Future Technology
Open Access Journal
ISSN 2832-0379

May 2026| Volume 05 | Issue 02 | Pages 60-68

Journal homepage: https://fupubco.com/futech

https://doi.org/10.55670/fpll.futech.5.2.6

Future Publishing LLC

Article

A Feistel-based modification of the PRESENT lightweight block cipher

Ghassan Salloom*, Karam Mohammed, Shrooq Hameed, Khanssaa Abdul Majed

Scientific Research Commission, Baghdad, Iraq

ARTICLE INFO

Article history:
Received 18 September 2025
Received in revised form
20 November 2025
Accepted 13 January 2026

Keywords:
Feistel, Avalanche, Complexity, Security

*Corresponding author
Email address:
ghassan.kh.salloom@src.edu.iq

DOI: 10.55670/fpll.futech.5.2.6

ABSTRACT

The PRESENT block cipher is commonly used in constrained devices, such as IoT nodes, low-power wireless sensors, and RFID tags, due to its low power consumption, small hardware footprint, and acceptable security margin. However, PRESENT's substitution layer depends on a fixed S-box, which can reduce resistance against specific analytical attacks (linear and differential analysis). In the proposed version, the static S-box is replaced with a dynamic, key-dependent S-box created by rounds of a lightweight Feistel network. The modified variant eliminates repetitive patterns in the substitution layer. In this study, we evaluate the security strength of the proposed method using differential analysis, linear analysis, avalanche rate, algebraic interpolation attacks, and integral/square attacks. Additionally, we measure the time and hardware complexity. The results demonstrate that the introduced version achieves an average avalanche value of 30.92 flipped bits with a standard deviation of 4.7, indicating strong diffusion with increased variability compared to PRESENT, while maintaining moderate computational and hardware complexity. The improved security of the introduced scheme is attained at the expense of a modest increase in hardware area, consistent with current lightweight cipher design trade-offs.

1. Introduction

Numerous well-established block ciphers like AES, 3DES, and Blowfish provide strong security. However, they often require more hardware, energy, and processing time than tiny embedded devices can typically allocate. Due to this mismatch, they are challenging to deploy on systems such as RFID tags, low-power wireless sensors, and many IoT nodes. This issue has motivated the development of lightweight cryptography algorithms designed specifically to deliver reasonable security while maintaining hardware cost and computational load as small as possible. PRESENT is a lightweight block cipher based on a substitution-permutation network (SPN). It was designed in 2007 by [1], which is suitable for constrained environments (e.g., RFID tags, network sensors), where hardware efficiency was one of its key design targets. The sizes are approximately 1570 GE for 80-bit keys and approximately 1884 GE for 128-bit keys. It is considered one of the few realistic options in such constrained environments and provides an acceptable level of security through many rounds of diffusion and confusion. This cipher consists of 31 rounds, the block length is 64 bits, and it supports two variant key lengths: 80 and 128 bits. There are four main core functions of operations in this cipher performed in every round: key scheduling, adding round key, S-box substitution, and permutation. The key schedule extracts a 64-bit key from an 80-bit key (or from a 128-bit key) and stores it in the state (i.e., the key register). The AddRoundKey function XORs the output of the last round (e.g., plaintext in the first round) with the round key for the current round, generated by the key schedule. The substitution layer (confusion layer) is a key component that provides non-linearity, enhancing the cipher's resistance to cryptanalysis. In the substitution layer, the static S-box takes a 4-bit input and maps it to a unique 4-bit output, according to a predefined lookup table. The final round is the permutation layer (diffusion layer), where this rearranges the 64 output bits from the S-box layer based on a predefined fixed pattern permutation. Finally, when all 31 rounds have completed, the output is once more XORed with a round key for the purpose of key whitening. Figure 1 shows the confusion and diffusion rounds of the PRESENT block cipher. On the other hand, a great volume of research has focused on improving the security of PRESENT without significantly increasing hardware requirements. Various approaches employ dynamic S-boxes or key-dependent components to enhance confusion and randomness. A. Abdelli et al. [2] introduced a modified PRESENT lightweight cipher that uses chaotic systems based on a logistic map, aimed at improving

its strength and randomness while preserving the simplicity and compactness of the original cipher.

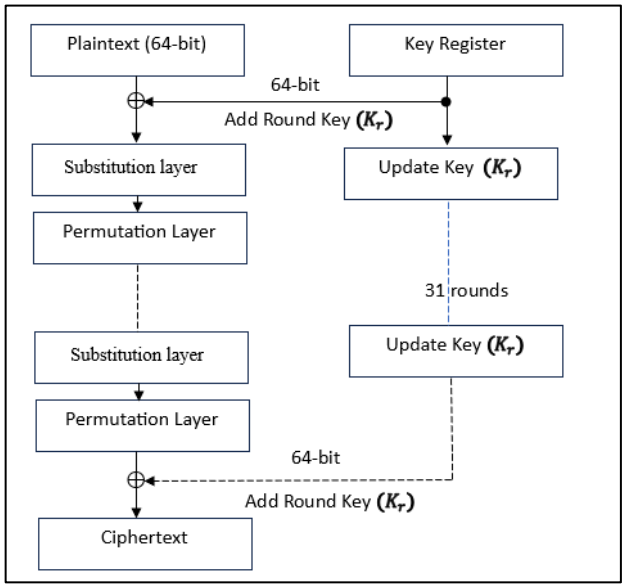

Figure 1. Block diagram of the PRESENT cipher

S. Salim et al. [3] introduced a new version of the key-dependent S-box; this technique relies on selecting one S-box out of 16 effective S-boxes. The security analysis showed that this method is more secure against linear and differential cryptanalysis. Moreover, this approach reduced the complexity of the proposed S-box to 21 per round, whereas the PRESENT S-box has a complexity of 28 per round. Based on FPGA implementation to enhance PRESENT cipher security, Ref. [4] presented a new design that uses a Substitution-Permutation (SP) network, with permutation layers that increase diffusion and confusion to resist unauthorized access. In this context, Ref. [5] presented an improved fault-detection scheme for the PRESENT lightweight block cipher, based on parity checking. The proposed scheme focuses on detecting both simple and multi-fault attacks, addressing scenarios targeting one or more bytes.

