symmetry

MDPI

Article

An Enhanced Key Schedule Algorithm of PRESENT-128 Block Cipher for Random and Non-Random Secret Keys

Maria Imdad [1,†], Sofia Najwa Ramli [1,*,†] and Hairulnizam Mahdin [2,†]

[1] Center of Information Security Research, Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, Batu Pahat 86400, Malaysia; maria.imdad123@gmail.com
[2] Center of Intelligence and Autonomous Systems, Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, Batu Pahat 86400, Malaysia; hairuln@uthm.edu.my
* Correspondence: sofianajwa@uthm.edu.my
† These authors contributed equally to this work.

Abstract: The key schedule algorithm (KSA) is a crucial element of symmetric block ciphers with a direct security impact. Despite its undeniable significance, the KSA is still a less focused area in the design of an encryption algorithm. PRESENT is a symmetric lightweight block cipher that provides the optimal balance between security, performance, and minimal cost in IoT. However, the linear functions in KSA lead to a slow and predictable bit transition, indicating the relationship between round keys. A robust KSA should produce random and independent round keys irrespective of the secret key. Therefore, this research aims to improve the KSA PRESENT-128 block cipher with enhanced randomness, round key bit difference, and the avalanche effect. The experiments on round keys and ciphertext with random, low density and high-density secret key datasets endorse the expected improvements. Moreover, the results show that the improved KSA produces random round keys that successfully pass the NIST randomness test. The bit transition from one round key to another is increased from 20% to 40%, where a greater inclination of the avalanche effect has an increased effect with 50% bit change. On the other hand, the improved KSA PRESENT requires an additional 0.001871 s to generate round keys, as a security cost trade-off.

Keywords: block cipher; cryptography; gate equivalence; key schedule algorithm; key sensitivity; plaintext sensitivity; PRESENT; randomness; symmetric cryptography

Citation: Imdad, M.; Ramli, S.N.; Mahdin, H. An Enhanced Key Schedule Algorithm of PRESENT-128 Block Cipher for Random and Non-Random Secret Keys. *Symmetry* **2022**, *14*, 604. https://doi.org/10.3390/sym14030604

Academic Editor: Kuo-Hui Yeh

Received: 7 February 2022
Accepted: 1 March 2022
Published: 18 March 2022

1. Introduction

The security of a symmetric block cipher depends on the strength of the encryption algorithm and the secret key. As encryption algorithms are publicly available, the security of a cipher resides in the strength and length of the secret key [1]. According to NIST [2–4], a minimum key length to resist brute force attacks is 112 bits. In symmetric block ciphers, the mathematical functions in encryption and the key schedule algorithm (KSA) directly influence the security of the encrypted data [5,6]. Therefore, these functions are carefully selected and combined to produce a strong ciphertext [7,8].

The secret key uses the KSA to generate the round keys for the encryption/decryption process in the symmetric block cipher. These round keys are directly XOR'ed with the cipher state (the encrypted data from the previous round) to conceal a particular round's input and output. So, to produce random round data, the round keys should be random and independent of each other. Keyspace for a 128-bit key is quite extensive, and it is practically impossible to test all secret key combinations during the algorithm design and testing. Still, there is a fair chance that the researcher's intentional investigation [9] or accidental discovery may lead to identifying such secret keys, making the cipher behave unexpectedly [10]. These secret keys are weak, semi-weak, and equivalent keys. Hence the security design of a KSA is equally important as the encryption [7] so that there are no weak or equivalent keys.

An algorithm with a hardware-oriented design incurring minimum possible delay is considered ideal in an IoT environment [11,12]. In the past few years, many lightweight encryption algorithms have been proposed for resource-constrained devices to provide optimum security and good speed; these algorithms include SKINNY [13], Loong [14], SFN [15], TEA [16], SIT [17], RECTANGLE [18] and PRESENT [19]. Many encryption algorithms outperform high-end devices, but their performance degrades when used in resource-constrained IoT devices [20–25], and a similar case was observed for AES [26]. The lightweight encryption solutions are designed with minimal gates in hardware, and out of the total 1000–10,000 GE on the radio frequency identification (RFID) tag [27,28], only 250–4000 GEs are dedicated for encryption algorithms. The PRESENT block cipher has a good balance between security, cost, and performance triangle for lightweight encryption algorithms, compared to other competitive solutions [29]. Consequently, PRESENT is one of the most widely used symmetric block ciphers aimed at resource-constrained devices, such as sensors and RFID tags [19,23], acquiring only 1570 and 1886 GEs for 80-bit and 128-bit versions, respectively [19].

Although PRESENT [19] performs very well in IoT, its limited focus on security is worth examining. One of the significant issues in PRESENT is its KSA, as highlighted by different researchers [20–22]. The KSA uses a combination of linear and nonlinear functions. A recent study [4] highlighted that PRESENT KSA fails the National Institute of Standards and Technology (NIST) statistical test on round keys for non-random secret keys. This is primarily because these non-linear functions in the "substitution box" (S-box) are applied to 8 bits, followed by XOR on only 5 bits, for each round key's generation, leading to a prolonged transition in round keys. Moreover, they also proved that the round keys pass the frequency test when the secret key is random because the shift operation changes the bit position, resulting in a new binary string as a round key. A similar shift operation results in no bit change for a non-random key with all zeros. Such keys result in a slow and predictable transition among round keys and thus reveal information about other round keys and secret keys to the attacker.

PRESENT was improved by many researchers [24,25]. Still, there is no clear evidence about improving its KSA and analyzing its effect on encrypted data, and this simple KSA is not specific to PRESENT only. The KSA of International Data Encryption Algorithm (IDEA) [30] has only one function of the circular shift, which makes it ideal for random keys but not ideal for non-random keys. Even the advanced encryption standard (AES) is vulnerable to cryptanalytic attacks because of KSA [31]. These KSA vulnerabilities have led to a new research area to overcome KSA's limitations to generate random and independent round keys.

A considerable amount of literature has been published on improving KSA for different encryption algorithms. The work in [32] improved the KSA of AES to optimize the number of active s-boxes; later, HM Hussain [31] enhanced the KSA to tolerate cryptanalytic attacks. Recently, the work in [33] has highlighted the concept of slow diffusion in the AES key schedule in the initial round. The proposed improvement has additional key permutations for better diffusion, validated using the avalanche effect and frequency test. The KSA of AES was also improved by Shivani [34] by introducing three secret keys instead of one to increase the keyspace and overall system security. In addition, a highly efficient and independent KSA has been introduced, which uses chaos to generate round keys [35]. This research indicates that randomness, avalanche effect, the difference between round keys, and linear cryptanalysis are important security measures. This independent KSA demonstrated better performance when compared with existing state-of-the-art solutions. In [36], an enhanced KSA was introduced using the triple transposition key process for the GOST encryption algorithm. Elbert M. Galas in [37] enhanced the KSA of XXTEA by using a random value called seed to enhance the confusion and diffusion among the round keys. Later, the KSA of the RECTANGLE algorithm was enhanced by authors in [38] to increase the randomness among the round keys. Moreover, they managed to keep the design simple to constrain the algorithmic complexity and KSA time.

The above-mentioned recent studies endorse the fact that security advancements are generating more attention in KSA, as the improvements have significantly elevated the security of the overall algorithm. This study focuses on the design of the KSA PRESENT. It improves the design to enhance the security of the overall cryptosystem while retaining the lightweight structure by keeping the 3000 GEs limit in consideration. Moreover, a fair comparison between the KSA PRESENT and improved KSA is performed for round keys, ciphertext and cost to compare the security improvements introduced. Keeping the security, cost and performance triangle in mind, the improved KSA is also compared against time.

The rest of this paper is organized as follows: Section 2 contains a theoretical explanation of the PRESENT encryption, and the KSA PRESENT in detail. Section 3 gives a detailed insight into the design and rationale for the improvements in the KSA PRESENT. Section 4 summarizes the data generation to compare the KSA PRESENT and the improved KSA, followed by a detailed discussion and results analysis in Section 5. Section 6 concludes this study and provides some future directions for interested researchers.

2. PRESENT Block Cipher

This section emphasizes the lightweight structure of the PRESENT block cipher [19]. The block size is 64 bits, whereas the key size has two variants with 80 and 128 bits referred to as PRESENT-80 and PRESENT-128, respectively. It has a traditional substitution permutation network (SPN) structure, making it suitable for a resource-constrained environment [12,23,39–41]. This research aims to extend the key schedule of PRESENT-128 only, because the minimum key length to resist a brute force attack is 112 bits [4].

2.1. PRESENT Encryption

PRESENT runs 31 rounds of encryption to convert the plaintext into ciphertext. Each round comprises three basic operations, given in Algorithm 1 as pseudocode (C language), starting with AddRoundKey; the first 64 bits of the round key are XOR'ed with the plaintext block. The second operation is sBoxLayer, where the 64-bit output from the previous operation undergoes non-linear transformations based on the substitution box, as presented in Table 1. This substitution box has a non-linear bijective mapping, where $S : F_2^4 \rightarrow F_2^4$ is followed by a permutation operation (pLayer) with a bit-by-bit permutation with $P : F_2^{64} \rightarrow F_2^{64}$. The substitution and permutation layers were inspired by SERPENT [42] and DES [38], at the design stage. There are 31 rounds of SPN with an extra layer as round 32, having only addRoundKey using the 31st round key to complete the encryption process. The basic encryption of PRESENT is depicted in Figure 1 below.

Algorithm 1 Pseudocode of PRESENT block cipher.

```
generateRoundKeys()
for i = 1 to 31 do
    addRoundKey(STATE, K_i)
    sBoxLayer(STATE)
    pLayer(STATE)
end for
addRoundKey(STATE, K_32)
```

Table 1. Substitution box PRESENT.

