International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING

ISSN:2147-6799 www.ijisae.org Original Research Paper

Round S-Boxes Development for Present-80 Lightweight Block Cipher Encryption Algorithm

Ahmed Abdullah Khalil[1], Maytham Mustafa Hammood[2], Awni M. Gaftan[3]

Submitted: 27/10/2022 **Revised:** 28/12/2022 **Accepted**: 26/01/2023

Abstract

lightweight cryptography is a branch of encryption. It is symmetric encryption that is made to work on a variety of hardware that has low processing power, low cost, limited memory, and limited processing time. With a lightweight design, consideration is given to how to maintain a balance between usability and functionality while still maintaining an appropriate level of security. The present block cipher algorithm is one of them; it has a symmetric key and two versions—one with an 80-bit key and the other with a 128-bit key—each of which employed blocks of a 64-bit length. Present-80 algorithm, which is commonly used in IOT applications and has an 80-bit key length. In this study, an enhancement to the Present-80 encryption algorithm was proposed. The proposal involves the usage of sixteen separate S-boxes, eight of which are used for processing each round and eight for updating the roundkey. One of these eight boxes is used for each round to update the roundkey, which is determined by choosing three positions from the key for each round before it is updated and makes up an address between 0 and 7. Three different positions are selected in the same way that the active s-box for that round is chosen. The active s-box for each round of encryption/decryption processing is selected by the three-position key for that round before it is updated. At the end of the study, the throughput of the size and execution time of the original and modified algorithms were compared using a variety of data formats, including text, audio, image, and video, with varied sizes. The 16 NIST statistical randomness packages were used to test the randomness of the ciphertext sequences produced by the enhanced algorithm, and all of these tests were passed successful.

Keywords: *lightweight, algorithm, proposed, enhanced, encryption, present-80, S-boxes, P-layer*

1. Introduction

Nowadays, it is necessary to take functionality, usability, and security into consideration while designing new devices or applications. Although the secure by design principles have been pushed for security to be included from the start, there are still implementation gaps, and design concerns of functionality and usability are one of these gaps [1]. Effective security requires a lot of work, but it should never prevent users from using functionalities that they need. Preferably, security must be adjusted when it obstructs important program features. At the same time as security shouldn't be overlooked, functionality shouldn't be. Three concepts are used as a measure to determine the degree of

---

[1] *Computer science and Mathematical College, Tikrit university*
Ahabka1@gmail.com
[2] *Computer science and Mathematical College, Tikrit university*
Maythamhammood@tu.edu.iq
[3] *Computer science and Mathematical College, Tikrit university*
Awny.muhammed@tu.edu.iq

security of any system to solve this issue as explain in Figure 1 functionality which is the collection of functions that the system provides constitutes, Usability which is The GUI elements that were used to create the system for simplicity and the security which is the limitations placed on accessing the system' s components.[2][3]

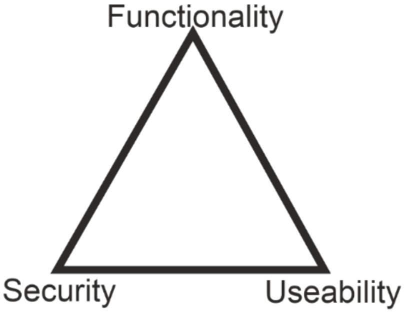

Fig 1: the Triad in design.

In our daily lives, low-power technologies are used in everything from home appliances to digital assistants and medical equipment. Given the low power circumstances these devices operate in and the fact that they frequently contain our valuable private information, it is imperative to offer a fair level of security.[4] Because to potential restrictions in both the hardware such as memory and software on these devices, standard encryption techniques do not always perform adequately for all applications, i.e., processor speed. Low-power devices will struggle since they do not have as much processing power as smartphones and laptops, for instance, if a video stream or a huge stream of data needs to be protected in a short time amount. As tradeoffs are made in a low-power system, security suffers since the available power is more constrained.[5]

For instance, smaller key sizes are desirable in such a setting, although doing so may lower the system's level of security. As a result, the objective of lightweight cryptography such present-80 is to use less memory, processing power, or other resources while yet offering some level of security [6].

From a software perspective, memory size, processing performance, and latency may be limitations for lightweight devices. Lightweight hardware may have limitations in terms of area, throughput, and power use. Lightweight cryptographic algorithms such as present-80 that can offer a reasonable level of security in a variety of applications are required while operating in these contexts [7]. Examples of devices with varied computing capabilities are shown in Figure 2.

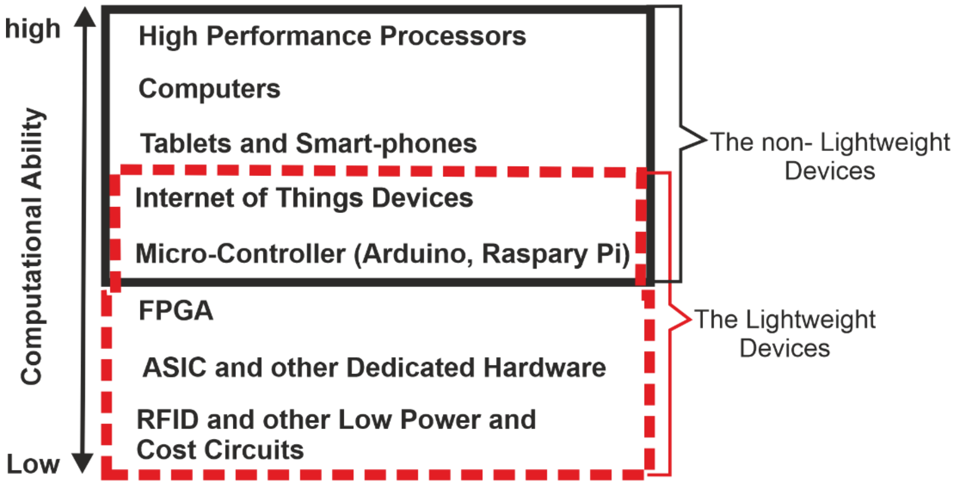

Fig 2: Computing power across different hardware platforms.

