Journal of Mathematics and Statistics 5 (1): 24-31, 2009
ISSN 1549-3644

The New Variable-Length Key Symmetric Cryptosystem

[1]Davood Rezaei Pour, [1]Mohamad Rushdan Md Said, [1]Kamel Ariffin Mohd Atan and [2]Mohamed Othman
[1]Institute for Mathematical Research (INSPEM)
[2]Faculty of Computer Science and Information Technology
University Putra Malaysia, 43400 UPM Serdang, Selangor Darul Ehsan, Malaysia

Abstract: Problem statement: In this study, we proposed a new 64-bit block cipher that accepted a variable-length key up to 512 bits, which was suitable for implementation in a variety of environments. **Approach:** The cipher algorithm was a 16-round Feistel network with a bijective function f and was made up of two key-dependent 16×16 S-boxes, bitwise rotations, and a carefully designed key schedule. **Results:** The block cipher, what we called NBC08, was designed to perform under the powerful operations supported in today's computers, resulting in an improved security/performance tradeoff over existing block ciphers. **Conclusion:** The study concluded the differential, linear and algebraic cryptanalysis on the NBC08 and showed that the cipher cannot be analyzed by any cryptanalytic attack. The statistical test results for NBC08 did not indicate a deviation from random behavior.

Key words: Block ciphers, Feistel structure, key dependent s-boxes, attacks

INTRODUCTION

We describe NBC08 a new block cipher supporting 64-bit blocks and variable key size, ranging from 192 to 512 bits. The "cryptographic core" of the NBC08 cipher is a Feistel structure, consisting of sixteen rounds. It is surrounded by two Non-Feistel keyed transformations, and thus, the algorithm structure is non-homogeneous[1]. These transformations in the first part and the last part are inverse of each other. These transformations increase the complexity of attacks on NBC08, resulting in improved security. The round function in the main core of encryption process is designed on the basis of this theory: The provable security against both linear and differential attacks[2]. This algorithm is used in all the standard modes of operation for block ciphers.

In this study, we describe the design principles of NBC08. It covers the high-level structure of the cipher, initial and final transforms, and used functions in NBC08. The key schedule is included the key expansion and the sub keys generation.

The statistical randomness tests investigate to existence of special weaknesses in algorithm. If an algorithm is accepted in all of statistical tests, then it doesn't have known weaknesses. The results show that these tests do not identify any deviation from random behavior.

NBC08 uses key dependent S-boxes, round dependent function and the rotation based on sub key value. This creates suitable shield against both linear and differential cryptanalysis.

MATERIALS AND METHODS

Specification:

The basic operations of NBC08: NBC08 cipher uses a variety of operations on 16-bit words. It combines exclusive-ors (XORs), additions, subtractions, rotations and S-box lookups. We describe these operations in Table 1.

The first three operations are used to "mix together" data values and key values (on 16-bit words), which are done very fast on modern processors.

Also, we use a left-rotation by y, for a 16-bit word x, inside the round function. Here y is viewed as a 4-bit integer, and the rotation amount (between 0 and 15) is specified by it.

Table 1: Symbol of operations

Operating symbols	Operations
⊞	Additions modulo 2^{16}
⊟	Subtractions modulo 2^{16}
⊕	XORs
x ↓ <<< ← y	Rotations
S	S-box

Corresponding Author: Davood RezaeiPour, Institute for Mathematical Research (INSPEM), University Putra Malaysia, 43400 UPM Serdang, Selangor Darul Ehsan, Malaysia Tel: +601-73507481, Fax: +603-89423789

