Hindawi
Security and Communication Networks
Volume 2020, Article ID 3189601, 9 pages
https://doi.org/10.1155/2020/3189601

Research Article

Cryptographic Strength Evaluation of Key Schedule Algorithms

Shazia Afzal,[1] Muhammad Yousaf,[1] Humaira Afzal,[2] Nawaf Alharbe,[3] and Muhammad Rafiq Mufti [4]

[1]*Riphah Institute of Systems Engineering, Riphah International University, Rawalpindi, Pakistan*
[2]*Department of Computer Science, Bahauddin Zakariya University, Multan, Pakistan*
[3]*College of Community, Taibah University, Medina, Saudi Arabia*
[4]*Department of Computer Science, COMSATS University Islamabad, Vehari Campus, Vehari, Pakistan*

Correspondence should be addressed to Muhammad Rafiq Mufti; rafiq_mufti@ciitvehari.edu.pk

Received 22 October 2019; Accepted 30 April 2020; Published 14 May 2020

Academic Editor: Roberto Di Pietro

Key schedule algorithms play an important role in modern encryption algorithms, and their security is as crucial as the security of the encryption algorithms themselves. Many studies have been performed on the cryptographic strength evaluation of the encryption algorithms; however, strength evaluation of the key schedule algorithms often obtains less attention that can lead towards the possible loophole in the overall encryption process. In this paper, a criterion is proposed to evaluate the cryptographic strength of the key schedule algorithms. This criterion includes different methods of data generation from subkeys and a suitable set of statistical tests. The statistical tests are used to explore the cryptographic properties such as diffusion, confusion, independence, and randomness in the subkeys generated by the key schedule algorithm. The proposed criterion has been applied to some of the key schedule algorithms of different block ciphers. The results confirm that the proposed criterion can effectively differentiate between strong- and weak-key schedule algorithms.

1. Introduction

The security of a block cipher mainly depends on the strength of the encryption algorithm and the secrecy of the secret key. The security of an encryption algorithm is achieved through the nonlinear and linear functions/components [1]. Substitution box(s) or some multiplication functions are used to introduce the nonlinearity in the encryption algorithm. Linear functions such as permutation, circular shifts, and MDS matrix diffuse the nonlinearity on the whole input block [2]. These functions enable the encryption algorithm to hide the relationship between plain text and ciphertext.

The security of an encryption algorithm is directly affected by the strength of the Key Schedule Algorithm (KSA). The KSA drives the subkeys from the secret key which are used in each round of an encryption algorithm. It is required that the subkeys generated by KSA may not have any linear relationship with each other as well as with the secret key. The KSA needs to be strong enough such that if an attacker gets knowledge of some bits of the subkeys or even knows a single subkey, then he could not be able to calculate the whole secret key [3]. The cipher designers take care of Shannon's principal during the design stage of an encryption algorithm to achieve confusion and diffusion properties. These properties should also be taken care of in a KSA in order to have independence among subkeys.

A simple and weak KSA generates the subkeys that have a linear relationship with each other and the secret key. Related-key attacks and slide attacks take the advantage of this simple relationship between the subkeys and secret key. Such relationship also makes the cipher vulnerable to linear and differential attack [4]. A strong KSA helps to make the overall cipher more resistant to linear and differential attack.

In comparison to the requirements of the strong encryption algorithm, there has been less focus on the requirements of a strong key schedule algorithm in the literature. Knudsen [5] pointed out that a key schedule is strong if all subkeys are equally secure. According to Blumenthal et al. [3], several attacks can take advantage of the linear relation between subkey bits. To achieve maximum avalanche effect in the subkeys, they proposed that a KSA must possess the property that all input key bits must equally affect the output subkeys of the KSA. In [6], the authors recommended that related key and differential attacks can be thwarted by maximizing the avalanche in the subkeys and avoiding the linearity in KSAs.

May et al. [7] proposed three desirable properties for the KSA, that is, collision resistance one-way function, independence between subkeys and secret-key bits, and implementation efficiency. Two statistical tests, namely, frequency test and Strict Avalanche Criteria (SAC) of the CryptX test suite, were used to measure the strength of KSA of AES. In [8], evaluation of the independence between the subkeys generated by the KSA is presented. The authors have shown that independence among the subkeys ensures the KSA's strength against the key-dependent attack. The avalanche effect is a desirable cryptographic property in the cryptographically secure algorithms. If a KSA shows good avalanche value, then it indicates more security. Low avalanche value shows the poor randomization in the algorithm which can help the cryptanalyst to partially and completely break the algorithm [9].

The strength of a KSA depends on the functions used to generate the subkeys. In the literature, many KSAs are proposed which use different techniques to generate the subkeys. For example, IDEA block cipher uses the linear function (circular shift) to generate the subkeys [10]. Similarly, KSA of PRESENT block cipher [11] uses the linear permutation as a major component to generate the subkeys. However, the nonlinear function (s-box) is used only on the limited number of bits. Twofish uses a complicated key schedule algorithm. It utilizes a combination of confusion and diffusion in its KSA to generate the subkeys [12]. The KSA of AES uses the linear and nonlinear functions to generate the subkeys [13]. Huang and Xuejia modified the KSA of AES [14] by taking the transpose of the subkey matrix. The authors claimed that this modification will make the new KSAs immune against the SQUARE, meet-in-the-middle, and the related-key attacks.

The authors in [15] are of the view that a strong key schedule can be designed by using only those functions which are the part of the encryption algorithm. The KSA of Serpent block cipher falls in this category [16]. May et al. [6] proposed a new KSA of AES based on all functions taken from the encryption algorithm. The proposed KSA passed the frequency and SAC tests from the CryptX test suite. The authors also claimed to reduce the code size of the overall block cipher. The authors in [17] identified the weaknesses in the key schedule algorithm of RC4. In [18], the authors described the benefits of using chaos in the key schedule algorithm, and in [19], the authors presented a multidimensional key algorithm for RC6. However, in all of these studies, no criteria were discussed for evaluating the cryptographic strength of the key schedule algorithms.

