5

A NOVEL 256-BIT BLOCK CIPHER

Mohamed Fahmy Tolba *mtolba@geganet.com.*

Mohamed Saeed Abdel Wahab *wahabms@hotmail.com.*
Ashraf Saad Hussien *ahrafh@acm.org.*
Mohamed Ahmed Abo El-Fotouh *midono1@hotmail.com.*
Faculty of Computers and Information Sciences,Ain Shams University,Cairo, Egypt.

Abstract: Tornado is a novel 256-bit block cipher that accepts a variable-length key up to 256 bits. The proposed cipher was built on an idea which gives it a dynamic nature. This nature means that the cipher execution flow varies according to the user's input key. The confusion and diffusion processes are performed based on 10 round functions named "Storm-functions". A selection and reordering operations are performed on those functions, using a fast hashing function based on the input key. The proposed cipher has a carefully designed key scheduling algorithm. The Software implementation of the proposed cipher gives high throughput. Performance study demonstrated that, Tornado is faster than the AES by a speed-up percentage of 253%. Yet, we believe that Tornado is more secure than AES.

Keywords: Cryptography, Dynamic encryption, Block cipher, Tornado.

1. INTRODUCTION

Shared-key (symmetric) encryption is perhaps the most fundamental cryptographic task. It is used in a large variety of applications, including protection of the secrecy of login passwords, ATM PINS, e-mail messages, video transmissions (such as pay-per-view movies), stored data files, and Internet-distributed digital content. It is also used to protect the integrity of banking and point-of-sale transactions, in the key distribution protocols (such as Kerberos), and many other applications. For many applications, the Data

Encryption Standard algorithm (DES) is nearing the end of its useful life. Its 56-bit key is too small, as shown by a recent distributed key search exercise. Although triple-DES can solve the key length problem, the DES algorithm was also designed primarily for hardware encryption, yet the great majority of applications that use it today implement it in software, where it is relatively inefficient.

For these reasons, the US National Institute of Standards and Technology (NIST) has issued a call for a successor algorithm, to be called the Advanced Encryption Standard (AES). The essential requirement is that AES should be both faster than triple DES and at least as secure: it should have a 128 bit block length and a 256 bit key length (though keys of 128 and 192 bits must also be supported).On October 2, 2000 the National Institute of Standards and Technology (NIST) announced that had been chosen to become the Advanced Encryption Standard.

In this paper, a new symmetric block cipher called Tornado is presented, which has both 256-bit block and key size; it has a novel idea that makes it impressive, as it operates in 232 different ways. This enforces blind search over the key space which is computationally infeasible. Tornado consists of 10 round functions “Storm-Functions”; the sequence of applying and choosing the candidates Storm-Functions depends on the users input key that makes Tornado dynamic key dependent algorithm. In addition to the creative key scheduling algorithm, that uses some of the cryptographic core of the algorithm to increase key scheduling security, this method was inspired by the famous Blowfish and Twofish algorithms.

The presented algorithm has been implemented in software using C/C++ language. This implementation shows a high speed-up factor in comparison to other famous and standard algorithms. This paper is organized as followed; first the building blocks of the cipher are presented. Subsequently, the design of Tornado key scheduling. This is followed by the cipher it-self, and its security. Finally, we present the performance of the cipher, our conclusion and future work.

2. TORNADO BUILDING BLOCKS

Tornado consists of 10 rounds each round consumes 8x32-bits of the encryption key. Rounds in Tornado are called Storm-functions. The sequence of applying Storm-Functions depends on the user’s input key after padding and modification. The building blocks of the cipher, presented in this section are the M-functions which are used in the encryption process.

Tornado has 10 M-functions, which are the basic elements in constructing Storm-functions (i.e. M-Functions are the elementary units of the cryptographic core of Tornado). Each M-function accepts three 32-bit words, and uses the last two words to modify the first one. Here, the general structure of the M-functions is presented:

```
Function M ”index” (input : X, Y, K)
        if (N == 1) then
                X= (~X)
        End if
        X=X Op1 Y
        X=X Op2 K
        X=RotateWord32Left(X, RV)
        Output(X)
```

The values of “N”, “Op1”, “Op2” and “RV” for all the M-functions are presented in table (1). Here, the addition process is addition modulo 232, the subtraction process is subtraction modulo 232, “~”denoted bitwise negation and “^” denotes bitwise exclusive-or. In Tornado, “X” is used to denote the word that needs to be modified, “Y” is the next word in the block that will affect the modification of “X” -note the block is considered to be circular, i.e. the next to “X (7)” is “X (0)”- and “K” is the part of the key that will be consumed in the modification process. So, each M function consists of 3 basic steps:

1. Key dependent operation that makes the key directly affects the input.
2. Data dependent operation assures that the arrangement of the words and their values affects the encryption of the block.
3. Fixed left rotation, that offers little confusion and linear diffusion inside the 32-bit word.

