Cybersecurity

RESEARCH

Open Access

RBFK cipher: a randomized butterfly architecture-based lightweight block cipher for IoT devices in the edge computing environment

Sohel Rana[1], M. Rubaiyat Hossain Mondal[1] and Joarder Kamruzzaman[2*]

Abstract

Internet security has become a major concern with the growing use of the Internet of Things (IoT) and edge computing technologies. Even though data processing is handled by the edge server, sensitive data is generated and stored by the IoT devices, which are subject to attack. Since most IoT devices have limited resources, standard security algorithms such as AES, DES, and RSA hamper their ability to run properly. In this paper, a lightweight symmetric key cipher termed randomized butterfly architecture of fast Fourier transform for key (RBFK) cipher is proposed for resource-constrained IoT devices in the edge computing environment. The butterfly architecture is used in the key scheduling system to produce strong round keys for five rounds of the encryption method. The RBFK cipher has two key sizes: 64 and 128 bits, with a block size of 64 bits. The RBFK ciphers have a larger avalanche effect due to the butterfly architecture ensuring strong security. The proposed cipher satisfies the Shannon characteristics of confusion and diffusion. The memory usage and execution cycle of the RBFK cipher are assessed using the fair evaluation of the lightweight cryptographic systems (FELICS) tool. The proposed ciphers were also implemented using MATLAB 2021a to test key sensitivity by analyzing the histogram, correlation graph, and entropy of encrypted and decrypted images. Since the RBFK ciphers with minimal computational complexity provide better security than recently proposed competing ciphers, these are suitable for IoT devices in an edge computing environment.

Keywords Avalanche effects, Block ciphers, Butterfly architecture, Edge computing, FELICS, IoT, Lightweight cryptosystems, MATLAB

Introduction

In the age of Industry 4.0, most businesses such as industries, health care, government agencies, and agriculture, are concentrating on digital transformation and automation to enhance production efficiency through the utilization of Internet of Things (IoT) devices in edge computing infrastructure (Wang et al. 2013; Xiao et al. 2019). IoT devices have become a core part of edge computing infrastructure, such as smart cities, automatic drive systems, and smart traffic management systems, to execute basic functions such as actuating, monitoring, and detecting real-world objects (Li et al. 2016; Gao et al. 2021; Sachdev 2020; Rafique et al. 2020). For faster and in-time decision-making capability, IoT devices need high-performance connectivity and low-latency feedback from the core cloud network (Amin and Hossain 2021). In a gas pressure monitoring system, a late response from a remote cloud network to the pressure sensor would

*Correspondence:
Joarder Kamruzzaman
Joarder.Kamruzzaman@federation.edu.au
[1] Institute of Information and Communication Technology, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh
[2] Centre for Smart Analytics, Federation University Australia, Gippsland, VIC 3842, Australia

result in the switch failing to shut down in time to prevent the gas line from being damaged by overpressure. Constraints in cloud technology (Gao et al. 2021), such as lack of locality awareness, bandwidth availability and real-time capabilities, pave the way for a new era of edge computing capable of storing, and processing substantial amounts of data close to IoT devices (Xiao et al. 2019; Zhang et al. 2018). Edge computing offer advantages in terms of latency and bandwidth usage due to the computation being closer to the data generation sources without being transmitted to the core cloud network.

Figure 1 illustrates edge computing (Gao et al. 2021; Amin and Hossain 2021; Pan and McElhannon 2018) referring to computation that takes place at the network's edge, outside of the cloud. The edge computational infrastructure (Gao et al. 2021) is designed to enable storage and considerable processing power close to IoT devices, dramatically reducing latency and optimizing bandwidth usage. Processing directly on the data collected by sensors, the edge gateways can support real-time applications through services that are spectral efficient, location-aware, privacy-conscious, real-time, and incur minimal cost (Pan and McElhannon 2018). In the edge computing framework, edge servers (Xiao et al. 2019; Cao et al. 2021) are nodes that perform local data processing and storage before delivering actual data generated by IoT devices to the cloud for further processing, if needed. The communication between an edge device and an edge is facilitated by an edge gateway. The edge server separates the core cloud network from the rest of the network. In most cases, the edge can be just one hop away from the IoT devices that create the data. An edge IoT device, such as wireless sensors, appliances, or any other data-capturing device, is connected to the internet. Ethernet, Bluetooth, Wi-Fi, NFC, ZigBee, and other protocols can be utilized for data delivery from edge servers to actual IoT devices (Gao et al. 2021).