X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S(X)	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

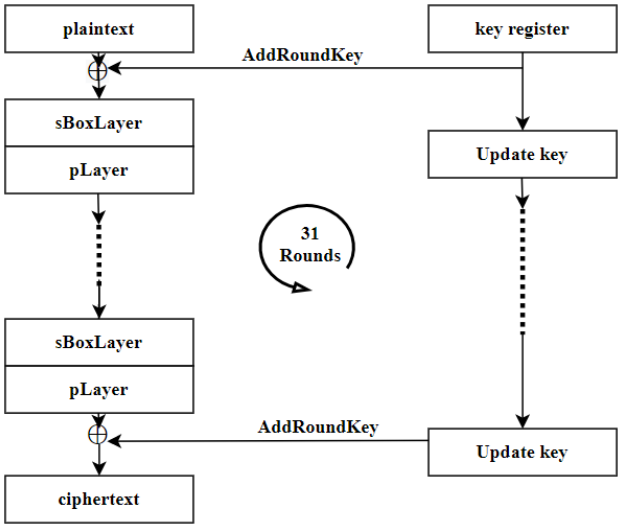


Figure 1. PRESENT block cipher encryption process.

2.2. The KSA PRESENT

The key schedule algorithm for PRESENT can operate on 80 and 128 bits, but for illustration purposes, this research focuses on PRESENT-128. Moreover, the minimum string requirement or input size recommended for performing the desired NIST statistical tests is also 100 bits ($n \geq 100$ bits), which is convenient for PRESENT-128 [43]. At the beginning of the process, a 128-bits secret key is stored in the key register, $K = k_{127}k_{126} \ldots k_0$. At the 1st round, the 64-bits round key k_1 comprises the leftmost 64-bit of the secret key in key register K. The steps of the KSA PRESENT are given below and repeated 31 times to generate 31 different round keys as depicted in Figure 2.

Step 1: Store the secret key to the key register, $K = k_{127}k_{126} \ldots k_0$

Step 2: Apply 61 bits shift to the left of K, where $K = [k_{66}k_{65} \ldots k_{127}, k_{126} \ldots k_{67}]$

Step 3: Substitute the leftmost four bits of K using S-box, where $K[k_{127}k_{126}k_{125}k_{124}] = S - box[k_{127}k_{126}k_{125}k_{124}]$

Step 4: Substitute the next leftmost four bits of K using S-box, where: $K[k_{123}k_{122}k_{121}k_{120}] = S - box[k_{123}k_{122}k_{121}k_{120}]$

Step 5: XOR five bits of with the least significant bit of round counter i, where $K[k_{66}k_{65}k_{64}k_{63}k_{62}] = [k_{66}k_{65}k_{64}k_{63}k_{62}] \oplus i$

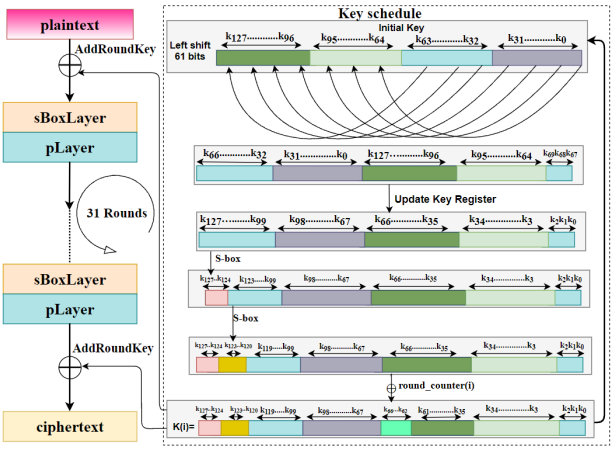


Figure 2. Detailed key schedule of PRESENT block cipher.

The following round key, k_i, consists of the leftmost 64-bits of the updated key register, K. The steps are repeated until 31 round keys are produced for the addRoundKey operation

in the PRESENT encryption. The KSA PRESENT is primarily designed to have a left shift, 8-bits substitution, and an XOR operation with the least significant bit of the round counter. This simple design is exceptionally considered for the security of low-power devices so that any extra computations are not needed, but this simple design has certain limitations, as given below:

1. No robust transition is observed in round keys starting from round 1. This is primarily because the left shift (linear function) is applied on 47% bits and only changes bit position. The nonlinear functions s-box and XOR modify 13 bits in each round; this difference is equivalent to 10% of all the bits.
2. If one bit is toggled in a secret key, the bit difference is not visible to all generated round keys. In a worst-case scenario [41], the one-bit difference between two secret keys contributes to only a 16-bit difference among 31 round keys. This is not the required avalanche effect among round keys. Therefore, an attacker can calculate the statistical dependence between round keys.
3. For a non-random secret key, the KSA PRESENT takes 16 rounds (on average) to reach a perfect random round key.

3. The Improved KSA PRESENT

This section discusses the proposed design of KSA PRESENT, as this design aims to overcome the weaknesses mentioned in Section 2.2. The main idea while designing the improved KSA is to maintain the lightweight structure of the PRESENT without weighing down the KSA with computationally expensive operations. Thus, the proposed KSA can work in coherence with the existing encryption algorithm without increasing complexity. The improved KSA design aims to improve the randomness between the round keys as well as the ciphertext.

The Proposed KSA

The improved KSA PRESENT has an initial step similar to the original KSA PRESENT. A complete diagram of the improvements is presented in Figure 3. A key register, K stores the 128-bit secret key, and the leftmost 64-bit in the current key register are extracted as the round key k_1. Later, the steps are followed to generate 31 round keys as given below:

Step 1: Store the secret key to a key register, $K = k_{127}k_{126}\ldots k_0$

Step 2: Split the key K into four blocks with 32 bits each:

1. $x_1 = [k_{127}\ldots k_{96}] \Rightarrow J_1 = [x_1 \oplus \phi]$
2. $x_2 = [k_{95}\ldots k_{64}] \Rightarrow J_2 = S - box[x_2]$
3. $x_3 = [k_{63}\ldots k_{32}] \Rightarrow J_3 = reverse(x_3 \oplus J_2)$
4. $x_4 = [k_{31}\ldots k_0] \Rightarrow J_4 = reverse(x_4 \oplus J_1)$

Step 3: Concatenate $K = J_1J_2J_3J_4$

Step 4: Apply 61 bits shift to the left of

1. $K = [k_{66}k_{65}\ldots k_{127}k_{126}\ldots k_{67}]$
2. Update the key register

Step 5: XOR five bits of K with the least significant bit of round counter i $K[k_{66}k_{65}k_{64}k_{63}k_{62}] = [k_{66}k_{65}k_{64}k_{63}k_{62}] \oplus i$

After all these steps, extract the leftmost 64-bits round key, k_i from K and use it in the AddRoundKey function, followed by an increment in the round counter i.

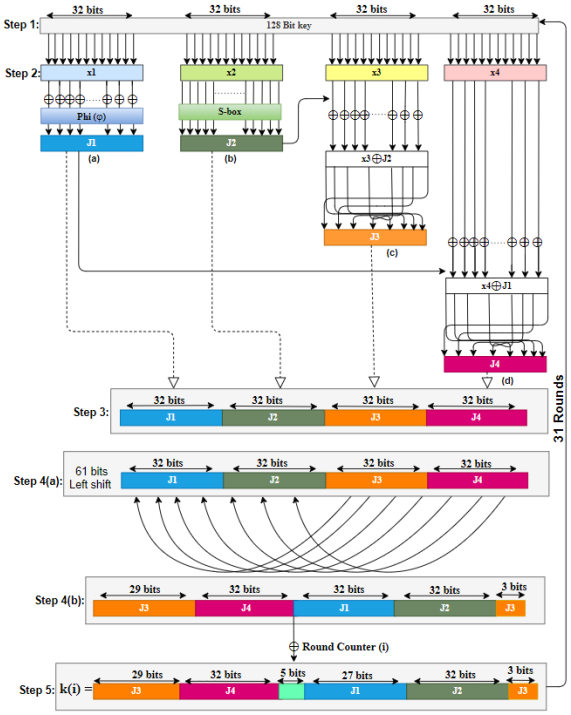


Figure 3. Detailed design of improved KSA PRESENT.

4. Key Schedule Evaluation

Breaking the encryption algorithm by launching a dictionary attack is time consuming and requires a lot of resources. Thus, attackers are trying a different way by gaining access to round keys or secret keys. Therefore, it is desirable to strengthen the KSA and the encryption algorithm to ensure security [41]. A strong KSA must generate random and independent round keys because the dependent round keys lead to different attacks, such as slide attacks, related key/round keys attack, meet-in-the-middle (MITM) attacks, and linear and differential cryptanalyses [20,44,45].

This section provides insight into the evaluation metrics, designed to validate the expected enhancements in the improved KSA and the original version. The KSA is evaluated from three perspectives—round keys, the ciphertext, and cost as depicted in Figure 4—to evaluate the security improvements introduced from the improved KSA. Three binary datasets that are in common for most of these tests are referred to as HD (high density), LD (low density), and R (random) [6,46]. An HD has all ones and only two zeros at most, and LD is the exact reverse of HD with all zeros and at most two ones. HD and LD are considered non-random datasets or biased datasets. The random dataset is created, using the "rand" function in Matlab. The secret key and the round key length is 128 bits, so these datasets will be 128 bits long and referred to as HDK, LDK and RK. The length of the plaintext is 64 bits, so each value from the datasets is 64 bits long and referred to as HDP, LDP and RP. The round keys and ciphertext will evaluate the algorithm from software security perspectives, whereas the cost is the exact gate requirement in the hardware environment. Each test is briefly described in this section, followed by the results and findings in the subsequent section.

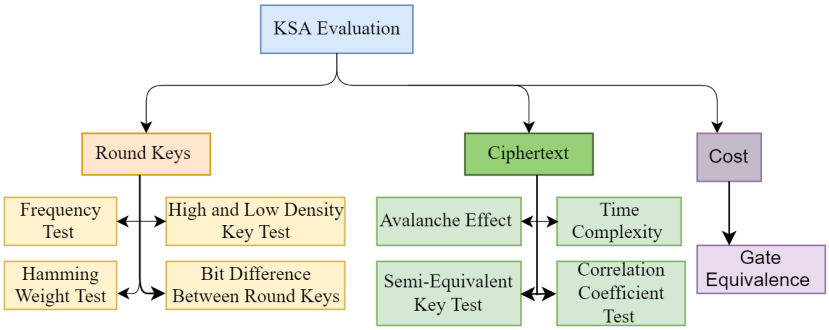


Figure 4. Key schedule algorithm (KSA) evaluation from round key, ciphertext and cost perspectives.