Table 1 The implementation of the M-functions

Function	N	Op1	Op2	RV
M1	0	+	-	3
M2	0	+	^	5
M3	0	-	+	6
M4	0	-	^	7
M5	0	^	+	9
M6	0	^	-	10
M7	1	+	^	11
M8	1	+	-	13
M9	1	^	+	15
M10	1	-	^	18

3. KEY SCHEDULING

Tornado algorithm uses 10 x 256-bit keys, which are delivered form the user's input key. Key scheduling algorithm uses some of the cryptographic core of the encryption algorithm to improve its security that was inspired by the famous Twofish algorithm [6]. Here, the present key setup process is faster than that used in Twofish because the later uses its encryption algorithm in the setup process while the former uses the M-functions (cryptographic core of Tornado) in the same process. The key setup consists of 6 stages as shown in figure (3.1):

The details of each stage in key setup are presented:

1. Padding and modifying the user input key "K" to 256-bits variable called "TheKey".

```
Function PaddingAndModefiying(input : K[])
        //The following code is in byte operations
        for  j=0 to 31 do
                TheKey[j]= K[j%Len]
                TheKey[j]=((TheKey[j%Len]+
                TheKey[j])+j)xor K[j%Len]
```

End for

//The following code is in 32-bit operations

for j=1 to 8

for i=0 to 7

TheKey [i]=M"j"(TheKey [i], TheKey [(i+1)%8)], TheKey [(i+2)%8])

Output(TheKey)

