Extended SAC: A Review and new Algorithms of Differential Cryptanalysis of 4-bit S-Boxes and Strict Avalanche Criterion of 4-bit BFs and 4-bit S-Boxes again with a new Extension to HO-SAC Criterion.

Sankhanil Dey Corresp., [1] , **Ranjan Ghosh** [1]

[1] University of Calcutta, Institute of Radio Physics and Electronics, Kolkata, West Bengal, India

Corresponding Author: Sankhanil Dey
Email address: sdrpe_rs@caluniv.ac.in

Bitwise-Xor of two 4 bit binary numbers
or 4-bit bit patterns entitled 4-bit differences carries information in Cryptography. The Method to Analyze Cryptographic cipher algorithms or 4-bit substitution boxes with 4-bit differences is known as Differential Cryptanalysis. In this paper a brief review of Differential Cryptanalysis of 4-bit bijective Crypto S-Boxes and a new algorithm to analyze them using 4-bit Boolean Functions (BFs) have been introduced. A brief review of Strict Avalanche Criterion (SAC) of 4-bit bijective Crypto S-Boxes and 4-bit BFs and two new algorithms of both the aforesaid criterions have been introduced in this paper. A New algorithm entitled extended Strict Avalanche Criterion (An Extension to Higher Order SAC or HO-SAC) has also been introduced. A new Analysis of Similarity of extended HO-SAC and Differential Cryptanalysis has also been elaborated in this paper.

Extended SAC: A Review and new Algorithms of Differential Cryptanalysis of 4-bit S-Boxes and Strict Avalanche Criterion of 4-bit BFs and 4-bit S-Boxes again with a new Extension to HO-SAC Criterion.

Sankhanil Dey[1], Ranjan Ghosh[2],
sankhanil12009@gmail.com[1], rghosh47@gmail.com[2],
Institute of Radio physics and electronics,
92, APC Road, Kolkata-700009,
University of Calcutta.

Abstract. Bitwise-Xor of two 4 bit binary numbers or 4-bit bit patterns entitled 4-bit differences carries information in Cryptography. The Method to Analyze Cryptographic cipher algorithms or 4-bit substitution boxes with 4-bit differences is known as Differential Cryptanalysis. In this paper a brief review of Differential Cryptanalysis of 4-bit bijective Crypto S-Boxes and a new algorithm to analyze them using 4-bit Boolean Functions (BFs) have been introduced. A brief review of Strict Avalanche Criterion (SAC) of 4-bit bijective Crypto S-Boxes and 4-bit BFs and two new algorithms of both the aforesaid criterions have been introduced in this paper. A New algorithm entitled extended Strict Avalanche Criterion (An Extension to Higher Order SAC or HO-SAC) has also been introduced. A new Analysis of Similarity of extended HO-SAC and Differential Cryptanalysis has also been elaborated in this paper.

Keywords with MSC: **Cryptography: MSC 94A60;Boolean Function: MSC 06E30;Application of Boolean Algebra: MSC94C10 .**

1. Introduction.

Finite Difference or Exclusive-or between two adjacent or nonadjacent 4-bit nibbles of plaintext ASCII bits and the corresponding 4-bit cipher-text ASCII bits, also carries information in cryptography. Each plaintext 4-bit, 8-bit or 32-bit words of Plaintext ASCII bits and their corresponding 4, 8 and 32 bit words of cipher-text ASCII bits are xored in bit level or presently in byte level in cryptography and it is referred to as 4-bit Differences [1]. The possible 4-bit nibbles of plaintext ASCII bits or cipher-text ASCII bits for 4-bit Boolean or switching logic are [0000], [0001], [0010], [0011], [0100], [0101], [0110], [0111], [1000], [1001], [1010], [1011], [1100], [1101], [1110] and [1111]. The aforesaid 4-bit nibbles are also the all possible Input and output differences of 4-bit bijective S-Boxes. They are also the all possible 4-bit binary equivalents of all possible elements of Input and output 4-bit bijective S-Boxes [2].

A 4-bit Bijective Crypto S-Box or S-Box Contains 16 unique and distinct data element varies from 0 to F in Hex. The positions of each element within 16 elements are defined as Index is also unique and distinct and varies from 0 to F in Hex. The Index constitutes an Identity S-Box called as Input S-Box in which the elements are monotonically increasing in a certain order. In the S-Box or Output S-Box, the elements are sequential, or partly sequential or non-sequential and the elements are also unique and distinct i.e. there is no repetition of any element within the S-Box [3].

In Differential Cryptanalysis of 4-bit S-Boxes the 16 distant Input S-Boxes have been obtained by xor operation of each of 16 Input Differences varies from 0 to F in hex to 16 elements of Input S-Box. The 16 Distant Output S-Boxes have been obtained by shuffling the elements Output S-boxes in certain order in which the elements of Input S-Boxes have been shuffled in concerned Distant Input S-Boxes. The 16 elements of each Output S-Box and the elements in corresponding position of corresponding Distant Output 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. The count of each element from 0 to F in Difference S-Box have been noted and put in Differential Distribution Table (DDT) for analysis of Output S-Box [4][5][6]. The concept has been reviewed in sec 2.

In this paper a new algorithm using 4-bit BFs for Differential Cryptanalysis of 4-bit S-Boxes have been introduced. An Input S-Box can be decomposed into four 4-bit Input Vectors (IPVs) with Decimal Equivalents 255 for 4th IPV, 3855 for IPV, 13107 for 2nd IPV, and 21845 for 1st IPV respectively. Now we complement each IPV 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 four IPVs Distant Input S-Boxes to obtain Distant Output S-Boxes. The four 4-bit BFs of Output S-Boxes are xored bitwise with four 4-bit BFs of Distant Output 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 balancedness, means utmost uncertainty. The Table in which the balancedness of 64 Difference BFs is noted is called as Differential Analysis Table (DAT). The Theory has been elaborated in section 2.

In Strict Avalanche Criterion, 4 IPVs of a 4-bit Output BF has been complemented one at a time. If in complemented four Output BFs 8 bit values has been changed and 8 bit values remains same for 4-bit BFs then the Output 4-bit BF is said to satisfy Strict Avalanche Criterion of 4-bit BFs [7][8]. Complementing 4th IPV means interchanging each 8 bit halves of a 4-bit Output BF, whereas complementing 3rd IPV means interchanging each 4 bit halves of each 8 bit halves, whereas complementing 2nd IPV means interchanging each 2 bit halves of each 4 bit halves of each 8 bit halves and complementing 1st IPV means interchanging each bit of all 2bit halves of a 16 bit long 4-bit BF. In this paper this shifting property is used to construct an algorithm of SAC of 4-bit BFs. If all four output BF of a 4-bit S-Box satisfy SAC for 4-bit Output BFs then the concerned Output S-Box is said to satisfy SAC of 4-bit S-Boxes [9][10]. Review of old algorithms and new algorithms are described in section 3.

. In Higher Order Strict Avalanche Criterion (HO-SAC) of 4-bit Output BF like SAC four IPVs of a 4-bit Output BF have been complemented two or three at a time [11].In this Paper a new algorithm of 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 this paper. The aforesaid review and new algorithm of Extended HO-SAC have been viewed in section 4 and the analogy in between Extended HO-SAC and Differential Cryptanalysis of 4-bit S-Boxes has been described in

section 5. In section 6 conclusion has been made respectively. Algorithms or Pseudo Code of Algorithms of the respective sections have been given in Appendix.