The above studies reflect that the evaluation of the KSA during its design stage has been the less focused area in the literature. Unfortunately, there is no known testing tool kit to evaluate the cryptographic strength of a KSA. The bit confusion, bit diffusion, independence, and randomness are desirable cryptographic properties of subkeys generated by any KSA. Therefore, it is proposed that a KSA may at least be tested against these properties.

This research work aims to propose testing criteria to evaluate the cryptographic strength of any key schedule algorithm. It includes methods of data generation from subkeys and a suitable set of statistical tests that explore the cryptographic properties of a key schedule algorithm. Studies show that although statistical tests are not the sufficient criteria for claiming the cryptographic strength of the cryptographic algorithms, however, it provides necessary criteria for a strong cryptographic algorithm. If an algorithm passes the statistical tests, it does not mean that the algorithm will withstand all the possible attacks; however, if an algorithm fails the necessary statistical tests, then it surely will not withstand even the basic attacks on the algorithm [20–23]. Therefore, along with the cryptographic properties, namely, confusion, diffusion, independence, correlation, and time complexity, the statistical tests of frequency, bit independence, bit correlation, and high/low-density key are considered in the proposed strength evaluation criterion of the KSA.

The remaining paper is organized as follows: Section 2 briefly describes the criteria to evaluate the cryptographic strength of any KSA. For analysis, four KSAs are selected from literature and are listed in Section 3. The data generation process for the strength evaluation of KSA is presented in Section 4 and results are discussed in Section 5. The concluding remarks are given in the last section of this paper.

2. Key Schedule Evaluation Criterion

In the current era of modern block ciphers, breaking the cipher is relatively harder and required lots of resources and time, so attackers are more interested in finding the secret key by other means, for example, brute force attacks, dictionary attacks, and deriving key bits of subkeys. Thus secure KSA is necessary for any block cipher and it directly affects the security of block cipher [8, 24]. A KSA must possess strong confusion and diffusion properties and all generated subkeys must be independent of each other so that any compromised subkey does not reveal any information about other subkeys or secret keys. A week key schedule algorithm makes a strong cipher vulnerable to many statistical and other cryptanalysis attacks. If the KSA generates the subkeys that have simple relationship, then it will degrade the security of the overall cipher, in order to resist against the related-key attacks, slide attacks, linear and differential attacks [4], boomerang and rectangle attacks [25], and SQUARE and meet-in-the-middle attacks [14]. Therefore, a key schedule algorithm should be well designed and complex.

In this paper, we define a Key Schedule Evaluation Criterion (KSEC) that can evaluate the cryptographic properties such as confusion, diffusion, randomness, and independence among subkeys. In this regard, four sets of statistical tests, namely, frequency, bit independence, bitwise uncorrelation, and high/low-density keys, are selected to evaluate the abovementioned cryptographic properties. Moreover, as complex KSA can affect the time efficiency of a block cipher, so time efficiency of KSA is also calculated.

The description of selected cryptographic sets of tests is given in the following sections. Before explaining the tests, it is necessary to express a key schedule algorithm in the 4-element model [8]. This model can be defined as

$$\mathrm{SK} = F(K, n, m, r) = \{K_1, K_2, \ldots\ldots\ldots., K_r\}. \quad (1)$$

Here, SK is a subkey set, F is the key schedule algorithm function, K is the secret key to length n-bits, m is the length of subkeys, and r is the number of subkeys.

2.1. Frequency Test. In random data, the proportion of zero and one should be close to 50%. The purpose of this test is to determine whether the numbers of 1's and 0's are equal in the subkeys [15] generated by a KSA. If the subkeys fail to pass the frequency test, it implies that the subkeys do not fulfill the basic requirement of randomness. Therefore, that KSA will be considered weak and there is no need to apply the remaining tests to explore other weaknesses. A brief description of the frequency test is given as follows.

Let a_0 and a_1 denote the number of 0's and 1's in an n-bit sequence and the statistics used is [16, 26, 27]

$$Z = \frac{(a_0 - a_1)^2}{n}. \quad (2)$$

Here, Z approximately follows a χ^2 (chi-square) distribution with 1 degree of freedom if $n \geq 10$.

2.2. Bit-Independence Tests (BITs). The second test in the KSEC is bit-independency tests (BITs) which is used to check the bit confusion and diffusion properties of a KSA. Confusion and diffusion properties in a KSA ensure that the relationship between the secret key and the subkeys is complex and a bit change in the secret key (K) will propagate the effect on all subkeys bits. The aim of these properties in the KSA is to prevent the cipher from the application of statistical attacks and cryptanalysis attacks like related-key attacks, slide attacks, and so on. BITs depend on the degree of completeness (d_c), avalanche effect (d_a), and strict avalanche effect (d_{sa}) [28]. To calculate these values, avalanche vectors are produced by XORing the subkeys generated by K and $K^{(i)}$ (for $i = 1, 2, \ldots, n$), where $K^{(i)}$ is obtained by complementing the ith bit of K. A brief description of BITs is given in the following sections.

