STATISTICAL ANALYSIS OF KEY SCHEDULE ALGORITHMS OF DIFFERENT BLOCK CIPHERS

Shazia Afzal [1], Umer Waqas [1], Mubeen Akhtar Mir [2], Muhammad Yousaf [1*]
[1] Riphah Institute of Systems Engineering, Riphah International University, Islamabad, Pakistan
[2] Pakistan Institute of Engineering and Applied Science, Islamabad, Pakistan
* Corresponding Author Email: muhammad.yousaf@riu.edu.pk

ABSTRACT: The key schedule algorithm of some ciphers is examined to evaluate their strength. The strength is evaluated through statistical analysis. Three data generation methods are used to determine the correlation and independence of sub-keys generated by a key schedule algorithm. Four basic statistical tests are used to evaluate these properties of key schedule. The results showed that the weak and strong key schedule can be distinguished by the help of these statistical tests.

Keywords: Auto-correlation Test, Independence Method, Key Schedule, Poker Test, Randomness, Sub-keys.

1. INTRODUCTION

The field of information security (IS) has grown and evolved significantly in recent years. IS is the field of defending information from unauthorized access, disclosure, use, modification, disruption, inspection, recording or destruction. Collected information is processed and stored on electronic computers and transmitted across networks to other computers. To protect the information at rest or in transit, the cryptographic algorithms are used. There are three types of cryptographic algorithms on the basis of the number of keys i.e. Secret Key Cryptosystem/Symmetric key Cryptosystem in which a single key is used for both encryption and decryption, Public Key Cryptosystem or Asymmetric Cryptosystem which used two keys, one for encryption (public key) and other for decryption (private key) and Hash/One-way function which performed mathematical transformation to convert a variable length input into a fixed length output and to be used as finger prints of input data..

The security of asymmetric cryptosystem lies on two parameters: the strength of algorithm and secrecy of the key. In practice cipher algorithms are publically known so the security of entire cryptosystem depends on the concealment of the secret or primary key. The key length should be long enough that there is no better way to break it except brute-force attack. Now a days, at least 128-bit key is recommended to be used for symmetric algorithms [1].

In a cipher algorithm a secret key is used to create the number of round keys (sub-keys) according to specified key scheduling algorithm. Each sub-key in a round of the encryption algorithm is mixed with input data which hide the correlation between input and output of the round.

The design of a good key schedule is a crucial part of cipher design. If the key schedule algorithm is not strong enough then the whole cryptosystem can be compromised by recovering the secret key due to some relationship between sub-keys [2]. In past, simple rotations or permutations are employed in the key schedule algorithm e.g. IDEA DES, TEA etc. use only simple permutations to generate the sub-keys from the secret key. After the development of cryptanalysis techniques the designers began to use the non-linear components in the key schedule algorithm. These components were used to avoid different attacks due to weak key schedule [3]. But still no clear criteria are developed to test the strength of key schedule algorithm. A weak key schedule generates nonrandom and related sub keys which make even a strong cipher algorithm vulnerable to attacks. So there is a need to define criteria for designing the strong key schedule.

The key schedule algorithm of Rijndeal [4] used the mixture of linear and non-linear transformations and generates 11 sub-keys of 128-bit length. The key schedule is based on the 32-bit words. Initial four words are generated from the master key. The remaining 40 words are generated by the iterative process. Successive groups of four 32-bit words are concatenated to produce the 128-bit sub-keys. The functions that are used in the key schedule algorithm are SubByte, Rotl (rotate left) and Rcon (round constant). Subbyte takes four bytes as an input and produces the four output bytes. Rotl is the byte rotation of the 32-bit word and Rcon are predefined 32-bit constants used to mix with first word of every successive group.

Ross Anderson et. al [5] explained the key schedule algorithm of the serpent. Serpent uses the 256-bit key to generate the 132 words of 32-bit length. User can enter the variable length key but the key schedule algorithm initially expand it to the 256-bit and then process it to calculate the 128-bit sub-keys. Affine recurrence and golden ratio is used in the key schedule algorithm with the round index to eliminate the weak keys and related keys. Finally s-boxes in bit slice mode are used to generate sub-keys.