2. A Review on Differential Cryptanalysis of 4-bit Bijective Crypto S-Boxes [2]. The given 4-bit Bijective crypto S-Box has been described in sub-section 2.1. The relation Between 4-bit S-Boxes and 4 bit BFs, The Differential Cryptanalysis of 4-bit S-Boxes are described in sub-section 2.2 and 2.3 respectively. DDT or Differential Distribution Table has been illustrated in sec 2.4.

2.1. 4-bit Crypto S-Boxes: A 4-bit bijective 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, 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 bijective crypto S-Boxes. The values of each element of the 1st row are distinct, unique and vary between 0 and F in hex. The values of the each element of the 2nd row of a bijective crypto S-Box are also distinct and unique and also vary between 0 and 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 [3][12][13].

2.2. Relation between 4-bit S-Boxes and 4-bit Boolean Functions (4-bit BFs). Index of Each element of a 4-bit bijective 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.2. 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 4-bit BF. column 1 through G of Row 2 is termed as 4th Input BF, Row 3 is termed as 3rd Input BF, Row 4 is termed as 2nd Input BF and Row 5 is termed as 1st Input BF whereas column 1 through G of Row 7 is termed as 4th Output BF, Row 8 is termed as 3rd Output BF, Row 9 is termed as 2nd Output BF and Row A is termed as 1st Output BF [3]. The decimal equivalent of each input BF and output BF are noted at column H of respective rows.

2.3. Review of Differential Cryptanalysis of 4-bit Crypto S-Boxes [9]. The column.1. in Table.3. from row 1 through G shows The 16 elements in a monotonically increasing sequence of Input 4-bit S-Box (ISB). The Input can also be concluded as an Identity 4-bit S-Box. The 1st 4-bit S-Box, out of 4 of 1st S-Box of Data Encryption Standard (DES) out of 8, has been considered as Output S-Box (OSB) in column 7 from row 1 through G. The review has been done in two different views; The S-box view has been described in sec 2.3.1. and the 4-bit binary pattern view has been described in sec. 2.3.2 respectively.

2.3.1. S-Box View of Differential Cryptanalysis of 4-bit Crypto S-Boxes. The S-Box of Input Difference element (ID) in which all elements have the same value 'B' in hex, is not a Crypto Box and is shown in row 1 through G of column. 3. of Table.3. The Distant Input S-Box (DISB) is shown in row 1 through G of column.5. In DISB each element row 1 through G is obtained by the xor operation of the elements in corresponding positions in row 1 through G of column.1. (ISB) and Column.3. (ID) respectively. In ISB for each element in in row 1 through G of column.1.in corresponding position in row 1 through G of column.7. there is an element of OSB. Now in DISB the elements of ISB have been shuffled in a particular order. In DOSB the corresponding elements of OSB has also been shuffled in that particular order in which ISB has been shuffled. Each element of the Difference S-Box or DSB in row 1 through G of column.12.has been obtained by xor operation of each element in corresponding positions of 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 Differential Distribution Table or DDT. It is shown in Table.4. as follows,

The count of each existing elements have been put into the Differential Distribution Table. As follows, in row 2 of Table.5. for Input Difference (ID) 'B' and Output Difference from 0 through F of row 1.

2.3.2 4-bit binary Pattern View of Differential Cryptanalysis of 4-bit Crypto S-Boxes. The Particular Input Difference '1101' is shown in row 1 through G of column. 4. of Table.3. The Distant 4-bit bit Patterns are shown in row 1 through G of column.5. The Distant 4-bit bit patterns are shown in row 1 through G of column.6. (Bin DISB) are obtained by the xor operation of the elements in corresponding positions in row 1 through G of column.2. (Bin ISB) and Column.4. (Bin ID) respectively. In Bin ISB for each element in in row 1 through G of column.2. in corresponding position in row 1 through G of column.8. there is an element of Bin OSB. Now in Bin DISB the elements of ISB have been shuffled in a particular order. In Bin DOSB the corresponding elements of OSB has also been shuffled in that particular order in which Bin ISB has been shuffled. Each element of row 1 through G of column.11.has been obtained by xor operation of each element in corresponding positions of row 1 through G of column.8. and row 1 through G of column.10. respectively. The repetition of each existing elements in Bin DSB have been counted and put into Differential Distribution Table or DDT. It is shown in Table.6. as follows,

The count of each existing elements have been put into the Differential Distribution Table. As follows, in row 2 of Table.7. for Binary Input Difference (Bin ID) '1101' and Output Difference from 0 through F of row 1.

2.4 Differential Cryptanalysis of 4-bit Bijective Crypto S-Boxes using 4-bit BFs. The Procedure to obtain Four Input BFs (IBFs) and Four Output BFs (OBFs) from a particular 4-bit Bijective Crypto S-Box has been described in sec.2.1. The procedure to obtain distant Input BFs (DIBFs) and Distant Output BFs (DOBFs) for a particular Input Difference (ID) has been described with example in sec.2.4.1.

2.4.1. Distant Input BFs (DIBFs) and Distant Output BFs (DOBFs) Generation from IBFs and OBFs for a specific ID. Meaning of 1 in ID is referred to as C or complement and 0 has also been referred to as N or No Complement as shown in Table.8. In 4-bit bit Patterns of Input Difference (Bin ID), 1 in a certain position 4 as in position 4 of row 1 through G of column 4 of table.3. means complement 4-bit BF, IBF4 (CIBF4) and 0 in a certain position 3 as in position 3 of row 1 through G of column 4 of table.3. means no operation on 4-bit BF IBF3 (CIBF3) or CIBF3 = IBF3. Similarly 1 in respective positions 2 and 1 as in positions 2 and 1 of row 1 through G of column 4 of table.3. means complement 4-bit BF IBF2 (CIBF2) and complement 4-bit BF IBF1 (CIBF1) respectively. CIBF4, CIBF3, CIBF2 and CIBF1 for Input S-Box (ISB) and Input Difference (ID) have been shown in row 1 through G of column.1. and column.3. of Table.3. respectively have been shown in column 1 through G of row 1, 2, 3 and 4 of table. 9. respectively.

Since complement of 4^{th} IBF means interchanging each 8 bit halves of 16 bit long 4^{th} IBF so The 2, 8 bit halves of OBF4 have been interchanged due to complement of 4^{th} IBF or CIBF4. The resultant OBF has been shown in column 1 through G of row 6 in Table.9. Again No Operation on 3^{rd} IBF means CIBF3 = IBF3 so resultant OBF is as same as STEP4 and has been shown in column 1 through G of row 7 in Table.9. Next to it since complement of 2^{nd} IBF means interchanging each 2 bit halves of each 4 bit halves of each 8 bit halves of resultant OBF has been shown in column 1 through G of row 7 in Table.9. and has been shown column 1 through G of row 8 in Table.9. Again since complement of 1^{st} IBF means interchanging each bit of each 2 bit halves of each 4 bit halves of each 8 bit halves of resultant OBF has been shown in column 1 through G of row 8 in Table.9 and has been shown in column 1 through G of row 9 in Table.9. The Complemented OBF4 has been the resultant OBF of STEP1 and has been shown in column 1 through G of row A in Table.9.

