Optimization of IDEA Key-Schedule Algorithm for Safe Use in Cloud

Masoud Rafighi#1andNaghmeMoatazedi*2
#Faculty of Information, communications and Security Technologies Complex,
Malek-Ashtar University of Technology, Tehran, Iran
1Rafighi@mut.ac.ir
*Department of Information Technology Engineering, Taali University, Qom, Iran
2Nmoatazedi93@gmail.com

***Abstract*—Nowadays, thanks to its significant advantages, cloud computing is one of the most attractive modern technologies which is mostly used for data storage. Cloud storage servers are targeted by various attacks which are neutralized with various methods including encryption which is the most used method for protecting information. IDEA encryption algorithm introduced in 1990 is a public key algorithm which only uses cyclic changing of a secret key in its own key schedule to generate sub-keysand therefore is very weak. In this paper, we tried to improve the security of IDEA key-schedule by utilizing the strengths of Twofish and Blowfish key-schedule algorithmsand then evaluated the new algorithm, named TB-IDEA, using CT1software.**

Keyword-Security, Encryption, IDEA, Key schedule algorithm, Cloud computing.

I. INTRODUCTION

Cloud is a server on a network infrastructure with unknown location that provides resources such as hardware, operating system, database, applications etc. for users in the form of services [1]. Cloud computing refers to applications and hardware that are provided through cloud data centers [2].

One of the main uses of cloud computing is storage of information and accessingthe information from anywhere in the world. However, since users receive services through the Internet, security remains an important challenge in cloud computing and for this reason, companies and people are not sure to put their sensitive information on cloud servers. Various attacks such as DoS, Account Cracking, MITM, Side Channel, etc. are perpetrated against the data stored on cloud servers which are tried to be neutralized through different ways including identity and access control, firewall, IDPS and DLP, using a third-party auditor, virtualization of security in the runtime, encryption and so on [3].

The most common method of protecting information is encryption. Encryption can be expressed as the art of securely composing and transferring messages in a hidden or undetectable form in an unsafe environment such as the Internet by means of encrypting plaintext with the help of different encryption algorithms [4].Early cryptography was only concerned with privacy; but since 1974the security objectives have developed. Presently the new generation’s outlook towards privacy is different which challenges cryptography. A sensitive issue is the possibility of replacement of cryptology with “cryptonomy”. It is difficult to predict the future, but future security objectives may require new tools that will keep cryptography researchers busy [5].

II. A REVIEW OF ALGORITHMS

The algorithms used in this paper are reviewed as follows.

A. IDEA Algorithm

In 1990, James MassayandXuejia Lai offered PES algorithm which was revised and modified by them in 1991 and named IPES and then it was commercialized and named IDEA [6]. IDEA encryption algorithm was a substitute for DES algorithm. IDEA is a symmetric block-based algorithm which important security features are Confusion and Diffusion. This algorithm gets a 64-bit input text and generates a 64-bit output text by 128-bit keys under 8 similar rounds and an output conversion. IDEA enjoys high speed of encryption/decryption at hardware and software levels [7].

IDEA key-schedule algorithm uses 52 sub-keys (k1, k2, … , k52); the 128-bit key is divided into eight 16-bit sub-keys. These 8 keys form the first set, 6 keys of which are used for the first round and 2 keys are used at the start of the second round [6]. Then, a 25-bit left rotation is performed and once more, the key is divided into 8 sub-keys, the first 4 keys are used for the second round and the next 4 keys are used for the third round. This procedure continues until the rounds are completed.

B. *Twofish Algorithm*

Twofish algorithm introduced in 1998 by Bruce Schneier is an 128-bit blockencryption method that works with different key lengthsup to 256-bit. This algorithm includes 16 Feistel network rounds with a bidirectional F function and a key schedule designed with a high accuracywith only a 1-bit rotation as its non-Feistelelement. Design of rounds and key schedule brings a balance between speed, size of software, key set time and memory [8]. This algorithm is very suitable for big microprocessors and smart cards and enjoys a high level of security and any attack in its best scenario can only break 5 rounds of it.

C. *Blowfish Algorithm*

BF, a symmetric block algorithm designed in 1993 by Bruce Schneier, has a fixed block size of 64-bit and a variable key length between 32 to 448bits. This algorithm uses Feistel network and the P-Boxes, S-Boxes and xor structures [9].

III. PROBLEM STATEMENT

We have a cloud storage server and a database containing user data on it. As already mentioned, this remote server is exposed to various attacks that threatenthe confidentiality, integrity, availability, etc.of the user data. An attacker can access and misuse the data in different ways. Also, a malicious cloud service provider can easily access the data. Thus, the best policy is encrypting the data before sending them to cloud storage servers.

As already mentioned in Section 2.1, IDEA encryption algorithm is a public key algorithm in which the rotation of key and its division into 16-bit keys are continued until all sub-keys are generated for each of the eight rounds. The idea of using only cyclic change of secret key for generating sub-keys is the reason that IDEA key-schedule algorithm is very weak and therefore the keys of this algorithm are categorized in the class of weak keys.

In the research conducted by Shazia Afzal etal [10], various tests have been administered for evaluating the independence of sub-keys of various algorithms at sub-key, byte and bit levels. The result of frequency test of Twofish scheduler is higher than 90% which shows the balance of ‘0’ and ‘1’ in the generated sequence. The results show that the generated sub-keys at sub-key level are independent and separate. However, the result of frequency test for IDEA scheduler algorithm shows the imbalance of ‘0’ and ‘1’ in the sequence; so, IDEA scheduler doesn’t meet the independence criteria at sub-keys level. Also, results of the test of independence of sub-keys at byte level show that the bytes of sub-keys generated by Twofishare statistically independent and separate while the bytes of IDEA sub-keys are strongly related. Twofish uses strong components such as MDS and S-BOX in its key-schedule algorithm and has a strong key schedule. The weak results of IDEA in all the tests show that this algorithm has a high level of sub-key dependenceat bit level.

Due to the simplicity of IDEA key-schedule algorithm, a chosen-key differential attack can be done on 3 rounds of it. There is also a Chosen-Key Ciphertext-only Timing attack on 8 rounds of IDEA that needs $5*2^{17}$ related-key queries and each of them needs to encode 2^{20} random unknown plain text blocks [11]. Therefore, in order to secure IDEA algorithm, we need a solution to improve this key-schedule algorithm.

