 29, 2 have 30 and 1 have 31 existing linear relations i.e. 24 S-boxes out of 32 are less secure from this attack and 8 out of 32 are immune to this attack. Again out of 32 DES S-boxes 1 have 126, 2 have 127, 2 have 130, 1 have 132, 2 have 133, 2 have 134, 1 have 135, 4 have 136, 2 have 137, 2 have 138, 1 have 141, 5 have 143, 1 have 144, 1 have 145, 1 have 147, 1 have 151, 1 have 154 and 1 have 158 8s in LAT. That is All S-boxes are less immune to this attack. Again out of 32 DES S-boxes 1 have 159, 1 have 160, 1 have 161, 4 have 162, 1 have 164, 8 have 166, 13 have 168, 1 have 173 and 2 have 174 0s in DDT. That is all S-boxes are secured from this attack. At last out of 32 DES S-boxes 1 have 0, 3 have 18, 2 have 21, 2 have 27, 10 have 30, 12 have 36, 1 have 39 and 1 have 42 8s in DAT i.e. they have been less secure to this attack. The comparative analysis has proved that linear approximation analysis is the most time efficient cryptanalytic algorithm for 4-bit S-boxes. In 'nosac' the lowest value is 0 and maximum value is 10 where in 'n2sac', 'n3sac' and 'nalsac' lowest values are 0, 0, 0 and maximum values are 16, 12 and 39 respectively. But numbers of optimum as well as better result i.e. 16 for 'nosac' is absent, close to 24 for 'n2sac', close to 16 for 'n3sac' and close to 64 for 'nalsac' has been very less in numbers. So the 32 DES 4-bit S-boxes are observed to be less secure.

Discussion on cryptanalysis of 4, 4-bit crypto S-boxes of 2 variants of Lucifer.

2 variants of Lucifer one by feistel [22], and one by Sorkin [23] contain total 4 crypto S-boxes. The cryptanalysis of the concerned 4, crypto S-boxes is shown in table.8. and the result is also discussed below.

Lucifer S-boxes	No.elr	No.8	N0.ddt	N8.dat	No.sac	N2sac	N3sac	Nalsac
F-3085124fd9ce6ba7	25	132	163	36	8	16	9	36
F-8d16c4fb325e907a	31	115	154	36	10	12	11	36
S-cf7aedb026319458	25	132	163	36	8	16	9	36
S-72e93b04cd1a5f85	28	58	151	18	6	5	7	18

Table.5.2. Cryptographic analysis of 4, 4 bit crypto S-boxes of 2 variants of Lucifer.

Discussion.

In table.5.2. out of 4, 4-bit Crypto S-boxes 2 have 25, 1 have 28 and 1 have 31 existing linear relations i.e all 4 crypto 4-bit S-boxes are almost secure from this attack. Again out of 4, 4-bit crypto S-boxes, 2 have 132, 1 have 115 and 1 have 58 8s in LAT i.e. 3 4-bit crypto S-boxes out of four are secure from this attack and one is a poor 4-bit crypto S-box from the angle of this attack. Again out of 4, 4-bit crypto S-boxes 2 have 163, one have 154 and one have 151 0s in DDT so all of four S-boxes are seen to secure from the attack. From the angle of this attack 3 have 36 and one have 18 8s in DAT so all of four 4-bit crypto S-boxes are less secure to this attack.

Now first S-box in table.5.2. has 8 out of total 16 SFO SAC satisfaction, 16 out of total 24 2nd order MHO SAC satisfaction, 9 out of total 16 3rd order MHO SAC satisfaction, 36 out of total 64 all MHO SAC satisfaction so from this angle it is a poor 4-bit crypto S-box from this angle.

Now second S-box in table.5.2. has 10 out of total 16 SFO SAC satisfaction, 12 out of total 24 2nd order MHO SAC satisfaction, 11 out of total 16 3rd order MHO SAC satisfaction, 36 out of total 64 all MHO SAC satisfaction so from this angle it is a almost good 4-bit crypto S-box from this angle.
Now third S-box in table.8. has 8 out of total 8 SFO SAC satisfaction, 16 out of total 24 2nd order MHO SAC satisfaction, 9 out of total 16 3rd order MHO SAC satisfaction, 36 out of total 64 all MHO SAC satisfaction so from this angle it is a poor 4-bit crypto S-box from this angle.

Now fourth S-box in table.5.2. has 8 out of total 6 SFO SAC satisfaction, 5 out of total 24 2nd order MHO SAC satisfaction, 7 out of total 16 3rd order MHO SAC satisfaction, 36 out of total 64 all MHO SAC satisfaction so from this angle it is a very poor 4-bit crypto S-box from this angle.