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

The DBFs of each OBF have been generated by bitwise Xor of OBFs and the corresponding DOBFs. The corresponding DBFs of OBF_4, OBF_3, OBF_2, OBF_1 are denoted as $DIFF_4$, $DIFF_3$, $DIFF_2$, $DIFF_1$ respectively.4^{th} DBF of ID '1101' has been shown in column 1 through G of row 3 of Table.10.

2.4.3 Analysis:

If the DBFs are balanced then the number of bits changed and remains unchanged among corresponding bits of OBFs and COBFs is maximum. So uncertainty of determining a particular change in bits is maximum. As the number of balanced DBFs are increased among 64 ($=2^4 \times 4$) possible DBFs then the security will increase. The no of 1s or balancedness of a particular DBF has been shown in column. 2. of row 2 of table.11.

2.4.4 DBFs Generation and Derivation of a Particular Row of Differential Analysis Table (DAT) for a Certain ID. Four IBFs in a order IBF4, IBF3, IBF2 and IBF1 for the S-Box given in Table.1. and four CIBFs CIBF4, CIBF3, CIBF2 and CIBF1 for a certain ID '1101' have been shown in column 1 through G of row 1, 2, 3, 4, 5, 6, 7 and 8 respectively in Table.12. Four OBFs in an order OBF4, OBF3, OBF2 and OBF1 for the S-Box given in Table.1. and four COBFs COBF4, COBF3, COBF2 and COBF1 for a certain ID '1101' have been shown in column 1 through G of row 9, A, B, C, D, E, F and G respectively in Table.12. The resultant DBFs, DIFF4, DIFF3, DIFF2, DIFF1, have been shown in column 1 through G of row H, I, J, K of Table.12. The number of 1s or Balancedness of four DBFs have been shown in row in column.2 through 5 of row 1 of Table.13.

2.4.6. Results. The results of Differential Boolean Function Analysis using DAT of 32 DES 4-bit Crypto S-Boxes and Their Comparison with No of zeros in DDT has been shown in Table.15.below.

3. Review of Strict Avalanche Criterion (SAC) of 4-bit BFs and SAC of 4-bit S-Boxes [7][8]. The Strict Avalanche Criterion or SAC of 4-bit BFs have been reviewed in sec. 3.1.1.and the new technique to find SAC of 4-bit BFs has been described in sec.3.1.2.SAC of 4-bit S-Boxes has also been reviewed in sec.3.2.1 and the new technique to find SAC of 4-bit S-Boxes using 4-bit BFs has been described in sec.3.2.2.

3.1.1. Review of Strict Avalanche Criterion (SAC) of 4-bit BFs. In Strict Avalanche Criterion of 4-bit BFs, IBF4, IBF3, IBF2 and IBF1 have been shown in column 1 through G of row 2, 3, 4 and 5 in Table.2, have been complemented one at a time. If due to aforesaid operation on OBF the number of bits has been changed for each IBF in COBF is 8 or half of the number of bits in a 4-bit BF then the OBF is said to satisfy SAC of 4-bit BFs.

IBF4, CIBF4, IBF3, CIBF3, IBF2, CIBF2, IBF1, CIBF1 have been shown in column 2 thorough H of row 1, 3, 7, 9 , D, F, J, L respectively of table.16. The OBF and COBF due to change in CIBF4, CIBF3, CIBF2 and CIBF1 have been shown in column 2 thorough H of row 2, 4, 8, A, E, G and K, M respectively. The Difference BF 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 COBF from OBF due to CIBF4, CIBF3, CIBF2 and CIBF1 have been 12, 8, 4, 12. So OBF does not Satisfy SAC of 4-bit BFs. To Satisfy SAC change in Number of bits in COBF from OBF due to CIBF4, CIBF3, CIBF2 and CIBF1 must be 8, 8, 8, 8.

3.1.2. New Method for Strict Avalanche Criterion (SAC) of 4-bit BFs. Since complement of 4th IBF means interchanging each 8 bit halves of 16 bit long 4th IBF so The 2, 8 bit halves of OBF has been interchanged due to complement of 4th IBF or CIBF in COBF. Next to it since complement of 3rd IBF means interchanging each 4 bit halves of each 8 bit halves of IBF3 in CIBF or so each 4 bit halves of each 8 bit halves of OBF in COBF have been interchanged due to complement of IBF3.Now complement of 2nd IBF means interchanging each 2 bit halves of each 4 bit halves of each 8 bit halves of OBF in COBF and complement of 1st IBF means interchanging each bit of each 2 bit halves of each 4 bit halves of each 8 bit halves of OBF in COBF.

IBF4, CIBF4, IBF3, CIBF3, IBF2, CIBF2, IBF1, CIBF1 have been shown in column 2 thorough H of row 1, 3, 7, 9 , D, F, J, L respectively of table.16. The OBF and COBF due to change in CIBF4, CIBF3, CIBF2 and CIBF1 have been shown in column 2 thorough H of row 2, 4, 8, A, E, G and K, M respectively. The Difference BF 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 COBF from OBF due to CIBF4, CIBF3, CIBF2 and CIBF1 have been 12, 8, 4, 12. So OBF does not Satisfy SAC of 4-bit BFs. To Satisfy SAC change in Number of bits in COBF from OBF due to CIBF4, CIBF3, CIBF2 and CIBF1 must be 8, 8, 8, 8.

3.2.1. Review of Strict Avalanche Criterion (SAC) of 4-bit S-Boxes [14].

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) have been given in column 2 through H of row 3, 1, 2 of Table.18 respectively. Each Output BF, OBF1, OBF2, OBF3, OBF4 has been shown in column 2 through H of row 4, 5, 6, 7 Table.18 respectively.

Now 16 INBs before flip and 16 INBs after flip in one bits particularly in 16 INBs bit position 1, 2, 3, 4 have been shown in row 2 through I of column 1, 2, 6, 7, B, C, G, H respectively of Table.19. The each corresponding bits of OBF1, OBF2, OBF3, OBF4 before and after flip have been shown in row 2 through I of column 3, 4, 8, 9, D, E, I, J respectively in Table.19. 1 in any position in row 2 through I of column 5, A, F, K illustrate dissimilarity in bits in corresponding positions of OBF1, OBF2, OBF3 and OBF4 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 I 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 is said to satisfy SAC of 4-bit S-Boxes. Here in Table. 19. row I shows the number of Bits changed in OBF1, OBF2, OBF3, OBF4 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 OBF so The BF and the given 4-bit S-Box does Satisfy SAC of 4-bit BFs and SAC of 4-bit S-Boxes.

3.2.2. New Method for Strict Avalanche Criterion (SAC) of 4-bit Crypto S-Boxes. If all four BFs of a given 4-bit Crypto S-Box satisfy SAC of 4-bit BFs then the S-Box is said to satisfy SAC of 4-bit S-Boxes.

4. Review and New Methods of Higher Order SAC (HO-SAC) [11] and Extended SAC Criterion of 4-bit Crypto S-Boxes.

The Method of HO-SAC of 4-bit BFs has been reviewed in sec 4.1. The two new methods of HO-SAC of 4-bit BFs and HO-SAC of 4-bit Crypto S-Boxes have been described in sec 4.2. The new SAC criterion entitled Extended SAC criterion of 4-bit BFs and 4-bit S-Boxes has been illustrated in sec. 4.3. All Algorithms of respective sections are given in Appendix.