Bruce Shenier et. al explained that Twofish key schedule algorithm [6] can pre-compute the sub-keys at maximum speed, or on-the-fly. Minimum memory is required and it is also suitable for dedicated hardware applications. Twofish utilize the encryption functions (MDS matrix and 8 s-boxes) in the key schedule algorithm which allow the generation of more secure sub-keys.

The cipher MARS accepts the variable key lengths [7]. MARS uses the s-box of order 9×32 in the key schedule algorithm. Initially, key schedule algorithm expands the input key then passes it through the s-box. 40 words of 32-bits are calculated by the key schedule algorithm. MARS employ the left rotation function and bit masking as a linear function in the key schedule algorithm.

RC6 key schedule algorithm uses the two magic constant in the key schedule algorithm [8] whereas, IDEA used simple permutation in the key schedule algorithm [9]. In IDEA, the block length is 64-bit and master key length is 128-bit. First eight sub keys are generated by utilizing the secret key. For second round the secret key is left shifted by 25-bits and next eight sub-keys are generated. This process will continue until

all sub-keys are generated for each of the eight rounds of the encryption algorithm.

Statistical tests can be used to test the key schedule algorithms strength. In this paper randomness testing is carried out on the key schedule algorithms that evaluate the randomness of the sub keys. When results showed fair randomness in sub keys then the key schedule is considered to be immune to related key and dependence between sub keys attacks. The rest of the paper is organized as: section II described the statistical tests for key schedule algorithm. Selected ciphers are listed in section III and results and analysis are presented in section IV and finally paper is closed with concluding remarks.

2. STATISTICAL TESTS

Block cipher randomness testing does not involve the key schedule algorithm testing which is an important part of any block cipher algorithm. Key schedule algorithm directly affects the security of the whole block cipher. For example, an attacker may construct a way to attack the block cipher algorithm by taking advantage of the weakness of key scheduling algorithm [10].

A well–designed Key-scheduling algorithm may ensure the statistical independence of each sub-key. In practice sub key is generated by inputting the previous sub key in the key schedule algorithm. So, by the design of a key-scheduling algorithm, it is impossible to make all sub-keys statistical independent and uncorrelated to each other [10]. However, it can be strived to make them as independent and uncorrelated as possible. Some percentage criteria for the statistical tests can be defined to evaluate the dependency and correlation between sub-keys.

2.1. Data Generation

To perform the evaluation task, key schedule algorithm can be expressed in 4-elements model. The model can be defined as

$$SK = F_{ks}(MK, n, m, r) \quad (1)$$

Where F_{ks} is key scheduling algorithm, MK is the secret key of length n-bits, m is the length of sub keys, r is the number of generated sub-keys and SK is a sub key set consisting of r-generated sub keys. Sub key set can be expressed as:

$$SK = \{K_1, K_2, \dots\dots\dots, K_r\} \quad (2)$$

Each K_i $(1 \le i \le r)$ is of m-bit length and $K_{i[j]}$ is j^{th} byte of K_i, where $1 \le j \le \frac{m}{8}$

To test the strength of F_{ks}, sequence is generated by using the sub-key set. There are three methods to generate the sequence that is discussed below [10]. In all three methods the process is described to generate a one sequence (Bseq) by inputting the one secret key in the F_{ks}. A set of 500 random vectors are generated by a PRNG named Blum-Blum Shub (BBS). These random vectors are used as secret keys in the F_{ks} and 500 sets of sub-keys are generated by each method which is illustrated in Fig. 1.

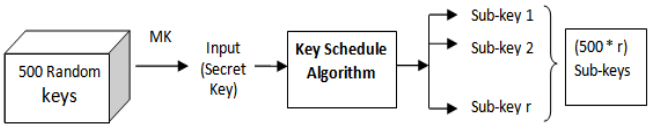

Fig. 1: Sub-Key Generation Process

From the generated 500 sets of sub-keys 500 sequences are generated for each method.

