(IJISE) 2026, Vol. No. 24, Jul-Dec **e-ISSN: 2454-6402, p-ISSN: 2454-812X**

Enhancing the LED-128 Lightweight Block Cipher Through a Hénon-Driven Dynamic S-Box Key Scheduling Mechanism

Farah Neamah Abbas
Department of Computer Science, Mustansiriyah University, Baghdad, Iraq

[1]*Received: 24/09/2026; Accepted: 27/10/2026; Published: 05/11/2026*

Abstract

People use Internet of Things (IoT) devices and wireless sensor networks repeatedly now. It is difficult to keep sensitive data safe, when systems have limited resources. Lightweight block ciphers are useful for encrypting data. The original LED-128 cipher has problems in how it schedules keys. The round keys aren't diverse enough, they don't have sufficient statistical randomness and the cipher isn't sensitive enough to key changes. Because of those problems, this research introduces an adjusted key schedule called the Hénon-Driven Dynamic S-Box Key Schedule (HD-SBKS) for the LED-128 block cipher. We tested the HD-SBKS mechanism with multiple statistical and cryptographic tools - those tools are the NIST Statistical Test Suite, Correlation Coefficient Analysis (CCA), Hamming Weight (HW) testing and Avalanche Effect (AE) analysis. The test data show the adjusted algorithm has better statistical numbers than the original LED-128 cipher. The adjusted algorithm passes all 15 NIST tests but the original cipher fails two tests at a significance level of 0.01. The average correlation coefficient is 0.008415 instead of the previous 0.030972, which means the plaintext plus ciphertext have less connection to each other. The round keys have an average Hamming weight of 32 bits instead of 26.62 bits, which is equal to the expected 50 % distribution of bits. The average avalanche effect is 49.11%, which is closer to the expected 50 % target than the previous 45.28%. The HD-SBKS mechanism is therefore a useful change for lightweight cryptography, when systems need to encrypt data better without changing the original cipher structure.

Keywords: *LED Algorithm; henon map; lightweight algorithms; IOT; cryptography algorithms*

1. Introduction

The develop Internet of Things technologies and wireless sensor networks quickly; connected devices transmit and exchange more private data than before - it is hard to keep this data secret and safe, when devices have little computing power plus memory but also energy [1],[2]. Many people study lightweight block ciphers because those algorithms protect data and need only small amounts of computation and hardware. The LED (Lightweight Encryption Device) block cipher is one of those algorithms it is useful because it has few parts and doesn't need many resources. The original LED-128 cipher has good qualities but it also has problems in its key schedule. The way it uses the key isn't dynamic or complicated when it expands the key [3], [4].

Because of this the round keys are less diverse, have worse statistical properties. Because it has a fixed substitution structure, the key generation process varies less based on the key. Because of those problems, someone needs to make

[1] *How to cite the article:* Abbas F.N.; November 2026; Enhancing the LED-128 Lightweight Block Cipher Through a Hénon-Driven Dynamic S-Box Key Scheduling Mechanism; *International Journal of Innovations in Scientific Engineering*, Jul-Dec 2026, Vol 24, 98-112

a key schedule that makes keys more sensitive and more random and more statistically independent, without making the cipher much more complicated than it was [5].

To fix those problems, this research has a Hénon-Driven Dynamic S-Box Key Schedule (HD-SBKS) for the LED-128 block cipher this method changes how the key expands and has 3 parts - a Hénon chaotic map based on the key, dynamic S-boxes made with affine transformations over GF(2^4) and a 64-bit Feistel structure to expand the key.
The Hénon map uses the 128-bit master key to make predictable intermediate words. Those words make thirteen dynamic S-boxes that depend on the key, which makes the nonlinear steps change more. After that the Feistel structure uses those dynamic S-boxes to make thirteen 64-bit round keys. The method makes the key schedule better - It keeps the original inside parts of LED-128, which have substitution and permutation operations. Because it is built this way, the method makes the numbers in the cipher more evenly distributed and the cipher changes more when someone alters the key. It does this without changing the basic parts of the cipher. The method has chaotic sequences and parts that change without a straight line, which are there so the round keys are harder to guess. Those parts make the master key plus the keys for encrypting more dependent on each other. Tests show how well the HD-SBKS mechanism works - The numbers from the tests are better than the numbers from the original LED-128 cipher.

The main contributions of this research can be summarized as follows:

1. HD-SBKS is a modified way to schedule keys - It changes the original LED-128 process to expand keys but the internal encryption steps are the same as before.
2. The system has the Hénon chaotic map, which it uses to make deterministic intermediate words from the 128-bit master key.
3. Thirteen dynamic S-boxes are made from the key - applying affine transformations over GF(2^4). They have bijective properties that don't change. A 64-bit Feistel structure is used to make thirteen round keys. Because of this the keys are more different than each other and have a more equal distribution.
4. tested in multiple ways, the results show the LED-128 block cipher algorithm has better statistical security numbers and changes more when the key changes.

The remainder of the paper, the second section presents related studies, the third section covers the Theoretical Background, the fourth section proposes an improvement to the LED algorithm, the fifth section discusses the results, and the final section presents the conclusions.

2. Review of the Literature

This section describes earlier papers that were about the Lightweight Encryption Device (LED) block cipher. The text is about changes to the parts inside the cipher and how people put the cipher into hardware - those papers are here to show the methods that individuals used and the outcomes they had but also how they made the LED algorithm run better.