4.1 Review of Higher Order SAC of 4-bit BFs. 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. 20. and Table.21. respectively. 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 it have been given in column 2 through H of row 3, 1, 2 of Table.20 respectively. Each Output BF, OBF1, OBF2, OBF3, OBF4 has been shown in column 2 through H of row 4, 5, 6, 7 Table.20 respectively.

If 2 IBFs 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 21-A to 21-F. If the Number of IBFs 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 21-H to 21-J.

In 2nd Order HO-SAC, any 2 IBFs shown in column 2 through H of row 1, 2 of sub table 21-A to 21-F of Table.21 have been complemented at time. The complemented IBFs or CIBFs have been shown in column 2 through H of row 4, 5 of sub table 21-A to 21-F of Table.21.The OBF has been shown in column 2 through H of row 3 of sub table 21-A to 21-F of Table.21. Now due to complement of 1st IBF the resultant OBF have been shown in row 6 and due complement of 2nd IBF at time the resultant OBF of the resultant OBF have been shown in column 2 through H of row 7 of sub table 21-A to 21-F of Table.21 respectively. The obtained complemented OBF or COBF and bitwise Xor or Hamming distance (Distant BF or DBF) between OBF and COBF have been shown in column 2 through H of row 8, 9 of sub table 21-A to 21-F of Table.21 respectively. The count of 1 out of 16 bits in DBF or dissimilar bits in COBF due to complement of 2 IBFs at time have been shown in A of sub table 21-A to 21-F of Table.21. If for the given OBF and for all 6 possible 2nd order HO-SAC test the counts have been shown in A of sub table 21-A to

21-F of Table.21 have been 8 then the given OBF is said to satisfy 2nd order HO-SAC of 4-bit BFs. But in table 21 all counts are not equal to 8 so the given OBF does not satisfy 2nd order HO-SAC of 4-bit BFs.

In 3rd Order HO-SAC, any 3 IBFs shown in column 2 through H of row 1, 2, 3 of sub table 21-G to 21-J of Table.21 have been complemented at time. The complemented IBFs or CIBFs have been shown in column 2 through H of row 5, 6, 7 of sub table 21-G to 21-J of Table.21. The OBF has been shown in column 2 through H of row 4 of sub table 21-G to 21-J of Table.21. Now due to complement of 1st IBF the resultant OBF have been shown in row 8 and due to complement of 2nd IBF at a time the resultant OBF of the resultant OBF in row 8 have been shown in column 2 through H of row 9 and due to complement of 3rd IBF at a time the resultant OBF of the resultant OBF in row 9 have been shown in column 2 through H of row A of sub table 21-G to 21-J of Table.21 respectively. The obtained complemented OBF or COBF and bitwise Xor or Hamming distance (Distant BF or DBF) between OBF and COBF have been shown in column 2 through H of row B, C of sub table 21-G to 21-J of Table.21. respectively. If for the given OBF and for all 4 possible 3rd order HO-SAC test the counts have been shown in D of sub table 21-G to 21-J of Table.21 have been 8 then the given OBF is said to satisfy 3rd order HO-SAC of 4-bit BFs. But in table 21 all counts are not equal to 8 so the given OBF does not satisfy 3rd order HO-SAC of 4-bit BFs.

If the given OBF satisfy both 2nd order HO-SAC and 3rd Order HO-SAC for 4-bit BFs together then The Given OBF is said to satisfy Total HO-SAC for 4-bit BFs. If Four BFs of a Crypto 4-bit S-Box satisfy HO-SAC of 4bit BFs individually then the S-Box has been said to satisfy HO-SAC of 4-bit Crypto S-Boxes.

4.2. 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 4.2.1. The Flip Method to do the same test of 4-bit BFs and of 4-bit S-Boxes has been described in sec. 4.2.2.

4.2.1 Shift Method of HO-SAC of 4-bit BFs and 4-bit Crypto S-Boxes. Complement of IBF4 is equivalent of interchanging two 8 bit halves of OBF. Complement of IBF3 is equivalent of interchanging each 4 bit halves of each 8 bit halves of given OBF. Now Complement of IBF2 means interchanging each 2 bit halves of each 4 bit halves of each 8 bit halves of given OBF and finally Complement of IBF1 means interchanging each two bits of all two bit halves of given OBF.

If 2 IBFs 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 21-A to 21-F. If the Number of IBFs 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 21-H to 21-J.

In 2nd Order HO-SAC, any 2 IBFs shown in column 2 through H of row 1, 2 of sub table 21-A to 21-F of Table.21 have been complemented at time. The complemented or shifted IBFs or CIBFs have been shown in column 2 through H of row 4, 5 of sub table 21-A to 21-F of Table.21.The OBF has been shown in column 2 through H of row 3 of sub table 21-A to 21-F of Table.21. Now due to complement of 1st IBF the resultant OBF have been shown in row 6 due to shift operation and due to complement of 2nd IBF at a time the resultant OBF of the resultant OBF have been shown in column 2 through H of row 7 of sub table 21-A to 21-F of Table.21 respectively. The obtained complemented OBF or COBF and bitwise Xor or Hamming distance (Distant BF or DBF) between OBF and COBF have been shown in column 2 through H of row 8, 9 of sub table 21-A to 21-F of Table.21 respectively. The count of 1s out of 16 bits in DBF or dissimilar bits in COBF due to complement of 2 IBFs at a time have been shown in A of sub table 21-A to 21-F of Table.21. If for the given OBF and for all 6 possible 2nd order HO-SAC test the counts have been shown in A of sub table 21-A to 21-F of Table.21 have been 8 then the given OBF is said to satisfy 2nd order HO-SAC of 4-bit BFs. But in table 21 all counts are not equal to 8 so the given OBF does not satisfy 2nd order HO-SAC of 4-bit BFs.

In 3rd Order HO-SAC, any 3 IBFs shown in column 2 through H of row 1, 2, 3 of sub table 21-G to 21-J of Table.21 have been complemented at time. The complemented IBFs or CIBFs have been shown in column 2 through H of row 5, 6, 7 of sub table 21-G to 21-J of Table.21. The OBF has been shown in column 2 through H of row 4 of sub table 21-G to 21-J of Table.21. Now due to complement or shift of 1st IBF the resultant OBF have been shown in row 8 after shift operation and due to complement of 2nd IBF at a time the resultant OBF of the resultant OBF in row 8 have been shown in column 2 through H of row 9 and due to complement of 3rd IBF at a time the resultant OBF of the resultant OBF in row 9 have been shown in column 2 through H of row A of sub table 21-G to 21-J of Table.21 respectively. The obtained complemented or shifted OBF or COBF and bitwise Xor or Hamming distance (Distant BF or DBF) between OBF and COBF have been shown in column 2 through H of row B, C of sub table 21-G to 21-J of Table.21. respectively. If for the given OBF and for all 4 possible 3rd order HO-SAC test the counts have been shown in D of sub table 21-G to 21-J of Table.21 have been 8 then the given OBF is said to satisfy 3rd order HO-SAC of 4-bit BFs. But in Table.21. all counts are not equal to 8 so the given OBF does not satisfy 3rd order HO-SAC of 4-bit BFs.