2.1.1. Sub-key Independence Method

To evaluate the dependency and correlation between two sub keys, all possible combinations of two sub-keys from *SK* set are XORed together. For example, K_1 is XORed with $K_2, K_3, \dots\dots\dots, K_r$ then K_2 with $K_3, \dots\dots\dots, K_r$ and so on. From K_i and K_j a bit string (BS) is generated as

$$BS_{ij} = K_i \oplus K_j \quad (3)$$

Bit strings results from the XOR of all combinations of sub keys are concatenated together to get a bit sequence (Bseq). This concatenated Bseq is then tested for randomness.

$$Bseq = BS_{12} || BS_{13} || BS_{14} || \dots\dots || BS_{r-1,r} \quad (4)$$

Total number of bits in Eq. (4) is

Sequence length= $C\,{}_2^r \times m$ (5)

Where '*C*' denotes the combination of all sub-keys.

2.1.2. Byte level Independence Method

To reveal the correlation between two bytes of sub keys, the sub-keys are divided into 8-bit block. In this method XOR is performed between all possible combinations of bytes of sub keys. The required sequence will be generated as:

$$Bseq = BS_{12} || BS_{13} || BS_{14} || \dots\dots || BS_{r-1,r} \quad (6)$$

where

$$(Ki \oplus K_j) = (K_{i[1]} \oplus K_j) || (K_{i[2]} \oplus K_j) || \dots\dots\dots || (K_{i[L]} \oplus K_j) \quad (7)$$

$$Where\ i \neq j \ and\ i = 1,2,\dots,r \quad j = 1,2,\dots,r$$

Total number of bits in Eq. (6) is

Sequence length = $C\,{}_2^r \times L \times L \times 8\ bits$ (8)

Here, L denote the number of bytes in sub key K_i $(1 \le i \le r)$.

2.1.3. Bit level Independence Method

To express the relationship between every bit of K_i and K_j, XOR the all possible combinations of bits of K_i and K_j. Generated data from this method can sufficiently reveal the relationship between every bit of different sub keys. Bit strings results from the XOR operation are concatenated together for the randomness testing. The required sequence is:

$$Bseq = BS_{12} || BS_{13} || BS_{14} || \dots\dots || BS_{r-1,r} \quad (9)$$

Where

$$BS_{ij} = (K_{i[1]} \oplus K_j) || (K_{i[2]} \oplus K_j) || \ldots\ldots\ldots || (K_{i[L]} \oplus K_j) \quad (10)$$

Here, $i \neq j, i = 1,2, \ldots., r, j = 1,2, \ldots., r$ and $(K_{i[k]} \oplus K_j)$ XOR the k^{th} byte of K_i with the all bits of K_j.

Total number of bits in Eq. (9) is

$$\text{Sequence length} = C\,{}_2^r \times m \times m \text{ bits} \quad (11)$$

2.2. Randomness Testing

Various statistical tests can be applied upon a sequence to test the randomness property, of that sequence. Statistical tests are used to test a null hypothesis (H_0). In this paper the null hypothesis H_0 is: The sequence being tested is random and alternative hypothesis H_a is: The sequence is not random [12]. Four local randomness tests are selected to test the null hypothesis i.e. frequency, runs, poker and autocorrelation tests. These basic tests sufficiently test the randomness property of a bit sequence and can be applied on small sequences. Other NIST tests can also be used to test the sequence randomness but care must also be taken about sequence length. Most of the NIST tests are applicable on the sequence of length 10^6 bit. For small sequences these NIST tests did not follow the prescribed distributions and shows misleading results.

The purpose of selected four basic tests is described below.

2.2.1. Frequency test

This test checked whether the number of '0' or '1' in the sequence meet the criterion of random sequence i.e. number of '0' and '1' are equal or not [11,13].