2.2.1. Completeness. A function $f\colon (\mathrm{GF}(2))^n \longrightarrow (\mathrm{GF}(2))^m$ of n input bits maps into m output bits is said to be complete, if each output bit depends upon all input bits [28]; that is,

$$\mathrm{GF}(2)^n \quad \text{with } \left(f(x)^i\right)_j \neq (f(x))_j. \quad (3)$$

$$\forall i = 1, \ldots\ldots\ldots, n \text{ and } j = 1, \ldots\ldots\ldots, m.$$

2.2.2. Avalanche Effect (AE). A function $f\colon (\mathrm{GF}(2))^n \longrightarrow (\mathrm{GF}(2))^m$ has the avalanche effect, if an average of 1/2 of the output bits changes with the change in the single bit of the input [28]; that is,

$$\frac{1}{2^n}\sum w\left(f(x)^i - f(x)\right) = \frac{m}{2}, \quad \forall i = 1, 2, \ldots\ldots, n. \quad (4)$$

2.2.3. Strict Avalanche Criteria (SAC). A function $f\colon (\mathrm{GF}(2))^n \longrightarrow (\mathrm{GF}(2))^m$ satisfies the strict avalanche criterion, if each output bit changes with a probability (Pr) of 1/2 whenever a single input bit is complemented [28, 29]; that is,

$$\mathrm{Pr}\left[\left(f\left(x^i\right)\right)_j \neq (f(x))_j\right] = \frac{1}{2}. \quad (5)$$

$$\forall i = 1, \ldots\ldots\ldots, n \text{ and } j = 1, \ldots\ldots\ldots, m.$$

A KSA is said to have a good degree of completeness, avalanche effect, and strict avalanche criterion if the following equalities are satisfied:

$$\begin{cases} d_c = 1, \\ d_a \approx 1, \\ d_{sa} \approx 1. \end{cases} \quad (6)$$

If a KSA fails to satisfy these limits, then KSA does not exhibit good confusion and diffusion properties.

2.3. Bitwise-Uncorrelation Tests (BUCT). The third test in the KSEC is BUCT which checks the correlation among the bits of subkeys [8]. A KSA is said to pass BUCT if all subkeys are bitwise not correlated with each other. The subkeys which hold this property exhibit immunity against the linear and differential attacks and the key-dependent attacks. To express the relationship between every bit of K_i and K_j, XOR all possible combinations of bits of K_i and K_j. Bit strings resulting from the XOR operation are concatenated to get the required sequence which is explained in the following:

$$\mathrm{BinarySeq} = (K_1 \oplus K_2) \| (K_1 \oplus K_3) \| \ldots \| K_2 \oplus K_3 \| \ldots\ldots \| \left(K_i \oplus K_j\right) \| .. \| (K_{r-1} \oplus K_r), \quad (7)$$

where

$$\left(K_i \oplus K_j\right) = \left(K_i[1] \oplus K_j\right) \| \left(K_i[2] \oplus K_j\right) \| \ldots\ldots\ldots \| \left(K_i[L] \oplus K_j\right) \quad (8)$$

Here, $i \neq j$, $i = 1, 2, \ldots, r$, $j = 1, 2, \ldots, r$, and $(K_i[L] \oplus K_j)$ is the XOR between the Lth byte of K_i with all bits of K_j.

The binary sequence generated by (7) is then tested for randomness by using four statistical tests including frequency, runs, poker, and autocorrelation which are described below. All subkeys are said to be bitwise uncorrelated if generated data by (7) passes all these four statistical tests.

2.3.1. Frequency Test. The definition of this test has been described above in Section 2.1. However, here the frequency test is applied to the data generated by (7).

2.3.2. Runs Test. Let O_i denote the number of runs of 0 of length i and 1_i denotes the number of runs of 1 of length i in an n-bit sequence [27, 30]. The length of expected continuous "0" or "1" is $e_i = (n - i - 3)/2^{i+2}$. Let L be the largest of i, $1 \le i \le L$. The following statistics is used to evaluate the length of runs in a sequence:

$$Z = \sum_{i=1}^{L} \frac{(I_i - e_i)^2}{e_i} + \sum_{i=1}^{L} \frac{(O_i - e_i)^2}{e_i}. \quad (9)$$

Here, Z follows χ^2 (chi-square) distribution with $(2L-2)$ degrees of freedom.

2.3.3. Poker Test. Poker test checks the number of times the P-bits block appears in an n-bit sequence [27]. Divide the sequence into B nonoverlapping blocks, each of length P. Suppose b_i is the i^{th} bit of a P-bit sequence. The following statistics is used to test the uniformity distribution of P-bit blocks:

$$Z = \frac{2^P}{B} \sum_{i=1}^{2^P} (b_i)^2 - B, \quad (10)$$

where $4 \le P \le 8$ and Z approximately follows a χ^2 (chi-square) distribution with $2^P - 1$ degree of freedom.

2.3.4. Autocorrelation Test. Autocorrelation test checks the degree of dependence between a sequence (S_i) and its shifted sequence (S_{i+d}) [27]. Let d be a fixed integer, $1 \le d \le (n/2)$; then the following test statistics are used to check the dependency among (S_i) and (S_{i+d}):

$$Z(d) = \sum_{i=0}^{n-d-1} S_i \oplus S_{i+d}, \quad (11)$$

where $n - d \ge 10$ and then $Z(d)$ follows a $N(0,1)$ distribution.

2.4. High/Low-Density Key Test. The purpose of this set of tests is to evaluate the strength of KSA when the secret key is nonrandom. A secure KSA is needed to generate random subkeys even when the input secret key is nonrandom. The low- and high-density bit vectors are used as nonrandom secret keys. The low-density keys are the set of secret keys in which one and two bits of the secret keys are 1, whereas other bits are zero. For high-density keys, the secret key has one or two zero bits, while the remaining bits are 1. The generated subkeys from nonrandom secret keys are tested against the randomness property. Four statistical tests, namely, frequency, runs, block frequency, and cusum [26], are selected for testing the randomness of subkeys. The said tests are taken from the NIST test suite since they can be applied on small sequences of subkeys (128-bit long) [26]. The other tests of NIST suite are not applied here due to the requirement of larger sequence length. The frequency and runs tests have been described in Sections 2.1 and 2.3, respectively; therefore only the block frequency and cusum tests are described in the following sections.