Ayoub Mhaouch et al. [6] (2024) made hardware for the Lightweight Encryption Device (LED) block cipher for IoT devices that have small amounts of memory and power. The paper has two hardware setups that make the LED encryption algorithm run faster. The hardware was made to have low wait times plus to output data at a high rate whilst it doesn't use many parts. The authors had their designs on an FPGA and compared their numbers to other block ciphers. The paper was written to show a hardware setup for running the LED block cipher on small devices.

Ayoub Mhaouch et al. [7] (2023) made a hardware design for the LED block cipher for Remote Keyless Entry (RKE) systems in cars. The setup was made to work when few hardware parts are available. The authors made an 8-bit sequential circuit and made a 4-bit circuit for the MixColumns step so the design used fewer parts. The 8-bit sequential

circuit had 627 LUTs+FFs and the 4-bit MixColumns circuit had 597 LUTs+FFs. The authors also tested the security, operation of the circuit with entropy, histograms, correlation, the Number of Pixel Change Rate (NPCR) and the Unified Average Changing Intensity (UACI). The data showed the design was a way to put the cipher into hardware for keyless entry systems that have small processors.

Mohammed Al-Shatari et al. [8] (2023) made a hardware setup for encryption and authentication using the LED block cipher and the PHOTON hash function for IoT edge hardware. The design has LED-64 encryption plus the PHOTON-80/20/16 hash function put together with the Encrypt-then-MAC method. To use fewer hardware parts, the authors used the same basic steps for both parts of the system and they had Look-Up Tables (LUTs) in place of Galois field multiplications for the MixColumns step. The design was written in Verilog HDL and was tested on Cyclone FPGA chips. The tests showed the design had about 14 % less logic area and had up to 46.04% less power use than separate LED and PHOTON setups.

Piyush Modi et al. [9] (2021) made hardware layouts for the LED and PRESENT block ciphers for systems that don't have many parts. The paper has two hardware layouts for the LED cipher. The first layout has pipeline registers between the round steps so that it processes data faster. The second layout processes one cell per clock cycle so that it uses fewer parts. The authors also made a 32-bit datapath for the PRESENT cipher. The layouts were placed on Field Programmable Gate Array (FPGA) chips, the authors checked the hardware use and the ratio between the circuit area and the data speed.

Wajih El Hadj Youssef et al. [10] (2020) made hardware implementations of ciphers for IoT devices that have few resources. The paper had hardware setups for LED-64/128, SIMON-64/128 and SIMECK-64/128 that had 32-bit datapaths. The designs were put on Spartan-3, Spartan-6 and Zynq-7000 FPGA chips and the authors checked the chip area plus the speed and the power use. The LED-64/128 circuit had a throughput of 210.13 Mbps on the Zynq-7000 chip. The authors also tested the security using images and information entropy also correlation coefficients to see how the designs handled attacks based on statistics.

Patil et al. [11] (2015) made a version of the LED block cipher called SR-LED. The algorithm changes two parts of the first LED cipher - the Substitution Box (S-Box) and the Key Schedule Algorithm. The S-Box was from the RECTANGLE block cipher and the key schedule was like the one in the SPECK block cipher - those changes were made to change the structure of the LED algorithm, make it harder to attack with related keys; this was the goal of the authors and what they said happened, not proof that it stops every attack with related keys. As reported in the review by Zakaria et al. (2023), SR-LED has a 64-bit block size, a 128-bit key size and a 12-round substitution permutation network (SPN) structure.Theoretical Background

2.1 The LED Block Cipher

LED is a lightweight block cipher with SPN structure, whose design has several AES-like features, for example, S-Boxes, ShiftRows, and MixColumns operations. The cipher state is conceptually arranged as a 4 × 4 matrix where each nibble represents an element from GF(2^4). For a 64-bit plaintext m, the 16 four-bit nibbles $m_0 \| m_1 \| \dots \| m_{15}$ are arranged as a 4 × 4 matrix. Unlike LED-64, LED-128 uses a 128-bit master key, which is divided into two 64-bit subkeys, K_0 and K_1. Each subkey can be represented as a 4 × 4 array of 4-bit nibbles [12].

$$= \begin{pmatrix} m_0 & m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 & m_7 \\ m_8 & m_9 & m_{10} & m_{11} \\ m_{12} & m_{13} & m_{14} & m_{15} \end{pmatrix}$$

The LED updates its state as it runs, which is what other block ciphers do. In the addRoundKey (AK) step, someone or something XORs the plaintext P with K and then this output is in a step operation. In LED-128 the first state is XORed with K_0. The cipher has twelve step operations in total. At the AddRoundKey points, K_0 and K_1 have a back-and-forth pattern where one is used and then the other is used. The last key addition has K_0 and the ciphertext is the result [13].

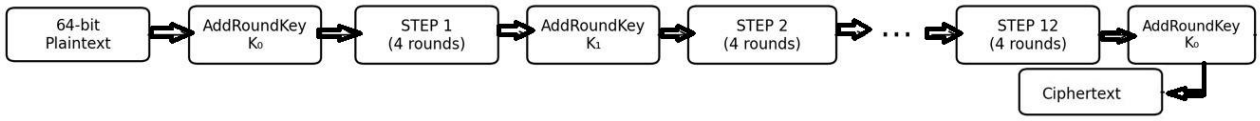


Figure 1. Encryption operation of the LED-128 block cipher.

Fig. 1 shows the process to encrypt data - LED doesn't have a key schedule. A step has 4 rounds. Every round has 4 operations inside it. LED-128 has twelve steps. Each step has 4 rounds. The cipher has 48 rounds. The user adds the key at the start and at the end of each step. Because of this there are thirteen AddRoundKey operations.

