Analysis of energy-efficient lightweight cryptograhic algorithm

AIPCP25-CF-ICAIPSS2025-00312 | Article

Analysis of energy-efficient lightweight cryptograhic algorithm

Tukhtajon Kozokova [a], Nozima Akhmedova, Mastura Tursunova

Tashkent University of Information Technologies named after Muhammad al-Khwarizmi, Tashkent, Uzbekistan

[a] *Corresponding author: qozoqovat1516@gmail.com*

Abstract. This article presents a comparative analysis of lightweight cryptographic algorithms used in IoT devices that operate with low resource consumption, along with the results of related research studies. In particular, the operational stages of the PRESENT lightweight encryption algorithm were analyzed and compared with the SPECK encryption algorithm under identical conditions (in the CrypTool 2.1 environment). The analysis results showed that to ensure security in IoT devices, the PRESENT lightweight encryption algorithm needs to be improved and the number of rounds reduced.

INTRODUCTION

Currently, rapidly developing Internet of Things (IoT) technologies play an important role in everyday life. IoT consists of several devices connected to each other and constantly exchanging data and information. As IoT technologies develop, security requirements are also growing. According to a report by ThreatLabz, the number of attacks on IoT devices is increasing by 45% annually [1]. Based on this data, it can be said that in order to strengthen security requirements, there is an urgent need to improve the efficiency and sophistication of lightweight cryptography algorithms, which form the basis of IoT technology security.

Lightweight cryptography refers to cryptographic algorithms designed for devices with limited computing power, memory, and energy consumption. The history of these algorithms coincides with the development of IoT technologies. In 2011, the National Institute of Standards and Technology (NIST) held a competition. Algorithms such as PRESENT, SIMON, SPECK, KATAN, and KLEIN took part in it. These algorithms were small in size and had low resource requirements, which gave them an advantage over other algorithms.

LITERATURE REVIEW AND METHODOLOGY

A large number of literature and articles were analyzed within the scope of the topic. Several studies were conducted to analyze lightweight encryption algorithms and their effectiveness, including the PRESENT encryption algorithm. In particular, the article by Muhammad Rana et al. [2] presents an analysis of lightweight encryption algorithms used in Internet of Things devices due to resource constraints. These algorithms were analyzed in terms of security effectiveness, speed, and energy consumption in devices. Christophe De Caenier et al. [3] developed the KATAN cipher, which uses block sizes of 32, 48, and 64 bits and an 80-bit key. In this algorithm, the key is written to the device. A copy of the text is loaded into the register, and the encryption process begins.

The PRINCE cipher, developed by Julia Borghoff et al. [4], uses a 64-bit block size and a 128-bit key. This cipher distributes the key throughout the text and has special mechanisms to prevent cryptographic attacks. Deukjo Hong et al. [5] developed a cipher based on the Feistel network. It also uses a 64-bit block size and a 128-bit key. The algorithm uses simple XOR and shift operations over the functions $F_0$and F_{31}. Ray Bolier et al. [6] developed ciphers with different block and key sizes. These ciphers are designed to improve the structure of hardware and software systems on processors. The cipher uses modular addition, XOR, and cyclic shift operations. An overview of lightweight cryptographic algorithms is discussed by Kong Jia Hao et al. They provide information on optimized algorithms for efficient use of energy and resources.

Gaurav Bansod [7] explained the hybrid PRESENT-GRP method. In this method, input data blocks are distributed through the PRESENT S-box, and the output data is reflected using the PRESENT-GRP algorithm and then transferred to the permutation level. The PRESENT-GRP algorithm is designed in a 4x4 S-box structure, which aims to reduce complexity and power consumption.

The design for performing operations on a 64-bit value is such that it uses only 16 4-bit PRESENT S-boxes and passes them to PRESENT GRP for permutation. The memory requirements of the PRESENT-GRP hybrid structure are significantly lower than those of existing algorithms. GRP uses seven stages to reduce the equivalent of P-box gates. The bits are grouped as follows: the first group contains bit 0 and bit 64, the second group contains bit 1 and bit 65, and so on. The implementation of encryption algorithms in lightweight hardware and software was also discussed by George Hatzivassilis et al. [8].

Power consumption is currently one of the most important factors in Internet of Things devices. In response to this requirement, Muhammad Rana published the following research results in his article. Figure 1 shows the power consumption of the Proposed, Piccolo, Rectangle, Print, Present, Iceberg, and High algorithms in the form of columns [2].