2. Description of Present-80 standard Encryption Algorithm

The most popular light-weight encryption method is PRESENT, a straightforward block cipher that is available for usage anywhere. It was presented in 2007 and standardized in ISO/IEC 29192, just like CLEFIA (a proprietary block cipher algorithm developed by Sony [8]. Since its inception, it has been the subject of extensive research, and many consider it to be among the best lightweight encryption algorithms ever created. The hardware-oriented Present cipher uses keys of 80/128 bits to encrypt a 64-bit block over 31 rounds using its substitution-permutation network (SPN) structure.[9] One S-box Layer and one P-Layer are both involved in every encryption or decryption round of the cipher.

Every round, the key register produces a 64-bit key that it XORs with plaintext. The substitution layer uses 4-bit S-boxes for both input and output.[10] The permutation layer (P-Layer) is straightforward Used in every round except for the pre-round of the 64-bit sequence that generated by S-Box. The Present-80 encryption algorithm is described in Figure 3 for both processing encryption and decryption.

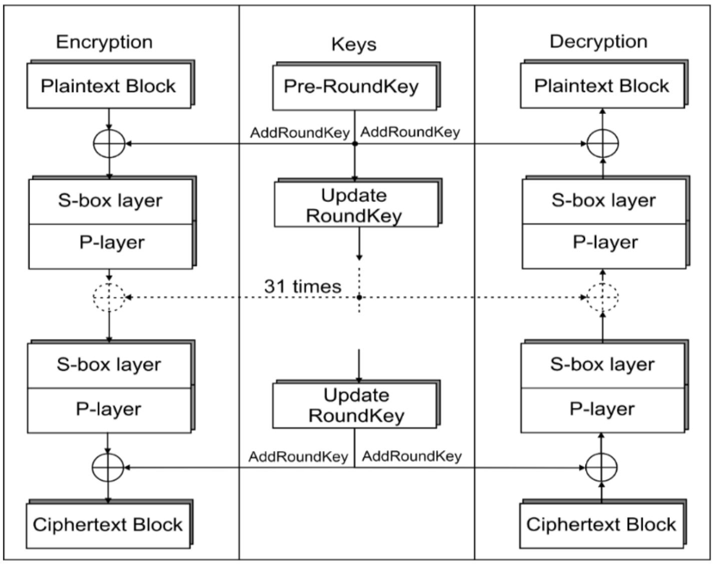

Fig 3: present-80 encryption/decryption processes

i. Present-80 Algorithm Phases

Each block in the Present-80 encryption method goes through a pre-round and post-rounds of 31 rounds, with each round going through the phases described below:

ii. The Round-Zero (Pre-Round)

At the pre-round the input block of 64-bit is XORed with the 64-bit cipher key to produce the state vector of 64-bit.

iii. Round Operations (Post-Rounds)

Three operations, SubBytes on the state vector using S-boxLayer, Permutation layer to provide P-Layer, and Key Expansion Function to provide addRoundKey are accomplished in each of the 31 post-rounds.

iv. Byte Substitution Layer (S-Box)

A nonlinear substitution block that is separately applied to each bit of the state vector makes up the S-Box layer. This layer produces new data that isn't linearly linked to the original data. This substitution block is used to convert 16 hexadecimal digits corresponding a 64-bit cipher block.[11] By scanning four-bit mask tables, each of which is sized for a microcontroller with a larger data bus, this replacement table, as shown in Table 1 and Figure 4.

Table 1: S-Box Table [12]

i/p	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
o/p	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

v. Bit Permutation Layer (P-Layer)

Bit i of the round is shifted to the P(i) position in order to combine a 64-bit data block. The order of these replacements is shown in Table 2 and Figure 4.

Table 2: Present-80 P-layer Matrix [13]

i/p	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
o/p	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i/p	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
o/p	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i/p	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
o/p	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i/p	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
o/p	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

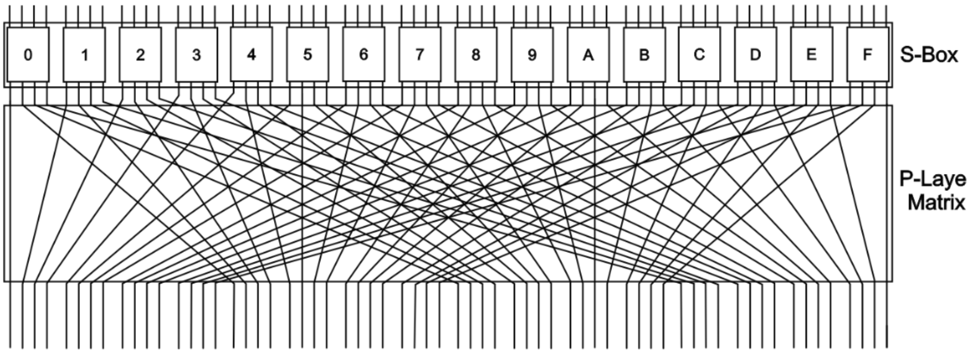

Fig 4: Present-80 S-box & P-layer

vi. Key expansion function

Key expansion is a crucial procedure since it produces unique keys for every round. It only uses the 64 bits of the new 80-bit main vector that are the most relevant to each round. The 64-bit key generated during the current round is then concatenated with the final 16 bits of the key vector to be used in the next round [14] as shown in Figure 5.