5.3 Analysis of generated 16 4-bit crypto S-boxes from 64 distinct 4-bit BFs.

In this subsection a detailed discussion on cryptanalysis of 16, 4-bit crypto S-boxes generated from 64 balanced nonlinear BFs with nonlinearity 4 and 2 is given . The result of application of cryptanalysis algorithms of 4-bit crypto S-boxes on 16 generated 4-bit crypto S-boxes are shown in table.5.3. below and results are discussed in the following discussion section in brief.

Ad.el.	S-boxes un IP 19	No.elr	No.8	N0.ddt	N8.dat	No.sac	N2sac	N3sac	Nalsac
0	019edb76f2c5a438	23	117	150	36	08	14	11	36
1	12afec8703d6b549	31	121	155	36	7	14	11	36
2	23b0fd9814e7c65a	22	135	157	36	9	16	9	36
3	34c10ea925f8d76b	39	128	157	27	5	11	9	27
4	45d21fba3609e87c	27	115	150	36	10	12	12	36
5	56e320cb471af98d	37	125	155	36	8	14	11	36
6	67f431dc582b0a9e	29	132	157	36	10	15	8	36
7	780542ed693c1baf	34	125	157	27	5	10	9	27
8	891653fe7a4d2cb0	23	117	150	36	8	14	11	36
9	9a27640f8b5e3dc1	31	121	155	36	7	14	11	36
A	ab3875109c6f4ed2	22	135	157	36	9	16	9	36
B	bc498621ad705fe3	39	128	157	27	5	11	9	27
C	cd5a9732be8160f4	27	115	150	36	10	12	12	36
D	de6ba843cf927105	37	125	155	36	8	14	11	36
E	ef7cb954d0a38216	29	132	157	36	10	15	8	36
F	f08dca65e1b49327	34	125	157	27	5	10	9	27

Table.5.3. Cryptographic analysis of 16, 4 bit crypto S-boxes under IP (x^4+x+1) with DE 19 over Galois field GF(2^4).

Discussion.

Out of total 16 4-bit crypto S-boxes 2 have 22, 2 have 23, 2 have 27, 2 have 29, 2 have 31, 2 have 34, 2 have 37 and 2 have 39 existing linear relations i.e. all of 16 4-bit crypto S-boxes are secure from this cryptanalytic attack. Again out of total 16 4-bit crypto S-boxes 2 have 115, 2 have 117, 2 have 121, 4 have 125, 2 have 128, 2 have 132 and 2 have 135 8s in LAT i.e. they are secure from linear cryptanalysis of 4-bit S-boxes. Now out of total 16 4-bit crypto S-boxes 4 have 150, 4 have 155 and 8 have 157 0s in DDT i.e. from this attack they are quite secure too. Again out of total 16 4-bit crypto S-boxes 4 have 27 and 12 have 36 8s in DAT i.e. they are in secure region of this attack.

S-boxes with additive element 0 to F in hex has a range 5 to 10 out of total 16 SFO SAC satisfactions, 10 to 16 out of total 24 2nd order MHO SAC satisfaction, 8 to 12 out of total 16 3rd order MHO SAC satisfaction, 27 to 36 out of total 64 all MHO SAC satisfaction so they are poor 4-bit crypto S-boxes from only SFO SAC angle but good secure 4-bit crypto S-boxes from MHO SAC angle.

6. Conclusion.

Here in this paper cryptography related properties of 4-bit BFs is reviewed in details. The FO-SAC, SFO-SAC and MHO-SAC is also described with their new methods and algorithms and at last a comparative study is done with generated 16 4-bit crypto S-boxes to 16 DES and 4 Lucifer S-boxes. The analysis proves that the generated 4-bit S-boxes can be termed as the best possible ones. About the new SAC methods and algorithms it can be concluded that they are less complex in implementation and less time consuming. So the new generated algorithms of FO-SAC, SFO-SAC and MHO-SAC and the generated 4-bit crypto S-boxes are prove to be the best possible ones.

7. Acknowledgements.

I would like to acknowledge Prof. (Dr.) Gopa Sen, Head Dept. Radio Physics and Electronics of the University of Calcutta and Prof. Amlan Chakrabarti, Director of the A K Choudhury School of Information Technology of the University of Calcutta for providing me the uninterrupted infrastructure to carry out the research. I would also like to thank TEQIP-Phase-II for providing me the financial support up to 30th November 2016.

References.

1. Adams, Carlisle, Tavares, Stafford, "The structured design of cryptographically good S-boxes", J. Cryptology (1990) 344 vol. 3, pp : 27-41.