If the given OBF satisfy both 2nd order HO-SAC and 3rd Order HO-SAC for 4-bit BFs together then The Given OBF is said to satisfy Total HO-SAC for 4-bit BFs. If Four BFs of a Crypto 4-bit S-Box satisfy HO-SAC of 4bit BFs individually then the S-Box has been said to satisfy HO-SAC of 4-bit Crypto S-Boxes.

4.2.2 Flip Method of HO-SAC of 4-bit BFs and 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 it have been given in column 2 through H of row 3, 1, 2 of Table.20 respectively. Each Output BF, OBF1, OBF2, OBF3, OBF4 has been shown in column 2 through H of row 4, 5, 6, 7 Table.20 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, 4 have been shown in row 2 through I of column 1, 2, 6, 7, B, C, G, H respectively of Table.22-A, Table.22-B and Table.22-C respectively . The each corresponding bits of OBF1, OBF2, OBF3, OBF4 before and after flip have been shown in row 2 through I of column 3, 4, 8, 9, D, E, I, J respectively in Table.22-A, Table.22-B and Table.22-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 I of column 5, A, F, K illustrate dissimilarity in bits in corresponding positions of OBF1, OBF2, OBF3 and OBF4 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 I column of column 5, A, F, K there are 8 1s and 8 0s then The given BF is said to Satisfy HO-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 is said to satisfy SAC of 4-bit S-Boxes. Here in Table. 22. row I shows the Number of Bits changed in OBF1, OBF2, OBF3, OBF4 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 OBF so The BF and the given 4-bit S-Box does Satisfy SAC of 4-bit BFs and SAC of 4-bit S-Boxes.

4.3. Extended HO-SAC Criterion of 4-bit BFs and 4-bit Crypto S-Boxes. If one IBF out of four at a time, Two IBFs out of four at a time, Three IBFs out of four at a time and all of four IBFs have been complemented at a time or flip of one bit of INB at a time, flip of two INBs at a time or flip of three INBs at a time, and flip of all of four INBs at a time and all DBFs are 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 16, 21, 18 and 22 respectively for the given 4-bit BF. The rest Criterion of 4 IBFs have been complemented at a time have been shown in Sub table 23-A of table. 23. The rest flip of all INBs at a time have been shown in Sub table 23-B of table. 23.

5. Analogy Between Extended HO-SAC of 4-bit S-Boxes and Differential Cryptanalysis of 4-bit S-Boxes and Improvement in Differential Cryptanalysis of 4-bit S-Boxes using 4-bit BFs.

Since Flip in one, two, three and four INBs or similarly complement of one, two, three and four IBFs is similar to xor operation of all 4-bit Input Difference with 4-bit bit patterns of each element of a 4-bit Crypto S-Box 0001-1111 the operations in Extended HO-SAC test shown in tables 16, 18, 21, 22, 23 is similar to Operations in Differential Cryptanalysis of 4-bit S-Boxes by Heys in table .3. In Differential Cryptanalysis of 4-bit S-Boxes using 4-bit BFs, Each BF has been tested through Extended HO-SAC Criterion to ensure security. So ot is claimed to be a better analytic tool of 4-bit Crypto-S-Boxes.

6. Conclusion.

In this paper a better look to SAC of 4-bit BFs and S-Boxes, HO-SAC 4-bit BFs and S-Boxes and Differential Cryptanalysis of 4-bit S-Boxes with proper extensions to theories have been presented. An Extension to HO-SAC entitled Extended HO-SAC criterion have also been defined to prove the enrichment of The Differential Cryptanalysis of 4-bit S-Crypto S-Boxes using 4-bit BFs. The total paper have been claimed with clarity of review sections and extension to theories. At last all these Algorithms have been tested and proved to be a better one to improvement of crypto-community.

7. References.

1. Stallings W., Cryptography and Network Security Principles and Practices, Delhi, Pearson Education, 4th Edition, 2008.
2. Forouzan B.A., Mukhopadhyay D., Cryptography and Network Security, New Delhi, TMH, 2nd Edition, 2011.
3. Adams, Carlisle, Tavares, Stafford, “The structured design of cryptographically good S-boxes”, J. Cryptology (1990) vol. 3, pp : 27-41.
4. Eli Biham, Adi Shamir(1990), “Differential Cryptanalysis of DES-like cryptosystems”. Advances in Cryptology-CRYPTO ’90. Springer- Verlag. pp: 2-21
5. Heys, Howard M, Tavares, Stafford E, “Substitution-Permutation Networks Resistant to Differential and Linear Cryptanalysis”, J. Cryptology (1996) 9: pp: 1–19.
6. Heys, Howard M, “A Tutorial on Linear and Differential Cryptanalysis”, Link: http://www.engr.mun.ca/~howard/PAPERS/ldc_tutorial.pdf.
7. Webster, A.F. and Travares, S.E."On Design of S-Boxes", Advances in Cryptology : Proc. of Crypto 85, Springer-Verlag pp. 523-534, 1986.
8. Vergili, Isil, Yiicel, melek D, “On satisfaction of some Security Criteria for Randomely Chosen S-Boxes”,Link: http://www.eee.metu.edu.tr/~yucel/OnSatisfacSomeSecurCriterForRanChosenSBox.pdf.
9. K.Kim,T.Matsumoto, and H.Imai” A recursive construction method of SBoxes Satisfying the Strict Avalanche Criterion.”in Proc. of CRYPTO’90(Springer-Verlag,1990), pp. 565-574.
10. A.V.Sokolov. “Constructive Method for the Synthesis of Nonlinear S-Boxes Satisfying the Strict Avalanche Criterion”,ISSN 0735-2727,Radioelectronics and Communication Systems,2013,vol.56,No.8,pp.415-423,Alerton Press Inc. 2013.
11. Forei, Rkjane “The Strict Avalanche Criterion Spectral Properties of Boolean Functions and an Extended Definition”, Link: https://bitbucket.org/.../The%20strict%20avalanche%20criterion-spectral.

12. Data Encryption Standard, Federal Information Processing Standards Publication (FIPS PUB) 46, National Bureau of Standards, Washington, DC (1977).
13. 2. Data Encryption Standard (DES), Federal Information Processing Standards Publication (FIPS PUB) 46-3, National Institute of Standards and Technology, Gaithersburg, MD (1999).

Appendix

1. Algorithm of Differential Cryptanalysis Binary Pattern View.

Start.
Step 0A: int ISB[4][16]; int IDIFF[4][16]; int ODIFF[4][16][16];
Step 0B: int ISB'[4][16][16]; int OSB[4][16]; int OSB'[4][16][16]; int DDT[16][16]; int Count[16];
Step 01: For I =1:16
For J =1:16
For K =1:4
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.
End For.
End For.
Step 02: For I =1:16
For J =1:16
For K =1:4
DDT[I][J]= Count[ISB[K][J]];
End For.
End For.
End For.
Stop.

2. Algorithm of Differential Cryptanalysis S-Box View.

Start.
Step 0A: int ISB[16]; int IDIFF[16]; int ODIFF[16][16];
Step 0B: int ISB'[16][16]; int OSB[16]; int OSB'[16][16]; int DDT[16][16]; int Count[16].
Step 01: For I =1:16
For J =1:16
ISB'[I][J] = ISB[J]^IDIFF[I];
OSB'[I][J] = OSB[ISB'[I][J]];
ODIFF[I][J] = OSB[J]^ OSB'[I][J];
End For.
End For.
Step 02: For I =1:16
For J =1:16
DDT[I][J]= Count[ISB[J]];
End For.
End For.
Stop.