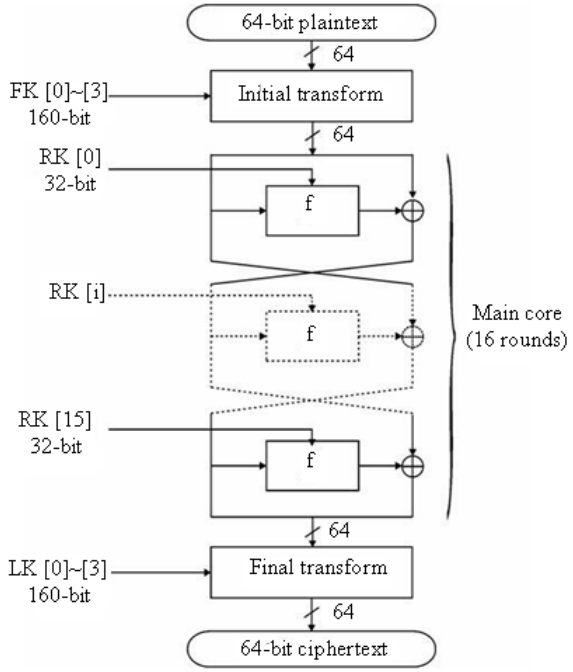

Fig. 1: High-level structure of the cipher

S7 and S9: S7 and S9 are 7×7 S-box and 9×9 S-box, respectively. These S-boxes are the main sources of nonlinearity in NBC08 which are designed by the nonlinear functions. Their inverses are $S7^{-1}$ and $S9^{-1}$, respectively.

S: S function is a 16×16 (keyed) S-box, which is made by two S-boxes (S7 and S9). Similarly, S^{-1} is the inverse of S, which is made by $S7^{-1}$ and $S9^{-1}$.

f: f is the round function in the encryption process and the cryptographic strength depends only on the properties of this function.

High-level structure of the cipher:

The cipher consists of three parts: Initial Transform, Main Core and Final Transform. The main core is a Feistel structure, consisting of sixteen rounds. The general structure of NBC08 cipher is depicted in Fig. 1.

This process uses a total of 1792 bits of sub key material, for encryption and decryption procedures. These sub keys are derived from the master key using the key schedule:

- 4 160-bit sub keys FK[0]~FK[3] for the first 4 rounds(Initial Transform)
- 16 32-bit sub keys RK[0]~RK[15] for 16 rounds of main core
- 4 160-bit sub keys LK[0]~LK[3] for the last 4 rounds(Final Transform)

Below, we describe the block cipher in details:

Initial transform: The operations of initial transform are shown in Fig. 2. This transformation is formed by the same four rounds. First, 64-bit input data block is divided into 4 16-bit sub blocks X1~X4. Then, it is repeated four rounds on 16-bit sub blocks.

The operations for the first round of Initial Transform are depicted in Fig. 2. In the first round, 160-bit sub key FK[0] is divided into 10 16-bit sub keys FK[0][0]~FK[0][9], which mid operations $\boxplus$, $\oplus$ and S function make the round operations. In the next rounds, the sub keys FK[1]~FK[3] are used.

Final transform: These operations are the inverse of initial transform, such that if the same key is used for these two transformations, then the operations thwart each other. This transformation is depicted in Fig. 3.

The 160-bit sub key LK[3] is used in the first round of final transform and sub keys LK[0]~LK[2] are used in the next rounds of final transform.

The round function f: Figure 4 shows the round function f in the encryption process of NBC08. First, 32-bit half block input is divided into 2 16-bit sub blocks left and right. Also, the input sub key of function f in the ith round, RK[i], is divided identically into RK[i][0] and RK[i][1].

In this function, the operations are done over all the 16-bit sub blocks. First, the input sub key of the function is combined with 2 input data sub blocks through a nonlinear process. Then, these 2 16-bit sub blocks create a 32-bit half block output duration of operations (3 rounds) with using of S function.

S function and its inverse S^{-1}: In the function f, S function is a 16×16 (keyed) S-box. For implementation, the dimensions of S-box are as a big called table. So, we choose the structure of Fig. 5 for S functions[3].

As Fig. 5 shows, the 16-bit input to S is divided into two parts, 9-bit and 7-bit. This non-identical division causes more resistance of algorithm against linear and differential attacks, because the bijective functions with odd dimensions are better than the functions with even dimensions, from the viewpoint of the provable security against both linear and differential attacks[4].

