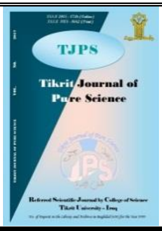
Tikrit Journal of Pure Science
ISSN: 1813 – 1662 (Print) --- E-ISSN: 2415 – 1726 (Online)
Journal Homepage: http://tjps.tu.edu.iq/index.php/j



Modify PRESENT Algorithm by New technique and key Generator by External unit

Ahmed Abdullah Khalil, Awny Muhammed Kaftan, Maytham Mustafa Hammoud
Department of Computer Science, Tikrit University, Tikrit, Iraq

https://doi.org/10.25130/tjps.v28i2.1340

ARTICLE INFO.

Article history:
-Received: 27 / 8 / 2022
-Accepted: 13 / 10 / 2022
-Available online: 26 / 4 / 2023

Keywords: lightweight algorithm, IOT, PRESENT, cryptography, Block cipher

Corresponding Author:
Name: Ahmed Abdullah Khalil
E-mail: ahmed.a.khalil35531@st.tu.edu.iq
Tel:

ABSTRACT

Recently, the widespread use of devices in the field of electronics has caused people to worry about safety. Using traditional cryptographic algorithms to build a complete cryptographic environment for embedded applications or the Internet of Things (IoT) will not be possible due to space, power, and speed limitations. The focus is on lightweight encryption to overcome these problems. In this paper, we have developed the (PRESENT) algorithm, which is among the lightweight algorithms that are used in the IOT and that keeps pace with the requirements of communication speed at the present time. We also used a physical part with the developed algorithm (an external unit that is a USB and using it as a fingerprint for the system) to generate the key while ensuring that the system does not work until after the external unit is connected to the calculator to provide more security for the system from breaches. The developed algorithm achieved a high speed of execution faster than the original algorithm and passed standard tests for various files (text, image, audio, video). In addition to increasing the degree of complexity than the original algorithm.

1. Introduction

As a result of the rapid development in the field of communications, especially in the field of data transmission, which prompted researchers to work on designing lightweight encryption algorithms to keep pace with this development and to achieve speed in the transmission of information, preservation and confidentiality. Because of power consumption, size, and speed limits, creating a full-fledged cryptography environment in embedded systems is not feasible. Lightweight cryptography that requires the least amount of memory space is preferred because of these restrictions. "Lightweight cryptography" refers to this type of cryptography **[1].** Lightweight algorithms outperform more traditional ones in the internet security protocol and provide an appropriate level of protection. Different applications have made use of various lightweight algorithms. Most lightweight cryptography methods, on the other hand, experience a security-complexity tradeoff **[2].**

In the last several years, a slew of small and light algorithms has emerged. These include the likes of the PRESENT, CLEFIA, LED, SEA and KANTAN algorithms **[3].**

The lightweight block cipher PRESENT, also known as PRESENT-80 and PRESENT-128, was created by Bogdanov et al. in 2007 and supports key lengths of 80 and 128. The cryptography world paid PRESENT a lot of attention since it was simple to use and secure. **[4]**. A diferential cryptanalysis for the 16-round PRESENT with a 265 time complexity was proposed by Wang in 2008 **[4]**. In 2009, Kenji Ohkuma showed how linear cryptanalysis might be used to break 24 rounds of PRESENT **[5]**. Cho utilized linear cryptanalysis to attack 25 rounds of PRESENT 80 in 2010 **[6]**. In 2015, Tay et al. announced an FPGA implementation of the compressed PRESENT cipher. By employing an 8-bit data channel and basic control logic in the form of counters, they were able to reduce the required number of S-Boxes **[7]**. In 2016, Liu et al. used Slim Modified Linear Cryptanalysis on a PRESENT-like cipher with public n-round key recovery S-boxes. In their attack, they have a good way to tell the difference between the right and wrong keys **[8]**. In 2020 Chatterjee & Chakraborty. For the PRESENT decoding process, they proposed a new lightweight encoder that modifies the previous encoder by reducing the cipher round, changing the Key Register update technology, and introducing a new layer between S-box and P-layer **[2]**.

In addition, the original algorithm (PRESENT) no longer keeps pace with the speed of communications, especially in video, audio and some images, as there is a delay in implementation of up to several minutes, which leads to a (delay) in communication between the two sending ends and sometimes the execution stops if the files are large in size. Our update addressed such a problem as well. Our change to the original algorithm's working technique and the addition of an external key generation unit added more complexity to the original algorithm.