2. Chan Yeob Yeun Design, Analysis and applications of cryptographic techniques, Department of Mathematics, Royal Holloway University of London, Year 2000.

3. H.M.Heys and S.E. Tavares. Substitution-permutation networks resistant to differential and linear cryptanalysis, Journal of Cryptology,9,1-19, year 1996.

4. Webster, A.F. and Travares, S.E."On Design of S-boxes", Advances in Cryptology : Proc. of Crypto 85, Springer-Verlag pp. 523-534, 1986,

5. Sarkar, Palash, gupta, kishan chand, "Computing Walsh Transform from the Algebraic Normal Form of a Boolean function", Link: http://ac.els-cdn.com/S1571065304005426/1-s2.0-S1571065304005426-main.pdf.

6. Carlet, Claude, "Vectorial Boolean Functions for Cryptography", University of Paris 8, France, Link: www.math.univ-paris13.fr/~carlet/chap-vectorial-fcts-corr.pdf.

7. Chakrabarty, K, Hayes, J.P, "Balanced Boolean functions", IEE Proc.-Comput. Digit. Tech., Vol. 145, No. I , January 1998, Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=675544.

8. Ali, Ayden Selcuk, "On Probability of Success in Linear and Differential Cryptanalysis", Link: https://www.cs.bilkent.edu.tr/~selcuk/teaching/cs519/LC_DC.pdf.

9. "BOOLEAN FUNCTIONS", T-79.5501, Cryptology, Background material, February 20, 2007, pp: 1-9, Link: www.tcs.hut.fi/Studies/T-79.5501/2007SPR/lectures/boolean.pdf.
10. Sala, Massimiliano, Simonetti, Ilaria, "An algebraic description of Boolean functions", 15/I/2007, BCRI–CGC–preprint, pp: 1-8, Link: http://www.bcri.ucc.ie.
11. H.M.Heys. A tutorial on linear and differential cryptanlysis.cryptologia,26,189-221,2002.
12. Dey, S and Ghosh, R, "A smart review and two new techniques using 4-bit Boolean functions for cryptanalysis of 4-bit crypto S-boxes.", Vol.40, issue.3, pp.1-19, International Journal of Computers and Applications, Taylor and Francis publishers, SCimago and country rank and Peerus indexed and UGC approved, year. 2018. ISSN. 1206-212X. DOI. **https://doi.org/10.1080/1206212X.2018.1504459**.
13. Mouha N., Wang Q., Gu D., Preneel B. Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. In: Wu CK., Yung M., Lin D. (eds) Information Security and Cryptology. Inscrypt 2011. Lecture Notes in Computer Science, vol 7537. Springer, Berlin, Heidelberg, year 2012.
14. Dey, S. and Ghosh, R. (2018) A Review of Existing 4-Bit Crypto S-Box Cryptanalysis Techniques and Two New Techniques with 4-Bit Boolean Functions for Cryptanalysis of 4-Bit Crypto S-Boxes*. Advances in Pure Mathematics, **8**, 272-306. ISSN Online: 2160-0384 ISSN Print: 2160-0368 DOI: **10.4236/apm.2018.83015.**
15. Dey Sankhanil and Ghosh, Ranjan, "Extended SAC: A Review on DC and SAC of 4-bit BFs and S-Boxes and a New Algorithm on DC of S-Boxes based on Various Types of SAC including the Extended Higher Order SAC" ,Vol-2, Number 3, pp-(1-10), April-2017, Journal-Circulation in Computer Science, CSL Press, NY, USA, ISSN. 2456-3692, **https://doi.org/10.22632/ccs-2017-251-56**.
16. Cusick T.W. Boolean functions satisfying a higher order strict avalanche criterion. In: Helleseth T. (eds) Advances in Cryptology — EUROCRYPT '93. EUROCRYPT 1993. Lecture Notes in Computer Science, vol 765. Springer, Berlin, Heidelberg, Year 1994.
17. Henning Schulzrinne ,Network Security: Secret Key Cryptography, Columbia University, New York, Year 2000.
18. Data Encryption Standard, Federal Information Processing Standards Publication (FIPS PUB) 46, National Bureau of Standards, Washington, DC (1977).
19. Data Encryption Standard (DES), Federal Information Processing Standards Publication (FIPS PUB) 46-3, National Institute of Standards and Technology, Gaithersburg, MD (1999).
20. Dey Sankhanil and Ghosh Ranjan, "A Review of Cryptographic Properties of S-Boxes with Generation and Analysis of Crypto Secure S-Boxes", International Journal of Electronics and Information Engineering, vol.8, no.1, pp.49-73, DOI: 10.6636/ IJEIE.201803.8(1).06. Mar 2018.
21. Sankhanil Dey, Amlan Chakrabarti , Ranjan Ghosh . (2019) 4-bit crypto S-boxes: Generation with irreducible polynomials over Galois field GF(2^4) and cryptanalysis., International Journal of Tomography and Simulation, ISSN: 2319-3336, Vol. 32, Issue No. 3, CESER publication.
22. Horst Feistel. Block Cipher Cryptographic System, US Patent 3,798,359. Filed June 30, 1971. (IBM).
23. A. Sorkin, (1984). LUCIFER: a cryptographic algorithm. Cryptologia, **8**(1), 22–35, 1984.