For resource-constrained devices such as embedded systems, radio frequency identification tags (RFID), and sensor networks, resources such as random access memory (RAM), read-only memory (ROM), computing power, and battery life are limited. The edge servers

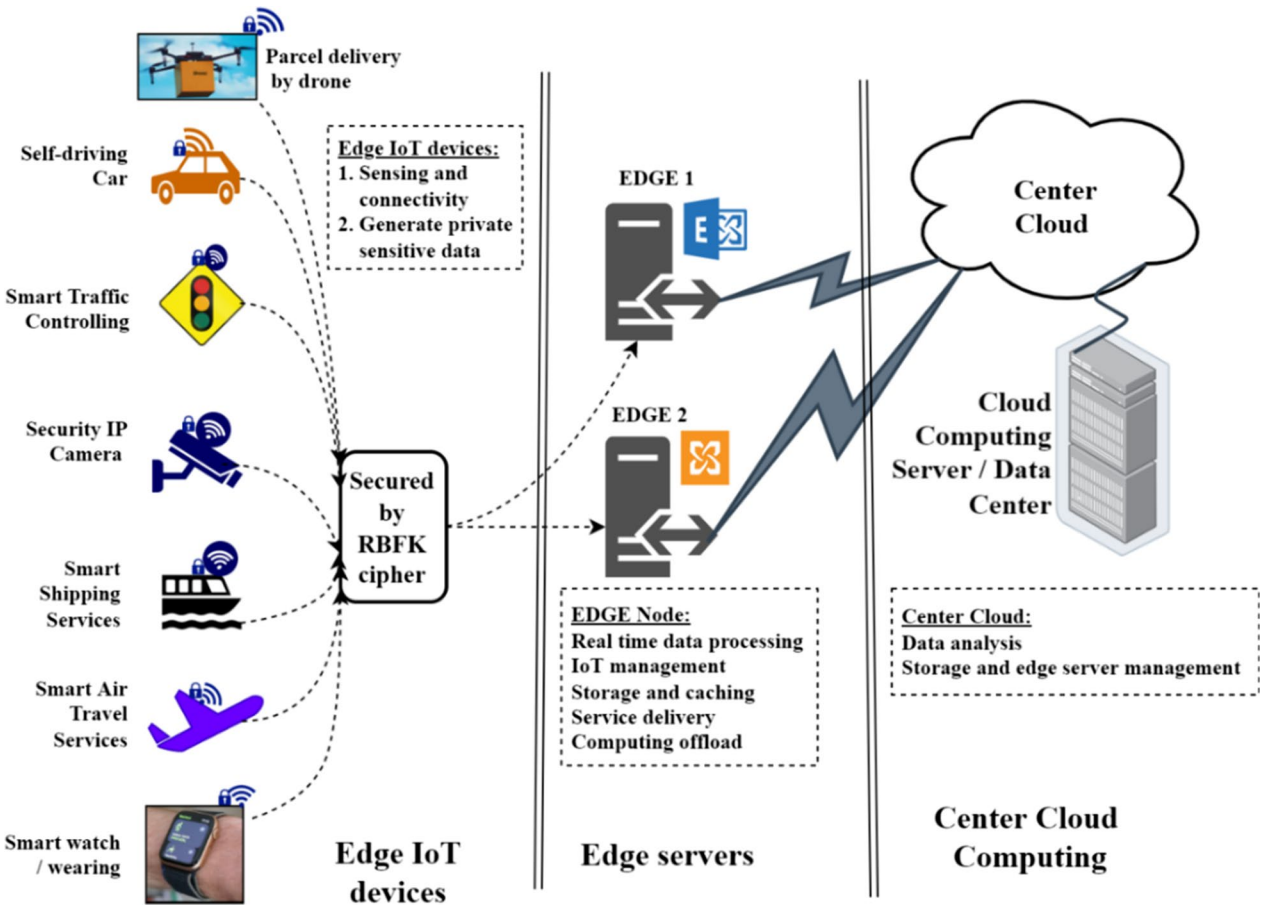


Fig. 1 Application of RBFK cipher for IoT devices in the edge framework

manage data processing in the edge framework, and IoT devices generate and store a significant volume of sensitive data that is subject to attack (Sachdev 2020). Figure 1 shows some examples of IoT devices and services, including traffic control systems, drone delivery, smart cities, smart agricultural fields, automatic drive system, embedded systems, wireless sensors, and smart healthcare devices (Narayanan et al. 2020). Furthermore, IoT devices lack sufficient graphical user interface (GUI), leaving users unaware of most of these devices' functionalities as well as their potential vulnerabilities.

Motivation

Physical attacks

As most IoT devices in the edge computing architecture are tiny and left unattended for long periods, they are vulnerable to physical attack (Gao et al. 2021; Amin and Hossain 2021) by intruders. As a consequence, the sensitive and secretive data produced by these devices must be secured by cryptographic protocols in a way that the installed protocol does not jeopardize the operational performance of the resource-constrained IoT devices.

Limited processing capability

Compared to a server, the computational power of an IoT device is low. As a result, an edge device is more exposed to potential attacks (Sachdev 2020). Many attacks that are ineffectual against desktop computers can pose major dangers to edge IoT devices as their protection system is much more vulnerable than that of desktop computers.

Performance degradation

It is highly expected for edge IoT devices to operate in real-time, i.e., incredibly quickly, to keep the circumstances safe from damage (Alwarafy et al. 2021), especially in critical applications like healthcare, industrial process, surveillance, and autonomous driving. Heavyweight ciphers such as Advanced Encryption Standard (AES) and Data Encryption Standard (DES) require significant resources to keep IoT devices working efficiently. As a result, while implementing cryptographic protocols, the performance degradation of edge IoT devices must be addressed.

Attack ignorance

Unlike servers and desktop computers, most IoT devices lack GUI, apart from the issue that they may have primitive LED displays (Gao et al. 2021; Amin and Hossain 2021). As a result, a client could have minimal information about a device's current condition, such as where it has been shut down or hacked. As a result, although an attack may be happening on an edge device, most users may be unable to detect it.