2.2.2. Runs test

The purpose of runs test is to test whether the behavior of changes in a sequence meet the criterion of random sequence [11, 13].

2.2.3. Poker test

Poker test checks the uniformity distribution of p-bits pattern in the whole sequence i.e. the number of times the p-bits block appears in the entire sequence should be same [11].

2.2.4. Autocorrelation test

The purpose of autocorrelation test is to test the degree of dependence between a sequence and its shifted sequence. Shifted value is denoted by d [11].

In the randomness testing, if any sequence fails the frequency test then there is no need to apply the remaining three tests. Since the randomness testing is the probabilistic approach, therefore 500 sequences are generated to draw conclusion on the independence among sub-keys [12]. Here, random secret keys are used for testing process; hence this can also be extended by included non-random secret keys.

3. SELECTED KEY SCHEDULE ALGORITHMS

Six block cipher are selected for analysis purpose. The key schedule algorithms of selected ciphers are implemented in software. The selected block ciphers are AES, MARS, Serpent, RC6, Twofish and IDEA. The key schedule algorithm of all block ciphers uses 128-bit secret key expects Serpent that take 256-bit secret key.

Sub-key length of all key schedule algorithms is 128-bit. AES, MARS, Serpent and TwoFish used linear and non-linear functions in key schedule whereas RC6 uses constants and linear functions in key schedule. The cipher IDEA uses only cyclic shift of secret key to generate the sub-keys. Each key schedule algorithm is tested for the independency and correlation property among sub-keys by applying above mentioned four tests.

4. TEST PARAMETERS

For each key schedule algorithm F_{ks}, 11(r = 11) sub-keys are generated for testing. For IDEA, only seven (r = 7) sub-keys are generated since IDEA uses 8 sub keys including secret key in the encryption algorithm. For each key schedule algorithm m, length of sub-key is taken as 128-bit. Sequence length of AES, MARS, Serpent, RC6 and TwoFish key schedule algorithms generated by the sub-key method, byte level method and bit level method, are 7040 bit, 112640 bit and 901120 bit respectively. For IDEA, these sequences length are 2688 bit, 43008 bit and 344064 bit.

500 sequences are generated by each of the three methods. To generate 500 sequences a random set of 500 secret keys is generated from the pseudo random number generator BBS. Testing process can also be extended for the non-random keys to test sub keys randomness.

To test the null hypothesis of randomness, level of significance α is chosen as 0.10. Other values of α can also be chosen depending upon how much dependency and correlation one can tolerate among the sub keys. α = 0.10 indicates that one would expect 10 out of 100 sequences may not pass the test this value of α. In this case, 50 out of 500 sequences may not pass the tests. The threshold value for passing test will be 90%.

5. RESULTS AND DISCUSSION

In this research work, defined passing criteria for the randomness testing of sequences are 90%. But as mentioned in the section II that it is impossible to make all sub keys independent hence the results near to 90% shall also be considered satisfactory. If any key schedule algorithm shows result far away from passing criteria of randomness testing, it will be concluded that the key schedule algorithm is vulnerable to key-dependent attacks.

500 sequences are generated in each method for the selected key schedule algorithms of six ciphers. In the following section the results are analyzed in detail.

5.1. Sub key Independent Method

Selected 500 random keys are entered as a secret key to the key schedule algorithms. The sub keys are generated and the sequences to test the independence between sub-keys are generated. Four randomness tests are performed on the generated sequences and the results are presented in Table 1.

The results of Frequency test for the key schedule of Serpent, RC6 and TwoFish are above 90% which shows the balance of zero and one in the generated sequence. Remaining three test results of these key schedules also satisfy the defined passing criteria. These results indicate that the sub keys generated by the key schedule of Serpent, RC6 and TwoFish are independent and uncorrelated at sub key level.