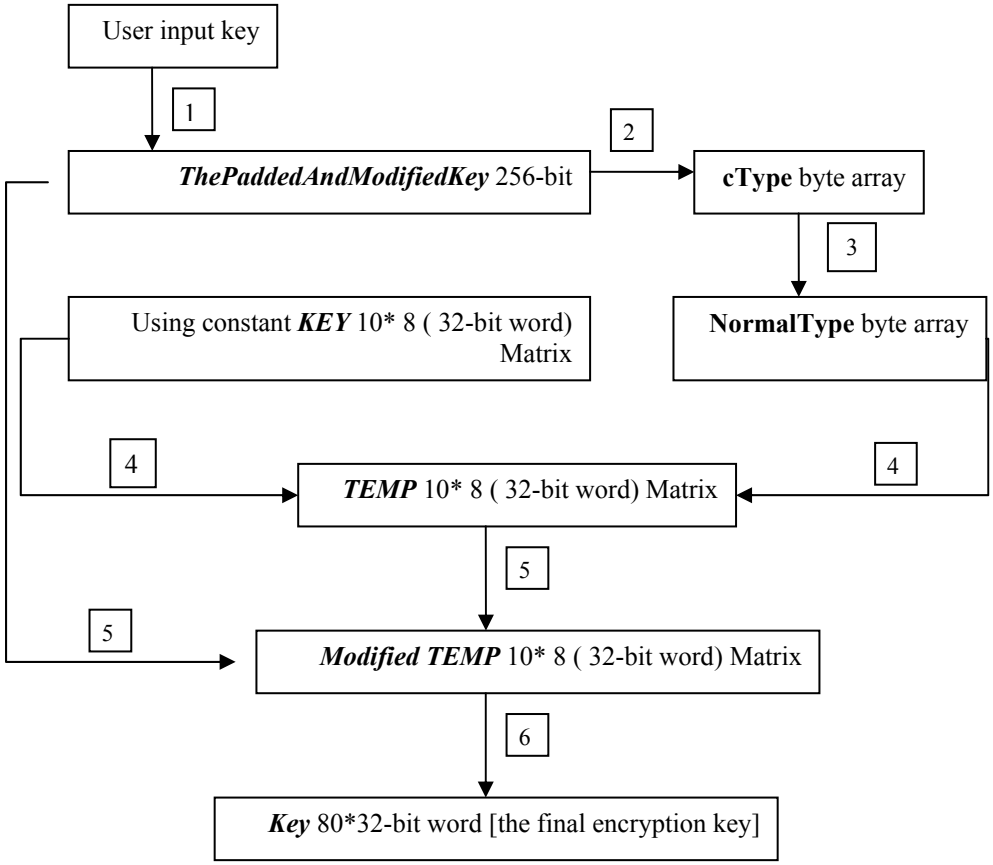

Figure (3.1) : Key scheduling stages

2. Generate the 32-bit integer (iType), form the padded 256-bit "TheKey", using fast and non-linear one-way hashing function that uses the M_Functions as its building blocks, then converting the "iType" to "cType" which is a 10 bytes array that represents the decimal digits of the "iType" (e.g. if "iType"=1324567890, then "cType"="1324567890").

```
Function Generate_cType(input: TheKey[])
        Value=M10(K[0],K[7],K[6]);
        Value=M1(K[1],K[2],Value);
        Value=M6(Value,K[5],K[3]);
        Value=M7(K[4],Value,K[0]);
        iType=Value;
        cType=ConvetWord32ToBytearray(iType)
```

3. Generate the "NormalType" byte array from "cType", which contains all the digits form '0'-'9', depending on "cType", the "NormalType" array is filled with unrepeated digits from left to right, then it is padded with all the unused digits in the "cType" array, in an ascending order.
4. Construct the "TEMP" 10x8 (32-bit word) matrix, using a constant "KEY" 10x8 (32-bit word) matrix and the "NormalType". The "KEY" matrix is a pre-computed constant matrix with random values, the values of the KEY matrix can be found in "Appendix A".

```
Function Construct_Temp_matrix(input: NormalType[])
        for i=0 to 9
        for j=0 to 7
            Temp[i][j]=KEY[NormalType [i]-'0'][j]
        for i=1 to 9
        for j=0 to 7
            Temp[i][j]+=Temp[i-1][j]
        for j=0 to 7
            Temp[0][j]+=Temp[9][j]
```

Note, the "NormalType" array is used to arrange the "TEMP" matrix, there are 10! different combinations, which depends on the input key.

Then the accumulating addition modulo 232 is performed, to change all the values of the Initial "KEY" matrix. Here, the addition is performed in a row circular fashion.

5. Modify the "TEMP" matrix using the M-functions (encryption core of the algorithm). In this stage, each row of the modified "TEMP" matrix is encrypted using different M-functions and the padded key "TheKey". Each row is encrypted in a circular fashion of data dependent encryption (i.e. encrypt 2nd element using the 3rd element … then finally the 1st element is used to encrypt the last element). The EncryptRow function, encrypts the input row "TEMP[i]", using the M-function number i+1 "2nd parameter", using the "TheKey" as a key.
   Function Modify_Temp_matrix(input: NormalType[])

   ```
   for i=0 to 9
       EncryptRow(TEMP[i],i+1, TheKey)
   end for
   ```

6. Copy the "TEMP" matrix to one dimensional Expanded Key array "Key" (80x32-bit word), which is used in the encryption process.

4. THE ENCRYPTION ALGORITHM

Tornado block cipher has both 256-bit block and key size. The algorithm seems to be impressive because it has 2^{32} different ways to operate. Consequently, blind search over the key space is enforced which is computationally infeasible. Tornado consists of 10 rounds which are sometimes referred to as Storm-function; the sequence of applying the Storm-Functions depend on the users input key that makes Tornado dynamic key dependent algorithm (can operate in 2^{32} different ways).

4.1. Storm-Functions

Storm-Functions are considered to be the cryptographic core of Tornado. Each Storm-Function consists of 8 M-Functions with different orders that operate on the 8x32-bit words, followed by rearrangement of the order of the input array. The order of applying the M-functions in each Storm-function is shown in table (3). M-Functions are responsible for performing the confusion process. The order of rearranging the input array in each Storm-Function is shown in table (4). Array rearrangement is responsible for performing the diffusion process. Figure (4.1) shows the

graphical representation of a sample Storm-Function the StormFunction0. Figure (4.2) shows the graphical representation of applying M-Functions in StormFunction0.