The generation of the addRoundKey is carried out for each round of encryption; otherwise, it is important to keep in mind that the first round of decryption uses the final addroundkey until it reaches the pre-round key with the final round, meaning that the final key used for encryption will be the first key used for decryption. [15][16][17]

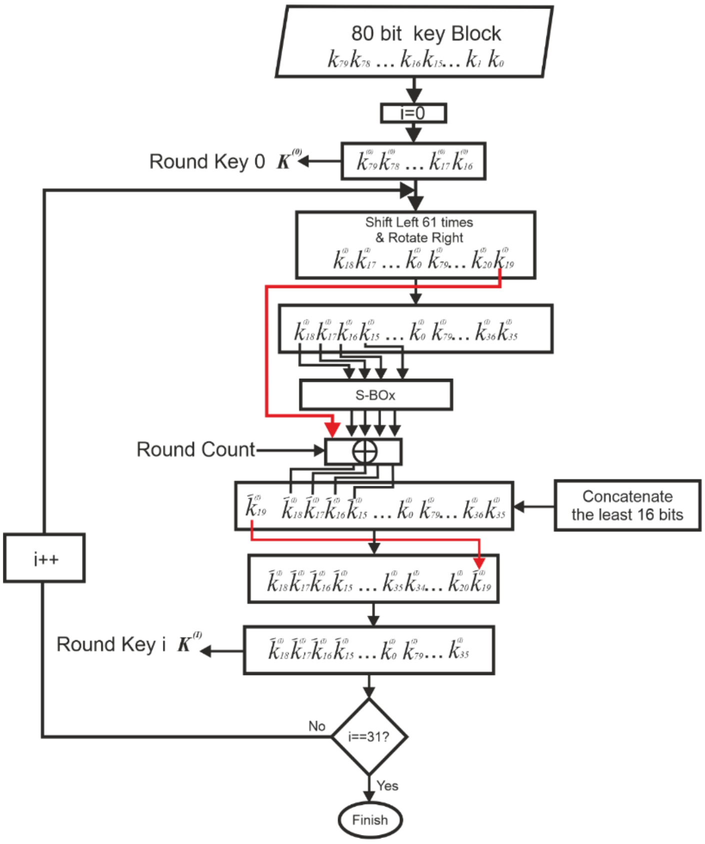

Fig 5: Present-80 standard algorithm expansion key

3. The Proposed Key Expansion

In the proposed enhanced algorithm, a basic key with a length of 80 bits is used, i.e. the same as that used in the standard algorithm. In the initial round (pre-round), the same left 64 bits from the basic key are used as in the standard algorithm. As for the 31 subsequent rounds, at the beginning of each round, through three locations of the key, an address from 0 to 7 specified to choose one of the S-boxes for the purpose of converting the 64-bit key after it has been shifted to the left 61 bits and rotated it to the right at the same time. 4 bits in hexadecimal form are entered into the identified S-box to obtain 4 bits output at a time and this process repeated16 times. The first bit of the key after shifting and rotating in MSB's place inserted to last 4 outgoing bits to become 5 bits, represent a number in range 0 to 31, these bits are mixes with the round number using bitwise XOR function. In the last step each bit returns to the it previous position, after concatenating the remaining 16 bits from the 80 bits of the previous round, replacing the LSB with new one to generate the next round key. Figure 6 shows the key expansion process.