Another area of research focuses on chaotic systems due to their properties, such as nonlinear dynamics, high sensitivity to initial conditions, and intrinsic randomness, which have emerged as a promising technique for improving security in lightweight designs by incorporating chaotic systems into existing ciphers. In this regard, Ref. [6] introduced a five-dimensional system designed for lightweight cryptography. The proposed system achieved high randomness and complexity, as confirmed by NIST statistical evaluations. Furthermore, Ref. [7] transformed the PRESENT lightweight cipher into D-PRESENT to improve avalanche rate, execution time, and throughput, and to reduce vulnerability to key linear and differential cryptanalysis. The results demonstrate that D-PRESENT outperforms the PRESENT cipher. Another research direction focuses on improving the security of the PRESENT cipher. Ref [8] introduces dynamic-keyed permutation networks that are directly integrated into PRESENT-80. This study demonstrates that PRESENT-80 is vulnerable to attacks that do not rely on side-channel techniques and that the proposed method further strengthens its resilience and prevents such attacks.

Furthermore, Ref. [9] uses a 4 × 4 S-box from PRESENT and reduces the number of rounds from 31 to 25 to improve software efficiency. Additionally, the introduced version has a PRESENT-like structure, with the number of layers reduced from 4 to 3 to increase efficiency and reduce cost. Extensive security analysis showed that the highest number of rounds that can be attacked is 18 (out of 25). In this field, PRESENT is a notable solution because it strikes a practical balance between performance and security. However, it still has one major limitation: its S-box is fixed. Using the same nonlinear mapping in every round creates structural regularities that can be exploited in certain analytical attacks. In particular, the static S-box pattern can provide clues for differential and linear cryptanalysis, which rely on detecting predictable relationships between inputs and outputs. To overcome the limitation of the fixed substitution layer in the original PRESENT design, this work presents a key-dependent S-box generation method. The proposed approach relies on a small Feistel structure driven by a 64-bit round key. During each encryption round, the evolution of the Feistel network combined with the applied round key produces a dynamically changing substitution mapping. As a result, a distinct S-box is generated for each round, thereby enhancing confusion and diffusion, albeit at the expense of a moderate increase in hardware area (≈20% GE).

2. Proposed work

Due to the traditional substitution layer of the PRESENT block cipher, which uses a static S-box across all encryption rounds, the PRESENT cipher exhibits structural predictability that can be exploited in known- and chosen-plaintext attacks to enable differential and linear analyses [10,11]. The introduced cipher is a modified version of the PRESENT lightweight block cipher, in which the static 4-bit substitution box (S-box) is replaced with a dynamic, key-dependent S-box generated in each encryption round using a lightweight Feistel network [12-15]. The generated S-box is a permutation of the 16 possible 4-bit values, thereby improving confusion properties and providing strong resistance to differential and linear attacks, while eliminating the repetitiveness of a fixed substitution box. Moreover, the modification preserves resource-limited environments.

2.1 Mathematical model

Consider the S-box domain X to be the 4-bit space formally defined as follows:

$$X = \{0,1,\ldots,15\} \in \{0,1\}^4 \quad (1)$$

Let the round key K_r for the encryption round r be a 64-bit word.

$$K_r \in \{0,1\}^{64} \quad (2)$$

Let $k_r^{(16)}$ denote the 16-bit value derived from the round key at round r. We extract the 16-bit subkey ${k_r}^{16}$ as follows:

$${k_r}^{16} = LSB_{16bit}\ (K_r) \quad (3)$$

We derive 2-bit subkeys $k_i \in \{k_0, k_1, \cdots k_4\}$, where each k_i is used during one Feistel round. The use of a 16-bit seed provides additional entropy and avoids simple subkey repetition while preserving compact, lightweight S-box generation logic [16,17]. To initialize the Feistel function, each input $x \in X$ is decomposed into two 2-bit halves, the left half L_0 and the right half R_0, where $L_0, R_0 \in \{0,1\}^2$ as follows:

$$x = (L_0||R_0) \quad (4)$$

To improve the nonlinearity and confusion of the dynamically generated S-box, we extend the Feistel round function with a lightweight nonlinear Boolean transformation. Let a 2-bit right-half value R_0 and a 2-bit subkey k_i. The nonlinear Feistel function is formally defined as:

$$f: \{0,1\}^2 \times \{0,1\}^2 \rightarrow \{0,1\}^2 \quad (5)$$

$$Let\ u = R_0 \oplus k_i = (u_o, u_1) \quad (6)$$

$$f_0 = u_0 \oplus (u_1 \wedge u_0),\ f_1 = (u_1 \oplus u_0) \oplus (u_1 \wedge u_0) \quad (7)$$

$$\text{then } f(R, k_i) = (f_0||f_1) \quad (8)$$

The above function includes the required nonlinearity and key dependency, introduces algebraic degree two, removes linearity in the Feistel function, and improves resistance against cryptoanalysis. Finally, we generate a key-dependent S-box $y_x[0,1,\ldots,15]$ based on the following sequential steps:

$$(L_{i+1}, R_{i+1}) = (R_i, L_i \oplus f(R_i, k_i)) \quad (9)$$

where $i = 0, 1, \ldots, r_F - 1$.

The final Feistel function output is then:

$$y_x = (R_{rF}||L_{rF}) \in \{0,1\}^4 \quad (10)$$

For each input value $x \in X$, the Feistel network generates.

$$y_x = \Phi_r(x)$$

where $\Phi_r : X \rightarrow X$ is a key-dependent nonlinear transformation.
The pseudocode for generating an S-box using the Feistel network is presented in Algorithm 1.