In this paper, we will increase the speed of the PRESENT algorithm and improve its security performance by reducing the process in the key generation mechanism, as well as increasing the complexity of the algorithm.

2. Description of present

The most popular light-weight encryption method is PRESENT, a straightforward block cipher that is available for usage anywhere. It was introduced in 2007 and standardized in ISO/IEC 29192, just like CLEFIA. 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 the course of 31 rounds. It has an SPN structure. Each encryption or decryption cycle of the cipher passes through one S-box Layer and one P-Layer **[9].**

The key register creates a 64-bit key per round that updates the key register and performs an XOR operation with plaintext. The approach uses 4-bit input and output Sboxes in the substitution layer to optimize the hardware. The permutation layer is easy to understand and simple. One of the simplest and most efficient lightweight algorithms, PRESENT uses only 1884 GEs and surpasses the competition in terms of performance, security, and storage. High power consumption is one of the algorithm's flaws **[10].**

Differential cryptanalysis, linear cryptanalysis, and brute pressure attacks are among the many attacks that Present is resistant to, most noticeably in PRESENT128. Furthermore, current passed every NIST test **[9] [11]**. In Fig. 1 the block diagram of PRESENT cipher.

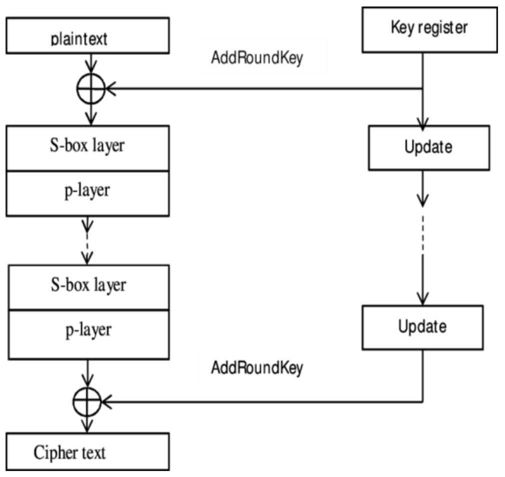


Fig. 1: Block Diagram of PRESENT cipher

The PRESENT algorithm is made up of the following phases:

1. The First Round

The operation Add Round Key will be executed. The status array is XORed with the first string of the plain text (the cipher key).

2. Round Operations

The following operations are carried out in each PRESENT:

- S-boxLayer.
- P-boxLayer.
- add Round Key.

The 16x4-bit S-boxes that make up the substitution layer. Fig.2 shows the S-box. Bit i of the state is transferred to bit position P in the permutation layer (i). Fig.3 depicts the permutation layer.

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

Fig. 2: PRESENT S-box[9]

The key scheduling in PRESENT contains a 61-bit left rotation, an S-box (two S-boxes for a 128-bit key), and an XOR with a round counter. The datapath and key schedule are both handled by the same S-box. The key is rotated to the left 61 bits, and the left-most 4 bits (8 bits for a 128-bit key) are passed through.

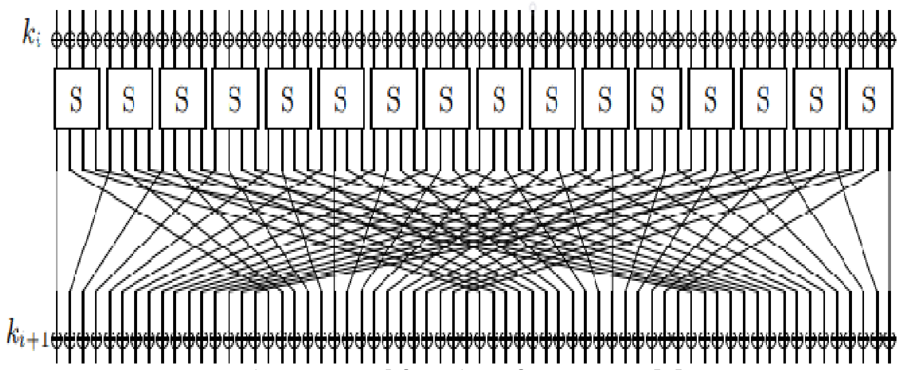


Fig. 3: Round function of PRESENT[9]

The S-box and the round counter value i are XORed with bits $k_{19}k_{18}k_{17}k_{16}k_{15}$ of key, with the least important bit of the round counter on the right.