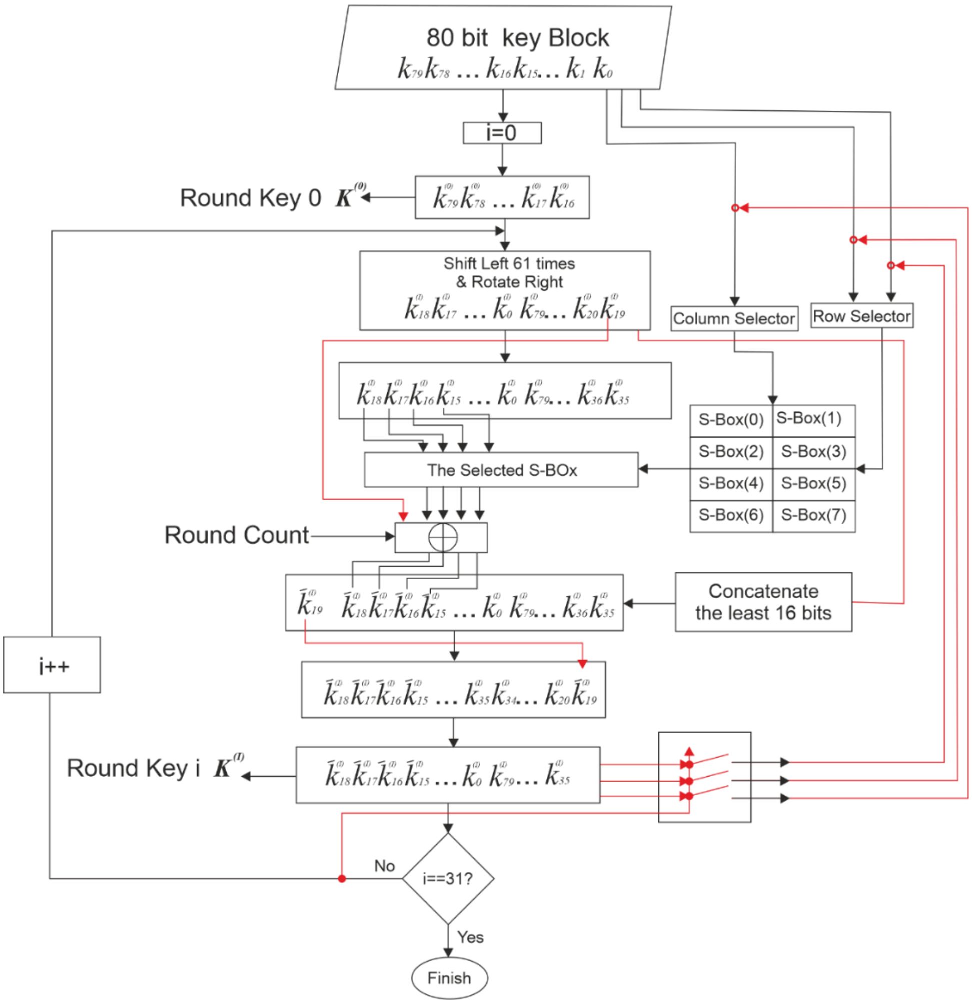

Fig 6: The proposed algorithm expansion key

4. The Proposed Encryption/Decryption Processes

The encryption method Starting with the entry of a 64-bit plaintext block, which is mixed XOR-function with 64-bits of the key before expansion, 3 distinct positions of the key that form an address that used to identify the round S-box among eight boxes. The generated sequence is changed into a new sequence using the selected round S-box. The generated sequence permuted according to P-layer matrix as shown in Table 2 and Figure 7, then mixed XOR with the 64-bit addroundkey sequence. This process is repeated 31 times, and the last round results in the 64-bit ciphertext block is obtained.

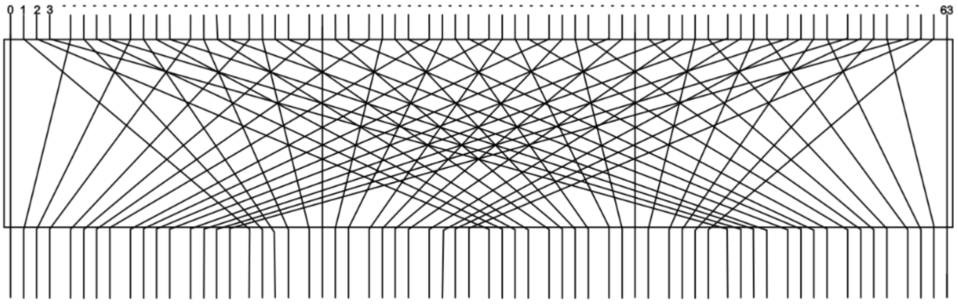

Fig 7: permutation layer

Figure 8 represent the proposed encryption algorithm.

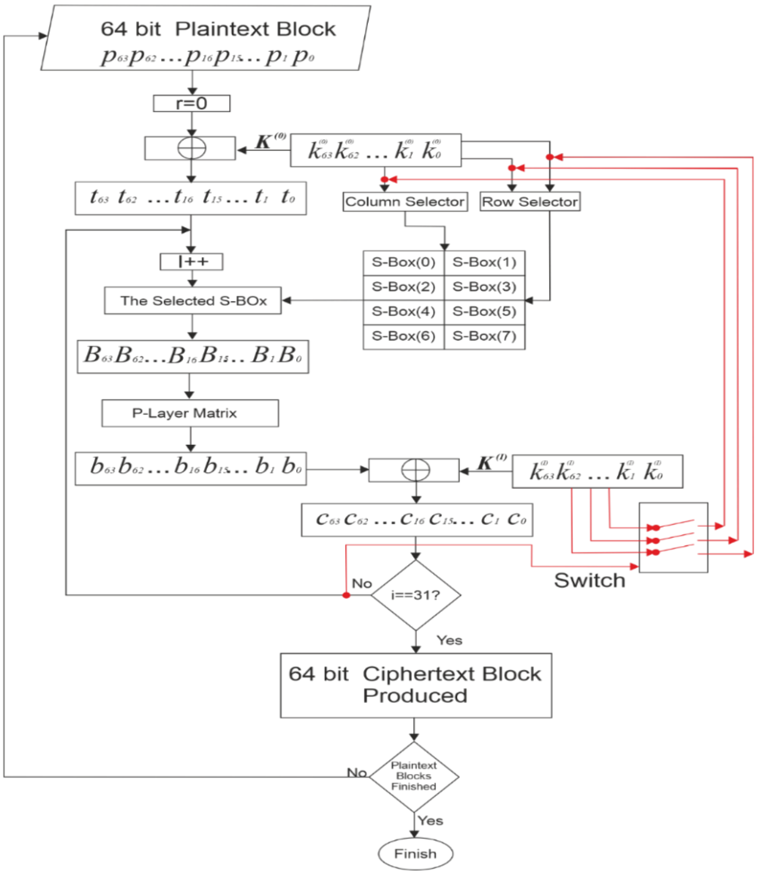

Fig 8: The proposed encryption algorithm

Whereas the decryption method starting with the entry of a 64-bit ciphertext block, which is mixed XOR-function with 64-bits of the last addroundkey. The produced sequence is permuted according to P-layer inverse matrix as shown in Figure 9, then 3 distinct positions of the key that form an address that used to identify the round S-box inverse among eight s-boxes inverse.

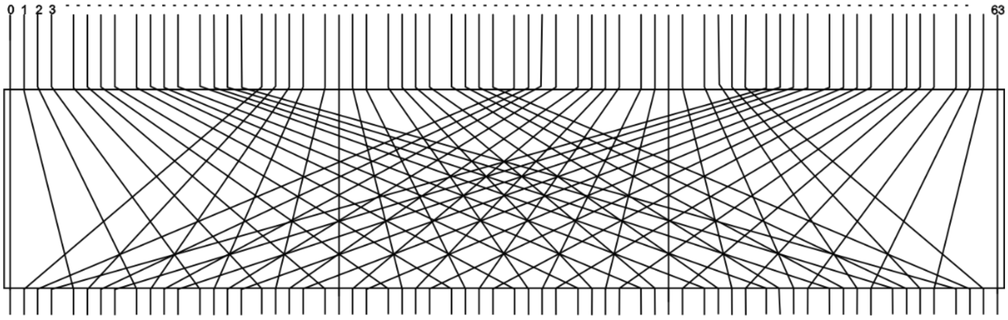

Fig 9: The permutation layer inverse

The generated sequence is changed into a new sequence using the selected round S-box inverse, and the 64-bit addroundkey sequence is mixed with the produced 64-bit sequence. This process is repeated 31 times, and the last round results in the 64-bit plaintext block is obtained. It is important to keep in mind that the first round of decryption uses the final addroundkey until it reaches the pre-round key with the final round. Figure 10 represent the proposed encryption algorithm.