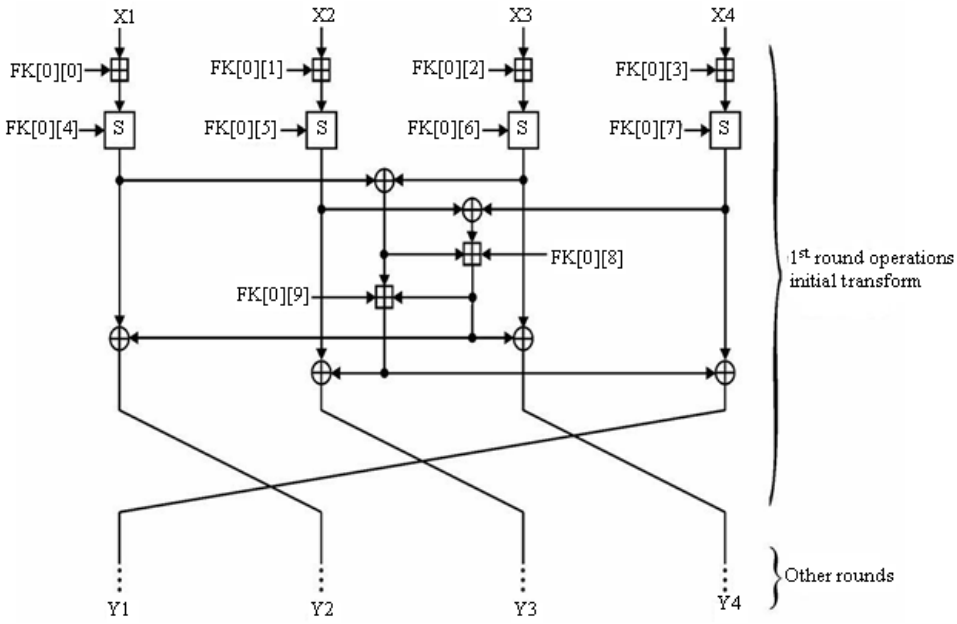

Fig. 2: Initial Transform in encryption process

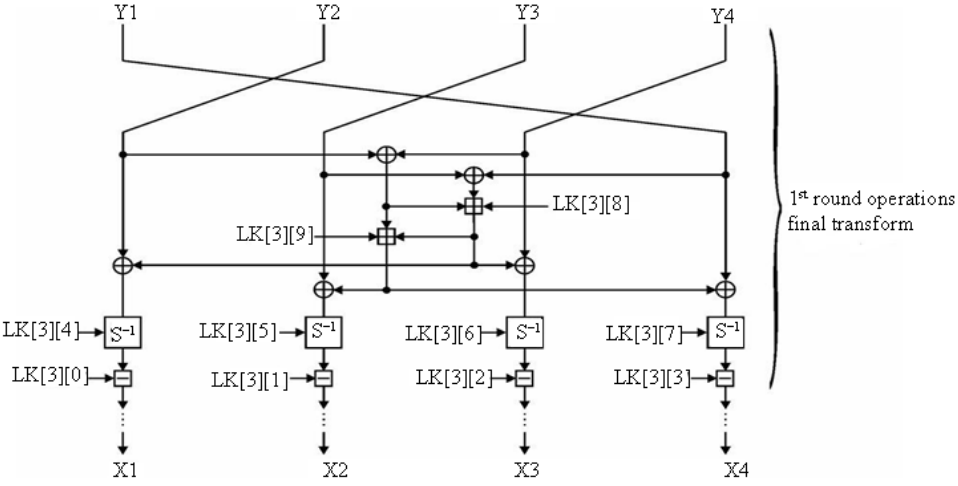

Fig. 3: Final Transform in encryption process

In Fig. 5, two S-boxes S7 and S9 are used with 7-bit and 9-bit input, respectively. These S-boxes are objective and designed by the nonlinear functions[5-8]. When S9 output bits are XORed with 7-bit data, it is expanded to 9-bit by adding two zero bits to the left and when S7 output is XORed with 9-bit data, it is decreased to 7-bit by truncating from the left.

The perfect nonlinear functions in $GF(2^9)$ and $GF(2^7)$ are used for generating of substitution tables (S-boxes) S9 and S7, respectively.