IV. PROPOSED ALGORITHM

Encryption can be traced back in the early works of Shannon in the late 40s and early 50s and since then it has undergone great developments. When using encryption in different systems we should be aware of the features, strengthsand weaknesses of algorithms. Although encryption has progressed significantly, the encryption algorithms are constantly attacked and these attacks have undermined their securityand make us think of enhancement of their security or even designing new algorithms.

We are going to use the strengths of Twofish and Blowfish key-schedule algorithmsto produce IDEA sub-keys in our proposed TB-IDEA algorithm with the aim to boost the security of IDEA. P-array of blowfish key-schedule algorithmand MDS matrices and Pseudo-Hadamard Transform (PHT) from Twofishkey-schedule algorithm which are considered the merits of these algorithms have been used. As shown in Table I, Blowfish P-array has 18,32-bit boxes.

TABLE I. P-Array of Blowfish Algorithm [12]

P1=0x243f6a88	P2=0x85a308d3	P3=0x13198a2e
P4=0x03707344	P5=0xa4093822	P6=0x299f31d0
P7=0x082efa98	P8=0xec4e6c89	P9=0x452821e6
P10=0x38d01377	P11=0xbe5466cf	P12=0x34e90c6c
P13=0xc0ac29b7	P14=0xc97c50dd	P15=0x3f84d5b5
P16=0xb5470917	P17=0x9216d5d9	P18=0x8979fb1b

First 128-bit user input key is divided into 32-bit blocks. Then, as shown in Fig. 1, the first, second, third and fourth 32-bit blocks xor with P1, P2, P3 and P4 respectively. Again, from the beginning, the first, second, etc. 32-bit blocks xor with P5, P6, etc. respectively until all P-arrays xor with key bits. Then, the resulted 32-bit blocks xor with P-arrays and key blocks xor together and a 32-bit output is obtained.

The main key is stored reversely and xor with the 32-bit output of previous step. Then, the resulted 128-bit output is divided into four 32-bit blocks. As shown in Fig. 1and Eq. (1), each of the 32-bit blocks is multiplied with MDS matrices separately [8].

$$\begin{bmatrix} z1 \\ z2 \\ z3 \\ z4 \end{bmatrix} = \begin{bmatrix} 01 & EF & 5B & 5B \\ 5B & EF & EF & 01 \\ EF & 5B & 01 & EF \\ EF & 01 & EF & 5B \end{bmatrix} \cdot \begin{bmatrix} y0 \\ y1 \\ y2 \\ y3 \end{bmatrix} \tag{1}$$

The output of the first 32-bit is rotated 8 bits to the left and is entered into PHT with the output of the second 32-bit block and are combined together. Again, the output of the second 32-bit block of PHT is rotated 9 bits to the left. This procedure is repeated for the third and fourth 32-bit blocks too.

By virtue of Eq. (2) and Eq. (3), PHT is a simple operation for combining the two outputs of the previous stage [8]. The inputs are ‘a’ and ‘b’ (in order to optimize computing, we have made some changes in the formulas):

$$a' = (a + b \; mod \; 2^{16}) * 1023 \tag{2}$$

$$b' = (a + 2b \; mod \; 2^{16}) * 1023 \tag{3}$$

If the lengths of ‘a’ and ‘b’ are less than 32 bits, padding operation is done: a ‘1’ is added to the end of bits and then ‘0’ bits are added until the desired length is achieved. Finally, four 32-bit blocks are formed. Each of these 32-bit outputs are divided into two 16-bit sections that are the sub-keys. In the first round, eight 16-bit sub-keys are generated. Then, the user’s input key is rotated 25 bits to the left and is used as the primary key to generate the next 8 sub-keys. All the above steps are repeated on the rotated key and the next 8 sub-keys are produced. All in all, the said steps are repeated 7 times to produce 52 sub-keys required for IDEA. Except for the first round in which the user’s key is directly entered into the algorithm, the remaining rounds use the key of previous stages and are rotated 25 bits to the left and undergo the process. Fig. 2 is an overview of TB-IDEA Scheduler.

