A review of cryptographic properties of S-boxes with Generation and Analysis of crypto secure S-boxes.

Sankhanil Dey[1] and Ranjan Ghosh[2],
Institute of Radio Physics and Electronics[1,2],
University of Calcutta,
92 A P C Road, Kolkata-700009,
sdrpe_rs@caluniv.ac.in[1], rghosh47@yahoo.co.in[2].

Abstract. In modern as well as ancient ciphers of public key cryptography, substitution boxes find a permanent seat. Generation and cryptanalysis of 4-bit as well as 8-bit crypto S-boxes is of utmost importance in modern cryptography. In this paper, a detailed review of cryptographic properties of S-boxes has been illustrated. The generation of crypto S-boxes with 4-bit as well as 8-bit Boolean functions (BFs) and polynomials over Galois field GF(p^q) has also been of keen interest of this paper. The detailed analysis and comparison of generated 4-bit and 8-bit S-boxes with 4-bit as well as 8-bit S-boxes of Data Encryption Standard (DES) and Advance Encryption Standard (AES) respectively, has incorporated with example. Detailed analysis of generated S-boxes claims a better result than DES and AES in view of security of crypto S-boxes.

Keywords. Substitution box, S-box, Boolean Functions, Strict Avalanche Criterion, Polynomials, Finite fields, Galois fields.

1. Introduction. Substitution box or S-box is of utmost importance in block ciphers of public key cryptography from the initial days. A 4-bit S-box has been defined as a box of (2^4 =) 16 elements varies from 0 to F in hex, arranged in a random manner as used in Data Encryption Standard or DES [AT90][HF71][NT77][NT99]. Similarly for 8 bit S-box, number of elements are 2^8 or 256 those varies from 0 to 255 as used in Advance Encryption Standard or AES [DR00][VM95]. So the construction of S-boxes is a major issue in cryptology from initial days. Use of irreducible polynomials to construct 8-bit S-boxes had already been adopted by crypto community. But the study of IPs has been limited to almost binary Galois field GF(2^q) as used in AES S-boxes [DR00][VM95]. So it is important to study 4-bit BFs, 8-bit BFs and polynomials over Galois Field GF(p^q) where p>2 in public key cryptography. A brief literature study on polynomials has been elaborated in sec.2.

A 4-bit Boolean Function (BF) gives 1-bit output for 4 input bits [AT90] and represented in the form of a 16-bit output (column) vector. The truth table of a 4-bit BF has been represented by a 16-bit output vector each of whose bit is an output bit corresponding to 16 possibilities of 4-bit sequential inputs from ‘0000’ to ‘1111’. The 16 rows of the 4-bit sequential inputs, each bit at the same column position comprises of 16 bits and thereby four 16-bit columns provide four 4-bit input vectors which are common for all 4-bit BFs. Since there are 16 output bits so there are 2^{16} (=65536) different possibilities whose decimal equivalent vary between 0 and 65535 [AT90]. Hence, 4-bit BFs have four 16-bit input vectors and 65536 possible 16-bit output vectors. Where an 8-bit BF gives 1-bit output for 8 input bits [DR00] and represented in the form of a 256-bit output (column) vector. The truth table of a 8-bit BF is represented by a 256-bit output vector each of whose bit is an output bit corresponding to 256 possibilities of 8-bit sequential inputs from ‘00000000’ to ‘11111111’. The 256 rows of the 8-bit sequential inputs, each bit at the same column position comprises of 256 bits and thereby eight 256-bit columns provide eight 8-bit input vectors which are common for all 8-bit BFs. The 256 output bits, there are 2^{256} different possibilities whose decimal equivalent vary between 0 and 2^{256}-1 [VM95]. Hence, 8-bit BFs have eight 256 bit long 8-bit input vectors and 2^{256} possible 256-bit output vectors. Hence for generation and security analysis of 4-bit or 8-bit S-boxes it is an urgent need to study cryptographic properties of S-boxes as well as security of S-boxes with 4-bit or 8-bit BFs. In other words a 4-bit S-box can be represented by a four valued 4-bit BF. If the 1st bit of the 4 output bits is taken sequentially for each element of the 16 elements of an S-box, one gets the 1st BF; 2nd sequence of output bit, the 2nd BF; 3rd sequence of output bit, the 3rd BF and 4th sequence of output bit, the 4th BF [AT90] respectively. Some cryptographic properties and security analysis of 4-bit S-boxes such as Output Bit Independence Criterion (BIC) of 4-bit S-boxes, SAC of 4-bit S-boxes, Higher order SAC of 4-bit S-boxes, Extended SAC of 4-bit S-boxes, Linear Cryptanalysis of 4-bit S-boxes, Differential Cryptanalysis of 4-bit S-boxes, and Differential Cryptanalysis with 4-bit BFs of 4-bit S-boxes as well as Linear Approximation Analysis of 4-bit S-boxes has been reported below in brief.

A 4-bit S-box consists of four 4-bit BFs. In Output Bit Independence Criterion or BIC the difference or xored BFs of all two possible 4-bit BFs of the concerned S-box has been taken under consideration. If all 6 difference 4-bit BFs have been balanced then the criterion has been satisfied for the concerned S-box. Since all 6 difference 4-bit BFs have been balanced so the prediction of a bit value to be one or zero is in at most uncertainty [AT90]. A brief review of BIC of 4-bit as well as 8-bit S-boxes has been illustrated in subsec.3.1of sec 3.

In Strict Avalanche Criterion, 4 IPVs of a 4-bit BF has been complemented one at a time. If in complemented four 4-bit BFs 8 bit values has been changed and 8 bit values remains same then the 4-bit BF has been said to satisfy Strict Avalanche Criterion of 4-bit BFs [AT90][CA90]. Complementing 4th IPV means interchanging each distinct 8 bit halves of a 4-bit Output BF, whereas complementing 3rd IPV means interchanging each distinct 4 bit halves of each distinct 8 bit halves, whereas complementing 2nd IPV means interchanging each distinct 2 bit halves of each distinct 4 bit halves of each distinct 8 bit halves and complementing 1st IPV means interchanging each bit of all distinct 2-bit halves of a 16 bit long 4-bit BF. In this paper this shifting property has been used to construct an algorithm of SAC of 4-bit BFs. Another new algorithm with flip of index bits has also been introduced in this paper. If all four 4-bit BFs of a 4-bit S-box satisfy SAC for 4-bit BFs then the concerned S-box has

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.26452v1 | CC BY 4.0 Open Access | rec: 22 Jan 2018, publ: 22 Jan 2018

been said to satisfy SAC of 4-bit S-boxes [AT90][CA90]. A detailed Review of old algorithm and new algorithms of SAC has been described in subsec.3.2 of section 3.

In Higher Order Strict Avalanche Criterion (HO-SAC) of 4-bit BFs four IPVs of a 4-bit S-box have been complemented two or three at a time [BS96]. If in complemented ten 4-bit BFs 8 bit values has been changed and 8 bit values remains same then the 4-bit BF has been said to satisfy HO-SAC of 4-bit BFs. A detailed review of old as well as two new algorithms with previous shift method and flip of index bits method has been introduced in this paper in subsec.3.3. of sec.3. In this Paper a detailed review of a new algorithm entitled Extended HO-SAC has been introduced in which four IPVs have been complemented at a time. An Analogy of Extended HO-SAC and Differential Cryptanalysis of 4-bit S-boxes have also been elaborated in subsec.3.3. of section.3.

In Differential Cryptanalysis of 4-bit crypto S-boxes the 16 distant input S-boxes have been obtained by xor operation with each of 16 input differences varies from 0 to F in hex to all 16 elements of input S-box one at a time. The 16 distant S-boxes have been obtained by shuffling the elements of the original S-box in a certain order in which the elements of the input S-boxes have been shuffled in concerned distant input S-boxes. The 16 elements of each S-box and the elements in corresponding position of corresponding distant S-box has been xored to obtain the Difference S-box. The Difference S-box may or may not be a Crypto S-box since it may not have all unique and distinct elements in it. The count of each element from 0 to F in Difference S-box have been noted and put in Difference Distribution Table (DDT) for security analysis of the S-box [HH96][HH02]. The concept has been reviewed in detail in subsec.3.4 of sec. 3.

In this paper a review of the new algorithm using 4-bit BFs for Differential Cryptanalysis of 4-bit crypto S-boxes have been reviewed. An input S-box can be decomposed into four 4-bit Input Vectors (IPVs) with Decimal Equivalents 255 for 4th IPV, 3855 for 3rd IPV, 13107 for 2nd IPV, and 21845 for 1st IPV respectively. Now we complement all IPVs one, two, three and four at a time to obtain 16 4-bit Distant input S-boxes. Each of four Output BFs is shifted according to the Shift of four IPVs of input S-boxes to form four IPVs of Distant input S-boxes to obtain Distant S-boxes. The four 4-bit output BFs of S-boxes are xored bitwise with four 4-bit BFs of Distant S-boxes to obtain four 4-bit Difference BFs. For 16 Distant Output S-boxes there are 64 Difference BFs. Difference BFs are checked for balanced-ness i.e. for at most uncertainty. The Table in which the balanced-nesses of 64 Difference BFs have been noted has been called as Differential Analysis Table (DAT). The Theory has been elaborated in subsec. 3.5 of sec.3.

In Linear Cryptanalysis of 4-bit crypto S-boxes, every 4-bit linear relations have been tested for a particular 4-bit crypto S-box. The presence of each 4-bit unique linear relation is checked by satisfaction of each of them for all 16, 4-bit unique input bit patterns and corresponding 4-bit output bit patterns, generated from the index of each element and each element respectively of that particular crypto S-box. If they are satisfied 8 times out of 16 operations for all 4-bit unique input bit patterns and corresponding 4-bit output bit patterns, then the existence of the 4-bit linear equation is at a stake. The probability of presence and absence of a 4-bit linear relation both are (= 8/16) ½. If a 4-bit linear equation is satisfied 0 times then it can be concluded that the given 4-bit linear relation is absent for that particular 4-bit crypto S-box. If a 4-bit linear equation is satisfied 16 times then it can also be concluded that the given 4-bit linear relation is present for that particular 4-bit crypto S-box. In both the cases full information is adverted to the cryptanalysts. The concept of probability bias was introduced to predict the randomization ability of that 4-bit S-box from the probability of presence or absence of unique 4-bit linear relations. The result is better for cryptanalysts if the probability of presence or absences of unique 4-bit linear equations are far away from ½ or near to 0 or 1. If the probabilities of presence or absence of all unique 4-bit linear relations are ½ or close to ½, then the 4-bit crypto S-box has been said to be linear cryptanalysis immune, since the existence of maximum 4-bit linear relations for that 4-bit crypto S-box is hard to predict [HH96][HH02]. Heys also introduced the concept of Linear Approximation Table (LAT) in which the numbers of times, each 4-bit unique linear relation have been satisfied for all 16, unique 4-bit input bit patterns and corresponding 4-bit output bit patterns of a crypto S-box have been noted. The result is better for a cryptanalysts if the numbers of 8s in the table are less. If numbers of 8s are much more than the other numbers in the table then the 4-bit crypto S-box has been said to be more linear cryptanalysis immune [HH96][HH02].

In another look an input S-box can be decomposed into four 4-bit Input Vectors (IPVs) with Decimal Equivalents 255 for 4th IPV, 3855 for 3rd IPV, 13107 for 2nd IPV, and 21845 for 1st IPV respectively. The S-box can also be decomposed into 4, 4-bit Output BFs (OPBFs). Each IPV can be denoted as a input variable of a linear relation and OPBF as a output variable and '+' as xor operation. Linear relations have been checked for satisfaction and 16-bit output variables (OPVs) due to linear relations have been checked for balanced-ness. Balanced OPVs indicates, out of 16 bits of IPVs and OPBFs, 8 bits satisfies the linear relation and 8 bits is out of satisfaction, i.e. best uncertainty. 256 4-bit linear relations have been operated on 4, 16-bit IPVs and 4, 16-bit OPBFs and 256 OPVs have been generated. The count of number of 1s in OPVs have been put in Linear Approximation Table or LAT. Better the number of 8s in LAT, better the S-box security[HH96][HH02]. The concept has been reviewed in brief in subsec. 3.6. of sec.3.

In this paper a detailed review of a new technique to find the existing Linear Relations or Linear Approximations for a particular 4-bit S-box has been reviewed. If the nonlinear part of the ANF equation of a 4-bit output BF is absent or calculated to be 0 then the equation is termed as a Linear Relation or Approximation. Searching for number of existing linear relations through this method is ended up with number of existing linear relations. I.e. the goal to conclude the security of a 4-bit crypto S-box has been attended in a very lucid manner by this method. The method has been reviewed in subsec.3.7. of sec.3.

Polynomials over Finite field or Galois field $GF(p^q)$ have been of utmost importance in Public Key Cryptography [BS96]. The polynomials over Galois field $GF(p^q)$ with degree q have been termed as Basic Polynomials or BPs over Galois field $GF(p^q)$ and Polynomials with degree <q have been termed as Elemental Polynomials or EPs over Galois field $GF(p^q)$ [SJ15]. The

EPs over Galois field $GF(p^q)$ with only constant terms have been termed as Constant Polynomials or CPs over Galois field $GF(p^q)$. The BPs over Finite field or Galois field $GF(p^q)$ that cannot be factored into at least two non-constant EPs have been termed as Irreducible polynomials or IPs over Finite field or Galois field $GF(p^q)$ and the rest have been termed as Reducible polynomials or RPs over Finite field or Galois field $GF(p^q)$ [SJ15]. The polynomials over Galois field $GF(p^q)$ with coefficient of the highest degree term as 1 have been termed as monic polynomials over Galois field $GF(p^q)$ and rest have been termed as non-monic Polynomials over Galois field $GF(p^q)$ [SJ15].

q bit crypto Substitution box or S-box have 2^q elements in an array where each element is unique and distinct and arranged in a random fashion varies from 0 to 2^q. Polynomials over Galois field $GF(p^q)$ have been termed as binary polynomials if p = 2. The binary number that has been constructed with binary coefficients of all q values with q = 0 at LSB and q = q at MSB has been termed as binary Coefficient Number or BCN of q+1 bits. The Binary Coefficient Number or BCN over Galois field $GF(p^q)$ has been similar with $\log_2{}^{q+1}$ bit BFs. The $\log_2{}^{q+1}$ bit S-boxes have been generated using $\log_2{}^{q+1}$ bit BCNs. In this paper crypto 4 and 8 bit S-boxes have been generated using BCNs and the procedure has been continued as a future scope to generate 16 and 32 bit S-boxes. The non-repeated coefficients of BPs over Galois field $GF(p^q)$, where $p = 2^{(\log_2{}^{q+1})}$ and q = p-1 have been used to generate $\log_2{}^{q+1}$ bit S-boxes. In this paper proper 4 and 8 bit S-boxes have been generated using BCNs and the procedure has been continued as a future scope to generate 16 and 32 bit S-boxes. In this paper polynomials over Galois Field $GF(p^q)$ and roll of IPs to construct substitution boxes have been reviewed in subsec. 4.1 and respectively of section.4. The generation of 4 and 8 bit S-boxes using BCNs have been elaborated in subsec 4.2 of section 4.. The generation of 4-bit and 8-bit S-boxes with coefficients of non-binary Galois Field polynomials has been depicted in subsec.4.3 of section 4. The cryptographic and security analysis of 32 DES 4-bit S-boxes has been given in subsec.4.4 of sec.4. Detailed cryptographic and security analysis of generated 10 4-bit S-boxes with discussed crypto related cryptographic properties and security criterion have also been given in subsec.4.4. of sec.4. Results have been discussed in Result and Discussion section in subsec.4.5 of sec.4.

Concluding remarks, Acknowledgement and Reference has been given in section 5, 6 and 7 respectively.

2. Literature Survey. In this section an exhaustive relevant literature survey with their specific references has been introduced to crypto literature on IPs and primitive polynomials. In early Twentieth Century Radolf Church initiated the search for irreducible polynomials over Galois Field $GF(p^q)$ for p = 2, 3, 5 and 7 and for p = 2, q = 1 through 11, for p =3, q = 1 through 7, for p = 5, q = 1 through 4 and for p = 7, q = 1 through 3 respectively. A manual polynomial multiplication among respected EPs gives RPs in the said Galois field. All RPs have been cancelled from the list of BPs to give IPs over the said Galois field $GF(p^q)$ [RC35]. Later The necessary condition for a BP to be an IPs had been generalized to Even 2 characteristics. It had also been applied to RPs and gives Irreducible factors mod 2 [RS62]. Next to it Elementary Techniques to compute over finite Fields or Galois Field $GF(p^q)$ had been descried with proper modifications [TD63]. In next the factorization of Polynomials over Galois Field $GF(p^q)$ had been elaborated [EB67]. Later appropriate coding techniques of Polynomials over Galois Field $GF(p^q)$ had been illustrated with example [TK68]. The previous idea of factorizing Polynomials over Galois Field $GF(p^q)$ [EB67] had also been extended to Large value of P or Large Finite fields [EB70]. Later Few Probabilistic Algorithms to find IPs over Galois Field $GF(p^q)$ for degree q had been elaborated with example [MR80]. Later Factorization of multivariate polynomials over Galois fields GF(p) had also been introduced to mathematics community [AL85]. With that the separation of irreducible factors of BPs [EB67] had also been introduced later [RM87]. Next to it the factorization of BPs with Generalized Reimann Hypothesis (GRH) had also been elaborated [LR88]. Later a Probabilistic Algorithm to find irreducible factors of Basic bivariate Polynomials over Galois Field $GF(p^q)$ had also been illustrated [DW90]. Later the conjectural Deterministic algorithm to find primitive elements and relevant primitive polynomials over binary Galois Field GF(2) had been introduced [MR90]. Some new algorithms to find IPs over Galois Field GF(p) had also been introduced at the same time [VS90]. Another use of Generalized Reimann Hypothesis (GRH) to determine irreducible factors in a deterministic manner and also for multiplicative subgroups had been introduced later [LR92]. The table binary equivalents of binary primitive polynomials had been illustrated in literature [MZ94]. The method to find roots of primitive polynomials over binary Galois field GF(2) had been introduced to mathematical community [IS96]. A method to search for IPs in a Random manner and factorization of BPs or to find irreducible factors of BPs in a random fashion had been introduced later [PX96]. After that a new variant of Rabin's algorithm [MR80] had been introduced with probabilistic analysis of BPs with no irreducible factors [GP97]. Later a factorization of univariate Polynomials over Galois Field GF(p) in sub quadratic execution time had also been notified [EV98]. Later a deterministic algorithm to factorize IPs over one variable had also been introduced [EJ01]. An algorithm to factorize bivariate polynomials over Galois Field GF(p) with hensel lifting had also been notified [GA02]. Next to it an algorithm had also been introduced to find factor of Irreducible and almost primitive polynomials over Galois Field GF(2) [BZ03]. Later a deterministic algorithm to factorize polynomials over Galois Field GF(p) to distinct degree factors had also been notified [SE04]. A detailed study of multiples and products of univariate primitive polynomials over binary Galois Field GF(2) had also been done [SM05]. Later algorithm to find optimal IPs over extended binary Galois Field $GF(2^m)$ [MS07] and a deterministic algorithm to determine Pascal Polynomials over Galois Field GF(2) [CF08] had been added to literature. Later the search of IPs and primitive polynomials over binary Galois Field GF(2) had also been done successfully [AA09]. At the same time the square free polynomials had also been factorized [CR09] where a work on divisibility of trinomials by IPs over binary Galois Field GF(2) [RW09] had also been notified. Later a probabilistic algorithm to factor polynomials over finite fields had been introduced [SM11]. An explicit factorization to obtain irreducible factors to obtain for cyclotomic polynomials over Galois Field $GF(p^q)$ had also been reported later [LQ12]. A fast randomized algorithm to obtain IPs over a certain Galois Field $GF(p^q)$ had been notified [JC13]. A deterministic algorithm to obtain factors of a polynomial over Galois field $GF(p^q)$ had also been notified at the same time [DM14]. A review of construction of IPs over finite fields and algorithms to Factor polynomials over finite fields had been reported to literature [GH14][NC14]. An algorithm to search for primitive polynomials had also been notified at the same time [WJ14]. The residue of division of BPs by IPs must be 1 and this reported to literature a bit later [SJ15]. The IPs with several coefficients of different categories had been illustrated in literature a bit later [HJ16]. The use of zeta function to factor polynomials over finite fields had been notified later on [BP17] At last Integer polynomials had also been described with examples [EWNN].

3. Review of crypto relevant properties of 4-bit and 8-bit Crypto S-boxes. In this section crypto relevant property of 4-bit BFs as well as 4-bit S-boxes has been reviewed. The subsec.3.1 has been dedicated to (Output) Bit Independence Criterion. In subsec. 3.2. Strict Avalanche Criterion (SAC) of 4-bit BFs and 4-bit S-boxes with new methods has been reviewed. The Higher order SAC or HO-SAC has been elaborated in subsec.3.3. A review of Differential Cryptanalysis of 4-bit S-boxes, Differential Cryptanalysis of 4-bit S-boxes with 4-bit BFs, Linear Cryptanalysis of 4-bit S-boxes and Linear Cryptanalysis of 4-bit S-boxes with 4-bit BFs or Linear Approximation Analysis has been reviewed in subsec.3.4, subsec.3.5, subsec.3.6, and subsec.3.7 respectively.

3.1 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.3.1.1 of sec 3. 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 have been illustrated in subsec.3.2.2.of sec.3. The (Output) Bit Independence Criterion (BIC) of 4-bit S-box has been described with example and Pseudo code in subsec.3.3. of sec.3.

3.1.1 4-bit Crypto S-boxes. A 4-bit Crypto S-box can be written as follows in Table.1, where the each element of the first row of Table.1, entitled as index, have been the position of each element of the crypto S-box within the given crypto S-box and the elements of the 2nd row, and entitled as S-box have been the elements of the given Substitution box. It can be concluded that the 1st row is fixed for all possible 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 [AT90][NT77][NT99].

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

Table.1. 4-bit crypto S-box.

3.1.2 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 has been a hexadecimal number and that can be converted into a 4-bit bit sequence that have been 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.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 is a 16 bit long input vectors (IPVs) and 4-bit output BFs (OPBFs) respectively. column 1 through G of Row 2 has been termed as 4th IPV, Row 3 has been termed as 3rd IPV, Row 4 has been termed as 2nd IPV and Row 5 has been termed as 1st IPV whereas column 1 through G of Row 7 has been termed as 4th OPBF, Row 8 has been termed as 3rd OPBF, Row 9 has been termed as 2nd OPBF and Row A has been termed as 1st OPBF [AT90]. The decimal equivalent of each IPV and OPBF has been noted at column H of respective rows.

Row	**Column**	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	**H.** Decimal
1	**Index**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**	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.2. Decomposition of 4-bit input S-box and given S-box (1st 4-bit S-box of 1st S-box out of 8 of DES) to 4-bit BFs.

3.1.3. (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) have been balanced for a particular 4-bit crypto S-box or 30 xored 8-bit DBFs have been balanced for a particular 8-bit crypto-S-box then the said 4-bit or 8-bit S-box has been said to satisfy output BIC of S-boxes [ST86][AT90]. The example of BIC of 4-bit S-boxes has been given in Table.3. below and Pseudo code with time complexity analysis have been given in subsec. 3.1.3.1 ,

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

In Table.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 has been each of four OPBFs, OPBF4, OPBF3, OPBF2, OPBF1 respectively. Column 1 through G of each row 6 through B has been each of six DBFs, DBF4,3, DBF4,2, DBF4,1, DBF3,2, DBF3,1 and DBF2,1 respectively. The analysis shows that 6 DBFs have been 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 has been said to satisfy (Output) Bit Independence Criterion of 4-bit S-boxes.

3.1.3.1. 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 j = 3: (i+1) // 2nd OPBF selection
              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.
        Enf for.
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.

3.2. A brief Review on Strict Avalanche Criterion (SAC) of 4-bit BFs and SAC of 4-bit S-boxes [AT90][CA90]. The Strict Avalanche Criterion or SAC of 4-bit BFs with pseudo code have been reviewed in sec. 3.2.1.and the new technique to find SAC of 4-bit BFs with pseudo code has been described in sec.3.2.2. and another technique of SAC of 4-bit BFs and SAC of 4-bit S-boxes with pseudo code has also been reviewed in sec.3.2.3

3.2.1. A brief Review on Strict Avalanche Criterion (SAC) of 4-bit BFs. A 4-bit BF has been said to satisfy SAC of 4-bit BFs if distances of OPBF from the complemented OPBFs (COPBFs) after complementation of four IPVs individually have been balanced. In Strict Avalanche Criterion of 4-bit BFs, IPV4, IPV3, IPV2 and IPV1 that have been shown in column 1 through G of row 2, 3, 4 and 5 in Table.2, have been complemented individually one at a time. If due to said operation on OPBF the number of bits has been changed for change of bits in each IPV in COPBF has been 8 or half of the number of bits in a 4-bit BF then the OPBF has been said to satisfy SAC of 4-bit BFs.