4.1. Round Key Evaluation

A secret key is a key used to generate the round keys for subsequent rounds. In this section, four tests are conducted to evaluate the randomness between round keys. The details of each test, subtests, the property to be evaluated, the number of secret keys, and the significance level are summarized in Table 2.

Table 2. Test, sub-tests, property to be tested, secret keys as input, output bits, and significance level of each test.

Test Name	Property to Be Tested	No. of Secret Keys	Output Bits	Significance Level
Frequency Test	Equal proportion of 0's and 1's	10,000 random	14,080,000 bits	$\alpha = 0.01$
High-Low Density test (Frequency, Block Frequency, Cusum, and Runs)	Randomness of round keys	5000 HDK, 5000 LDK	6,400,000 bits for HDK, 6,400,000 bits for LDK	$\alpha = 0.01$
Bit Difference between Round Keys	The number of bit differences among consecutive round keys	33 HDK, 33 LDK, and 34 Random	4224 bits for HDK, 4224 bits for LDK, 4352 bits for Random	An average of 50% of the bit difference among round keys when one bit is flipped
Hamming Weight Test	The population count of a bit string	1 key	2048 bits	Hamming weight 50% or nearly equal to it.

4.1.1. Frequency Test

For a given random and non-random dataset of keys, there must be an equal proportion of ones and zeros in the round keys being produced [47]. Thus, this test is a basic NIST test to prove randomness in round keys [43]. If any KSA fails to produce random round keys, there is no need to apply any further tests, as the requirements become stricter in the upcoming tests, and the algorithm will not pass them. The evaluation can be expressed using (1)–(3).

$$p\text{-value} = erfc\left(\frac{S_{obs}}{\sqrt{2}}\right) \tag{1}$$

where,

$$S_{obs} = \frac{|S_n|}{\sqrt{n}} \tag{2}$$

$$S_n = X_1 + X_2 + \ldots X_n \tag{3}$$

Here, n represents the length of the input bit string. S_n is the absolute value of the sequence being observed, and S_{obs} is the absolute value divided by the square root of the length of the string. Moreover, $erfc(.)$ is a complementary error function explained extensively in [39] based on the above equations. For evaluation purposes, both KSAs are subjected to 10,000 random secret keys.

This frequency test observes the computed p-value from the input sequence of bits. As per the decision rule, if p-value ≥ 0.01, then the sequence is concluded as being random with a confidence of 99%; otherwise, it is non-random.

4.1.2. High-Density and Low-Density Key

A well-designed KSA needs to generate a random bit string of the round key, even if the secret key is HDK or LDK [6,46]. Two datasets of LDK and HDK are created, each with 5000 keys.

This subsection discusses the NIST tests used to test the round key's randomness using the above mentioned non-random secret key datasets (HDK, LDK) as the input sequences. There are four tests: frequency, block frequency, cumulative sums (cusum), and runs, as depicted in Table 2. The frequency test is described in Section 4.1.1, whereas the other three tests are explained here.

- Block Frequency Test: This test computes the proportion of zeros and ones in the M-bit block and determines whether the proportion of ones is approximately $M/2$. The p-value ≤ 0.01 indicates a significant deviation from the proportion of zeros and ones in at least one of the blocks and indicates a non-random sequence. The evaluation can be expressed using (4)–(7).

$$p\text{-value} = igmac\left(\frac{N}{2}, \frac{\chi^2}{2}\right) \tag{4}$$

where,

$$N = \left(\frac{n}{M}\right) \tag{5}$$

$$\chi^2 = 4M \sum_{i=1}^{N} \left(\pi_i - \frac{1}{2}\right)^2 \tag{6}$$

$$\pi_i = \frac{\sum_{i=1}^{N} \epsilon_{(i-1)M+j}}{M} \tag{7}$$

Here, M is the length of each block, which can be any value greater than 19, n is the length of the binary string, and $igmac(.)$ is the incomplete gamma function [43]. The π_i value is used in the calculation of χ^2 for $1 \leq i \leq N$.

- Cusum test: The cusum test is a random walk test where the maximum excursion (from zero) is defined by the cumulative sum of the adjusted $(-1, +1)$ digits in the sequence. The test is performed in two modes, 0 (forward) and 1 (reverse). For a random sequence, the p-value for both modes must be more than or equal to 0.01 for 99% confidence. Equation (8) calculates the p-value for modes 0 and 1, where n is the binary string length, and ϕ is the standard normal cumulative probability distribution function. Meanwhile, z is the largest excursion of the cumulative sum in the $(-1, +1)$ sequence with $z = max(1 \leq k \leq n|S_k|)$ and S_k is the largest value of partial sums.

$$\begin{aligned} p\text{-value} =& 1 - \sum_{k=(-n/z+1)/4}^{(-n/z+1)/4} \left[\phi\left(\frac{(4k+1)Z}{\sqrt{n}}\right) - \phi\left(\frac{(4k-1)Z}{\sqrt{n}}\right)\right] + \ldots \\ & \sum_{k=(-n/z-3)/4}^{(-n/z+1)/4} \left[\phi\left(\frac{(4k+3)Z}{\sqrt{n}}\right) - \phi\left(\frac{(4k-3)Z}{\sqrt{n}}\right)\right] \end{aligned} \tag{8}$$

- Runs test: The runs test is performed to identify the total number of runs in sequence, and these runs are the uninterrupted sequence of identical bits. Ideally, there should be diverse sequences of zeros and ones of variable length for a string to be finally declared as random. As with previous tests, if the p-value ≤ 0.01, it is random; otherwise it is non-random.Under this test, p-value is computed using (9), where n is the length of the bit string, $V_{n(obs)}$ is the total number of runs, and π is the proportion of ones in the sequence.

$$p\text{-value} = erfc\left(\frac{\left|V_{n(obs)} - 2n\pi(1-\pi)\right|}{2\sqrt{2n}\pi(1-\pi)}\right) \tag{9}$$

4.1.3. Bit Differences between Round Keys

This test is inspired by [6] and is aimed to observe the complex relationship between round keys. This complexity will lead to the evaluation of confusion in round keys. Two consecutive round keys will be XOR'ed and the total number of ones will be counted for comparison purposes. The improved KSA expects the bit difference of 50% and more to have a good confusion followed by a complex relationship between secret key and round keys.

4.1.4. Hamming Weight Test

Julio Cesar Hernandez-Castro [41] performed a probabilistic meta-heuristic search to identify the set of round keys with very low Hamming weights. These keys are ideal to launch a differential cryptanalysis attack on an encryption algorithm. Round keys are directly XOR'ed with round data, and to bring significant changes in this round data, round keys should have enough randomness. Moreover, a higher number of zeros or ones in round keys makes subsequent keys predictable by bringing minor changes in the round data. Hence, the Hamming weight test ensures that a perfect balance of zeros and ones is achieved between round keys for maximum randomness in the ciphertext.

Consequently, an ideally balanced binary string with n bits should have a Hamming weight of $n/2$ [41].

In the PRESENT block cipher, 64 leftmost bits are used for each round's encryption process, and the rest of the bits are ignored. Thus, the calculation for the Hamming weight is applied only to these 64 bits, as these directly participate in the encryption process. As the value of n is 64, each string's ideal Hamming weight value is $n/2 = 32$. Researchers in [41] presented a specific PRESENT-128 secret key, 0x484a04d32c22f3ae28200190103481f3, yielding a cumulative Hamming weight of 433 in 31 round keys. We have 31 round keys and 1 secret key, with a total of 32 keys in the KSA PRESENT. Therefore, the Hamming weight value should be 32 rounds $\times$ 32 bits $=$ 1024 bits, rightly equivalent to 50% of the total bits. The same key is used as a secret key for both the original and improved KSAs, and the Hamming weight of each of the round keys is calculated using (10).

$$\text{Hamming Weight} = \frac{\text{Number of non-zero bits}}{\text{Total bits}} \tag{10}$$

4.2. Ciphertext Evaluation

This section gives a detailed insight into the methods to evaluate the ciphertext produced from the KSA and improved KSA PRESENT. These are used to evaluate the strength of the improved KSA PRESENT so that it can be considered a replacement for the existing KSA. Four experiments are performed on ciphertext to evaluate it in terms of the avalanche effect, the correlation coefficient, semi-equivalent key, and time complexity. Table 3 shows the tests, their respective sub-tests on key and plaintext, and the total number of datasets for the secret key, plaintext, and ciphertext bits. This test helps us evaluate the impact of KSA's improvements on the ciphertext.

Table 3. Test, sub-tests, number of secret keys as input, number of plaintexts, and output bits.

Test Name	Sub-Tests	No. of Secret Keys	No. of Plaintext	Output Bits
Avalanche Effect	Key	128 Random	1 Random	$128 \times 64 = 8192$ bits
		128 HDK	1 Random	$128 \times 64 = 8192$ bits
		128 LDK	1 Random	$128 \times 64 = 8192$ bits
	Plaintext	1 Random	64 Random	$64 \times 64 = 4096$ bits
		1 Random	64 HDP	$64 \times 64 = 4096$ bits
		1 Random	64 LDP	$64 \times 64 = 4096$ bits
Correlation Coefficient Test	Key	5 RK	100 Random	$500 \times 64 = 32{,}000$ bits
		5 HDK	100 Random	$500 \times 64 = 32{,}000$ bits
		5 LDK	100 Random	$500 \times 64 = 32{,}000$ bits
Semi-Equivalent Key test		2 Secret Keys	1 HDP	128 bits
Time Complexity		100 RK	100 Random	Average time for 100 KSA

4.2.1. Avalanche Effect (AE)

The avalanche effect (AE) measures the effect of change in the ciphertext by changing one bit in the associated plaintext or the key. The total number of ciphertext bits affected by the modification in plaintext is called the avalanche effect. This test measures the non-linear characteristics of the proposed algorithm. The value for a good avalanche effect should be 50% of the bits, ensuring that each ciphertext bit is affected by the plaintext or the key bit [48–50].