2.4.1. Block Frequency Test. The n-bit input sequence is divided into $B = n/M$ nonoverlapping blocks to perform this test. This test determines that the frequency of "ones" may be approximately equal to $M/2$ in each M-bit block. The following test statistics [26, 30] is used to calculate the frequency of one or zero in B:

$$\chi^2 = 4M \sum_{i=1}^{B} \left(\pi_i - \frac{1}{2}\right)^2, \quad (12)$$

where π_i is the proportion of 1 in each M-bit block. The test statistics follow a χ^2 (chi-square) distribution with B degree of freedom.

2.4.2. Cumulative Sum (Cusum) Test. In this test, 0's and 1's of the n-bit sequence are converted to values −1 and +1 using X_i, where $X_i = (2\varepsilon_i - 1)$. Then calculate the cumulative sum S_i of partial subsequences, each starting with X_1 for forward (0) mode or X_n for backward (1) mode [26]; that is,

$$\begin{aligned} S_k &= S_{k-1} + X_k \quad \text{for mode } 0, \\ S_k &= S_{k-1} + X_{n-k+1} \quad \text{for mode } 1. \end{aligned} \quad (13)$$

The test statistics Z is given in the following equation:

$$Z = \max_{1 \le k \le n} |S_k|, \quad (14)$$

where Z is a normal distribution with N (0, 1).

2.5. Time Complexity. Time complexity is another parameter to compare the computational efficiency of different KSAs [7]. A block cipher with good security and lesser execution time is preferred in real-time applications. Since KSA is an important part of a block cipher, so its execution time also contributes to the time complexity of an entire block cipher. A good key schedule is one that exhibits good confusion, diffusion, and independence properties as well as less execution time. The execution time of KSAs is also estimated and is given in the results and discussion section.

Each test described above needs some values of the parameter for the calculation of the results. Table 1 presents the values of different parameters and the minimum number of secret keys needed to perform these tests. Since the frequency, bitwise-uncorrelation and weak key tests involve the hypothesis testing; therefore, their threshold level is determined by the level of significance α. The value of α can be changed according to the requirement.

Table 1: Parameter values.

No	Tests	Property	Number of secret keys	Threshold level	Length of sequence (bits)
1	Frequency test	Balance of zero and one	10,000 random keys	$\alpha = 0.03$	M
2	Bit-independence tests (i) Completeness (ii) Avalanche (iii) SAC	Confusion and diffusion	10,000 random keys	$d_c \approx d_a \approx d_{sa} \approx 1$	$(n \times 10{,}000)$
3	Bitwise-uncorrelation tests (i) Frequency (ii) Runs (iii) Poker ($P = 4$) (iv) Autocorrelation ($d = 2$)	Correlation among sub keys	500 random keys	$\alpha = 0.10$	$(rC_2 \times m \times m)$ $^*C = ((r!)/2!(r-2)!)$
4	High/low-density key tests (i) Frequency (ii) Block frequency ($M = 20$) (iii) Runs (iv) Cusum	Randomness of sub keys	8257 high/low-density keys	$\alpha = 0.01$	M

3. Key Schedule Algorithms

As the minimum recommended key size for block cipher is 128, therefore, for testing the security strength of KSAs, the proposed evaluation criteria are applied to the popular published algorithms that have secret key length of 128 bits or 256 bits. The KSAs of AES, Serpent, PRESENT, IDEA, and Twofish are selected and analyzed through the proposed criterion. The standard implementation of these algorithms with key size of 128/256 bits is considered. The subkey length used in the round function for these algorithms is different but for comparison purpose we took all the key lengths equal to 128 bits.

Triple DES is not selected for the evaluation process as it is not considered secure any more by NIST in 2017 and also Microsoft in 2018 [31, 32]. Similarly, we did not select Blowfish (1993); instead we selected its superior version of Twofish (1998) for the evaluation purpose [33]. Table 2 describes the secret key length and subkey length for these KSAs. In this work, 11 subkeys are generated against each secret key by all KSAs since these numbers of subkeys are found to be sufficient to depict the statistical characteristics of a KSA.

Table 2: Secret key size and subkey size of selected KSA.

Ciphers	Block size (bits)	Secret key size (bits)	Subkey size (bits)
AES	128	128	128
Serpent	128	256	128
PRESENT	64	128	128
IDEA	64	128	128
Twofish	128	128	128

4. Data Generation

The data generation methodology for each test is described in the following sections.

4.1. Frequency Test. To test the balance property of subkeys, a set of ten thousand random secret keys $\{K1, K2, ..., K10, K10000\}$ is taken and the subkeys are generated from each selected KSA. The SK set for KSAs can be presented as given in the following:

$$\mathrm{SK} = F(K_i, n, 128, 11), \tag{15}$$

where n is the length of the secret key and $i = 1, 2, \ldots, 10000$. The frequency test is performed on 11 sets of 10,000 subkeys generated for each round. The chosen level of significance α is 0.03 which indicates that 97% of sequences of subkeys should pass this test.

4.2. Bit Independence Tests. In BITs, ten thousand-random secret key set is taken $\{K1, K2, K3, \ldots, K10{,}000\}$ and the subkeys are generated from all KSAs. The complement set of each secret key is also produced by changing the j^{th} bit of K_i, that is, $\{K_i^j\}$ for all $i = 1, 2, \ldots, 10000$ and $j = 1, 2, \ldots, n$. The SK set for KSAs can be represented as

$$\mathrm{SK} = F(K_i, n, 128, 11), \tag{16}$$

where n is the length of the secret key and SK set for the complement values is

$$\mathrm{SK} = F\left(K_i^j, n, 128, 11\right). \tag{17}$$

For each type of KSA, the avalanche vectors are generated to calculate the values of d_c, $d_a$andd_{sa} as described in Section 2.2.