Algorithm 1. GenerateFeistel_SBox_PerRound(Kr, rF)

```
1. subkey16 ← low16bits(Kr)
2. Function Fn (R: 2-bit, K2: 2-bit) → 2-bit:
3.    u ← R XOR K2
4.    u0 ← u & 1
5.    u1 ← (u >> 1) & 1
6.    f0 ← u0 XOR (u1 AND u0)
7.    f1 ← (u1 XOR u0) XOR (u1 AND u0)
8.   return (f1 << 1) | f0
9.   For x = 0 to 15 do
10.   L ← (x >> 2) & 0b11     // top 2 bits
11.   R ← x & 0b11            // low 2 bits
12.   for i = 0 to (rF-1) do
13.      K2 ← (subkey16 >> (2*i)) & 0b11
14.      temp ← R
15.      R ← L XOR Fn(R, K2)
16.      L ← temp
17.   y[x] ← (L << 2) | R     // 4-bit Feistel output
```

To compute the nonlinearity of the proposed function f analytically, we define:

$$u = (u_o, u_1),\ f_0 = u_o \oplus (u_o \wedge u_1) \quad (11)$$

$$f_1 = (u_o \oplus u_1) \oplus (u_o \wedge u_1) \quad (12)$$

These are Boolean functions of two variables. For $n = 2$, the maximum Boolean nonlinearity is:

$$2^{n-1} - 2^{n/2} = 2 - 1 = 1 \quad (13)$$

To simplify:

$$f_0 = u_o(1 \oplus u_1) \quad (14)$$

Truth table: $00 \rightarrow 0,\ 01 \rightarrow 0, 10 \rightarrow 1. 11 \rightarrow 0$ →this is not affine; thus, $NL(f_0) = 1$.

$$f_1 = u_1 \oplus u_0 \oplus u_o u_1 \quad (15)$$

Truth table: $00 \rightarrow 0,\ 01 \rightarrow 1, 10 \rightarrow 1. 11 \rightarrow 1$ →this is not affine; thus, $NL(f_1) = 1$.
Each coordinate of f has algebraic degree 2 and achieves the maximum possible nonlinearity for a two-variable Boolean function, i.e., $NL(f_0) = NL(f_1) = 1$.

2.2 Empirical evaluation of the Feistel round (r_F)

In this section, we perform an empirical test to evaluate the optimal number of Feistel rounds required (r_F) and preserve low computational and hardware complexity. This ensures that each S-box generation consistently achieves a high avalanche rate and acceptable differential/linear resistance. We select a small range of (r_F), specifically between 2 and 7. We generate many key-dependent S-boxes and compute the required matrices as follows:

- For each round K_r
- For each Feistel round count r_F (2 to 7)
- Generate an S-box $S_r[0,\ldots,15]$
- Compute the maximum of the differential distribution table ($\triangle_{max}$)
- Compute the maximum of the linear approximation table ($|LAT|_{max}$)
- Avalanche rate.
- Estimate the cost of hardware (GE)

Table 1 illustrates actual measured values based on the candidate number of Feistel rounds (r_F).

Table 1. Empirical evaluation of S-box quality

r_F	$\triangle_{max}$	$\|LAT\|_{max}$	Avalanche rate	GE
2	8	6	0.6	~175
3	6	5	0.51	~270
4	2.4	2.4	0.45	~ 350
5	2.4	2.4	0.43	~440
6	2.4	2.4	0.42	~ 530
7	2.4	2.4	0.42	~615

Table 1 indicates that $r_F = 4$ constitutes a practical knee point: fewer rounds lead to better diffusion, higher statistical dependence, and acceptable hardware costs. In contrast, additional rounds provide only marginal security improvements at the expense of increased hardware costs. Consequently, $r_F = 4$ was selected as a balanced configuration.

2.3 Bijectivity of the Feistel-based 4-bit S-box Lemma:

Let $F: \{0,1\}^2 \times \{0,1\}^2 \rightarrow \{0,1\}^2$ be any (possibly nonlinear and not necessarily injective) function. For a fixed sequence of 2-bit subkeys $\{k_0, k_1, \ldots, k_{r_F-1}\}$, we define ith the Feistel round transformation:

$$T_i : (0,1)^2 \times \{0,1\}^2 \rightarrow (0,1)^2 \times \{0,1\}^2 \quad (16)$$

by

$$L_{i+1} = R_i, \qquad R_{i+1} = L_i \oplus F(R_i, k_i) \quad (17)$$

Let $T = T_{r_F-1} \circ \cdots \circ T_1 \circ T_0$

Then T is a bijection on $\{0,1\}^4$. Consequently, the generated 4-bit S-box $S(x) = T(x)$ is a permutation of $\{0, \ldots, 15\}$.

Proof. Fix i. Given (L_{i+1}, R_{i+1}), we can uniquely recover (L_i, R_i). From $L_{i+1} = R_i$. We obtain $R_i = L_{i+1}$. Substitute into the $R_{i+1} = L_i \oplus F(R_i, k_i)$. Yields:

$$L_i = R_{i+1} \oplus F(L_{i+1}, k_i) \quad (18)$$

Thus, (L_i, R_i) is uniquely determined by (L_{i+1}, R_{i+1}) so that each round transformation T_i is invertible. Since the composition of invertible mappings is invertible, T is invertible and bijective on $\{0,1\}^4$. Thus S is a permutation of $\{0, 1, \ldots, 15\}$. We note that the bijectivity holds for any F, even if $F(., k_i)$ is not injective for some subkey (i.e., the collisions in F do not affect the invertibility of the Feistel mapping. We also verified bijectivity programmatically; for $r_F = 4$, all generated S-boxes were permutations of {0,...,15}, and inverse mappings successfully recovered all 16 inputs. In addition to this, in a Monte Carlo experiment with over 5,000 randomly sampled subkey sets, the bijectivity rate was 100% (Table 2).

Table 2. Bijectivity verification of Feistel-generated 4-bit S-boxes

Test	SubKeys	Permutation	Inverse
Fixed example	{00,01,10,11}	Pass	Pass
Random example	{01,01,01,01}	Pass	Pass
Monte Carlo	5,000 random sets	100% Pass	100% Pass

In Table 2, for a fixed example, the S-box generated is [6, 0, 8, 2, 9, C, F, 3, D, E, 5, 1, A, 7, 4, B]. For a random example, the S-box is [1, 4, 3, 9, 0, 5, C, D, 2, F, E, 10, 8, B, 7, 6].

2.4 Fiestel evaluation (Worked example)

The following example illustrates the construction of a key-dependent permutation S-box based on the Feistel network rounds.

Step 1: Round key

Assume the following 64-bit round key:

$$K_r = \text{0x0123456789AB00E3} \quad (19)$$

Step 2: Seed extraction

From K_r, a 16-bit round is extracted by selecting the least significant 16 bits:

$$k_r^{(16)} = LSB16(Kr) = \text{0x00E3}=(0000000011100011)_2 \quad (20)$$

Step 3: Subkey derivation

Partition $k_r^{(16)}$ into four 2-bit subkeys:

$$(k_0, k_1, k_2, k_3) = (11_2, 00_2, 10_2, 11_2) = (3,0,2,3) \quad (21)$$

Step 4: Let $r_f = 4$ and the nonlinear function:

$$\text{F(R,k)=S2[R}\oplus\text{k], S2=[2,0,3,1]} \quad (22)$$

For each $x \in \{0,1,2,\ldots,15\}$, the first input $x = 0=(0000)_2$, split into two 2-bit halves: $(L_0, R_0) = (0, 0)$. The Feistel round evolution for $x = 0$ is shown in Table 3.