1. Add Constants (AC). The round constants are Xored with the state.
2. Sub Cells (SC). Each nibble in the state is updated by a lookup using the S-Box of PRESENT .
3. Shift Rows (SR). Row i of the state is rotated i distance to the left, for i = 0, 1, 2, 3.
4. MixColumnsSerial (MC). Each column of the state is updated by the multiplication of itself with the corresponding column of a fixed Matrix M.

$$M = \begin{pmatrix} 4 & 1 & 2 & 2 \\ 8 & 6 & 5 & 6 \\ B & E & A & 9 \\ 2 & 2 & F & B \end{pmatrix}$$

The four transformations are applied sequentially in each round: AC, SC, SR, and MC. The PRESENT S-Box remains part of the original LED SubCells operation in both LED-64 and LED-128[14].

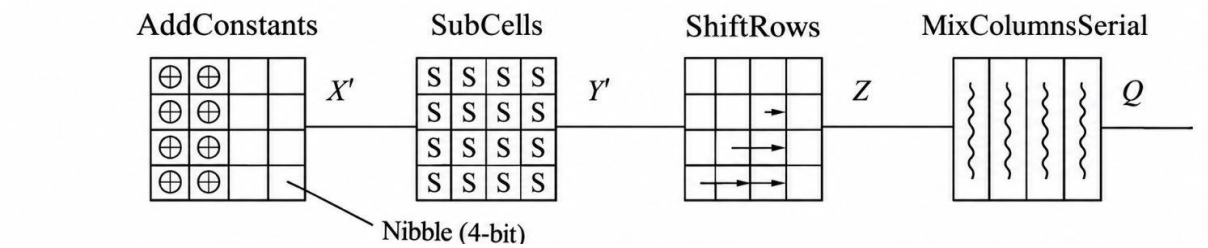


Figure 2. The four transformations in one LED round.

2.2 Chaotic Map

A chaotic map is a dynamical system in mathematics - This system changes a lot when its starting numbers change even slightly and the resulting numbers are very different than the original ones. People use it in cryptography because it makes keys less predictable and makes data have more random qualities [15].

The Henon map

The Henon map is a two dimensional dynamical system that has discrete time steps it is amongst the two dimensional systems that have the fewest parts whilst still showing chaotic behavior. defined as taking point (x_n,y_n) and mapping to new point[16]:

$x_n+1=1-axn2+y_n$ …..(1)

$y_n+1=bx_n$ ……(2)

classic henon attractor with parameters a=1.4 and b=0.3

3. The Proposed HD-SBKS Key Scheduling Mechanism

were present a way to schedule keys in the LED-128 block cipher, called the Hénon-Driven Dynamic S-Box Key Schedule (HD-SBKS), As shown in Figure 3 and Algorithm 1. The goal is to expand keys - transforming them in ways that depend on the key itself and aren't linear; that the round keys aren't identical to one another. The original cipher has problems with how it uses keys. We fix those problems - using a Hénon map to make not predictable intermediate values, dynamic S-boxes made by affine transformations over GF (2^4) and a 64-bit Feistel network to make round keys. The standard LED-128 cipher has an existing way to schedule keys. In this work the way to expand keys is different but the steps inside the cipher that encrypt and decrypt data are the same as before. This change is only in the key schedule plus doesn't alter the base structure that encrypts data in LED-128. The HD-SBKS method has several steps - initializing the master key, making intermediate words with the Hénon map - which builds dynamic S-boxes, expanding the round keys with a Feistel network and combining those steps with the standard LED-128 cipher. Those parts make thirteen round keys of 64 bits each and those keys are used to encrypt and decrypt data.

3.1 Master Key Initialization

The 128-bit master key is divided into four 32-bit words:

$$MK = K_0 \parallel K_1 \parallel K_2 \parallel K_3 \quad \ldots(3)$$

The initial values of the Hénon map are derived from the master key as follows:

$$u = K_0 \oplus ROTL_{32}(K_2, 7) \oplus 0x48454E4F \quad \ldots..(4)$$

$$v = K_1 \oplus ROTL_{32}(K_3, 13) \oplus 0x4E324453 \ldots\ldots(5)$$

The initial state is represented using the Q32.32 fixed-point format, where:

$$S = 2^{32} \quad \ldots\ldots()$$

$$X_0 = \lfloor (3Su) / 2^{32} \rfloor - 3S/2 \ldots\ldots(6)$$

$$Y_0 = \lfloor (Sv) / 2^{32} \rfloor - S/2 \quad \ldots\ldots(7)$$

Here, X_0 and Y_0 represent the initial state variables of the key-dependent map.

3.2 Hénon-Based Intermediate Word Generation

The Hénon map is employed to generate deterministic key-dependent sequences:

$x_{n+1} = 1 - 1.4x_n^2 + y_n$ ……(8)

$y_{n+1} = 0.3x_n$ ……(9)

For fixed-point implementation, the corresponding integer equations are:

$X_{n+1} = S - \text{trunc}(7X_n^2 / (5S)) + Y_n$ ……(10)

$Y_{n+1} = \text{trunc}(3X_n / 10)$ ……(11)

To make the results reproducible, implements this must state exactly how the signed arithmetic works, how wide the intermediate multiplication is and what happens during an overflow. Once the first 64 iterations are complete and left out, each state that follows has two 32-bit words that are taken from it

$W_{2j} = (X_j \gg 16) \,\&\, (2^{32} - 1)$ ……(12)

$W_{2j+1} = (Y_j \gg 16) \,\&\, (2^{32} - 1)$ ……(13)

The process generates 52 intermediate words, which are used to construct thirteen dynamic S-boxes.