3. Modify present algorithm

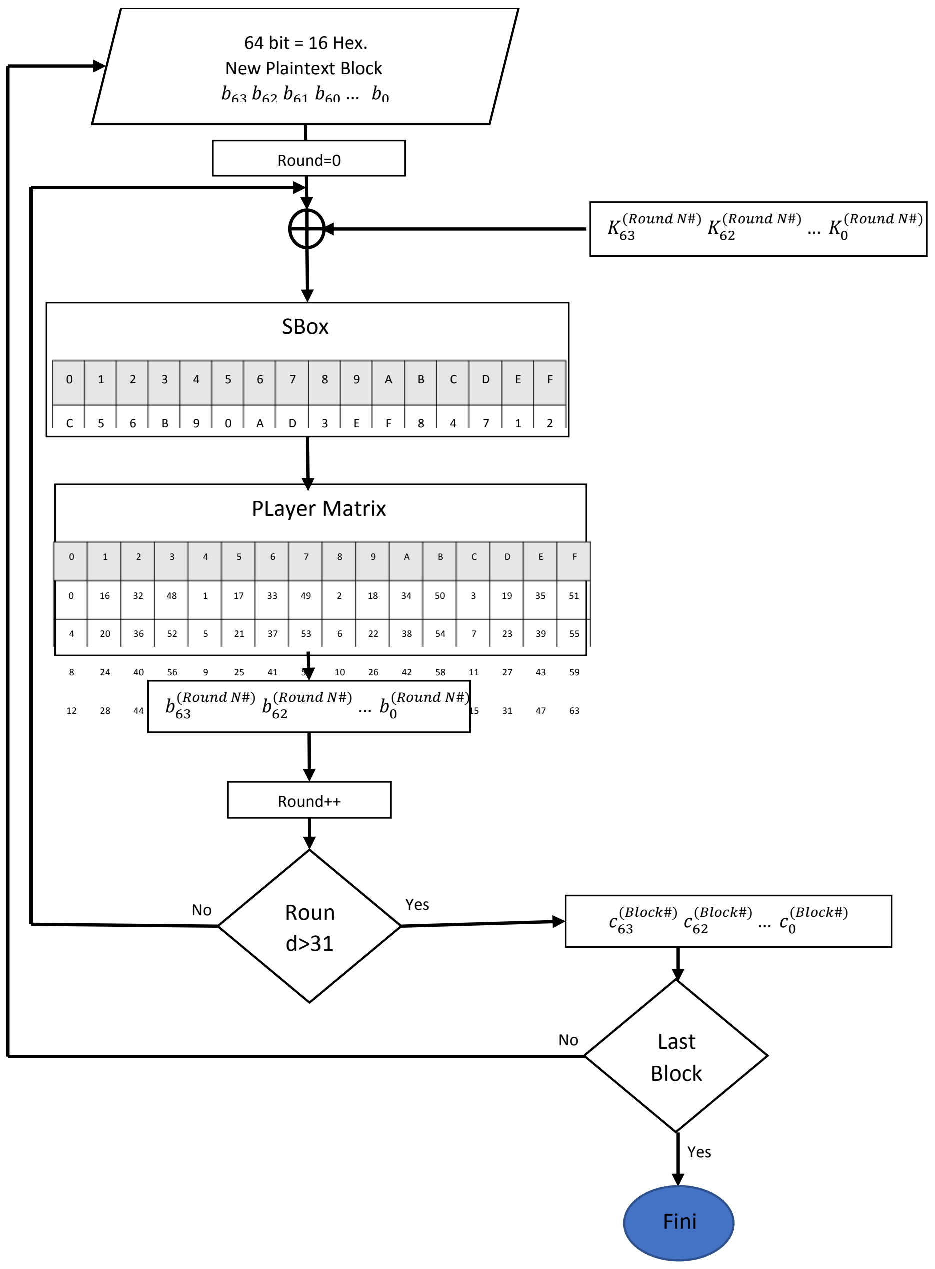
Now, we will present an algorithm developed for the PRESENT algorithm, Because the original algorithm no longer keeps pace with the speed of modern communications, which affected the speed of communication between the sending and receiving ends, as well as the key generation mechanism directly depends on the master key, as each key is generated by performing some operations on the key of the previous round until 32 rounds are completed for each block. Here, we presented a development of the algorithm based on a key generator from an external unit, in addition to a modification to the technique of working rounds within the original algorithm, in addition to a change in dealing with the final key. and the steps of its work are summarized as follows:

1. After dividing the plain text into blocks, each block consists of 64 bits and equals 16 hexadecimal.
2. We insert the first block for the first cycle and it is combined by XORed operation with the 64-bit key to get 16 Hexa.
3. Input the output of the step (2) into the S-Box and we make a permutation or transfer of bits.
4. Enter the output of step (3) into the P-layer Matrix (5*16).
5. Then we test Round we have reached 31 or not
- If (no), we repeat steps (2-5)
- Yes, combine with a block ciphertext Ci

i=0, … 63

6. From 5: have we reached the last block: -
- If (no), we repeat steps (1-6).
- If (yes) then we reach the last block,
- End

-


64 bit = 16 Hex.
New Plaintext Block
$b_{63}\, b_{62}\, b_{61}\, b_{60} \dots\ b_0$
Round=0
$\oplus$
$K_{63}^{(Round\ N\#)}\ K_{62}^{(Round\ N\#)} \dots\ K_0^{(Round\ N\#)}$
SBox
PLayer Matrix
$b_{63}^{(Round\ N\#)}\ b_{62}^{(Round\ N\#)} \dots\ b_0^{(Round\ N\#)}$
Round++
No
Roun d>31
Yes
$c_{63}^{(Block\#)}\ c_{62}^{(Block\#)} \dots\ c_0^{(Block\#)}$
No
Last Block
Yes
Fini

SBox

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

PLayer Matrix

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
8	24	40	56	9	25	41	[illegible]	10	26	42	58	11	27	43	59
12	28	44	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	15	31	47	63

Fig.4 block diagram of Modified PRESENT

- Key Generator unit

This unit an external unit, which is consist of the following contains: -

1. The main part consists of 8 linear feedback shaft registers (8-LFSRs).
2. Memory vector (8*256) as a function (F($x_{1\dots}x_8$)).
3. Secondary part of LFSRs (3-LFSRs) as LFSR-A, LFSR-B and LFSR-C.

- **Algorithm of key Generator unit**

1. The output of main part of LFSRs ($\mathrm{LFSR}_{1\dots}\ \mathrm{LFSR}_8$) is Byte ($x_0^i \dots x_8^i$).
2. Apply :- $\sum_{k=0}^{7} 2^k$ to produce the Byte selector.
3. From step (2) use the Byte selector to give one Byte as address to memory vector (EPROM) to select one Byte.
4. From the (3) we select one bit (000,001,010,011,100,101,110,111).
5. The output of system (3- LFSRs) is three bit ($a_1^i . a_2^i . a_3^i$) and

$$d_R^i = (a_1^i \oplus a_2^i \oplus a_3^i).$$

6. Apply (XOR) operation on the Results (3,4 and the XORing the bits of the output of 1) i.e. $Z^i = f^i \oplus d_L^{i-1} \oplus d_R^{i-1}$