Table 3. Feistel round for x=0

i	k_i	L_i	R_i	$R_i \oplus k_i$	$F(R_i, k_i)$	L_{i+1}	$R_{i+1} = L_i \oplus$F
0	3	0	0	$2 \oplus 3 = 3$	S2[3]=1	0	0⊕1=1
1	0	0	1	$1 \oplus 0 = 1$	S2[1]=0	1	0⊕0=0
2	2	1	0	$2 \oplus 2 = 2$	S2[2]=3	0	1⊕3=2
3	3	0	2	$3 \oplus 3 = 1$	S2[1]=0	2	0⊕0=0

From the above table, after 4 rounds:

$(L_4, R_4) = (2,0)$, $S(0) = ((L_4||R_4) = (1000)_2 = 8$; thus, the first element of S-box is S(0)=8.

The second input $x = 1$, $(L_0, R_0) = (00, 01)_2 = (0, 1)$, the Feistel round evolution for $x = 1$ is shown in Table 4.

Table 4. Feistel round for x=1

i	k_i	L_i	R_i	$R_i \oplus k_i$	$F(R_i, k_i)$	L_{i+1}	$R_{i+1} = L_i \oplus$F
0	3	0	1	$1 \oplus 3 = 2$	S2[2]=3	1	0⊕3=3
1	0	1	3	$3 \oplus 0 = 3$	S2[3]=1	3	1⊕1=0
2	2	3	0	$0 \oplus 2 = 2$	S2[2]=3	0	3⊕3=0
3	3	0	0	$0 \oplus 3 = 3$	S2[3]=1	0	0⊕1=1

After four rounds, we obtain:

$(L_4, R_4) = (0,1)$, $S(1) = ((L_4||R_4) = (0001)_2 = 1$. Thus, the second element of S-box is S(1)=1.

After processing all the input values ($x \in \{0,1,2,\ldots,15\}$), we obtain the complete 4-bit permutation S-box:

$$S = [8,1,E,7,A,3,C,5,9,0,F,6,B,2,D,4] \quad (23)$$

This example illustrates how a specific round key deterministically generates a permutation S-box through the introduced Feistel network mechanism. Different round keys result in various subkey sequences. This results in different S-boxes, while bijectivity is guaranteed by the Feistel network structure.

3. Performance evaluation

Hardware performance evaluation of lightweight ciphers is typically tested using FPGA and ASIC platforms to report gate equivalents, flip-flops, clock cycles, and throughput [18]. This work employs software-level benchmarking, complemented by GE estimates, for the substitution layer. A full hardware synthesis work is deferred for future work to complement the current evaluation. This section aims to estimate the performance of the modified cipher in terms of time complexity, encryption throughput, energy-oriented metric, and hardware complexity (gate equivalent (GE)). The results are then compared with the original PRESENT cipher. The practical tests were held on an MSI Notebook with an Intel(R) 13th Gen Intel(R) Core™ i7-13620H CPU 2.40 GHz and 16 GB of RAM under the 64-bit operating system Windows 11, using a .NET 6.0 environment with the Visual Studio 2017 C++ implementation.

The time complexity in terms of Big O for both encryption algorithms depends on the number of rounds, the S-box substitution operation, and the bit permutation process. The algorithm of the original PRESENT cipher performs $r = 31$ rounds with a static 4-bit S-box lookup table, resulting in an approximate complexity of $O(r \times n)$, where $n = 64$ (length of block size). In the introduced Feistel-based cipher, each round dynamically generates the substitution box during four iterations ($r_F = 4$), increasing the complexity to $O(r \times n + c)$, where c is constant and denotes the cost of the Feistel network structure. To quantify the constant c, from the modified Feistel function:

$a = u_1 \wedge u_0$ (1 AND),
$f_0 = u_0 \oplus a$ (1 XOR),
$b = u_1 \oplus u_0$ (1 XOR),
$f_1 = b \oplus a$ (1 XOR). (24)

Thus, for each f

$$C_f = 1\ AND\ + 3\ XOR \quad (25)$$

One Feistel function needs the 2-bit update

$L \oplus F \rightarrow 2\ XOR$, so:

$$C_{FeistelRound} = 1\ AND + 5\ XOR \quad (26)$$

To produce one 4-bit S-box per round, we evaluate 16 inputs across r_F rounds:

$$c = 16.r_F\ (1\ AND + 5\ XOR) \quad (27)$$

Therefore, both algorithms maintain linear complexity. However, the proposed scheme introduces a slightly higher constant factor (c) due to the Feistel network structure-based dynamic substitution. Moreover, as the size of the plaintext (N) increases, the overall time complexity becomes linear in relation to the number of blocks or the plaintext size, since PRESENT and the modified variant are block ciphers that utilize a fixed block size (64 bits).

$$T_{PRESENT} = N.O(64 \times 31) = O(N) \quad (28)$$

$$T_{Feistel} = N.O(64{\times}31 + c) = O(N) \quad (29)$$

To measure the computational efficiency of the generated S-box by the proposed modified cipher. Both the original PRESENT cipher and the modified version were implemented using C++. Each algorithm was executed for 100,000 plaintext blocks (64 bits each), and $r_F = 4$. The results of the time measurements for single-round S-box generation are shown in Table 5.