IPV4, CIPV4, IPV3, CIPV3, IPV2, CIPV2, IPV1, CIPV1 have been shown in column 2 thorough H of row 1, 3, 7, 9 , D, F, J, L respectively of table.4. The OPBFs and COPBFs due to complementation of CIPV4, CIPV3, CIPV2 and CIPV1 have been shown 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 have been shown in column 2 thorough H of row 5, B, H, N respectively. Now change in Number of bits in COPBFs from OPBF due to change of bits in CIPV4, CIPV3, CIPV2 and CIPV1 have been 12, 8, 4, 12. So the given OPBF does not satisfy SAC of 4-bit BFs. To Satisfy SAC of 4-bit BFs change in Number of bits in four COPBFs from OPBFs due to change of bits in CIPV4, CIPV3, CIPV2 and CIPV1 must be 8, 8, 8, 8. If four OPBFs of a particular crypto S-box satisfy SAC of 4-bit BFs individually then the said crypto S-box has been said to satisfy SAC of 4-bit crypto S-boxes.

Pseudo Code. Let BF[16].bit0 has been a bit level array of 16 bits of a 4-bit BF out of 65536 4-bit BFs. and BF[16] has been an array of 16 bits of a 4-bit BF. CV[16].bit0 has been a bit level array of 16 bits to store either 00FF, 0F0F, 3333, 5555 in hex. CVC[16].bit0 has been a bit level array of 16 bits to store either FF00, F0F0, CCCC, AAAA in hex. Here ^ represents Bitwise Xor operation. NL represents Number of bits changed in Lower Halves and NU represents Number 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.
```

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

3.2.2. New Method for Strict Avalanche Criterion (SAC) of 4-bit BFs. Since complement of 4th IPV means interchanging each distinct 8 bit halves of 16 bit long 4th IPV so the 2, 8 bit halves of OPBF has been interchanged due to complement of 4th IPV or CIPV4 in COPBF. Next to it since complement of 3rd IPV means interchanging each distinct 4 bit halves of each distinct 8 bit halves of IPV3 in CIPV3 so each distinct 4 bit halves of each distinct 8 bit halves of OPBF have been interchanged due to complement of IPV3 in COPBF. Now complement of 2nd IPV means interchanging each distinct 2 bit halves of each distinct 4 bit halves of each distinct 8 bit halves of OPBF in COPBF and complement of 1st IPV means interchanging each bit of each distinct 2 bit halves of 16 bit long OPBF in COPBF.

IPV4, CIPV4, IPV3, CIPV3, IPV2, CIPV2, IPV1, CIPV1 have been shown in column 2 thorough H of row 1, 3, 7, 9 , D, F, J, L respectively of table.4. The OPBFs and COPBFs due to complementation of CIPV4, CIPV3, CIPV2 and CIPV1 have been shown 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 have been shown in column 2 thorough H of row 5, B, H, N respectively. Now change in Number of bits in COPBFs from OPBF due to change of bits in CIPV4, CIPV3, CIPV2 and CIPV1 have been 12, 8, 4, 12. So the given OPBF does not satisfy SAC of 4-bit BFs. To Satisfy SAC of 4-bit BFs change in Number of bits in COPBFs from OPBFs due to change of bits in CIPV4, CIPV3, CIPV2 and CIPV1 must be 8, 8, 8, 8. If four OPBFs of a particular crypto S-box satisfy SAC of 4-bit BFs individually then the said crypto S-box has been said to satisfy SAC of 4-bit crypto S-boxes.

Pseudo Code.

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

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

3.2.3. Review of a new method of Strict Avalanche Criterion (SAC) of 4-bit BFs and 4-bit S-boxes [AT90][ST86]. Each and every column of 1 through G from row 2 through 5 and row 7 through A of 4 IPVs and 4 OPBFs in Table.2 constitutes 16 4-bit input binary numbers and 16 4-bit output binary numbers respectively. The binary value of OPBF before and after flip of 16 input binary numbers sequentially in same position 1 bit at a time has been equated. If they are equal 8 times [discarding same occurrences] and out of equality 8 times for flip of 1 bit at a time in the same position of 16 Input binary numbers sequentially for four positions one by one then the 4-bit BF has been said to satisfy SAC of 4-bit BFs.

All 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) have been given in column 2 through H of row 3, 1, 2 of Table.5 respectively. Each Output BF, OPBF1, OPBF2, OPBF3, OPBF4 has been shown in column 2 through H of row 4, 5, 6, 7 Table.5 respectively.

Now 16 INBs before flip and 16 INBs after flip in one bits particularly in bit position 1, 2, 3, 4 have been shown in row 2 through H of column 1, 2, 6, 7, B, C, G, H respectively of Table.6. The each corresponding bits of OPBF1, OPBF2, OPBF3, OPBF4 before and after flip have been shown in row 2 through H of column 3, 4, 8, 9, D, E, I, J respectively in Table.6. 1 in any position in row 2 through H of column 5, A, F, K illustrate dissimilarity in bits in corresponding positions of OPBF1, OPBF2, OPBF3 and OPBF4 duly before and after flip in one bits in 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 SAC of 4-bit BFs. If all four BFs of a given 4-bit crypto S-box satisfy SAC of 4-bit BFs then the S-box has been said to satisfy SAC of 4-bit S-boxes. Here in Table.6. row I shows the number 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 in at least one position for the given OPBF so the concerned OPBF and the given 4-bit S-box does not satisfy SAC of 4-bit BFs and SAC of 4-bit S-boxes respectively.

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.4. 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	**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.5. S-box and OBFs for SAC Test of 4-bit BFs as well as 4-bit Crypto 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	**B-Flip**	**A-Flip**	**1**	**1’**	**C**	**B-Flip**	**A-Flip**	**2**	**2’**	**C**	**B-Flip**	**A-Flip**	**3**	**3’**	**C**	**B-Flip**	**A-Flip**	**4**	**4’**	**C**
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				

Table.6. SAC Test of 4-bit BFs and 4-Bit Crypto S-boxes.

Pseudo Code. The flipping of bits on particular positions are made by proposing 1-bit in four e_v vectors as, e_0 {0001}, e_1 {0010}, e_2 {0100} and e_3 {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 e_v vextor
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. A brief Review of old and new methods of Higher Order SAC (HO-SAC) and Extended SAC Criterion of 4-bit Crypto S-boxes [CF90].

The Method and algorithm of HO-SAC of 4-bit BFs has been reviewed in subsec 3.3.1. The two new methods and algorithms of HO-SAC of 4-bit BFs and HO-SAC of 4-bit crypto S-boxes have been described in subsec 3.3.2. The new SAC criterion entitled Extended SAC criterion of 4-bit BFs and 4-bit S-boxes has been illustrated with its review and algorithm in subsec. 3.3.3.

3.3.1 Review of Higher Order SAC of 4-bit BFs. If two or three IPVs have been complemented at a time and the Difference BF of complemented OPBFs and OPBF have been balanced for all possible 10 conditions then the OPBF has been said to satisfy HO-SAC of 4-bit BFs. If four OPBF of particular crypto S-box satisfy HO-SAC of 4-bit BFs individually then the concerned S-box has been said to satisfy HO-SAC of 4-bit crypto S-boxes. The Given S-box and INB with position of each bits of it and a review of HO-SAC of 4-bit BFs have been illustrated in Table. 7. and Table.8. respectively.

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	**OPBF1**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
5	**OPBF2**	1	1	1	0	0	1	0	0	0	0	1	1	1	0	0	1
6	**OPBF3**	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	1
7	**OPBF4**	0	0	1	1	0	1	1	0	1	0	0	0	1	1	0	1

Table.7. S-box and OBFs for HO-SAC Test of 4-bit BFs as well as 4-bit crypto S-boxes.

R|C	**8-A**	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	**IPV3**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1

3	**OPBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
4	**CIPV4**	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
5	**CIPV3**	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
6	**Step 1**	0	1	0	1	0	1	0	0	1	0	1	0	0	1	1	1
7	**Step 2**	0	1	0	0	0	1	0	1	0	1	1	1	1	0	1	0
8	**COPBF**	0	1	0	0	0	1	0	1	0	1	1	1	1	0	1	0
9	**DBF**	1	1	1	0	0	0	1	0	0	0	1	0	1	0	1	0
A	Number of bits changed in COPBF												6				

R|C	**8-B**	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	**IPV2**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
3	**OPBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
4	**CIPV4**	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
5	**CIPV2**	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
6	**Step 1**	0	1	0	1	0	1	0	0	1	0	1	0	0	1	1	1
7	**Step 2**	0	1	0	1	0	0	0	1	1	0	1	0	1	1	0	1
8	**COPBF**	0	1	0	1	0	0	0	1	1	0	1	0	1	1	0	1
9	**DBF**	1	1	1	1	0	1	1	0	1	1	1	1	1	0	0	1
A	Number of bits changed in COPBF												12				

R|C	**8-C**	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	**IPV1**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
3	**OPBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
4	**CIPV4**	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
5	**CIPV1**	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
6	**Step 1**	0	1	0	1	0	1	0	0	1	0	1	0	0	1	1	1
7	**Step 2**	1	0	1	0	1	0	0	0	0	1	0	1	1	0	1	1
8	**COPBF**	1	0	1	0	1	0	0	0	0	1	0	1	1	0	1	1
9	**DBF**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
A	Number of bits changed in COPBF												8				

R|C	**8-D**	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
1	**IPV3**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
2	**IPV1**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
3	**OPBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
4	**CIPV3**	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
5	**CIPV1**	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
6	**Step 1**	0	1	1	1	1	0	1	0	0	1	0	0	0	1	0	1
7	**Step 2**	1	0	1	1	0	1	0	1	1	0	0	0	1	0	1	0
8	**COPBF**	1	0	1	1	0	1	0	1	1	0	0	0	1	0	1	0
9	**DBF**	0	0	0	1	0	0	1	0	1	1	0	1	1	1	1	0
A	Number of bits changed in COPBF												8				

R|C	**8-E**	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
1	**IPV2**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
2	**IPV1**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
3	**OPBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
4	**CIPV2**	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
5	**CIPV1**	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
6	**Step 1**	1	0	1	0	1	1	0	1	0	1	0	1	0	0	0	1
7	**Step 2**	0	1	0	1	1	1	1	0	1	0	1	0	0	0	1	0
8	**COPBF**	0	1	0	1	1	1	1	0	1	0	1	0	0	0	1	0
9	**DBF**	1	1	1	1	1	0	0	1	1	1	1	1	0	1	1	0

A	Number of bits changed in COPBF												12				

R|C	**8-F**	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
1	**IPV3**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
2	**IPV2**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
3	**OPBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
4	**CIPV3**	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
5	**CIPV2**	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
6	**Step 1**	0	1	1	1	1	0	1	0	0	1	0	0	0	1	0	1
7	**Step 2**	1	1	0	1	1	0	1	0	0	0	0	1	0	1	0	1
8	**COPBF**	1	1	0	1	1	0	1	0	0	0	0	1	0	1	0	1
9	**DBF**	0	1	1	1	1	1	0	1	0	1	0	0	0	0	0	1
A	Number of bits changed in COPBF												8				

R|C	**8-G**	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	**IPV3**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
3	**IPV2**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
4	**OPBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
5	**CIPV4**	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
6	**CIPV3**	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
7	**CIPV2**	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
8	**Step 1**	0	1	0	1	0	1	0	0	1	0	1	0	0	1	1	1
9	**Step 2**	0	1	0	0	0	1	0	1	0	1	1	1	1	0	1	0
A	**Step 3**	0	0	0	1	0	1	0	1	1	1	0	1	1	0	1	0
B	**COPBF**	0	0	0	1	0	1	0	1	1	1	0	1	1	0	1	0
C	**DBF**	1	0	1	1	0	0	1	0	1	0	0	0	1	1	1	0
D	Number of bits changed in COPBF												8				

R|C	**8-H**	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	**IPV3**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
3	**IPV1**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
4	**OPBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
5	**CIPV4**	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
6	**CIPV3**	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
7	**CIPV1**	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
8	**Step 1**	0	1	0	1	0	1	0	0	1	0	1	0	0	1	1	1
9	**Step 2**	0	1	0	0	0	1	0	1	0	1	1	1	1	0	1	0
A	**Step 3**	1	0	0	0	1	0	1	0	1	0	1	1	0	1	0	1
B	**COPBF**	1	0	0	0	1	0	1	0	1	0	1	1	0	1	0	1
C	**DBF**	0	0	1	0	1	1	0	1	1	1	1	0	0	0	0	1
D	Number of bits changed in COPBF												8				

R|C	**8-I**	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	**IPV2**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
3	**IPV1**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
4	**OPBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
5	**CIPV4**	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
6	**CIPV2**	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
7	**CIPV1**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
8	**Step 1**	0	1	0	1	0	1	0	0	1	0	1	0	0	1	1	1

9	**Step 2**	0	1	0	1	0	0	0	1	1	0	1	0	1	1	0	1
A	**Step 3**	1	0	1	0	0	0	1	0	0	1	0	1	1	1	1	0
B	**COPBF**	1	0	1	0	0	0	1	0	0	1	0	1	1	1	1	0
C	**DBF**	0	0	0	0	0	1	0	1	0	0	0	0	1	0	1	0
D	Number of bits changed in COPBF											4					

R|C	**8-J**	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
1	**IPV3**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
2	**IPV2**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
3	**IPV1**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
4	**OPBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
5	**CIPV3**	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
6	**CIPV2**	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
7	**CIPV1**	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
8	**Step 1**	0	1	1	1	1	0	1	0	0	1	0	0	0	1	0	1
9	**Step 2**	1	1	0	1	1	0	1	0	0	0	0	1	0	1	0	1
A	**Step 3**	1	1	1	0	0	1	0	1	0	0	1	0	1	0	1	0
B	**COPBF**	1	1	1	0	0	1	0	1	0	0	1	0	1	0	1	0
C	**DBF**	0	1	0	0	0	0	1	0	0	1	1	1	1	1	1	0
D	Number of bits changed in COPBF											8					

Table.8. Table of HO-SAC Test and Complement Method of HO-SAC Test of 4-bit BFs

All elements of the given S-box, Index of each element of the given S-box in hex (INH) and 4 bit binary form (INB) with position of each bit from 1 to 4 have been given in column 2 through H of row 3, 1, 2 of Table.7 respectively. Each Output BF, OPBF1, OPBF2, OPBF3, OPBF4 has been shown in column 2 through H of row 4, 5, 6, 7 Table.7 respectively.

If 2 IPVs have been complemented at a time then the Higher Order SAC or HO-SAC of 4-bit BFs can be termed as 2nd Order HO-SAC of 4-bit BFs illustrated in sub table 8-A to 8-F. If the Number of IPVs complemented at a time is 3 then the Higher Order SAC or HO-SAC of 4-bit BFs can be termed as 3rd Order HO-SAC of 4-bit BFs illustrated in sub table 8-G to 8-J.

In 2nd Order HO-SAC, any 2 IPVs shown in column 2 through H of row 1, 2 of sub table 8-A to 8-F of Table.8 have been complemented at a time. The complemented IPVs or CIPVs have been shown in column 2 through H of row 4, 5 of sub table 8-A to 8-F of Table.8.The OPBF has been shown in column 2 through H of row 3 of sub table 8-A to 8-F of Table.8. Now due to complement of 1st IPV the resultant complemented OPBF has been shown in row column 2 through H of row 6 and due to complement of 2nd IPV at a time the resultant OPBF of the complemented OPBF have been shown in column 2 through H of row 7 of sub table 8-A to 8-F of Table.8 respectively. The obtained complemented OPBF or COPBF and bitwise xor or Hamming distance (Difference BF or DBF) between OPBF and COPBF have been shown in column 2 through H of row 8, 9 of sub table 8-A to 8-F of Table.8 respectively. The count of 1s in DBF or dissimilar bits in OPBF and COPBF due to complement of two IPVs at a time have been shown in sub table 8-A to 8-F of Table.8. For the given OPBF and for all 6 possible 2nd order HO-SAC tests the counts of 1s in DBF have been shown in row A of sub table 8-A to 8-F of Table.8 have been 8 then the given OPBF has been said to satisfy 2nd order HO-SAC of 4-bit BFs. But in table 8 all counts have not been equal to 8 so the given OPBF does not satisfy 2nd order HO-SAC of 4-bit BFs.

In 3rd Order HO-SAC, any 3 IPVs shown in column 2 through H of row 1, 2, 3 of sub table 8-G to 8-J of Table.8 have been complemented at a time. The complemented IPVs or CIPVs have been shown in column 2 through H of row 5, 6, 7 of sub table 8-G to 8-J of Table.8.The OPBF has been shown in column 2 through H of row 4 of sub table 8-G to 8-J of Table.8. Now due to complement of 1st IPV the resultant complemented OPBF has been shown in row column 2 through H of row 8 and due to complement of 2nd IPV at a time the resultant OPBF of the complemented OPBF have been shown in column 2 through H of row 9 and for 3rd IPV COPBF has been shown in column 2 through H of row A of sub table 8-G to 8-J of Table.8 respectively. The obtained complemented OPBF or COPBF and bitwise xor or Hamming distance (Difference BF or DBF) between OPBF and COPBF have been shown in column 2 through H of row B, C of sub table 8-G to 8-J of Table.8 respectively. The count of 1s in DBF or dissimilar bits in OPBF and COPBF due to complement of three IPVs at a time have been shown in sub table 8-G to 8-J of Table.8. For the given OPBF and for all 4 possible 3rd order HO-SAC tests the counts of 1s in DBF have been shown in row D of sub table 8-G to 8-J of Table.8 have been 8 then the given OPBF has been said to satisfy 3rd order HO-SAC of 4-bit BFs. But in table 8 all counts have not been equal to 8 so the given OPBF does not satisfy 3rd order HO-SAC of 4-bit BFs.

If the given OPBF satisfy both 2nd order HO-SAC and 3rd Order HO-SAC for 4-bit BFs together then the Given OPBF has been said to satisfy total HO-SAC for 4-bit BFs. If Four BFs of a crypto 4-bit S-box satisfy total HO-SAC of 4bit BFs individually then the S-box has been said to satisfy HO-SAC of 4-bit Crypto S-boxes.

Pseudo code. Let BF[16].bit0 has been a bit level array of 16 bits of a 4-bit BF out of 65536 4-bit BFs. and BF[16] has been an array of 16 bits of a 4-bit BF. CV[16].bit0 has been a bit level array of 16 bits to store either 00FF, 0F0F, 3333, 5555 in hex. CVC[16].bit0 has been a bit level array of 16 bits to store either FF00, F0F0, CCCC, AAAA in hex. Here ^ represents Bitwise Xor operation. NL represents Number of bits changed in lower halves and NU represents Number of bits changed in upper halves.

```
Start.
//Initialization of Variables.
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 10 steps have been evaluated to obtain 10 complemented COPBFs and weight of DBFs
Step 01: wt[{(BF[16].bit0 & 00FF)^(BF[16].bit0>>8&00FF)}+WT{(BF[16].bit0&FF00)^(BF[16].bit0>>8&FF00)}
             ^{(BF[16].bit0 & 0F0F)^(BF[16].bit0>>4&0F0F)}+WT{(BF[16].bit0&F0F0)^(BF[16].bit0>>4&F0F0)}]      = N.
Step 02: wt[{(BF[16].bit0 & 00FF)^(BF[16].bit0>>8&00FF)}+WT{(BF[16].bit0&FF00)^(BF[16].bit0>>8&FF00)}
             ^{(BF[16].bit0 & 3333)^(BF[16].bit0>>2&3333)}+WT{(BF[16].bit0&CCCC)^(BF[16].bit0>>2&CCCC)}]   = N .
Step 03: wt[{(BF[16].bit0 & 00FF)^(BF[16].bit0>>8&00FF)}+WT{(BF[16].bit0&FF00)^(BF[16].bit0>>8&FF00)}
             ^{(BF[16].bit0 & 5555)^(BF[16].bit0>>1&5555)}+WT{(BF[16].bit0&AAAA)^(BF[16].bit0>>1&AAAA)}] = N.
Step 04: wt[{(BF[16].bit0 & 0F0F)^(BF[16].bit0>>4&0F0F)}+WT{(BF[16].bit0&F0F0)^(BF[16].bit0>>4&F0F0)}
             ^{(BF[16].bit0 & 3333)^(BF[16].bit0>>2&3333)}+WT{(BF[16].bit0&CCCC)^(BF[16].bit0>>2&CCCC)}]  = N
Step 05: wt[{(BF[16].bit0 & 0F0F)^(BF[16].bit0>>4&0F0F)}+WT{(BF[16].bit0&F0F0)^(BF[16].bit0>>4&F0F0)}
             ^{(BF[16].bit0 & 5555)^(BF[16].bit0>>1&5555)}+WT{(BF[16].bit0&AAAA)^(BF[16].bit0>>1&AAAA)}  = N.
Step 06: wt[{(BF[16].bit0 & 3333)^(BF[16].bit0>>2&3333)}+WT{(BF[16].bit0&CCCC)^(BF[16].bit0>>2&CCCC)}
             ^{(BF[16].bit0 & 5555)^(BF[16].bit0>>1&5555)}+WT{(BF[16].bit0&AAAA)^(BF[16].bit0>>1&AAAA)}] = N.
Step 07: wt[{(BF[16].bit0 & 00FF)^(BF[16].bit0>>8&00FF)}+WT{(BF[16].bit0&FF00)^(BF[16].bit0>>8&FF00)}
             ^{(BF[16].bit0 & 0F0F)^(BF[16].bit0>>4&0F0F)}+WT{(BF[16].bit0&F0F0)^(BF[16].bit0>>4&F0F0)}
             ^{(BF[16].bit0 & 3333)^(BF[16].bit0>>2&3333)}+WT{(BF[16].bit0&CCCC)^(BF[16].bit0>>2&CCCC)}]   = N.
Step 08: wt[{(BF[16].bit0 & 00FF)^(BF[16].bit0>>8&00FF)}+WT{(BF[16].bit0&FF00)^(BF[16].bit0>>8&FF00)}
             ^{(BF[16].bit0 & 0F0F)^(BF[16].bit0>>4&0F0F)}+WT{(BF[16].bit0&F0F0)^(BF[16].bit0>>4&F0F0)}
             ^{(BF[16].bit0 & 5555)^(BF[16].bit0>>1&5555)}+WT{(BF[16].bit0&AAAA)^(BF[16].bit0>>1&AAAA)}] = N
Step 09: wt[{(BF[16].bit0 & 00FF)^(BF[16].bit0>>8&00FF)}+WT{(BF[16].bit0&FF00)^(BF[16].bit0>>8&FF00)}
              ^{(BF[16].bit0 & 3333)^(BF[16].bit0>>2&3333)}+WT{(BF[16].bit0&CCCC)^(BF[16].bit0>>2&CCCC)}
            ^{(BF[16].bit0 & 5555)^(BF[16].bit0>>1&5555)}+WT{(BF[16].bit0&AAAA)^(BF[16].bit0>>1&AAAA)}] = N.
Step 10: wt[{(BF[16].bit0 & 0F0F)^(BF[16].bit0>>4&0F0F)}+WT{(BF[16].bit0&F0F0)^(BF[16].bit0>>4&F0F0)}
             ^{(BF[16].bit0 & 3333)^(BF[16].bit0>>2&3333)}+WT{(BF[16].bit0&CCCC)^(BF[16].bit0>>2&CCCC)}
             ^{(BF[16].bit0 & 5555)^(BF[16].bit0>>1&5555)}+WT{(BF[16].bit0&AAAA)^(BF[16].bit0>>1&AAAA)}] = N.
// Test of HO-SAC criterion.
Step 11:  If N=8 for Step 01, to Step 15. Then BF[16].bit0 Satisfies HO-SAC of 4-bit BFs.
          ELSE BF[16].bit0 Does not Satisfies Extended HO-SAC of 4-bit BFs..
Stop.
```

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

3.3.2. Two new Methods of Higher Order SAC or HO-SAC of 4-bit BFs and 4-bit Crypto S-boxes. The Shift Method to do HO-SAC test of 4 bit BFs has been described in section 3.3.2.1. The Flip Method to do the same test of 4-bit BFs and of 4-bit S-boxes has been described in sec. 3.3.2.2.

3.3.2.1 Shift Method of HO-SAC of 4-bit BFs and 4-bit Crypto S-boxes. Complement of IPV4 has been equivalent of interchanging two distinct 8 bit halves of OPBF. Complement of IPV3 has been equivalent of interchanging each distinct 4 bit halves of each distinct 8 bit halves of the given OPBF. Now Complement of IPV2 means interchanging each distinct 2 bit halves of each distinct 4 bit halves of each distinct 8 bit halves of given OPBF and finally Complement of IPV1 means interchanging each two distinct bits of all distinct two bit halves of given OPBF.

If two IPVs have been complemented at a time then the Higher Order SAC or HO-SAC of 4-bit BFs can be termed as 2nd Order HO-SAC of 4-bit BFs illustrated in sub table 8-A to 8-F. If the Number of IPVs complemented at a time has been three then the Higher Order SAC or HO-SAC of 4-bit BFs can be termed as 3rd Order HO-SAC of 4-bit BFs illustrated in sub table 8-G to 8-J.

In 2nd Order HO-SAC, any 2 IPVs shown in column 2 through H of row 1, 2 of sub tables 8-A to 8-F of Table.8 have been complemented at a time. The complemented or shifted IPVs or CIPVs have been shown in column 2 through H of row 4, 5 of sub tables 8-A to 8-F of Table.8.The OPBF has been shown in column 2 through H of row 3 of sub tables 8-A to 8-F of Table.8. Now due to complement of 1st IPV the resultant OPBF have been shown in column 2 through H of row 6 after shift operation and due to complement of 2nd IPV at a time the complemented OPBF of the resultant OPBF of the first operation have been shown in column 2 through H of row 7 of sub tables 8-A to 8-F of Table.8 respectively. The obtained complemented OPBF or COPBF and bitwise xor or Hamming distance (Difference BF or DBF) between OPBF and COPBF have been shown in column 2 through H of row 8, 9 of sub tables 8-A to 8-F of Table.8 respectively. The count of 1s out of 16 bits in DBF or dissimilar bits in COPBF than OPBF due to complement of 2 IPVs at a time have been shown in row A of sub table 8-A to 8-F of Table.8. If for the given OPBF and for all 6 possible 2nd order HO-SAC tests the counts have been shown in A of sub table 8-A to 8-F of Table.8 have been 8 then the given OPBF has been said to satisfy 2nd order HO-SAC of 4-bit BFs. But in table 8 all counts have not been equal to 8 so the given OPBF does not satisfy 2nd order HO-SAC of 4-bit BFs in this case.

In 3rd Order HO-SAC, any 3 IPVs shown in column 2 through H of row 1, 2 and 3 of sub tables 8-G to 8-J of Table.8 have been complemented at a time. The complemented or shifted IPVs or CIPVs have been shown in column 2 through H of row

5, 6 and 7 of sub tables 8-G to 8-J of Table.8.The OPBF has been shown in column 2 through H of row 4 of sub tables 8-G to 8-J of Table.8. Now due to complement of 1st IPV the resultant OPBF have been shown in column 2 through H of row 8 after shift operation and due to complement of 2nd IPV at a time the complemented OPBF of the resultant OPBF of the first operation have been shown in column 2 through H of row 9 and the complemented OPBF of the resultant OPBF of the 2nd operation have been shown in column 2 through H of row A of sub tables 8-G to 8-J of Table.8 respectively. The obtained complemented OPBF or COPBF and bitwise xor or Hamming distance (Difference BF or DBF) between OPBF and COPBF have been shown in column 2 through H of row B, C of sub tables 8-G to 8-J of Table.8 respectively. The count of 1s out of 16 bits in DBF or dissimilar bits in COPBF than OPBF due to complement of 3 IPVs at a time have been shown in row D of sub table 8-G to 8-J of Table.8. If for the given OPBF and for all 6 possible 2nd order HO-SAC tests the counts have been shown in D of sub table 8-G to 8-J of Table.8 have been 8 then the given OPBF has been said to satisfy 3rd order HO-SAC of 4-bit BFs. But in table 8 all counts have not been equal to 8 so the given OPBF does not satisfy 3rd order HO-SAC of 4-bit BFs in this case.

If the given OPBF satisfy both 2nd order HO-SAC and 3rd Order HO-SAC for 4-bit BFs together then the OPBF has been said to satisfy Total HO-SAC for 4-bit BFs. If four 4-bit BFs of a Crypto 4-bit S-box satisfy total HO-SAC of 4bit BFs individually then the S-box has been said to satisfy HO-SAC of 4-bit crypto S-boxes.