4.3. Bitwise-Uncorrelation Tests (BUCT). BUCT evaluates the dependency among the bits of subkeys. In BUCT, 500-random secret key set is taken. The SK set for KSAs is given as follows:

$$\mathrm{SK} = F(K_i, n, 128, 11), \tag{18}$$

where n is the secret key length and $i = 1, 2, \ldots, 500$. The 500 sequences are generated by the method described in Section 2.3. The length of each sequence is given as follows:

$$\text{binary sequence} = rC_2 \times m \times m, \quad (19)$$

where ($m = 128$ and $r = 11$).

The generated binary sequence will be tested against randomness by using four basic statistical tests, namely, frequency, runs, poker ($P = 4$), and autocorrelation ($d = 2$) [8, 30]. The threshold level α is set at 0.10; that is, 10 (90%) out of 100 sequences are expected to be rejected (accepted).

4.4. High/Low-Density Key Tests. In the previous tests, all KSAs are tested by taking the random input secret keys. A cryptographically strong KSA should also generate random subkeys when the input secret key is weak/nonrandom. For this purpose, high- and low-density keys are selected and used as secret keys in the KSAs. Four tests are selected from the NIST test suite, namely, frequency, block frequency, runs, and cusum tests. In NIST test suite, block frequency test is the parameterized test and value of block length is selected as 20. The number of the nonrandom secret keys in high and low density is 8257. The SK set of KSAs for high and low density is given as follows:

$$\text{SK} = F(K_i, n, 128, 11), \quad (20)$$

where n is the secret key length and $i = 1, 2, \ldots, 8257$. The total number of sequences in high- and low-density tests is (8257×11) 90827. The passing percentage for this number of sequences is 98% with the level of significance equal to 0.01 [26]. All parametric values required to generate data for different statistical tests have been summarized in Table 1.

5. Results and Discussion

In the following sections, the results of the statistical tests for the KSAs are discussed.

5.1. Frequency Test. Frequency test is related to the confusion property of the KSA. A balance and uniformly distributed sequence makes the statistical attacks more complex. The results of the frequency test for KSAs of AES, Serpent, PRESENT, IDEA, and Twofish are shown in Figure 1. For each round, the subkeys of KSA of AES pass the frequency test with a value greater than 98%. The results show a good balance of zero and one in each subkey for the random input secret key.

The KSA of Serpent also shows a good balance between the distribution of zero and one for the random input secret key. For each round subkey, the passing percentage is approximately equal to 99% (Figure 1).

Since KSA of PRESENT uses the rotation function, therefore, generated subkeys will also be random when the input secret key is random. Figure 1 shows that the passing percentage is approximately equal to 99% for each round subkey.

KSA of IDEA passed the frequency test with a high percentage for the subkey of each round. Since KSA of IDEA uses the 25-bit circular shift to generate subkeys, therefore, this rotation does not affect the distribution of zero and one in the subkeys. Thus the passing percentage is 99.02. The KSA of Twofish passed the frequency test which shows good balance of distribution of zeros and ones for the random input secret key.

The KSAs of IDEA and PRESENT pass frequency test (for random input secret key); even though they are known to be vulnerable to many attacks [34–37], it can be inferred that the frequency test is necessary but not sufficient for strength evaluation of KSA in case of random input secrete keys. Therefore, other tests are also required for the evaluation of KSAs.

5.2. Bit Independence Tests. The KSA of AES does not fulfill the BIT criteria since the calculated values for d_c and d_a are far away from 1 for each round. Also, the value of SAC (d_{sa}) is 0.77 even after 11th round subkey (Table 3). The results of SAC shows that key schedule algorithm of AES lacks in the confusion and diffusion property. The main problem in the KSA of AES is the relationship that the original key and the generated words have. If the value of one word is known, then the other words and secret key can be deduced by different methods. Also, KSA of AES has few nonlinear elements and slower diffusion structure to generate the subkeys. Most of the cryptanalysis attacks on AES takes the advantage of this weakness in its KSA, for example, related-key attacks, linear and differential attacks, and combined attack [5, 38]. After a little modification, the SAC value of KSA can be improved which can make AES cipher more secure [6, 9, 13].

The KSA of Serpent passes the BIT criteria as the values of $d_c = 1, d_a = 0.9999$ and $d_{\text{sa}} = 0.991989$ after the first round. From Table 3, it can be observed that the SAC value of the KSA of Serpent is approximately equal to 1 for each round subkey. The high SAC value indicates that the KSA of Serpent gives the strong immunity to cipher against statistical, related-key attacks and slide attacks.

The KSA of PRESENT fails the BIT as the value of d_c is 1 and the values of d_a are found to be far away from 1 for each round subkey. The KSA also does not pass SAC property ($d_{\text{sa}} \approx 0.50$) even for the 11th round which can also be seen from Table 3. The failure of the SAC test indicates that the KSA of PRESENT generates the nonrandom subkeys and the algorithm does not hide the relationship between subkeys and secret key completely. The weaknesses in the KSA might help the attacker to launch the cryptanalysis attacks, like related-key attacks, slide attacks, and so on, to partially break the cipher [36].

The calculated values of $d_c$andd_a are 0 for each round subkey of the KSA of IDEA. From Table 3, it can be seen that the SAC values of the KSA lie on the x-axis. Hence, the KSA of IDEA fails all tests of BIT ($d_c = 0, d_a \approx 0$and$d_{\text{sa}} = 0$) for each round subkey which indicates that the KSA does not provide good confusion and diffusion properties to subkeys. KSA of IDEA is a weak KSA and has linear relationship between the subkeys and secret key. This simple structure degrades the security of overall cipher. This weaknesses in KSA can make the IDEA cipher more vulnerable to related-key attacks, chosen-key differential attack, and linear and differential attacks [39, 40].