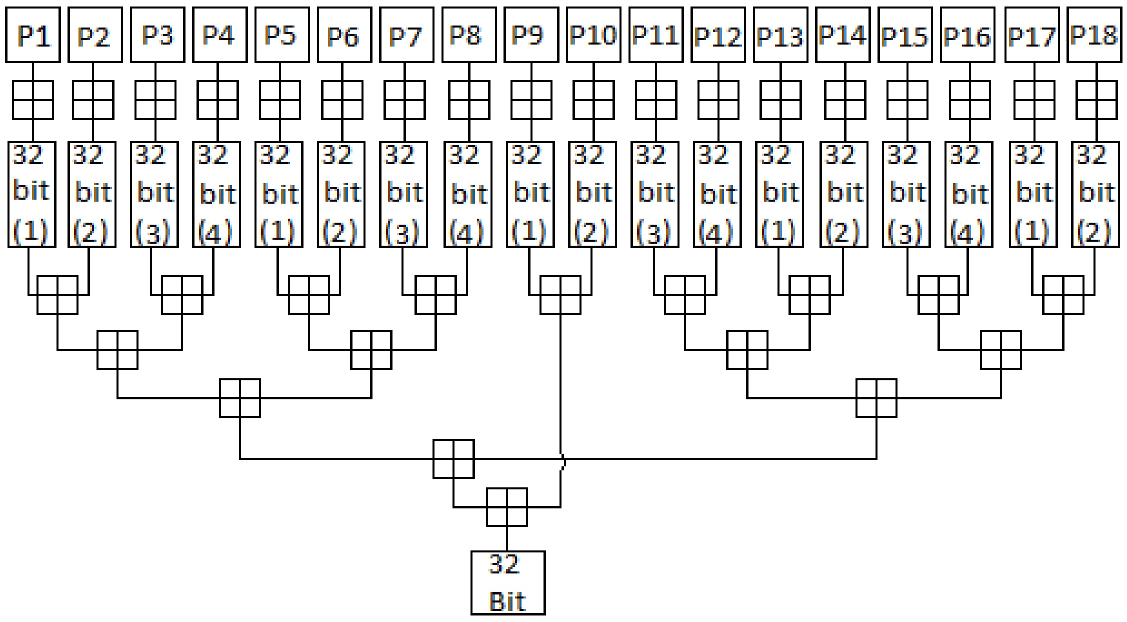


Fig.1.Xoring P-arrey and key blocks

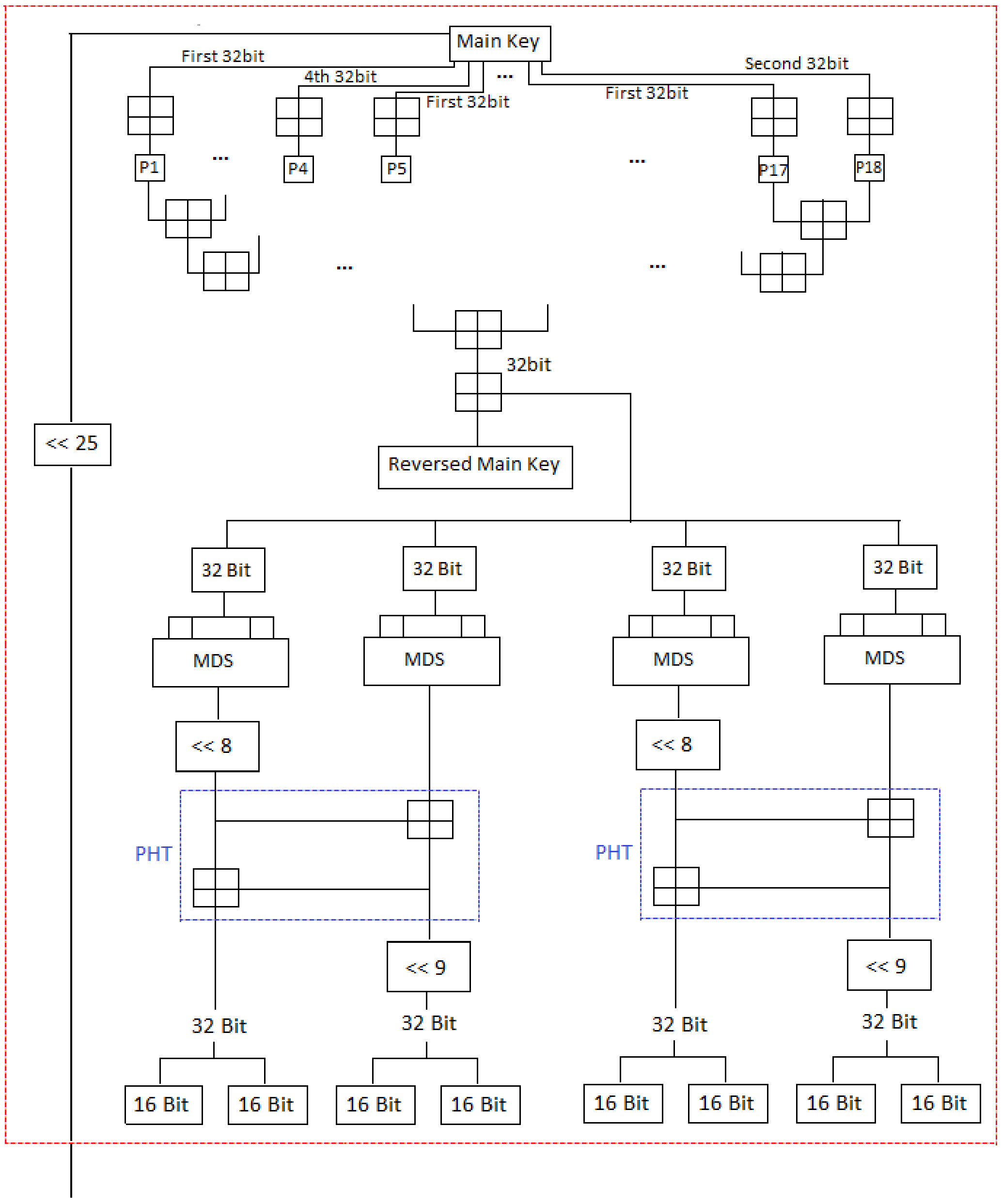


Fig. 2. Overview of TB-IDEA key-schedule

D. *TB-IDEA Key-Schedule Psuedo-Code*

The pseudo-code of TB-IDEA key-schedule is as follows:

```
// Define P-Array
p[1] = "0x243f6a88";            p[2] = "0x85a308d3";
p[3] = "0x13198a2e";            p[4] = "0x03707344";
p[5] = "0xa4093822";            p[6] = "0x299f31d0";
p[7] = "0x082efa98";            p[8] = "0xec4e6c89";
p[9] = "0x452821e6";            p[10] = "0x38d01377";
p[11] = "0xbe5466cf";           p[12] = "0x34e90c6c";
p[13] = "0xc0ac29b7";           p[14] = "0xc97c50dd";
p[15] = "0x3f84d5b5";           p[16] = "0xb5470917";
p[17] = "0x9216d5d9";           p[18] = "0x8979fb1b";
// Define MDS Matrices
MDS[1] = "01";         MDS[2] = "EF";         MDS[3] = "5B";
MDS[4] = "5B";         MDS[5] = "5B";         MDS[6] = "EF";
MDS[7] = "EF";         MDS[8] = "01";         MDS[9] = "EF";
MDS[10] = "5B";        MDS[11] = "01";        MDS[12] = "EF";
```

```
MDS[13]="EF";                    MDS[14] = "01";                    MDS[15] = "EF";
MDS[16] = "5B";
// Do All Steps for 7 rounds ---------------
#Region of Xoring P-array and 32bits of MainKey; Xoring output and Reversed MainKey
SubMainkey1 = First 32bit of MainKey;
SubMainkey2 = Second 32bit of MainKey;
SubMainkey3 = Third 32bit of MainKey;
SubMainkey4 = Fourth 32bit of MainKey;
//step1
    a1 = p[1] XOR SubMainKey1;
    b1 = p[2] XOR SubMainKey2;
    c1 = p[3] XOR SubMainKey3;
    d1 = p[4] XOR SubMainKey4;
    a2 = p[5] XOR SubMainKey1;
    b2 = p[6] XOR SubMainKey2;
    c2 = p[7] XOR SubMainKey3;
    d2 = p[8] XOR SubMainKey4;
    a3 = p[9] XOR SubMainKey1;
    b3 = p[10] XOR SubMainKey2;
    c3 = p[11] XOR SubMainKey3;
    d3 = p[12] XOR SubMainKey4;
    a4 = p[13] XOR SubMainKey1;
    b4 = p[14] XOR SubMainKey2;
    c4 = p[15] XOR SubMainKey3;
    d4 = p[16] XOR SubMainKey4;
    a5 = p[17] XOR SubMainKey1;
    b5 = p[18] XOR SubMainKey2;
//step2
    e1 = a1 XOR b1 ;
    e1 = c1 XOR d1 ;
    e1 = a2 XOR b2 ;
    e1 = c2 XOR d2 ;
    e1 = a3 XOR b3 ;
    e1 = c3 XOR d3 ;
    e1 = a4 XOR b4 ;
    e1 = c4 XOR d4 ;
    e1 = a5 XOR b5 ;
//step3
    f1 = e1 Xore2 ;
    f2 = e3 XOR e4 ;
    f3 = e6 XOR e7 ;
    f4 = e8 XOR e9 ;
//step4
    g1 = f1 XOR f2 ;
    g2 = f3 XOR f4 ;
//step5
    h1 = g1 XOR g2 ;
    i1 = h1 XOR e5 ;
// Xoring final 32 bit of P-Array and 32bits of mainkey
    m1= Reverse of Mainkey XOR i1 ;
    #end region
    #Region of MDS and PHT functions
// Produce four 32 bits from output of previous step(m1)
    Part1 = First 32bit of m1;
    Part2 = Second 32bit of m1;
```

```
        Part3 = Third 32bit of m1;
        Part4 = Fourth 32bit of m1;
// Operate on First & Second 32 Bits
// Divide First 32bits to 4 8bits
        Z1 = First 8bit of Part1;
        Z2 = Second 8bit of Part1;
        Z3 = Third 8bit of Part1;
        Z4 = Fourth 8bit of Part1;
// Multiple 8bits and MDS matrix
        Y1 = (MDS1 * Z1) + (MDS2 * Z2) + (MDS3 * Z3) +
(MDS4 * Z4);
        Y2 = (MDS5 * Z1) + (MDS6 * Z2) + (MDS7 * Z3) +
(MDS8 * Z4);
Y3 = (MDS9 * Z1) + (MDS10 * Z2) + (MDS11 * Z3) +
(MDS12 * Z4);
        Y4 = (MDS13 * Z1) + (MDS14 * Z2) + (MDS15 * Z3) +
(MDS16 *Z4);
        Y = Y1Y1Y3Y4;
        8LeftShift = do 8 left shift on Y;
// Divide Second 32bits to 4 8bits
        z1 = First 8bit of Part2;
        z2 = Second 8bit of Part2;
        z3 = Third 8bit of Part2;
        z4 = Fourth 8bit of Part2;
// Multiple 8bits and MDS matrix
        y1 = (MDS1 * z1) + (MDS2 * z2) + (MDS3 * z3) +
(MDS4 * z4);
        y2 = (MDS5 * z1) + (MDS6 * z2) + (MDS7 * z3) +
(MDS8 * z4);
        y3 = (MDS9 * z1) + (MDS10 * z2) + (MDS11 * z3) +
(MDS12 * z4);
        y4 = (MDS13 * z1) + (MDS14 * z2) + (MDS15 * z3) +
(MDS16 * z4);
        y = y1y2y3y4;
// PHT Operation
        u1 = ((8LeftShift * Y) mod (2 ^ 16)) * 1023;
        u2 = ((8LeftShift * (2 * Y)) mod (2 ^ 16)) *
           1023;
// 1023 = 1111111111
// *Now do Padding, If the length of u1 and u2 is less than 32 bit:
        u3 = do 9 left shift on u2;
// Produce four 16bit SubKeys from First and Second 32bits of MainKey
        Key1 = First 16bit of u1;
        Key2 = Second 16bit of u1;
        Key3 = First 16bit of u3;
        Key4 = Second 16bit of u3;