The formula used to calculate AE is adopted from [51] and is defined in (11). Here, x is the length of the plaintext/ciphertext, which is 64 bits in this algorithm, and c_i and p_i are the ith ciphertext and plaintext bits, respectively. The calculated value is further categorized as a percentage to have a fair comparison using (12). For evaluation purposes, two different approaches are used, from the secret key and the plaintext perspectives. First, the secret key is kept constant while the plaintext is subjected to bit change. Secondly, the plaintext is kept constant for the key evaluation, and the secret key is subjected to bit flip. Table 3 shows the dataset specifications for this test.

$$AE = \frac{1}{x} \sum_{i=1}^{x} |c_i - p_i| \tag{11}$$

$$\text{Percentage of bits changed} = AE \times 100 \tag{12}$$

4.2.2. Correlation Coefficient Test

This test aims at analyzing the non-linear association between plaintext and ciphertext [51–54]. The confusion effect of an algorithm is calculated using (13). In this equation, the value of AE directly comes from (11), where c_i and p_i are the ith ciphertext and plaintext bits, respectively. This test is applied using three different types of keys and 1500 plain texts in total, shown in Table 3. This test aims to identify the relationship between the plaintext and its corresponding ciphertext. This test is performed from an attacker's perspective, who has access to the ciphertext and is establishing a connection with the plaintext. In the given scenario, the obtained values of R range from 0, $+1$ to -1 (both positive and negative directions).

$$R = \frac{\sum_{(i=1)}^{s} (p_i - AE)(c_i - AE)}{\sqrt{\sum_{(i=1)}^{s} (p_i - AE)^2} \sqrt{\sum_{(i=1)}^{s} (c_i - AE)^2}} \tag{13}$$

4.2.3. Semi-Equivalent Key Test

The invertible mappings in the KSA PRESENT ensure that there are no equivalent keys in any rounds [41]. As in [7], equivalent keys encrypt any plaintext to the same ciphertext, primarily because they have similar key expansion. The research was

later extended to search for semi-equivalent keys, which produce nearly similar key expansion, providing very low Hamming weight [9], leading to small bit differences in the ciphertext. One such key pair is listed as 0x2a1145cfc**e**0db6e38eaff175d39c90dc and 0x2a1145cfc**f**0db6e38eaff175d39c90dc, which only differ in one bit (bold and underlined). These keys are subjected to the Hamming weight test using (10) to validate the expected improvement.

4.2.4. Time Complexity

Time complexity is considered one of the essential factors in the encryption process [46]. As KSA directly affects the performance of the cryptosystem, there must be a time comparison between techniques. The KSA is kept simple to control the time complexity of the overall encryption process. If the time complexity of the key generation increases, it will directly affect the whole encryption's performance by increasing its time. A highly efficient KSA with computationally expensive functions is not ideal for resource-constrained devices because speed with optimal security is the ultimate requirement. Therefore, it is necessary to compare the KSA PRESENT's time with the improved variant. The average time of 100 KSA's is calculated for the time complexity analysis.

4.3. Cost

The design of an algorithm is directly affected by the implementation complexity. A lightweight cryptographic solution is only considered implementable if its hardware implementation is inexpensive. The algorithm's applicability is directly associated with the area required for its hardware implementation, as these algorithms are designed to secure systems and perform well on low power devices. The area for hardware implementation is measured in gate equivalence (GE). It measures computational complexity and the chip area required (cost) for implementation. For a lightweight cryptographic solution, this value should not exceed the 3000 GE limit. This GE is calculated for the original and improved KSA, using the fundamentals of the base algorithm [19] and is compared for the cost analysis.

5. Results and Discussion

It is well known that statistical tests are not enough to withstand attacks, yet an algorithm that does not pass this test is more vulnerable to attacks [55,56]. In this section, a detailed comparison of the KSA PRESENT and the improved version is performed.For a fair comparison of these algorithms, the parameter values, the inputs, and the implementation environment are kept the same. The comparison is divided into three major subsections as round keys evaluation, ciphertext evaluation and cost evaluation.

5.1. Round Key Evaluation

KSA evaluation can be better considered if the tests are performed on the generated round keys. The tests evaluate the generated round keys from five different perspectives, and the results are discussed in the following sections.

5.1.1. Frequency Test

The sequence is only considered random if all the round keys pass the frequency test because if this test is failed, then there is no need to perform other statistical tests [57]. For this purpose, a dataset of 10,000 random secret keys is used as input for both KSAs using Equations (1)–(3). Both algorithms pass the frequency test, as the secret keys are random, and thus, the generated round keys are also random. Figure 5 exhibits the passing proportion of 11 round keys produced from random secret keys using the KSA and the improved KSA PRESENT. Both KSAs pass the frequency test, achieving an approximately equal proportion. These results highlight that good frequency can be achieved in round keys, provided that the secret key is random.

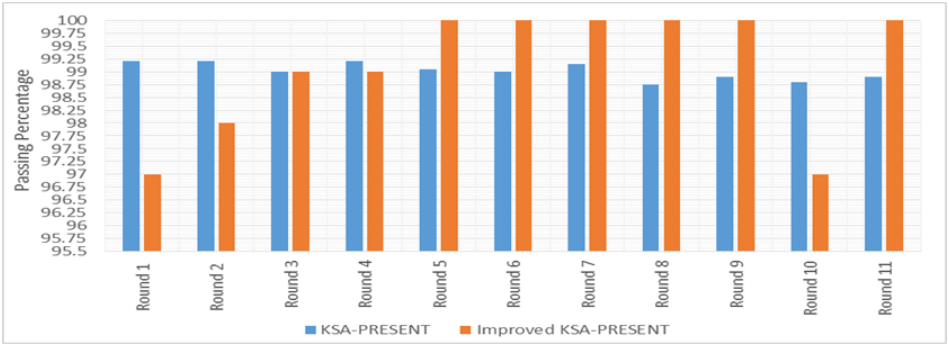


Figure 5. Frequency test on 11 round keys KSA PRESENT and improved KSA PRESENT.

5.1.2. High and Low-Density Key Tests

The comparative results of NIST statistical tests are presented in Table 4 for HDK and LDK. Similar results on AES are also performed in the base paper [6] because it is a standard, and its KSA produces round keys with high randomness based on those four tests. From Table 4, proportions of KSA AES pass all statistical tests, representing the high randomness of round keys generated from nonrandom secret keys. Meanwhile, the KSA PRESENT does not pass any test for non-random keys. This transition is slowly propagative, and this is one of the leading reasons for successful attacks on the reduced round PRESENT [20,45]. The improved KSA PRESENT successfully passes these statistical tests because of the new design considerations which produce random round keys from non-random secret keys. AES has a complex KSA with mostly non-linear functions, but for PRESENT, the KSAs are required to be less complicated owing to their application environment of primarily low-power devices.

Table 4. *p*-Value observations for high- and low-density key tests.

Test Name	KSA AES		KSA PRESENT		Improved KSA PRESENT	
	HDK	LDK	HDK	LDK	HDK	LDK
Frequency Test	99.36	98.28	0.00	0.00	100	100
Block Frequency Test	99.47	98.20	0.00	0.00	100	100
Cusum Test (Forward)	99.41	98.50	0.00	0.00	100	100
Cusum Test (Reverse)	99.43	98.46	0.00	0.00	100	100
Runs Test	98.0	98.30	0.00	0.00	100	99

5.1.3. Bit Differences between Round Keys

The PRESENT encryption algorithm is designed so that the round keys are directly XOR'ed with round data. Random round keys should be XOR'ed with round data to make it less predictable and more random. The KSA PRESENT has the left-shift function applied to 61 bits in each round key; this operation directly affects the maximum bits in each iteration. Tables 5 and 6 show the percentage of the bit difference among two consecutive round keys generated by the KSA and the improved KSA PRESENT, respectively. This difference is due to the one-bit change in different positions of each secret key.

Table 5. Bit difference between round keys using KSA PRESENT.

	High Density Key		Low Density Key		Random Key			
XOR between Round Keys	**No. of Bit Diff**	**% of Bit Diff**	**No. of Bit Diff**	**% of Bit Diff**	**No. of Bit Diff**	**% of Bit Diff**	**Average Bit Difference**	**Average % of Bit Difference**
$k1 \oplus k2$	7.88	6.16	6.24	4.87	64.55	50.43	26.22	20.49
$k2 \oplus k3$	8.88	6.94	8.88	6.94	62.62	48.92	26.79	20.93
$k3 \oplus k4$	10.82	8.45	10.06	7.86	62.76	49.03	27.88	21.78
$k4 \oplus k5$	12.03	9.40	13.00	10.16	63.24	49.40	29.42	22.99
$k5 \oplus k6$	13.12	10.25	14.12	11.03	63.47	49.59	30.24	23.62
$k6 \oplus k7$	14.97	11.69	16.12	12.59	63.15	49.33	31.41	24.54
$k7 \oplus k8$	16.06	12.55	17.24	13.47	63.59	49.68	32.30	25.23
$k8 \oplus k9$	19.61	15.32	20.81	16.26	63.15	49.33	34.52	26.97
$k9 \oplus k10$	20.70	16.17	21.94	17.14	63.41	49.54	35.35	27.62
$k10 \oplus k11$	22.27	17.40	23.69	18.51	63.71	49.77	36.56	28.56
$k11 \oplus k12$	23.15	18.09	24.63	19.24	64.35	50.28	37.38	29.20
$k12 \oplus k13$	25.73	20.10	27.21	21.26	64.00	50.00	38.98	30.45
$k13 \oplus k14$	26.73	20.88	28.21	22.04	63.88	49.91	39.61	30.94
$k14 \oplus k15$	28.73	22.44	30.20	23.59	64.32	50.25	41.08	32.10
$k15 \oplus k16$	29.73	23.22	31.21	24.38	65.12	50.87	42.02	32.83
$k16 \oplus k17$	34.73	27.13	36.21	28.29	64.76	50.60	45.23	35.34
$k17 \oplus k18$	35.73	27.91	37.21	29.07	64.18	50.14	45.70	35.71
$k18 \oplus k19$	37.73	29.47	39.21	30.63	64.12	50.09	47.02	36.73
$k19 \oplus k20$	38.76	30.28	40.24	31.44	64.15	50.12	47.72	37.28
$k20 \oplus k21$	42.73	33.38	44.21	34.54	64.29	50.23	50.41	39.38
$k21 \oplus k22$	45.52	35.56	46.94	36.67	64.59	50.46	52.35	40.90
$k22 \oplus k23$	45.18	35.30	45.97	35.91	65.24	50.97	52.13	40.73
$k23 \oplus k24$	45.85	35.82	47.33	36.98	65.15	50.90	52.78	41.23
$k24 \oplus k25$	46.88	36.62	46.88	36.62	65.12	50.87	52.96	41.37
$k25 \oplus k26$	47.21	36.88	48.18	37.64	65.06	50.83	53.48	41.78
$k26 \oplus k27$	45.73	35.72	47.18	36.86	64.50	50.39	52.47	40.99
$k27 \oplus k28$	48.45	37.86	48.36	37.78	64.97	50.76	53.93	42.13
$k28 \oplus k29$	48.82	38.14	48.58	37.95	64.91	50.71	54.10	42.27
$k29 \oplus k30$	50.91	39.77	49.27	38.49	65.03	50.80	55.07	43.02
$k30 \oplus k31$	53.48	41.78	49.76	38.87	64.76	50.60	56.00	43.75
$k31 \oplus k32$	54.67	42.71	52.18	40.77	64.74	50.57	57.19	44.68

Table 6. Bit difference between round keys using improved KSA PRESENT.