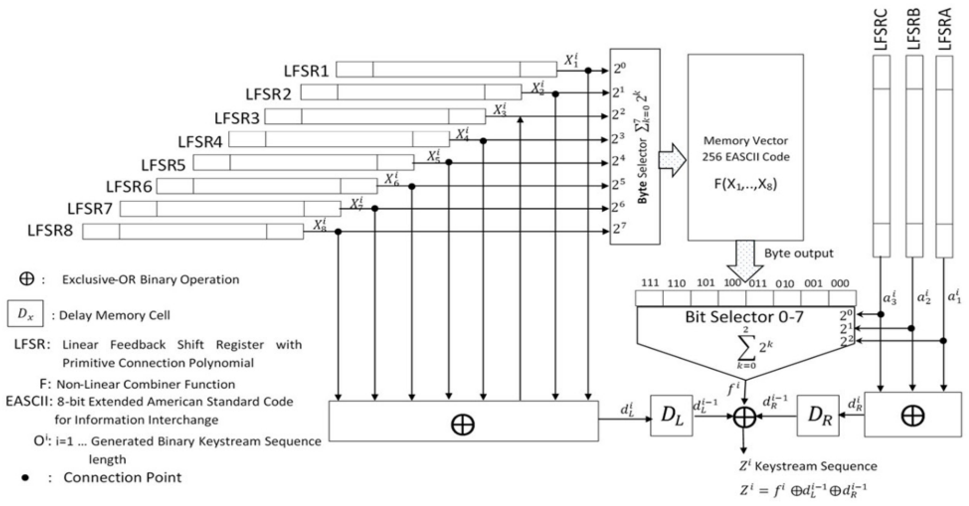

LFSR1
LFSR2
LFSR3
LFSR4
LFSR5
LFSR6
LFSR7
LFSR8
X_1^i … X_8^i
2^0 2^1 2^2 2^3 2^4 2^5 2^6 2^7
Byte Selector $\sum_{k=0}^{7} 2^k$
Memory Vector
256 EASCII Code
F(X1,..,X8)
Byte output
LFSRC
LFSRB
LFSRA
111 110 101 100 011 010 001 000
Bit Selector 0-7
$\sum_{k=0}^{2} 2^k$
a_3^i a_2^i a_1^i
f^i
d_L^i D_L d_L^{i-1}
d_R^{i-1} D_R d_R^i
Z^i Keystream Sequence
$Z^i = f^i \oplus d_L^{i-1} \oplus d_R^{i-1}$
⊕ : Exclusive-OR Binary Operation
D_x : Delay Memory Cell
LFSR: Linear Feedback Shift Register with Primitive Connection Polynomial
F: Non-Linear Combiner Function
EASCII: 8-bit Extended American Standard Code for Information Interchange
O^i: i=1 … Generated Binary Keystream Sequence length
● : Connection Point

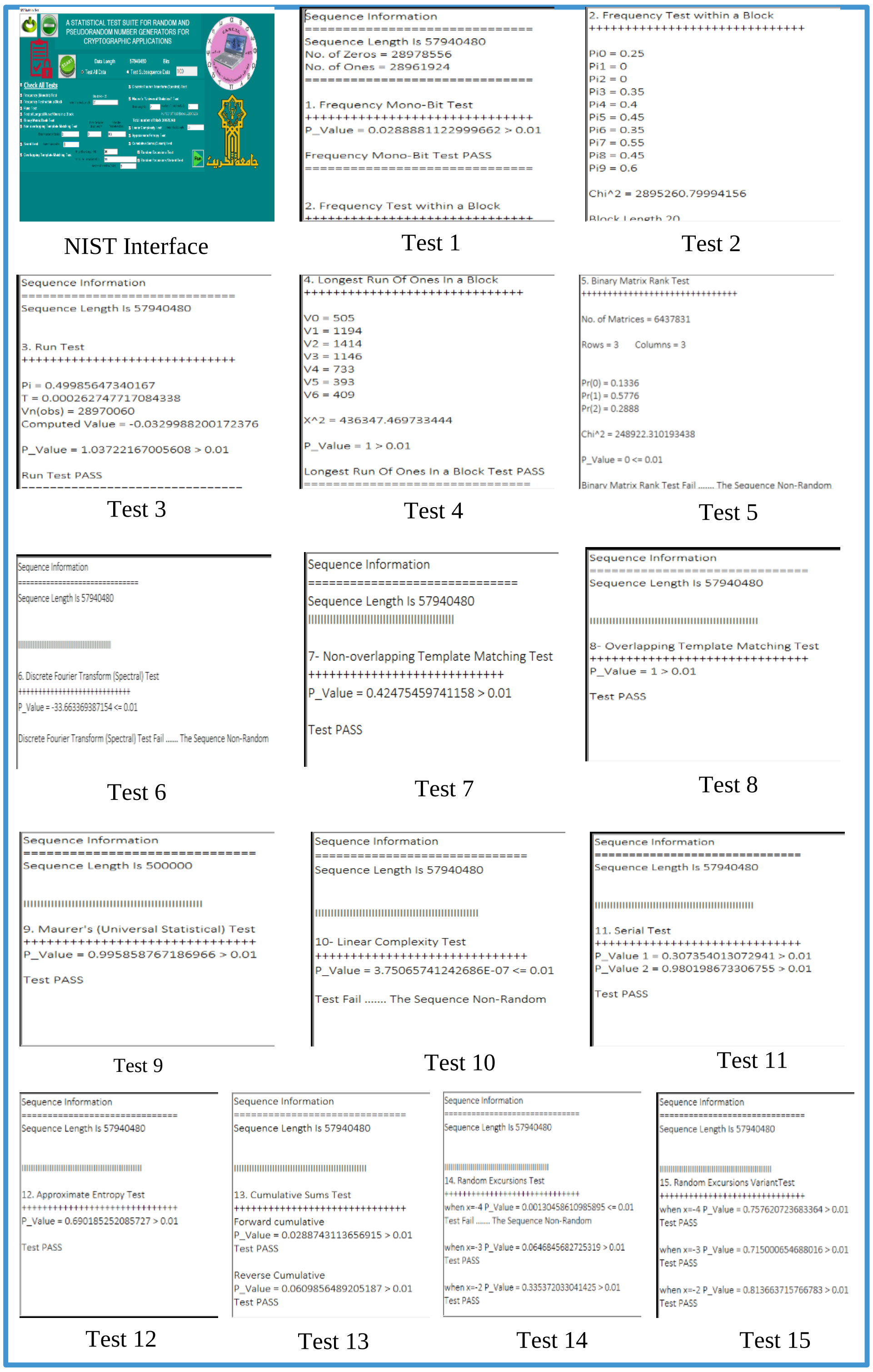
Fig.5 block diagram of the key generation mechanism of the optimized PRESENT algorithm