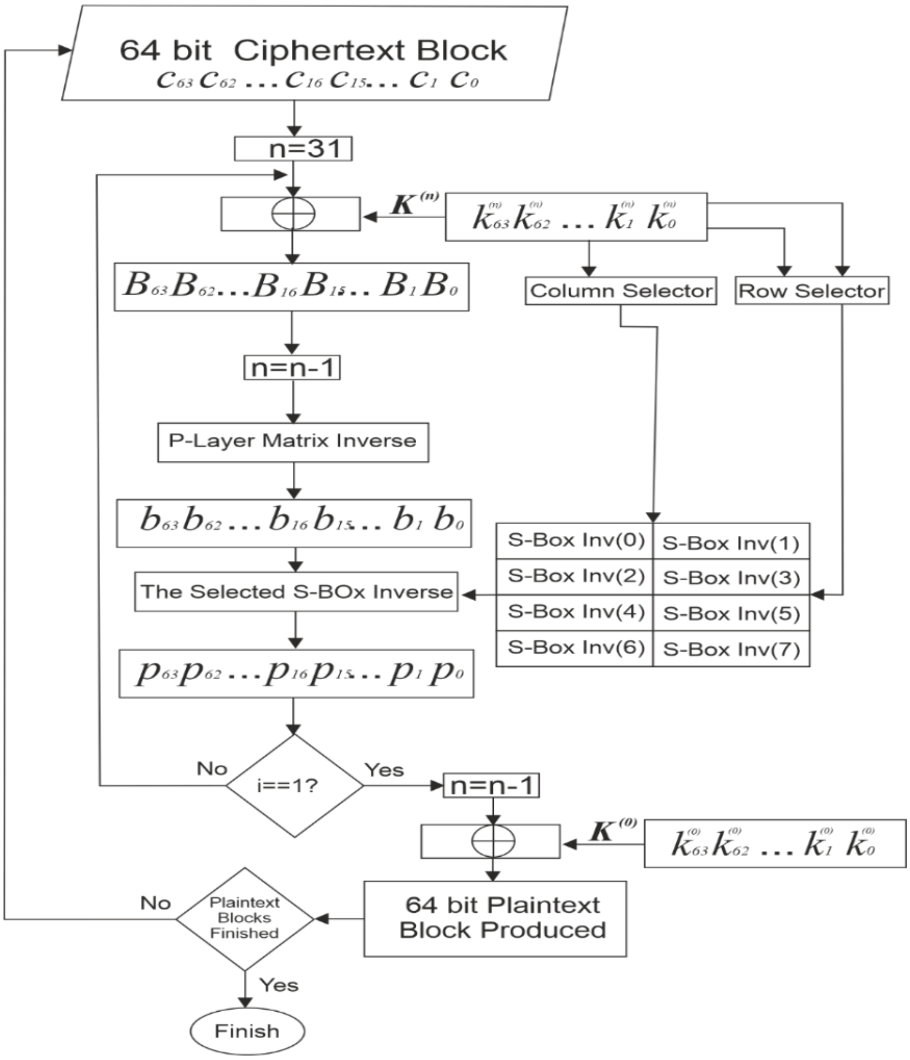

Fig 8: The proposed encryption algorithm

5. Performance and Randomness

In the experiments of this study, the performance factors and statistical randomness tests are applied in order to reach the balance in the levels of performance and randomness of the proposed enhanced algorithm.

I. Performance Factors

A laptop with a core i7 processor, 2.8 GHz, and 32 GB of RAM is used for the experiment. Microsoft Visual Basic dot net programming language 2022 is used to program the block Ciphers algorithms of standard and enhanced, the experiment uses to encrypt files of various sizes. The following variables were used as performance metrics and to compare the standard and enhanced algorithms:

i. Input data types (text, audio, video, image, word document, pdf and application)
ii. Encryption process Time
iii. Decryption process Time
iv. Throughput of Encryption of different block ciphers with different input data types
v. Throughput of decryption of different block ciphers with different input data types

Throughput is calculated by dividing the whole sequence, measured in Megabytes, by the total time required for encryption and decryption [18] . Power consumption of an encryption or decryption process decreases as throughput value increases.[19][20] Similar to this, if a process's throughput is decreased, its power consumption increases, increasing the lead battery's consumption in the process's wake.

II. Statistical Randomness Properties

The National Institute of Standards and Technology (NIST) Test Suite is a statistical collection of sixteen tests that was created to evaluate the randomness of binary sequences generated by cryptographic random number generators (RNG), both software or hardware based.[21] These tests concentrate on a wide range of potential non-randomness in a sequence. Several different subtests can be used to break down some tests.[22]

Each NIST test is described by a test statistic of one of the three types below and evaluates the sequence's randomness in accordance with:

1. Bits tests examine several aspects of bits, including their frequency of change (bit runs), their proportion, and their cumulative amounts.
2. m-bit blocks tests examine how m-bit blocks, which are typically smaller than 30 bits, are distributed throughout the sequence or its components.
3. M-bit tests examine intricate aspects of M-bits, usually greater than 1000 bits, such as the sequence's rank when regarded as a matrix, its spectrum, or the linear complexity of the bitstream. The tests only produce valid results (p-values) for specific values of their parameters since the reference distributions of the NIST test statistics are approximated by asymptotic distributions (e.g., normal or χ^2) only. The acceptable parameter values for each test that NIST recommends [23] are listed in Table 3.

Microsoft Visual Basic dot net programming language 2022 is used to program the NIST statistical randomness tests packages.