Table (3): The order of applying M-functions in Storm-Functions

StormFunction number	The order of applying M-functions							
0	1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8	9
2	3	4	5	6	7	8	9	10
3	4	5	6	7	8	9	10	1
4	5	6	7	8	9	10	1	2
5	6	7	8	9	10	1	2	3
6	7	8	9	10	1	2	3	4
7	8	9	10	1	2	3	4	5
8	9	10	1	2	3	4	5	6
9	10	1	3	3	4	5	6	7

Table (4): The order of array rearrangements in Storm-functions

StormFunction number	The order of rearrangement of X array							
0	0	3	5	1	7	4	2	6
1	0	5	3	7	1	4	6	2
2	0	2	7	5	3	1	6	4
3	0	4	2	7	5	1	3	6
4	0	5	7	2	4	6	1	3
5	5	0	7	3	1	6	4	2
6	4	0	2	5	7	3	1	6
7	2	5	0	3	7	1	4	6
8	7	2	0	4	6	1	3	5
9	2	7	5	0	3	6	1	4

The order of applying the M-functions in each Storm-Function is regular; each storm function begins with the M-function having the same number plus one. This is followed by the next seven functions in a circular manner where M1 follows M10. The order of the array rearrangement found in table (4) is much more complicated, because is in each row the next index is at least three positions away from its preceding one and each row contains at most two indices in the same position.

4.2. Cryptographic Core

The input (256-bit) is divided to 8x32-bit words and the encryption is applied as follows:

```
Function TornadoEncrypt(input:In[],cType[],Key[])
        X=In
            for i=0 to 9
                C=cType[i]-'0'
                StormFunctionC(X,Key)
                Key+=8
            end for
        Output(X)
```

Depending on the current digit in the "cType" array "C", the corresponding Storm-Function is applied to the array "X", consuming 8x32-bit form input "Key". (i.e. if cType[i]=='0', StormFunction0(X,Key) is applied). Hence, Tornado is a dynamic key dependent algorithm which can operates in 232 different ways and enforces blind search over the key space that is computationally infeasible.

4.3. Decryption Process

Decryption process uses dStorm-Functions to neutralize the effect of the Storm-Functions used in the encryption process. It applies dStorm-functions in reverse order of applying Storm-Functions in the encryption process. The input (256-bit) is divided to 8x32-bit words and the decryption is applied as follows:

```
Function TornadoDecrypt(input:In[],cType[],Key[])
        //Note in this function the array key is accessed from
        // its end
        X=In
```

```
for i=9 to 0
        C=cType[i]-'0'
        dStormFunctionC(X,Key)
        Key-=8
end for
Output(X)
```

Depending on the current digit in the "cType" array "C", the corresponding dStorm-function is applied to the array "X", consuming 8x32-bit form input "Key". (i.e. if cType[i]=='0', dStormFunction0(X,Key) is applied to remove the effect of Storm-Function0(X,Key)). In dStorm-Functions first the reverse array rearrangement (of the corresponding Storm-Function) is applied to the input, then the reverse of M-functions in reverse order of that used in encryption is applied.