As a result, these S-boxes provide good resistance against linear and differential attacks[9].
The functions used are:

$S7(X) = (47X+123)^{111} \bmod 131$
$S9(X) = (47X+213)^{383} \bmod 529$

For achieving high speed in software, all outputs of S7 and S9 are calculated for different inputs. Each of these outputs is stored in memory arrays which are used

in the program as called tables. In C implementation, we can define the called tables (corresponding S7 and S9), as shown below:

```
unsigned short const
S7[128] =  {
71, 95, 16,  5, 17,103, 38, 89, 68, 23,111,125, 48, 42,
80,  4, 37, 36, 30, 91,117, 97, 52, 43, 76, 13, 93, 96,120,
27, 60,105, 65, 53, 26, 81, 84,121, 61, 49,113, 33, 66,
6, 59,102,123,  3, 18,124, 85, 55, 99, 14, 83, 64,126, 50,
15,  51,108,  92,  56,  87,  75,  29,  19,  25,109,  63,116,
58,114, 70, 82,119, 78, 94, 73, 88, 110, 28, 45,100, 12,
79, 98,106, 10, 77,  8, 46, 67, 54, 86, 72, 74,104,115,
57, 11, 35, 34, 32,  1,122,118,  7,127,112, 69, 41, 0, 44,
40, 62,  2,  9, 22,101, 21,107, 31, 39, 20, 47, 24, 90
};

S9[512] =  {
458,11,257,116,381,511,256,469,260,270,169,308,358,
210,  26,489,  177,163,416,273,  66,323,  18,367,337,
19,114,335,397,  69,245,  80,  383,  6,431,296,  97,133,
23,  59,  49,173,175,174,  15,  50,477,  76,406,468,
57,409,421,111,120,376,232,  51,  65,291,238,  28,494,
95,364,189,298,119,289,390,492,139,242,216,456,251,
165,64,433,91,161,37,462,14,290,274,109,400,317,239,
354,389,219,386,322,201,193,309,0,247,501,496,300,3
56,328,286,440,181,89,204,265,196,357,62,370,295,22
4,186,79,485,12,441,22,326,123,124,378,74,7,185,345,
203,395,84,304,70,214,244,369,205,88,403,197,125,23
1,411,336,234,437,113,466,321,218,457,158,  73,  36,
17,240,  46,  282,442,384,276,  47,327,424,380,  71,473,
41,429,510,211,63,314,344,31,208,452,140,147,233,0,2
78,305,217,475,443,207,118,225,200,303,425,474,407,
268,453,172,90,199,48,191,509,162,426,227,318,506,3
3,319,280,94,379,263,448,507,85,83,347,194,132,235,1
37,112,447,  60,253,320,117,104,460,215,341,258,131,
39,  4,275,145,  93,  9,188,159,294,168,226,497,372,285,
53,266,222,495,151,373,220,135,272,377,2,449,255,39
9,254,110,54,334,446,153,67,228,338,130,350,213,385,
44,353,316,243,434,82,170,96,43,187,108,164,154,351,
81,329,310,279,198,16,136,77,483,455,391,281,121,14
3,348,  87,502,283,362,157,410,146,331,423,382,166,
21,106,293,396,105,271,343,418,122,299,413,183,
1,288,156,241,  10,171,464,  98,179,  25,  24,129,107,484,
61,142,436,428,499,393,401,180,32,292,508,486,363,2
30,366,284,209,445,375,155,  75,361,467,  29,346,102,
13,38,45,355,432,478,419,115,488,387,430,249,398,48
7,307,20,269,439,86,306,42,287,144,435,127,99,472,40
4,202,103,149,371,229,3,490,52,461,427,264,325,463,
8,480,394,236,126,342,360,58,479,134,374,35,212,339,
352,27,68,206,252,315,405,182,402,392,148,451,160,1
52,190,454,470,55,302,250,92,101,450,333,150,503,31
2,237,471,340,141,349,481,223,505,500,412,465,417,
246,178,438,330,476,30,267,221,5,262,167,491,359,49
3,72,176,195,498,482,259,415,332,365,192,368,301,29
7,459,504,184,313,261,408,248,128,277,100,56,420,13
8,  78,444,324,388,422,414,311,  34
};
```

S7 and S9 are not keyed and are used as fixed tables. In Fig. 5, it uses XOR operation for making S, which is dependent on the key. As noticed, 16-bit key K is divided into 7-bit K7 and 9-bit K9, which are 7-bit from right and 9-bit from left, respectively. The interference of key K in S structure increases the complexity of system.

The function S^{-1} is the inverse of S function. The inverses of S7 and S9 are used in this function, namely $S7^{-1}$ and $S9^{-1}$. The tables $S7^{-1}$ and $S9^{-1}$ are obtained easily from S7 and S9, respectively. The operations of S^{-1} are depicted in Fig. 6.