	High Density Key		Low Density Key		Random Key			
XoR between Round Keys	**No. of Bit Diff**	**% of Bit Diff**	**No. of Bit Diff**	**% of Bit Diff**	**No. of Bit Diff**	**% of Bit Diff**	**Average Bit Difference**	**Average % of Bit Difference**
$k1 \oplus k2$	63.34	49.48	71.48	55.85	74.58	58.26	69.80	58.38
$k2 \oplus k3$	54.18	42.33	76.55	59.80	63.74	49.79	64.82	55.65
$k3 \oplus k4$	57.39	44.84	63.85	49.88	63.71	49.77	61.65	52.12
$k4 \oplus k5$	68.27	53.34	57.36	44.82	63.32	49.47	62.99	53.71
$k5 \oplus k6$	66.82	52.20	58.79	45.93	63.56	49.66	63.06	53.73
$k6 \oplus k7$	67.18	52.49	64.82	50.64	62.97	49.20	64.99	56.04
$k7 \oplus k8$	62.24	48.63	66.64	52.06	62.97	49.20	63.95	54.88
$k8 \oplus k9$	65.24	50.97	67.94	53.08	64.41	50.32	65.86	56.64
$k9 \oplus k10$	63.88	49.91	61.97	48.41	64.62	50.48	63.49	53.94
$k10 \oplus k11$	67.18	52.49	64.39	50.31	65.50	51.17	65.69	56.16
$k11 \oplus k12$	65.00	50.78	65.55	51.21	64.56	50.44	65.03	55.67
$k12 \oplus k13$	63.06	49.27	65.21	50.95	63.62	49.70	63.96	54.73
$k13 \oplus k14$	64.12	50.10	63.58	49.67	63.41	49.54	63.70	54.49
$k14 \oplus k15$	65.36	51.07	62.55	48.86	63.24	49.40	63.71	54.55
$k15 \oplus k16$	64.45	50.36	64.21	50.17	63.82	49.86	64.16	54.90
$k16 \oplus k17$	63.45	49.57	63.91	49.93	63.62	49.70	63.66	54.39
$k17 \oplus k18$	63.30	49.46	64.18	50.14	63.76	49.82	63.75	54.45
$k18 \oplus k19$	64.82	50.64	65.06	50.83	63.88	49.91	64.59	55.35
$k19 \oplus k20$	65.42	51.11	63.00	49.22	64.44	50.34	64.29	54.87
$k20 \oplus k21$	63.94	49.95	61.64	48.15	65.71	51.33	63.76	53.96
$k21 \oplus k22$	64.36	50.28	62.82	49.08	65.26	50.99	64.15	54.50
$k22 \oplus k23$	65.67	51.30	62.12	48.53	64.12	50.09	63.97	54.60

Table 6. *Cont.*

	High Density Key		Low Density Key		Random Key			
XoR between Round Keys	**No. of Bit Diff**	**% of Bit Diff**	**No. of Bit Diff**	**% of Bit Diff**	**No. of Bit Diff**	**% of Bit Diff**	**Average Bit Difference**	**Average % of Bit Difference**
$k23 \oplus k24$	64.48	50.38	63.52	49.62	65.50	51.17	64.50	54.83
$k24 \oplus k25$	64.03	50.02	64.39	50.31	63.53	49.63	63.98	54.77
$k25 \oplus k26$	64.36	50.28	65.30	51.02	64.74	50.57	64.80	55.37
$k26 \oplus k27$	62.39	48.75	63.97	49.98	64.09	50.07	63.48	54.07
$k27 \oplus k28$	64.36	50.28	63.91	49.93	64.53	50.41	64.27	54.83
$k28 \oplus k29$	64.55	50.43	63.85	49.88	66.62	52.05	65.00	55.10
$k29 \oplus k30$	62.30	48.67	65.24	50.97	65.29	51.01	64.28	54.64
$k30 \oplus k31$	63.42	49.55	63.46	49.57	65.74	51.36	64.20	54.44
$k31 \oplus k32$	64.52	50.40	63.36	49.50	64.29	50.23	64.06	54.65

The test evaluates the average of the bit difference for 33 HDK, 33 LDK, and 34 random secret keys as input to both the KSA PRESENT and the improved KSA PRESENT. The result shows that the KSA PRESENT generates 6% to 43% of the bit difference among round keys when the secret key is HDK, while the improved KSA PRESENT generates 42% to 54% of the bit difference for the same keys. Using the LDK secret keys datasets, the percentage of the bit difference ranges from 4.8% to 41% for the KSA PRESENT and 44% to 60% for the improved KSA PRESENT. In comparison, the improved KSA PRESENT generates sturdy bit differences among round keys, but the KSA PRESENT generates low bit differences in the early rounds, and then improves later. As depicted in Figure 6, the improved KSA PRESENT outperforms the KSA PRESENT in each round. Using a random secret key dataset, the percentage of the bit change by the KSA PRESENT ranges between 48% and 51%, and the improved KSA PRESENT ranges between 49% and 58.3%. The average percentage of change for KSA PRESENT is between 20% and 45%, and the improved KSA PRESENT is between 52% and 58.4%. Ideally, the transition of bits between round keys should be at least 50%. The slow bit transition in early rounds of the KSA PRESENT can help the attackers predict the earlier round keys and access the secret key.

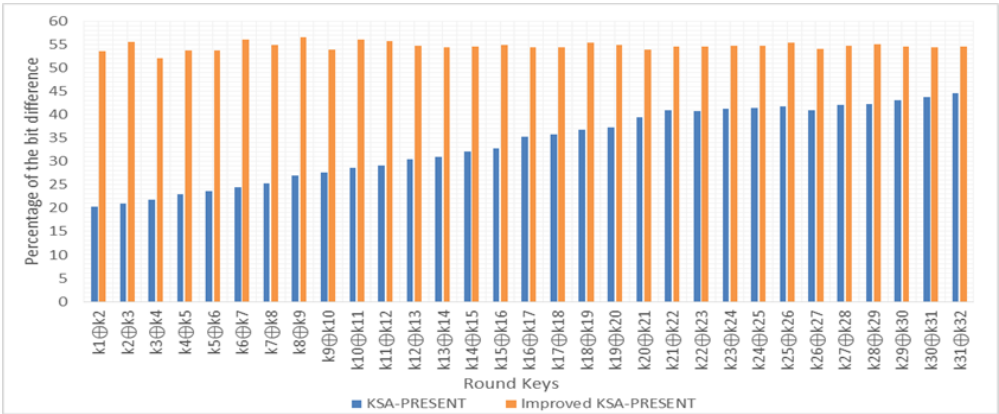


Figure 6. The average percentage of bit difference between generated round keys by the KSA PRESENT and the improved version.

5.1.4. Hamming Weight Test

Low Hamming weight round keys can act as a helping hand for differential cryptanalysis attacks as discussed earlier. The KSA PRESENT produces low Hamming round keys if subjected to a special key [41]. The obtained results are summarized in Table 7, with the round key's Hamming weight and round number calculated using (10).

Table 7. The Hamming weight of round keys using the special secret key.

Round	Hamming Weight		Round	Hamming Weight		Round	Hamming Weight	
	KSA PRESENT	Improved KSA		KSA PRESENT	Improved KSA		KSA PRESENT	Improved KSA
0	27	27	11	7	36	22	17	33
1	15	30	12	12	31	23	10	31
2	23	30	13	3	34	24	17	35
3	10	34	14	12	36	25	12	37
4	20	34	15	4	30	26	20	33
5	9	28	16	14	29	27	12	25
6	18	35	17	4	28	28	25	32
7	7	28	18	13	33	29	14	31
8	18	32	19	5	32	30	24	40
9	6	29	20	14	30	31	17	27
10	16	29	21	8	30	Total	433	1009

The KSA PRESENT produces round keys with Hamming weights ranging from 4 to 47. On the other hand, the improved algorithm produces round keys with Hamming weights ranging from 25 to 40, and the ideal value in each round is 32. The average Hamming weight for KSA PRESENT is 13.53, followed by 31.5 for the improved KSA PRESENT, indicating that the improved KSA produces round keys that can significantly resist differential cryptanalysis attacks [41]. The ideal Hamming weight value for any secret key, along with its round keys, should be 1024 bits. The values presented in Table 7 show that the Hamming weight value for KSA PRESENT is $433/1024 \times 100 = 42.28\%$, whereas when the same key is applied to the improved KSA, the obtained value is $1009/1024 \times 100 = 98.5\%$. These findings indicate that the improved KSA achieves a significantly better Hamming weight and can be a better alternative to the existing KSA.

5.2. Ciphertext Evaluation

Ciphertext evaluation is an integral part of assessing the strength of the improved KSA. It is practically impossible to rule out the encryption algorithm that produces the ciphertext for the final evaluation. Thus, this section evaluates the impact of KSA on the encryption algorithm from four different perspectives, including the avalanche effect, correlation coefficient test, and time complexity. This gives a detailed insight into the effect of the improved KSA on the overall security system.