Row	**Column**	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G
1	**S-box**	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7
2	**OPBF4**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
3	**OPBF3**	1	1	1	0	0	1	0	0	0	0	1	1	1	0	0	1
4	**OPBF2**	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	1
5	**OPBF1**	0	0	1	1	0	1	1	0	1	0	0	0	1	1	0	1
6	**DBF4,1**	1	0	0	1	0	0	0	1	1	1	0	1	1	0	0	1
7	**DBF4,2**	0	0	1	0	1	0	0	1	1	0	1	1	0	1	0	1
8	**DBF4,3**	0	1	0	0	0	0	1	1	0	1	1	0	1	1	0	1
9	**DBF3,2**	0	1	1	0	1	0	1	0	1	1	0	1	1	0	0	0
A	**DBF3,1**	1	1	0	1	0	0	1	0	1	0	1	1	0	1	0	0
B	**DBF2,1**	1	0	1	1	1	0	0	0	0	1	1	0	1	1	0	0

Table.4.3. BIC Analysis of 1st 4-bit S-box out of 4 of 1st S-box of DES.

Row ------ Col.	**4-bit Input**			**1-bit o/p**
	DE	**HE**	**4 Bits**	
	1	**2**	**3**	**4**
1	0	0	0000	0
2	1	1	0001	1
3	2	2	0010	0
4	3	3	0011	1
5	4	4	0100	0
6	5	5	0101	1
7	6	6	0110	0
8	7	7	0111	1
9	8	8	1000	0
10	9	9	1001	1
11	10	A	1010	0
12	11	B	1011	1
13	12	C	1100	0
14	13	D	1101	1
15	14	E	1110	0
16	15	F	1111	1

Table 2.1.a Truth Table of a 4-bit BF

Row ----- Col.	**DE**	**HE**	**IV1**	**IV2**	**IV3**	**IV4**	**OV**
	1	**2**	**3**	**4**	**5**	**6**	**7**
			MSB				
1	0	0	0	0	0	0	1
2	1	1	0	0	0	1	0
3	2	2	0	0	1	0	1
4	3	3	0	0	1	1	0
5	4	4	0	1	0	0	1
6	5	5	0	1	0	1	0
7	6	6	0	1	1	0	1
8	7	7	0	1	1	1	0
9	8	8	1	0	0	0	1

10	9	9	1	0	0	1	0
11	10	A	1	0	1	0	1
12	11	B	1	0	1	1	0
13	12	C	1	1	0	0	1
14	13	D	1	1	0	1	0
15	14	E	1	1	1	0	1
16	15	F	1	1	1	1	0
DE. of IVs			255	3855	13107	21845	43690
			LSB				

Table 2.1.b 16-Bit Input Vectors (IVs).

Row ------ Col.	**4-bit Input**			**4-bit Output**		
	DE	**HE**	**4 Bit IVs**	**DE**	**HE**	**4 Bit OVs**
	1	**2**	**3**	**4**	**5**	**6**
1	0	0	0000	0	0	0000
2	1	1	0001	1	1	0001
3	2	2	0010	2	2	0010
4	3	3	0011	3	3	0011
5	4	4	0100	4	4	0100
6	5	5	0101	5	5	0101
7	6	6	0110	6	6	0110
8	7	7	0111	7	7	0111
9	8	8	1000	8	8	1000
10	9	9	1001	9	9	1001
11	10	A	1010	10	A	1010
12	11	B	1011	11	B	1011
13	12	C	1100	12	C	1100
14	13	D	1101	13	D	1101
15	14	E	1110	14	E	1110
16	15	F	1111	15	F	1111

Table 2.1.C.Truth Table of a 4-bit S-box

Table.2.2 16 Linear and 16 Affine BFs of which 11 each are obtained by XOR operations of 4 Basic Linear BFs