Key schedule: The key schedule has been chosen according to the following criteria:

- The main key is from 192 to 512 bits
- The bits in each sub key should depends on the all bits in the master key
- There are no specific relations between the sub keys

Since the conditions above are satisfied, the algorithm NBC08 has no weak keys.

For generating of required sub keys in the encryption process, we mainly utilize the round function in 3-round Feistel structure using CBC mode. The key schedule consists of two algorithms, the Key Expansion procedure which expands master key to 512-bit key length, and Sub keys Generation which generates the sub keys FK[0]~FK[3], RK[0]~RK[15] and LK[0]~LK[3].

Key expansion: In our study, the master key length increases to 512-bit. We use 2 512-bit arrays consisting of 32 16-bit words EK and SK. These arrays and the bits numbering manner are depicted in Fig. 7.

The master key bits settle in the array EK. The first bit of master key is bit 0 in EK. The remaining bits of EK are completed by bit 0. SK is a fixed key array consisting of random binary sequence 512-bit. This vector is constant for an algorithm; actually it plays the user key role. The information about SK contents is not important because it satisfies the random conditions. Below we define SK.

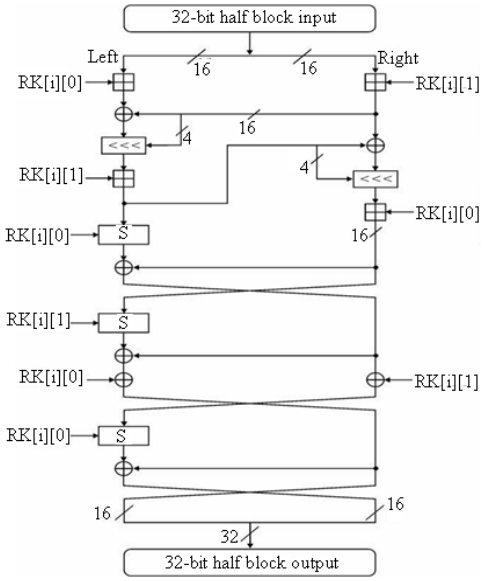

Fig. 4: The Round Function f

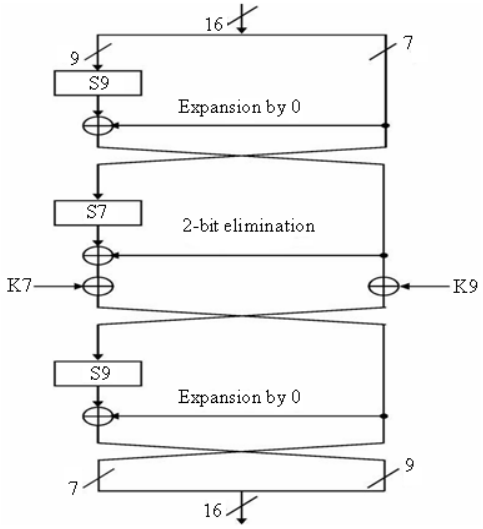

Fig. 5: S Function

```
unsigned short const SK[32] =
{
0x9e59, 0xc9ac, 0xfb36, 0x7a45, 0xa1ab, 0x146d, 0xfb96, 0x36f8,
0xea17, 0x183c, 0xc200, 0xaddc, 0x9099, 0xd956, 0x4fe2, 0x1c1c,
0x2afe, 0xc694, 0x1fc0, 0xbb5b, 0x1e89, 0x5f4c, 0x6e6f, 0x8da7,
0x7c98, 0xe31e, 0xdb92, 0x3076, 0x4245, 0xeb86, 0x90a5, 0x7678
};
```