5.2.1. Avalanche Effect (AE)

The security of the cryptosystem is directly dependent on the plaintext, and the key as the encryption algorithm is publicly available, based on Kerckhoffs's principle. In this section, the effect of the key and the plaintext change is explored in detail. The findings are summarized in Tables 8 and 9 for key and plaintext, respectively. The study gives a deep insight into how many ciphertext bits will change if only one bit of key or plaintext is changed. Ideally, this value should be 32 bits (for PRESENT), exactly 50% or more of the bits. These values are calculated using (11), followed by a percentage conversion from (12) for fair comparison and easy table categorization. Subsequently, the obtained values are categorized into four categories based on the avalanche effect percentage, with 30% to 40%, 40% to 50%, equal to 50% (ideal), and greater than 50% (perfect). Similar datasets are applied to both the KSA PRESENT and its improved version.

The average values in Tables 8 and 9 indicate that the avalanche effect is significantly increased using the improved KSA PRESENT. The observations in Table 8 for the 30% to 40% category are 7.031% for the KSA PRESENT, reduced to 4.427% using the improved KSA PRESENT, indicating few values with smaller AE. Similarly, the observations for 50% AE are 5.73% for the KSA PRESENT, increased to 9.375% using the improved KSA PRESENT. For all other observations, the values are close for both the KSA PRESENT and its improved variant.

Table 8. Avalanche effect using flipped key bits on KSA PRESENT and improved KSA PRESENT.

Key	KSA PRESENT				Improved KSA PRESENT			
	>30≤40%	>40<50%	=50%	>50%	>30≤40%	>40<50%	=50%	>50%
Random	9	56	6	57	9	48	5	66
HDK	11	53	9	55	2	46	18	62
LDK	7	47	7	67	6	57	13	52
Avg.	7.031	40.625	5.73	46.61	4.427	39.32	9.375	46.875

Table 9. Avalanche effect using flipped plaintext bits on KSA PRESENT and improved KSA PRESENT.

Plaintext	KSA PRESENT				Improved KSA PRESENT			
	>30≤40%	>40<50%	=50%	>50%	>30≤40%	>40<50%	=50%	>50%
Random	4	28	5	27	1	23	11	29
HDP	0	25	10	29	0	29	10	25
LDP	0	28	9	27	0	19	15	30
Avg.	2.08	42.19	12.5	43.23	0.52	36.98	18.75	43.75

The AE value for plaintext in Table 9 for category 30% to 40%, is 2.08% for the KSA PRESENT and is reduced to 0.52% for the improved KSA PRESENT, showing fewer observations with small AE. Following a similar pattern, the observations for the KSA PRESENT are 42.19% and 36.98% for the improved KSA PRESENT, for the 40% to 50% category. For the exactly 50% category, the observation for the KSA PRESENT is only 12.5% and is increased to 18.75% for the improved KSA PRESENT. For all other categories, the observations are very similar for both KSAs. Figure 7 summarizes all the values obtained for AE in a graphical format. Figure 7a–c shows random, HDK, and LDK key datasets observations, respectively. Figure 7d–f shows random HDP and LDP datasets' observations, respectively. These comprehensive results prove that the curve of the avalanche effect is more inclined toward a higher percentage of change using the improved KSA PRESENT.

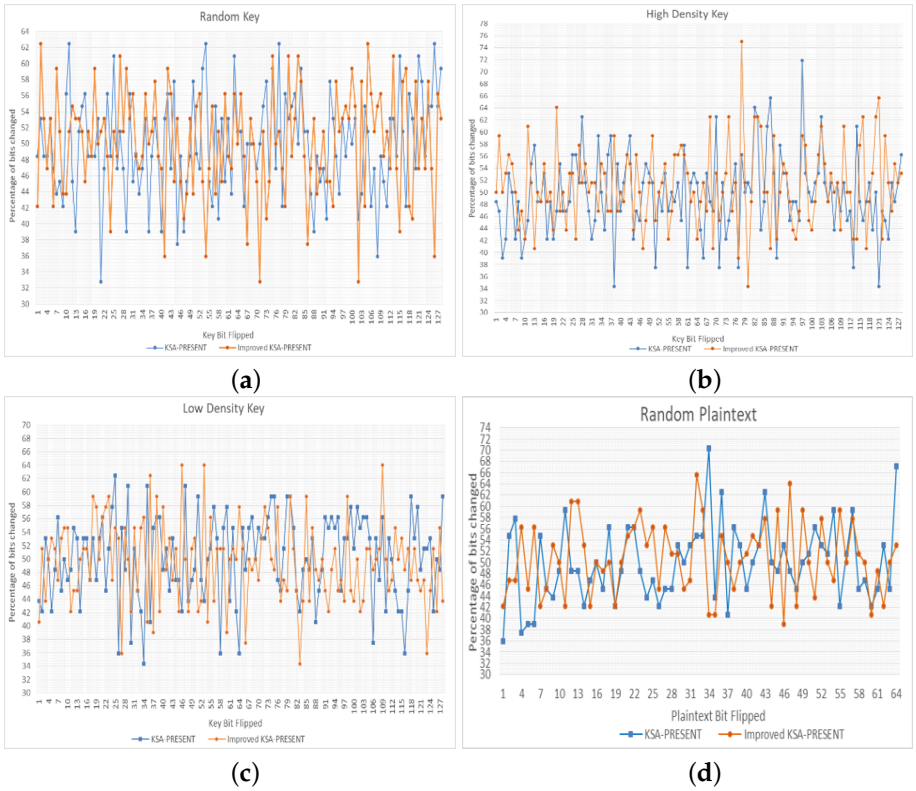
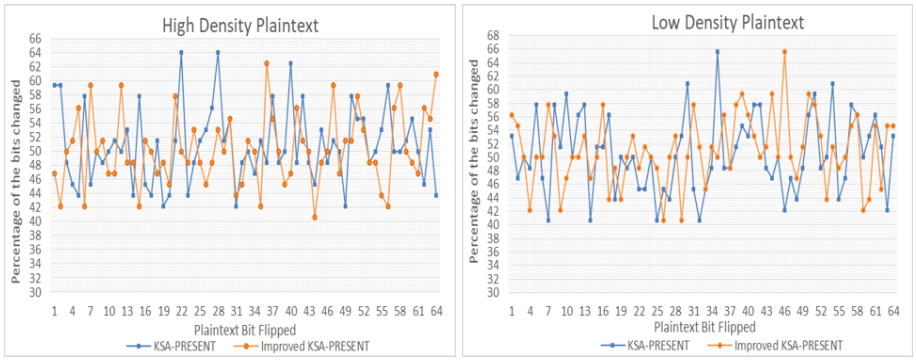


Figure 7. *Cont.*



(**e**) (**f**)

Figure 7. Key (**a**–**c**), plaintext (**d**–**f**) bits flipped, and the percentage of bits changed in ciphertext.

5.2.2. Correlation Coefficient

The correlation coefficient is the ultimate test to find the relationship between plaintext and its corresponding ciphertext. Equation (13) is used to calculate the correlation value for 1500 different inputs for the KSA PRESENT and its improved version. The obtained value, R, depicts this relationship, as its values include 0, $+1$ and -1. Here, the positive and negative values of the correlation depict the direction of the relationship [51]. Value zero indicates a perfect value; the ciphertext is entirely independent of plaintext, and the attacker cannot establish any connection between the plaintext and ciphertext by looking at any one of them [58]. Values from 0 to 0.3 indicate a very weak relationship between plaintext and ciphertext, whereas those from 0.3 to 0.7 indicate a moderate relationship both in the positive and negative directions. A very strong relationship is observed if the R-value is from 0.7 to 1, followed by a perfect relationship if $R = +1$ or $R = -1$.

The output values from the three datasets are categorized according to their relationships and summarized in Table 10. The KSA PRESENT and improved KSA PRESENT are subjected to five different RK, LDK, and HDK, each. Each key is applied to a total of 100 plaintexts, and the obtained R-value is categorized according to the correlation between plaintext and ciphertext. Furthermore, an average of each dataset is also calculated for a fair comparison of both schemes. Comparing the results for KSA, the improved KSA PRESENT significantly outperforms the existing scheme for $R = 0$ for all three datasets. There is an increase in the weak plaintext and ciphertext relationships for all other observations and a decrease in strong relationships. Hence, it can be established that the improved KSA PRESENT can be a good alternative for situations where an attacker tries to access the plaintext by using ciphertext to establish a connection between them through the correlation properties. Moreover, it can also be seen that there are no observations of moderate, strong, or perfect relations between the plaintext and ciphertext.

Table 10. Correlation analysis between plaintext and ciphertext.

	KSA PRESENT					Improved KSA PRESENT				
Key	$R = 0$	$0 < \lvert R \rvert \leq 0.3$	$-0.3 \leq \lvert R \rvert < 0$	$0.3 < R < 0.7$	$-0.7 < R < -0.3$	$R = 0$	$0 < \lvert R \rvert \leq 0.3$	$-0.3 \leq \lvert R \rvert < 0$	$0.3 < R < 0.7$	$-0.7 < R < -0.3$
RK1	8	40	49	3	0	9	40	51	0	0
RK2	13	30	56	1	0	16	43	38	3	0
RK3	6	52	42	0	0	11	53	34	2	0
RK4	7	37	56	0	0	7	37	55	1	0
RK5	9	43	46	2	0	11	38	50	1	0
%age	8.6	40.4	49.8	1.2	0	10.8	42.2	45.6	1.4	0

Table 10. *Cont.*

	KSA-PRESENT					Improved KSA PRESENT				
Key	$R = 0$	$0 < R \leq 0.3$	$-0.3 \leq R < 0$	$0.3 < R < 0.7$	$-0.7 < R < -0.3$	$R = 0$	$0 < R \leq 0.3$	$-0.3 \leq R < 0$	$0.3 < R < 0.7$	$-0.7 < R < -0.3$
LDK1	5	44	50	1	0	14	43	43	0	0
LDK2	8	51	39	2	0	4	41	52	3	0
LDK3	9	45	42	4	0	9	41	47	3	0
LDK4	11	32	56	1	0	9	42	48	1	0
LDK5	7	47	46	0	0	10	45	43	2	0
%age	8.1	43.23	47.13	1.53	0	9.467	42.37	46.43	1.73	0
HDK1	15	44	40	1	0	10	45	42	3	0
HDK2	4	49	43	4	0	15	38	45	2	0
HDK3	13	50	35	2	0	10	46	43	1	0
HDK4	8	46	43	3	0	9	46	44	1	0
HDK5	10	39	47	4	0	13	42	42	3	0
Total	133	649	690	28	0	157	640	677	26	0

5.2.3. Semi-Equivalent Key Test

This test uses a semi-equivalent secret key as input to both the KSA PRESENT and its improved variant. Table 11 summarizes the findings from both KSA for round keys as well as for ciphertext. These two keys differ with only one bit, and that bit is bold in the first column of Table 11. The difference between the 32 keys (31 round keys and 1 secret key) is listed in the second column indicating an increase from 16 to 779 bits using the improved KSA. Two keys differ by only one bit, so their values are compared for ciphertext too, using the consistent HD plaintext. By keeping the plaintext constant and observing the change in the ciphertext, the bit difference is increased from 43.75% to 53.13%. These findings prove that even when the improved KSA is subjected to semi-equivalent keys, it increases the round keys difference and bit difference in the ciphertext.