Table 5. Performance evaluation of the PRESENT and the modified version

Cipher	Execution Time (s)	Throughput (mb/s)	Cycle/ Block	S-box Generation Time (ms)
PRESENT	27.2796	0.235	654.711	-
Modified variant	32.3879	0.187	788.710	0.136

The above table shows a comparative analysis of the S-box generation time for both ciphers. The original PRESENT uses a static 4-bit S-box that is constant across all rounds and sessions. Therefore, its generation time is negligible and occurs only once during design. On the other hand, the modified variant dynamically regenerates its S-box using the Feistel network structure, producing a key-dependent nonlinear substitution. The measured results suggest that while the dynamic S-box introduces a minimal computational overhead (approximately 0.136 ms per round), the runtime overhead is negligible. In addition to computational efficiency, we quantify an energy-oriented metric based on MSI CPU power telemetry and execution time. Let P_m and P_p denote the average CPU power during the modified version execution and the PRESENT cipher execution, respectively.

$$E_{bit} = \frac{P_{avg}}{N_b} \cdot t \quad \left[\frac{j}{bit}\right] \quad (30)$$

where $N_b = 100{,}000 \times 64 = 10^6 \times 6.4\ bit$, in terms of energy μJ, we have:

$$\mu J/bit = 10^6 . \frac{P_{avg}}{N_b} \cdot t \quad (31)$$

where t is the execution time. So, plug in directly:

For PRESENT:

$$\mu J/bit = 10^6 . \frac{P_p .27.2796}{10^6 \times 6.4} \quad (32)$$

For the modified version:

$$\mu J/bit = 10^6 . \frac{P_m .32.3879}{10^6 \times 6.4} \quad (33)$$

Thus, the ratio of average CPU power becomes:

$$\frac{E_{bit,m}}{E_{bit,p}} = \frac{P_m}{P_p} . \frac{32.3879}{27.2796} \approx \frac{P_m}{P_p} \times 1.18 \quad (34)$$

Since the proposed design introduces extra key-dependent nonlinear processing, the modified cipher introduces minimal energy overhead: approximately 1.18 × P_p. Therefore, it is suitable for IoT settings where a moderate area/energy overhead is acceptable (e.g., sensor nodes and embedded controllers).

Table 6. Line-by-line primitive Table

Feistel network	Primitive count	GE primitive	Subtotal GE
Nonlinear f unit	2 AND, 3 XOR	1, 4	2×1 + 3×4 = **14**
Feistel combinational (XOR for L ⊕ F)	1 XOR	4	4
Registers for Feistel round (L, R)	4 DFF (2 bits each × 2 regs)	8	4*8 = **32**
Round counter / small	4 D-FF	8	32
Controller FSM & control multiplexers	~10 2:1 MUX (bitwise) + logic	4× 10 = 40	40
Sort	small comparators & swap logic	—	16
S-box memory	16 × 4-bit SRAM macro	—	128
S-box write / read driver	small logic overhead	—	32
Lookup path	small MUX	—	32
Margin	—	—	20
TOTAL (sequential Feistel generator)			348

Table 6 counts the hardware area. It assumes the Feistel evaluator is a single combinational f with small registers (sequential reuse for 16 inputs). The results illustrate that the hardware cost of the generated S-box using the lightweight Feistel network structure is about (340 − 360). The PRESENT cipher is cheap. The hardware cost is about 28 GE. However,

a static S-box pattern can provide an indication for differential and linear cryptanalysis. In contrast, the Feistel generator has a higher hardware area cost due to the necessary circuitry to produce and store a new S-box per round. The result is a trade-off of area (one-time) for security (round uniqueness). Furthermore, to position the modified version within the existing lightweight block cipher, Table 7 shows the hardware cost of the proposed cipher with representative lightweight ciphers. To measure the cost of hardware, we use common primitive GE assumptions, which are as follows:

- 2-input NAND = 1 GE (basis)
- 2-input XOR = 4 GE
- 2-input OR = 1 GE
- 1-bit D-FF = 8 GE
- 1-bit 2:1 MUX = 4 GE

The analysis considers the basic hardware primitives: XOR, AND/OR, flip-flops (FF), and multiplexers (MUX). These are used in constructing each S-box component, as shown in Table 6. Table 7 illustrates that under the serial reuse architecture assumed in this work, the proposed Feistel-based S-box generator introduces a moderate additional area cost compared to other lightweight ciphers.

Table 7. Hardware cost comparison with lightweight designs

Cipher	S-box design	Dynamic Key-Dependnt	GE	Notes
DRcipher-96	Pseudo random rounds	no	1546	Not dynamic S-box
DRcipher -128	Pseudo random rounds	no	1646	Synthesized ASIC
DBST-128	Dynamic S-box	Yes (session)	1697	Explicit dynamic S-box
Modified version	Dynamic S-box	Yes (round)	1892	Serial reuse assumed

4. Security analysis

In this section, we examine the security characteristics of the proposed version and compare them with the properties of the PRESENT cipher. From a theoretical perspective, the use of a round-dependent substitution layer modifies the nonlinear structure from round to round, which is likely to affect both confusion and the propagation of differences when combined with the fixed permutation layer. Based on this rationale, the analysis focuses on differential and linear cryptanalysis, followed by an assessment of the avalanche effect. The analysis is then extended to consider the implications for algebraic interpolation-based attacks and integral/square attacks, offering a broader security perspective. All experimental results reported in this section were obtained using Python implementations.

4.1 Branch characteristics and diffusion

The branch number is commonly used to describe how effectively a transformation distributes differences, and it is defined as the sum of the input and output Hamming weights for nonzero differences. In PRESENT, the S-box has a fixed branch behavior that remains constant across rounds. In the proposed design, however, the S-box is generated dynamically, so its local diffusion properties can vary from one round to another. When this variation is combined with the fixed permutation layer of PRESENT, it becomes more difficult for differential and linear trails to maintain a simple, repetitive structure over many rounds. As a result, the substitution layer contributes to local diffusion in a round-dependent manner, rather than through a single fixed pattern.

4.2 Differential analysis

The differential uniformity of an S-box is defined as:

$$\delta_S = max_{a\neq 0,b}|\{x \in \{0,1\}^n : S(x)\oplus S(x\oplus a) = b\}| \quad (35)$$

Based on the above equation, the optimal differential uniformity is 2. To conduct the empirical differential analysis, we generate S-boxes for 10,000 random 16-bit seeds (derived from round keys). Then we compute the differential distribution table (DDT) by recording the maximum of the DDT for each S-box. Table 8 presents the DDT parameters for both ciphers.

Table 8. DDT for both ciphers

Metric	S-box based PRESENT	S-box based Feistel
MAX LAT	4	2-3
AVG. LAT	0.9	1.2

Thus, based on the above table, the modified version reduces the maximal differential propagation, improving resistance to differential cryptanalysis. Moreover, each encryption round uses a different S-box, preventing the reuse of differential trails across rounds.