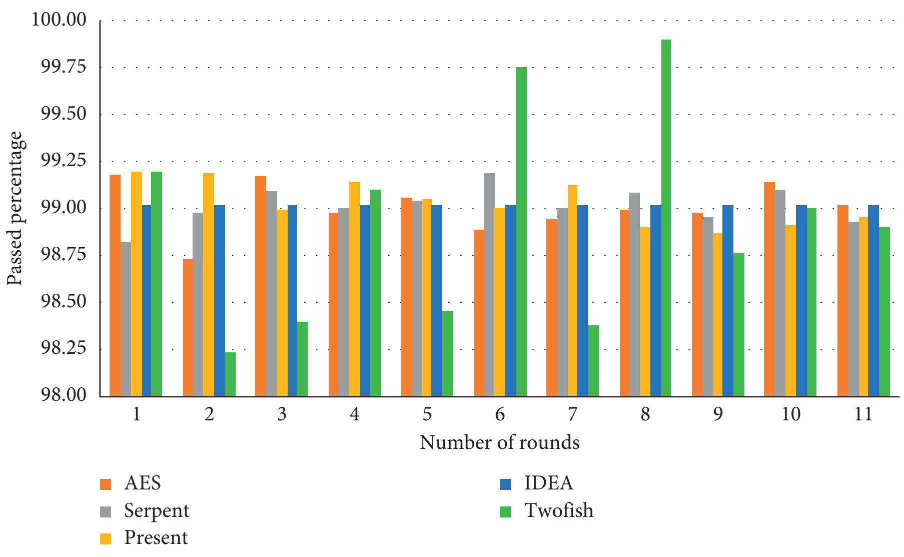

Figure 1: Frequency test for KSA.

Table 3: Degree of SAC.

Round	Degree of SAC				
	AES	Serpent	PRESENT	IDEA	Twofish
1	0.005246	0.991989	0.501471	0	0.0438
2	0.068341	0.992027	0.50092	0	0.552418
3	0.299979	0.992026	0.500904	0	0.8937
4	0.490651	0.992061	0.501116	0	0.974712
5	0.692194	0.992074	0.50234	0	0.97483
6	0.772531	0.992049	0.502879	0	0.9749
7	0.774886	0.991972	0.502932	0	0.974652
8	0.774951	0.992088	0.503098	0	0.984835
9	0.774865	0.991994	0.503074	0	0.985114
10	0.774976	0.992011	0.503908	0	0.99235
11	0.774905	0.992036	0.504398	0	0.99528

The KSA of Twofish passed the BIT criteria after the four rounds. The KSA of Twofish uses the strong linear components and nonlinear components to hide the relationship between the subkeys and secret key. This complex relationship leads the cipher strong against many cryptanalysis attacks which utilizes the avalanche weakness in KSA.

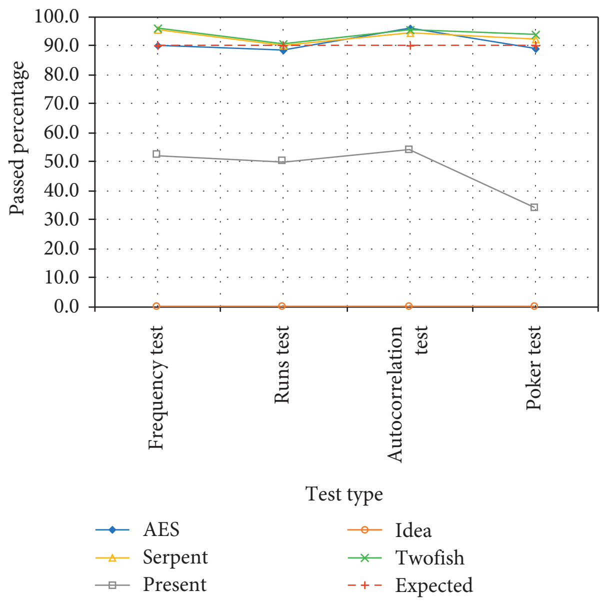

Figure 2: Bit correlation test.

5.3. Bitwise-Uncorrelation Test (BUCT). The results of the four basic tests are presented by the line graph in Figure 2 which shows that AES-KSA has a percentage value greater than or equal to 88% for all tests. In the case of Serpent-KSA, the value of the percentage for all tests is greater than or equal to 90%. It can be noticed from the graph that the percentage value of PRESENT-KSA is less than 55% for each test which is far away from the passing threshold (90%). However, the IDEA-KSA shows 0% value for all tests which indicates that all 500 sequences do not pass this test. The zero percentage result predicts the unbalance distribution of zero and one. KSAs of Twofish have good percentage (91%) for all the BUCT tests.

It can be concluded from the results that KSAs of AES and Serpent have good independence among the subkeys, whereas KSA of PRESENT has a correlation among the subkeys. On the other hand, KSA of IDEA has a strong correlation among the subkeys. KSAs of Serpent and Twofish have good independence among the subkeys which can make ciphers more immune against the cryptanalysis attacks related to KSA. KSAs of AES also have good results but are little bit lower than 90%. These results strengthen the result concluded from BIT test. On the other hand, the BUCT results show that the KSA of PRESENT and IDEA generates the subkeys that are statistically correlated. This correlation can aid an attacker to launch the cryptanalysis attacks like key-dependent attacks and related-key attacks on reduced round of PRESENT and IDEA cipher [5, 40, 41].

5.4. High/Low-Density Key Tests. A well-designed and complex KSA with good avalanche effect should generate the random subkeys when the input secret key is weak/

Table 4: Results of low/high-density keys test.

Test name	AES		Serpent		PRESENT		IDEA		Twofish	
	LDK (%)	HDK (%)	LDK (%)	HDK (%)	LDK (%)	HDK (%)	LDK (%)	HDK (%)	LDK (%)	HDK (%)
Frequency test	98.28	99.36	98.77	99.02	0.00	0.00	0.00	0.00	99.22	98.17
Block frequency test	98.20	99.47	98.25	99.22	0.00	0.00	0.00	0.00	99.81	99.67
Cusum test, forward	98.50	99.41	98.16	99.13	0.00	0.00	0.00	0.00	98.44	99.00
Cusum test, backward	98.46	99.43	99.00	99.13	0.00	0.00	0.00	0.00	1.00	98.53
Runs	98.30	98.00	98.20	99.01	0.00	0.00	0.00	0.00	97.66	1.00

Table 5: Execution time for KSAs.