3.3 Dynamic S-box Construction

make S-boxes that change so that the key schedule has nonlinear transformations that depend on the key. To do this we have the first PRESENT S-box and affine transformations over GF(2^4).

$S_p = (C, 5, 6, B, 9, 0, A, D, 3, E, F, 8, 4, 7, 1, 2)$ ……(14)

Let GL(4, 2) denote the group of invertible 4 × 4 binary matrices. Its cardinality is:

$|\text{GL}(4, 2)| = 20160$ ……(15)

For each round index i, where $0 \le i \le 12$, four intermediate words are used to determine the transformation parameters:

$B_i = \text{GL}_4[W_{4i} \bmod 20160]$ ……(16)

$A_i = \text{GL}_4[W_{4i+1} \bmod 20160]$ ……(17)

$b_i = W_{4i+2} \bmod 16$ ……(18)

$a_i = W_{4i+3} \bmod 16$ ……(19)

The resulting dynamic S-box is defined as:

$S_i^{Ks}(z) = A_i[S_p(B_i(z \oplus b_i))] \oplus a_i$ ……(20)

All matrix operations are performed over GF (2).

Since A_i and B_i are invertible matrices and the PRESENT S-box is bijective, each generated dynamic S-box is also bijective.

Its inverse is given by:

$(S_i^{Ks})^{-1}(z) = B_i^{-1}[S_p^{-1}(A_i^{-1}(z \oplus a_i))] \oplus b_i$ ……(21)

Implementation requirement: The enumeration order of GL(4, 2) must be fixed and published so that the generated S-boxes can be reproduced exactly.

3.4 Feistel-Based Round Key Expansion

The 128-bit master key is divided into two 64-bit halves:

$$MK = K_L \parallel K_R \quad \ldots\ldots(22)$$

The initial Feistel state is:

$$X_0 = K_L, \quad Y_0 = K_R \quad \ldots\ldots(23)$$

For each iteration i, a round constant is calculated as:

$$C_i = (i + 1) \cdot 0x9E3779B97F4A7C15 \bmod 2^{64} \quad \ldots\ldots(24)$$

The rotation amount is:

$$r_i = (5 + 7i) \bmod 64 \quad \ldots\ldots(25)$$

The nonlinear input is computed as:

$$T_i = ROTL_{64}(Y_i, r_i) \oplus C_i \quad \ldots\ldots(26)$$

The 64-bit value T_i is divided into sixteen 4-bit nibbles:

$$T_i = t_{i,0} \parallel t_{i,1} \parallel \cdots \parallel t_{i,15} \quad \ldots\ldots(27)$$

Each nibble is processed using the corresponding dynamic S-box:

$$U_i = S_i^{Ks}(t_{i,0}) \parallel \cdots \parallel S_i^{Ks}(t_{i,15}) \quad \ldots\ldots(28)$$

The Feistel round function is defined as:

$$F_i(Y_i) = U_i \oplus ROTL_{64}(U_i, 17) \oplus ROTL_{64}(U_i, 41) \oplus C_i \quad \ldots\ldots(29)$$

The Feistel state is updated according to:

$$X_{i+1} = Y_i \quad \ldots\ldots(30)$$

$$Y_{i+1} = X_i \oplus F_i(Y_i) \quad \ldots\ldots(31)$$

Finally, the 64-bit round key is generated as:

$$RK_i = X_i \oplus ROTL_{64}(Y_i, 17) \oplus C_i \quad \ldots\ldots(32)$$

Repeating this procedure for i = 0, …, 12 produces thirteen 64-bit round keys:

$$RK_0, RK_1, RK_2, \ldots, RK_{12} \quad \ldots\ldots(33)$$

3.5 Integration with LED-128

The generated round keys are incorporated into the original LED-128 encryption structure without modifying its internal round transformations.

The modified encryption process is expressed as:

$$S_0 = P$$

$$S_{i+1} = Step_i(S_i \oplus RK_i), \quad 0 \le i \le 11 \quad \ldots\ldots(34)$$

The ciphertext is obtained by the final key addition:

$$C = S_{12} \oplus RK_{12} \quad \ldots\ldots(35)$$

Here, $Step_i$ represents the original LED-128 internal transformation, including its prescribed round operations.

The decryption process begins with:

$S_{12} = C \oplus RK_{12}$ ……(36)

Then, the intermediate states are recovered in reverse order:

$S_i = Step_i^{-1}(S_{i+1}) \oplus RK_i, \quad i = 11, \ldots, 0$ ……(37)