```
Start.
Step 00:  For 1:16 BF[16].bit0 = BF[16].// Relocate the 16 bits of OPBF to bit level array BF[16].bit0.
Step 1A: CBF[16].bit0 = (BF[16].bit0>>8); // Circular shift of each distinct 8 bit halves of 16 bit long OPBFs.
Step 1B: CBF[16].bit0 = (CBF[8A].bit0>>4)&& (CBF[8B].bit0>>4); //Circular shift of each distinct 4-bit halves of 4-bit OPBFs.
Step 1C: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0; // Difference BF have been obtained by bitwise xor of OPBFs and COPBFs.
Step 1D: Count = IF(DBF[16].bit0==1);// Count Number of 1s in DBF.
Step 2A: CBF[16].bit0 = (BF[16].bit0>>8); // Circular shift of each distinct 8 bit halves of 16 bit long OPBFs.
// In next step Each distinct 2 bit halves of each distinct 4-bit Halves have been circularly shifted.
Step 2B: CBF[16].bit0 = (CBF[4A].bit0>>2)&& (CBF[4B].bit0>>2)&& (CBF[4C].bit0>>2)&& (CBF[4D].bit0>>2);
Step 2C: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0; // Difference BF have been obtained by bitwise xor of OPBFs and COPBFs.
Step 2D: Count = IF(DBF[16].bit0==1); // Count Number of 1s in DBF.
Step 3A: CBF[16].bit0 = (BF[16].bit0>>8); // Circular shift of each distinct 8 bit halves of 16 bit long OPBFs.
// In next step each bit of each distinct two bit halves has been circularly shifted.
Step 3B: CBF[16].bit0 = (CBF[2A].bit0>>1)&&(CBF[2B].bit0>>1)&&(CBF[2C].bit0>>1)&&(CBF[2D].bit0>>1)&&
                        (CBF[2E].bit0>>1)&&(CBF[2F].bit0>>1)&&(CBF[2G].bit0>>1)&&(CBF[2H].bit0>>1);
Step 3C: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0; // Difference BF have been obtained by bitwise xor of OPBFs and COPBFs.
Step 3D: Count = IF(DBF[16].bit0==1); // Count Number of 1s in DBF.
Step 4A: CBF[16].bit0 = (BF[8A].bit0>>4)&& (BF[8B].bit0>>4); //Circular shift of each distinct 4-bit halves of 4-bit OPBFs.
// In next step Each distinct 2 bit halves of each distinct 4-bit Halves have been circularly shifted.
Step 4B: CBF[16].bit0 = (CBF[4A].bit0>>2)&& (CBF[4B].bit0>>2)&& (CBF[4C].bit0>>2)&& (CBF[4D].bit0>>2);
Step 4C: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0; // Difference BF have been obtained by bitwise xor of OPBFs and COPBFs.
Step 4D: Count = IF(DBF[16].bit0==1); // Count Number of 1s in DBF.
Step 5A: CBF[16].bit0 = (BF[8A].bit0>>4)&& (BF[8B].bit0>>4); //Circular shift of each distinct 4-bit halves of 4-bit OPBFs.
// In next step each bit of each distinct two bit halves has been circularly shifted.
Step 5B: CBF[16].bit0 = (CBF[2A].bit0>>1)&&(CBF[2B].bit0>>1)&&(CBF[2C].bit0>>1)&&(CBF[2D].bit0>>1)&&
                        (CBF[2E].bit0>>1)&&(CBF[2F].bit0>>1)&&(CBF[2G].bit0>>1)&&(CBF[2H].bit0>>1);
Step 5C: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0; // Difference BF have been obtained by bitwise xor of OPBFs and COPBFs.
Step 5D: Count = IF(DBF[16].bit0==1); // Count Number of 1s in DBF.
// In next step Each distinct 2 bit halves of each distinct 4-bit Halves have been circularly shifted.
Step 6A: CBF[16].bit0 = (BF[4A].bit0>>2)&& (BF[4B].bit0>>2)&& (BF[4C].bit0>>2)&& (BF[4D].bit0>>2);
// In next step each bit of each distinct two bit halves has been circularly shifted.
Step 6B: CBF[16].bit0 = (CBF[2A].bit0>>1)&&(CBF[2B].bit0>>1)&&(CBF[2C].bit0>>1)&&(CBF[2D].bit0>>1)&&
                        (CBF[2E].bit0>>1)&&(CBF[2F].bit0>>1)&&(CBF[2G].bit0>>1)&&(CBF[2H].bit0>>1);
Step 6C: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0; // Difference BF have been obtained by bitwise xor of OPBFs and COPBFs.
Step 6D: Count = IF(DBF[16].bit0==1); // Count Number of 1s in DBF.
Step 7A: CBF[16].bit0 = (BF[16].bit0>>8); // Circular shift of each distinct 8 bit halves of 16 bit long OPBFs.
Step 7B: CBF[16].bit0 = (CBF[8A].bit0>>4)&& (CBF[8B].bit0>>4); //Circular shift of each distinct 4-bit halves of 4-bit OPBFs.
// In next step Each distinct 2 bit halves of each distinct 4-bit Halves have been circularly shifted.
Step 7C: CBF[16].bit0 = (CBF[4A].bit0>>2)&& (CBF[4B].bit0>>2)&& (CBF[4C].bit0>>2)&& (CBF[4D].bit0>>2);
Step 7D: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0; // Difference BF have been obtained by bitwise xor of OPBFs and COPBFs.
Step 7E: Count = IF(DBF[16].bit0==1); // Count Number of 1s in DBF.
Step 8A: CBF[16].bit0 = (BF[16].bit0>>8); // Circular shift of each distinct 8 bit halves of 16 bit long OPBFs.
Step 8B: CBF[16].bit0 = (CBF[8A].bit0>>4)&& (CBF[8B].bit0>>4); //Circular shift of each distinct 4-bit halves of 4-bit OPBFs.
// In next step each bit of each distinct two bit halves has been circularly shifted.
Step 8C: CBF[16].bit0 = (CBF[2A].bit0>>1)&&(CBF[2B].bit0>>1)&&(CBF[2C].bit0>>1)&&(CBF[2D].bit0>>1)&&
                        (CBF[2E].bit0>>1)&&(CBF[2F].bit0>>1)&&(CBF[2G].bit0>>1)&&(CBF[2H].bit0>>1);
Step 8D: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0; // Difference BF have been obtained by bitwise xor of OPBFs and COPBFs.
Step 8E: Count = IF(DBF[16].bit0==1); // Count Number of 1s in DBF.
Step 9A: CBF[16].bit0 = (BF[16].bit0>>8); // Circular shift of each distinct 8 bit halves of 16 bit long OPBFs.
// In next step Each distinct 2 bit halves of each distinct 4-bit Halves have been circularly shifted.
Step 9B: CBF[16].bit0 = (CBF[4A].bit0>>2)&& (CBF[4B].bit0>>2)&& (CBF[4C].bit0>>2)&& (CBF[4D].bit0>>2);
// In next step each bit of each distinct two bit halves has been circularly shifted.
Step 9C: CBF[16].bit0 = (CBF[2A].bit0>>1)&&(CBF[2B].bit0>>1)&&(CBF[2C].bit0>>1)&&(CBF[2D].bit0>>1)&&
                        (CBF[2E].bit0>>1)&&(CBF[2F].bit0>>1)&&(CBF[2G].bit0>>1)&&(CBF[2H].bit0>>1);
Step 9D: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0; // Difference BF have been obtained by bitwise xor of OPBFs and COPBFs.
Step 9E: Count = IF(DBF[16].bit0==1); // Count Number of 1s in DBF.
Step 10A: CBF[16].bit0 = (CBF[8A].bit0>>4)&& (CBF[8B].bit0>>4); //Circular shift of each distinct 4-bit halves of 4-bit OPBFs.
// In next step Each distinct 2 bit halves of each distinct 4-bit Halves have been circularly shifted.
Step 10B: CBF[16].bit0 = (CBF[4A].bit0>>2)&& (CBF[4B].bit0>>2)&& (CBF[4C].bit0>>2)&& (CBF[4D].bit0>>2);
// In next step each bit of each distinct two bit halves has been circularly shifted.
Step 10C: CBF[16].bit0 = (CBF[2A].bit0>>1)&&(CBF[2B].bit0>>1)&&(CBF[2C].bit0>>1)&&(CBF[2D].bit0>>1)&&
                         (CBF[2E].bit0>>1)&&(CBF[2F].bit0>>1)&&(CBF[2G].bit0>>1)&&(CBF[2H].bit0>>1);
Step 10D: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0; // Difference BF have been obtained by bitwise xor of OPBFs and COPBFs.
Step 10E: Count = IF(DBF[16].bit0==1); // Count Number of 1s in DBF.
// Evaluation of HO-SAC criterion.
Step 11 : IF Count = 8 for Step 1D, TO 6D and 7E to 10E. BF[16] Satisfies HO-SAC of 4-bit BFs.
          ELSE BF[16] does not Satisfy HO-SAC of 4-bit BFs.
Stop.
```

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

3.3.2.2 Flip Method of HO-SAC of 4-bit BFs and 4-Bit Crypto S-boxes. In Flip Method two or three bits in all possible particular positions of all of 16 INBs have been flipped at a time. The bit values of OPBF before and after flip of bits in INB have been checked for equality. If they are same then in DBF the corresponding bit value appears as 0 else 1. If 10 DBFs have been balanced i.e. it contains equal number of 0s and 1s then the OPBF has been said to satisfy HO-SAC of 4-bit BFs. If four 4-bit BFs of a crypto S-box satisfy HO-SAC of 4-bit BFs individually then the concerned S-box has been said to satisfy HO-SAC of 4-bit crypto S-boxes.

All the elements of the given S-box, Index of each element of the given S-box in hex (INH) and 4 bit binary form (INB) with position of each bit from 1 to 4 of each INB have been given in column 2 through H of row 3, 1, 2 of Table.7 respectively. Each Output BF, OPBF1, OPBF2, OPBF3, OPBF4 has been shown in column 2 through H of row 4, 5, 6, 7 Table.7 respectively.

Now 16 INBs before flip and 16 INBs after flip in two and three bits at a time particularly in 16 INBs bit position 1, 2, 3 and 4 have been shown in row 2 through H of column 1, 2, 6, 7, B, C, G, H respectively of Table.9-A, Table.9-B and Table.9-C respectively . The each corresponding bits of OPBF1, OPBF2, OPBF3, OPBF4 before and after flip have been shown in row 2 through H of column 3, 4, 8, 9, D, E, I, J respectively in Table.9-A, Table.9-B and Table.9-C respectively. If flip occurs in 2 bit positions of INB at a time then the test has been termed as 2nd Order HO-SAC of 4-bit BFs and if flip occurs in 3 bit positions of INB at a time then the test has been termed as 3rd Order HO-SAC of 4-bit BFs.

1 in any position in row 2 through H of column 5, A, F, K illustrate dissimilarity in bits in corresponding positions of OPBF1, OPBF2, OPBF3 and OPBF4 duly before and after flip in one bits in bit positions 1, 2, 3, 4 respectively.

If out of 16 positions in each row from 2 through H of column 5, A, F, K of table.9-A and table.9.B and from row 2 through H of column 5, A of table.9.C, there are 8 1s and 8 0s then the given OPBF has been said to Satisfy HO-SAC of 4-bit BFs. If all four BFs of a given 4-bit crypto S-box satisfy HO-SAC of 4-bit BFs then the S-box has been said to satisfy HO-SAC of 4-bit S-boxes. Here in Table. 9. row I shows the Number 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 12 in all or at least one for the given OPBF so the concerned OPBF and the given 4-bit S-box does Satisfy HO-SAC of 4-bit BFs and HO-SAC of 4-bit S-boxes respectively.

Table.9-A

Col | Row	**Flip of 2 bit of INB at Pos. 21**					**Flip of 2 bits of INB at Pos. 31**					**Flip of 2 bits of INB at Pos. 41**					**Flip of 2 bits of INB at Pos. 32**				
	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H	I	J	K
1	**B-Flip**	**A-Flip**	**1**	**1’**	**C**	**B-Flip**	**A-Flip**	**2**	**2’**	**C**	**B-Flip**	**A-Flip**	**3**	**3’**	**C**	**B-Flip**	**A-Flip**	**4**	**4’**	**C**
2	0000	0011	1	0	1	0000	0101	1	1	0	0000	1001	1	1	0	0000	0110	1	1	0
3	0001	0010	0	1	1	0001	0100	0	0	0	0001	1000	0	0	0	0001	0111	0	1	1
4	0010	0001	1	0	1	0010	0111	1	1	0	0010	1011	1	1	0	0010	0100	1	0	1
5	0011	0000	0	1	1	0011	0100	0	1	1	0011	1010	0	0	0	0011	0101	0	1	1
6	0100	0111	0	1	1	0100	0001	0	0	0	0100	1101	0	1	1	0100	0010	0	1	1
7	0101	0110	1	1	0	0101	0000	1	1	0	0101	1100	1	0	1	0101	0011	1	0	1
8	0110	0100	1	1	0	0110	0011	1	0	1	0110	1111	1	0	1	0110	0000	1	1	0
9	0111	0101	1	0	1	0111	0010	1	1	0	0111	1110	1	0	1	0111	0001	1	0	1
A	1000	1011	0	1	1	1000	1101	0	1	1	1000	0001	0	0	0	1000	1110	0	0	0
B	1001	1010	1	0	1	1001	1100	1	0	1	1001	0000	1	1	0	1001	1111	1	0	1
C	1010	1001	0	1	1	1010	1111	0	0	0	1010	0011	0	0	0	1010	1100	0	0	0

D	1011	1000	1	0	1	1011	1100	1	0	1	1011	0010	1	1	0	1011	1101	1	1	0
E	1100	1111	0	0	0	1100	1001	0	1	1	1100	0101	0	1	1	1100	1010	0	0	0
F	1101	1110	1	0	1	1101	1000	1	0	1	1101	0100	1	0	1	1101	1011	1	1	0
G	1110	1100	0	1	1	1110	1011	0	1	1	1110	0111	0	1	1	1110	1000	0	0	0
H	1111	1101	0	0	0	1111	1010	0	0	0	1111	0110	0	1	1	1111	1001	0	1	1
I	**No of Bits Changed due to Flip** 12					**No of Bits Changed due to Flip** 8					**No of Bits Changed due to Flip** 8					**No of Bits Changed due to Flip** 8				

Table.9-B																				
Col | Row	**Flip of 2 bit of INB at Pos. 42**					**Flip of 2 bits of INB at Pos. 43**					**Flip of 2 bits of INB at Pos. 321**					**Flip of 2 bits of INB at Pos. 421**				
	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H	I	J	K
1	**B-Flip**	**A-Flip**	**5**	**5'**	**C**	**B-Flip**	**A-Flip**	**6**	**6'**	**C**	**B-Flip**	**A-Flip**	**1**	**1'**	**C**	**B-Flip**	**A-Flip**	**2**	**2'**	**C**
2	0000	1010	1	0	1	0000	1100	1	0	1	0000	0111	1	1	0	0000	1011	1	1	0
3	0001	1011	0	1	1	0001	1101	0	1	1	0001	0110	0	1	1	0001	1010	0	0	0
4	0010	1000	1	0	1	0010	1110	1	0	1	0010	0101	1	1	0	0010	1001	1	1	0
5	0011	1001	0	1	1	0011	1111	0	0	0	0011	0100	0	0	0	0011	1000	0	0	0
6	0100	1110	0	0	0	0100	1000	0	0	0	0100	0011	0	0	0	0100	1111	0	0	0
7	0101	1111	1	0	1	0101	1001	1	1	0	0101	0010	1	1	0	0101	1110	1	0	1
8	0110	1100	1	0	1	0110	1010	1	0	1	0110	0001	1	0	1	0110	1101	1	1	0
9	0111	1101	1	1	0	0111	1011	1	1	0	0111	0000	1	1	0	0111	1100	1	0	1
A	1000	0010	0	1	1	1000	1100	0	0	0	1000	1111	0	0	0	1000	0011	0	0	0
B	1001	0011	1	0	1	1001	1101	1	1	0	1001	1110	1	0	1	1001	0010	1	1	0
C	1010	0000	0	1	1	1010	1110	0	1	1	1010	1101	0	1	1	1010	0001	0	0	0
D	1011	0001	1	0	1	1011	1111	1	1	0	1011	1100	1	0	1	1011	0000	1	1	0
E	1100	0110	0	1	1	1100	1000	0	1	1	1100	1011	0	1	1	1100	0111	0	1	1
F	1101	0111	1	1	0	1101	1001	1	0	1	1101	1010	1	0	1	1101	0110	1	1	0
G	1110	0100	0	0	0	1110	1010	0	1	1	1110	1001	0	1	1	1110	0101	0	1	1
H	1111	0101	0	1	1	1111	1011	0	0	0	1111	1000	0	0	0	1111	0100	0	0	0
I	**No of Bits Changed due to Flip** 12					**No of Bits Changed due to Flip** 8					**No of Bits Changed due to Flip** 8					**No of Bits Changed due to Flip** 4				

Table.22. Description of Flip Method of HO-SAC Test of 4-bit BFs.

Table.9-C										
Col | Row	**Flip of 2 bit of INB at Pos. 431**					**Flip of 2 bits of INB at Pos. 432**				
	1	2	3	4	5	6	7	8	9	A
1	**B-Flip**	**A-Flip**	**3**	**3'**	**C**	**B-Flip**	**A-Flip**	**4**	**4'**	**C**
2	0000	1101	1	1	0	0000	1110	1	0	1
3	0001	1100	0	0	0	0001	1111	0	0	0
4	0010	1111	1	0	1	0010	1100	1	0	1
5	0011	1110	0	0	0	0011	1101	0	1	1
6	0100	1001	0	1	1	0100	1010	0	0	0
7	0101	1000	1	0	1	0101	1011	1	1	0
8	0110	1011	1	1	0	0110	1000	1	0	1
9	0111	1010	1	0	1	0111	1001	1	1	0
A	1000	0101	0	1	1	1000	0110	0	1	1
B	1001	0100	1	0	1	1001	0111	1	1	0
C	1010	0111	0	1	1	1010	0100	0	0	0
D	1011	0110	1	1	0	1011	0101	1	1	0
E	1100	0001	0	0	0	1100	0010	0	1	1
F	1101	0000	1	1	0	1101	0011	1	0	1
G	1110	0011	0	0	0	1110	0000	0	1	1
H	1111	0010	0	1	1	1111	0001	0	0	0
I	**No of Bits Changed due to Flip** 8					**No of Bits Changed due to Flip** 8				

Table.9. Description of Flip Method of HO-SAC Test of 4-bit BFs (Continued..).

```
Start.
Step 0A: For I=0:16 For J=0:16 D[I][J] = 0; // Initialization of two dimensional Array D[16][16].
// Initialization of  $e_v$ vector.
Step 0B: ev[4] ={{0,0,1,1},{0,1,0,1},{1,0,0,1},{1,1,0,0},{1,0,1,0},{0,1,1,0},{0,1,1,1},{1,1,0,1},{1,1,1,0},{1,0,1,1}};
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 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
Step 04: if (r==1) D[f][v]++; // count 1 in DBFs
//In next two steps SAC criterion has been evaluated.
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.

3.3.3. Extended HO-SAC Criterion of 4-bit BFs and 4-bit Crypto S-boxes. If one IPV out of four at a time, two IPVs out of four at a time, three IPVs out of four at a time and all of four IPVs have been complemented at a time or flip of one bit of INB at a time, flip of two bits of INB at a time or flip of three bits of INB at a time, and flip of all of four bits in INB at a time and all resultant DBFs have been balanced then the 4-bit BF has been said to satisfy Extended SAC of 4-bit BFs. If all four 4-bit BFs of a 4-bit crypto S-box satisfy Extended SAC of 4-bit BFs then the S-box has been said to satisfy Extended SAC of 4-bit S-boxes. Complement of one, two, or three bits at a time or flip of one, two or three INBs at a time is similar to table 6, 7, 8 and 9 respectively for the given 4-bit BF. The rest Criterion of 4 IPVs have been complemented at a time have been shown in sub table 10-A of table. 10. The rest flip of all INBs at a time have been shown in sub table 10-B of table. 10 .