Security trade-off

It is a significant issue to find a balance between the lightweight nature and the high level of security provided by lightweight block ciphers (McKay et al. 2017a). Modern cryptographic schemes like AES (Stallings 2005) and Rivest–Shamir–Adleman (RSA) are strong enough, however, they slow down the performance of lightweight IoT devices. On the other hand, lightweight cryptographic algorithms are easier to break but ideal for IoT devices with limited resources (Rana et al. 2018; Usman et al. 2017). Hence, developing a strong cipher with low computational complexity is a pressing need, though challenging, to ensure the security of IoT devices in an edge computing environment.

To address the above-mentioned security challenges in IoT devices, various block ciphers such as AES, data encryption standard (DES) (Stallings 2005), PRESENT (Papapagiannopoulos 2016), SIT (Usman et al. 2017), Speck (Beaulieu et al. 2014), SIMON (Beaulieu et al. 2014), and others are commonly employed to assure security. Many of these cryptographic algorithms, such as DES, employ the Feistel Architecture, while others, such as AES, use the substitution-permutation-network (SPN) Architecture. Since the security of ciphers largely depends on the round keys, key scheduling in cryptographic algorithms should be performed securely to generate strong round keys with a higher avalanche effect.

Contributions

In this research, a randomized butterfly architecture of the Fast Fourier transform for key (RBFK) cipher is proposed for IoT devices in an edge computing environment to provide security and efficient use of limited resources. Based on the Feistel architecture, the RBFK cipher is a symmetric key as well as a lightweight block cipher. The modified butterfly architecture is used in the key scheduling system to produce strong round keys used by the encryption part of the proposed cipher. Due to the butterfly architecture, the RBFK ciphers have a larger avalanche effect, assuring strong security. The proposed cipher satisfies Shannon's confusion and diffusion characteristics. With a block size of 64 bits, the RBFK cipher offers two key sizes: 64 and 128 bits. Both variants of the proposed cipher provide a high level of security while being computationally simple. Since RBFK ciphers with minimum computational complexity offer superior security to previously proposed competing ciphers, they are suited for IoT devices in an edge computing context.