3. Algorithm of Differential Cryptanalysis Using 4-bit BFs.

Start.
Step 0A: int IBF[4][16]; int IDIFF[4][16][16]; int ODIFF[4][16][16];
Step 0B: int IBF'[4][16][16]; int OBF[4][16]; int OBF'[4][16][16]; int DAT[4][16]; int Count[16][16];
Step 01: For I = 1:16 For J = 1:4 For K = 1:16 OBF'[J][K][I] = OBF[IBF[J][K]^IDIFF[J][K][I]][K];
Step 02: For I = 1:16 For J = 1:4 For K = 1:16 ODIFF[J][K][I] = OBF'[J][K][I]^ OBF[J][K];
Step 03: For I = 1:16 For J = 1:4 For K = 1:16 Count[J][K] = IF(ODIFF[J][K][I]==1);
Step 04: For I = 1:4 For J = 1:16 DAT[I][J] = Count[I][J];
Stop.

4. Algorithm of SAC of 4-bit BFs.

Let BF[16].bit0 is a bit level array of 16 bits of a 4-bit BF out of 65536 4-bit BFs. And BF[16] is an array of 16 bits of a 4-bit BF. CV[16].bit0 is a bit level array of 16 bits to store either 00FF, 0F0F, 3333, 5555 in hex. CVC[16].bit0 is a bit level array of 16 bits to store either FF00, F0F0, CCCC, AAAA in hex. Here ^ represents Bitwise Xor operation. NL represents 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.
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.

5. Algorithm of 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);
Step 1B: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0;
Step 1C: Count = IF(DBF[16].bit0==1);
Step 2A: CBF[16].bit0 = (BF[8A].bit0>>4)&& (BF[8B].bit0>>4);
Step 2B: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0;
Step 2C: Count = IF(DBF[16].bit0==1);
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;
Step 3C: Count = IF(DBF[16].bit0==1);
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;
Step 4C: Count = IF(DBF[16].bit0==1);
Step 05 : IF Count = 8 for Step 1C, Step 2C, Step 3C, Step 4C. BF[16] Satisfies SAC of 4-bit BFs.
ELSE BF[16] does not Satisfy SAC of 4-bit BFs.
Stop.

6. Algorithm of SAC of 4-bit BFs Using Flip Method.

The flipping of bits on particular positions are made by proposing 1-bit 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;
Step 0B: ev[4] ={{0,0,0,1},{0,0,1,0},{0,1,0,0},{1,0,0,0}};
Step 01: For S=0:4 For I=0:16 For J=0:16 t[S][I][J] = 16bt4x[S][I][J] ^ ev[S]
Step 02: For S=0:4 For I=0:16 For J=0:16 r=16bt4bf[S][I][J] ^ 16bt4bf[t[S][I][J]];
Step 04: if (r==1) D[f][v]++;
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.

7. Algorithm of HO-SAC of 4-bit BFs.

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.
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.
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..

8. Algorithm of HO-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);
Step 1B: CBF[16].bit0 = (CBF[8A].bit0>>4)&& (CBF[8B].bit0>>4);
Step 1C: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0;
Step 1D: Count = IF(DBF[16].bit0==1);
Step 2A: CBF[16].bit0 = (BF[16].bit0>>8);
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;
Step 2D: Count = IF(DBF[16].bit0==1);
Step 3A: CBF[16].bit0 = (BF[16].bit0>>8);
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;
Step 3D: Count = IF(DBF[16].bit0==1);
Step 4A: CBF[16].bit0 = (BF[8A].bit0>>4)&& (BF[8B].bit0>>4);
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;
Step 4D: Count = IF(DBF[16].bit0==1);
Step 5A: CBF[16].bit0 = (BF[8A].bit0>>4)&& (BF[8B].bit0>>4);
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;
Step 5D: Count = IF(DBF[16].bit0==1);
Step 6A: CBF[16].bit0 = (BF[4A].bit0>>2)&& (BF[4B].bit0>>2)&& (BF[4C].bit0>>2)&& (BF[4D].bit0>>2);

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;
Step 6D: Count = IF(DBF[16].bit0==1);
Step 7A: CBF[16].bit0 = (BF[16].bit0>>8);
Step 7B: CBF[16].bit0 = (CBF[8A].bit0>>4)&& (CBF[8B].bit0>>4);
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;
Step 7E: Count = IF(DBF[16].bit0==1);
Step 8A: CBF[16].bit0 = (BF[16].bit0>>8);
Step 8B: CBF[16].bit0 = (CBF[8A].bit0>>4)&& (CBF[8B].bit0>>4);
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;
Step 8E: Count = IF(DBF[16].bit0==1);
Step 9A: CBF[16].bit0 = (BF[16].bit0>>8);
Step 9B: CBF[16].bit0 = (CBF[4A].bit0>>2)&& (CBF[4B].bit0>>2)&& (CBF[4C].bit0>>2)&& (CBF[4D].bit0>>2);
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;
Step 9E: Count = IF(DBF[16].bit0==1);
Step 10A: CBF[16].bit0 = (CBF[8A].bit0>>4)&& (CBF[8B].bit0>>4);
Step 10B: CBF[16].bit0 = (CBF[4A].bit0>>2)&& (CBF[4B].bit0>>2)&& (CBF[4C].bit0>>2)&& (CBF[4D].bit0>>2);
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;
Step 10E: Count = IF(DBF[16].bit0==1);
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.

Step A: CBF[16].bit0 = (BF[16].bit0>>8);
Step B: CBF[16].bit0 = (BF[8A].bit0>>4)&& (BF[8B].bit0>>4);
Step C: CBF[16].bit0 = (BF[4A].bit0>>2)&& (BF[4B].bit0>>2)&& (BF[4C].bit0>>2)&& (BF[4D].bit0>>2);
Step D: 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);

9. Algorithm of HO-SAC of 4-bit BFs Using Flip Method.
Start.
Step 0A: For I=0:16 For J=0:16 D[I][J] = 0;
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]
Step 02: For S=0:4 For I=0:16 For J=0:16 r=16bt4bf[S][I][J] ^ 16bt4bf[t[S][I][J]];
Step 04: if (r==1) D[f][v]++;
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.

10. Algorithm of Extended SAC of 4-bit BFs.

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.
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

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..

Algorithm of Extended SAC of 4-bit BFs Using Shift Method.