Table 1: Results by Sub-Key Independent Method

Cipher	*Frequency Test*	*Runs Test*	*Poker Test*	*Auto-correlation test*
AES-128	76.6%	92.40%	99.6%	90.6%
MARS	48.4%	78%	98%	86%
Serpent	95.4%	97%	100%	94.2%
RC6	93.8%	92.4%	100%	92.8%
TwoFish	96.2%	98.6%	100%	94.8%
IDEA	45%	100%	97.4%	43.2%

Frequency test percentage of AES key schedule is 76.6% which is far away from the passing criteria and point out the unbalance distribution of zero and one in the sequence. Other three test results for the AES fall into the passing range but since frequency test is failed hence it is concluded that the sub keys of AES are correlated. The result of frequency tests for key schedule algorithm of IDEA is 45% which showed the unbalanceness of zero and one in the sequences. Autocorrelation test result of IDEA is 43.2% which demonstrate the correlation between the sequences and their shifted versions. key schedule of IDEA passes the runs and poker test with high percentage since frequency and autocorrelation tests do not fall into the passing range, therefore, it is indicated that key schedule of IDEA does not pass the independence criteria. MARS key schedule algorithm also does not pass the frequency and runs tests. It indicates the dependency and correlation among the generated sub keys of MARS. The results showed that the TwoFish key schedule algorithm is best among the 6 key schedule algorithms.

5.2. Byte level Independence Method

Sub-keys are generated by inputting the 500 random keys into the key schedule algorithms. To test the independence between sub-keys at byte level, the sequences are generated by the byte level independence method described in section II. The statistical tests are performed on the generated sequences to test the randomness. The results are given in Table 2.

Table 2: Results by Byte Level Independent Method

Cipher	*Frequency Test*	*Runs Test*	*Poker Test*	*Auto-correlation test*
AES-128	90.6%	95.20%	98%	90.6%
MARS	74%	85.60%	98%	74%
Serpent	95%	98.40%	99.8%	95%
RC6	97%	97.8%	100%	97%
TwoFish	97.6%	98.2%	100%	97.6%
IDEA	0%	-	-	0%

AES, Serpent, RC6 and TwoFish key schedules results showed that percentage of passed sequences for all tests is above 90%. This high percentage indicates that the bytes of sub-keys generated by the AES, Serpent, RC6 and TwoFish key schedule algorithms are statistically independent and uncorrelated. The result of key schedule algorithm of MARS described that it passes frequency test with percentage of 74. The 74% indicate that distribution of zero and one is not uniform. Other three test results of MARS key schedule algorithm are 85.60%, 98% and 89.9% that are near or above the 90%. However, MARS do not pass the frequency test, hence, bytes of sub keys are somewhat correlated. Since key schedule algorithm of IDEA uses only the permutation function it showed the poor results for the frequency test. Since frequency test showed 0% results therefore there is no need to apply the other tests of randomness. Results of IDEA show the strong correlation between the bytes of the sub-keys.

5.3. Bit level Independence Method

To test the independence between the sub-keys at bit level, data is generated by using the 500 random keys. This method is stronger than the remaining two methods since it checks the correlation between sub keys at bit level. Each bit of the sub keys is tested for the correlation with the all bits of other sub keys. If any key schedule algorithm passes the bit level correlation test then the key schedule is suppose to be immune to the key-dependent attacks. This method clearly distinguishes between the weak and strong key schedule algorithm. The test results for this method are given in Table 3.

TwoFish key schedule algorithm showed good results as compared to the rest of the algorithms. TwoFish passed all the randomness tests with high percentage. TwoFish uses strong components (MDS and s-box) in the key schedule algorithm which make it a strong key schedule algorithm. AES key schedule also showed good independency among sub keys at bit level. Only autocorrelation test percentage is 88.4%, however it is very close to 90%. It can be concluded that the sub keys of AES are satisfactorily statistically independent. Serpent key schedule algorithm also generates the independent sub keys as indicated by the results.