R|C	**10-A**	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	**IPV3**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
3	**IPV2**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
4	**IPV1**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
5	**OPBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
6	**CIPV4**	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
7	**CIPV3**	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
8	**CIPV2**	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
9	**CIPV1**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
A	**Step 1**	0	1	0	1	0	1	0	0	1	0	1	0	0	1	1	1
B	**Step 2**	0	1	0	0	0	1	0	1	0	1	1	1	1	0	1	0
C	**Step 3**	0	0	0	1	0	1	0	1	1	1	0	1	1	0	1	0
D	**Step 4**	0	0	1	0	1	0	1	0	1	1	1	0	0	1	0	1
E	**COPBF**	0	0	1	0	1	0	1	0	1	1	1	0	0	1	0	1
F	**DBF**	1	0	0	0	1	1	0	1	1	0	1	1	0	0	0	1
G	Number of bits changed in COPBF											8					

10-B					
Col | Row	**Flip of 2 bit of INB at Pos. 4321**				
	1	2	3	4	5
1	**B-Flip**	**A-Flip**	**3**	**3’**	**C**
2	0000	1111	1	0	1
3	0001	1100	0	0	0
4	0010	1101	1	1	0
5	0011	1100	0	0	0
6	0100	1011	0	1	1
7	0101	1010	1	0	1
8	0110	1001	1	1	0
9	0111	1000	1	0	1
A	1000	0111	0	1	1
B	1001	0100	1	1	0
C	1010	0101	0	1	1
D	1011	0100	1	0	1
E	1100	0011	0	0	0
F	1101	0010	1	1	0
G	1110	0001	0	0	0
H	1111	0000	0	1	1
I	**No of Bits Changed due to Flip** 8				

Table.10. Table of Extension to SAC and HO-SAC of 4-bit BFs and 4-bit Crypto S-boxes in both Complement and Flip Method.

Pseudo code. Let BF[16].bit0 has been a bit level array of 16 bits of a 4-bit BF out of 65536 4-bit BFs. and BF[16] has been an array of 16 bits of a 4-bit BF. CV[16].bit0 has been a bit level array of 16 bits to store either 00FF, 0F0F, 3333, 5555 in hex. CVC[16].bit0 has been a bit level array of 16 bits to store either FF00, F0F0, CCCC, AAAA in hex. Here ^ represents Bitwise Xor operation. NL represents Number of bits changed in lower halves and NU represents Number of bits changed in upper halves.

```
Start.
//Initialization of Variables.
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.
```

```
// In next 15 steps DBFs and weight of DBFs have been obtained by xor of OPBF and COPBFs.
Step 01: wt{(BF[16].bit0 & 00FF)^(BF[16].bit0>>8&00FF)}+WT{(BF[16].bit0&FF00)^(BF[16].bit0>>8&FF00)}      = N
Step 02: wt{(BF[16].bit0 & 0F0F)^(BF[16].bit0>>4&0F0F)}+WT{(BF[16].bit0&F0F0)^(BF[16].bit0>>4&F0F0)}      = N
Step 03: wt{(BF[16].bit0 & 3333)^(BF[16].bit0>>2&3333)}+WT{(BF[16].bit0&CCCC)^(BF[16].bit0>>2&CCCC)}   = N
Step 04: wt{(BF[16].bit0 & 5555)^(BF[16].bit0>>1&5555)}+WT{(BF[16].bit0&AAAA)^(BF[16].bit0>>1&AAAA)}  = N
Step 05: wt[{(BF[16].bit0 & 00FF)^(BF[16].bit0>>8&00FF)}+WT{(BF[16].bit0&FF00)^(BF[16].bit0>>8&FF00)}
             ^{(BF[16].bit0 & 0F0F)^(BF[16].bit0>>4&0F0F)}+WT{(BF[16].bit0&F0F0)^(BF[16].bit0>>4&F0F0)}]      = N.
Step 06: wt[{(BF[16].bit0 & 00FF)^(BF[16].bit0>>8&00FF)}+WT{(BF[16].bit0&FF00)^(BF[16].bit0>>8&FF00)}
             ^{(BF[16].bit0 & 3333)^(BF[16].bit0>>2&3333)}+WT{(BF[16].bit0&CCCC)^(BF[16].bit0>>2&CCCC)}]   = N .
Step 07: wt[{(BF[16].bit0 & 00FF)^(BF[16].bit0>>8&00FF)}+WT{(BF[16].bit0&FF00)^(BF[16].bit0>>8&FF00)}
             ^{(BF[16].bit0 & 5555)^(BF[16].bit0>>1&5555)}+WT{(BF[16].bit0&AAAA)^(BF[16].bit0>>1&AAAA)}] = N.
Step 08: wt[{(BF[16].bit0 & 0F0F)^(BF[16].bit0>>4&0F0F)}+WT{(BF[16].bit0&F0F0)^(BF[16].bit0>>4&F0F0)}
             ^{(BF[16].bit0 & 3333)^(BF[16].bit0>>2&3333)}+WT{(BF[16].bit0&CCCC)^(BF[16].bit0>>2&CCCC)}]  = N
Step 09: wt[{(BF[16].bit0 & 0F0F)^(BF[16].bit0>>4&0F0F)}+WT{(BF[16].bit0&F0F0)^(BF[16].bit0>>4&F0F0)}
             ^{(BF[16].bit0 & 5555)^(BF[16].bit0>>1&5555)}+WT{(BF[16].bit0&AAAA)^(BF[16].bit0>>1&AAAA)}  = N.
Step 10: wt[{(BF[16].bit0 & 3333)^(BF[16].bit0>>2&3333)}+WT{(BF[16].bit0&CCCC)^(BF[16].bit0>>2&CCCC)}
             ^{(BF[16].bit0 & 5555)^(BF[16].bit0>>1&5555)}+WT{(BF[16].bit0&AAAA)^(BF[16].bit0>>1&AAAA)}] = N.
Step 11: wt[{(BF[16].bit0 & 00FF)^(BF[16].bit0>>8&00FF)}+WT{(BF[16].bit0&FF00)^(BF[16].bit0>>8&FF00)}
             ^{(BF[16].bit0 & 0F0F)^(BF[16].bit0>>4&0F0F)}+WT{(BF[16].bit0&F0F0)^(BF[16].bit0>>4&F0F0)}
             ^{(BF[16].bit0 & 3333)^(BF[16].bit0>>2&3333)}+WT{(BF[16].bit0&CCCC)^(BF[16].bit0>>2&CCCC)}]   = N.
Step 12: wt[{(BF[16].bit0 & 00FF)^(BF[16].bit0>>8&00FF)}+WT{(BF[16].bit0&FF00)^(BF[16].bit0>>8&FF00)}
             ^{(BF[16].bit0 & 0F0F)^(BF[16].bit0>>4&0F0F)}+WT{(BF[16].bit0&F0F0)^(BF[16].bit0>>4&F0F0)}
             ^{(BF[16].bit0 & 5555)^(BF[16].bit0>>1&5555)}+WT{(BF[16].bit0&AAAA)^(BF[16].bit0>>1&AAAA)}] = N
Step 13: wt[{(BF[16].bit0 & 00FF)^(BF[16].bit0>>8&00FF)}+WT{(BF[16].bit0&FF00)^(BF[16].bit0>>8&FF00)}
              ^{(BF[16].bit0 & 3333)^(BF[16].bit0>>2&3333)}+WT{(BF[16].bit0&CCCC)^(BF[16].bit0>>2&CCCC)}
            ^{(BF[16].bit0 & 5555)^(BF[16].bit0>>1&5555)}+WT{(BF[16].bit0&AAAA)^(BF[16].bit0>>1&AAAA)}] = N.
Step 14: wt[{(BF[16].bit0 & 0F0F)^(BF[16].bit0>>4&0F0F)}+WT{(BF[16].bit0&F0F0)^(BF[16].bit0>>4&F0F0)}
             ^{(BF[16].bit0 & 3333)^(BF[16].bit0>>2&3333)}+WT{(BF[16].bit0&CCCC)^(BF[16].bit0>>2&CCCC)}
             ^{(BF[16].bit0 & 5555)^(BF[16].bit0>>1&5555)}+WT{(BF[16].bit0&AAAA)^(BF[16].bit0>>1&AAAA)}] = N.
Step 15: wt{(BF[16].bit0 & 00FF)^(BF[16].bit0>>8&00FF)}+WT{(BF[16].bit0&FF00)^(BF[16].bit0>>8&FF00)}
            ^wt{(BF[16].bit0&0F0F)^(BF[16].bit0>>4&0F0F)}+WT{(BF[16].bit0&F0F0)^(BF[16].bit0>>4&F0F0)}
            ^wt{(BF[16].bit0&3333)^(BF[16].bit0>>2&3333)}+WT{(BF[16].bit0&CCCC)^(BF[16].bit0>>2&CCCC)}
            ^wt{(BF[16].bit0 & 5555)^(BF[16].bit0>>1&5555)}+WT{(BF[16].bit0&AAAA)^(BF[16].bit0>>1&AAAA) = N
// In next step Extended SAC criterion has been evaluated.
Step 16:  If N=8 for Step 01, to Step 15. Then BF[16].bit0 Satisfies Extended SAC of 4-bit BFs.
          ELSE BF[16].bit0 Does not Satisfies Extended SAC of 4-bit BFs..
Stop.
```

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

Pseudo code of Extended SAC of 4-bit BFs Using Shift Method.

```
Start.
Step 00:  For 1:16 BF[16].bit0 = BF[16].
Step 1A: CBF[16].bit0 = (BF[16].bit0>>8); // Circular shift of each distinct 8 bit halves of 16 bit long OPBFs.
Step 1B: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0; // Difference BF have been obtained by bitwise xor of OPBFs and COPBFs.
Step 1C: Count = IF(DBF[16].bit0==1); // Count Number of 1s in DBF.
Step 2A: CBF[16].bit0 = (BF[8A].bit0>>4)&& (BF[8B].bit0>>4); //Circular shift of each distinct 4-bit halves of 4-bit OPBFs.
Step 2B: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0; // Difference BF have been obtained by bitwise xor of OPBFs and COPBFs.
Step 2C: Count = IF(DBF[16].bit0==1); // Count Number of 1s in DBF.
// In next step Each distinct 2 bit halves of each distinct 4-bit Halves have been circularly shifted.
Step 3A: CBF[16].bit0 = (BF[4A].bit0>>2)&& (BF[4B].bit0>>2)&& (BF[4C].bit0>>2)&& (BF[4D].bit0>>2);
Step 3B: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0; // Difference BF have been obtained by bitwise xor of OPBFs and COPBFs.
Step 3C: Count = IF(DBF[16].bit0==1); // Count Number of 1s in DBF.
// In next step each bit of each distinct two bit halves has been circularly shifted.
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);
Step 4B: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0; // Difference BF have been obtained by bitwise xor of OPBFs and COPBFs.
Step 4C: Count = IF(DBF[16].bit0==1); // Count Number of 1s in DBF.
Step 5A: CBF[16].bit0 = (BF[16].bit0>>8); // Circular shift of each distinct 8 bit halves of 16 bit long OPBFs.
Step 5B: CBF[16].bit0 = (CBF[8A].bit0>>4)&& (CBF[8B].bit0>>4); //Circular shift of each distinct 4-bit halves of 4-bit OPBFs.
Step 5C: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0; // Difference BF have been obtained by bitwise xor of OPBFs and COPBFs.
Step 5D: Count = IF(DBF[16].bit0==1);// Count Number of 1s in DBF.
Step 6A: CBF[16].bit0 = (BF[16].bit0>>8); // Circular shift of each distinct 8 bit halves of 16 bit long OPBFs.
```

```
// In next step Each distinct 2 bit halves of each distinct 4-bit Halves have been circularly shifted.
Step 6B: CBF[16].bit0 = (CBF[4A].bit0>>2)&& (CBF[4B].bit0>>2)&& (CBF[4C].bit0>>2)&& (CBF[4D].bit0>>2);
Step 6C: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0; // Difference BF have been obtained by bitwise xor of OPBFs and COPBFs.
Step 6D: Count = IF(DBF[16].bit0==1); // Count Number of 1s in DBF.
Step 7A: CBF[16].bit0 = (BF[16].bit0>>8); // Circular shift of each distinct 8 bit halves of 16 bit long OPBFs.
// In next step each bit of each distinct two bit halves has been circularly shifted.
Step 7B: CBF[16].bit0 = (CBF[2A].bit0>>1)&&(CBF[2B].bit0>>1)&&(CBF[2C].bit0>>1)&&(CBF[2D].bit0>>1)&&
                        (CBF[2E].bit0>>1)&&(CBF[2F].bit0>>1)&&(CBF[2G].bit0>>1)&&(CBF[2H].bit0>>1);
Step 7C: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0; // Difference BF have been obtained by bitwise xor of OPBFs and COPBFs.
Step 7D: Count = IF(DBF[16].bit0==1); // Count Number of 1s in DBF.
Step 8A: CBF[16].bit0 = (BF[8A].bit0>>4)&& (BF[8B].bit0>>4); //Circular shift of each distinct 4-bit halves of 4-bit OPBFs.
// In next step Each distinct 2 bit halves of each distinct 4-bit Halves have been circularly shifted.
Step 8B: CBF[16].bit0 = (CBF[4A].bit0>>2)&& (CBF[4B].bit0>>2)&& (CBF[4C].bit0>>2)&& (CBF[4D].bit0>>2);
Step 8C: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0; // Difference BF have been obtained by bitwise xor of OPBFs and COPBFs.
Step 8D: Count = IF(DBF[16].bit0==1); // Count Number of 1s in DBF.
Step 9A: CBF[16].bit0 = (BF[8A].bit0>>4)&& (BF[8B].bit0>>4); //Circular shift of each distinct 4-bit halves of 4-bit OPBFs.
// In next step each bit of each distinct two bit halves has been circularly shifted.
Step 9B: CBF[16].bit0 = (CBF[2A].bit0>>1)&&(CBF[2B].bit0>>1)&&(CBF[2C].bit0>>1)&&(CBF[2D].bit0>>1)&&
                        (CBF[2E].bit0>>1)&&(CBF[2F].bit0>>1)&&(CBF[2G].bit0>>1)&&(CBF[2H].bit0>>1);
Step 9C: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0; // Difference BF have been obtained by bitwise xor of OPBFs and COPBFs.
Step 9D: Count = IF(DBF[16].bit0==1); // Count Number of 1s in DBF.
// In next step Each distinct 2 bit halves of each distinct 4-bit Halves have been circularly shifted.
Step 10A: CBF[16].bit0 = (BF[4A].bit0>>2)&& (BF[4B].bit0>>2)&& (BF[4C].bit0>>2)&& (BF[4D].bit0>>2);
// In next step each bit of each distinct two bit halves has been circularly shifted.
Step 10B: CBF[16].bit0 = (CBF[2A].bit0>>1)&&(CBF[2B].bit0>>1)&&(CBF[2C].bit0>>1)&&(CBF[2D].bit0>>1)&&
                        (CBF[2E].bit0>>1)&&(CBF[2F].bit0>>1)&&(CBF[2G].bit0>>1)&&(CBF[2H].bit0>>1);
Step 10C: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0; // Difference BF have been obtained by bitwise xor of OPBFs and COPBFs.
Step 10D: Count = IF(DBF[16].bit0==1); // Count Number of 1s in DBF.
Step 11A: CBF[16].bit0 = (BF[16].bit0>>8); // Circular shift of each distinct 8 bit halves of 16 bit long OPBFs.
Step 11B: CBF[16].bit0 = (CBF[8A].bit0>>4)&& (CBF[8B].bit0>>4); //Circular shift of each distinct 4-bit halves of 4-bit OPBFs.
// In next step Each distinct 2 bit halves of each distinct 4-bit Halves have been circularly shifted.
Step 11C: CBF[16].bit0 = (CBF[4A].bit0>>2)&& (CBF[4B].bit0>>2)&& (CBF[4C].bit0>>2)&& (CBF[4D].bit0>>2);
Step 11D: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0; // Difference BF have been obtained by bitwise xor of OPBFs and COPBFs.
Step 11E: Count = IF(DBF[16].bit0==1); // Count Number of 1s in DBF.
Step 12A: CBF[16].bit0 = (BF[16].bit0>>8); // Circular shift of each distinct 8 bit halves of 16 bit long OPBFs.
Step 12B: CBF[16].bit0 = (CBF[8A].bit0>>4)&& (CBF[8B].bit0>>4); //Circular shift of each distinct 4-bit halves of 4-bit OPBFs.
// In next step each bit of each distinct two bit halves has been circularly shifted.
Step 12C: CBF[16].bit0 = (CBF[2A].bit0>>1)&&(CBF[2B].bit0>>1)&&(CBF[2C].bit0>>1)&&(CBF[2D].bit0>>1)&&
                        (CBF[2E].bit0>>1)&&(CBF[2F].bit0>>1)&&(CBF[2G].bit0>>1)&&(CBF[2H].bit0>>1);
Step 12D: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0; // Difference BF have been obtained by bitwise xor of OPBFs and COPBFs.
Step 12E: Count = IF(DBF[16].bit0==1); // Count Number of 1s in DBF.
Step 13A: CBF[16].bit0 = (BF[16].bit0>>8); // Circular shift of each distinct 8 bit halves of 16 bit long OPBFs.
// In next step Each distinct 2 bit halves of each distinct 4-bit Halves have been circularly shifted.
Step 13B: CBF[16].bit0 = (CBF[4A].bit0>>2)&& (CBF[4B].bit0>>2)&& (CBF[4C].bit0>>2)&& (CBF[4D].bit0>>2);
// In next step each bit of each distinct two bit halves has been circularly shifted.
Step 13C: CBF[16].bit0 = (CBF[2A].bit0>>1)&&(CBF[2B].bit0>>1)&&(CBF[2C].bit0>>1)&&(CBF[2D].bit0>>1)&&
                        (CBF[2E].bit0>>1)&&(CBF[2F].bit0>>1)&&(CBF[2G].bit0>>1)&&(CBF[2H].bit0>>1);
Step 13D: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0; // Difference BF have been obtained by bitwise xor of OPBFs and COPBFs.
Step 13E: Count = IF(DBF[16].bit0==1); // Count Number of 1s in DBF.
Step 14A: CBF[16].bit0 = (CBF[8A].bit0>>4)&& (CBF[8B].bit0>>4); //Circular shift of each distinct 4-bit halves of 4-bit OPBFs.
// In next step Each distinct 2 bit halves of each distinct 4-bit Halves have been circularly shifted.
Step 14B: CBF[16].bit0 = (CBF[4A].bit0>>2)&& (CBF[4B].bit0>>2)&& (CBF[4C].bit0>>2)&& (CBF[4D].bit0>>2);
// In next step each bit of each distinct two bit halves has been circularly shifted.
Step 14C: CBF[16].bit0 = (CBF[2A].bit0>>1)&&(CBF[2B].bit0>>1)&&(CBF[2C].bit0>>1)&&(CBF[2D].bit0>>1)&&
                         (CBF[2E].bit0>>1)&&(CBF[2F].bit0>>1)&&(CBF[2G].bit0>>1)&&(CBF[2H].bit0>>1);
Step 14D: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0; // Difference BF have been obtained by bitwise xor of OPBFs and COPBFs.
Step 14E: Count = IF(DBF[16].bit0==1); // Count Number of 1s in DBF.
Step 15A: CBF[16].bit0 = (BF[16].bit0>>8); // Circular shift of each distinct 8 bit halves of 16 bit long OPBFs.
Step 15B: CBF[16].bit0 = (CBF[8A].bit0>>4)&& (CBF[8B].bit0>>4); //Circular shift of each distinct 4-bit halves of 4-bit OPBFs.
// In next step Each distinct 2 bit halves of each distinct 4-bit Halves have been circularly shifted.
Step 15C: CBF[16].bit0 = (CBF[4A].bit0>>2)&& (CBF[4B].bit0>>2)&& (CBF[4C].bit0>>2)&& (CBF[4D].bit0>>2);
// In next step each bit of each distinct two bit halves has been circularly shifted.
```

```
Step 15D: CBF[16].bit0 = (CBF[2A].bit0>>1)&&(CBF[2B].bit0>>1)&&(CBF[2C].bit0>>1)&&(CBF[2D].bit0>>1)&&
                         (CBF[2E].bit0>>1)&&(CBF[2F].bit0>>1)&&(CBF[2G].bit0>>1)&&(CBF[2H].bit0>>1);
Step 15E: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0; // Difference BF have been obtained by bitwise xor of OPBFs and COPBFs.
Step 15F: Count = IF(DBF[16].bit0==1); // Count Number of 1s in DBF.
// In next step Extended SAC criterion has been evaluated.
Step 16 : IF Count = 8 for Step 1C, TO 4C, 5D to 10D and 11E to 14E and 15F BF[16] Satisfies Extended-SAC of 4-bit BFs.
          ELSE BF[16] does not Satisfy Extended SAC of 4-bit BFs.
Stop.
```