// Do Previous MDS and PHT operation on third and fourh 32bit of MainKey too, then do  25 left shift on
main key
MainKey = do 25 left shift on MainKey of previous step ;
```

V. Evaluation Software

Cryptanalysis is done with two purposes: 1) reducing or removing the security of an encryption system and accessing information, and 2) evaluating an encryption algorithm and examining its strengths and weaknesses.

In this paper CrypTool1 software has been used for evaluation. CT1 is an open source Windows application for encryption and decryption which was originallydesigned as an internal business program for information security training and then has been developed to become an important open source project in encryption and IT security awareness. Some of the methods used in this software for cryptanalysis include:

E. *Entropy*

The ability to guess the value of a random variable is an important criterion for variable quality [13]. Confidentiality of key is defined as lack of information on the part of the adversary. The desirable type of entropy for encryption is min-entropywhich measures the degree of difficulty of guessing the information. In fact, researchers believe that the traditional measurement with Shannon entropy is not a suitable model to be used in encryption and min-entropy is a good alternative. Min-entropy is used in this article too. Entropy is measured based on bit/character: the higher entropy, the more will be the chance of even occurrence of all the 26 characters.

F. *Autocorrelation*

In correlation test a text is compared with other altered versions of the same text. Characters which matchin both texts are determined. In fact, it is a mathematical tool for finding repeating patterns.

G. *N-Gram*

In computational linguistics, an interrelated sequence of n elements of a given sequence of text is called an n-gram. According to the software, elements can be phonemes, syllables, letters or words. An n-gram with size of 1 is called as unigram, with size of 2 is called bigram and with size of 3 is called trigram. Larger sizes are pointed out as four-gram, five-gram and so on. In this test the frequency of the letters is also displayed. In a simple text which is long enough, depending on the language, each of the characters occurs with a certain frequency. Each of the characters of the plain text is assigned to one or more encrypted characters. To break a cryptograph, the frequency of characters in an encoded text is compared to the frequency of that character in the plain text.

H. *Poker*

This test examines uniform distribution of P-bit pattern in the entire sequence; that is, the number of times that the P-bit blocks are appeared in the entire sequence should be the same [10]. Accordingly, the sequence is divided into blocks with a length of m. Then, the probability of each of 2^m possible subsequences with the length of m available in the sequence is determined. Poker test determines whether all subsequences with the length of m occur evenly in the sequence or not.

In this test we have a factor called alpha importance degree with values of 0.01, 0.05 and 0.10. If its value is too high, the test may deny the sequence that actually has been generated by a random bit generator, and if its value is too low, there is the risk that the test accepts the sequence that has not been produced by a random bit generator. The maximum amount of the test is a statistical value that is dependent on the alpha importance degree and the test result is a statistical value generated by the test that is compared with the maximum amount of the test. K-tuple value is a group of K elements. K-tuple is used as a parameter for poker test. For example, tuple 3 provides 2^3=8 different vectors with the length of 3 each that are analyzed by the test individually.

For evaluation, we encrypted "New IDEA SubKeys" expression that is a 128-bit phrase as follows:

- First, we considered a 128-bit key as a primary key and generated two sets of sub-keys using IDEA key schedule and TB-IDEA key schedule.
- Next, we put the 128-bit plain text into binary form and divided it into two 64-bit parts.
- Then, we encrypted each of two 64-bit parts once using IDEA sub-keys and the next time using TB-IDEA sub-keys.
- Finally, we gave these two encrypted texts to the software in base64 and evaluated them.