Block cipher	Time (sec)
AES	8
Serpent	80
PRESENT	0.0001
IDEA	0
Twofish	128

nonrandom. To test this property of KSA, high/low-density test are included in the evaluation criteria. The results of this test are presented in Table 4 for all KSAs. From the table, it can be seen that the passing percentage values for KSA of AES and Serpent against low- and high-density keys are above 98%. It indicates that the KSA of AES and Serpent generates random subkeys for the nonrandom input secret key.

As indicated in Table 4, the results of the KSAs of PRESENT show that the passing percentage value is zero for both high- and low-density keys in the case of PRESENT; that is, no sequence passes the statistical tests.

KSA of IDEA involves only the circular shift. Circular shifts do not convert the nonrandom behavior into the random one. It can be seen from Table 4 that the KSA fails all the tests of the low and high density of input secret key. The results showed that the KSA of PRESENT and IDEA generates the nonrandom subkeys when the input secret key is nonrandom.

5.5. Time Complexity. The time of one execution for KSA can be used for comparison between KSAs. One execution of KSA implies the time taken by the KSA to generate the 11 subkeys. Each KSA is implemented in C language and the CPU speed is 3.00 GHz. Table 5 presents the comparison of execution time between the KSAs.

It can be seen from Table 5 that the KSA of Serpent and Twofish takes more time to generate the subkeys than all other KSAs. KSA of both the algorithms passed all the tests but their computational overhead is high. On the other hand, AES takes 8 seconds to generate the 11 subkeys. The execution time for KSA of PRESENT is 0.0001 seconds to generate 11 subkeys, whereas IDEA-KSA takes 0 seconds to generate the same number of subkeys.

6. Conclusion

A new Key Schedule Evaluation Criterion (KSEC) has been proposed to evaluate the cryptographic strength of any KSA. The KSEC is based upon five tests, namely, (1) frequency, (2) bit independency, (3) bitwise uncorrelation, (4) high/low-density key, and (5) time complexity. It evaluates the confusion, diffusion, independence, and randomness properties of a KSA through the proposed statistical tests. The selected KSAs are evaluated against the KSEC. The tests results show that the KSAs can be differentiated as weak or strong KSA. It is also evaluated that the KSAs of PRESENT and IDEA (which use simple shift operations) fail to meet the KSEC, which indicates that the subkeys do not exhibit randomness and bit independence properties. Moreover, the KSA of AES does not pass the SAC test of BIT, which infers that the subkeys of AES are correlated. On the other hand, the KSA of Serpent and Twofish passed all the tests; however, their computational overhead is high.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] C. Li, F. Zhao, C. Liu, L. Lei, and J. Zhang, "A hyperchaotic color image encryption algorithm and security analysis," *Security and Communication Networks*, vol. 2019, Article ID 8132547, 8 pages, 2019.

[2] A. Biryukov and D. Khovratovich, "Related-key cryptanalysis of the full AES-192 and AES-256," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, pp. 1–18, Berlin, Heidelberg, August 2009.

[3] U. Blumenthal and S. M. Bellovin, "A better key schedule for DES-like ciphers," in *Proceedings of the Pragocrypt*, Prague, Czech Republic, September 1996.

[4] L. R. Knudsen and J. E. Mathiassen, "On the role of key schedules in attacks on iterated ciphers," in *European Symposium on Research in Computer Security*, pp. 322–334, Springer, Berlin, Heidelberg, 2004.

[5] L. R. Knudsen, "Practically secure Feistel ciphers," in *International Workshop on Fast Software Encryption*, pp. 211–221, Springer, Berlin, Heidelberg, 1993.

[6] J. Kelsey, S. Bruce, and W. David, "Key-schedule cryptanalysis of IDEA, G-DES, GOST, SAFER, and TRIPLE-DES," in *Advances in Cryptology—CRYPTO '96*, pp. 237–251, Springer, Berlin, Heidelberg, 1996.

[7] L. May, M. Henricksen, W. Millan, G. Carter, and E. Dawson, "Strengthening the key schedule of the AES," in *Information*

Security and Privacy, pp. 226–240, Springer, Berlin Heidelberg, 2002.

[8] S. Afzal, U. Waqas, A. M. Mubeen, and M. Yousaf, "Statistical analysis of key schedule algorithms of different block ciphers," *Science International*, vol. 27, no. 3, 2015.

[9] M. K. Pehlivanoğlu, M. T. Sakalli, N. Duru, and F. B. Sakalli, "The new approach of AES key schedule for lightweight block ciphers," *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 19, no. 3, pp. 21–26, 2017.

[10] J. Daemen, G. René, and V. Joos, "Weak keys for IDEA," in *Annual International Cryptology Conference*, pp. 224–231, Springer, Berlin, Heidelberg, 1993.

[11] A. Bogdanov, L. R. Knudsen, G. Leander et al., "PRESENT: an ultra-lightweight block cipher," in *Cryptographic Hardware and Embedded Systems—CHES 2007*, pp. 450–466, Springer, Berlin, Heidelberg, 2007.

[12] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson, "Twofish: a 128-bit block cipher," *NIST AES Proposal*, vol. 15, no. 1, pp. 23–91, 1998.

[13] J. Daemen and V. Rijmen, "The design of Rijndael," in *AES-The Advanced Encryption Standard Process*, Springer Science & Business Media, Berlin, Germany, 2013.

[14] J. Huang, Y. Hailun, and L. Xuejia, "Transposition of AES key schedule," in *International Conference on Information Security and Cryptology*, pp. 84–102, Springer, Cham, Switzerland, 2017.