Table 3: NIST statistical tests acceptable parameter values

Test No.	Test Name	n	M or m
1.	Frequency (Monobit)	$n \geq 100$	-
2.	Frequency within a Block	$n \geq 9000$	$20 \leq M \leq \frac{n}{100}$
3.	Runs Test	$n \geq 100$	-
4.	Longest Run of Ones in a Block	$n \geq 128$	-
5.	Binary Matrix Rank	$n \geq 38912$	-
6.	Discrete Fourier Transform (Spectral)	$n \geq 1000$	-
7.	Non-overlapping Template Matching	$n \geq 8m - 8$	$2 \leq m \leq 21$
8.	Overlapping Template Matching	$n \geq 10,00,000$	-
9.	Maurer's (Universal Statistical)	$n \geq 13,42,400$	-
10.	Linear Complexity	$n \geq 10,00,000$	$500 \leq M \leq 5000$
11.	Serial	$n \geq 10,00,000$	$2 < m < [\, log_2 n] - 2$
12.	Approximate Entropy	$n \geq 100$	$m < [\, log_2 n] - 5$
13.	Cumulative Sums	$n \geq 100$	-
14.	Random Excursions	$n \geq 10,00,000$	-
15.	Random Excursions Variant	$n \geq 10,00,000$	-
16.	Lempel-Ziv Compression	$n \geq 10,00,000$	-

6. Experimental Results

Experimental results for the standard and the proposed algorithms are shown in Table 4, the table shows the encryption time of different blocks where input data is in the form of text, image, audio, video, document, pdf and application (execution file) of different sizes. In this table encryption throughput of different block ciphers is also calculated.

Table 4: Throughput standard and the proposed algorithms comparisons

index	File type	File size in MB	Encryption Process				Decryption Process			
			Standard Alg. Time (ms)	Throughput MB/Sec	Proposed Alg. Time (ms)	Throughput MB/Sec	Standard Alg. Time (ms)	Throughput MB/Sec	Proposed Alg. Time (ms)	Throughput MB/Sec.
	Image (PNG)	2.75	16079	0.17	13067	0.21	10788	1.53	9976	0.28
	text	0.247	16	15.44	6	15.44	16	15.44	4	0.06
	Document	0.974	5391	0.18	4777	0.20	3344	0.29	3025	0.32
	Audio	4.64	27381	0.10	23045	0.20	17739	0.26	17699	0.26
	Video	3.43	19511	0.18	18114	0.19	12218	0.28	11987	0.29
	Pdf	0.413	2345	0.18	2125	0.19	1534	0.27	1407	0.29
	Application	3.04	17677	0.17	16245	0.19	11335	0.27	10768	0.28

After analyzing Table 4, it is concluded that encryption and decryption time of the standard algorithm is higher than encryption and decryption time the proposed algorithm, the Comparisons Time (in Milliseconds)

I. The Statistical Randomness Tests

The NIST statistical randomness package tests have been performed on all ciphertexts produced by the proposed enhanced algorithm from the various files used in this study, which are listed in Table 2.

Tables 5 through 9 display the results, which show the success of all sequences generated by the proposed enhanced algorithm by passing all main and sub tests. The P-values for all tests were greater than the threshold $\alpha = 0.01$, which must be satisfied for each test to be considered pass.

Table 5: NIST Randomness Tests Frequency, Frequency Test within a Block, Runs and Longest Run of Ones in a Block

index	File type	File length in bit	The Frequency (Monobit)		Frequency Test within a Block		The Runs Test		Test for the Longest Run of Ones in a Block	
1.	Image (PNG)	23156032	0.750	**Pass**	0.032	**Pass**	0.187	**Pass**	0.113	**Pass**
2.	text	896	0.593	**Pass**	0.148	**Pass**	0.225	**Pass**	0.136	**Pass**
3.	Document	7983232	0.330	**Pass**	0.123	**Pass**	0.969	**Pass**	0.083	**Pass**
4.	Audio	38963904	0.190	**Pass**	0.099	**Pass**	0.087	**Pass**	0.19	**Pass**
5.	Video	36286656	0.043	**Pass**	0.078	**Pass**	0.738	**Pass**	0.013	**Pass**
6.	Pdf	4131648	0.321	**Pass**	0.015	**Pass**	0.12	**Pass**	0.213	**Pass**
7.	Application	25517696	0.201	**Pass**	0.089	**Pass**	0.011	**Pass**	0.113	**Pass**

Table 6: NIST Randomness Binary Matrix Rank, The Discrete Fourier Transform, Non-overlapping Template Matching and Overlapping Template Matching

index	File type	File length in bit	The Binary Matrix Rank Test		The Discrete Fourier Transform (Spectral) Test		The Non-overlapping Template Matching Test		The Overlapping Template Matching Test	
.	Image (PNG)	23156032	0.110	**Pass**	0.121	**Pass**	1	**Pass**	0.011	**Pass**
.	text	896	0.02	**Pass**	0.011	**Pass**	0.714	**Pass**	0.012	**Pass**
.	Document	7983232	0.043	**Pass**	0.139	**Pass**	0.028	**Pass**	1	**Pass**
.	Audio	38963904	0.146	**Pass**	0.148	**Pass**	0.011	**Pass**	1	**Pass**
.	Video	36286656	0.21	**Pass**	0.122	**Pass**	0.014	**Pass**	0.99	**Pass**
.	Pdf	4131648	0.099	**Pass**	0.122	**Pass**	0.020	**Pass**	0.09	**Pass**
.	Application	25517696	0.011	**Pass**	0.152	**Pass**	0.215	**Pass**	1	**Pass**