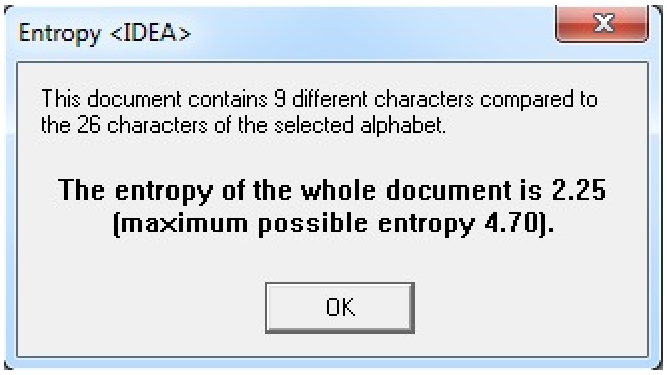
VI. EVALUATION OF IDEA CIPHER TEXT

Encrypted form of the phrase by IDEA scheduler in base64 is as follows:

77+9Z3rvv73vv73Yk2Pvv73vv73Llu+/vRDvv73vv70=

I. *Entropy Test*

The input text includes 9 characters and its entropy is 2.25 (Fig. 3). The maximum entropy for a text could be 4.70, so, the resulted entropy is 47% which shows low to middle difficulty of guessing the information.



Entropy <IDEA>

This document contains 9 different characters compared to the 26 characters of the selected alphabet.

The entropy of the whole document is 2.25 (maximum possible entropy 4.70).

Fig. 3. Entropy test result of IDEA cipher

J. *Autocorrelation Test*

The input block has been divided into 10 offsets and compared (Fig. 4). For example, in the first part, 10 characters of the two texts are similar which shows maximum correlation and we have the lowest correlation in the fourth offset. The less the correlation, the more will be the independence of ‘0’ and ‘1’ bits in the input text and the security.

K. *N-Gram Test*

Table II shows unigram, bigram, trigram and four-gram results on IDEA cryptograph which represent the number of repeats of a letter or group of letters in terms of repetition and percentage. For example, the highest repetition in unigram is for the letter ‘v’ with a percentage of 56.5217, that is, 13 repetitions.

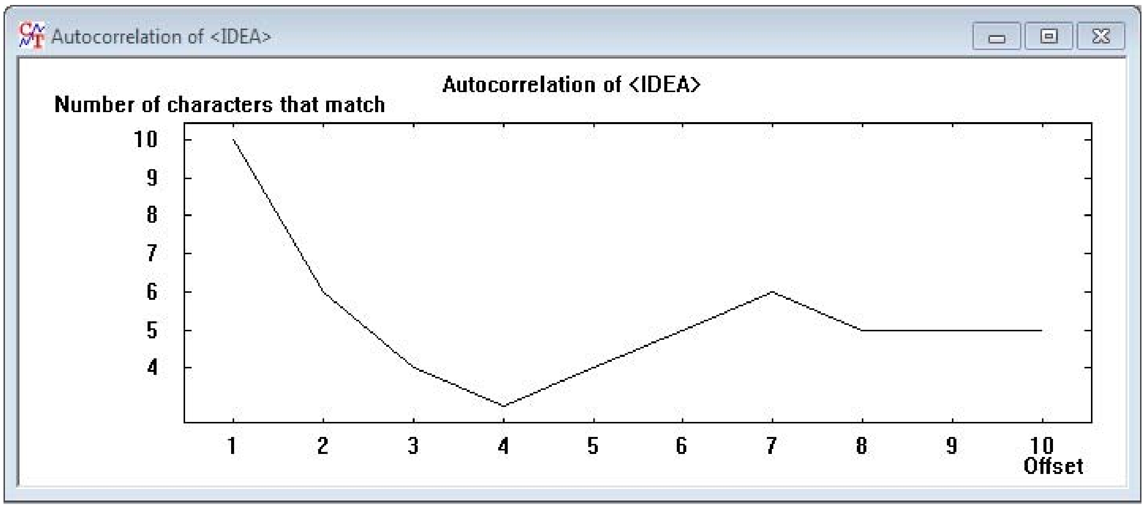


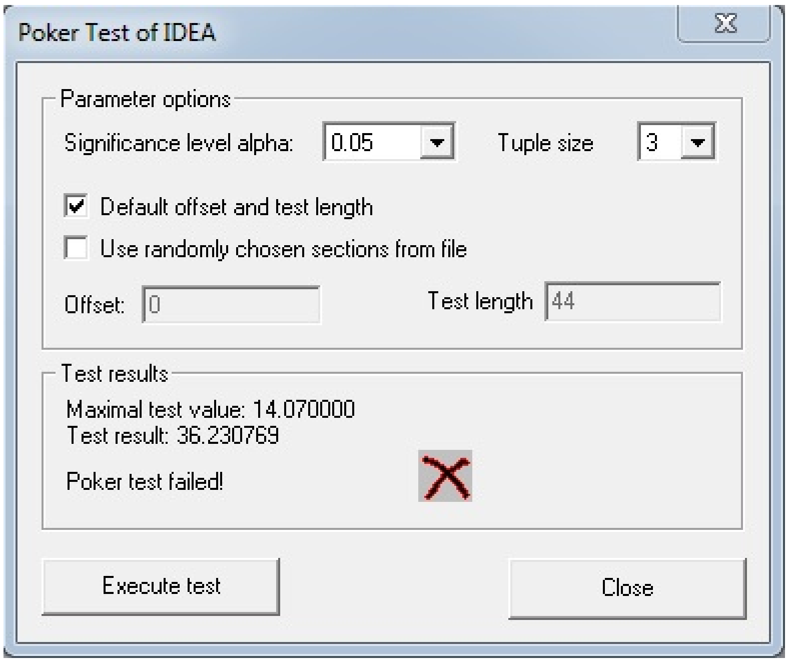
Fig. 4. Autocorrelation test result of IDEA cipher