Table 11. Semi-equivalent keys, round keys and ciphertext difference.

Secret Key	**Round Key Difference**		**Ciphertext Difference**	
	KSA PRESENT	**Improved KSA**	**KSA PRESENT**	**Improved KSA**
0x2a1145cfc**e** 0db6e38eaff1 75d39c90dc	16 bits 16/2048 × 100 = 0.78%	779 bits 779/2048 × 100 = 38.04%	101010100000011 1010011000100101 1111001111111111 10111111101000101	0001010111101001 1100111100000011 1100110101000110 1000100000011010
0x2a1145cfc**f** 0db6e38eaff1 75d39c90dc			1101001001111010 0100100001011101 1001010010100111 0110001100011101	0011101100100001 1011100010011101 0001010001011101 0101101000001001
Similar bits between two ciphertext			36	30
Avalanche effect			43.75%	53.13%

5.2.4. Time Complexity

A complex KSA may provide better security, yet it takes more time to generate round keys. In a resource-constrained environment, the proposed security model should consider an optimal time complexity to improve the performance of the cryptosystem. The time complexity comparison is performed in Matlab 2019, with CPU speed 3 GHz. The proposed KSA is compared against KSA of AES and the original PRESENT presented in Table 12. In comparison, the KSA for AES-128 takes approximately 8 s to generate 11 round keys for the encryption, while the same number of keys is generated in 0.002122 s by PRESENT. The

improved KSA takes 0.005358 s to generate 11 round keys. This significant time difference between the KSAs indicates that the KSA for the AES algorithm has more complex functions than the KSA PRESENT and improved KSA PRESENT.

Table 12. Time complexity of KSA, AES, PRESENT and improved KSA PRESENT.

Algorithm	KSA Time (11 Round Keys)	KSA Time (31 Round Keys)
AES-128	8 s	-
KSA PRESENT	0.002122 s	0.00580 s
Improved KSA PRESENT	0.005358 s	0.007671 s

The KSA is carefully designed with a combination of linear and non-linear functions, which result in better security, yet the time complexity is not increased much. Meanwhile, the KSA PRESENT takes 0.00580 s to generate 31 round keys, and the improved KSA PRESENT takes 0.007671 s to generate the same number of round keys. These values are obtained by taking the average of 100 different keys in KSA. This time difference of 0.001871 s is negligible as a trade-off for the security improvement demonstrated by the proposed KSA for the avalanche effect, randomness, and correlation coefficient.

5.3. Cost

Area requirements for PRESENT (128) are 1886 GE and are presented in Table 13, along with the cost calculation for the improved KSA. The cost of implementation is increased from 1886 to 2502.85 GEs (an additional 633 GEs). Firstly, it is still below 3000, which is the cost limit for lightweight cryptographic solutions [39]. Secondly, there is always a trade-off between achieving better security and cost.

Table 13. Cost comparison of KSA PRESENT and improved KSA PRESENT.

Module (Encryption)	GE	Module (KSA)	GE-Original	GE-Improved
Data Sate	384.39	Ks:Key State	768.78	768.78
S-Layer	448.45	Ks:Phi Storage	—	192.195
P-Layer	0	Ks:Phi-XoR	—	85.42
Counter.State	28.36	Ks: S-Box	56.06	224.225
Counter.Combinatorial	12.35	Ks:XoR	–	85.42*2
Other	3.67	Ks:Rotation	0	0
		Ks:Counter-XoR	13.35	13.35
		Key XoR	170.84	170.84
		Total	1886.25	2502.85

6. Conclusions

This research aims to investigate the security impact on the improvements of the KSA of the symmetric block cipher, PRESENT. An improved KSA with the golden ratio, XOR, and reverse operations overcomes the slow transition and predictable linear behavior in round keys of the existing KSA. The results accentuate that the improved KSA enhances the overall security of the PRESENT block cipher. The original KSA only provides random round keys when the secret key is random, but the improved KSA provides round key randomness for low-density, high-density, and random secret keys. Bit change in the round keys increases from 20% to 54% on average, with increased Hamming weight from 42.8% to 98.5%, and the round key difference for a semi-equivalent key is increased from 0.78% to 38% using the improved KSA. Moreover, the ciphertext analysis proved that the avalanche effect has increased using the improved KSA, and more values are observed with 50% or more avalanche effect. The correlation coefficient of plaintext and ciphertext has a higher number of zeros for the improved KSA, and this pattern is consistent in all random, low-, and high-density keys. As a trade-off to these improvements, the round key generation time is also increased by 0.001871 s on average, and there is a 32.6% increase in the required

number of gates for hardware implementation, which is minimal considering the garnered security improvements.

This research has two significant findings that can be considered for future directions. To start, KSA plays a very critical role in the security strength of cryptographic algorithms. The random round keys directly influence the randomness of the algorithm. Secondly, it is not necessary to introduce complex mathematical operations in KSA to achieve better security, but a simple KSA with carefully selected linear and nonlinear operations can also enhance both round keys and ciphertexts.

Author Contributions: Conceptualization, M.I., S.N.R. and H.M.; methodology, M.I., S.N.R.; software, M.I.; validation, M.I., S.N.R. and H.M.; formal analysis, M.I.; investigation, M.I.; resources, M.I.; data curation, M.I.; writing—original draft preparation, M.I.; writing—review and editing, S.N.R.; visualization, M.I.; supervision, S.N.R. and H.M.; project administration, S.N.R.; funding acquisition, S.N.R. All authors have read and agreed to the published version of the manuscript.

Funding: Fundamental Research Grant Scheme Vot No. FRGS/1/2019/ICT03/UTHM/03/1. This research is supported by Ministry of Higher Education (MOHE) through Fundamental Research Grant Scheme (FRGS/1/2019/ICT03/UTHM/03/1).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All relevant data are available in the submitted files.

Acknowledgments: The authors would like to thank all reviewers for their helpful comments.

Conflicts of Interest: The authors declare that they have no conflict of interest.

References

1. Knudsen, L.R.; Robshaw, M. *The Block Cipher Companion*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2011.
2. Barker, E.; Roginsky, A. Transitions: Recommendation for transitioning the use of cryptographic algorithms and key lengths. *NIST Spec. Publ.* **2011**, *800*, 131A.
3. Barker, E.; Roginsky, A. *Transitioning the Use of Cryptographic Algorithms and Key Lengths*; Technical Report; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2018.
4. Salameh, J.N.B. A New Technique for Sub-Key Generation in Block Ciphers. *World Appl. Sci. J.* **2012**, *19*, 1630–1639.
5. Ebrahim, M.; Khan, S.; Khalid, U.B. Symmetric algorithm survey: A comparative analysis. *Int. J. Comput. Appl.* **2013**, *61*, 12–19.
6. Afzal, S.; Yousaf, M.; Afzal, H.; Alharbe, N.; Mufti, M.R. Cryptographic strength evaluation of key schedule algorithms. *Secur. Commun. Netw.* **2020**, *2020*, 3189601. [CrossRef]
7. Disina, A.H.; Pindar, Z.A.; Jamel, S.B.H. Enhanced caeser cipher to exclude repetition and withstand frequency cryptanalysis. *J. Netw. Inf. Secur. Vol.* **2014**, *2*, 7–13.
8. Mushtaq, M.F.; Jamel, S.; Disina, A.H.; Pindar, Z.A.; Shakir, N.S.A.; Deris, M.M. A survey on the cryptographic encryption algorithms. *Int. J. Adv. Comput. Sci. Appl.* **2017**, *8*, 333–444.
9. Marinakis, G. Selection of sampling keys for cryptographic tests. *Cryptol. Eprint Arch.* **2021**, *2021*, 1–11. [CrossRef]
10. Blumenthal, U.; Bellovin, S.M. A better key schedule for DES-like ciphers. In Proceedings of the Pragocrypt'96, Prague, Czech Republic, 30 September–3 October 1996.
11. Pereira, G.C.; Alves, R.C.; Silva, F.L.d.; Azevedo, R.M.; Albertini, B.C.; Margi, C.B. Performance evaluation of cryptographic algorithms over IoT platforms and operating systems. *Secur. Commun. Netw.* **2017**, *2017*, 2046735. [CrossRef]
12. Rashidi, B. Flexible structures of lightweight block ciphers PRESENT, SIMON and LED. *IET Circuits Devices Syst.* **2020**, *14*, 369–380. [CrossRef]
13. Beierle, C.; Jean, J.; Kölbl, S.; Leander, G.; Moradi, A.; Peyrin, T.; Sasaki, Y.; Sasdrich, P.; Sim, S.M. The SKINNY family of block ciphers and its low-latency variant MANTIS. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 14–18 August 2016.
14. Liu, B.T.; Li, L.; Wu, R.X.; Xie, M.M.; Li, Q.P. Loong: A family of involutional lightweight block cipher based on SPN structure. *IEEE Access* **2019**, *7*, 136023–136035. [CrossRef]
15. Li, L.; Liu, B.; Zhou, Y.; Zou, Y. SFN: A new lightweight block cipher. *Microprocess. Microsyst.* **2018**, *60*, 138–150. [CrossRef]
16. Wheeler, D.J.; Needham, R.M. TEA, a tiny encryption algorithm. In Proceedings of the International Workshop on Fast Software Encryption, Leuven, Belgium, 14–16 December 1994.
17. Usman, M.; Ahmed, I.; Aslam, M.I.; Khan, S.; Shah, U.A. SIT: A lightweight encryption algorithm for secure internet of things. *Int. J. Adv. Comput. Sci. Appl.* **2017**, *8*, 402–411. [CrossRef]