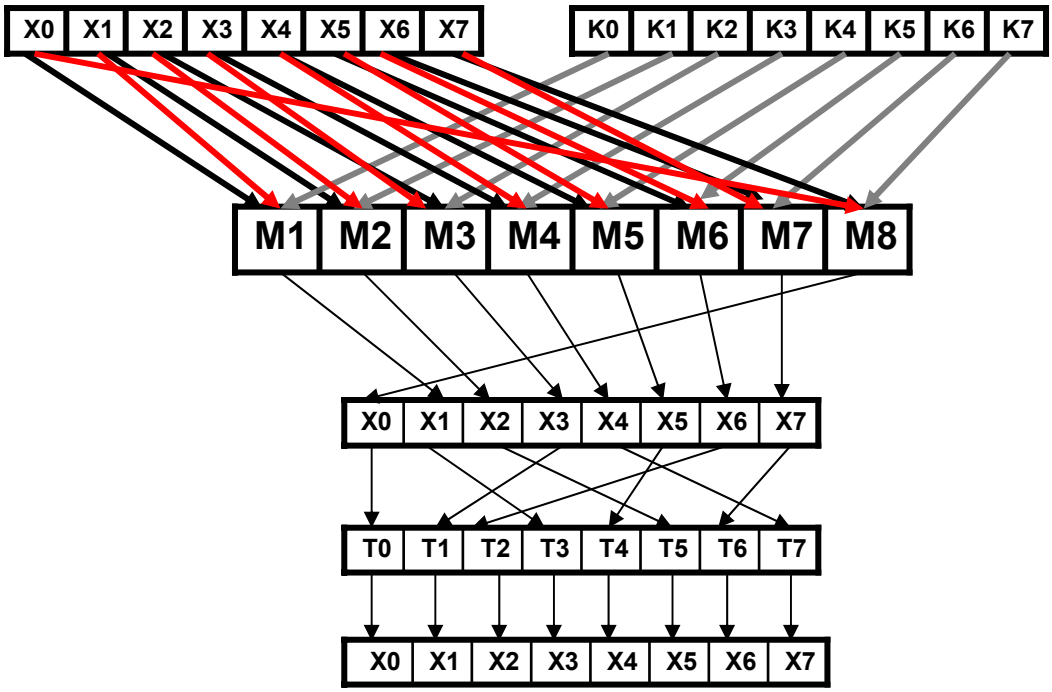

Figure (4.1): The graphical representation of the StormFunction0

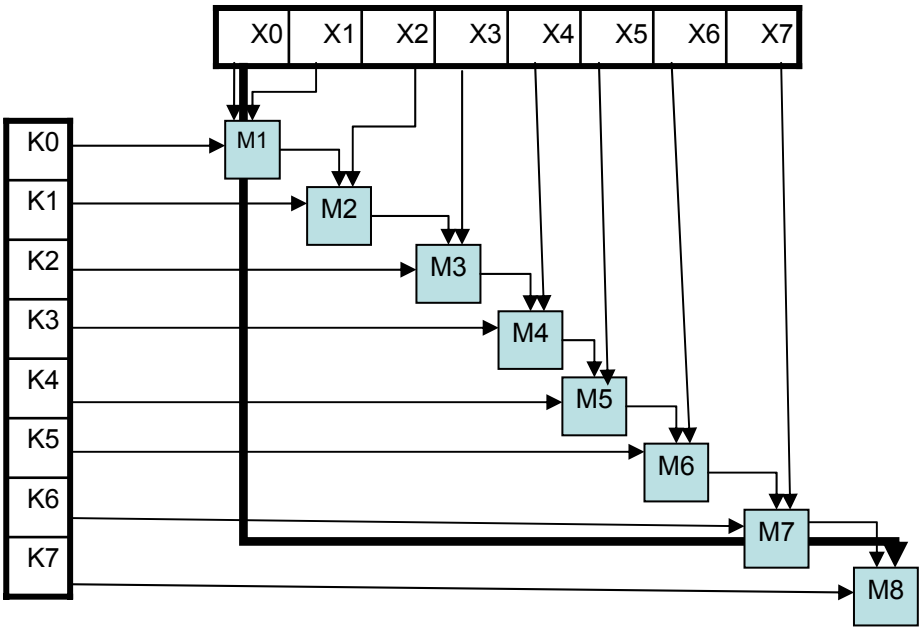

Figure (4.2): The graphical representation of applying M-Functions in StormFunction0

5. SECURITY OF TORNADO

The dynamic sequence of applying Storm-Functions, which is key dependent, is the backbone of the security of Tornado. This nature adds a lot of effort just to know which sequence was used [2^{32} different sequences can be used] , this is to enforce exhaustive-search over the key space which is 2256 different trail, which is computationally infeasible.

In addition of the strong key scheduling associated with Tornado, that uses the encryption building blocks of the algorithm. In key scheduling process chooses one initial matrix among 10! different formulations, that increases the complexity of sub-key attacks. Then the user's input key after modification is used to encrypt the initial matrix, which eliminates sub-key attacks, and enforces again exhaustive-search over the key space. Here we present some possible attacks on Tornado.

5.1. Dictionary Attacks

As the block size is 128 bits, the dictionary attack will require 2^{128} different plain-texts to allow the attacker to encrypt or decrypt arbitrary

messages under an unknown key. This attack applies to any block cipher with 128-bit blocks regardless of its design [3]. In the present case Tornado has 256-bit block size. So, a dictionary attack will requires 2^{256} different plain-texts.