Finally:

$$P = S_0$$

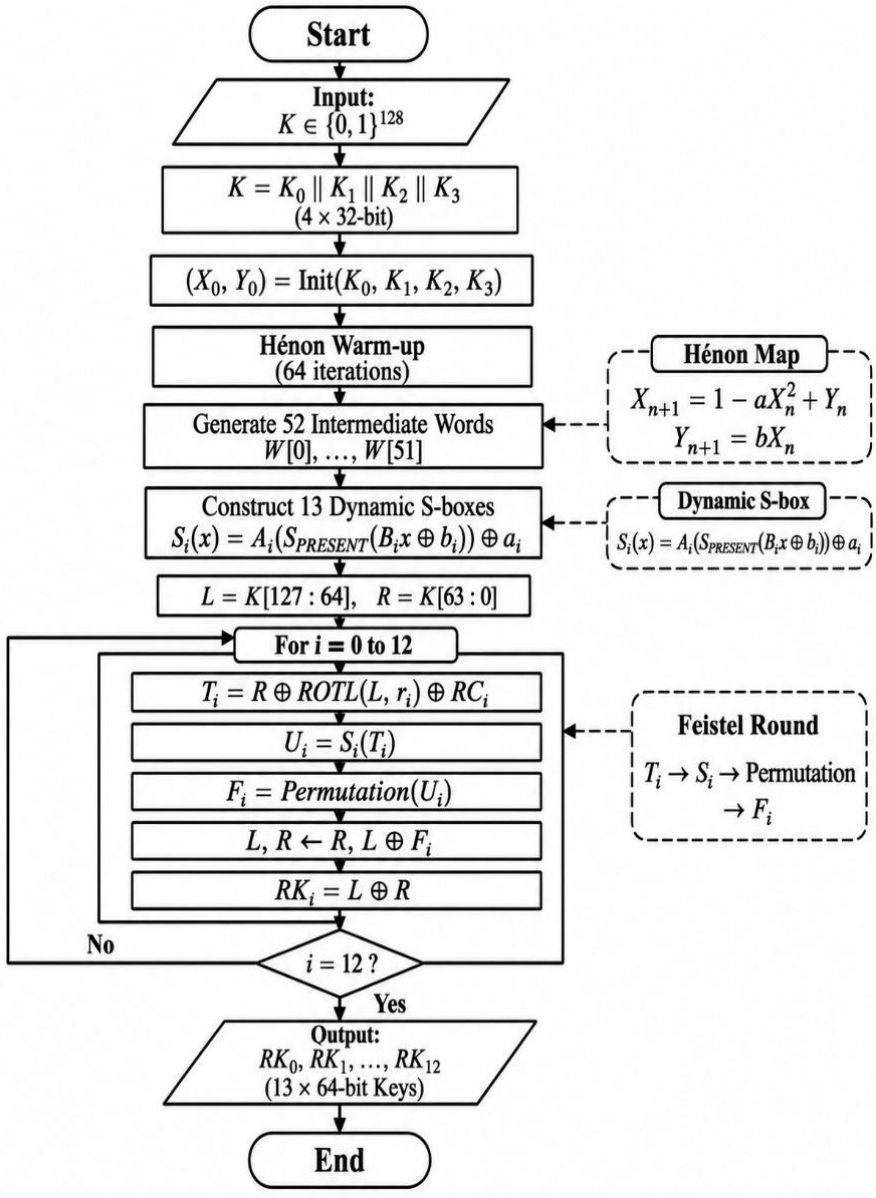


Figure 3 **Proposed enhanced LED128 Algorithm**

Algorithm 1. Proposed HD-SBKS Key Generation

Input: Master Key $K \in \{0,1\}^{128}$

Output: Round Keys $RK[0...12] \in \{0,1\}^{64}$

Step 1: Start.

Step 2: Divide the master key into four 32-bit words:

$(K_0, K_1, K_2, K_3) \leftarrow Split(K, 32)$

Step 3: Initialize the key-dependent Hénon map:

$(X_0, Y_0) \leftarrow Initialize\ (K_0, K_1, K_2, K_3)$

Step 4: Apply the Hénon warm-up phase:

$(X, Y) \leftarrow HenonWarmup(X_0, Y_0, 64)$

Step 5: Generate 52 intermediate words:

For j = 0 to 25:

$(X, Y) \leftarrow Henon(X, Y)$

$W[2j] \leftarrow Extract32(X)$

$W[2j+1] \leftarrow Extract32(Y)$

End for

Step 6: Construct 13 dynamic S-boxes:

For i = 0 to 12:

$A_i \leftarrow GL_4(W[4i])$

$B_i \leftarrow GL_4(W[4i+1])$

$a_i \leftarrow Extract4(W[4i+2])$

$b_i \leftarrow Extract4(W[4i+3])$

$S_i(x) \leftarrow A_i(S_PRESENT(B_i \cdot x \oplus b_i)) \oplus a_i$

End for

Step 7: Initialize the Feistel structure:

$(L, R) \leftarrow Split(K, 64)$

Step 8: Generate 13 round keys using Feistel key expansion:

For i = 0 to 12:

$RC_i \leftarrow RoundConstant(i)$

$r_i \leftarrow Rotation(i)$

$T \leftarrow R \oplus ROTL64(L, r_i) \oplus RC_i$

$U \leftarrow S_i(T)$

$F \leftarrow Permutation(U)$

$L_new \leftarrow R$

$R_new \leftarrow L \oplus F$

$L \leftarrow L_new$

$R \leftarrow R_new$

$RK[i] \leftarrow L \oplus R$

End for

Step 9: Return the generated round keys:

$Output \leftarrow RK[0], RK[1], ..., RK[12]$

Step 10: End.

4. The Result

were put the IoT cryptosystem into a Raspberry Pi device using Python code. We got data from connected sensors and ran it through the Enhanced LED-128 algorithm - this algorithm has 64-bit data blocks and a 128-bit master key. We tested how safe the algorithm is and how it handles numbers plus we compared it to the original LED-128 algorithm. To do this we used 4 methods. We used the NIST Statistical Test Suite to see if the cipher text has random patterns. We used the Pearson Correlation Coefficient to see how the plaintext connects to the cipher text. We used Hamming Weight to count the bits in the round keys and we tested the Avalanche Effect to see how the output changes when the master key is different. We did those tests to see how the algorithm works and how it is safer.