18. Zhang, W.; Bao, Z.; Lin, D.; Rijmen, V.; Yang, B.; Verbauwhede, I. RECTANGLE: A bit-slice lightweight block cipher suitable for multiple platforms. *Sci. China Inf. Sci.* **2015**, *58*, 1–15. [CrossRef]
19. Bogdanov, A.; Knudsen, L.R.; Leander, G.; Paar, C.; Poschmann, A.; Robshaw, M.J.; Seurin, Y.; Vikkelsoe, C. PRESENT: An ultra-lightweight block cipher. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Vienna, Austria, 10–13 September 2007.
20. Collard, B.; Standaert, F.X. A statistical saturation attack against the block cipher PRESENT. In Proceedings of the Cryptographers Track at the RSA Conference, San Francisco, CA, USA, 20–24 April 2009.
21. Nakahara, J.; Sepehrdad, P.; Zhang, B.; Wang, M. Linear (hull) and algebraic cryptanalysis of the block cipher PRESENT. In Proceedings of the International Conference on Cryptology and Network Security, Kanazawa, Japan, 12–14 December 2009.
22. Özen, O.; Varıcı, K.; Tezcan, C.; Kocair, Ç. Lightweight block ciphers revisited: Cryptanalysis of reduced round PRESENT and HIGHT. In Proceedings of the Australasian Conference on Information Security and Privacy, Brisbane, Australia, 1–3 July 2009.
23. Lo, O.; Buchanan, W.J.; Carson, D. Correlation power analysis on the PRESENT block cipher on an embedded device. In Proceedings of the 13th International Conference on Availability, Reliability and Security, Hamburg, Germany, 27–30 August 2018.
24. De Cnudde, T.; Nikova, S. Securing the present block cipher against combined side-channel analysis and fault attacks. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2017**, *25*, 3291–3301. [CrossRef]
25. Lara-Nino, C.A.; Morales-Sandoval, M.; Diaz-Perez, A. Novel FPGA-based low-cost hardware architecture for the PRESENT block cipher. In Proceedings of the 2016 Euromicro Conference on Digital System Design (DSD), Limassol, Cyprus, 31 August–2 September 2016.
26. Madakam, S.; Ramaswamy, R.; Tripathi, S. Internet of Things (IoT): A literature review. *J. Comput. Commun.* **2015**, *3*, 164. [CrossRef]
27. Sarma, S. *Towards the Five-Cent Tag*; Technical Report; MIT AUTO-ID Center, Massachusetts Institute of Technology: Cambridge, MA, USA, 2001.
28. Weis, S.A. Security and Privacy in Radio-Frequency Identification Devices. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2003.
29. Thakor, V.A.; Razzaque, M.A.; Khandaker, M.R. Lightweight cryptography algorithms for resource-constrained IoT devices: A review, comparison and research opportunities. *IEEE Access* **2021**, *9*, 28177–28193. [CrossRef]
30. Basu, S. International data encryption algorithm (idea)—A typical illustration. *J. Glob. Res. Comput. Sci.* **2011**, *2*, 116–118.
31. Hussien, H.M.; Muda, Z.; Yasin, S.M. New key expansion function of Rijndael 128-bit resistance to the related-key attacks. *J. Inf. Commun. Technol.* **2018**, *17*, 409–434.
32. Derbez, P.; Fouque, P.A.; Jean, J.; Lambin, B. Variants of the AES key schedule for better truncated differential bounds. In Proceedings of the International Conference on Selected Areas in Cryptography, Calgary, AB, Canada, 15–17 August 2018.
33. De Los Reyes, E.M.; Sison, A.M.; Medina, R. Modified AES cipher round and key schedule. *Indones. J. Electr. Eng. Inform. (IJEEI)* **2019**, *7*, 28–35.
34. Sachdeva, S.; Kakkar, A. Implementation of AES-128 using multiple cipher keys. In Proceedings of the International Conference on Futuristic Trends in Network and Communication Technologies, Solan, India, 9–10 February 2018.
35. Harmouch, Y.; El Kouch, R. The benefit of using chaos in key schedule algorithm. *J. Inf. Secur. Appl.* **2019**, *45*, 143–155. [CrossRef]
36. Rahim, R.; Suprianto, S.; Multazam, M. GOST enhancement key processing with Triple Transposition Key. *J. Phys. Conf. Ser.* **2019**, *1402*, 066093. [CrossRef]
37. Galas, E.M.; Gerardo, B.D. Implementing randomized salt on round key for corrected block tiny encryption algorithm (XXTEA). In Proceedings of the 2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN), Chongqing, China, 12–15 June 2019.
38. Zakaria, A.A.; Azni, A.; Ridzuan, F.; Zakaria, N.H.; Daud, M. Modifications of Key Schedule Algorithm on RECTANGLE Block Cipher. In Proceedings of the International Conference on Advances in Cyber Security, Penang, Malaysia, 8–9 December 2020.
39. Dhanda, S.S.; Singh, B.; Jindal, P. Lightweight cryptography: A solution to secure IoT. *Wirel. Pers. Commun.* **2020**, *112*, 1947–1980. [CrossRef]
40. Chom Thungon, L.; Ahmed, N.; Hussain, M. Comparison of AES and PRESENT Block Cipher for 6LoWPAN Based Internet-of-Things. *Int. J. Comput. Intell. IoT* **2018**, *1*, 255–259.
41. Hernandez-Castro, J.C.; Peris-Lopez, P.; Aumasson, J.P. On the key schedule strength of present. In Proceedings of the Data Privacy Management and Autonomous Spontaneus Security, Leuven, Belgium, 15–16 September 2011.
42. Anderson, R.; Biham, E.; Knudsen, L. Serpent: A proposal for the advanced encryption standard. *NIST Aes Propos.* **1998**, *174*, 1–23.
43. Rukhin, A.; Soto, J.; Nechvatal, J.; Smid, M.; Barker, E. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*; Technical Report; Booz-Allen and Hamilton Inc.: McLean, VA, USA, 2001.
44. Wang, M. Differential cryptanalysis of reduced-round PRESENT. In Proceedings of the International Conference on Cryptology in Africa, Casablanca, Morocco, 11–14 June 2008.
45. Kumar, M.; Yadav, P.; Kumari, M. Flaws in differential cryptanalysis of reduced round present. *Cryptol. Eprint Arch.* **2010**, *2010*, 1–8.
46. Duta, C.L.; Mocanu, B.C.; Vladescu, F.A.; Gheorghe, L. Randomness evaluation framework of cryptographic algorithms. *Int. J. Cryptogr. Inf. Secur.* **2014**, *4*, 31–49 . [CrossRef]

47. Sulaiman, S.; Muda, Z.; Juremi, J.; Mahmod, R.; Yasin, S.M. A new shift column transformation: An enhancement of Rijndael key scheduling. *Int. J. Cyber-Secur. Digit. Forensics (IJCSDF)* **2012**, *1*, 160–166.
48. Abikoye, O.C.; Haruna, A.D.; Abubakar, A.; Akande, N.O.; Asani, E.O. Modified advanced encryption standard algorithm for information security. *Symmetry* **2019**, *11*, 1484. [CrossRef]
49. Thorat, C.; Inamdar, V. Implementation of new hybrid lightweight cryptosystem. *Appl. Comput. Inform.* **2018**, *16*, 195–206. [CrossRef]
50. Biswas, A.; Majumdar, A.; Nath, S.; Dutta, A.; Baishnab, K. LRBC: A lightweight block cipher design for resource constrained IoT devices. *J. Ambient Intell. Humaniz. Comput.* **2020**. [CrossRef]
51. Zakaria, A.A.; Azni, A.; Ridzuan, F.; Zakaria, N.H.; Daud, M. Extended RECTANGLE algorithm using 3D bit rotation to propose a new lightweight block cipher for IoT. *IEEE Access* **2020**, *8*, 198646–198658. [CrossRef]
52. Sallam, A.I.; Faragallah, O.S.; El-Rabaie, E.S.M. HEVC selective encryption using RC6 block cipher technique. *IEEE Trans. Multimed.* **2017**, *20*, 1636–1644. [CrossRef]
53. Li, H.; Yang, G.; Ming, J.; Zhou, Y.; Jin, C. Transparency order versus confusion coefficient: A case study of NIST lightweight cryptography S-Boxes. *Cybersecurity* **2021**, *4*, 35. [CrossRef]
54. Prakasam, P.; Madheswaran, M.; Sujith, K.; Sayeed, M.S. An Enhanced Energy Efficient Lightweight Cryptography Method for various IoT devices. *ICT Express* **2021**, *7*, 487–492.
55. Simion, E. The relevance of statistical tests in cryptography. *IEEE Secur. Priv.* **2015**, *13*, 66–70. [CrossRef]
56. Sỳs, M.; Klinec, D.; Kubíček, K.; Švenda, P. Booltest: The fast randomness testing strategy based on Boolean functions with application to DES, 3-DES, MD5, MD6 and SHA-256. In Proceedings of the International Conference on E-Business and Telecommunications, Madrid, Spain, 24–26 July 2017.
57. Marton, K.; Suciu, A. On the interpretation of results from the NIST statistical test suite. *Sci. Technol.* **2015**, *18*, 18–32.
58. Noura, H.; Chehab, A.; Sleem, L.; Noura, M.; Couturier, R.; Mansour, M.M. One round cipher algorithm for multimedia IoT devices. *Multimed. Tools Appl.* **2018**, *77*, 18383–18413. [CrossRef]