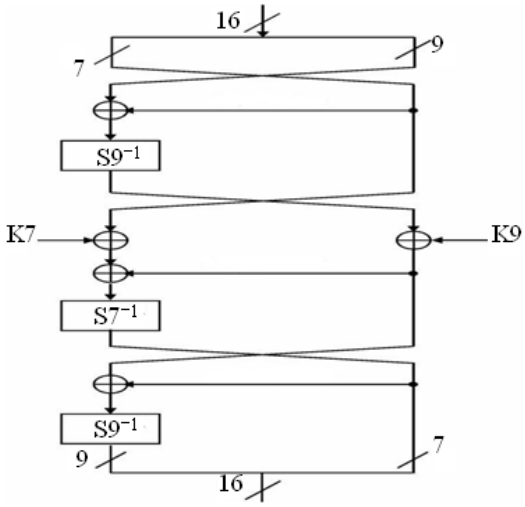

Fig. 6: S^{-1} Function

For the expanding of master key length to 512-bit, we add the SK contents to EK using accumulation manner. The following C program is used for this purpose,

```
EK [0] += SK[0]
for (int i=1;  i<31  ;  ++ i)
    EK [i] += EK [i-1]+  SK[i] ;
```

Sub keys generation: We use the structure shown in Fig. 8, for the sub keys generation:

As seen in Fig. 8, a 3-round Feistel structure using f function in CBC mode is used for sub keys generation. The initial inputs are: (EK[26],EK[27]),(EK[28],EK[29]),(EK[30],EK[31]) which are used as 3 sub keys for 3 rounds. (EK[26],EK[27]) means that 2 16-bit words are put together and make a 32-bit word.

(EK[0] ,EK[1], EK[2] ,EK[3]) is 64-bit input and (EK[22] ,EK[23], EK[24] ,EK[25]) is initial value for CBC method.

The output of the first step with CBC creates the first 64-bit sub key. So, for generating the required sub keys, the structure of Fig. 8 should perform 28 times.

The elements of EK array are used in quadric sets as the inputs of CBC method. When this array ended, the elements, again as quadric sets settle in the input.

The sub keys in the encryption process are generated by the outputs of the sub key generation algorithm as follows:

FK[0] is fulfilled by the first 160-bit and also FK[3] by the fourth 160-bit. Afterward, the next generated 32-bit is used as RK[0] and continues till RK[15] is also fulfilled. The remaining bits are used for LK[0]~LK[3].