TABLE II. N-Gram test result of IDEA cipher

Unigram				Bigram			
No.	Character Sequence	Frequency in %	Frequency	No.	Character Sequence	Frequency in %	Frequency
1	V	56.5217	13	1	VV	42.8571	6
2	L	8.6957	2	2	DV	7.1429	1
3	R	8.6957	2	3	LL	7.1429	1
4	D	4.3478	1	4	LU	7.1429	1
5	K	4.3478	1	5	PV	7.1429	1
6	P	4.3478	1	6	RD	7.1429	1
7	U	4.3478	1	7	RV	7.1429	1
8	Y	4.3478	1	8	VR	7.1429	1
9	Z	4.3478	1	9	YK	7.1429	1
TriGram				**FourGram**			
No.	Character Sequence	Frequency in %	Frequency	No.	Character Sequence	Frequency in %	Frequency
1	DVV	16.6667	1				
2	LLU	16.6667	1				
3	PVV	16.6667	1	1	RDVV	50.0000	1
4	RDV	16.6667	1	2	VRDV	50.0000	1
5	RVV	16.6667	1				
6	VRD	16.6667	1				

L. *Poker Test*

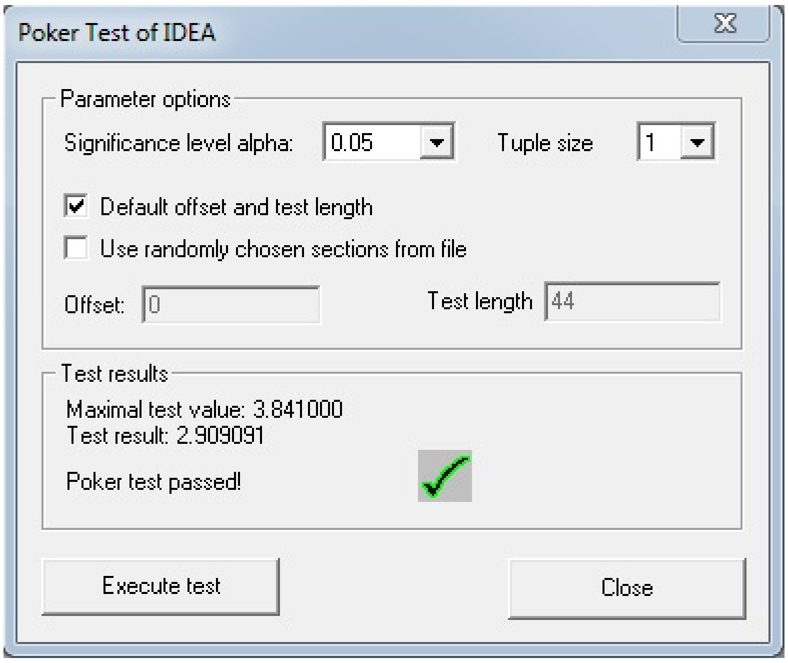
For IDEA we have chosen importance degree of 0.05 and tuple 3, and as Fig. 5 shows, the test failed. The change to tuple 1 resulted in test success (Fig. 6). The results indicate that the values have been selected correctly and subsequences occur evenly in the whole sequence with a probability of 75%.


Poker Test of IDEA

Parameter options
Significance level alpha: 0.05
Tuple size 3
[✓] Default offset and test length
[ ] Use randomly chosen sections from file
Offset: 0
Test length 44

Test results
Maximal test value: 14.070000
Test result: 36.230769
Poker test failed!

Fig. 5. Poker test result of IDEA cipher with tuple size 3


Poker Test of IDEA

Parameter options

Significance level alpha: 0.05 Tuple size 1

[✓] Default offset and test length

[ ] Use randomly chosen sections from file

Offset: 0 Test length 44

Test results

Maximal test value: 3.841000
Test result: 2.909091

Poker test passed! ✓

Execute test Close

Fig. 6. Poker test result of IDEA cipher with tuple size 1

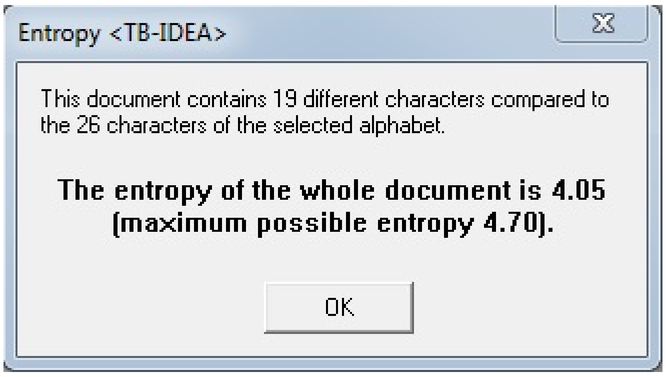
VII. Evaluation of TB-IDEA Cipher Text

The encrypted form of the phrase by TB-IDEA scheduler in base64 is as follows:

wonCvsKnLMOwwpfChsKGwrnDjMODbsKAw6rCviA=

M. Entropy Test

The input text contains 19 characters and has more characters than the IDEA cryptograph (Fig. 7) and its entropy is 4.05, that is, 86%, indicating that unlike IDEA, this algorithm enjoys a high degree of difficulty in guessing information.


Entropy <TB-IDEA>

This document contains 19 different characters compared to the 26 characters of the selected alphabet.

The entropy of the whole document is 4.05 (maximum possible entropy 4.70).

OK

Fig. 7. Entropy test result of TB-IDEA cipher

N. Autocorrelation Test

The input block is divided into 18 offset and compared (Fig. 8). For example, in the 12th part, 8 characters of the two texts match which is the maximum correlation. Offsets 2, 3, 7, 10, 13, 14, 17 and 18 have minimum correlations.

Comparing the result of two correlation test on encrypted texts, we realize that correlation between bits of the text encrypted with IDEA sub-keys is higher than that of the text encrypted with TB-IDEA sub-keys. In other words, the bits of IDEA cryptograph have a low level of independence while the bits of TB-IDEA cryptograph have a high level of independence.