Step 00: For 1:16 BF[16].bit0 = BF[16].
Step 1A: CBF[16].bit0 = (BF[16].bit0>>8);
Step 1B: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0;
Step 1C: Count = IF(DBF[16].bit0==1);
Step 2A: CBF[16].bit0 = (BF[8A].bit0>>4)&& (BF[8B].bit0>>4);
Step 2B: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0;
Step 2C: Count = IF(DBF[16].bit0==1);
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;
Step 3C: Count = IF(DBF[16].bit0==1);
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;
Step 4C: Count = IF(DBF[16].bit0==1);
Step 5A: CBF[16].bit0 = (BF[16].bit0>>8);
Step 5B: CBF[16].bit0 = (CBF[8A].bit0>>4)&& (CBF[8B].bit0>>4);
Step 5C: DBF[16].bit0 = CBF[16].bit0^ BF[16].bit0;
Step 5D: Count = IF(DBF[16].bit0==1);
Step 6A: CBF[16].bit0 = (BF[16].bit0>>8);
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;
Step 6D: Count = IF(DBF[16].bit0==1);
Step 7A: CBF[16].bit0 = (BF[16].bit0>>8);
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;
Step 7D: Count = IF(DBF[16].bit0==1);
Step 8A: CBF[16].bit0 = (BF[8A].bit0>>4)&& (BF[8B].bit0>>4);
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;
Step 8D: Count = IF(DBF[16].bit0==1);
Step 9A: CBF[16].bit0 = (BF[8A].bit0>>4)&& (BF[8B].bit0>>4);
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;
Step 9D: Count = IF(DBF[16].bit0==1);
Step 10A: CBF[16].bit0 = (BF[4A].bit0>>2)&& (BF[4B].bit0>>2)&& (BF[4C].bit0>>2)&& (BF[4D].bit0>>2);

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;
Step 10D: Count = IF(DBF[16].bit0==1);
Step 11A: CBF[16].bit0 = (BF[16].bit0>>8);
Step 11B: CBF[16].bit0 = (CBF[8A].bit0>>4)&& (CBF[8B].bit0>>4);
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;
Step 11E: Count = IF(DBF[16].bit0==1);
Step 12A: CBF[16].bit0 = (BF[16].bit0>>8);
Step 12B: CBF[16].bit0 = (CBF[8A].bit0>>4)&& (CBF[8B].bit0>>4);
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;
Step 12E: Count = IF(DBF[16].bit0==1);
Step 13A: CBF[16].bit0 = (BF[16].bit0>>8);
Step 13B: CBF[16].bit0 = (CBF[4A].bit0>>2)&& (CBF[4B].bit0>>2)&& (CBF[4C].bit0>>2)&& (CBF[4D].bit0>>2);
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;
Step 13E: Count = IF(DBF[16].bit0==1);
Step 14A: CBF[16].bit0 = (CBF[8A].bit0>>4)&& (CBF[8B].bit0>>4);
Step 14B: CBF[16].bit0 = (CBF[4A].bit0>>2)&& (CBF[4B].bit0>>2)&& (CBF[4C].bit0>>2)&& (CBF[4D].bit0>>2);
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;
Step 14E: Count = IF(DBF[16].bit0==1);
Step 15A: CBF[16].bit0 = (BF[16].bit0>>8);
Step 15B: CBF[16].bit0 = (CBF[8A].bit0>>4)&& (CBF[8B].bit0>>4);
Step 15C: CBF[16].bit0 = (CBF[4A].bit0>>2)&& (CBF[4B].bit0>>2)&& (CBF[4C].bit0>>2)&& (CBF[4D].bit0>>2);
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;
Step 15F: Count = IF(DBF[16].bit0==1);

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.

11. Algorithm 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;
Step 0B: ev[4] ={{0,0,0,1},{0,0,1,0},{0,1,0,0},{1,0,0,0},
{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]
Step 02: For S=0:4 For I=0:16 For J=0:16 r=16bt4bf[S][I][J] ^ 16bt4bf[t[S][I][J]];
Step 04: if (r==1) D[f][v]++;
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.

Table 1(on next page)

Table.1.

Title. Table.1.

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 bijective Crypto S-Box.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3202v1 | CC BY 4.0 Open Access | rec: 28 Aug 2017, publ: 28 Aug 2017

Table 2(on next page)

Table.2.

Title. Table.2.

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	**IBF4**	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	00255
3	**IBF3**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	03855
4	**IBF2**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	13107
5	**IBF1**	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	**OBF4**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0	42836
8	**OBF3**	1	1	1	0	0	1	0	0	0	0	1	1	1	0	0	1	58425
9	**OBF2**	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	1	36577
A	**OBF1**	0	0	1	1	0	1	1	0	1	0	0	0	1	1	0	1	13965

Table.2. Decomposition of 4-bit Input and Output (1st 4-bit S-Box of 1st S-Box out of 8 of DES) S-Boxes to 4-bit BFs.

Table 3(on next page)

Table.3.

Title. Table.3.

C	1	2	3	4	5	6	7	8	9	A	B	C
ROW	ISB	Bin ISB 4321	ID	Bin ID 4321	DISB	Bin DISB 4321	OSB	Bin OSB 4321	DOSB	Bin DOSB 4321	Bin DSB 4321	DSB
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.3. Table of Differential Cryptanalysis of 1st 4-bit S-Box of 1st S-Box out of 8 of DES.

Table 4(on next page)

Table.4.

1 **Title. Table.4.**

2

3

4

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

5

6 **Table.4. Count of repetition of each existing element in DSB.**

7

Table 5(on next page)

Table.5.

Title. Table.5.

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.5. The Part of DDT with Input Difference 'B'.

Table 6(on next page)

Table.6.

Title. Table.6.

R|C	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
1	**DSB 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.6. Count of repetition of each existing element in Bin DSB.

Table 7 (on next page)

Table.7.

Title. Table.7.

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.7. The Part of DDT with Input Difference '1101'.

Table 8(on next page)

Table.8.

Title. Table.8.

ID	1	1	0	1
Complement	C	C	N	C

Table.8. Complement of IBFs Due to a Particular ID

Table 9(on next page)

Table.9.

Title. Table.9.

Row | Col		1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G
1	**CIBF4**	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
2	**CIBF3**	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
3	**CIBF2**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
4	**CIBF1**	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
5	**OBF4**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
6	**STEP4**	0	1	0	1	0	1	0	0	1	0	1	0	0	1	1	1
7	**STEP3**	0	1	0	0	0	1	0	1	0	1	1	1	1	0	1	0
8	**STEP2**	0	0	0	1	0	1	0	1	1	1	0	1	1	0	1	0
9	**STEP1**	0	0	1	0	1	0	1	0	1	1	1	0	0	1	0	1
A	**COBF4**	0	0	1	0	1	0	1	0	1	1	1	0	0	1	0	1

Table. 9. Construction of DIBFs and DOBFs.

Table 10(on next page)

Table.10.

Title. Table.10.

R|C		1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G
1	**OBF4**	1	0	1	0	0	1	1	0	0	0	0	1	0	1	0	0
2	**COBF4**	0	0	1	0	1	0	1	0	1	1	1	0	0	1	0	1
3	**DIFF4**	1	0	0	0	1	1	0	0	1	1	1	1	0	0	0	1

Table.10. DBF Generation.

Table 11(on next page)

Table.11.

Title. Table.11.

R|C	1	2
1	**Difference BF**	**Total Number of 1s**
2	DIFF4	8

Table. 11. DBF and Balancedness.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3202v1 | CC BY 4.0 Open Access | rec: 28 Aug 2017, publ: 28 Aug 2017

Table 12(on next page)

Table.12.

Title. Table.12.

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

Table 13(on next page)

Table.13.

Title. Table.13.

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

Table.13. Balancedness of four DBFs.

Table 14(on next page)

Table.14.

Title. Table.14.

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.14. DAT for 1st 4-bit S-Box of 1st S-Box of DES

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3202v1 | CC BY 4.0 Open Access | rec: 28 Aug 2017, publ: 28 Aug 2017

Table 15(on next page)

Table.15.