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

Pseudo code of Extended SAC of 4-bit BFs Using Flip Method.

```
Start.
Step 0A: For I=0:16 For J=0:16 D[I][J] = 0; // Initialization of 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},// Initialization of $e_v$ vector
                {0,0,1,1},{0,1,0,1},{1,0,0,1},{1,1,0,0},
                {1,0,1,0},{0,1,1,0},{0,1,1,1},{1,1,0,1},
                {1,1,1,0},{1,0,1,1},{1,1,1,1}};
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 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]];// DBFs generation for 16 $e_v$s.
Step 04: if (r==1) D[f][v]++; // count of 1s in DBF.
// In next two steps Extended SAC criterion has been evaluated.
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 Extended SAC of 4-bit BFs then the given S-Box Satisfies Extended SAC of 4-bit S-Box.
         ELSE the given S-Box does not Satisfy Extended 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.

3.4. A Brief Review of Differential Cryptanalysis of 4-bit S-boxes and a new Technique with Boolean Functions for Differential Cryptanalysis of 4-bit S-boxes.

The given 4-bit Crypto S-box has been described in sub-section 3.4.1. The relation Between 4-bit Crypto S-boxes and 4-bit BFs has been illustrated in subsec. 3.4.2., The Differential Cryptanalysis of 4-bit Crypto S-boxes and DDT or Differential Distribution Table has been illustrated in subsec. 3.4.3. The Differential Cryptanalysis of 4-bit S-boxes with 4-bit BFs has been described in subsec.3.4.4.

3.4.1 4-bit Crypto S-boxes: A 4-bit Crypto S-box can be written as Follows in Table.11, where the each element of the first row of Table.11, 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 Substitution box. It can be concluded that the 1st row is fixed for all possible 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 [AT90][NT77][NT99].

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

Table.11. 4-bit crypto S-box.

3.4.2 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.12. 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 is a 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 [AT90]. The decimal equivalent of each IPV and OPBF are noted at column H of respective rows.

Row	Column	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	**H.** Decimal

1	**Index**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**	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.12. Decomposition of 4-bit input S-box and given S-box (1st 4-bit S-box of 1st S-box out of 8 of DES) to 4-bit BFs.

3.4.3 Review of Differential Cryptanalysis of 4-bit Crypto S-boxes [HH96]**[**HH02**].** In Differential Cryptanalysis of 4-bit Crypto S-boxes, Elements of 4-bit input S-box (ISB) have been xored with a particular 4-bit Input Difference (ID) to obtain a Distant input S-box (DISB). The Distant S-boxes (DSB) have been obtained from original S-box (SB) by shuffling the elements of SB in such order in the way in which the elements of ISB have been shuffled to obtain DISB for a Particular ID. Each element of Difference S-box (DFSB) have been obtained by the xor operation of corresponding elements of SB and DSB. The Count of each Hexadecimal number from 0 to F have been put into the concerned cell of Differential Distribution Table or DDT. As the number of 0s in DDT increases, information regarding concerned Output Difference (OD) increases so the S-box has been determined as weak S-box. The 4-bit Sequence of each element of ISB, ID, DISB, DSB, DFSB have been given in BIN ISB, BIN ID, BIN DISB, BIN DSB, BIN DFSB respectively.

The column.1. in Table.13. from row 1 through G shows the 16 elements of ISB in a monotonically increasing sequence or order. The ISB can also be concluded as an Identity 4-bit S-box. The elements of 1st 4-bit S-box, out of 4 of 1st S-box of Data Encryption Standard (DES) out of 8, has been considered as S-box (SB), in column 7 from row 1 through G. The elements of ID, DISB, DSB, DFSB has been shown in row 1 through G of column. 3, 5, 9 and C of Table.3 respectively. The 4-bit Binary equivalents of each elements of ISB, ID, DISB, SB, DSB, DFSB, has been shown in row 1 through G of column. 2, 4, 6, 8, A and B of Table.13 respectively.

The review has been done in two different views; The S-box view has been described in subsec.3.4.3.1. in which the concerned column of interest are row 1 through G of column 1, 3, 5, 7, 9 and C respectively. The 4-bit binary pattern view has also been described in subsec.3.4.3.2 in which concerned column of interest are row 1 through G of column 2, 4, 6, 8, A and b respectively. The Pseudo Code of two algorithms with their time complexity comparison has been illustrated in subsec.3.4. 3.3.

COL	1	2	3	4	5	6	7	8	9	A	B	C
R O W	I S B	Bin ISB 4321	I D	Bin ID 4321	D IS B	Bin DISB 4321	S B	Bin OSB 4321	D S B	Bin DSB 4321	Bin DFSB 4321	D FS B
1	0	0000	B	1011	B	1011	E	1110	C	1100	0010	2
2	1	0001	B	1011	A	1010	4	0100	6	0110	0010	2
3	2	0010	B	1011	9	1001	D	1101	A	1010	0001	7
4	3	0011	B	1011	8	1000	1	0001	3	0011	0010	2
5	4	0100	B	1011	F	1111	2	0010	7	0111	0101	5
6	5	0101	B	1011	E	1110	F	1111	0	0000	1111	F
7	6	0110	B	1011	D	1101	B	1011	9	1001	0010	2
8	7	0111	B	1011	C	1100	8	1000	5	0101	1101	D
9	8	1000	B	1011	3	0011	3	0011	1	0001	0010	2
A	9	1001	B	1011	2	0010	A	1010	D	1101	0001	7
B	A	1010	B	1011	1	0001	6	0110	4	0100	0010	2
C	B	1011	B	1011	0	0000	C	1100	E	1110	0010	2
D	C	1100	B	1011	7	0111	5	0101	8	1000	1101	D
E	D	1101	B	1011	6	0110	9	1001	B	1011	0010	2
F	E	1110	B	1011	5	0101	0	0000	F	1111	1111	F
G	F	1111	B	1011	4	0100	7	0111	2	0010	0101	5

Table.13. Table of Differential Cryptanalysis of 1st 4-bit S-box of 1st S-box out of 8 of DES.

3.4.3.1 S-box View of Differential Cryptanalysis of 4-bit Crypto S-boxes. The S-box with a particular input difference or ID from 0 to F in which all elements have the same value 'B' in hex, is not a Crypto Box but an S-box and is shown in row 1 through G of column. 3. of Table.13. The Distant input S-box (DISB) is shown in row 1 through G of column.5 of the said table. In DISB each row element from row 1 through G is obtained by the xor operation of the elements in corresponding positions of each element of DISB from row 1 through G of column.1. (ISB) and Column.3. (ID) respectively. In ISB for each row element from row 1 through G of column.1.just in corresponding position from row 1 through G of column.7, there is an element of SB. Now in DISB the elements of ISB have been shuffled in a particular order and In DSB the corresponding elements of SB has also been shuffled in that particular order. Each element of the Difference S-box or DFSB from row 1 through G of column.C. has been obtained by xor operation of each element in corresponding positions from row 1 through G of column.7. and row 1 through G of column.9. respectively. The repetition of each existing elements in DSB have been counted and put into Difference Distribution Table or DDT. It is shown in Table.14. as follows,

R|C	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
1	**DSB el**	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	**Count**	0	0	8	0	0	2	0	2	0	0	0	0	0	2	0	2

Table.14. Count of repetition of each existing element in DSB.

The count of each existing elements in DFSB have been put into Difference Distribution Table. as follows, in row 2 of Table.15. For Input Difference (ID) = 'B' and Output Difference from 0 through F of row 1.

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

Table.15. The Part of DDT with Input Difference 'B'.

3.4.3.2. 4-bit binary Pattern View of Differential Cryptanalysis of 4-bit Crypto S-boxes. The corresponding four bit bit patterns of input S-box elements (ISB) has been shown from row 1 through G of column.2 in Table.13. and termed as Bin ISB. The Particular Input Difference '1101' is shown in each row from 1 through G of column. 4. in Table.13. The Distant 4-bit input bit patterns are shown from row 1 through G of column.6. (Bin DISB) are obtained by the xor operation of the elements in corresponding positions of each element of BIN DISB from row 1 through G of column.2. (Bin ISB) and Column.4. (Bin ID) respectively. In Bin ISB for each element from row 1 through G of column.2. in corresponding position from row 1 through G of column.8, there is an element of Bin SB. Now in Bin DISB the elements of ISB have been shuffled in a particular order and in Bin DSB the corresponding elements of SB has also been shuffled in that particular order. Each element from row 1 through G of column.11. has been obtained by xor operation of each element in corresponding positions from row 1 through G of column.8. and row 1 through G of column.10. respectively. The repetition of each existing elements in Bin DFSB have been counted and put into Difference Distribution Table or DDT. It is shown in Table.16. as follows,

R|C	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
1	**DFSB el**	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
2	**Count**	0	2	8	0	0	2	0	0	0	0	0	0	0	2	0	2

Table.16. Count of repetition of each existing element in Bin DSB.

The count of each existing elements in Bin DFSB have been put into the Differential Distribution Table. as follows, in row 2 of Table.17a. for Binary Input Difference (Bin ID) '1101' and Output Difference from 0 through F of row 1.

Row	1	**Output Difference (In Hex)**															
1	**Input Difference**	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	**1101**	0	2	8	0	0	2	0	0	0	0	0	0	0	2	0	2

Table.17a. The Part of DDT with Input Difference '1101'.

The Total DDT or Difference Distribution table for 16 IDs for the given S-box has been shown below in table 17b.

Table.7b		**Output Difference**															
DDT		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
Input DIfference	**0**	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	2	0	0	0	2	0	2	4	0	4	2	0	0
	2	0	0	0	2	0	6	2	2	0	2	0	0	0	0	2	0
	3	0	0	2	0	2	0	0	0	0	4	2	0	2	0	0	4
	4	0	0	0	2	0	0	6	0	0	2	0	4	2	0	0	0
	5	0	4	0	0	0	2	2	0	0	0	4	0	2	0	0	2
	6	0	0	0	4	0	4	0	0	0	0	0	0	2	2	2	2
	7	0	0	2	2	2	0	2	0	0	2	2	0	0	0	0	4
	8	0	0	0	0	0	0	2	2	0	0	0	4	0	4	2	2
	9	0	2	0	0	2	0	0	4	2	0	2	2	2	0	0	0
	A	0	2	2	0	0	0	0	0	6	0	0	2	0	0	4	0
	B	0	0	8	0	0	2	0	2	0	0	0	0	0	2	0	2
	C	0	2	0	0	2	2	2	0	0	0	0	2	0	6	0	0
	D	0	4	0	0	0	0	0	4	2	0	2	0	2	0	2	0
	E	0	0	2	4	2	0	0	0	6	0	0	0	0	0	2	0
	F	0	2	0	0	6	0	0	0	0	4	0	2	0	0	2	0

Table.17b. Difference Distribution Table or DDT of the Given S-box.

3.4.4 Pseudo Code for Differential Cryptanalysis of 4-bit Crypto S-boxes and its Time Complexity Analysis.

The Pseudo Code of Algorithm for 4-bit binary pattern view with Time Complexity has been depicted in subsec. 3.4.4.1., the Pseudo Code of algorithm for S-box view with Time Complexity has been depicted in subsec. 3.4.4.2. and The comparison of time complexity of two algos has been given in subsec 3.4.4.3.

3.4.4.1 Pseudo Code of Algorithm of Differential Cryptanalysis 4-bit binary pattern view.

The Pseudo Code has been given as follows,

```
Start.     // Start of Pseudo Code
// Variable Declarations, Two Dimensional Array ISB[4][16] is for 4-bit bit patterns for Input S-box, IDIFF[4][16] is for 4-bit bit patterns of Input Difference, Three Dimensional Array ODIFF[4][16][16] is for all 4-bit bit patterns of Output Difference for 16 IDIFFs.
Step 0A: int ISB[4][16]; int IDIFF[4][16]; int ODIFF[4][16][16];
// Variable Declarations, ISB'[4][16][16] is for 4-bit bit patterns of All elements of 16 distant ISBs. OSB[4][16] is for 4-bit bit patterns of the given S-box or Output S-box , OSB'[4][16][16] is for 4-bit bit patterns of All elements of 16 distant OSBs, DDT[16][16] is for Difference Distribution Table, and Count[16] is for count of each element in ODIFF for 16 OSBs.
Step 0B: int ISB'[4][16][16]; int OSB[4][16]; int OSB'[4][16][16]; int DDT[16][16]; int Count[16];
// Differential Cryptanalysis Block.
Step 01:  For I =1:16 ; For J =1:16 ; For K =1:4; // Start of For Loop I, J, K respectively
                    ISB'[K][I][J] = ISB[K][J]^IDIFF[K][I];
                    OSB'[K][I][J] = OSB[ISB'[K][I][J]];
                    ODIFF[K][I][J] = OSB[K][J]^ OSB'[K][I][J];
          End For K. End For J. End For I.// End of For loop K, J, I respectively
// Generation of Difference Distribution Table.
Step 02:  For I =1:16 For J =1:16 For K =1:4 // Start of For Loop I, J, K respectively
          DDT[I][J]= Count[ISB[K][J]];
          End For K. End For J. End For I. // End of For loop K, J, I respectively
Stop. // End of Pseudo Code
```

Time Complexity of the Given Algorithm. Since Differential Cryptanalysis block contains 3 nested loops so the time Complexity of the Algorithm has been O(n^3).

3.4.4.2. Pseudo Code of Algorithm of Differential Cryptanalysis S-box View.

The Pseudo Code has been given as follows,

```
Start. // Start of Pseudo Code
// Variable Declarations, One Dimensional Array ISB[16] is for for Input S-box in Hex, IDIFF[16] is for Input Difference in Hex, Three Dimensional Array ODIFF[16][16] is for all Output Difference in Hex for 16 IDIFFs.
Step 0A: int ISB[16]; int IDIFF[16]; int ODIFF[16][16];
// Variable Declarations, ISB'[16][16] is for All elements in Hex of 16 distant ISBs. OSB[16] is for elements in Hex of the given S-box or Output S-box , OSB'[16][16] is for All elements in Hex of 16 distant OSBs, DDT[16][16] is for Difference Distribution Table, and Count[16] is for count of each element in ODIFF for 16 OSBs.
Step 0B: int ISB'[16][16]; int OSB[16]; int OSB'[16][16]; int DDT[16][16]; int Count[16].
```

```
// Differential Cryptanalysis block
Step 01:  For I =1:16; For J =1:16; // For Loop I and J respectively.
              ISB'[I][J] = ISB[J]^IDIFF[I];
              OSB'[I][J] = OSB[ISB'[I][J]];
              ODIFF[I][J] = OSB[J]^ OSB'[I][J];
          End For J. End For I.// End of For Loop J and I respectively.
Step 02:  For I =1:16; For J =1:16 // For Loop I and J respectively.
              DDT[I][J]= Count[ISB[J]];
          End For J. End For I. // End of For Loop J and I respectively.
Stop. // End of Pseudo Code
```

Time Complexity of the Given Algorithm. Since Differential Cryptanalysis block contains 2 nested loops so the time Complexity of the Algorithm has been O(n^2).

3.4.4.3 Comparison of Time Complexity of Two views of Differential Cryptanalysis of 4-bit S-boxes.

The Comparison of time complexity of two algos has been given in Table.17C as follows,

View	4-bit BP View	S-box View
Time Complexity	O(n^3)	O(n^2)

Table.17C. Time Complexity Comparison of Two Algos.

It can be concluded from the comparison that the Execution Time reduces in S-box view than the 4-bit Binary Pattern view. So in can be concluded from above review work that the execution time of Differential Cryptanalysis depends upon the view of the algorithm and the S-box view has been proved to be a better algorithm than 4-bit binary pattern view algorithm.

3.5 Differential Cryptanalysis of 4-bit Bijective Crypto S-boxes with 4-bit BFs. The Procedure to obtain four Input Vectors (IPVs) and Four Output BFs (OPBFs) from the elements of a particular 4-bit Crypto S-box has been described in sec.2.1. The procedure to obtain distant Input Vectors (DIPVs) and Distant Output BFs (DOPBFs) for a particular Input Difference (ID) of the said S-box has been described with example in sec.3.5.1. Generation of Difference 4-bit BFs, Analysis of Algorithm and Generation of Difference Analysis Algorithm in subsec.3.5.2, subsec.3.5.3 and subsec.3.5.4. respectively. The Differential Analysis Table of the given S-box, Pseudo Code of Algorithm with Time Complexity and Comparison of Time complexity of three Algos. have been given in subsec.3.5.5, subsec.3.5.6 and subsec.3.5.7. respectively.

3.5.1. Distant Input BFs (DIBFs) and Distant Output BFs (DOBFs) Generation from IBFs and OBFs for a specific ID. Within 4 bits of binary input difference (Bin ID), 1 in position p means do complement of p^{th} IPV and 0 means no operation on p^{th} IPV. Similarly in the given example 1 in position 4 of Bin ID, as in position 4 from row 1 through G of column 4 of table.13. indicates do complement of 4-bit IPV, IPV4 i.e. CIPV4 and 0 in position 3 as in position 3 from row 1 through G of column 4 of table.13. means no operation on 4-bit IPV, IPV3 (CIPV3) or CIPV3 = IPV3. Similarly 1 in respective positions 2 and 1 as in positions 2 and 1 from row 1 through G of column 4 of table.13. means do complement 4-bit IPV, IPV2 (CIPV2) and do complement of 4-bit IPV, IPV1 (CIPV1) respectively. CIPV4, CIPV3, CIPV2 and CIPV1 for Input S-box (ISB) and Input Difference (ID) have been shown from row 1 through G of column.1. and column.3. of Table.13. respectively.

Here the 4th OPBF has been taken as an example of OPBF and termed as OPBF. Since complement of 4th IPV means interchanging each 8 bit halves of 16 bit long 4th IPV so The 2, 8 bit halves of OPBF have been interchanged due to complement of 4th IPV. The resultant OPBF has been shown from column 1 through G of row 6 in Table.19. Again No Operation on 3rd IPV means CIPV3 = IPV3 so resultant OPBF is as same as STEP1 and has been shown from column 1 through G of row 7 in Table.19. Next to it, the complement of 2nd IPV means interchanging each 2 bit halves of each 4 bit halves of each 8 bit halves of resultant OPBF. The resultant OPBF has been shown from column 1 through G of row 8 in Table.19. Again the complement of 1st IPV means interchanging each bit of each 2 bit halves of each 4 bit halves of each 8 bit halves of resultant OPBF, The resultant OPBF After operation has been shown in column 1 through G of row 9 in Table.19. The Complemented OPBF has been the resultant OPBF of STEP4 and has been shown from column 1 through G of row A in Table.19.

ID	1	0	1	1
Complement	C	N	C	C

Table.18. Complement of IPVs Due to a Particular ID

Row	**Col**	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G
1	**CIPV4**	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
2	**CIPV3**	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
3	**CIPV2**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
4	**CIPV1**	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
5	**OPBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
6	**STEP1**	0	1	0	1	0	1	0	0	1	0	1	0	0	1	1	1
7	**STEP2**	0	1	0	1	0	1	0	0	1	0	1	0	0	1	1	1
8	**STEP3**	0	1	0	1	0	0	0	1	1	0	1	0	1	1	0	1
9	**STEP4**	1	0	1	0	0	0	1	0	0	1	0	1	1	1	1	0
A	**COPBF**	1	0	1	0	0	0	1	0	0	1	0	1	1	1	1	0

Table. 19. Construction of DIBFs and DOBFs.

3.5.2 Generation of Difference Boolean Functions or DBFs for a certain ID.

The DBFs of each OPBF have been generated by bitwise Xor of OPBFs and the corresponding COPBFs. The corresponding DBFs of OBPF4, OBPF3, OBPF2, OPBF1 are denoted as DIFF4, DIFF3, DIFF2, DIFF1 respectively. Generation of 4th DBF of ID '1011' has been shown in column 1 through G of row 3 of Table.20.

R|C		1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G
1	**OPBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
2	**COPBF**	1	0	1	0	0	0	1	0	0	1	0	1	1	1	1	0
3	**DIFF**	0	0	0	0	0	1	0	1	0	0	0	0	1	0	1	0

Table.20. DBF Generation.

3.5.3 Analysis:

If the DBFs are balanced then the number of bits changed and remains unchanged among corresponding bits of OPBFs and COPBFs is maximum. So uncertainty of determining a particular change in bits is maximum. As the number of balanced DBFs are increased among 64 (=2^4×4) possible DBFs then the security will increase. The number of 1s or balanced-ness of the above DBF shown from row 1 through G of row 3 of table.20. has been shown in column. 2. of row 2 of table.21.

R|C	1	2
1	**Difference BF**	**Total Number of 1s**
2	DIFF	4

Table. 21. Balanced-ness of DBFs.

3.5.4 DBFs Generation and Derivation of a Particular Row of Differential Analysis Table (DAT) for a Certain ID.

Four IPVs in the order IPV4, IPV3, IPV2 and IPV1 for the S-box given in Table.1. and four CIPVs, CIPV4, CIPV3, CIPV2 and CIPV1 for a certain ID '1011' have been shown from column 1 through G of row 1, 2, 3, 4, 5, 6, 7 and 8 respectively in Table.22. Four OPBFs in the order OPBF4, OPBF3, OPBF2 and OPBF1 for the S-box given in Table.11. and four COPBFs COPBF4, COPBF3, COPBF2 and COPBF1 for a certain ID '1011' have been shown from column 1 through G of row 9, A, B, C, D, E, F and G respectively in Table.22. The resultant DBFs, DIFF4, DIFF3, DIFF2, DIFF1, have been shown in column 1 through G of row H, I, J, K of Table.22. The number of 1s or Balanced-ness of four DBFs have been shown in row from column.2 through 5 of row 1 in Table.23.

Row|Col		1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G
1	**IBF4**	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
2	**IBF3**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
3	**IBF2**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
4	**IBF1**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
5	**CIBF4**	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
6	**CIBF3**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
7	**CIBF2**	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
8	**CIBF1**	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
9	**OBF4**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
A	**OBF3**	1	1	1	0	0	1	0	0	0	0	1	1	1	0	0	1
B	**OBF2**	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	1
C	**OBF1**	0	0	1	1	0	1	1	0	1	0	0	0	1	1	0	1
D	**COBF4**	1	0	1	0	0	0	1	0	0	1	0	1	1	1	1	0
E	**COBF3**	1	1	0	0	1	0	0	1	0	1	1	1	0	0	1	0
F	**COBF2**	0	1	1	1	1	0	0	0	0	0	0	1	0	1	1	1
G	**COBF1**	0	0	0	1	1	0	1	1	1	1	0	0	0	1	1	0
H	**DIFF4**	0	0	0	0	0	1	0	1	0	0	0	0	1	0	1	0
I	**DIFF3**	0	0	1	0	1	1	0	1	0	1	0	0	1	0	1	1
J	**DIFF2**	1	1	1	1	0	1	1	0	1	1	1	1	0	1	1	0
K	**DIFF1**	0	0	1	0	1	1	0	1	0	1	0	0	1	0	1	1

Table. 22. Generation of a Particular Row of Differential Analysis Table (DAT).

R|C	1	2	3	4	5
	Difference BFs	**DIFF4**	**DIFF3**	**DIFF2**	**DIFF1**
1	**No. of ones.**	4	8	C	8

Table.23. Balanced-ness of four DBFs.

3.5.5. Differential Analysis Table or DAT. The Balanced-ness of four DBFs for each ID have been shown from column 2 through 5 of row 2 through H of DAT or Table.24.

R|C	**1**	**2**	**3**	**4**	**5**
1	**ID in Hex**	**DIFF1**	**DIFF2**	**DIFF3**	**DIFF4**
2	0	0	0	0	0
3	1	8	8	8	C
4	2	C	8	C	4
5	3	8	8	8	C
6	4	8	C	8	8
7	5	8	8	8	8
8	6	C	8	C	8
9	7	8	C	8	8
A	8	C	C	C	C
B	9	8	8	8	8
C	10	4	8	4	C
D	11	8	C	8	4
E	12	C	4	C	8
F	13	8	8	8	8
G	14	4	8	4	8
H	15	8	4	8	8

Table.24. DAT for 1st 4-bit S-box of 1st S-box of DES

3.5.6 Pseudo Code for Differential Cryptanalysis of 4-bit Crypto S-boxes and its Time Complexity Analysis.

The Pseudo Code has been given as follows,