DEIB	IBVs	Linear Relations	Linear BFs(C=0)		Affine BFs (C=1)		ANF Coefficients
	$x_1x_2x_3x_4$		0123456789abcdef	DEBF	0123456789abcdef	DEBF	0-1234-56789a-bcde-f
RowNo	1	2	3	4	5	6	7
0	0 0 0 0	$F_0(x) = C$	0000000000000000	00000	1111111111111111	65535	C-0000-000000-0000-0
1	0 0 0 1	$F_1(x) = C+x_1$	0000000011111111	00255	1111111100000000	65280	C-1000-000000-0000-0
2	0 0 1 0	$F_2(x) = C+x_2$	0000111100001111	03855	1111000011110000	61680	C-0100-000000-0000-0
3	0 0 1 1	$F_3(x) = C+x_1+x_2$	0000111111110000	04080	1111000000001111	61455	C-1100-000000-0000-0
4	0 1 0 0	$F_4(x) = C+x_3$	0011001100110011	13107	1100110011001100	52428	C-0010-000000-0000-0
5	0 1 0 1	$F_5(x) = C+x_1+x_3$	0011001111001100	13260	1100110000110011	52275	C-1010-000000-0000-0
6	0 1 1 0	$F_6(x) = C+x_2+x_3$	0011110000111100	15420	1100001111000011	50115	C-0110-000000-0000-0
7	0 1 1 1	$F_7(x) = C+x_1+x_2+x_3$	0011110011000011	15555	1100001100111100	49980	C-1110-000000-0000-0
8	1 0 0 0	$F_8(x) = C+x_4$	0101010101010101	21845	1010101010101010	43690	C-0001-000000-0000-0
9	1 0 0 1	$F_9(x) = C+x_1+x_4$	0101010110101010	21930	1010101001010101	43605	C-1001-000000-0000-0
10	1 0 1 0	$F_a(x) = C+x_2+x_4$	0101101001011010	23130	1010010110100101	42405	C-0101-000000-0000-0
11	1 0 1 1	$F_b(x) = C+x_1+x_2+x_4$	0101101010100101	23205	1010010101011010	42330	C-1101-000000-0000-0
12	1 1 0 0	$F_c(x) = C+x_3+x_4$	0110011001100110	26214	1001100110011001	39321	C-0011-000000-0000-0
13	1 1 0 1	$F_d(x) = C+x_1+x_3+x_4$	0110011010011001	26265	1001100101100110	39270	C-1011-000000-0000-0
14	1 1 1 0	$F_e(x) = C+x_2+x_3+x_4$	0110100101101001	26985	1001011010010110	38550	C-0111-000000-0000-0
15	1 1 1 1	$F_f(x) = C+x_1+x_2+x_3+$	0110100110010110	27030	1001011001101001	38505	C-1111-000000-0000-0

DEIB Stands for '<u>D</u>ecimal <u>E</u>quivalent of <u>I</u>nput <u>B</u>its' and DEBF stands for '<u>D</u>ecimal <u>E</u>quivalent of <u>B</u>oolean <u>F</u>unction'

Table.2.3 Properties of 4-bit BFs.

BF BF CBF CBF FO-SAC SFO-SAC SFO MHO-SAC MHO
(Dec) (BInary) 10 L (Dec) (BInary) 10 L ANF Coefficients Mm 8421-----3569AC-7BDE-F---SUM-----3569AC-7BDE-F---SUM
----- ----- ---------------- -- - ----- ---------------------- -- - ---------------- ---- ------ ----------- ------- - ------ -------------------- -------
00001 0000000000000001 1f 2 65534 1111111111111110 f1 4 C-0000-000000-0000-1 F1 0000------000000--0000--0----000-------000000--0000--0----000
00129 0000000010000001 2e 2 65406 1111111101111110 e2 4 C-0001-001011-0111-0 E2 0000------000000--0000--0----000-------000000--0000--0----000
00022 0000000000010110 3d 2 65513 1111111111101001 d3 4 C-0000-000000-0111-1 D3 0000------000000--0000--0----000-------000000--0000--0----000
00831 0000001100111111 88 2 64704 1111110011000000 88 4 C-0000-000111-0000-0 C4 1110-----001011--0001--0----004--------111111-1110---0---009
00313 0000000100111001 5b 2 65222 1111111011000110 b5 4 C-0000-000011-1010-1 B5 0000-----000000--0000--0----000-------000000--0000--0----000
01427 0000010110010011 6a 2 64108 1111101001101100 a6 4 C-0001-011011-0000-0 A6 1111-----111111--1111--1----011-------111111--1011--1----010