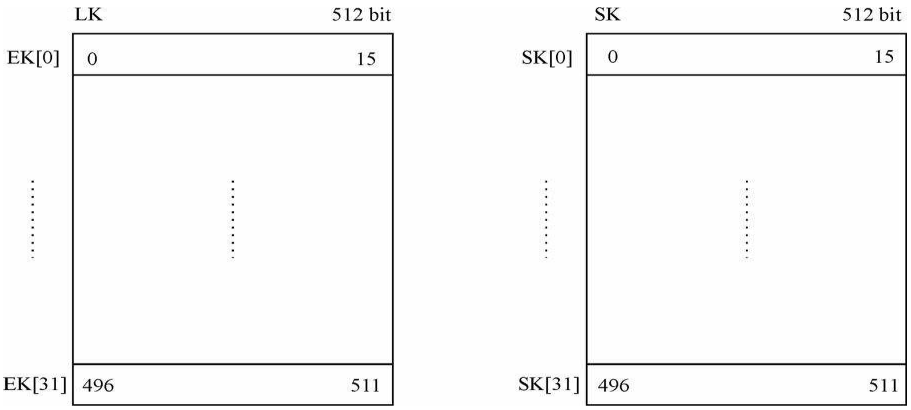

Fig. 7: The arrays SK and EK

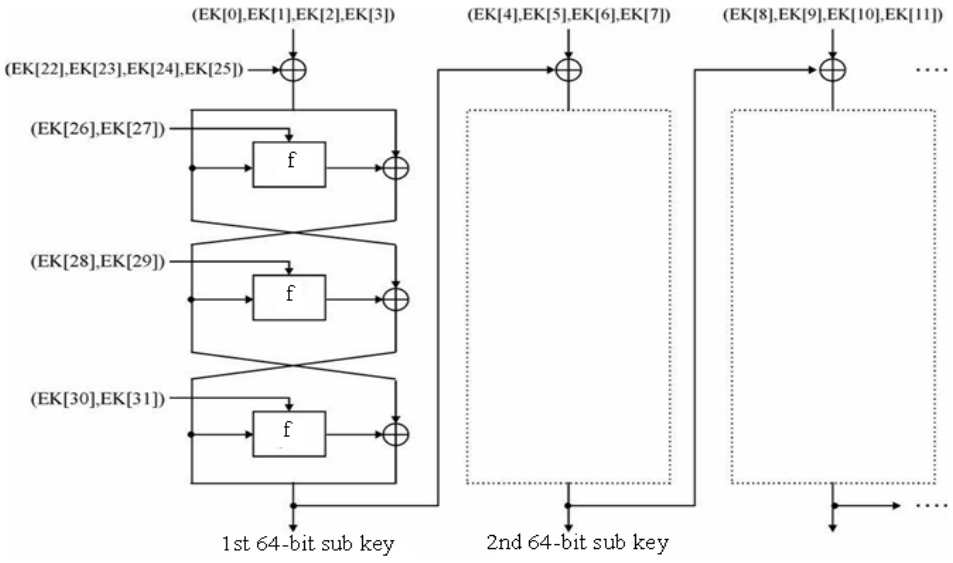

Fig. 8: Sub keys generation

RESULTS AND DISCUSSION

Security analysis: To estimate security of NBC08, some known attacks of block ciphers are considered below.

Statistical analysis: The NBC08 block cipher is tested using sixteen statistical tests[10,11]. We used 500 samples of about 10^6 bit sequences for each test. Table 2 shows results of the NIST statistical test suite for NBC08. In Table 2, the parenthesis beside the name of the statistical test shows the input parameters used in the test. From the Table 2 we see that the statistical test results for NBC08 do not indicate a deviation from random behavior. These tests are essential but not sufficient for security.

Differential and linear cryptanalysis: NBC08 uses key dependent S-boxes, round dependent function and the rotation based on sub key value. This creates suitable shield against both linear and differential cryptanalysis.

Suppose that we have a set of n pairs of plaintexts/ciphertexts, and then the attacker will try to find differential or linear property between the plaintext/ciphertext pairs with a high probability to utilize it in extracting some bits of the master key. In the proposed algorithm, the attacker will not be able to know the sequences of the operations and the rotations used in the algorithm since the order of the sequences and operations depends on the master key.

Table 2: Results of statistical analysis of NBC08

	Proportion	
Statistical test	High density	Low density
Frequency	0.995 (pass)	0.988 (pass)
Block Frequency (m = 100)	0.994 (pass)	0.992 (pass)
Runs	0.991 (pass)	0.981 (pass)
Long Runs of Ones	0.990 (pass)	0.993 (pass)
Rank	0.989 (pass)	0.991 (pass)
Spectral DFT	0.998 (pass)	0.990 (pass)
Non-overlapping Templates (m = 9)	0.989 (pass)	0.991 (pass)
Overlapping Templates (m = 9)	0.981 (pass)	0.982 (pass)
Universal	0.989 (pass)	0.980 (pass)
Lempel-Ziv Complexity	0.989 (pass)	0.982 (pass)
Linear Complexity (M = 500)	0.985 (pass)	0.990 (pass)
Serial (m = 5)	0.990 (pass)	0.983 (pass)
Approximate entropy (m = 5)	0.983 (pass)	0.989 (pass)
Cusum	0.989 (pass)	0.985 (pass)
Random Excursions	0.986 (pass)	0.988 (pass)
Random Excursions Variant	0.987 (pass)	0.982 (pass)

In this algorithm, we have 6 different operations, and 32 different rotations. The permutation of the operations is 6! and the permutation of the rotation is 32!. This provides us $6!\times32!\approx2^{127}$ different sequences. It means that the attacker should try 2^{127} different cases. For every case, the attacker has to find the linear or differential properties, and then uses the available pairs of plaintexts/ciphertexts to find some bits of the key. However, this attack is more effective than the exhaustive key search.

Moreover, with having n pairs of plaintexts/ciphertexts, attacker should use all the pairs to extract ℓ bits from the key, and apply plaintexts/ciphertexts $n(2^{127})$ times depending on the different sequences. These operations are considered better than the exhaustive key search.

For a 192-bit key length, the operations are $O(2^{192})$. By considering that the attacker has extracted ℓ bits, then the operations are $n(2^{127})$, and the exhaustive search for the rest of the remaining bits is $2^{192-\ell}$. Therefore, the attack will be better than exhaustive key search if $2^{192}>2^{192-\ell}$ $(n)2^{127}$, namely $2^{\ell}>n(2^{127})$. However this is so difficult to attain. For example, if the attacker has n = 50 plaintexts, then he/she should extract more than 130 bits from the user key to be able to achieve the attack faster than the exhaustive key search, that is $\ell>130$ bits. With n = 50 plaintexts, it is very difficult to extract these bits from the input user key faster than the exhaustive key search. The same case can be performed for other key lengths.