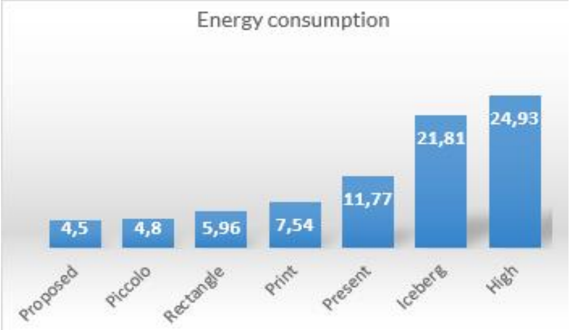


FIGURE 1. Energy consumption of various lightweight algorithms

Description of a lightweight algorithm. PRESENT is a lightweight block cipher algorithm developed by Andrei Bogdanov and co-authors in 2007. This algorithm is primarily intended for systems with limited hardware and software resources, such as Internet of Things devices, RFID technologies, and sensor networks. It was standardized by the International Organization for Standardization (ISO/IEC) in 2012 and is primarily adapted for lightweight cryptography.

Advantages of the algorithm. Optimized for a small number of transistors, low power consumption, and security. Despite its many advantages, the PRESENT algorithm has a number of disadvantages. The encryption speed of the algorithm in software is low. PRESENT was developed primarily for efficient operation in a hardware environment. The speed of operation in software is lower than that of other lightweight algorithms. Another disadvantage of the algorithm is the key length. An 80-bit key length is not sufficient for long-term security, as with increasing computing power, the key can be cracked by brute force. There is a 128-bit PRESENT algorithm, but the 80-bit key length variant is more commonly used. The algorithm is resistant to differential and linear cryptanalysis. The algorithm consists of 31 rounds, which is more than other lightweight encryption algorithms. This can lead to increased energy consumption. The algorithm does not include additional mechanisms for authentication or data integrity. This may make it unsuitable for certain security requirements.

How it works. The PRESENT block cipher algorithm is an example of a Substitution-Permutation Network (SPN) and consists of 31 rounds. The block length is 64 bits, and the key length is 80 and 128 bits. Considering the areas of application, the 80-bit key length version is widely used. This usually provides the level of security required for low-security applications.

The algorithm consists of 31 rounds, XOR operations with input data $1 \leq i \leq 32$ for round key K_i, where K_{32} is used for subsequent whitening, linear permutation at the bit level, and a nonlinear permutation layer. The nonlinear layer uses a single 4-bit S-box, which is used 16 times in parallel in each round. The encryption algorithm is shown in Figure 1, and the sequence of each stage is defined as follows.

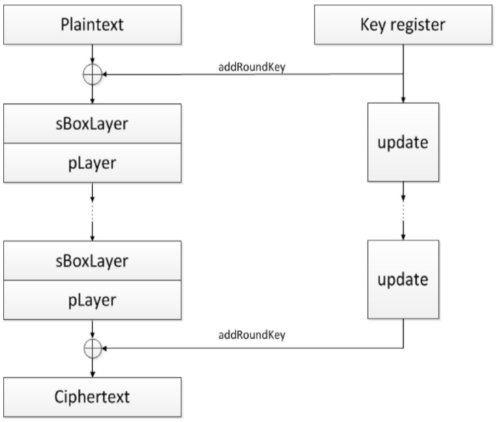


FIGURE 2. Description of the current encryption algorithm

Add Round Key. Given a round key $K = k_{63}^{i} \dots \dots k_{0}^{i}$, where $1 \leq i \leq 32$, and the current state $b_{63} \dots \dots b_0$ add Round Key consists of $0 \leq j \leq 63$,

$$b_j \rightarrow b_j \oplus \kappa_j^i \tag{1}$$

Layer s Box. In the PRESENT encryption algorithm using S-box 4-bit S: $F_2^4 \longrightarrow F_2^4$ is represented in the hexadecimal number system S-box in Table 1 below.

TABLE 1. Hexadecimal value of table S

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

The current state of sBoxLayer: $b_{63} \dots \dots b_0$ is considered as sixteen 4-bit words. $w_{15} \dots \dots w_0$, where $w_i = b_{4*i+3}||b_{4*i+2}||b_{4*i+1}||b_{4*i}$ for $0 \leq i \leq 15$, and the output $S(w_i)$ is specified.

pLayer. The permutation table for the PRESENT encryption algorithm is shown in Table 2. Bit i takes the values $P(i)$.