4.3 Linear analysis

In terms of linear analysis, the linear approximation table (LAT) of an S-box is defined as follows:

$$LAT[a,b] = |\{x: a\cdot x\oplus b\cdot S(x) = 0\}| - 2^{n-1} \quad (36)$$

To perform this test, we generate S-boxes for 10,000 random 16-bit seeds (round keys); the empirical linear analysis computes LAT for each S-box that is generated for both ciphers. The maximum absolute value $|\ LAT_{max}\ |$ measures linear bias, and a smaller value indicates more resistance to linear cryptanalysis. Table 9 presents the results of this test.

Table 9. LAT for both ciphers

Metric	S-box based PRESENT	S-box based Feistel
MAX LAT	4	2-3
AVG. LAT	0.9	1.2

According to Table 9, the Feistel S-box has a lower maximum bias, reducing the success probability of linear attacks. The dynamic per-round S-boxes prevent combining linear approximations across rounds. Unlike PRESENT, which utilizes a single static S-box with constant differential and linear properties as illustrated in Tables 8 and 9, the median and percentile values of the modified version across 10,000 independently generated S-boxes indicate that most S-boxes exhibit moderate differential and linear bounds, with higher values appearing infrequently, as shown in Table 10. Based on the above distribution analysis, it shows that most generated S-boxes exhibit favorable differential and linear properties, with the 90th and 95th percentile values remaining close to the median. A small number of S-boxes reach higher maximum DDT or LAT values. However, these occurrences

are uncommon and expected given the limited size of the 4-bit domain. Since the overall distribution is concentrated and weak instances do not appear persistently, no rejection or filtering mechanism is applied. Since the substitution layer is regenerated dynamically across rounds, isolated weaker S-boxes do not remain over multiple rounds. This limits their impact on security.

Table 10. Distribution of maximum DDT and LAT values of the proposed dynamic S-box

Modified cipher	Max DDT	MAX LAT
Mean	2.6	2.4
Median	2	2
90th percentile	3	3
95th percentile	3	3
Maximum observed	4	4

4.4 Avalanche effect

The avalanche effect is a property that measures the diffusion strength of a cryptographic system [20,21]. To perform this test, we generate 10,000 64-bit random plaintexts P and then apply the following steps:

- Encrypt P to get ciphertext $C = E(P)$.
- Flip exactly one bit of P to get $P' = P \oplus (1 \ll i)$, i is a position of the bit from 0 to 63.
- Encrypt again $C' = E(P')$.
- Measure the Hamming distance: $d = H(C \oplus C')$.

Then, we accumulate the mean avalanche and min/max bit flips. The variability of avalanche behavior is evaluated using the standard deviation, which is a widely accepted measure of statistical dispersion [21]. A higher standard deviation indicates a wider spread of output bit changes, resulting in increased variability in the cipher's response, as shown in Table 11.

Table 11. Avalanche rate for both ciphers

Metric	Modified version	PRESENT
Mean bit flips	30.92 bits	32.14 bits
Minimum	18 bits	28 bits
Maximum	40 bits	36 bits
Std. deviation	4.7	2.1

Table 11 shows a comparison of the PRESENT cipher and the modified version. The PRESENT cipher produced an average rate of 32.14, which is close to the theoretical optimum of 32 bits for a 64-bit block. In contrast, the modified scheme produced a slightly lower average of 30.9 bits of changes. Furthermore, this table illustrates that the standard deviation of the output bit changes for the proposed cipher is higher than that of PRESENT (4.7 compared to 2.1), which can be interpreted as a wider dispersion of avalanche outcomes. Thus, this confirms that the observed behavior reflects increased variability rather than a uniform shift in the mean. We evaluate compliance with the strict avalanche criterion (SAC) by estimating the flip-probability matrix and reporting mean/max deviations from 0.5, where SAC analysis measures the balance of individual output bits. For each input bit $i \in [0,63]$ and output bit $j \in [0,63]$, we estimate the matrix:

$$P_{i,j} = P_r[C_j(x) \neq P_r[C_j(x \oplus 2^i)] = 0.5 \quad (37)$$

Thus, SAC is "good" when $P_{i,j} \approx 0.5$ for all (i,j). We plot $P_{i,j}$ as a heatmap, where the x-axis is the output bit and the y-axis is the input bit. The color is flip probability, as shown in Figure 2 and Figure 3.

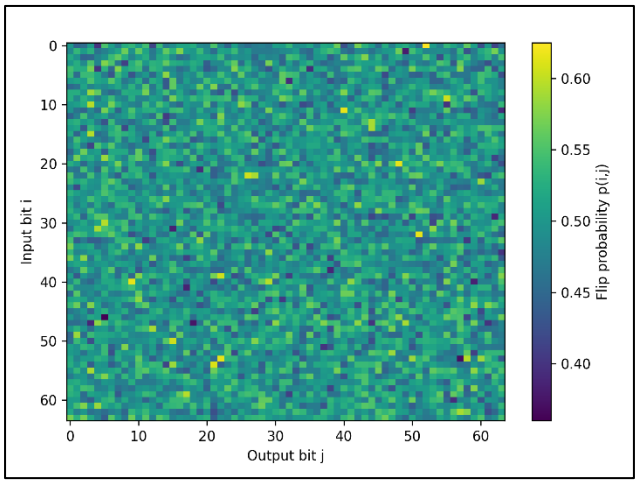

Figure 2. SAC flip probabiities (Modified)

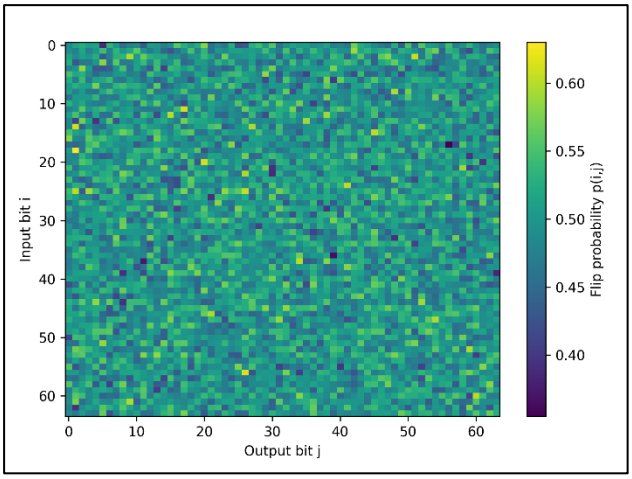

Figure 3. SAC flip probabilities (PRESENT)

Figure 2 and Figure 3 show the SAC for both ciphers. For the PRESENT cipher, the heatmap shows a relatively uniform and stable pattern, reflecting the use of a fixed substitution layer across all rounds. This behavior is consistent with the narrower avalanche distribution and lower standard deviation observed in Table 10. In contrast, the heatmap of the introduced variant shows a slightly increased variation, while remaining centered around the ideal probability of 0.5. This behavior is from the round-dependent S-box generation, which alters nonlinear dependencies between input and output bits from round to round. As a result, individual avalanche trials may involve different subsets of active output bits, generating a wider dispersion in the total number of flipped bits.