4.1 NIST Statistical Test Suite Analysis

use multiple statistical methods to check if cryptographic systems and their cipher texts are random and secure [17]. The National Institute of Standards & Technology (NIST) statistical test suite is one of those methods. Many people use this suite - it is a standard tool to test how random cryptographic keys and cipher texts are. The suite has 15 statistical tests that look at different parts of randomness [18]. In this study we had the NIST test suite to see plus compare how random the proposed Enhanced LED-128 algorithm is compared to the original LED-128 cipher. The dataset for the experiment was 1 000 bits. Those bits were in blocks that were 64 bits in size (m = 64) and we had different encryption keys. Table 1 has the results from the 15 NIST statistical tests on the cipher texts that came from the proposed Enhanced LED-128 algorithm and the original LED-128 algorithm.

Table 1. The NIST Suite Analysis Results for the original and enhanced algorithms

NIST Analysis	LED128	Our work
Frequency Test	0.12991	0.24987
Block Frequency	0.07010	0.26065
Non Overlapping	0.23991	0.64098
Approximate Entropy	0.01601	0.13098
Longest Run	0.28010	0.29086
Rank Test	0.01999	0.04986
FFT Test	0.01001	0.41009
Overlapping Template	0.01250	0.42080
Universal Test	0.01202	0.34703
Random Excursions	0.55897	0.66091
Cumulative Sums	0.45908	0.52681
Runs Test	0.16846	0.20832
Random Excursions Variant	0.2093	0.60114
Serial Test	0.06998	0.25093
Linear Complexity	0.24098	0.73425

Table 1 has the P-value results of the NIST statistical tests for the bit sequences that the proposed Enhanced LED-128 algorithm and the original LED-128 algorithm made. We assessed randomness using a significance level (α) of 0.01. Each NIST test has a P-value. We use this value to see if the tested bit sequence meets the criteria for randomness. A sequence passes the test when its P-value is 0.01 or more. It fails the test when the value is less than 0.01. The proposed Enhanced LED-128 algorithm had P-values that were larger in number in most of the NIST tests and all 15 NIST tests had values that were above 0.01. The original LED-128 algorithm had P-values that were smaller in number in some

tests. Because of this the proposed enhanced algorithm is more random than the original algorithm when it encrypts sensor data sequences.

4.2 Correlation Coefficient Analysis (CCA)

used Correlation Coefficient Analysis (CCA) to measure how plaintext and ciphertext relate to each other statistically and to check how the encryption algorithms confuse and diffuse data[19],[20]. In this study we test both the first version of the LED-128 algorithm plus the updated LED-128 algorithm in the same setting, with the same inputs of plaintext and the same key. The test has ten blocks of data, from Block 1 to Block 10. For each block we find the Pearson correlation coefficient between the matching plaintext and ciphertext numbers. Table 2 has the Correlation Coefficient Analysis results for LED-128 and the updated LED-128.

Table 2. Correlation Coefficient Analysis results for the original and enhanced algorithms

Block	LED-128	Our work
Block 1	0.032991	0.007902
Block 2	0.021106	0.013012
Block 3	0.029875	0.005096
Block 4	0.030001	0.008096
Block 5	0.037901	0.011990
Block 6	0.023889	0.006098
Block 7	0.039870	0.010021
Block 8	0.026991	0.005097
Block 9	0.035987	0.007865
Block 10	0.031112	0.008975
Average	**0.030972**	**0.008415**

Table 2 shows the original LED-128 algorithm has an average correlation coefficient of 0.030972 and the proposed Enhanced LED-128 algorithm has an average correlation coefficient of 0.008415, which is smaller. Because of this the proposed Enhanced LED-128 algorithm links the plaintext and cipher text less linearly than the original LED-128 algorithm does. When the average correlation coefficient is lower, the input and the encrypted data are more statistically independent. When we analyze the correlation coefficient, the proposed Enhanced LED-128 algorithm is better at making the linear correlation between plaintext plus cipher text smaller. This fact shows that its statistical encryption characteristics are acceptable.

4.3 Hamming Weight Test

The Hamming Weight (HW) test is a method to check how zeros and ones are spread out in the round keys that a key scheduling algorithm makes. Imdad et al. (2022) stated that a properly made key schedule has round keys with an equal count of zeros and ones, even if the input secret key doesn't have an equal count of bits. an n-bit round key has a target Hamming weight of n/2, which means that half of its bits are ones [21], [22]. In this study the Hamming Weight test is used on the thirteen 64-bit round keys that the proposed Hénon-Driven Dynamic S-Box Key Schedule (HD-SBKS) makes for the LED-128 block cipher. The Hamming weight of each round key is the count of its non-zero bits and those counts are compared to 32. In addition, the total Hamming weight percentage is calculated to see how equal the bits are across the round keys. When values are near 50 %, the count of zeros plus ones is equal, which shows how the bits behave statistically in the proposed key schedule.

$$\text{HW}(\%)=\frac{\text{Total Hamming Weight}}{\text{Total Number of Bits}} * 100 \quad \ldots\ldots(38)$$

Table 3 Hamming Weight Test results for the original and enhanced algorithms

Round	Original	Our work
RK_0	20	31
RK_1	25	33
RK_2	29	30
RK_3	31	34
RK_4	18	32
RK_5	34	29
RK_6	27	35
RK_7	22	31
RK_8	35	33
RK_9	24	30
RK_{10}	30	34
RK_{11}	19	32
RK_{12}	32	32
Total HW	**346**	**416**
Average HW	**26.62**	**32.00**
HW Percentage	**41.59%**	**50.00%**

Table 3 shows the results - HD-SBKS, which we propose, produces round keys where the binary bits are spread out more evenly than the bits that the original LED-128 key schedule produces. Our method has an average Hamming weight of 32 - this number is 50 % of the key bits in total - this is the target state where 64-bit round keys have an equal number of ones and zeros.