Table 7: NIST Randomness Maurer's "Universal Statistical", Lempel-Ziv, Linear Complexity and Serial

index	File type	File length in bit	Maurer's "Universal Statistical" Test		The Lempel-Ziv Compression Test		The Linear Complexity Test		The Serial Test		
	Image (PNG)	23156032	0.989	**Pass**	0.200	**Pass**	0.914	**Pass**	**P-value 1**	0.011	**Pass**
									P-value 2	0.154	**Pass**
	text	896	0.749	**Pass**	0.302	**Pass**	0.332	**Pass**	**P-value 1**	0.18	**Pass**
									P-value 2	0.024	**Pass**
	Document	7983232	0.987	**Pass**	0.092	**Pass**	0.012	**Pass**	**P-value 1**	0.044	**Pass**
									P-value 2	0.012	**Pass**
	Audio	38963904	0.904	**Pass**	0.199	**Pass**	0.261	**Pass**	**P-value 1**	0.011	**Pass**
									P-value 2	0.011	**Pass**
	Video	36286656	0.99	**Pass**	0.476	**Pass**	0.076	**Pass**	**P-value 1**	0.053	**Pass**
									P-value 2	0.076	**Pass**
	Pdf	4131648	0.978	**Pass**	0.026	**Pass**	.0105	**Pass**	**P-value 1**	0.18	**Pass**
								Pass	**P-value 2**	0.18	**Pass**
	Application	3.04	0.126	**Pass**	0.201	**Pass**	0.026	**Pass**	**P-value 1**	0.899	**Pass**
									P-value 2	0.14	**Pass**

Table 8: NIST Randomness Approximate Entropy and Cumulative Sums (Cusums)

index	File type	File length in bit	The Approximate Entropy Test		The Cumulative Sums (Cusums) Test		
8.	Image (PNG)	23156032	0.011	**Pass**	**Forward**	0.879	**Pass**
					Backward	0.591	**Pass**
9.	text	896	0.199	**Pass**	**Forward**	0.321	**Pass**
					Backward	0.752	**Pass**
10.	Document	7983232	0.015	**Pass**	**Forward**	0.182	**Pass**
					Backward	0.652	**Pass**
11.	Audio	38963904	0.133	**Pass**	**Forward**	0.066	**Pass**

					Backward	0.006	**Pass**
12.	Video	36286656	0.11	**Pass**	**Forward**	0.085	**Pass**
					Backward	0.017	**Pass**
13.	Pdf	4131648	0.321	**Pass**	**Forward**	0.015	**Pass**
					Backward	0.013	**Pass**
14.	Application	25517696	0.631	**Pass**	**Forward**	0.034	**Pass**
					Backward	0.033	**Pass**

Table 9: NIST Random Excursions and Random Excursions Variant

index	File type	File length in bit	The Random Excursions Test		The Random Excursions Variant Test	
1.	Image (PNG)					
	-4	23156032	0.558	**Pass**	0.489	**Pass**
	-3		0.484	**Pass**	0.555	**Pass**
	-2		0.476	**Pass**	0.670	**Pass**
	-1		0.246	**Pass**	0.902	**Pass**
	1		0.993	**Pass**	0.561	**Pass**
	2		0.652	**Pass**	0.503	**Pass**
	3		0.946	**Pass**	0.560	**Pass**
	4		0.154	**Pass**	0.613	**Pass**
2.	text					
	-4	896	0.011	**Pass**	1.000	**Pass**
	-3		0.304	**Pass**	0.572	**Pass**
	-2		0.128	**Pass**	0.273	**Pass**
	-1		0.259	**Pass**	0.343	**Pass**
	1		0.832	**Pass**	0.206	**Pass**
	2		0.928	**Pass**	0.361	**Pass**
	3		0.985	**Pass**	0.480	**Pass**
	4		0.996	**Pass**	0.550	**Pass**
3.	Document					
	-4	7983232	0.997	**Pass**	0.593	**Pass**
	-3		0.044	**Pass**	0.874	**Pass**
	-2		0.044	**Pass**	0.838	**Pass**
	-1		0.488	**Pass**	0.724	**Pass**
	1		0.920	**Pass**	0.289	**Pass**
	2		0.536	**Pass**	0.683	**Pass**
	3		0.016	**Pass**	1.000	**Pass**
	4		0.997	**Pass**	0.688	**Pass**
4.	Audio					
	-4	38963904	0.829	**Pass**	0.791	**Pass**
	-3		0.609	**Pass**	0.661	**Pass**
	-2		0.580	**Pass**	0.968	**Pass**
	-1		0.973	**Pass**	0.441	**Pass**
	1		0.326	**Pass**	0.327	**Pass**
	2		0.772	**Pass**	0.075	**Pass**
	3		0.186	**Pass**	0.065	**Pass**
	4		0.498	**Pass**	0.146	**Pass**
5.	Video					

	-4	36286656	0.576	**Pass**	0.324	**Pass**
	-3		0.433	**Pass**	0.336	**Pass**
	-2		0.774	**Pass**	0.118	**Pass**
	-1		0.178	**Pass**	0.049	**Pass**
	1		0.578	**Pass**	0.079	**Pass**
	2		0.185	**Pass**	0.146	**Pass**
	3		0.260	**Pass**	0.131	**Pass**
	4		0.011	**Pass**	0.116	**Pass**
6.	Pdf					
	-4	4131648	0.693	**Pass**	0.578	**Pass**
	-3		0.765	**Pass**	0.162	**Pass**
	-2		0.136	**Pass**	0.079	**Pass**
	-1		0.675	**Pass**	0.231	**Pass**
	1		0.732	**Pass**	0.197	**Pass**
	2		0.295	**Pass**	0.339	**Pass**
	3		0.729	**Pass**	0.593	**Pass**
	4		0.744	**Pass**	0.626	**Pass**
7.	Application					
	-4	25517696	0.414	**Pass**	0.447	**Pass**
	-3		0.754	**Pass**	0.368	**Pass**
	-2		0.502	**Pass**	0.302	**Pass**
	-1		0.233	**Pass**	0.655	**Pass**
	1		0.662	**Pass**	0.118	**Pass**
	2		0.740	**Pass**	0.197	**Pass**
	3		0.932	**Pass**	0.317	**Pass**
	4		0.978	**Pass**	0.398	**Pass**

Conclusions

This study led to the conclusion that for devices with low resources, it is possible to maintain an acceptable level of usability, functionality, and security by employing lightweight encryption techniques. Also, this study came to the conclusion that by making the proposed algorithm more complex to attack and by improving the statistical properties of the generated ciphertext sequences, it may be possible to increase security while keeping usability and functionality.