4. The result and discussion: After implementing the proposed algorithm and comparing the results with the original algorithm as shown below.

TABLE I: Comparison between the original and modified PRESENT algorithm in terms of execution speed

NO	File type	File size	Execution time		Notes
			Original PRESENT	Modified PRESENT	
1.	jpg	96 KB	2 seconds	1.85 seconds	We note that the execution speed of the developed algorithm increased significantly, and the larger the file size, the more clear the difference
2.	jpeg	136 KB	2.45 seconds	2.31 seconds	
3.	ogg	348 KB	4.75 seconds	4.12 seconds	
4.	docx	492 KB	6.97 seconds	5.88 seconds	
5.	Mp4	6.91 MB	3 minutes	1.25 minutes	

It appears in the above table that the developed algorithm handled the speed and was not affected by the file size. The complexity of the algorithm also increased to 2^{64*32} after what was in the original algorithm2^{64}. After that, the results were tested with the National Institute of Standards and Technology (NIST) test group, where one of the encrypted files was tested. Figure (6) shows Test set.


A STATISTICAL TEST SUITE FOR RANDOM AND PSEUDORANDOM NUMBER GENERATORS FOR CRYPTOGRAPHIC APPLICATIONS
START
Data Length 57940480 Bits
Test All Data
Test Subsequence Data 100
Check All Tests
Run
جامعة تكريت

NIST Interface

Test 1
```
Sequence Information
==============================
Sequence Length Is 57940480
No. of Zeros = 28978556
No. of Ones = 28961924
==============================

1. Frequency Mono-Bit Test
++++++++++++++++++++++++++++++
P_Value = 0.0288881122999662 > 0.01

Frequency Mono-Bit Test PASS
==============================

2. Frequency Test within a Block
++++++++++++++++++++++++++++++
```

Test 2
```
2. Frequency Test within a Block
++++++++++++++++++++++++++++++

Pi0 = 0.25
Pi1 = 0
Pi2 = 0
Pi3 = 0.35
Pi4 = 0.4
Pi5 = 0.45
Pi6 = 0.35
Pi7 = 0.55
Pi8 = 0.45
Pi9 = 0.6

Chi^2 = 2895260.79994156

Block Length 20
```

Test 3
```
Sequence Information
==============================
Sequence Length Is 57940480


3. Run Test
++++++++++++++++++++++++++++++

Pi = 0.49985647340167
T = 0.000262747717084338
Vn(obs) = 28970060
Computed Value = -0.0329988200172376

P_Value = 1.03722167005608 > 0.01

Run Test PASS
------------------------------
```

Test 4
```
4. Longest Run Of Ones In a Block
++++++++++++++++++++++++++++++

V0 = 505
V1 = 1194
V2 = 1414
V3 = 1146
V4 = 733
V5 = 393
V6 = 409

X^2 = 436347.469733444

P_Value = 1 > 0.01

Longest Run Of Ones In a Block Test PASS
==============================
```

Test 5
```
5. Binary Matrix Rank Test
++++++++++++++++++++++++++++++

No. of Matrices = 6437831

Rows = 3     Columns = 3


Pr(0) = 0.1336
Pr(1) = 0.5776
Pr(2) = 0.2888

Chi^2 = 248922.310193438

P_Value = 0 <= 0.01

Binary Matrix Rank Test Fail ....... The Sequence Non-Random
```