Title . Table.15.

4-Bit S-Box	Total no of 8s in DAT	Total no of 0s in DDT
E 4 D 1 2 F B 8 3 A 6 C 5 9 0 7	36	168
0 F 7 4 E 2 D 1 A 6 C B 9 5 3 8	36	168
4 1 E 8 D 6 2 B F C 9 7 3 A 5 0	36	168
F C 8 2 4 9 1 7 5 B 3 E A 0 6 D	42	166
F 1 8 E 6 B 3 4 9 7 2 D C 0 5 A	30	162
3 D 4 7 F 2 8 E C 0 1 A 6 9 B 5	30	166
0 E 7 B A 4 D 1 5 8 C 6 9 3 2 F	21	166
D 8 A 1 3 F 4 2 B 6 7 C 0 5 E 9	36	168
A 0 9 E 6 3 F 5 1 D C 7 B 4 2 8	30	162
D 7 0 9 3 4 6 A 2 8 5 E C B F 1	30	168
D 6 4 9 8 F 3 0 B 1 2 C 5 A E 7	21	166
1 A D 0 6 9 8 7 4 F E 3 B 5 2 C	30	174
7 D E 3 0 6 9 A 1 2 8 5 B C 4 F	36	168
D 8 B 5 6 F 0 3 4 7 2 C 1 A E 9	36	168
A 6 9 0 C B 7 D F 1 3 E 5 2 8 4	36	168
3 F 0 6 A 1 D 8 9 4 5 B C 7 2 E	36	168
2 C 4 1 7 A B 6 8 5 3 F D 0 E 9	30	162
E B 2 C 4 7 D 1 5 0 F A 3 9 8 6	36	166
4 2 1 B A D 7 8 F 9 C 5 6 3 0 E	27	160
B 8 C 7 1 E 2 D 6 F 0 9 A 4 5 3	18	166
C 1 A F 9 2 6 8 0 D 3 4 E 7 5 B	36	159
A F 4 2 7 C 9 5 6 1 D E 0 B 3 8	36	164
9 E F 5 2 8 C 3 7 0 4 A 1 D B 6	18	168
4 3 2 C 9 5 F A B E 1 7 6 0 8 D	30	162
4 B 2 E F 0 8 D 3 C 9 7 5 A 6 1	30	168
D 0 B 7 4 9 1 A E 3 5 C 2 F 8 6	30	166
1 4 B D C 3 7 E A F 6 8 0 5 9 2	36	168
6 B D 8 1 4 A 7 9 5 0 F E 2 3 C	0	173
D 2 8 4 6 F B 1 A 9 3 E 5 0 C 7	30	161
1 F D 8 A 3 7 4 C 5 6 B 0 E 9 2	27	174
7 B 4 1 9 C E 2 0 6 A D F 3 5 8	18	166
2 1 E 7 4 A 8 D F C 9 0 3 5 6 B	39	168

Table.15. Differential BF Analysis using DAT and Comparison to no of zeros in DDT for 32 DES 4-bit S-Boxes.

Table 16(on next page)

Table.16.

Title . Table.16.

R|C	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
1	**IBF4**	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
2	**OBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
3	**CIBF4**	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
4	**COBF**	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 COBF												12				

R|C	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
7	**IBF3**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
8	**OBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
9	**CIBF3**	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
A	**COBF**	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 COBF												8				

R|C	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
D	**IBF2**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
E	**OBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
F	**CIBF2**	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
G	**COBF**	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 COBF												4				

R|C	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
J	**IBF1**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
K	**OBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
L	**CIBF1**	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
M	**COBF**	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 COBF												12				

11 **Table.16. SAC Criterion for 4-bit BFs.**

12

13

14

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3202v1 | CC BY 4.0 Open Access | rec: 28 Aug 2017, publ: 28 Aug 2017

Table 17 (on next page)

Table.18.

Title . Table.18.

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.18. S-Box and OBFs for SAC Test of 4-bit BFs as well as 4-bit Crypto S-Boxes.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3202v1 | CC BY 4.0 Open Access | rec: 28 Aug 2017, publ: 28 Aug 2017

Table 18(on next page)

Table.19.

Title . Table.19.

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.19. SAC Test of 4-bit BFs and 4-Bit Crypto S-Boxes.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3202v1 | CC BY 4.0 Open Access | rec: 28 Aug 2017, publ: 28 Aug 2017

Table 19(on next page)

Table 20.

Title . Table.20.

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.20. S-Box and OBFs for HO-SAC Test of 4-bit BFs as well as 4-bit Crypto S-Boxes.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3202v1 | CC BY 4.0 Open Access | rec: 28 Aug 2017, publ: 28 Aug 2017

Table 20(on next page)

Table.21.

Title . Table.21.

R|C	21-A	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
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	**OBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
4	**CIBF4**	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
5	**CIBF3**	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	**COBF**	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 COBF												6				

R|C	21-B	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
1	**IBF4**	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
2	**IBF2**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
3	**OBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
4	**CIBF4**	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
5	**CIBF2**	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	**COBF**	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 COBF												12				

R|C	21-C	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
1	**IBF4**	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
2	**IBF1**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
3	**OBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
4	**CIBF4**	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
5	**CIBF1**	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	**COBF**	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 COBF												8				

R|C	21-D	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
1	**IBF3**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
2	**IBF1**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
3	**OBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
4	**CIBF3**	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
5	**CIBF1**	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	**COBF**	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 COBF												8				

15
16

R|C	**21-E**	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
1	**IBF2**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
2	**IBF1**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
3	**OBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
4	**CIBF2**	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
5	**CIBF1**	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	**COBF**	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 COBF												12				

17

R|C	**21-F**	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
1	**IBF3**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
2	**IBF2**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
3	**OBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
4	**CIBF3**	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
5	**CIBF2**	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	**COBF**	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 COBF												8				

18

R|C	**21-G**	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
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	**OBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
5	**CIBF4**	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
6	**CIBF3**	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
7	**CIBF2**	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	**COBF**	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 COBF												8				

19
20
21
22
23
24

R|C	**21-H**	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
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	**IBF1**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
4	**OBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
5	**CIBF4**	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

6	**CIBF3**	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
7	**CIBF1**	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	**COBF**	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 COBF												8				

25

R|C	**21-I**	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
1	**IBF4**	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
2	**IBF2**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
3	**IBF1**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
4	**OBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
5	**CIBF4**	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
6	**CIBF2**	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
7	**CIBF1**	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	**COBF**	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 COBF												4				

26

R|C	**21-J**	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
1	**IBF3**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
2	**IBF2**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
3	**IBF1**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
4	**OBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
5	**CIBF3**	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
6	**CIBF2**	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
7	**CIBF1**	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	**COBF**	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 COBF												8				

27
28 **Table.21. Table of HO-SAC Test and Complement Method of HO-SAC Test of 4-bit BFs**
29

30

Table 21(on next page)

Table.22.

Title . Table.22.

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

Table 22(on next page)

Table.23.

Title . Table.23.

R|C	23-A	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
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	**OBF**	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0
6	**CIBF4**	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
7	**CIBF3**	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
8	**CIBF2**	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
9	**CIBF1**	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	**COBF**	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 COBF												8				

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