References

[1] M. Gutfleisch, J. H. Klemmer, N. Busch, Y. Acar, M. A. Sasse, and S. Fahl, “How does usable security (not) end up in software products? results from a qualitative interview study,” in *2022 IEEE Symposium on Security and Privacy (SP)*, 2022, pp. 893–910.

[2] M. L. Villani *et al.*, “A Modular Architecture of Command-and-Control Software in Multi-Sensor Systems Devoted to Public Security,” *Information*, vol. 14, no. 3, p. 162, 2023.

[3] G. Bella, J. Ophoff, K. Renaud, D. Sempreboni, and L. Viganò, “Perceptions of beauty in security ceremonies,” *Philos. Technol.*, vol. 35, no. 3, p. 72, 2022.

[4] S. Gupta, M. Kumar, G. Singh, and A. Chanda, “Development of a Novel Footwear Based Power Harvesting System,” *e-Prime-Advances Electr. Eng. Electron. Energy*, p. 100115, 2023.

[5] H. Chen *et al.*, “BMS: Bandwidth-aware Multi-interface Scheduling for energy-efficient and delay-constrained gateway-to-device communications in IoT,” *Comput. Networks*, p. 109645, 2023.

[6] M. Abujoodeh, L. Tamimi, and R. Tahboub, “Toward Lightweight Cryptography: A Survey,” in *Computational Semantics*, IntechOpen, 2023.

[7] J. Kuang, Y. Guo, and L. Li, “IIoTBC:

A Lightweight Block Cipher for Industrial IoT Security.," *KSII Trans. Internet Inf. Syst.*, vol. 17, no. 1, 2023.

[8] F. Thabit, S. Alhomdy, A. H. A. Al-Ahdal, and S. Jagtap, "A new lightweight cryptographic algorithm for enhancing data security in cloud computing," *Glob. Transitions Proc.*, vol. 2, no. 1, pp. 91–99, 2021.

[9] S. S. Dhanda, B. Singh, and P. Jindal, "Lightweight cryptography: a solution to secure IoT," *Wirel. Pers. Commun.*, vol. 112, pp. 1947–1980, 2020.

[10] G. Bansod, N. Pisharoty, and A. Patil, "BORON: an ultra-lightweight and low power encryption design for pervasive computing," *Front. Inf. Technol. Electron. Eng.*, vol. 18, pp. 317–331, 2017.

[11] S. Ashaq, M. Nazish, M. Ali, I. Sultan, and M. T. Banday, "FPGA Implementation of PRESENT Block Cypher with Optimised Substitution Box," in *2022 Smart Technologies, Communication and Robotics (STCR)*, 2022, pp. 1–6.

[12] N. Im, S. Choi, and H. Yoo, "S-Box Attack Using FPGA Reverse Engineering for Lightweight Cryptography," *IEEE Internet Things J.*, vol. 9, no. 24, pp. 25165–25180, 2022.

[13] S. R. Ferianto and A. D. Handayani, "Strict Avalanche Criterion (SAC) Test on Lightweight Blockcipher Algorithms".

[14] C. Tezcan, "Key lengths revisited: GPU-based brute force cryptanalysis of DES, 3DES, and PRESENT," *J. Syst. Archit.*, vol. 124, p. 102402, 2022.

[15] K. Jang, G. Song, H. Kim, H. Kwon, H. Kim, and H. Seo, "Efficient implementation of PRESENT and GIFT on quantum computers," *Appl. Sci.*, vol. 11, no. 11, p. 4776, 2021.

[16] J. Guo, T. Peyrin, A. Poschmann, and M. Robshaw, "The LED block cipher," in *Cryptographic Hardware and Embedded Systems–CHES 2011: 13th International Workshop, Nara, Japan, September 28–October 1, 2011. Proceedings 13*, 2011, pp. 326–341.

[17] B. Chaitra, V. G. K. Kumar, and R. C. Shatharama, "A survey on various lightweight cryptographic algorithms on FPGA," *IOSR J. Electron. Commun. Eng.*, vol. 12, no. 1, pp. 45–59, 2017.

[18] M. F. Mushtaq, S. Jamel, A. H. Disina, Z. A. Pindar, N. S. A. Shakir, and M. M. Deris, "A survey on the cryptographic encryption algorithms," *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 11, pp. 333–344, 2017.

[19] A. Ramesh and A. Suruliandi, "Performance analysis of encryption algorithms for Information Security," in *2013 International Conference on Circuits, Power and Computing Technologies (ICCPCT)*, 2013, pp. 840–844.

[20] C. Liu, Y. Zhang, J. Xu, J. Zhao, and S. Xiang, "Ensuring the security and performance of IoT communication by improving encryption and decryption with the lightweight cipher uBlock," *IEEE Syst. J.*, vol. 16, no. 4, pp. 5489–5500, 2022.

[21] L. Jin, M. Yi, Y. Xiao, L. Sun, Y. Lu, and H. Liang, "A dynamically reconfigurable entropy source circuit for high-throughput true random number generator," *Microelectronics J.*, p. 105690, 2023.

[22] J. Rajski, M. Trawka, J. Tyszer, and B. Wlodarczak, "A Lightweight True Random Number Generator for Root of Trust Applications," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, 2023.

[23] A. Slavković and J. Seeman, "Statistical data privacy: A song of privacy and utility," *Annu. Rev. Stat. Its Appl.*, vol. 10, 2023.