[15] S. Sulaiman, Z. Muda, J. Juremi, R. Mahmod, and S. M. Yasin, "A new shift column transformation: an enhancement of Rijndael key scheduling," *International Journal of Cyber-Security and Digital Forensics (IJCSDF)*, vol. 1, no. 3, pp. 160–166, 2012.

[16] A. Ross, B. Eli, and K. Lars, "Serpent: a proposal for the advanced encryption standard," in *First Advanced Encryption Standard (AES) Conference*, Association for Computing Machinery (ACM), Ventura, CA, USA, 1998.

[17] S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the key scheduling algorithm of RC4," in *International Workshop on Selected Areas in Cryptography*, pp. 1–24, Springer, Berlin, Heidelberg, 2001.

[18] Y. Harmouch and R. El Kouch, "The benefit of using chaos in key schedule algorithm," *Journal of Information Security and Applications*, vol. 45, pp. 143–155, 2019.

[19] R. E. J. Paje, A. M. Sison, and R. P. Medina, "Multidimensional key RC6 algorithm," in *Proceedings of the 3rd International Conference on Cryptography, Security and Privacy—ICCSP'19*, pp. 33–38, Kuala Lumpur, Malaysia, January 2019.

[20] E. Simion, "The relevance of statistical tests in cryptography," *IEEE Security & Privacy*, vol. 13, no. 1, pp. 66–70, 2015.

[21] C. H. Kim, "Improved differential fault analysis on AES key schedule," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 41–50, 2011.

[22] M. Ukrop, *Randomness analysis in authenticated encryption systems*, Ph.D. thesis, Masarykovauniverzita, Fakultainformatiky, Brno, Czechia, 2016.

[23] M. Sýs, D. Klinec, K. Kubíček, and P. Švenda, "BoolTest: the fast randomness testing strategy based on boolean functions with application to DES, 3-DES, MD5, MD6, and SHA-256," in *International Conference on E-Business and Telecommunications, 2017*, pp. 123–149, Springer, Cham, Switzerland, 2017.

[24] U. Waqas, S. Afzal, M. A. Mir, and M. Yosuf, "Generation of AES-like S-boxes by replacing affine matrix," in *Proceedings of the 2014, IEEE 12th International Conference on Frontiers of Information Technology*, pp. 159–164, Islamabad, Pakistan, December 2014.

[25] C. Swenson, *Modern Cryptanalysis: Techniques for Advanced Code Breaking*, pp. 150–165, John Wiley & Sons, Hoboken, NY, USA, 2008.

[26] A. Rukhin, J. Sota, J. Nechvatal, et al., A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic applications. NIST SP800-20-Rev1a, 2010.

[27] U. M. Maurer, "A universal statistical test for random bit generators," *Journal of Cryptology*, vol. 5, no. 2, pp. 89–105, 1992.

[28] B. Preneel, "NESSIE project," in *Encyclopedia of Cryptography and Security*, Springer, Berlin, Germany, 2000.

[29] C. M. Adams, "Simple and effective key scheduling for symmetric ciphers," in *Proceedings of the Workshop Record of the Workshop on Selected Areas in Cryptography (SAC 94)*, pp. 129–133, Kingston, Canada, May 1994.

[30] C.-L. Duta, B.-C. Mocanu, F.-A. Vladescu, and L. Gheorghe, "Randomness evaluation framework of cryptographic algorithms," *International Journal on Cryptography and Information Security*, vol. 4, no. 1, pp. 31–49, 2014.

[31] Computer Security Resource Center, *Update to Current Use and Deprecation of TDEA*, NIST, Gaithersburg, MA, USA, 2017, https://csrc.nist.gov/News/2017/Update-to-Current-Use-and-Deprecation-of-TDEA.

[32] Microsoft, *Technical Reference Details about Encryption in Office 365*, Microsoft, Redmond, WA, USA, 2019, https://docs.microsoft.com/en-us/microsoft-365/compliance/technical-reference-details-about-encryption.

[33] D. Rane Deepali, "Superiority of twofish over blowfish," *International Journal of Scientific Research and Management*, vol. 4, no. 11, pp. 4744–4746, 2016.

[34] E. Biham, O. Dunkelman, and N. Keller, "A new attack on 6-round IDEA," in *Fast Software Encryption*, pp. 211–224, Springer, Berlin, Heidelberg, 2007.

[35] J. C. Hernandez-Castro, P. Peris-Lopez, and J. P. Aumasson, "On the key schedule strength of present," in *Data Privacy Management and Autonomous Spontaneus Security*, pp. 253–263, Springer, Berlin, Heidelberg, 2012.

[36] J. Huang, S. Vaudenay, and X. Lai, "On the key schedule of lightweight block ciphers," in *Progress in Cryptology—INDOCRYPT 2014*, pp. 124–142, Springer, Cham, Switzerland, 2014.

[37] O. Özen, K. Varıcı, C. Tezcan, and Ç. Kocair, "Lightweight block ciphers revisited: cryptanalysis of reduced round PRESENT and HIGHT," in *Australasian Conference on Information Security and Privacy*, pp. 90–107, Springer, Berlin, Heidelberg, 2009.

[38] R. Saha, G. Geetha, G. Kumar, and T. H. Kim, "RK-AES: an improved version of AES using a new key generation process with random keys," *Wiley Security and Communication Networks*, vol. 2018, Article ID 9802475, 11 pages, 2018.

[39] M. Rafighi and N. Moatazedi, "Optimization of IDEA key-schedule algorithm for safe use in cloud," *International Journal of Engineering and Technology*, vol. 9, no. 2, pp. 902–915, 2017.

[40] E. Biham, O. Dunkelman, and N. Keller, "New cryptanalytic results on IDEA," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 412–427, Springer, Berlin, Heidelberg, 2006.

[41] B. Song, *Observations on the cryptologic properties of the AES algorithm*, Ph.D. thesis, University of Wollongong, Wollongong, Australia, 2004.