```
Start. // Start of Pseudo Code
// Variable Declarations, One Dimensional Array ISB[16] is for for Input S-box in Hex, IDIFF[16] is for Input Difference in Hex, Three Dimensional Array ODIFF[16][16] is for all Output Difference in Hex for 16 IDIFFs. Bin_ODIFF[4][16][16] is for all 4-bit bit patterns of Output Difference for 16 IDIFFs.
Step 0A: int ISB[16]; int IDIFF[16]; int ODIFF[16][16];
// Variable Declarations, ISB'[16][16] is for All elements in Hex of 16 distant ISBs. OSB[16] is for elements in Hex of the given S-box or Output S-box , OSB'[16][16] is for All elements in Hex of 16 distant OSBs, DAT[16][16] is for Difference Analysis Table, and Count[16] is for count of each element in ODIFF for 16 OSBs.
Step 0B: int ISB'[16][16]; int OSB[16]; int OSB'[16][16]; int DAT[4][16]; int Count[16].
// Differential Cryptanalysis block
Step 01: For I =1:16; For J =1:16; // For Loop I and J respectively.
                ISB'[I][J] = ISB[J]^IDIFF[I];
                OSB'[I][J] = OSB[ISB'[I][J]];
                ODIFF[I][J] = OSB[J]^ OSB'[I][J];
                For K=1:4 Bin_ODIFF[K][I][J] = Hex to Binary(ODIFF[I][J])
         End For J. End For I.// End of For Loop J and I respectively.
Step 03: For I =1:4; For J =1:16; For K = 1:16 // For Loop I and J respectively.
                DAT[I][J]= Count[Bin_ODIFF[I][J][K]];
         End For J. End For I. // End of For Loop J and I respectively.
Stop. // End of Pseudo Code
```

Time Complexity of the Given Algorithm. Since Differential Cryptanalysis block contains 2 nested loops so the time Complexity of the Algorithm has been O(n^2).

3.5.7 Comparison of Time Complexity of Two views of Differential Cryptanalysis of 4-bit S-boxes and Differential Cryptanalysis with 4-bit BFs.

The Comparison of time complexity of three algos has been given in Table.25 as follows,

View	4-bit BP View	S-box View	With 4-bit BFs
Time Complexity	O(n^3)	O(n^2)	O(n^2)

Table.25. Time Complexity Comparison of Three Algos.

It can be concluded from the comparison that the Execution Time reduces in S-box view and With 4-bit BFs than the 4-bit Binary Pattern view. So in can be concluded from above review work and new algorithm that the execution time of Differential Cryptanalysis depends upon the view of the algorithm and the S-box view has been proved to be a better algorithm than 4-bit binary pattern view algorithm. The With 4-bit BFs Algo has also been proved to be the better one since The DAT table construction is less time consuming than DDT construction since DDT constitutes of 256 entries while DAT constitutes of 64 entries so in can also be concluded from comparison that Differential Cryptanalysis with 4-bit BFs has been proven to be the best algorithm among 3 Algorithms since it takes less execution time among three algorithms.

3.6 A Brief Review of Linear Cryptanalysis of 4-bit Crypto S-boxes and a new Technique With Boolean Functions for Linear Cryptanalysis of 4-bit Crypto S-boxes or Linear Approximation Analysis. The review of related relevant property of 4-bit BFs, Algebraic Normal form of 4-bit BFs has been illustrated in subsec.3.6.1. The review of Linear Cryptanalysis of 4-bit Crypto S-boxes has been described in brief in subsec.3.6.2. At last the new technique to analyze 4-bit S-boxes by 4-bit Linear Approximations or Linear Approximation Analysis has been described in brief in subsec. 3.6.3.

3.6.1 A review of Boolean Functions (BF) and its Algebraic Normal Form (ANF)

A 4-bit Boolean Function (BF) accepts 4 bits as input $\{x_1x_2x_3x_4\}$having 16 combinations of decimal values varying between 0 and 15 and provides 1-bit output for each combination of input. The input-output relation is given in a Truth Table which provides 16-bit output vector corresponding to four 16-bit input vectors, each one attached to x_1, x_2, x_3 and x_4. The 4-bit BF is a mapping from $(0,1)^4$ to $(0,1)^1$and its functional relation, F(x) can be expressed in Algebraic Normal Form (ANF) with 16 coefficients as given in eq. (1) below,

$$F(x) = a_0 + (a_1.x_1 + a_2.x_2 + a_3.x_3 + a_4.x_4) + (a_5.x_1.x_2 + a_6.x_1.x_3 + a_7.x_1.x_4 + a_8.x_2.x_3 + a_9.x_2.x_4 + a_{10}.x_3.x_4) + \\ + (a_{11}.x_1.x_2.x_3 + a_{12}.x_1.x_2.x_4 + a_{13}.x_1.x_3.x_4 + a_{14}.x_2.x_3.x_4) + a_{15}.x_1.x_2.x_3.x_4 \quad \ldots \quad \ldots \quad (1)$$

where x represents the decimal value or the hex value of 4 input bits represented by $\{x_1x_2x_3x_4\}$, BF assumes 1-bit output, '**.**' 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 non-linear coefficients of 6 terms with 2-AND-operated-input-bits, (a_{11} to a_{14}) are 4 nonlinear coefficients of 4 terms with 3-AND-operated-input-bits and a_{15} is a non-linear coefficient of one term with 4-AND-operated-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.(2),

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;
0anf.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)

The DEBF (Decimal Equivalent of BF) 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 output of a BF, the ANF coefficients from anf.bit0 through anf.bit15 are calculated sequentially using eq. (2).

3.6.2 A Review on Linear Cryptanalysis of 4-bit Crypto S-boxes [HH96][HH02]. The given 4-bit Crypto S-box has been described in sub-section 3.6.2.1. The relation of 4-bit S-boxes with 4 bit BFs and with Linear Approximations are described in sub-section 3.6.2.2 and 3.6.2.3 respectively. LAT or Linear Approximation Table has also been illustrated in sec 3.6.2.4. Algorithm of Linear Cryptanalysis with Time Complexity Analysis has been described in sec. 3.6.2.5.

3.6.2.1. 4-bit Crypto S-boxes: A 4-bit Crypto S-box can be written as Follows in Table.26, where the each element of the first row of Table.26, 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 Substitution box. It can be concluded that the 1st row is fixed for all possible 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 [AT90][NT77][NT99].

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

Table.26. 4-bit Crypto S-box.

3.6.2.2. 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.27. 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 is a 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 [AT90]. The decimal equivalent of each IPV and OPBF are noted at column H of respective rows.

Row	**Column**	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	**H.** Decimal
1	**Index**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**	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.27. Decomposition of 4-bit input S-box and given S-box (1st 4-bit S-box of 1st S-box out of 8 of DES) to 4-bit BFs.

3.6.2.3. 4-bit Linear Relations. The elements of input S-box have been shown under column heading 'I' and the Input Vectors have been shown under field IPVs (Input Vectors) and subsequently under column headings 1, 2, 3 and 4. The 4th input vector has been depicted under column heading '4', 3rd input vector has been depicted under column heading '3', 2nd input vector has been depicted under column heading '2' and 1st input vector has been depicted under column heading '1'. The elements of S-box have been shown under column heading 'SB' and the Output 4-bit BFs are shown under field OPBFs (Output Boolean Functions) and subsequently under column headings 1, 2, 3 and 4. The 4th Output BF has been depicted under column heading '4', 3rd Output BF has been depicted under column heading '3', 2nd Output BF has been depicted under column heading '2' and 1st Output BF has been depicted under column heading '1' of table.28.

I	**IPVs**				**S**	**OPBFs**			
	4	**3**	**2**	**1**	**B**	**4**	**3**	**2**	**1**
0	0	0	0	0	**E**	1	1	1	0
1	0	0	0	1	**4**	0	1	0	0
2	0	0	1	0	**D**	1	1	0	1
3	0	0	1	1	**1**	0	0	0	1
4	0	1	0	0	**5**	0	1	0	1
5	0	1	0	1	**9**	1	0	0	1
6	0	1	1	0	**0**	0	0	0	0
7	0	1	1	1	**7**	0	1	1	1
8	1	0	0	0	**2**	0	0	1	0
9	1	0	0	1	**F**	1	1	1	1

A	1	0	1	0	**B**	1	0	1	1
B	1	0	1	1	**8**	1	0	0	0
C	1	1	0	0	**3**	0	0	1	1
D	1	1	0	1	**A**	1	0	1	0
E	1	1	1	0	**6**	0	1	1	0
F	1	1	1	1	**C**	1	1	0	0

Table. 28. IPVs and OPBFs for given S-box

The IPEs or Input Equations are all possible xored terms that can be formed using four IPVs 4, 3, 2 and 1. On the other hand OPEs are possible xored terms that can be formed using four OPVs 4, 3, 2 and 1. All possible IPEs and OPEs are listed under the column and also row heading (IPE = OPE) from row 2 through H and column 1 through G respectively. Each cell is a linear equation equating IPE to OPE. Such as $L_{1+2+4,2+3}$ is the linear equation formed by IPE '1+2+3' i.e. the xored combination of three IPVs 1, 2 and 4 and OPE '2+3' i.e. the xored combination of two OPBFs 2 and 3. The 256 possible 4-bit Linear Equations are shown in Table 29.

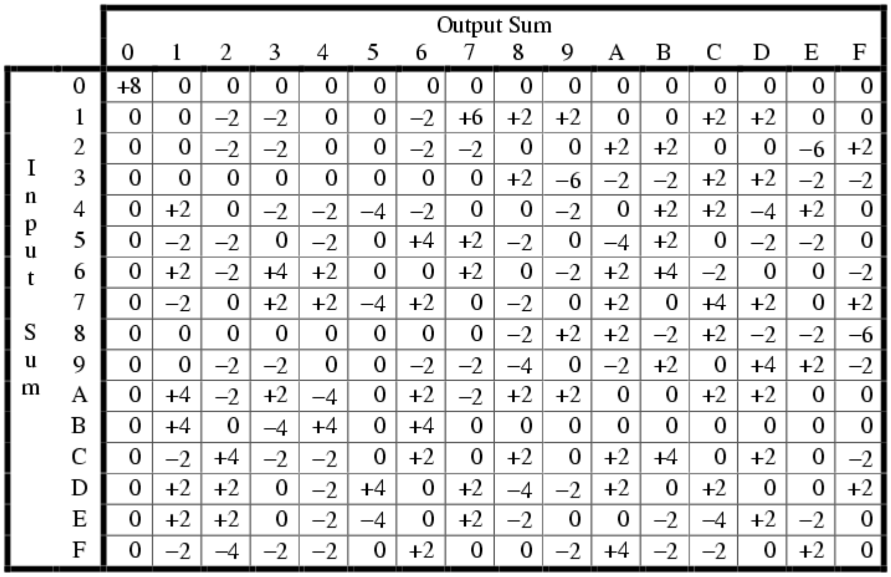
Rows	**Columns**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**
1	**IPE = OPE**	**0**	**1**	**2**	**3**	**4**	**1+2**	**1+3**	**1+4**	**2+3**	**2+4**	**3+4**
2	**0**	$L_{0,0}$	$L_{0,1}$	$L_{0,2}$	$L_{0,3}$	$L_{0,4}$	$L_{0,1+2}$	$L_{0,1+3}$	$L_{0,1+4}$	$L_{0,2+3}$	$L_{0,2+4}$	$L_{0,3+4}$
3	**1**	$L_{1,0}$	$L_{1,1}$	$L_{1,2}$	$L_{1,3}$	$L_{1,4}$	$L_{1,1+2}$	$L_{1,1+3}$	$L_{1,1+4}$	$L_{1,2+3}$	$L_{1,2+4}$	$L_{1,3+4}$
4	**2**	$L_{2,0}$	$L_{2,1}$	$L_{2,2}$	$L_{2,3}$	$L_{2,4}$	$L_{2,1+2}$	$L_{2,1+3}$	$L_{2,1+4}$	$L_{2,2+3}$	$L_{2,2+4}$	$L_{2,3+4}$
5	**3**	$L_{3,0}$	$L_{3,1}$	$L_{3,2}$	$L_{3,3}$	$L_{3,4}$	$L_{3,1+2}$	$L_{3,1+3}$	$L_{3,1+4}$	$L_{3,2+3}$	$L_{3,2+4}$	$L_{3,3+4}$
6	**4**	$L_{4,0}$	$L_{4,1}$	$L_{4,2}$	$L_{4,3}$	$L_{4,4}$	$L_{4,1+2}$	$L_{4,1+3}$	$L_{4,1+4}$	$L_{4,2+3}$	$L_{4,2+4}$	$L_{4,3+4}$
7	**1+2**	$L_{1+2,0}$	$L_{1+2,1}$	$L_{1+2,2}$	$L_{1+2,3}$	$L_{1+2,4}$	$L_{1+2,1+2}$	$L_{1+2,1+3}$	$L_{1+2,1+4}$	$L_{1+2,2+3}$	$L_{1+2,2+4}$	$L_{1+2,3+4}$
8	**1+3**	$L_{1+3,0}$	$L_{1+3,1}$	$L_{1+3,2}$	$L_{1+3,3}$	$L_{1+3,4}$	$L_{1+3,1+2}$	$L_{1+3,1+3}$	$L_{1+3,1+4}$	$L_{1+3,2+3}$	$L_{1+3,2+4}$	$L_{1+3,3+4}$
9	**1+4**	$L_{1+4,0}$	$L_{1+4,1}$	$L_{1+4,2}$	$L_{1+4,3}$	$L_{1+4,4}$	$L_{1+4,1+2}$	$L_{1+4,1+3}$	$L_{1+4,1+4}$	$L_{1+4,2+3}$	$L_{1+4,2+4}$	$L_{1+4,3+4}$
A	**2+3**	$L_{2+3,0}$	$L_{2+3,1}$	$L_{2+3,2}$	$L_{2+3,3}$	$L_{2+3,4}$	$L_{2+3,1+2}$	$L_{2+3,1+3}$	$L_{2+3,1+4}$	$L_{2+3,2+3}$	$L_{2+3,2+4}$	$L_{2+3,3+4}$
B	**2+4**	$L_{2+4,0}$	$L_{2+4,1}$	$L_{2+4,2}$	$L_{2+4,3}$	$L_{2+4,4}$	$L_{2+4,1+2}$	$L_{2+4,1+3}$	$L_{2+4,1+4}$	$L_{2+4,2+3}$	$L_{2+4,2+4}$	$L_{2+4,3+4}$
C	**3+4**	$L_{3+4,0}$	$L_{3+4,1}$	$L_{3+4,2}$	$L_{3+4,3}$	$L_{3+4,4}$	$L_{3+4,1+2}$	$L_{3+4,1+3}$	$L_{3+4,1+4}$	$L_{3+4,2+3}$	$L_{3+4,2+4}$	$L_{3+4,3+4}$
D	**1+2+3**	$L_{1+2+3,0}$	$L_{1+2+3,1}$	$L_{1+2+3,2}$	$L_{1+2+3,3}$	$L_{1+2+3,4}$	$L_{1+2+3,1+2}$	$L_{1+2+3,1+3}$	$L_{1+2+3,1+4}$	$L_{1+2+3,2+3}$	$L_{1+2+3,2+4}$	$L_{1+2+3,3+4}$
E	**1+2+4**	$L_{1+2+4,0}$	$L_{1+2+4,1}$	$L_{1+2+4,2}$	$L_{1+2+4,3}$	$L_{1+2+4,4}$	$L_{1+2+4,1+2}$	$L_{1+2+4,1+3}$	$L_{1+2+4,1+4}$	$L_{1+2+4,2+3}$	$L_{1+2+4,2+4}$	$L_{1+2+4,3+4}$
F	**1+3+4**	$L_{1+3+4,0}$	$L_{1+3+4,1}$	$L_{1+3+4,2}$	$L_{1+3+4,3}$	$L_{1+3+4,4}$	$L_{1+3+4,1+2}$	$L_{1+3+4,1+3}$	$L_{1+3+4,1+4}$	$L_{1+3+4,2+3}$	$L_{1+3+4,2+4}$	$L_{1+3+4,3+4}$
G	**2+3+4**	$L_{2+3+4,0}$	$L_{2+3+4,1}$	$L_{2+3+4,2}$	$L_{2+3+4,3}$	$L_{2+3+4,4}$	$L_{2+3+4,1+2}$	$L_{2+3+4,1+3}$	$L_{2+3+4,1+4}$	$L_{2+3+4,2+3}$	$L_{2+3+4,2+4}$	$L_{2+3+4,3+4}$
H	**1+2+3+4**	$L_{1+2+3+4,0}$	$L_{1+2+3+4,1}$	$L_{1+2+3+4,2}$	$L_{1+2+3+4,3}$	$L_{1+2+3+4,4}$	$L_{1+2+3+4,1+2}$	$L_{1+2+3+4,1+3}$	$L_{1+2+3+4,1+4}$	$L_{1+2+3+4,2+3}$	$L_{1+2+3+4,2+4}$	$L_{1+2+3+4,3+4}$

Rows	**Columns**	**C**	**D**	**E**	**F**	**G**
1	**IPE=OPE**	**1+2+3**	**1+2+4**	**1+3+4**	**2+3+4**	**1+2+3+4**
2	**0**	$L_{0,1+2+3}$	$L_{0,1+2+4}$	$L_{0,1+3+4}$	$L_{0,2+3+4}$	$L_{0,1+2+3+4}$
3	**1**	$L_{1,1+2+3}$	$L_{1,1+2+4}$	$L_{1,1+3+4}$	$L_{1,2+3+4}$	$L_{1,1+2+3+4}$
4	**2**	$L_{2,1+2+3}$	$L_{2,1+2+4}$	$L_{2,1+3+4}$	$L_{2,2+3+4}$	$L_{2,1+2+3+4}$
5	**3**	$L_{3,1+2+3}$	$L_{3,1+2+4}$	$L_{3,1+3+4}$	$L_{3,2+3+4}$	$L_{3,1+2+3+4}$
6	**4**	$L_{4,1+2+3}$	$L_{4,1+2+4}$	$L_{4,1+3+4}$	$L_{4,2+3+4}$	$L_{4,1+2+3+4}$
7	**1+2**	$L_{1+2,1+2+3}$	$L_{1+2,1+2+4}$	$L_{1+2,1+3+4}$	$L_{1+2,2+3+4}$	$L_{1+2,1+2+3+4}$
8	**1+3**	$L_{1+3,1+2+3}$	$L_{1+3,1+2+4}$	$L_{1+3,1+3+4}$	$L_{1+3,2+3+4}$	$L_{1+3,1+2+3+4}$
9	**1+4**	$L_{1+4,1+2+3}$	$L_{1+4,1+2+4}$	$L_{1+4,1+3+4}$	$L_{1+4,2+3+4}$	$L_{1+4,1+2+3+4}$
A	**2+3**	$L_{2+3,1+2+3}$	$L_{2+3,1+2+4}$	$L_{2+3,1+3+4}$	$L_{2+3,2+3+4}$	$L_{2+3,1+2+3+4}$
B	**2+4**	$L_{2+4,1+2+3}$	$L_{2+4,1+2+4}$	$L_{2+4,1+3+4}$	$L_{2+4,2+3+4}$	$L_{2+4,1+2+3+4}$
C	**3+4**	$L_{3+4,1+2+3}$	$L_{3+4,1+2+4}$	$L_{3+4,1+3+4}$	$L_{3+4,2+3+4}$	$L_{3+4,1+2+3+4}$
D	**1+2+3**	$L_{1+2+3,1+2+3}$	$L_{1+2+3,1+2+4}$	$L_{1+2+3,1+3+4}$	$L_{1+2+3,2+3+4}$	$L_{1+2+3,1+2+3+4}$
E	**1+2+4**	$L_{1+2+4,1+2+3}$	$L_{1+2+4,1+2+4}$	$L_{1+2+4,1+3+4}$	$L_{1+2+4,2+3+4}$	$L_{1+2+4,1+2+3+4}$
F	**1+3+4**	$L_{1+3+4,1+2+3}$	$L_{1+3+4,1+2+4}$	$L_{1+3+4,1+3+4}$	$L_{1+3+4,2+3+4}$	$L_{1+3+4,1+2+3+4}$
G	**2+3+4**	$L_{2+3+4,1+2+3}$	$L_{2+3+4,1+2+4}$	$L_{2+3+4,1+3+4}$	$L_{2+3+4,2+3+4}$	$L_{2+3+4,1+2+3+4}$
H	**1+2+3+4**	$L_{1+2+3+4,1+2+3}$	$L_{1+2+3+4,1+2+4}$	$L_{1+2+3+4,1+3+4}$	$L_{1+2+3+4,2+3+4}$	$L_{1+2+3+4,1+2+3+4}$

Table.29. 256, 4-bit Linear Equations with input Equations (IPE) and output Equations (OPE).

3.6.2.4 Linear Approximation Table (LAT) [6].

According to Heys each linear equation is tested for each of 16 4-bit patterns shown in each row under the field IPVs and subsequently under the column headings 1, 2, 3 and 4 and the corresponding 16 4-bit patterns under field OPBFs and subsequently under the column headings 1, 2, 3 and 4. If a linear equation satisfies 8 times out of 16 then the existence of the linear equation is highly unpredictable. That is the probability is ½. If the numbers of satisfaction of each linear equation is noted in respective cells of Table.20. then it is called as Linear Approximation Table or LAT. The Linear Approximation Table for the given S-box has been shown in table.30.



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

Table.30. Linear Approximation Table (LAT) for given S-box

3.6.2.5 Pseudo Code of Algorithm with Time Complexity Analysis of Linear Cryptanalysis of 4-bit Crypto S-boxes. The algorithm to execute the linear cryptanalysis for 4-bit Crypto S-boxes following Heys [HH96][HH02] considers 4–bit Boolean variables Ai and Bj whose i and j are the decimal indices varying from 0 to 15 and Ai and Bj are taking corresponding bit values from [0000] to [1111]. The algorithm to fill the (16 x 16) elements of the LAT is,

```
for(i=0;i<16;i++){
        A=0;
        for(k=0;k<16;k++) A=A+(Ai0.Xk0+Ai1.Xk1+Ai2.Xk2+Ai3.Xk3)%2;
        for(j=0;j<16;j++){
                B=0;
                for(k=0;k<16;k++)B= B+(Bj0.Yk0+Bj1.Yk1+Bj2.Yk2+Bj3.Yk3)%2;
                Sij  = (A+B)%2;
                if (Sij==0) Cij++; Nij  = Cij – 8;

        }
}
```

Time Complexity of the given Algorithm. Since the Pseudo Code contains two nested loops so the time complexity of the given algorithm has been $O(n^2)$.

3.7 Linear Approximation Analysis:

A Crypto 4-bit S-box (1st 4-bit S-box out of 32 4-bit S-boxes of DES) has been described in sub-section 3.7.1. The Table for four input vectors, Output 4-bit BFs and corresponding ANFs has been depicted in sub-section 3.7.2. The analysis has been described in sub-section 3.7.3. The result of Analysis has been given in sub-section 3.7.4.

3.7.1 4-bit Crypto S-boxes: A 4-bit Crypto S-box can be written as Follows in Table.31, where the each element of the first row of Table.31, 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 Substitution box. It can be concluded that the 1st row is fixed for all possible 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 [AT90][NT77][NT99].

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

Table.31. 4-bit Crypto S-box.

3.7.2 Input Vectors (IPVs)-Output BFs (OPBFs)-Algebraic Normal Forms (ANFs). The elements of input S-box have been shown under column heading 'ISB' and the Input Vectors have been shown under the field IPVs (Input Vectors) and subsequently under column headings 1, 2, 3 and 4. The 4th input vector has been depicted under column heading '4', 3rd input vector has been depicted under column heading '3', 2nd input vector has been depicted under column heading '2' and 1st input vector has been depicted under column heading '1'. The elements of S-box have been shown under column heading 'OSB' and the Output 4-bit BFs have been shown under field OPBFs (Output Boolean Functions) and subsequently under column headings 1, 2, 3 and 4. The 4th Output BF has been depicted under column heading '4', 3rd Output BF has been depicted under column heading '3', 2nd Output BF has been depicted under column heading '2' and 1st Output BF has been depicted under column heading '1'. The corresponding ANFs for 4 OPBFs, OPBF-4th , OPBF-3rd, OPBF-2nd , OPBF-1st, are depicted under field 'ANFs' subsequently under column heading 4, 3, 2 and 1 respectively of Table.32..

ISB	IPVs	**OSB**	OPBFs	ANFs
	4321		**4321**	**4321**
0	0000	E	1110	1110
1	0001	4	0100	1010
2	0010	D	1101	0011
3	0011	1	0001	1100
4	0100	2	0010	1101
5	0101	F	1111	0110
6	0110	B	1011	0111
7	0111	8	1000	0011
8	1000	3	0011	1010
9	1001	A	1010	0110
A	1010	6	0110	1010
B	1011	C	1100	1000
C	1100	5	0101	0101
D	1101	9	1001	0010
E	1110	0	0000	1010
F	1111	7	0111	0000

Table32. Input and Output Boolean Functions
With Corresponding ANF Coefficients of the given S-box.

3.7.3 Linear Approximation Analysis (LAA). An Algebraic Normal Form or ANF equation is termed as Linear Equation or Linear Approximation if the Nonlinear Part or NP (i.e. The xored value of all product terms of equation 2 for corresponding 4 bit values of IPVs, with column heading 4, 3, 2, 1) is 0 and The Linear part or LP for corresponding 4 bit values of IPVs, with column heading 4, 3, 2, 1 is equal to corresponding BF bit values. The corresponding ANF coefficients of output BFs F(4), F(3), F(2), and F(1) are given under row heading ANF(F4), ANF(F3), ANF(F2) and ANF(F1) respectively from row 2 through 5 and column 4 through J. In which Column 4 of row 2 through 5 gives the value of Constant Coefficient (a_0 according to eqn.2.) of ANF(F4), ANF(F3), ANF(F2) and ANF(F1) respectively. Column 5 through 8 of row 2 through 5 gives the value of respective Linear Coefficients more specifically a_1, a_2, a_3, a_4 (according to eqn. 2.) of ANF(F4), ANF(F3), ANF(F2) and ANF(F1). They together termed as LP or Linear Part of the respective ANF Equation. Column 9 through J of row 2 through 5 gives the value of respective Non-Linear Coefficients more specifically a_5 to a_{15} (according to eqn. 2.) of ANF(F4), ANF(F3), ANF(F2) and ANF(F1). They together termed as NP or Non-Linear Part of the respective ANF Equation.

The 4th, 3rd, 2nd, 1st IPV for the given S-box have been noted in the Field 'IPVs' under column heading 4, 3, 2, 1 respectively from row 8 through M of Table.23. The 4 output BFs F4, F3, F2, F1 are noted at column 4, 8, C, G from row 8 through M respectively. The corresponding LP, NP, Satisfaction (SF) values (LP = BF)are noted at column 5 through 7, 9 through B, C through F and H to J from row 8 through M respectively of Table.33.

R|C	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H	I	J
1	**Co-Effs**			**C**	**LP**				**NP**										
2	**ANF(F4)**			1	1	0	1	1	0	0	0	1	0	1	1	0	0	1	0
3	**ANF(F3)**			1	0	0	1	1	1	1	0	0	1	0	0	1	0	0	0
4	**ANF(F2)**			1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	0
5	**ANF(F1)**			0	0	1	0	1	0	1	1	0	0	0	0	1	0	0	0
6	**I**	**IPVs**	**S**	**F**	**L**	**N**	**S**	**F**	**L**	**N**	**S**	**F**	**L**	**N**	**S**	**F**	**L**	**N**	**S**
7	**D**	**4321**	**B**	**4**	**P**	**P**	**F**	**3**	**P**	**P**	**F**	**2**	**P**	**P**	**F**	**1**	**P**	**P**	**F**
8	0	0000	E	1	0	0	1	1	0	0	1	1	1	0	0	0	1	0	1
9	1	0001	4	0	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0
A	2	0010	D	1	1	0	0	1	1	0	0	0	1	0	1	1	0	0	1
B	3	0011	1	0	1	1	1	0	1	0	1	0	1	1	1	1	1	0	0
C	4	0100	2	0	1	0	1	0	1	0	1	1	0	0	1	0	1	0	1
D	5	0101	F	1	0	0	1	1	0	1	1	1	0	1	1	1	1	0	0
E	6	0110	B	1	1	1	1	0	1	0	1	1	0	1	1	1	1	0	0
F	7	0111	8	1	0	1	1	0	0	1	1	0	0	0	0	0	1	0	1
G	8	1000	3	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	1
H	9	1001	A	1	1	0	0	0	0	0	0	1	0	1	1	0	0	1	1
R|C	**I D**	**IPVs 4321**	**S B**	**F 4**	**L P**	**N P**	**S F**	**F 3**	**L P**	**N P**	**S F**	**F 2**	**L P**	**N P**	**S F**	**F 1**	**L P**	**N P**	**S F**
I	A	1010	6	0	0	0	0	1	1	1	1	1	0	1	1	0	0	1	1
J	B	1011	C	1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0
K	C	1100	5	0	0	0	0	1	1	1	1	0	1	1	1	1	1	0	0
L	D	1101	9	1	1	0	0	0	0	1	1	0	1	1	1	1	1	0	0
M	E	1110	0	0	0	0	0	0	1	0	1	0	1	1	1	0	1	1	1
N	F	1111	7	0	1	0	1	1	0	0	1	1	1	0	0	1	1	1	1

Table.33. Linear Approximation Analysis

3.7.4. Result

No. of LA with BF1	No. of LA with BF2	No. of LA with BF3	No. of LA with BF4
7	4	2	8

Total Number of Existing Linear Approximations: 21.

3.7.5 Pseudo Code with Time Complexity Analysis of the Linear Approximation Analysis Algorithm: The Nonlinear Part for the given analysis has been termed as NP. The ANF coefficients are illustrated through array anf[16]. IPVs are termed as x_1, x_2, x_3, x_4 for IPV1, IPV2, IPV3, IPV4 respectively. The Pseudo Code of algorithm of the above analysis is given below,