4.4 Avalanche Effect Analysis

The Avalanche Effect (AE) is used to assess the sensitivity of a block cipher to a small change in its input, particularly the secret key or plaintext [23], [24]. In this study, key sensitivity is considered because the proposed HD-SBKS modifies the LED-128 key-scheduling mechanism while retaining the internal encryption transformations. The evaluation was conducted using 15 different 64-bit plaintext blocks. For every block the plaintext was the same and we changed each of the 128 master key bits one at a time. We looked at the new ciphertext next to the first ciphertext that came from the first master key. We used the Hamming distance to see how many ciphertext bits were different and we figured out the avalanche percentage in this way:

$$\mathrm{AEK_K} = \frac{\mathrm{HW(C1 \oplus C2)}}{64} \times 100 \quad \ldots\ldots(39)$$

In this context C1 = E (K, Bi) is the reference ciphertext and C2 = E(K $\oplus$ 2^j, Bi) is the ciphertext after bit j of the 128-bit master key is changed. HW is the Hamming weight and the block size is 64 bits. It is good when the AE value is near 50 %, which is about 32 changed bits out of 64 bits. Table 4 compares the original LED-128 and the new HD-SBKS. The mean AE is 45.28% for original LED-128 and 49.11% for HD-SBKS - which means the value for the new method is nearer to 50 %. The result matches how the system was planned, because the key dependent Hénon sequence, dynamic S-boxes and Feistel based expansion make the generated round keys depend more on the master key.

Table 4. Avalanche Effect results for the original and enhanced algorithms

Data	Original LED	Our work
B1	44.35	49.11
B2	45.99	47.14
B3	45.99	50.51
B4	47.34	50.01
B5	43.28	48.55
B6	46.98	49.01
B7	46.01	47.99
B8	44.55	51.06
B9	47.04	49.20
B10	47.11	47.99
B11	44.99	49.20
B12	43.12	50.16
B13	46.08	49.09
B14	43.55	49.13
B15	42.95	48.55
Average	45.28	49.11

5. Conclusion

Were made a key scheduling mechanism called the Hénon-Driven Dynamic S-Box Key Schedule (HD-SBKS) so that the LED-128 block cipher works better. This mechanism has a Hénon chaotic map that depends on the key, S-boxes that we make from affine transformations over GF(2^4) and a 64-bit Feistel key expansion structure. It makes thirteen round keys that aren't the same as each other. In our tests the cipher worked better than the original LED-128 algorithm. The cipher passed all 15 NIST tests but the original algorithm didn't pass two tests. The average correlation coefficient was 0.008415 instead of 0.030972, which means the plaintext and ciphertext have less to do with each other mathematically. The average Hamming weight went up from 26.62 bits to 32 bits, which is exactly half of the bits. The avalanche effect was 49.11% instead of 45.28%, which is close to the expected 50 %. Those results show the HD-SBKS has numbers closer to ideal statistical values, reacts more when the key changes and spreads round keys more evenly than the original LED-128 key schedule. The enhancement is still the same as it was before; people can use it for cryptography on small devices, e.g. hardware for the Internet of Things that doesn't have much computing power or memory.

References

[1] S. Kumar, D. Kumar, R. Dangi, G. Choudhary, N. Dragoni, and I. You, “A Review of Lightweight Security and Privacy for Resource-Constrained IoT Devices,” Computers, Materials & Continua, vol. 78, no. 1, pp. 31–63, 2024. DOI: https://doi.org/10.32604/cmc.2023.047084

[2] M. Rana, Q. Mamun, and R. Islam, “Lightweight Cryptography in IoT Networks: A Survey,” Future Generation Computer Systems, vol. 129, pp. 77–89, 2022.

[3] P. S. Suryateja and K. V. Rao, “A Survey on Lightweight Cryptographic Algorithms in IoT,” Cybernetics and Information Technologies, vol. 24, no. 1, pp. 21–34, 2024. DOI: https://doi.org/10.2478/cait-2024-0002

[4] S. A. Ansari and S. Ali, "A Systematic Review of Lightweight Cryptographic Schemes for Security and Privacy in IoT," Discover Computing, vol. 28, Art. no. 266, 2025. DOI: https://doi.org/10.1007/s10791-025-09755-3

[5] NIST, Ascon-Based Lightweight Cryptography Standards for Constrained Devices: Authenticated Encryption, Hash, and Extendable Output Functions, NIST SP 800-232, 2025.

[6] A. Mhaouch, W. Ayadi, S. Ridha, K. Issa, A. Ben Abdelali, and M. Machhout, "An efficient hardware implementation of LED lightweight block cipher," in Proc. 7th IEEE Int. Conf. Advanced Technologies, Signal and Image Processing (ATSIP), 2024, pp. 312–316, doi: 10.1109/ATSIP62566.2024.10638985.

[7] A. Mhaouch, W. Elhamzi, A. Ben Abdelali, and M. Atri, "Efficient design for a hardware implementation of the LED block cipher," Journal Européen des Systèmes Automatisés, vol. 56, no. 5, pp. 725–733, Oct. 2023, doi: 10.18280/jesa.560502.