4.5 Algebraic and interpolation attacks

On the other hand, this section evaluates the security strength of the proposed scheme against algebraic and interpolation attacks. Algebraic and interpolation attacks commonly use static algebraic models of the substitution layer across all rounds. In the PRESENT block cipher, the

static S-box allows attackers to reuse the same algebraic equations or interpolation polynomials throughout the encryption phase. In contrast, the introduced scheme employs a round-dependent, key-generated S-box that changes from round to round. This variability precludes the construction of a single global algebraic system and considerably complicates the reuse of equations across rounds. Although this does not eliminate the theoretical possibility of algebraic attacks, it increases their difficulty by requiring round-specific modeling, thereby raising the cost of attacks.

4.6 Integral/square attack

Integral cryptanalysis represents a particular cryptanalytic attack that is applicable to block ciphers built based on substitution-permutation networks (SPN) [22]. Initially popular for byte-oriented designs, later work showed that bit-based integral properties can also be exploited for bit-oriented lightweight ciphers, including reduced-round PRESENT [23].

Lemma 1 (Disruption of Iterative Integral/Square Distinguishers): Let the modified version use a round-dependent substitution layer S_r, where each S_r is a bijective 4-bit S-box generated from the corresponding round key K_r. Then any integral/square distinguisher that depends on the repeated application of a fixed nonlinear layer across rounds, as in the original PRESENT cipher, cannot be directly iterated across multiple rounds of the modified variant.

Proof (sketch): In the static PRESENT algorithm, the substitution round function is defined as follows:

$$R(x) = P(S(x \oplus K_r)) \tag{38}$$

where the substitution layer function S is static across all rounds. This property allows integral/square distinguishers to track constant subsets through multiple rounds by repeatedly applying the same nonlinear mapping. In the proposed cipher, the substitution layer function is defined as:

$$R_r(x) = P(S_r(x \oplus K_r)) \tag{39}$$

where $S_r \neq S_{r+1}$ in general, since each S-box is derived from round key (K_r). Even if an integral property (e.g., bitwise balance or zero-sum) holds after round r, its propagation into the next ($r + 1$) depends on a different nonlinear mapping with potentially different differential and linear characteristics (DDT/LAT entries and active-bit behavior).

Thus, the attacker cannot assume invariant propagation conditions across rounds, which are essential to extend classic square distinguishers to many rounds. Constructing a full-round distinguisher would require modeling the entire sequence $\{S_r\}$, greatly increasing the complexity compared to the static-S-box case.

5. Security hardware trade-off analysis

The PRESENT cipher was originally designed with hardware efficiency as a primary goal, achieving an implementation footprint of approximately 1570 GE for an 80-bit key and 1884 GE for a 128-bit key. In the proposed version, the static S-box is replaced with a dynamic, key-dependent S-box generated by a lightweight Feistel network architecture. This modification includes additional hardware costs due to the need for extra combinational logic, including XOR networks, small nonlinear functions, and bit-permutation stages. A detailed gate-equivalent (GE) analysis shows that the rounds of the Feistel structure contribute approximately 340–360 GE. Thus, the total area increases to around 1930 GE, representing a ≈20% rise in hardware cost.

From a security perspective, the overhead is justified. The dynamic S-box ensures that each round uses a distinct substitution layer, effectively eliminating repeated patterns that adversaries could exploit. Empirical evaluation confirms that the modified cipher achieves several improvements. The avalanche rate is close to the ideal 50% bit change, reflecting stronger diffusion. DDT is typically around 2–3, which reduces the highest probability of differential characteristics. The maximum linear bias drops from 4 to about 2–3, making linear approximations (LAT) harder to exploit. These enhancements provide significantly stronger resistance against differential and linear cryptanalysis, as well as improved protection against structural attacks targeting repeated S-boxes. Finally, the proposed modification demonstrates a clear trade-off: a moderate increase in hardware cost (≈20%) results in a substantial improvement in security. For resource-constrained devices where long-term robustness outweighs marginal area savings, this trade-off is acceptable and positions the cipher as a more resilient option for lightweight cryptographic applications.

6. Conclusion

In this paper, the modified variant replaces PRESENT's original 4-bit static S-box with a dynamic alternative created through a compact Feistel construction, where each round generates a new S-box from the corresponding round key. By removing the fixed substitution layer, i.e., eliminating repetitive patterns, the introduced method avoids predictable structures and prevents attackers from relying on precomputed differential or linear trails. The dynamic S-box exhibits lower differential uniformity, with DDT typically around 2–3 instead of the value 4 observed in PRESENT, which lowers the highest probability of differential characteristics. A similar improvement is observed in linear behavior, as the maximum linear bias decreases from 4 to the range of 2–3, making linear approximations (LAT) more difficult to exploit. Because the substitution box changes every round, linear correlations cannot accumulate in a consistent manner across the full encryption process. Furthermore, the avalanche rate is close to the ideal 50% bit change, reflecting the stronger diffusion introduced by the nonlinear Feistel function. The variation of the S-box also offers natural resistance to several structural attacks, such as algebraic interpolation-based attacks and integral/square attacks that typically exploit static substitution layers. Although the dynamic approach increases hardware cost (roughly 340–360 GE versus 28 GE for the fixed S-box), the additional complexity is modest relative to the improved security, and the overall implementation remains suitable for lightweight environments, i.e., still acceptable given hardware costs less than 3000 GE. Overall, the modified PRESENT variant achieves a more favorable balance between security and efficiency and appears suitable for IoT and embedded platforms where both hardware budget and security per gate equivalent are important.