5.2. Modes of Operation

After encrypting about 2^{64} plaintext blocks in the CBC or CFB mode, one can expect to find two equal cipher text blocks. This enables an attacker to compute the exclusive-or of the two corresponding plaintext blocks. With progressively more plaintext blocks, plaintext relationships can be discovered with progressively higher probability. In addition, when the algorithm is used in feed forward mode as a hash function, a collision can be found with an effort somewhat more than 2^{64}. This is for 128-bit block size, in Tornado this value will be increased to about 2^{128}.

5.3. Key-Collision Attacks

For key size k, key collision attacks can be used to forge messages with complexity only $2^{k/2}$. Thus, the complexity of forging messages under 256-bit keys is only 2^{128}. This attack applies to any deterministic block cipher, and depends only on its key size, regardless of its design.

5.4. Linear Analysis

In linear analysis one tries to find a subset of the bit positions in the plaintext, ciphertext and expanded keys, so that for a uniformly chosen plaintext and expanded key, the probability that the sum of the bits in these positions is equal to zero modulo 2, will be bounded away from 1/2. Such a subset is called a linear approximation of the cipher, and the difference between the obtained probability and 1/2 is called the bias of the approximation. In general, the goal of linear analysis is to find approximations with large bias, since an approximation with bias _ typically corresponds to an attack with work-load and data-complexity of about (1/x) [10].

The design of the M-functions assures that there are no operations cancel each other when applying the same or different M-functions on the same data. In addition, to the non-linear behavior of the cipher it-self (as it works in 2^{32} different ways). This attack seems to be inapplicable, further investigation on this attack is needed to be done.

6. PERFORMANCE OF TORNADO

Extensive testing has been performed to measure the speed of the different algorithms which are DES[12], Triple DES[13], IDEA[14], BLOWFISH, TWOFISH[6], SEPERENT[3], RIJNDAEL[4], MARS[1], RC6[11] and TORNADO, on an AMD Athlon 700 MHz 512KB cache PC-Platform. Table (6) shows the result of the test which is represented simultaneously by figure (6.1).

Table(6): Performance of different algorithms

Algorithm Name	Mega bytes / Sec
DES-EDE 3	3.91
IDEA	10.8
DES	11.1
SEPERENT	11.92
RIJNDAEL (256-BIT)	14.02
BLOWFISH	14.53
TWOFISH	20.46
MARS	25.97
RC6	30.16
TORNADO	35.539

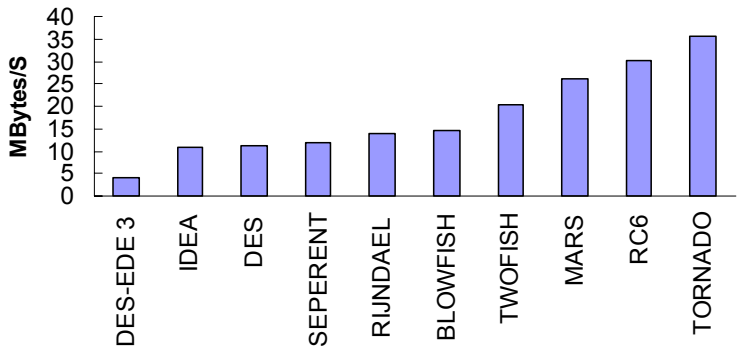

Figure (6.1) : Benchmark of the speed of different algorithms

Figure (6.1) shows that the present algorithm (Tornado) is the fastest cipher over the range of the chosen famous and powerful algorithms, including DES and Rijndael which are the old standard and the newest one. Tornado is faster than RC6 by a factor of 117.83 %, faster than Rijndael by a factor of 253.48 %, faster than DES by a factor of 320.17 %, and faster of Triple DES by a factor of 908.92 %, the rest of the speedup factors can be directly obtained from table (6).

7. CONCLUSIONS AND FUTURE WORK

Tornado is a fast, novel and secure cipher that is characterized by its dynamic nature, as it works in 232 different ways, which adds much strength to it. Also, this dynamic nature enforces exhaustive-search over the key space. The present algorithm has a powerful carefully designed key scheduling. Tornado shows high throughput as it is faster than, DES with factor of 320.17 %, Triple DES with factor of 908.92 %, and Rijndael (AES) with factor of 253.48 %. Yet we believe that Tornado is more secure than AES. More work is needed to be done in studying the algorithm security.