Organization

The remaining portions of the article are organized as described below: The next section, which comes after the introduction, is a review of previous works. Following that is a description of the RBFK cipher that has been proposed, as well as the key scheduling technique. The next sections, respectively, discuss the implementation specifics and the security analysis of the proposed solution. After that, some concluding remarks and some recommendations for the future are presented.

Related works

To ensure security, real-world applications of mobile IoT frameworks (Pan and McElhannon 2018) such as Mobile Ad-hoc Network (MANET), Vehicular Ad-hoc Network (VANET), and Flying Ad-hoc Network (FANET) use a variety of cryptographic protocols. However, most IoT-based infrastructures are unaware of well-defined and well-suited protocol usage for edge IoT devices with low processing power, RAM, ROM or program memory, and battery capacity, along with many other factors. Some modern block ciphers were studied and reported in this section, along with their performance, in resource-constrained devices in an edge computing framework.

The bulk of prior introduced encryption methods, such as AES and DES, are constructed on the SPN network (Preneel 1998) and the Feistel architecture, respectively. The primary goal of implementing these primitives is to increase the security of keys and ciphertext by ensuring a greater avalanche effect. The basic problem with these primitives is that they are computationally expensive, and thereby, degrade the performance of edge IoT devices when employed. As a result, producing typical avalanche effects using lightweight mathematical processes is a difficult task in cryptography.

The FFT (Ferreira et al. 2021) is a commonly used low-complexity implementation of the Discrete Fourier Transform (DFT) for signal processing in resource-limited devices. FFT computing is powered by the Butterfly architecture. The Cooley–Tukey algorithm (Ferreira et al. 2021) is one of the most widely used FFT techniques. The complex series of FFT is computed using this technique. This algorithm uses a divide-and-conquer strategy to partition the whole DFT issue into multiple potential smaller DFTs. The basic FFT is made up of radix–2 butterfly architectural units.

In Jha (2011), Jha evaluated the performance and security of current lightweight ciphers used in resource-constrained systems, such as HIGHT, TEA, KATAN, and KLEIN. The author employed the AtTiny45 microcontroller as a small computing device to measure performance parameters such as RAM, ROM, and power usage. In addition, to evaluate the avalanche effect, the confusion and diffusion properties of generated ciphertext were measured.

In 2018, Rana et al. (2018) proposed a cipher for resource-constrained systems that utilized fewer data and consumed less power than previously published ciphers. It was built on two core principles of genetic algorithms, namely, crossover and mutation. The authors contributed to the key generation scheme by replacing the matrix operation with non-linear bit scrambling. They also used FELICS to compare the execution time and memory use of their proposed technique. MATLAB was used to test the security strength of their cipher by encrypting several images.

Secure IoT (SIT) is a lightweight block cipher for IoT devices presented in Usman et al. (2017), which was designed based on the Feistel architecture and SPN network. They devised a cipher that combines key generation and encryption in one package. They offer only 64-bit key size with 64-bit block size. The encryption process consists of five rounds. A 64-bit cipher key is used to produce five distinct keys in the key generation section where a 64-bit input is partitioned into four blocks, each holding 16-bit data, after initial permutation. Each of the four blocks provides input to the F-function. A 4×4 matrix is used to translate the output of the F-function. In this scenario, the matrix introduces nonlinearity. The F-function consists of two P-Boxes; used in key scheduling and implementing linear transformation.

In Gao et al. (2021), Gao et al. proposed a lightweight block cipher for automatic drive systems to provide security. The cipher employs a key expansion mechanism to create five-round keys, as well as an encryption method to convert plaintext to ciphertext. To obtain confusion in encrypted text, the encryption procedure consists of five rounds. They evaluated the security of their proposed cipher by encrypting several images using histograms, correlation graphs, and image entropy. They do not, however, illustrate how to fit the cipher in an automatic drive system in terms of processing power and memory utilization. Some researchers (e.g., Volna et al. 2012 and Komal et al. 2015) investigated the use of artificial neural networks for cryptographic applications where the weights and the architecture of the network essentially represent the key. However, a neural network generates a black box model and is computationally expensive, and for any modern cipher of n-bit, the network needs to be trained for all possible 2^n samples which makes this approach impractical for IoT devices in edge computing framework.