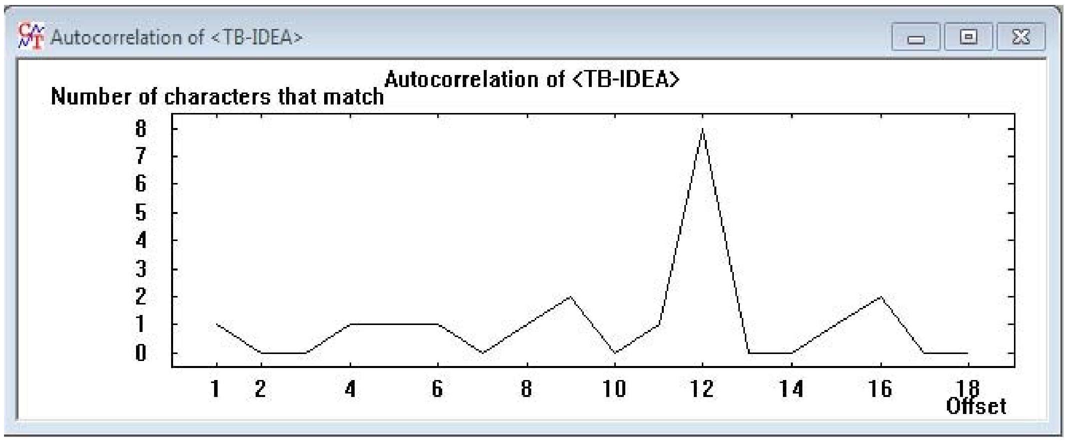


Fig. 8. Autocorrelation test result of TB-IDEA cipher

O. *N-Gram Test*

Table III shows unigram, bigram, trigram and four-gram results on TB-IDEA algorithm. For example, the highest unigram is related to the letter ‘w’ with a percentage of 13.1579, that is 5 repetitions. As can be seen, the TB-IDEA level of has more characters than the IDEA has. Also the percentage and frequency of the cryptograph of this algorithm in comparison with the IDEA cryptograph are low which shows improvement in encryption.

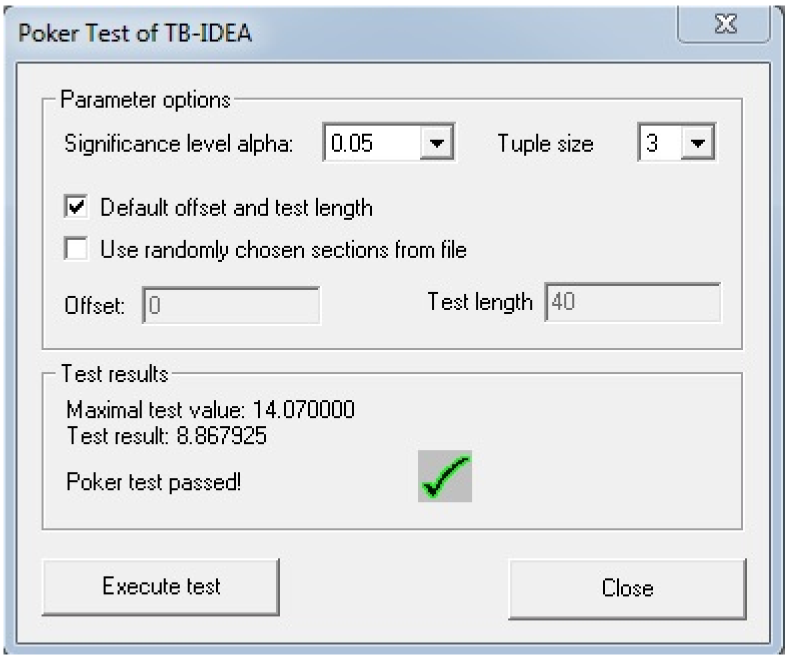
P. *Poker Test*

For TB-IDEA we have chosen importance degree of 0.05 and tuple 3 and the test has been successful (Fig. 9) which indicates that the values have been selected correctly and subsequences by tuple 3 occur evenly with a probability of 63% in the whole sequence. Since the poker test on the IDEA cryptograph failed by tuple 3, we compare the two test results with tuple 1 (Fig. 10). TB-IDEA cryptograph has a test result of 20% which shows that subsequences occur evenly with a relatively low probability and shows a significant improvement in comparison with the 75% of the probability of likelihood of IDEA cryptograph.

TABLE III. N-Gram test result of TB-IDEA cipher

Unigram				Bigram			
No	Character Sequence	Frequency in %	Frequency	No	Character Sequence	Frequency in %	Frequency
1	W	13.1579	5	1	SK	8.3333	3
2	C	7.8947	3	2	CV	5.5556	2
3	K	7.8947	3	3	MO	5.5556	2
4	N	7.8947	3	4	AW	2.7778	1
5	O	7.8947	3	5	BS	2.7778	1
6	S	7.8947	3	6	CH	2.7778	1
7	A	5.2632	2	7	DB	2.7778	1
8	D	5.2632	2	8	DJ	2.7778	1
9	M	5.2632	2	9	FC	2.7778	1
10	R	5.2632	2	10	GW	2.7778	1
11	V	5.2632	2	11	HS	2.7778	1
12	B	2.6316	1	12	IA	2.7778	1
13	F	2.6316	1	13	JM	2.7778	1
14	G	2.6316	1	14	KA	2.7778	1
15	H	2.6316	1	15	KG	2.7778	1
16	I	2.6316	1	16	KN	2.7778	1
17	J	2.6316	1	17	LM	2.7778	1
18	L	2.6316	1	18	NC	2.7778	1
19	P	2.6316	1	19	ND	2.7778	1

TriGram				FourGram			
No	Character Sequence	Frequency in %	Frequency	No.	Character Sequence	Frequency in %	Frequency
1	BSK	2.9412	1	1	BSKA	3.1250	1
2	CHS	2.9412	1	2	CHSK	3.1250	1
3	CVI	2.9412	1	3	CVIA	3.1250	1
4	CVS	2.9412	1	4	CVSK	3.1250	1
5	DBS	2.9412	1	5	DBSK	3.1250	1
6	DJM	2.9412	1	6	DJMO	3.1250	1
7	FCH	2.9412	1	7	FCHS	3.1250	1
8	GWR	2.9412	1	8	GWRN	3.1250	1
9	HSK	2.9412	1	9	HSKG	3.1250	1
10	JMO	2.9412	1	10	JMOD	3.1250	1
11	KAW	2.9412	1	11	KGWR	3.1250	1
12	KGW	2.9412	1	12	KNLM	3.1250	1
13	KNL	2.9412	1	13	LMOW	3.1250	1
14	LMO	2.9412	1	14	MODB	3.1250	1
15	MOD	2.9412	1	15	MOWW	3.1250	1
16	MOW	2.9412	1	16	NCVS	3.1250	1
17	NCV	2.9412	1	17	NDJM	3.1250	1
18	NDJ	2.9412	1	18	NLMO	3.1250	1
19	NLM	2.9412	1	19	ODBS	3.1250	1