The PRESENT algorithm uses 80- or 128-bit keys. The sequence of operations for an 80-bit key is shown below. The user key is stored in the K register and is represented as follows:

$$K = k_{79}k_{78} \dots \dots k_0 \tag{2}$$

Where k_{79} is the most significant (leftmost) bit and $k_0$is the least significant (rightmost) bit. Extracting the round key. In each round, a 64-bit round key K_i is extracted, consisting of the leftmost 64 bits of the current state of register K.

$$K_i = k_{79}k_{78} \dots \dots k_{16} = \kappa_{63} \dots \dots \kappa_0 \tag{3}$$

TABLE 2. Permutation table

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

Here, K_i generates an 80-bit key, and its 64 bits are extracted from the left side of the K register.

Key update: After generating the round key K_i, the K register is updated. The update process is as follows.

The bits of the K register are shifted one position to the left. At the same time, the leftmost 61 bits are shifted to the right (left shift operation). Using the S-block, a nonlinear substitution is applied to the upper 4 bits of the key. Depending on the round number, the last bits of the key are updated.

In each round, a new key is generated in the order specified above, and the K register is updated. This process is called key scheduling and plays an important role in improving the reliability of the algorithm. Generating a new key in each round complicates the algorithm and the process of analyzing it.

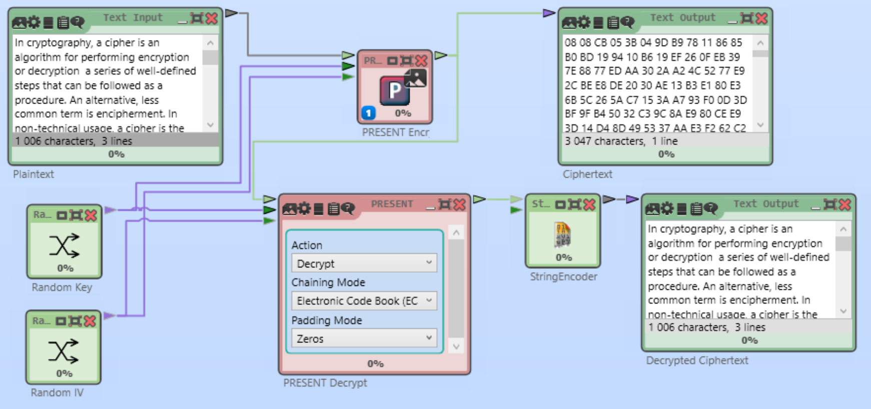
RESEARCH RESULTS

The PRESENT encryption algorithm was tested in the CrypTool 2.1 software environment, and the Speck block cipher was selected as the second algorithm for comparison. The following results were obtained. The following plaintext was entered for the PRESENT encryption algorithm. The algorithm's operation is shown in Figure 3.

In Figure 3, the PRESENT encryption algorithm uses the plaintext "In cryptography, a cipher is an algorithm for performing encryption or decryption a series of well-defined steps that can be followed as a procedure. An alternative, less common term is encipherment. In non-technical usage, a cipher is the same thing as a code; however, the concepts are distinct in cryptography. In classical cryptography, ciphers were distinguished from codes. Codes operated by substituting according to a large codebook which linked a random string of characters or numbers to a word or phrase. For example, UQJHSE could be the code for Proceed to the following coordinates. When using a cipher the original information is known as plaintext, and the encrypted form as ciphertext. The ciphertext message contains all the information of the plaintext message, but is not in a format readable by a human or computer without the proper mechanism to decrypt it; it should resemble random gibberish to those not intended to read it." as plaintext and outputs the ciphertext in the "Text output" field.

The lightweight SPECK encryption algorithm was chosen to compare the capabilities of the algorithm. SPECK is a popular lightweight encryption algorithm proposed by the US National Security Agency (NSA) in 2013. Its purpose is to provide security on a limited number of devices.

In Figure 4, the SPECK encryption algorithm received "In cryptography, a cipher is an algorithm for performing encryption or decryption a series of well-defined steps that can be followed as a procedure. An alternative, less common term is encipherment. In non-technical usage, a cipher is the same thing as a code; however, the concepts are distinct in cryptography. In classical cryptography, ciphers were distinguished from codes. Codes operated by substituting according to a large codebook which linked a random string of characters or numbers to a word or phrase. For example, UQJHSE could be the code for Proceed to the following coordinates. When using a cipher the original information is known as plaintext, and the encrypted form as ciphertext. The ciphertext message contains all the information of the plaintext message, but is not in a format readable by a human or computer without the proper mechanism to decrypt it; it should resemble random gibberish to those not intended to read it." as plaintext, and the ciphertext was output to the "Text output" field.