8. REFERENCES

[1] Burwick, C., Coppersmith, D., D'Avignon, E., Gennaro, R., Halevi, S., Jutla, C., Matyasm M., O'Connor, L., Peyravian, M., Safford, D .and Zunic, N.(1998),"MARS - a candidate cipher for AES", NIST AES Proposal.

[2] RSA Data Security Inc., www.rsa.com.

[3] Anderson, R., Biham, E. and Knudsen, L. (1998), "Serpent: A Proposal for the Advanced Encryption Standard", NIST AES Proposal.

[4] Daemen, J., and Rijmen, V.(1998), "AES Proposal: Rijndael", NIST AES Proposal.

[5] Nechvatal, J., Barker, E., Bassham, L., Burr. W., Dworkin, M., Foti, J. and Roback E.(2000), "Report on the Development of the Advanced Encryption Standard (AES)".

[6] Schneier, B.(1998), "Twofish: A 128-bit block cipher", Technical report, Counterpane Systems, Minneapolis, USA.

[7] Knudsen L. (1994), "Block Ciphers - Analysis, Design and Applications", Ph.D. Thesis, Arhus University, Denmark.

[8] Oorschot, P., Wiener, M.(1994), "Parallel Collision Search with Application to Hash Functions and Discrete Logarithms", in Proceedings of the 2nd ACM Conference on Computer and Communications Security (ACM), pp 210-218.

[9] Biham, E.(1996), "How to Forge DES-Encrypted Messages in 228 Steps", Technical Report CS884, Technion.

[10] Matsui, M.(1994), "Linear cryptanalysis method for DES cipher". Advances in Cryptology, EURO-CRYPT 93, Lecture Notes in Computer Science, vol. 765, T. Helleseth ed., Springer-Verlag, pages 386–397.

[11] Contini, S., Rivest, R., Robshaw ,M., and Yin,Y.(1998) "The Security of the RC6 Block Cipher". Version 1.0. August 20. Available at http://www.rsa.com/rsalabs/aes.

[12] Biham,E.(1997), "A Fast New DES Implementation in Software", in Fast Software Encryption , 4th International Workshop, Springer LNCS v 1267,pp 260-271.

[13] Davies,D., Murphy,S.(1995),"Pairs and Triplets of DES S Boxes", in Journal of Cryptology v 8 no , pp 1-25.

[14] Kelsey, J., Schneier,B., Wagner,D.(1996), "Key-Schedule Cryptanalysis of IDEA, GDES, GOST, SAFER and Triple-DES", in Advances in Cryptology , Springer LNCS v 1109.pp 237{251}.

APPENDIX A

The values of the initial constant matrix **KEY.**

KEY[10][8]=

{{0xc74fb678,0x141582a8,0xca65895c,0xfcdca86e,0xd1790a28,0xb9a3c9ef,0xd4d88a7f,0xd4d893dd},

{0x606f682a,0xc8d6ec52,0xfd720f68,0xa6f55300,0x615c68ed,0x2b27521b,0x80e28fd5,0x6587f82e},

{0xf5353543,0x13da4d39,0xb606c50d,0x8d67bde7,0x2b6cd371,0x2ecb6cc6,0x8fc38598,0x4b00e94b},

{0x04351870,0x8de9196e,0x87ae1c90,0x7d3488fa,0x8e454749,0x641246a5,0x81f2726f,0x2a9cbf75},

{0x2c332b3e,0xcac5950b,0xab6bcf4a,0xc5807f20,0x5adcf151,0x144fb49b,0xd6c3e5c4,0xb9ee490d},

{0x9ccb26fb,0x1b780052,0xa35a1e79,0x365ebfc4,0x33226ece,0x244f11e6,0x4b3260dd,0x393dc369},

{0x12601dca,0xadb935fa,0x15eb17f8,0xd473499b,0x68008c8c,0xc72cdc61,0x6011963b,0x115e7f6a},

{0x4c2c7cbe,0x61f49a05,0x4460e0db,0x7fcda920,0x6b987604,0x3134f025,0xe09cf81e,0x1099add0},

{0x4ba6a5bf,0xb4a86fe8,0xcf5caa99,0xb22ea40a,0x6128aedb,0xe0fca4df,0x2e2ce912,0xa56daa4e},

{0x63d4fcaf,0xff0179ce,0xd725c532,0x98ca2735,0x7a6323b0,0xf82c3342,0x658e234b,0x85fa76d1}}