```
Start.
Step 1. NP = (anf[5].&$x_1$&$x_2$)^(anf[6]&$x_1$ &$x_3$)+( anf[7]&$x_1$ &$x_4$)+(anf[8] &$x_2$ &$x_3$)+(anf[9]&$x_2$ &$x_4$)+(anf[10]&$x_3$ &$x_4$)(anf[11]&$x_1$ &$x_2$ &$x_3$)+(anf[12]&$x_1$ &$x_2$ &$x_4$)+(anf[13]&$x_1$ &$x_3$ &$x_4$) +(anf[14] &$x_2$ &$x_3$ &$x_4$)+(anf[15]&$x_1$ &$x_2$ &$x_3$ &$x_4$))
Step 2. LP= anf[0] ^(anf[1].&$x_1$)^ (anf[2].&$x_2$)^ (anf[3].&$x_3$)^ (anf[4].&$x_4$).
Step 3. if(NP==0&& BF($x_1x_2x_3x_4$) == LP) then Linear equation.
        else Nonlinear equation.
Stop.
```

Time Complexity. Since the analysis contains no loops so the Time complexity of the algorithm has been O(n).

3.7.6. Comparison of Execution time Complexity of Linear Cryptanalysis of 4-bit Crypto S-boxes and Linear Approximation Analysis of 4-bit S-boxes. The Comparison of time complexity of two algorithms has been given in Table.34 as follows,

View	4-bit LC	4-bit LA
Time Complexity	O(n^2)	O(n)

Table.34. Time Complexity Comparison of Two Algos.

It can be concluded from the comparison that the Execution time reduces in Linear Approximation Analysis than the Linear Cryptanalysis of 4-bit Crypto S-boxes. So in can be concluded from above review work that the execution time of 4-bit LA Algorithm is much less that 4-bit LC Algorithm so 4-bit LA algorithm has been proved to be much better algorithm.

4. S-box Generation. In this section polynomials over Galois Field GF(p^q) and roll of IPs to construct substitution boxes have been reviewed in subsec. 4.1 of section.4. The generation of 4 and 8 bit S-boxes using BCNs have been elaborated in subsec 4.2 of section 4.. The generation of 4-bit and 8-bit S-boxes with Coefficients of non-binary Galois Field Polynomials has been depicted in subsec.4.3 of section 4. The cryptographic and security analysis of 32 DES 4-bit S-boxes has been given in subsec.4.4 of sec.4. Detailed cryptographic and security analysis of generated 10 4-bit crypto S-boxes with discussed crypto related cryptographic properties and security criterion have also been given in subsec.4.4. of sec.4. Results have been discussed in Result and Discussion section in subsec.4.5 of sec.4.

4.1. Polynomials over Galois field GF(p^q) and log $_2$ $^{q+1}$ bit S-boxes. In this section the sub section 4.1.1. has been devoted to a small review of Polynomials. The sub section 4.1.2. has been of utmost importance since in it a four bit crypto or proper S-box has been defined in brief. At last in sub section 4.1.3. The equation among 2^{15} Galois field Polynomials and a 4-bit crypto S-box has been elaborated in details.

4.1.1. Polynomials over Galois field GF(p^q). Polynomials over Galois field GF(p^q) have been of utmost importance in cryptographic applications. Polynomials with degree q have been termed as Basic Polynomials over Galois field GF(p^q) and Polynomials with degree less than q have been termed as Elemental Polynomials over Galois field GF(p^q). Polynomials with leading coefficient as 1 have been termed as Monic Polynomials irrespective of BPs and EPs over Galois field GF(p^q). An example, of the said criteria have been described as follows, the Example of Basic Polynomial or BP over Galois field GF(p^q) has been given below,

$$BP(x) = co_q\, x^q + co_{q-1}\, x^{q-1} + co_{q-1}\, x^{q-2} + \ldots\ldots\ldots\ldots\ldots\ldots + co_2\, x^2 + co_1\, x^1 + a_0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{(i)}$$

In equation (i) BP(x) has been represented as Basic Polynomial or BP over Galois field GF(p^q) since the highest degree term of the said polynomial over Galois field GF(p^q) has been q. The BP has been called as a Monic BP over Galois field GF(p^q) if $co_{q\,=}$ 1. The number of Terms in a BP over Galois field GF(p^q) has been (q+1). The number of possible values of a particular coefficient $co_{q,}$ where $0 \le p \le q$ has been from 0 to p i.e. (p+1). If the value of q has been <q then The polynomial over Galois field GF(p^q) has been termed as Elemental Polynomial or EPs over Galois field GF(p^q). If a BP or EP contains only constant term then the polynomial has been termed as Constant Polynomial or CP over Galois field GF(p^q). If a BP over Galois field GF(p^q) can be factored into two non-constant EPs then the BP can be termed as Reducible Polynomials or RPs over Galois field GF(p^q). If the two factor of a BP over Galois field GF(p^q) have been the BP itself and a constant Polynomial or CP then The BP have been said as an Irreducible Polynomial or IP over Galois field GF(p^q).

4.1.2. 4-bit Crypto S-boxes: A 4-bit crypto S-box can be written as Follows, where the each element of the first row of Table.35, 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 Substitution box. It can be concluded that the 1st row is fixed for all possible crypto S-boxes. The values of each element of the 1st row are distinct, unique and vary between 0 and F. The values of the each element of the 2nd row of a crypto S-box have also been distinct and unique and also vary between 0 and F. 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 [AT90][NT77][NT99].

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

Table.35. 4-bit bijective Crypto S-box.

4.1.3 Relation between 4-bit S-boxes and Polynomials over Galois field GF (2^{15}). 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. From row 2 through 5 and row 7 through A of each column from 1 through G of Table.36. shows the 4-bit bit sequences of the corresponding hexadecimal numbers of the index of each element of the given S-box and each element of the 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 is a Basic Polynomial or BP over Galois field GF(2^{15}). column 1 through G of Row 2 has been termed as 4th IGFP, Row 3 has been termed as 3rd IGFP, Row 4 has been termed as 2nd IGFP and Row 5 has been termed as IGFP whereas column 1 through G of Row 7 has been termed as 4th OGFP, Row 8 has been termed as 3rd OGFP, Row 9 has been termed as 2nd OGFP and Row A has been termed as 1st OGFP. The decimal equivalents of each IGFP and OGFP have been noted at column H of respective rows. Here IGFP stands for Input Galois Field Polynomial and OGFP stands for Output Galois Field Polynomials. The respective Polynomials have been shown in Row 1 through 8 of column 3 of Table.3.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.26452v1 | CC BY 4.0 Open Access | rec: 22 Jan 2018, publ: 22 Jan 2018

Row	Column	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	**H.** Decimal Equivalent
1	**Index**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**	
2	**IGFP4**	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	00255
3	**IGFP3**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	03855
4	**IGFP2**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	13107
5	**IGFP1**	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	**OGFP4**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0	42836
8	**OGFP3**	1	1	1	0	0	1	0	0	0	0	1	1	1	0	0	1	58425
9	**OGFP2**	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	1	36577
A	**OGFP1**	0	0	1	1	0	1	1	0	1	0	0	0	1	1	0	1	13965

Table.36. Input and Output BCNs of the Substitution Box

Col	1	2	3
Row	**Index**	**DCM Eqv.**	**Polynomials over Galois Field GF(2^{15}).**
1	IGFP4	00255	BP(x) = $x^7+x^6+x^5+x^4+x^3+x^2+x^1+1$.
2	IGFP3	03855	BP(x) = $x^{11}+x^{10}+x^9+x^8+x^3+x^2+x^1+1$.
3	IGFP2	13107	BP(x) = $x^{13}+x^{12}+x^9+x^8+x^5+x^4+x^1+1$.
4	IGFP1	21845	BP(x) = $x^{14}+x^{12}+x^{10}+x^8+x^6+x^4+x^2+1$.
5	OGFP4	42836	BP(x) = $x^{15}+x^{13}+x^{10}+x^9+x^8+x^6+x^4+ x^2$.
6	OGFP3	58425	BP(x) = $x^{15}+x^{14}+x^{13}+x^{10}+x^5+x^4+x^3+1$.
7	OGFP2	36577	BP(x) = $x^{15}+x^{11}+x^{10}+x^9+x^7+x^6+x^5+1$.
8	OGFP1	13965	BP(x) = $x^{13}+x^{12}+x^{10}+x^9+x^7+x^3+x^2+1$.

Table.37. Respective Polynomials of IGFP4 through IGFP1 and OGFP4 through OGFP1

4.2 4 and 8 bit S-box Generation by respective BCNs over Binary Galois Field GF(2^q) where q €(15 and 255) respectively. In this paper 4 and 8 bit identity S-boxes have been taken for example for generation of 4 and 8 bit S-boxes over binary Galois Fields GF(2^q) where q €(15 and 255) respectively. The generation of identity 4-bit S-box from four BCNs over binary Galois Field GF(2^{15}) have been elaborated in sub section 4.2.1 and The generation of identity 8-bit S-box from Eight BCNs over Binary Galois Field GF(2^{255}) have been elaborated in sub section 4.2.2. The Algorithm for generation of log $_2{}^{q+1}$ bit S-boxes over Binary Galois Field GF(2^q) has been depicted with Time Complexity of the algorithm in sub section 4.2.3.

4.2.1. Generation of 4-bit Identity Crypto S-box from four Polynomials over Binary Galois Field GF(2^{15}).

The Concerned 4-bit identity S-box has been shown in table.38 where each element of the first row of Table.38, entitled as index, have been 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 identity Substitution box. It can be concluded that the 1st row has been fixed for all possible crypto S-boxes. The values of each element of the 1st row are distinct, unique and vary between 0 and F. The values of the each element of the 2nd row of the identity crypto S-box have also been distinct and unique and also vary between 0 and F. The values of the elements of the fixed 1st row are sequential and monotonically increasing where for the 2nd row, they are also sequential and monotonically increasing for this identity S-box. Here the given Substitution Box is the 4-bit identity crypto 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**	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Table.38. 4-bit Identity Crypto S-box.

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. From row 2 through 5 and row 7 through A of each column from 1 through G of Table.39. shows the 4-bit bit sequences of the corresponding hexadecimal numbers of the index of each element of the given S-box and each element of the 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 is a Basic Polynomial over Galois field GF(2^{15}). column 1 through G of Row 2 has been termed as 4th IGFP, Row 3 has been termed as 3rd IGFP, Row 4 has been termed as 2nd IGFP and Row 5 has been termed as IGFP whereas column 1 through G of Row 7 has been termed as 4th OGFP, Row 8 has been termed as 3rd OGFP, Row 9 has been termed as 2nd OGFP and Row A has been termed as 1st OGFP. The decimal equivalents of each IGFP and OGFP have been noted at column H of respective rows. Where IGFP stands for Input Galois Field Polynomials and OGFP stands for Output Galois Field Polynomials. The respective Polynomials have been shown in Row 1 through 8 of column 3 of Table.40.

Row	Column	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	**H.** Decimal Equivalent
1	**Index**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**	
2	**IBCN4**	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	00255
3	**IBCN3**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	03855
4	**IBCN2**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	13107
5	**IBCN1**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	21845

6	S-box	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
7	OBCN4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	00255
8	OBCN3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	03855
9	OBCN2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	13107
A	OBCN1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	21845

Table.39. Input and Output BCNs of the Identity Substitution Box

Col	1	2	3
Row	Index	DCM Eqv.	Polynomials over Galois Field GF(2^{15}).
1	IGFP4	00255	BP(x) = $x^7+x^6+x^5+x^4+x^3+x^2+x^1+1$.
2	IGFP3	03855	BP(x) = $x^{11}+x^{10}+x^9+x^8+x^3+x^2+x^1+1$.
3	IGFP2	13107	BP(x) = $x^{13}+x^{12}+x^9+x^8+x^5+x^4+x^1+1$.
4	IGFP1	21845	BP(x) = $x^{14}+x^{12}+x^{10}+x^8+x^6+x^4+x^2+1$.
5	OGFP4	00255	BP(x) = $x^7+x^6+x^5+x^4+x^3+x^2+x^1+1$.
6	OGFP3	03855	BP(x) = $x^{11}+x^{10}+x^9+x^8+x^3+x^2+x^1+1$.
7	OGFP2	13107	BP(x) = $x^{13}+x^{12}+x^9+x^8+x^5+x^4+x^1+1$.
8	OGFP1	21845	BP(x) = $x^{14}+x^{12}+x^{10}+x^8+x^6+x^4+x^2+1$.

Table.40. Respective Polynomials of IGFP4 through IGFP1 and OGFP4 through OGFP1

4.2.2 Generation of 8-bit Identity Crypto S-box from Eight Polynomials over Binary Galois Field GF(2^{255}).

The concerned 8-bit identity S-box has been shown in table.41 where each element of the first row of Table.41, entitled as index, are the position of each element of the S-box within the given S-box and the elements of the column 1 through G of 2nd to 17th row, entitled as S-box, have been the elements of the given 8-bit identity Substitution box sequentially. It can be concluded that the 1st row is fixed for all possible 8-bit bijective crypto S-boxes. The values of each element of the 1st row are distinct, unique and vary between 0 and F. The values of the each element of the column 1 through G of 2nd row to 17th row of the 8-bit identity crypto S-box are also distinct and unique and vary between 0 and 256. The values of the elements of the fixed 1st row are sequential and monotonically increasing where for the 2nd to 17th row, they can be sequential or partly sequential or non-sequential and for this case elements are sequential and monotonically increasing. Here the given substitution box has been the 8-bit identity crypto 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	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
3		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
4		32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
5		48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63.
6		64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
7		80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
8		96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
9		112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
10		128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
11		144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
12		160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
13		176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
14		192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
15		208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
16		224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
17		240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Table.41. 8-bit identity crypto S-box.

Index of Each element of an 8-bit crypto S-box and the element itself is a hexadecimal number and that can be converted into a 256-bit long 8 bit bit sequence. From row 2 through 9 and row A through H of column 2 of Table.42. shows the 8-bit bit sequences of the corresponding hexadecimal numbers of the index of each element of the given S-box and each element of the S-box itself. Each row from 2 through 9 and A through H of column 2 constitutes a 256 bit, bit sequence that is a Basic Polynomial over Galois field GF(2^{255}). column 2 of Row 2 has been termed as 8th IGFP, Row 3 has been termed as 7th IGFP, Row 4 has been termed as 6th IGFP ,Row 5 has been termed as 5th IGFP, Row 6 has been termed as 4th IGFP, Row 7 has been termed as 3rd IGFP, Row 8 has been termed as 2nd IGFP and Row 9 has been termed as 1st IGFP whereas column 2 of Row A has been termed as 8th OGFP, Row B has been termed as 7th OGFP, Row C has been termed as 6th OGFP, Row D has been termed as 5th OGFP, Row E has been termed as 4th OGFP, Row F has been termed as 3rd OGFP, Row G has been termed as 2nd OGFP and Row H has been termed as 1st IGFP. The Binary Coefficient Number of each IGFP and OGFP from MSB [256th bit] to LSB [0th bit] have been given in corresponding rows of each IGFP and OGFP. Where IGFP stands for Input Galois Field Polynomials and OGFP for Output Galois Field Polynomials. The respective polynomial for IGFP8 and OGFP8 has been shown in Table.43

.

Row	Col.	MSB Polynomials (BCNs)[col.2] LSB
1	1	
2	IGFP8	0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 1111111111111111111111111111111111111111111111111111111111111111 1111111111111111111111111111111111111111111111111111111111111111
3	IGFP 7	0000000000000000000000000000000000000000000000000000000000000000 1111111111111111111111111111111111111111111111111111111111111111 0000000000000000000000000000000000000000000000000000000000000000 1111111111111111111111111111111111111111111111111111111111111111
4	IGFP 6	0000000000000000000000000000000011111111111111111111111111111111 0000000000000000000000000000000011111111111111111111111111111111 0000000000000000000000000000000011111111111111111111111111111111 0000000000000000000000000000000011111111111111111111111111111111
5	IGFP 5	0000000000000000111111111111111100000000000000001111111111111111 0000000000000000111111111111111100000000000000001111111111111111 0000000000000000111111111111111100000000000000001111111111111111 0000000000000000111111111111111100000000000000001111111111111111
6	IGFP 4	0000000011111111000000001111111100000000111111110000000011111111 0000000011111111000000001111111100000000111111110000000011111111 0000000011111111000000001111111100000000111111110000000011111111 0000000011111111000000001111111100000000111111110000000011111111
7	IGFP 3	0000111100001111000011110000111100001111000011110000111100001111 0000111100001111000011110000111100001111000011110000111100001111 0000111100001111000011110000111100001111000011110000111100001111 0000111100001111000011110000111100001111000011110000111100001111
8	IGFP 2	0011001100110011001100110011001100110011001100110011001100110011 0011001100110011001100110011001100110011001100110011001100110011 0011001100110011001100110011001100110011001100110011001100110011 0011001100110011001100110011001100110011001100110011001100110011
9	IGFP 1	0101010101010101010101010101010101010101010101010101010101010101 0101010101010101010101010101010101010101010101010101010101010101 0101010101010101010101010101010101010101010101010101010101010101 0101010101010101010101010101010101010101010101010101010101010101
A	OGFP8	0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 1111111111111111111111111111111111111111111111111111111111111111 1111111111111111111111111111111111111111111111111111111111111111
B	OGFP 7	0000000000000000000000000000000000000000000000000000000000000000 1111111111111111111111111111111111111111111111111111111111111111 0000000000000000000000000000000000000000000000000000000000000000 1111111111111111111111111111111111111111111111111111111111111111
C	OGFP 6	0000000000000000000000000000000011111111111111111111111111111111 0000000000000000000000000000000011111111111111111111111111111111 0000000000000000000000000000000011111111111111111111111111111111 0000000000000000000000000000000011111111111111111111111111111111
D	OGFP 5	0000000000000000111111111111111100000000000000001111111111111111 0000000000000000111111111111111100000000000000001111111111111111 0000000000000000111111111111111100000000000000001111111111111111 0000000000000000111111111111111100000000000000001111111111111111
E	OGFP 4	0000000011111111000000001111111100000000111111110000000011111111 0000000011111111000000001111111100000000111111110000000011111111 0000000011111111000000001111111100000000111111110000000011111111 0000000011111111000000001111111100000000111111110000000011111111
F	OGFP 3	0000111100001111000011110000111100001111000011110000111100001111 0000111100001111000011110000111100001111000011110000111100001111 0000111100001111000011110000111100001111000011110000111100001111

		0000111100001111000011110000111100001111000011110000111100001111
G	**OGFP 2**	0011001100110011001100110011001100110011001100110011001100110011 0011001100110011001100110011001100110011001100110011001100110011 0011001100110011001100110011001100110011001100110011001100110011 0011001100110011001100110011001100110011001100110011001100110011
H	**OGFP 1**	0101010101010101010101010101010101010101010101010101010101010101 0101010101010101010101010101010101010101010101010101010101010101 0101010101010101010101010101010101010101010101010101010101010101 0101010101010101010101010101010101010101010101010101010101010101

Table.42. BCNs for 8 IGFPs and OGFPs.

BCNs of	**Polynomial**
IGFP8 &OGFP8	$x^{127}+ x^{126}+ x^{125}+ x^{124}+ x^{123}+ x^{122}+ x^{121}+ x^{120}+ x^{119}+ x^{118}+ x^{117}+ x^{116}+ x^{115}+ x^{114}+ x^{113}+ x^{112}+$ $x^{111}+ x^{110}+ x^{109}+ x^{108}+ x^{107}+ x^{106}+ x^{105}+ x^{104}+ x^{103}+ x^{102}+ x^{101}+ x^{100}+ x^{99}+ x^{98}+ x^{97}+ x^{96}+$ $x^{95}+ x^{94}+ x^{93}+ x^{92}+ x^{91}+ x^{90}+ x^{89}+ x^{88}+ x^{87}+ x^{86}+ x^{85}+ x^{84}+ x^{83}+ x^{82}+ x^{81}+ x^{80}+$ $x^{79}+ x^{78}+ x^{77}+ x^{76}+ x^{75}+ x^{74}+ x^{73}+ x^{72}+ x^{71}+ x^{70}+ x^{69}+ x^{68}+ x^{67}+ x^{66}+ x^{65}+ x^{64}+$ $x^{63}+ x^{62}+ x^{61}+ x^{60}+ x^{59}+ x^{58}+ x^{57}+ x^{56}+ x^{55}+ x^{54}+ x^{53}+ x^{52}+ x^{51}+ x^{50}+ x^{49}+ x^{48}+$ $x^{47}+ x^{46}+ x^{45}+ x^{44}+ x^{43}+ x^{42}+ x^{41}+ x^{40}+ x^{39}+ x^{38}+ x^{37}+ x^{36}+ x^{35}+ x^{34}+ x^{33}+ x^{32}+$ $x^{31}+ x^{30}+ x^{29}+ x^{28}+ x^{27}+ x^{26}+ x^{25}+ x^{24}+ x^{23}+ x^{22}+ x^{21}+ x^{20}+ x^{19}+ x^{18}+ x^{17}+ x^{16}+$ $x^{15}+ x^{14}+ x^{13}+ x^{12}+ x^{11}+ x^{10}+ x^{9}+ x^{8}+ x^{7}+ x^{6}+ x^{5}+ x^{4}+ x^{3}+ x^{2}+ x+ 1.$

Table.43. Respective Polynomial of IGFP8 and OGFP8 of the Given 8 bit S-box.

4.2.3 Algorithm to generate S-box from Polynomials over Galois field $GF(2^{15})$ or $GF(2^{255})$.

START.

Step OA. Choose 4 Galois field Polynomials over Galois field $GF(2^{15})$ or 8 Galois field Polynomials over Galois field $GF(2^{255})$.

Step.01. If Number of Terms in BCNs are Half of Number of total terms Then Step 02. Else Step 0A.

Step.02. Convert to decimal the 4 or 8 bit binary number generated by bits in same position of 4 BCNs for Galois field Polynomials over Galois field $GF(2^{15})$ or 8 Galois field Polynomials over Galois field $GF(2^{255})$.

STOP.

Time Complexity of the given Algorithm. O(n).

4.3. 4 and 8 bit S-box Generation by respective BCNs over Non Binary Galois Field $GF(16^{15})$ and Galois Field $GF(256^{255})$ respectively. The coefficients of each polynomial over non binary Galois Field $GF(16^{15})$ forms a 4-bit S-box. The Coefficient of highest or lowest degree term must be the 1st element in 4-bit S-box, the value of other elements are the value of coefficients with immediate degree less than or greater than the previous one. Let The Polynomial be,