Poker Test of TB-IDEA

Parameter options

Significance level alpha: 0.05 Tuple size 3

[✓] Default offset and test length

[ ] Use randomly chosen sections from file

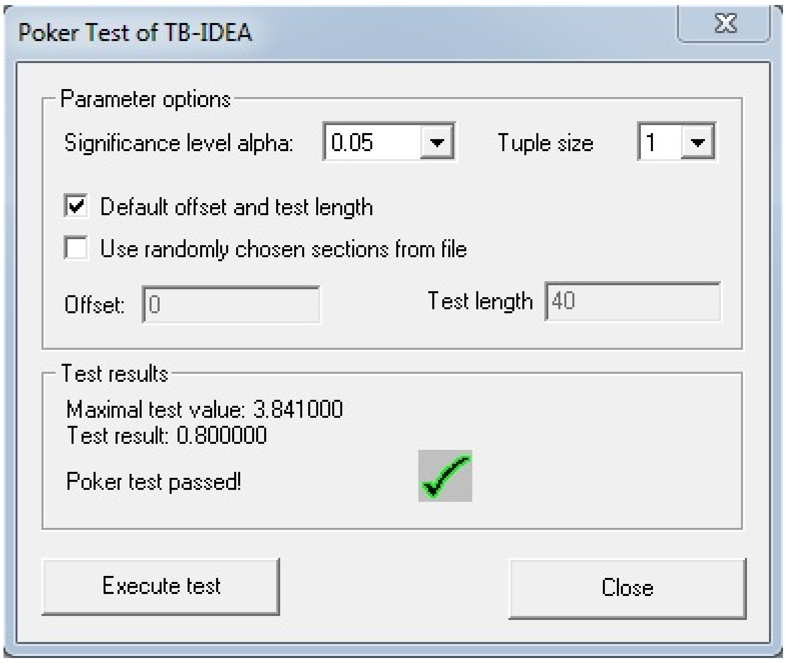
Offset: 0 Test length 40

Test results

Maximal test value: 14.070000
Test result: 8.867925

Poker test passed!

Fig. 9. Poker test result of TB-IDEA cipher with tuple size 3


Poker Test of TB-IDEA

Parameter options

Significance level alpha: 0.05 Tuple size 1

[✓] Default offset and test length

[ ] Use randomly chosen sections from file

Offset: 0 Test length 40

Test results

Maximal test value: 3.841000
Test result: 0.800000

Poker test passed! ✓

Fig. 10. Poker test result of TB-IDEA cipher with tuple size 1

VIII. Conclusion

As already mentioned, the world of encryption needs new algorithms and improvement of the existing ones to stay secure. The security of an encryption algorithm depends on the encryption operations and its key-schedule algorithm. In this paper, we attempted to make some improvements to IDEA key-schedule algorithm using Twofish and Blowfish schedulers which are secure enough and we proposed TB-IDEA algorithm. Then, we encrypted "New IDEA SubKeys" phrase once using the sub-keys generated by IDEA scheduler and the next time using the sub-keys generated by TB-IDEA scheduler and evaluated them using CT1 software that provides entropy, autocorrelation, N-Gram and poker tests.

According to Table II that summarizes the results of these tests on the two encrypted text, we concluded that we have not only optimized IDEA, but also improved its security significantly. TB-IDEA can be used along with other encryption algorithms to enhance the security of the data stored on cloud servers.

TABLE IV. Test results on IDEA and TB-IDEA ciphers

Algorithms / Tests	IDEA Algorithm		TB-IDEA Algorithm	
Entropy	47%		86%	
Autocorrelation	Relatively high		Relatively low	
N-Gram	Most	Least	Most	Least
Unigram	56.5217	4.3478	13.1579	2.6316
Bigram	42.8571	7.1429	8.3333	2.7778
Trigram	16.6667	16.6667	2.9412	2.9412
FourGram	50.0000	50.0000	3.1250	3.1250
Poker	75%		20%	

REFERENCES

[1] S. S. Manvi, G. K. Shyam,"Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey",Elsevier Journal of Network and Computer Applications,pp. 424-440, 2013.
[2] R.Kanday,"A survey on cloud computing security"in IEEE International Conference on Computer Science,pp. 302-31,2012.
[3] E. G. Amoroso,"Practical methods for securing the cloud,IEEE Cloud Computing,pp. 28-38,2014.
[4] Dr. S.A.M Rizvi, Dr. S. Z. Hussain, N. Wadhwa,"Performance analysis of AES and TwoFish encryption schemes",in IEEE International Conference on Communication Systems and Network Technologies, pp.76-79,2011.
[5] Y. Desmedt,What is the Future of Cryptography?, Lecture Notes on Computer Science. Berlin, Heidelberg. Springer, 2016, Vol.
[6] J. Daemen, R. Govaerts, J. Vandewalle,"Weak keys for IDEA",in Springer Annual International Cryptology Conference, pp. 224-231, 2001.
[7] S. Yang, H. Piao, L. Zhang, X. Zheng,"An improved IDEA algorithm based on USB security key",in IEEE International Conference on Natural Computation,pp. 184-188,2007.
[8] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, Ch. Hall, N. Ferguson,"Twofish: A 128-Bit block cipher"in First Advanced Encryption Standard Conference, 1998.
[9] S. Manku, K. Vasanth,"Blowfish encryption algorithm for information security",Asian Research Publishing Network, Vol. 10,2015.
[10] Sh. Afzal, U. Waqas, M. Akhtar Mir, M. Yousaf,"Statical analysis of key schedule algorithms of different block ciphers",Science International, pp. 1835-1839, 2015.
[11] J. Kelsey, B. Schneier, D. Wagner,"Key-schedule cryptanalysis of IDEA, G-DES, GOST, SAFER and Triple-DES",in Springer Annual International Cryptology Conference, pp.237-251, 1996.
[12] B.Schneier. Blowfish Source Code webpage on Schneier. [Online]. Available: www.schneier.com/academic/blowfish/download.html
[13] L. Reyzin,Some notions of Entropy for Cryptography,Lecture Notes on Computer Science. Berlin, Heidelberg. Springer,2011, Vol. 6673.

AUTHOR PROFILE

MasoudRafighi was born in Tehran, Iran on 1983/08/10. He is received his PhD degree in Engineering Information Technology from Qom University. He receives M.Sc degree in computer engineering software from Azad University North Tehran Branch, Tehran, IRAN. He has recently been active in software engineering and has developed and taught various software related courses for the Institute and university for Advanced Technology, the University of Iran. His research interests are in software measurement, software complexity, requirement engineering, maintenance software, software security and formal methods of software development. He has written a book on software complexity engineering and published many papers.

NaghmeMoatazedi was born in Alborz, Iran on 1993/07/03. She is received M.Sc degree in information security from Taali University Qom. She receives her B.S. degree in Information Technology from Kashani University Qazvin. Her research interests are in cloud computing, IoT, network and its security.