[8] M. Al-Shatari, F. A. Hussin, A. Abd Aziz, T. A. E. Eisa, X.-T. Tran, and M. E. E. Dalam, "IoT edge device security: An efficient lightweight authenticated encryption scheme based on LED and PHOTON," Applied Sciences, vol. 13, no. 18, Art. no. 10345, Sep. 2023, doi: 10.3390/app131810345.

[9] P. Modi, P. Singh, and B. Acharya, "Effective hardware architectures for LED and PRESENT ciphers for resource-constrained applications," International Journal of High Performance Systems Architecture, vol. 10, no. 2, pp. 89–104, 2021, doi: 10.1504/IJHPSA.2021.119151.

[10] W. El Hadj Youssef, A. Abdelli, F. Dridi, and M. Machhout, "Hardware implementation of secure lightweight cryptographic designs for IoT applications," Security and Communication Networks, vol. 2020, Art. no. 8860598, pp. 1–13, 2020, doi: 10.1155/2020/8860598.

[11] A. Patil, G. Bansod, and N. Pisharoty, "Hybrid lightweight and robust encryption design for security in IoT," International Journal of Security and Its Applications, vol. 9, no. 12, pp. 85–98, 2015, doi: 10.14257/IJSIA.2015.9.12.10.

[12] A. Bogdanov et al., "PRESENT: An Ultra-Lightweight Block Cipher," in Proc. Cryptographic Hardware and Embedded Systems (CHES), 2007, pp. 450–466. DOI: https://doi.org/10.1007/978-3-540-74735-2_31

[13] J. Guo, T. Peyrin, A. Poschmann, and M. Robshaw, "The LED Block Cipher," in Proc. Cryptographic Hardware and Embedded Systems (CHES), 2011, pp. 326–341.

[14] G. Hatzivasilis, K. Fysarakis, I. Papaefstathiou, and C. Manifavas, "A Review of Lightweight Block Ciphers," Journal of Cryptographic Engineering, vol. 8, no. 2, pp. 141–184, 2018.

[15] M. Hénon, "A Two-Dimensional Mapping with a Strange Attractor," Communications in Mathematical Physics, vol. 50, pp. 69–77, 1976. DOI: https://doi.org/10.1007/BF01608556

[16] R. B. Naik and U. Singh, "A Review on Applications of Chaotic Maps in Pseudo-Random Number Generators and Encryption," Annals of Data Science, vol. 11, no. 1, pp. 25–50, 2024. DOI: https://doi.org/10.1007/s40745-021-00364-7

[17] L. E. Bassham et al., "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," NIST Special Publication 800-22 Rev. 1a, 2010. DOI: https://doi.org/10.6028/NIST.SP.800-22r1a

[18] National Institute of Standards and Technology, "Decision to Revise NIST SP 800-22 Rev. 1a," NIST, Apr. 2022. Available: https://csrc.nist.gov/News/2022/decision-to-revise-nist-sp-800-22-rev-1a csrc.nist.gov+1

[19] "A Difference-Equation-Based Robust Image Encryption Scheme with Chaotic Permutations and Logic Gates," Journal of Mathematical Imaging and Vision, 2022. DOI: https://doi.org/10.1007/s10851-022-01099-7

[20] "Image Encryption Algorithms: A Survey of Design and Evaluation Metrics," Cryptography, vol. 4, no. 1, Art. no. 7, 2024. link.springer.com+1

[21] S. Afzal, M. Yousaf, H. Afzal, N. Alharbe, and M. R. Mufti, "Cryptographic Strength Evaluation of Key Schedule Algorithms," Security and Communication Networks, vol. 2020, Art. no. 3189601, 2020. DOI: https://doi.org/10.1155/2020/3189601

[22] M. Imdad, S. N. Ramli, and H. Mahdin, "An Enhanced Key Schedule Algorithm of PRESENT-128 Block Cipher for Random and Non-Random Secret Keys," Symmetry, vol. 14, no. 3, Art. no. 604, 2022. DOI: https://doi.org/10.3390/sym14030604 Wiley Online Library+1

[23] D. K. Biss, "A Lower Bound on the Number of Functions Satisfying the Strict Avalanche Criterion," Discrete Mathematics, vol. 185, nos. 1–3, pp. 29–39, 1998. DOI: https://doi.org/10.1016/S0012-365X(97)00180-5

[24] E. J. Madarro-Capó, E. C. R. Piñón, G. Sosa-Gómez, and O. Rojas, "Practical improvement in the implementation of two avalanche tests to measure statistical independence in stream ciphers," *Computation*, vol. 12, no. 3, Art. no. 60, 2024, doi: 10.3390/computation12030060.

About Author



Dr. Farah Neamah Abbas is an academician and researcher in the field of Computer Science, specializing in Information Security, Cryptography, and Machine Learning. She completed her Bachelor of Science in Computer Science from Al-Mustansiriyah University, Baghdad, Iraq, and earned her Master of Science in Computer Science from the Iraqi Commission for Computers and Information in 2014. She has been serving as a faculty member in the Department of Computer Science, Faculty of Education, Mustansiriyah University, Iraq, since 2014.

Her research interests include data security, encryption algorithms, Internet of Things (IoT), image encryption, and machine learning applications. She has authored and contributed to several research publications in international journals covering areas such as cryptography, network security, air-quality monitoring, and artificial intelligence. She has also actively participated in academic committees, research conferences, scientific workshops, and professional development activities, contributing to teaching, research, and academic administration.