Ethical issue

The authors are aware of and comply with best practices in publication ethics, specifically with regard to authorship (avoidance of guest authorship), dual submission, manipulation of figures, competing interests, and compliance with policies on research ethics. The author adheres to the

publication requirement that the submitted work is original and has not been published elsewhere.

Data availability statement
The manuscript contains all the data. However, more data will be available upon request from the authors.

Conflict of interest
The authors declare no potential conflict of interest.

References

[1] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin and C. Vikkelsoe, "PRESENT: An Ultra-Lightweight Block Cipher," in Cryptographic Hardware and Embedded Systems (CHES), Vienna, Austria, , 2007. https://doi.org/10.1007/978-3-540-74735-2_31

[2] A. Abdelli, W. E. h. Youssef, F. Kharroubi, L. Khriji and M. Machhout, "A novel enhanced chaos based present lightweight cipher scheme," Electrical and Computer Engineering, vol. 99, no. 1, 2024. https://doi.org/10.1088/1402-4896/ad1560

[3] A. S. Salim and S. I. F. Taha, "Improving PRESENT Lightweight Algorithm," 2013. https://doi.org/10.5120/12931-9874

[4] R. Bharathi and N. Parvatham, "Light-Weight Present Block Cipher Model for IoT Security on FPGA.," Intelligent Automation & Soft Computing , vol. 33, no. 1, pp. 1079-8587, 2022. https://doi.org/10.32604/iasc.2022.022067

[5] N. Maatallah, H. Mestiri, A. A. Mohamed and M. Machhout, "Enhancing IoT Security for Sustainable Development: A Parity Checking Approach for Fault Detection in PRESENT Block Cipher," Engineering, Technology & Applied Science Research, vol. 15, no. 2, pp. 1982-21988, 2025. https://doi.org/10.48084/etasr.10109

[6] H. K. H. Zaid M. Jawad Kubba, "Modified PRESENT Encryption algorithm based on new 5D Chaotic system," IOP Conference Series:, 2020. https://doi.org/10.1088/1757-899X/928/3/032023

[7] R. D. Labio and E. D. Festijo, "D-PRESENT: A Lightweight Block Cipher with Dynamic Key-Dependent Substitution Boxes," International Conference on Advanced Computer Science and Information Systems (ICACSIS), 2020. https://doi.org/10.1109/ICACSIS51025.2020.9263158

[8] M. Lewandowski and S. Katkoori, "Enhancing PRESENT-80 and Substitution-Permutation Network Cipher Security with Dynamic "Keyed" Permutation Networks," Computer Society Annual Symposium on VLSI (ISVLSI), 2021. https://doi.org/10.1109/ISVLSI51109.2021.00063

[9] Zhang and Wentao, "RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms," Science China Information Sciences, vol. 58, no. 12 , pp. 1-15, 2015. https://doi.org/10.1007/s11432-015-5459-7

[10] M. Wang, Y. Wang and H. Wu, "Cryptanalysis of the PRESENT Block Cipher," in International Conference on Cryptology , Casablanca, Morocco, 2008. https://doi.org/10.1007/978-3-540-89754-5_14

[11] J. Y. Cho, "Linear cryptanalysis of reduced-round PRESEN," in Information Security and Cryptology (ICISC), Seoul, South Korea, 2009. https://doi.org/10.1007/978-3-642-14423-3_16

[12] A. Lindman, "Synergy: A Lightweight Block Cipher with Variable Bit Rotation Feistel Network," Cryptology ePrint Archive, 2025. https://eprint.iacr.org/2025/001

[13] Y. e. a. Li, "A Novel Feistel-Based Low-Latency Block Cipher for IoT Applications," IEEE Internet of Things Journal , 2025. https://doi.org/10.1109/JIOT.2025.3600289

[14] M. Alawida, "Tree-Feistel Cipher Standard for IoT Communication System," IEEE Internet of Things Journal, 2025. https://doi.org/10.1109/JIOT.2025.3520123

[15] Al-Nofaie, S. Meteb, S. Sharaf and a. R. Molla, "Design trends and comparative analysis of lightweight block ciphers for IoTs," Applied Sciences , vol. 7740, 2025. https://doi.org/10.3390/app15147740

[16] A. Bogdanov, G. Leander, K. Nyberg and and M. Wang, "On the construction of S-boxes using Feistel structures," in Workshop on Cryptographic Hardware and Embedded Systems (CHES), 214. https://doi.org/10.1007/978-3-662-44709-3_2

[17] S. Kutzner, G. Leander and a. A. Poschmann, "Construction of lightweight S-boxes using Feistel and MISTY structures," in Selected Areas in Cryptography (SAC), 2015. https://doi.org/10.1007/978-3-319-31301-6_11

[18] K. Mohajerani, R. Haeussler, R. Nagpal, F. Farahmand, A. Abdulgadir, J.-P. Kaps and K. Gaj, "FPGA Benchmarking of Round 2 Candidates in the NIST Lightweight Cryptography Standardization Process: Methodology, Metrics, Tools, and Results," Farnoud Farahmand, Abubakr Abdulgadir, Jens-Peter Kaps and Kris Gaj, 2020. https://eprint.iacr.org/2020/1207

[19] J. P. Bhoge and P. N. Chatur, "Avalanche Effect of AES Algorithm," International Journal of Computer Science and Information Technologies, vol. 5, no. 3, pp. 3101-3103, 2014. http://ijcsit.com/docs/Volume%205/vol5issue03/ijcsit20140503117.pdf

[20] D. Upadhyay, N. Gaikwad, M. Zaman and S. Sampalli, "Investigating the Avalanche effect of Various Cryptographically Secure Hash Functions and Hash-based Applications," vol. 10, pp. 112472 - 112486, 2022. https://doi.org/10.1016/j.measen.2022.100569

[21] NIST/SEMATECH, "Engineering Statistical Handbook," 2012. [Online]. Available: https://www.itl.nist.gov/div898/handbook/.

[22] M. I. Mihailescu and S. L. Nita, Pro Cryptography and Cryptanalysis, Bucharest, Romania: Springer, 2021, p. 483–499. https://doi.org/10.1007/978-1-4842-6367-9

[23] S. Wu and M. Wang, "Integral Attacks on Reduced-Round PRESENT," in Information and Communications Security, 2013. https://doi.org/10.1007/978-3-319-02726-5_12

This article is an open-access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).