Test 6
```
Sequence Information
==============================
Sequence Length Is 57940480


||||||||||||||||||||||||||||||

6. Discrete Fourier Transform (Spectral) Test
++++++++++++++++++++++++++
P_Value = -33.663369387154 <= 0.01

Discrete Fourier Transform (Spectral) Test Fail ....... The Sequence Non-Random
```

Test 7
```
Sequence Information
==============================
Sequence Length Is 57940480
||||||||||||||||||||||||||||||

7- Non-overlapping Template Matching Test
++++++++++++++++++++++++++
P_Value = 0.42475459741158 > 0.01

Test PASS
```

Test 8
```
Sequence Information
==============================
Sequence Length Is 57940480

||||||||||||||||||||||||||||||

8- Overlapping Template Matching Test
++++++++++++++++++++++++++++++
P_Value = 1 > 0.01

Test PASS
```

Test 9
```
Sequence Information
================================
Sequence Length Is 500000

||||||||||||||||||||||||||||||

9. Maurer's (Universal Statistical) Test
++++++++++++++++++++++++++++++++
P_Value = 0.995858767186966 > 0.01

Test PASS
```

Test 10
```
Sequence Information
==============================
Sequence Length Is 57940480

||||||||||||||||||||||||||||||

10- Linear Complexity Test
++++++++++++++++++++++++++++++
P_Value = 3.75065741242686E-07 <= 0.01

Test Fail ....... The Sequence Non-Random
```

Test 11
```
Sequence Information
==============================
Sequence Length Is 57940480

||||||||||||||||||||||||||||||

11. Serial Test
++++++++++++++++++++++++++++++
P_Value 1 = 0.307354013072941 > 0.01
P_Value 2 = 0.980198673306755 > 0.01

Test PASS
```

Test 12
```
Sequence Information
==============================
Sequence Length Is 57940480

||||||||||||||||||||||||||||||

12. Approximate Entropy Test
++++++++++++++++++++++++++++++
P_Value = 0.690185252085727 > 0.01

Test PASS
```

Test 13
```
Sequence Information
==============================
Sequence Length Is 57940480

||||||||||||||||||||||||||||||

13. Cumulative Sums Test
++++++++++++++++++++++++++++++
Forward cumulative
P_Value = 0.0288743113656915 > 0.01
Test PASS

Reverse Cumulative
P_Value = 0.0609856489205187 > 0.01
Test PASS
```

Test 14
```
Sequence Information
==============================
Sequence Length Is 57940480

||||||||||||||||||||||||||||||
14. Random Excursions Test
++++++++++++++++++++++++++++++
when x=-4 P_Value = 0.00130458610985895 <= 0.01
Test Fail ....... The Sequence Non-Random

when x=-3 P_Value = 0.0646845682725319 > 0.01
Test PASS

when x=-2 P_Value = 0.335372033041425 > 0.01
Test PASS
```

Test 15
```
Sequence Information
==============================
Sequence Length Is 57940480

||||||||||||||||||||||||||||||
15. Random Excursions VariantTest
++++++++++++++++++++++++++++++
when x=-4 P_Value = 0.757620723683364 > 0.01
Test PASS

when x=-3 P_Value = 0.715000654688016 > 0.01
Test PASS

when x=-2 P_Value = 0.813663715766783 > 0.01
Test PASS
```

Fig. 6: Results of NIST tests for a file encrypted with the modified PRESENT algorithm

After the results of the tests appeared, it was found that there were 11 Pass tests and 4 failed tests, and this result is good for the developed algorithm, as the number of failed tests does not affect the quality and performance of the algorithm.

Conclusions

After applying the developed algorithm, we reached several conclusions, where a foreign key generation unit was added, which contributed to increasing the speed of the algorithm, as the change in Round technology did not affect the time and also the mechanism of using the keys was changed and did not affect the work of the algorithm, but the above-mentioned changes also increased the complexity The algorithm is up to (60%) than the original algorithm and also passed the standard tests of various files. To achieve the goal and develop the original algorithm as imposed in the research. As a future work, researchers can use the idea of an external unit to generate the key using other algorithms such as (LED, CLFFIA, LEA, PICCOLO, SKINNY, SPARX, etc.).