BP(x) = $0x^{15}+1x^{14}+2x^{13}+3x^{12}+4x^{11}+5x^{10}+6x^{9}+7x^{8}+8x^{7}+9x^{6}+10x^{5}+11x^{4}+12x^{3}+13x^{2}+14x+15$…………..(ii)

For the above Polynomial The Constituted 4-bit S-box have been given in Table 44.

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

Table.44. constituted 4-bit Crypto S-box.

The Polynomial with coefficients in reverse order,

BP(x) = $15x^{15}+14x^{14}+13x^{13}+12x^{12}+11x^{11}+10x^{10}+9x^{9}+8x^{8}+7x^{7}+6x^{6}+5x^{5}+4x^{4}+3x^{3}+2x^{2}+1x+0$…………….(iii)

For the above Polynomial The Constituted 4-bit S-box have been given in Table 45.

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

Table.45. constituted 4-bit Crypto S-box.

The coefficients of each polynomial over non binary Galois Field $GF(256^{255})$ forms an 8-bit S-box. The Coefficient of highest or lowest degree term must be the 1st element in 4-bit S-box, the value of other elements are the value of coefficients with immediate degree less than or greater than the previous one. Let The Polynomial be, Let the Polynomial be given in Table.46.,

Polynomial BP(x) =
$0.x^{255}+ 1.x^{254}+ 2.x^{253}+ 3.x^{252}+ 4.x^{251}+ 5.x^{250}+ 6.x^{249}+ 7.x^{248}+ 8.x^{247}+ 9.x^{246}+ 10.x^{245}+ 11.x^{244}+ 12.x^{243}+ 13.x^{242}+ 14.x^{241}+ 15.x^{240}+16.x^{239}+ 17.x^{238}+ 18. x^{237}+ 19.x^{236}+ 20.x^{235}+ 21.x^{234}+ 22.x^{233}+ 23.x^{232}+ 24.x^{231}+ 25.x^{230}+ 26.x^{229}+ 27.x^{228}+ 28x^{227}+ 29.x^{226}+ 30.x^{225}+ 31.x^{224}+32. X^{223}+ 33. x^{222}+ 34.x^{221}+ 35.x^{220}+ 36.x^{219}+ 37.x^{218}+ 38.x^{217}+39. x^{216}+40. x^{215}+ 41.x^{214}+ 42.x^{213}+ 43.x^{212}+ 44.x^{211}+ 45.x^{210}+ 46.x^{209}+ 47.x^{208}+48.x^{207}+ 49.x^{206}+ 50.x^{205}+ 51.x^{204}+ 52.x^{203}+ 53.x^{202}+ 54.x^{201}+ 55.x^{200}+ 56.x^{199}+ 57.x^{198}+ 58.x^{197}+ 59.x^{196}+ 60.x^{195}+ 61.x^{194}+$

$62.x^{193}+ 63.x^{192}+ 64.x^{191}+ 65.x^{190}+ 66.x^{189}+ 67.x^{188}+68.x^{187}+ 69.x^{186}+ 70.x^{185}+ 71.x^{184}+ 72.x^{183}+ 73.x^{182}+ 74.x^{181}+ 75.x^{180}+ 76.x^{179}+$
$77.x^{178}+ 78.x^{177}+ 79.x^{176}+80.X^{175}+ 81.x^{174}+ 82.x^{173}+ 83.x^{172}+84. x^{171}+ 85.x^{170}+ 86.x^{169}+ 87.x^{168}+88. x^{167}+ 89.x^{166}+ 90.x^{165}+ 91.x^{164}+$
$92.x^{163}+ 93.x^{162}+ 94.x^{161}+ 95.x^{160}+96.x^{159}+ 97.x^{158}+ 98.x^{157}+ 99.x^{156}+ 100.x^{155}+ 101.x^{154}+ 102.x^{153}+ 103.x^{152}+ 104.x^{151}+105.x^{150}+106.x^{149}+$
$107.x^{148}+108. x^{147}+109.x^{146}+110. x^{145}+ 111.x^{144}+112.X^{143}+113.x^{142}+114.x^{141}+115.x^{140}+116.x^{139}+117. x^{138}+118. x^{137}+119. x^{136}+120.$
$x^{135}+121.x^{134}+122.x^{133}+123.x^{132}+124.x^{131}+125.x^{130}+126.x^{129}+127.x^{128}+128.x^{127}+129. x^{126}+ 130.x^{125}+131.x^{124}+ 132.x^{123}+133.x^{122}+134.x^{121}+$
$135.x^{120}+ 136.x^{119}+ 137.x^{118}+ 138.x^{117}+ 139.x^{116}+ 140.x^{115}+ 141.x^{114}+ 142.x^{113}+ 143.x^{112}+144.x^{111}+ 145.x^{110}+ 146.x^{109}+ 147.x^{108}+$
$148.x^{107}+ 149.x^{106}+ 150.x^{105}+ 151.x^{104}+ 152.x^{103}+ 153.x^{102}+ 154.x^{101}+ 155.x^{100}+ 156.x^{99}+ 157.x^{98}+ 158.x^{97}+ 159.x^{96}+160.x^{95}+ 161.x^{94}+$
$162.x^{93}+ 163.x^{92}+ 164.x^{91}+ 165.x^{90}+ 166.x^{89}+ 167.x^{88}+ 168.x^{87}+ 169.x^{86}+ 170.x^{85}+ 171.x^{84}+ 172.x^{83}+ 173.x^{82}+ 174.x^{81}+ 175.x^{80}+176.x^{79}+$
$177.x^{78}+ 178.x^{77}+ 179.x^{76}+ 180.x^{75}+ 181.x^{74}+ 182.x^{73}+ 183.x^{72}+ 184.x^{71}+ 185.x^{70}+ 186.x^{69}+ 187.x^{68}+ 188.x^{67}+ 189.x^{66}+ 190.x^{65}+$
$191.x^{64}+192.x^{63}+193. x^{62}+ 194.x^{61}+ 195.x^{60}+ 196.x^{59}+ 197.x^{58}+ 198.x^{57}+ 199.x^{56}+ 200.x^{55}+ 201.x^{54}+ 202.x^{53}+ 203.x^{52}+ 204.x^{51}+ 205.x^{50}+$
$206.x^{49}+ 207.x^{48}+208.x^{47}+ 209.x^{46}+ 210.x^{45}+ 211.x^{44}+ 212.x^{43}+ 213.x^{42}+ 214.x^{41}+ 215.x^{40}+ 216.x^{39}+ 217.x^{38}+ 218.x^{37}+219.x^{36}+ 220.x^{35}+$
$221.x^{34}+ 222.x^{33}+ 223.x^{32}+224.x^{31}+ 225.x^{30}+ 226.x^{29}+ 227.x^{28}+ 228.x^{27}+ 229.x^{26}+ 230.x^{25}+ 231.x^{24}+ 232.x^{23}+ 233.x^{22}+ 234.x^{21}+ 235.x^{20}+$
$236.x^{19}+ 237.x^{18}+ 238.x^{17}+ 239.x^{16}+240.x^{15}+ 241.x^{14}+242. x^{13}+ 243.x^{12}+ 244.x^{11}+ 245.x^{10}+ 246.x^{9}+ 247.x^{8}+ 248.x^{7}+ 249.x^{6}+ 250.x^{5}+$
$251.x^{4}+ 252.x^{3}+ 253.x^{2}+ 254x+ 255.$

Table.46. Polynomial to Construct 8-bit Identity S-box.

For the above Polynomial The Constituted 8-bit S-box have been given in Table 47.

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	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
3		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
4		32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
5		48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63.
6		64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
7		80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
8		96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
9		112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
10		128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
11		144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
12		160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
13		176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
14		192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
15		208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
16		224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
17		240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Table.47. Constituted Identity 8-bit S-box.

Note. The 32-bit S-boxes can be constituted by polynomials over Galois field GF[$(2^{32})^{(2^{\wedge}32-1)}$] and the 64-bit S-boxes can be constituted by polynomials over Galois field GF[$(2^{64})^{(2^{\wedge}64-1)}$].

4.4. Cryptographic analysis of 32 DES 4-bit S-boxes and 10 better 4-bit S-boxes with relevant cryptographic properties of 4-bit crypto S-boxes.

In subsec.4.4.1.the cryptographic analysis procedures of the said cryptographic properties have been described. The cryptographic analysis of 32 DES 4-bit S-boxes has been evaluated in subsec.4.4.2. cryptographic analysis of 10 generated better S-boxes has been described in subsec.4.4.3

4.4.1. Analysis Procedure. For SAC, HO-SAC and Extended SAC of 4-bit S-boxes as the numbers of satisfied COPBFs have been increased it will give better security and optimum value gives at most security.

In Difference Distribution Table there have been 256 cells, i.e. 16 rows and 16 columns. Each row has been for each input difference varies from 0 to F. Each column in each row represents each output difference varies from 0 to F for each input difference. 0 in any cell indicates absence of that output difference for subsequent input difference. Such as 0 in 2nd cell of Table.7.b of relevant DDT means for input difference 0 the corresponding output difference o has been absent. If number of 0 is 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 has been said to be almost secure from this attack.

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.

In Linear Analysis 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 has been put into the concerned cell. 8 in a cell indicate that the particular linear relation has been 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 have been 143. That is close to 128. So the S-box has been said to be less secure from this attack.

The value of $^{n}C_{r}$ has been 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 has been the largest. 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 No of Linear Approximations 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 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 S-Box or not a good Crypt analytically immune S-Box.

If the values of total number of Existing Linear equations for a 4-bit S-Box are 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 S-Box is declared as a good 4-bit S-Box or 4-bit 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 S-Box or S-Box with medium security. The 4-bit S-Boxes having existing linear equations less than 16 are declared as Poor 4-bit S-Box or vulnerable to cryptanalytic attack.

4.4.2. Cryptographic analysis of 32 DES 4-bit S-boxes. The cryptographic analysis of 32 DES 4-bit S-boxes with the said relevant cryptographic properties of 4-bit BFs has been given below in table.48. Here in table 48. column heading 'noelr gives numbers of existing linear relations in a particular 4-bit crypto S-box. Column heading 'nobal' gives numbers of balanced DBFs in linear cryptanalysis. 'n0dif' gives numbers of 0s in difference distribution table or DDT and 'nodif' gives numbers of 8s in DAT. 'nosac' gives numbers of COPBFs satisfy SAC of 4-bit BFs and 'n3sac','n3sac' and 'nalsac' gives numbers of COPBFs satisfy 2nd order SAC of 4-bit BFs, 3rd order SAC of 4-bit BFs and Extended SAC of 4-bit BFs respectively.

S-box	noelr	nobal	n0dif	nodif	nosac	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.48. Cryptographic analysis of 32 DES S-boxes.

4.4.3. Cryptographic analysis of 10 generated better 4-bit S-boxes. The cryptographic analysis of 10 generated better 4-bit S-boxes with the said relevant cryptographic properties of 4-bit BFs has been given below in table.49. Here in table 49. column heading 'noelr gives numbers of existing linear relations in a particular 4-bit crypto S-box. Column heading 'nobal' gives numbers of balanced DBFs in linear cryptanalysis. 'n0dif' gives numbers of 0s in difference distribution table or DDT and 'nodif' gives numbers of 8s in DAT. 'nosac' gives numbers of COPBFs satisfy SAC of 4-bit BFs and 'n3sac','n3sac' and 'nalsac' gives numbers of COPBFs satisfy 2nd order SAC of 4-bit BFs, 3rd order SAC of 4-bit BFs and Extended SAC of 4-bit BFs respectively.

S-box	noelr	nobal	n0dif	nodif	nosac	n2sac	n3sac	nalsac
01235b8694ca7def	33	162	189	39	16	7	16	39
01235b86a4f97edc	33	200	206	45	16	13	16	45
10324a967b8fced5	27	156	175	39	16	11	8	39
103268957abcfde4	31	147	167	42	16	12	11	42
0132c5794a86fbed	26	164	189	39	16	7	16	39
1032c5684a97ebfd	28	162	189	39	16	7	16	39
1032c56879a4dbfe	27	196	206	39	16	7	16	39
1023c46a5b87e9fd	35	148	182	42	16	9	16	42
0123c7495b86eadf	23	149	170	42	16	11	13	42
103249adc65be87f	30	134	166	39	16	8	13	39

Table.49. cryptographic analysis of 10 generated Better 4-bit S-boxes

4.5. Results and Discussion. In table.48. 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 have been less secure from this attack and 8 out of 32 have been 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 have been 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 has been 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 has been observed to be less secure.

But in table.49. out of 10 generated better 4-bit S-boxes range of 'noelr' has been 27 to 33 so it can be concluded that these S-boxes have been more immune to this attack. Now range of 'nobal' has been 134 to 200 i.e. very secure to linear cryptanalysis since number of 8s in LAT is very large in number. Again range of 'n0dif' has been 166 to 206 i.e. the result is very similar to 32 DES 4-bit S-boxes. Now All 10 4-bit S-boxes 'nosac' have been 16. i.e. they satisfy SAC of 4-bit S-boxes. Again the ranges of 'n2sac', 'n3sac', 'nalsac' have been 7 to 13, 13 to 16 and 39 to 45 respectively. I.e. most of them satisfies 3rd order SAC of 4-bit S-boxes and for 2nd order SAC of 4-bit S-boxes the results have been very similar to DES 4-bit S-boxes. In case of 'nalsac' or Extended SAC the results are better than DES 32 4-bit S-boxes.

Now it is to be noted that all non-crypto S-boxes and 16! Crypto S-boxes can be generated by these two procedures by IPs over Galois field GF(p^q). The crypto S-boxes have then be chosen through the analysis of relevant cryptographic properties of 4-bit S-boxes. The procedure is same for 8, 16, 32 and 64 bit S-boxes. The generated 8, 16, 32 or 64 bit S-boxes can be chosen like the way the way the 4-bit S-boxes have been chosen in this paper.

5. Conclusion. From results and discussion it can be concluded that generated and analyzed 4-bit S-boxes are better S-boxes than the 32 4-bit DES S-boxes. All algorithms of cryptographic properties and S-box generation have been given in the paper. The review and algorithms have been presented in a very lucid manner in the paper for convenient understanding of readers. The generation of 4-bit and 8-bit S-boxes has been very easy and lucid and the chosen generated 4-bit S-boxes can be claimed to be the best 4-bit and 8-bit S-boxes.

6. Acknowledgement. For this exhaustive work I would like to acknowledge my supervisors Dr. Ranjan Ghosh , Associate Professor, Institute of Radio Physics and Electronics, University of Calcutta and Prof.(Dr.) Amlan chakrabarti Dean, Faculty counsil of Post graduate Studies in Engineering and Technology, University of Calcutta for their continuous encouragement and help. I would also like to acknowledge Prof.(Dr.) Debatosh Guha Head Dept. Institute of Radio Physics and Electronics, University of Calcutta for providing me with the nice infrastructure.

References.

[CA90]Adams, C. & Tavares, S. J. Cryptology (1990) 3: 27. https://doi.org/10.1007/BF00203967

[AT90] Adams, Carlisle, Tavares, Stafford, "The structured design of cryptographically good S-boxes", J. Cryptology (1990) 344 vol. 3, pp : 27-41.

[HF71]Feistel, H. "Block Cipher Cryptographic System", US Patent 3798359 (Filed June 30, 1971).

[NT77]Data Encryption Standard, Federal Information Processing Standards Publication (FIPS PUB) 46, National Bureau of Standards, Washington, DC (1977).

[NT99]Data Encryption Standard (DES), Federal Information Processing Standards Publication (FIPS PUB) 46-3, National Institute of Standards and Technology, Gaithersburg, MD (1999).

[RC35] Church R, Tables of irreducible polynomials for the first four prime moduli, The Annals of Maths., 2nd Series, vol. 36, no. 1, 198-209, Jan (1935) http://www.jstor.org/stable/1968675.

[RS62] Swan, Richard G. Factorization of polynomials over finite fields. Pacific J. Math. 12 (1962), no. 3, 1099--1106.https://projecteuclid.org/euclid.pjm/1103036322.

[TD63] Thomas C. Bartee, David I. Schneider, Computation with finite fields, Information and Control,Volume 6, Issue 2, June 1963, Pages 79-98, https://doi.org/10.1016/S0019-9958(63)90129-3.

[EB67] E.R.Berlekamp, Factoring Polynomials Over Finite Fields, Bell System Technical Journal(Blackwell Publishing Ltd), 46(8) 1853-1859,http://dx.doi.org/10.1002/j.1538-7305.1967.tb03174.x. DOI:10.1109/TIT.1968.1054226.

[TK68]T.Kasami, S hu.Lin, W. Peterson,(1968) Polynomial Codes, IEEE Transactions on Information Theory, Volume: 14, Issue: 6, Nov 1968,PP:807-814,

[EB70] E.R.Berlekamp, Factoring polynomials over large finite fields, Math. Comp. 24 (1970), 713-735, https://doi.org/10.1090/S0025-5718-1970-0276200-X.

[MR80] Michael O. Rabin, Probabilistic Algorithms In Finite Fields,May 1980, SIAM Journal on Computing 9(2):273-280,DOI:10.1137/0209024.

[AL85] A.K.Lenstra, Factoring multivariate polynomials over finite fields, Journal of Computer and System Sciences, Volume 30, Issue 2, April 1985, Pages 235-248, https://doi.org/10.1016/0022-0000(85)90016-9.

[RM87] Robert J. McEliece, Factoring Polynomials over Finite Fields, In book: Finite Fields for Computer Scientists and Engineers, pp.75-96, January 1987, DOI:10.1007/978-1-4613-1983-2_7.

[LR88] Lajos Rónyai, Factoring polynomials over finite fields, Journal of Algorithms,Volume 9, Issue 3, September 1988, Pages 391-400, https://doi.org/10.1016/0196-6774(88)90029-6.

[DW90] Da Qing Wan, Factoring multivariate polynomials over large finite fields, Math. Comp. 54 (1990), 755-770, https://doi.org/10.1090/S0025-5718-1990-1011448-0.

[MR90] Marc Rybowicz, (1990) Search of primitive polynomials over finite fields, Journal of Pure and Applied Algebra,Volume 65, Issue 2, 24 August 1990, Pages 139-151, https://doi.org/10.1016/0022-4049(90)90115-X.

[VS90] Victor Shoup, New Algorithms for Finding Irreducible Polynomials over Finite Fields (1990),Vol.54(189).Jan.1990.435-447.

[LR92] Lajos Rónyai, Galois Groups and Factoring Polynomials over Finite Fields, SIAM Journal on Discrete Mathematics 5(3) August 1992,DOI:10.1137/0405026.

[MZ94] Miodrag Zivkovic,Table of Primitive Binary Polynomials. July 1994,Mathematics of Computation 63(207):301-301,DOI.10.1090/S0025-5718-1994-1240662-8.

[IS96] Igor Shparlinski, On finding primitive roots in finite fields,Theoretical Computer Science,Volume 157, Issue 2, 5 May 1996, Pages 273-275,https://doi.org/10.1016/0304-3975(95)00164-6.

[PX96] P. Flajolet, X. Gourdon, D. Panario, Random polynomials and polynomial factorization, in: Lecture Notes in Computer Science, 1099, Springer-Verlag, New York/Berlin, 1996, pp. 232-243.

[GP97] Gao S., Panario D. (1997) Tests and Constructions of Irreducible Polynomials over Finite Fields. In: Cucker F., Shub M. (eds) Foundations of Computational Mathematics. Springer, Berlin, Heidelberg.

[EV98] Erich Kaltofen, Victor Shoup, (1998)Subquadratic-time factoring of polynomials over finite fields,Mathematics of Computation of the American Mathematical Society,67(223) 1179-1197.

[EJ01] Eric Bach, Joachim von zur Gathen, Hendrik W.Lenstra Jr, Factoring Polynomials over Special Finite Fields, Finite Fields and Their Applications Volume 7, Issue 1, January 2001, Pages 5-28, https://doi.org/10.1006/ffta.2000.0306.

[GA02] Gao, S., Lauder, A.G.B.: Hensel lifting and bivariate polynomial factorisation over finite fields. Math. Comp. 71, 1663–1676 (2002).

[BZ03] Richard P. Brent and Paul Zimmermann (2003),Algorithms for Finding Almost Irreducible and Almost Primitive Trinomials,in proceedingsof Brent 2003 Algorithms FF.

[NE04]Nirmal R. Saxena and Edward J. McCluskey,(2004) Primitive Polynomial Generation Algorithms Implementation and Performance Analysis,TECHNICAL REPORT(CRC TR 04-03),April-2004, Center for Reliable Computing.

[SE04] Shuhong Gao, Erich Kaltofen, Alan G.B.Lauder, Deterministic distinct-degree factorization of polynomials over finite fields, Journal of Symbolic Computation, Volume 38, Issue 6, December 2004, Pages 1461-1470, https://doi.org/10.1016/j.jsc.2004.05.004.

[SM05] Subhamoy Maitraa, Kishan Chand Gupta, Ayineedi Venkateswar,Results on multiples of primitive polynomials and their products over GF(2), Theoretical Computer Science, Volume 341, Issues 1–3, 5 September 2005, Pages 311-343,https://doi.org/10.1016/j.tcs.2005.04.011.

[MS07] Michael Scott,(2007) Optimal Irreducible Polynomials for GF(2^m) Arithmetic,Published 2007 in IACR Cryptology ePrint Archive.

[CF08]Fernandez, Carlos K, (2008) Pascal Polynomials Over GF(2), Master's thesis, NAVAL POSTGRADUATE SCHOOL MONTEREY CA DEPT OF MATHEMATICS, JUN 2008, Accession Number:ADA483773.

[CS08] Chandan Saha,(2008) A Note on Irreducible Polynomials and Identity Testing.

[AA09] Asimi Ahmed,Aboubakr Lbekkouri, Determination of irreducible and primitive polynomials over a binary finite field,January 2009,Conference: Workshop sur les Technologies de l'Information et de la Communication, Agadir, Maroc, 24-25 Décembre 2009, pp 94.

[CR09] Chelsea Richards, Algorithms for Factoring Square-Free Polynomials over Finite Fields, August 7, 2009.

[RW09] Ryul Kim, Wolfram Koepf,Divisibility of Trinomials by Irreducible Polynomials over F2, International Journal of Algebra, Vol.3, 2009, no.4, 189-197.

[SM11] Sajid Hanif, Muhammad Imran, (2011) Factorization Algorithms for Polynomials over Finite Fields, Degree Project, School of Computer Science, Physics and Mathematics, Linnaeus University.

[LQ12] Wang, L. & Wang, Q. Des. Codes Cryptogr. (2012) 63: 87. https://doi.org/10.1007/s10623-011-9537-6.

[JC13] Couveignes, JM. & Lercier, R. Isr. J. Math. (2013) 194: 77. https://doi.org/10.1007/s11856-012-0070-8.

[DM14] David Marquis, (2014)Deterministic Factorization of Polynomials over Finite Fields, Thesis:MS in Pur Mathematics, Carleton University, Ottawa, Canada.

[GH14] Gaustav Hammarhjelm, Construction of Irreducible Polynomials over Finite Fields, U.U.D.M Project Report 2014:17, Uppasala Universitet.

[NC14] Cavanna, Nicholas, "Polynomial Factoring Algorithms and their Computational Complexity" (2014).Honors Scholar Theses. 384. http://digitalcommons.uconn.edu/srhonors_theses/384.

[WJ14] WANG Jiantao, Dong Zheng.Simple method to find primitive polynomials of degree n over $GF(2)$ where $2^{n}-1$ is a Mersenne prime[OL]. [4 March 2014],http://www.paper.edu.cn/lwzx/en_releasepaper/content/4587059.

[SJ15] JKM Sadique Uz Zaman, Sankhanil Dey, Ranjan Ghosh, An Algorithm to find the Irreducible Polynomials over Galois Field GF(p^m), January 2015,International Journal of Computer Applications 109(15):24-29,DOI:10.5120/19266-1012.

[HJ16] Junsoo Ha, Irreducible polynomials with several prescribed coefficients, Finite Fields and Their Applications,Volume 40, July 2016, Pages 10-25,https://doi.org/10.1016/j.ffa.2016.02.006.

[BP17] Bjorn Poonen, Using zeta functions to factor polynomials over finite fields, Oct.3, 2017,arXiv:1710.00970.

[EWNN] Weisstein, Eric W. "Integer Polynomial." From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/IntegerPolynomial.html.

[AT90] Adams, Carlisle, Tavares, Stafford, “The structured design of cryptographically good S-boxes”, J. Cryptology (1990) 344 vol. 3, pp : 27-41.

[HF71]Feistel, H. "Block Cipher Cryptographic System", US Patent 3798359 (Filed June 30, 1971).

[NT77]Data Encryption Standard, Federal Information Processing Standards Publication (FIPS PUB) 46, National Bureau of Standards, Washington, DC (1977).

[NT99]Data Encryption Standard (DES), Federal Information Processing Standards Publication (FIPS PUB) 46-3, National Institute of Standards and Technology, Gaithersburg, MD (1999).

[DR00] Joan Daemen,Vincent Rijmen (2000), AES Proposal: Rijndael,http://csrc.nist.gov/encryption/aes/ Last Visited: 7th February 2001.

[VM95] Serge Vaudenay and Shiho Moriai (1994), Comparison of the Randomness Provided by Some AES Candidates, EUROCRYPT 1994, Springer Verlag no. 950, pg. 386-397.

[SDS17] Link. https://www.academia.edu/35326794/Title._List_of_DEs_of_All_Monic_IPs_Over_Galois_Field_GF_7_7_.

[SDH17] Link. https://www.academia.edu/35326783/Title._List_of_DEs_of_All_Monic_IPs_Over_Galois_Field_GF_101_3_.