Table 3: Results by bit level independence Method

Cipher	*Frequency Test*	*Runs Test*	*Poker Test*	*Auto-correlation test*
AES-128	90.25%	92%	95.8%	88.4%
MARS	83%	87%	97%	91.2%
Serpent	95.6%	90.6%	100%	94.2%
RC6	77.4%	83%	91.8%	97.2%
TwoFish	93%	95.60%	99.6%	93.6%
IDEA	0%	-	-	-

Frequency test of MARS key schedule algorithm is 83% and 87% for the runs test. But the poker and autocorrelation test of MARS satisfied the threshold level. 83% frequency test of MARS demonstrates that there is some dependency between the sub keys. RC6 key schedule algorithm also did not pass the frequency and run test which described the dependency at bit level among generated sub keys. IDEA key schedule algorithm showed 0% results for frequency test which indicate that all 500 sequences did not pass the randomness tests. The poor results of IDEA for all tests indicate high dependency of sub keys at bit level.

6. CONCLUSION

Key schedule algorithm is an important part of any block cipher algorithm and its strength directly affects the security of the cipher. In literature, there are no defined criteria to test the strength of key schedule algorithm. In this paper, three different data generation methods are described to test the independence and relationship among the sub keys generated

by the key schedule algorithm. Method 1 checks the dependency between sub keys at the whole sub key level. Method 2 checks the correlation between the bytes of sub keys and Method 3 reveals the correlation among the bits of sub keys. A weak key schedule algorithm may pass the first method but second and third method pointed out all the weaknesses of the key schedule algorithm. Frequency, runs, poker and autocorrelation tests are used to test the randomness of the generated sequences. The results have shown that the key schedule algorithms which use only linear permutation do not pass the statistical tests e.g. IDEA. During design stage these test must also be performed to test the dependence and correlation between sub-keys so that many attacks can be mitigated which take the advantage of dependency or linear relationship between sub-keys and break the whole cipher.

REFERENCES

[1] J.N.B. Salameh ,A New Technique for Sub-Key Generation in Block Ciphers, World Applied Sciences Journal 19.11 (2012) 1630-1639, DOI: 10.5829/idosi.wasj.2012.19.11.1871.

[2] U. Blumenthal, M. Bellovin ,A Better Key Schedule For DES-LIKE Ciphers, Proceedings of Pragocrypt 1996.

[3] wikipedia, Key schedule, http://en.wikipedia.org/wiki/Key_schedule , (5 May 2014).

[4] J. Nechvatal, E. Barker, L. Bassham, W. Burr, M. Dworkin, Report on the development of the Advanced Encryption Standard (AES), NIST, ADA396556, http://csrc.nist.gov/archive/aes/round2/r2report.pdf (2000).

[5] R. Anderson, E. Biham, L. Knudsen, Serpent: A proposal for the advanced encryption standard, NIST AES Proposal 174 (1998).

[6] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, Twofish: A 128-bit block cipher, NIST AES Proposal 15 (1998).

[7] C. Burwick, D. Coppersmith, E.D. Avignon, MARS-a candidate cipher for AES, NIST AES Proposal 268 (1998).

[8] R.L. Rivest, M. J. B. Robshaw, R. Sidney, The RC6TM block cipher, First Advanced Encryption Standard (AES) Conference. (1998).

[9] J. Daemen, R. Govaerts, J. Vandewalle, Weak keys for IDEA, In Advances in Cryptology—Crypto'93 (pp. 224-231), Springer Berlin Heidelberg, (1994) DOI: 10.1007/3-540-48329-2_20.

[10] Z.Wenzheng & Y. Cao ,Randomness test of key schedule algorithm ,2006.

[11] U.M. Maurer,A universal statistical test for random bit generators, Journal of cryptology 5.2 (1992) 89-105.

[12] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, A statistical test suite for random and pseudorandom number generators for cryptographic applications, BOOZ-ALLEN AND HAMILTON INC MCLEAN VA, (2001).

[13] C. Loredana et al, Randomness Evaluation Framework of Cryptographic Algorithms, International Journal on Cryptography and Information Security (IJCIS), Vol. 4, No. 1, (March 2014).