Text Input

In cryptography, a cipher is an algorithm for performing encryption or decryption a series of well-defined steps that can be followed as a procedure. An alternative, less common term is encipherment. In non-technical usage, a cipher is the

1 006 characters, 3 lines

Text Output

```
08 08 CB 05 3B 04 9D B9 78 11 86 85
B0 BD 19 94 10 B6 19 EF 26 0F EB 39
7E 88 77 ED AA 30 2A A2 4C 52 77 E9
2C BE E8 DE 20 30 AE 13 B3 E1 80 E3
6B 5C 26 5A C7 15 3A A7 93 F0 0D 3D
BF 9F B4 50 32 C3 9C 8A E9 80 CE E9
3D 14 D4 8D 49 53 37 AA E3 F2 62 C2
```

3 047 characters, 1 line

PRESENT

Action: Decrypt

Chaining Mode: Electronic Code Book (EC

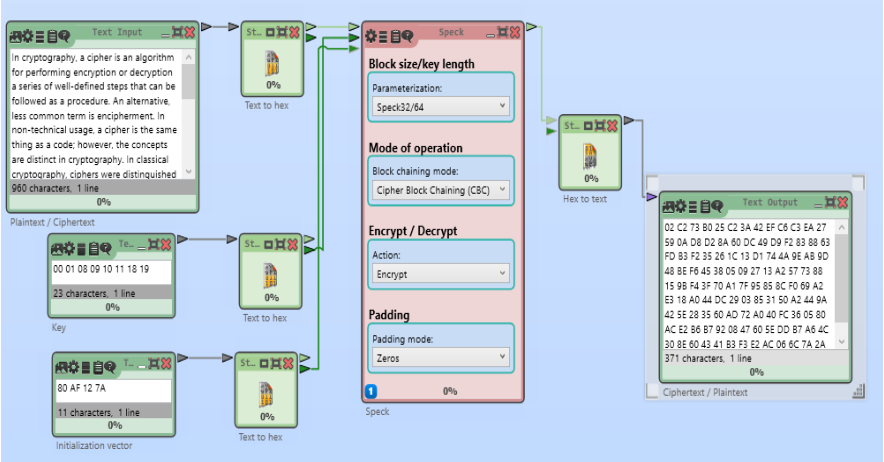
Padding Mode: Zeros

Text Output

In cryptography, a cipher is an algorithm for performing encryption or decryption a series of well-defined steps that can be followed as a procedure. An alternative, less common term is encipherment. In non-technical usage, a cipher is the

1 006 characters, 3 lines

FIGURE 3. The process of working with the PRESENT encryption algorithm in software



In cryptography, a cipher is an algorithm for performing encryption or decryption a series of well-defined steps that can be followed as a procedure. An alternative, less common term is encipherment. In non-technical usage, a cipher is the same thing as a code; however, the concepts are distinct in cryptography. In classical cryptography, ciphers were distinguished

960 characters, 1 line

00 01 08 09 10 11 18 19

23 characters, 1 line

80 AF 12 7A

11 characters, 1 line

Block size/key length

Parameterization: Speck32/64

Mode of operation

Block chaining mode: Cipher Block Chaining (CBC)

Encrypt / Decrypt

Action: Encrypt

Padding

Padding mode: Zeros

```
02 C2 73 B0 25 C2 3A 42 EF C6 C3 EA 27
59 0A D8 D2 8A 60 DC 49 D9 F2 83 88 63
FD B3 F2 35 26 1C 13 D1 74 4A 9E AB 9D
48 BE F6 45 38 05 09 27 13 A2 57 73 88
15 9B F4 3F 70 A1 7F 95 85 8C F0 69 A2
E3 18 A0 44 DC 29 03 85 31 50 A2 44 9A
42 5E 28 35 60 AD 72 A0 40 FC 36 05 80
AC E2 B6 B7 92 08 47 60 5E DD B7 A6 4C
30 8E 60 43 41 B3 F3 E2 AC 06 6C 7A 2A
```

371 characters, 1 line

FIGURE 4. The SPECK encryption algorithm in action in the software

An analytical table was compiled based on the results obtained. Table 3 shows the results of the PRESENT and SPECK encryption algorithms. The table shows the input data for the algorithms (plaintext, key, and initialization vector) and their output results (ciphertext and decrypted plaintext).

When selecting encryption algorithms for Internet of Things (IoT) devices, energy efficiency is an important consideration. Table 4 presents a comparative analysis of the energy efficiency of the PRESENT and SPECK encryption algorithms for IoT devices.