Algebraic cryptanalysis: In performing algebraic attack[12] to block ciphers, we have to derive an over-defined system of algebraic equations. Since the full-round NBC08 has a high degree as a vector Boolean function, so it is impossible to convert any equation system in NBC08 into an over-defined system.

CONCLUSION

The cipher algorithm NBC08 is designed to perform the encryption and decryption processes over 64-bit data blocks. The key length is from 192 to 512 bits. It is used in all the standard modes of operation for block ciphers. NBC08 is easy and fast. The implementation NBC08 is done with less volume, high speed and is optimum over the 8, 16 and 32-bit processors, since the operations are done on 16-bit words. The current implementation is written in C and runs at rates of about 64 M bit sec^{-1}, on an 800MHz Pentium with the Windows XP operating system.

NBC08 is used as "building blocks" in the design of other cryptographic algorithms, such as stream ciphers, Message Authentication Codes (MACs) and hash functions.

The statistical test results for NBC08 do not indicate a deviation from random behavior: NBC08 has provable security against both linear and differential attacks. This algorithm is resistant against exhaustive key search. The other known attacks over block ciphers are not practical on the NBC08.

REFERENCES

1. Schneier, B., 1996. Applied Cryptography: Protocols, Algorithms and Source Code in C. 2nd Edn., John Wiley and Sons, Inc. USA., pp: 758. ISBN-10: 0471117099.
2. Nyberg, K. and L.R. Knudsen, 1995. Provable security against a differential attack. J. Cryptol., 8: 27-37. https://eprints.kfupm.edu.sa/59816/1/59816.pdf
3. Matsui, M., 1997. New block encryption algorithm MISTY. Proceedings of the 4th International Workshop on Fast Software Encryption. Jan. 20-22, Springer-Verlag London, UK., pp: 54-58. http://portal.acm.org/citation.cfm?id=740728
4. Matsui, M., 1996. New structure of block ciphers with provable security against differential and linear cryptanalysis. Proceedings of the 3rd International Workshop on Fast Software Encryption, Feb. 21-23, Springer-Verlag London, UK, pp: 205-218. http://portal.acm.org/citation.cfm?id=740582

5. Nyberg, K., 1991. Perfect nonlinear S_boxes. Proceeding of the Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology, April 8-11, Brighton, UK., pp: 378-386. http://dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/E91/378.PDF
6. Nyberg, K., 1993. On the construction of highly nonlinear permutations. Lecture Notes in Comput. Sci., 658: 92-98. http://www.tcs.hut.fi/Publications/info/knyberg.nyberg:ec92.shtml
7. Pieprzyk, J., 1990. Nonlinearity of exponent permutations. Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology, Apr. 10-13, Houthalen, Belgium, pp: 80-92. http://portal.acm.org/citation.cfm?id=111563.111573
8. Beth, T. and C. Ding, 1994. On almost perfect nonlinear permutations. Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology, May 23-27, Lofthus, Norway, pp: 65-76. http://portal.acm.org/citation.cfm?id=188324
9. Knudsen, L.R., 1994. Practically secure feistel ciphers, fast software encryption. Proceeding of the Cambridge Security Workshop, Dec. 1993, Springer-Verlag, Cambridge, UK., pp: 211-221. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.36.1000
10. Menezes, A.J., P.C. Van Oorschot and S.A. Vanstone, 1997. Handbook of Applied Cryptography. Reviser Edn., CRC Press, Inc., Boca Raton, FL, USA., ISBN 0849385237, pp: 780.
11. Rukhin, A., J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray and S. Vo, 2001. A statistical test suite for random and pseudorandom number generators for cryptographic applications. http://csrc.nist.gov/publications/nistpubs/800-22-rev1/SP800-22rev1.pdf
12. Courtois, N. and J. Pieprzyk, 2002. Cryptanalysis of block ciphers with over-defined systems of equations. Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology, Dec. 1-5, Springer-Verlag London, UK., pp: 267-287. http://portal.acm.org/citation.cfm?id=647098.717146&coll=GUIDE&dl=GUIDE