References

[1]. Chaitra, B., Kiran Kumar, V. G., & Shatharama Rai, C. (2017). Comparative Study of cryptographic encryption algorithms. *IOSR Journal of Electronic and Communication Engineering (IOSR-JECE)*, *12*(3).

[2]. Chatterjee, R., & Chakraborty, R. (2020, March). A modified lightweight PRESENT cipher for IoT security. In *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)* (pp. 1-6). IEEE.

[3]. M. Prabu, R. Nenregen, S. Saravana Kumar ,"Comparing Lightweight Algorithms for Embedded System Security" J. Clerk Maxwell, International Journal of Pure and Applied Mathematics, Volume 119 No. 12 2018, 14191-14198.

[4]. Wang, M. (2008, June). Differential cryptanalysis of reduced-round PRESENT. In *International Conference on Cryptology in Africa* (pp. 40-49). Springer, Berlin, Heidelberg.

[5]. Ohkuma, K. (2009, August). Weak keys of reduced-round PRESENT for linear cryptanalysis. In *International Workshop on Selected Areas in Cryptography* (pp. 249-265). Springer, Berlin, Heidelberg.

[6]. Cho, J. Y. (2010, March). Linear cryptanalysis of reduced-round PRESENT. In *Cryptographers' Track at the RSA Conference* (pp. 302-317). Springer, Berlin, Heidelberg.

[7]. Tay, J. J., Wong, M. D., Wong, M. M., Zhang, C., & Hijazin, I. (2015, August). Compact FPGA implementation of PRESENT with Boolean S-Box. In *2015 6th Asia Symposium on Quality Electronic Design (ASQED)* (pp. 144-148). IEEE.

[8]. Liu, G., Jin, C., & Kong, Z. (2016). Key recovery attack for PRESENT using slender-set linear cryptanalysis. *Science China Information Sciences*, *59*(3), 1-14.

[9]. Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M. J., ... & Vikkelsoe, C. (2007, September). PRESENT: An ultra-lightweight block cipher. In *International workshop on cryptographic hardware and embedded systems* (pp. 450-466). Springer, Berlin, Heidelberg.

[10]. Sallam, S., & Beheshti, B. D. (2018, October). A survey on lightweight cryptographic algorithms. In *TENCON 2018-2018 IEEE Region 10 Conference* (pp. 1784-1789). IEEE.

[11].Bagheri, N., Ebrahimpour, R., & Ghaedi, N. (2013). New differential fault analysis on PRESENT. *EURASIP Journal on Advances in Signal Processing*, *2013*(1), 1-10.

تعديل خوارزمية (PRESENT) بواسطة تقنية جديدة ومولد مفتاح بواسطة وحدة خارجية

احمد عبدالله خليل، عوني محمد كفطان، ميثم مصطفى حمود

قسم علوم الحاسوب، كليى علوم الحاسوب والرياضيات ، جامعة تكريت، تكريت، العراق

الملخص

في الآونة الأخيرة، أدى الاستخدام الواسع النطاق للأجهزة في مجال الإلكترونيات إلى قلق الناس بشأن السلامة. لن يكون استخدام خوارزميات التشفير التقليدية لبناء بيئة تشفير كاملة للتطبيقات المضمنة أو إنترنت الأشياء (IOT) ممكنًا بسبب قيود المساحة والطاقة والسرعة. ينصب التركيز على التشفير الخفيف للتغلب على هذه المشاكل. في هذا البحث قمنا بتطوير خوارزمية (PRESENT) والتي هي من ضمن الخوارزميات الخفيفة التي تستخدم في IOT والتي تواكب متطلبات سرعة الاتصالات في الوقت الحاضر . كذلك قمنا باستخدام جزء مادي مع الخوارزمية المطورة (وحدة خارجية عبارة عن USB واستخدامه كبصمة خاصة بالنظام) لتوليد المفتاح مع ضمان عدم عمل المنظومة الا بعد ربط الوحدة الخارجية بالحاسبة لتوفير امان أكثر للمنظومة من الخرق. وقد حققت الخوارزمية المطورة سرعة عالية عند التنفيذ أسرع من الخوارزمية الاصلية واجتازت الاختبارات القياسية ولمختلف الملفات (نص، صورة، صوت، فديو). اضافة الى زيادة درجة التعقيد عن الخوارزمية الاصلية.