TABLE 3. Mutual properties of algorithms

Feature	PRESENT	SPECK
Plaintext	Analysis of symmetric block ciphers using CrypTool 2.1	Analysis of symmetric block ciphers using CrypTool 2.1
Key	Random key	00 01 08 09 10 11 18 19
IV (Initialization Vector)	Random IV	80 AF 12 7A
Ciphertext	67 9C A2 76 3F 9D 13 32 DB B3 50 A6	A7 32 B7 A7 33 6C 72 28

TABLE 4. Comparative table of symmetric encryption algorithms by energy efficiency

Feature	PRESENT	SPECK
Block size	64 bits	32, 64, 128 bits
Key length	80, 128 bits	64, 96, 128 bits
Number of rounds	31	22
Energy consumption for encryption	4.5 µJ	2.5 µJ
Resource requirement	Low	Very low
Security level	High	Medium to high
Speed	Slower in software implementation	Faster in software implementation

CONCLUSIONS

In conclusion, lightweight cryptographic algorithms provide effective encryption and security capabilities for Internet of Things devices. The lightweight cryptographic algorithm PRESENT is one of them. There is a need for lightweight cryptographic algorithms that reduce energy consumption in IoT devices, which indicates that there are challenges in further improving the performance of lightweight cryptographic algorithms. Improving the performance of a lightweight encryption algorithm suitable for Internet of Things devices (medical devices, industrial Internet of Things) with high security requirements has been identified as an important task. The following results were presented as a result of a comparative analysis. In particular, the PRESENT algorithm is a good choice when hardware resources are limited. It is suitable for low-power IoT devices. If energy efficiency and speed are required, the SPECK encryption algorithm is preferable. If security and hardware resources need to be minimized, PRESENT is a good choice. Reducing the number of iterations to 28 to reduce the time required to increase speed leads to optimization of delays and, at the same time, to a reduction in energy consumption. The practical results of this proposal were set as a task for implementation in the next stage of the research.

REFERENCES

1. Zscaler ThreatLabz. (2024, November 22). New ThreatLabz report: Mobile remains the top threat vector with 111% spyware growth. Zscaler.
2. Rana, M., Mamun, Q, Islam, R. Balancing Security and Efficiency: A Power Consumption Analysis of a Lightweight Block Cipher. Electronics 2024, 13, 4325. https://doi.org/ 10.3390/electronics13214325
3. Christophe De Canniere, Orr Dunkelman, Miroslav Knezevic., KATAN and KTANTAN-a family of small and efficient hardware-oriented block ciphers., Cryptographic Hardware and Embedded Systems-CHES 2009., Springer LNCS, vol. 5747, 2009, pp. 272–288.
4. J. Borghoff., PRINCE–a low-latency block cipher for pervasive computing applications., Advances in Cryptology–ASIACRYPT., Springer LNCS, vol. 7658, 2012, pp. 208–225.
5. Ray Beaulieu, Douglas Shors, Jason Smith, StefanTreatman Clark, Bryan Weeks, Louis Wingers., The SIMON and SPECK families of lightweight block ciphers., IACR Cryptology ePrint Archive (2013).
6. Deukjo Hong, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee., HIGHT: A new block cipher suitable for low-resource device., Cryptographic Hardware and Embedded Systems., Springer Berlin Heidelberg, 2006, pp. 46–59.
7. Gaurav Bansod, Nishchal Raval., Implementation of a new lightweight encryption design for embedded security, IEEE Trans. Inf. Forensics Security. 10 (1) (2015) 142–151.
8. G Hatzivasilis, K Fysarakis, I Papaestathi, H Favas., Review of light weight block ciphers., J. Cryptogr. Eng. 8 (2) (2017) 141–184.
9. Dixit R, Kumar L, Verma S, Gupta K, Jain S., An overview of lightweight cipher., CEUR Workshop Proceedings. https://doi.org/10.1016/j.aci.2018.
10. Bogdanov A, Knudsen L. Leander G, Paar "PRESENT: An ultra-lightweight block cipher"., In Workshop on Cryptographic Hardware and Embedded Systems (pp. 450-466). Springer, Berlin, Heidelberg.
11. Fernando M., Mahmud S., Wang S., Lightweight encryption for IoT applications: A comprehensive review., Discover Internet of Things, 3(1), 100100. https://doi.org/10.1016/j.discint.2023.100100 .
12. Qozoqova T.Q., Shamshiyeva B.M., Applying the CryptoSMT software tool to symmetric block encryption algorithms. pp-750-754
13. Kozokova T., Abdullayeva G., Analysis of lightweight cryptographic algorithms in information security., Second International Scientific and Practical Conference on Actual Problems of Mathematical Modeling and Information Technology-APMMIT2024, Volume 3377, https://doi.org/10.1063/5.0299641