Proposed RBFK cipher

The guiding principle of designing RBFK is to offer a lightweight symmetric key block cipher suitable for edge IoT devices. The RBFK cipher works in three phases: key expansion, encryption process, and decryption process. The key expansion, which generates round keys to encrypt plaintext using the encryption method, is the initial component of the cipher. With a 64-bit block size, the proposed RBFK offers two key size versions: RBFK-64 for the 64-bit key size and RBFK-128 for the 128-bit key size. The key expansion technique in the RBFK-64 variant generates five distinct keys while the RBFK-128 variant creates ten distinct keys.

As a result, the encryption procedure must be robust enough to resist the cipher from being broken by attackers. Typically, lightweight block ciphers require 5–20 rounds of similar operations to encrypt the plaintext into ciphertext with a higher avalanche effect. As we increase the number of rounds, the cipher becomes computationally expensive. In a number of studies (Gao et al. 2021; Usman et al. 2017), it is reported that a block cipher needs at least 5 rounds of iteration to get a good level of diffusion in the ciphertext. In several block ciphers, the traditional round is 10–20 to reach diffusion. In the proposed RBFK cipher, a round number of 5 is chosen. This is because the RBFK cipher is designed for securing edge IoT devices that lack resources like program memory and processing power. Even with 5 rounds of iteration, the RBFK cipher obtains enough diffusion level which results in an avalanche effect of more than 50% to guarantee high key sensitivity. Hence, this paper presents a lightweight block that ensures a higher avalanche effect while consuming low energy to well suit the edge IoT devices.

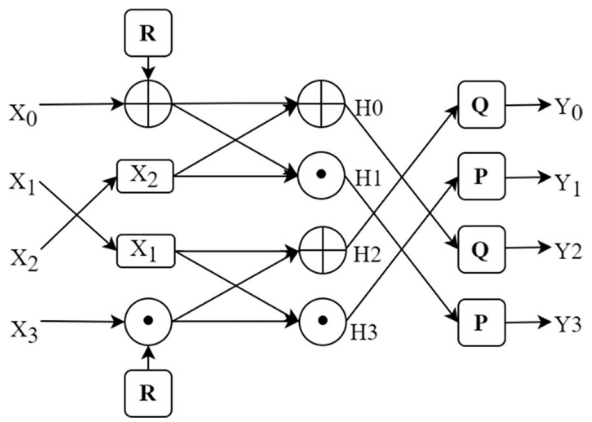


Fig. 2 Internal structure of RBFK function

RBFK function in key expansion

The Randomized Butterfly architecture of FFT for Key Scheduling (RBFK) function is derived from the butterfly structure of FFT. Figure 2 illustrates the structure of the RBFK function. The input layer X = [X0, X1, X2, X3]; the intermediate layer H = [H0, H1, H2, H3]; and the output layer Y = [Y0, Y1, Y2, Y3] are the three levels of this function. The input for this function is four 4-bit values, and the output is the same size as the input. To generate results, the proposed RBFK structure consists of XNOR as well as XOR, which are two popular bitwise operators for cryptographic algorithms, particularly for symmetric ciphers. The XNOR and XOR are invertible and thus enable the same operation for the encryption and decryption processes. As a result, no separate inverse operation needs to be designed for the decryption process. This will result in decreasing the space complexity of a cipher.

At first, input X_0 is XORed with a pseudo-random number R and also X_3 is XNORed with R which is generated by the following equations.

$$X = \sum_{i=0}^{4} X_i \tag{1}$$

$$R = (X + P)\ mod\ M;\ \ P = 1, 2, 3, 4 \tag{2}$$

where Xi = [X0, X1, X2, X3] and M is an integer number that ranges from 2 to 16. Also, P refers to the RBFK block number that is P = 1 for RBFK block1 and so on. The nonlinearity of the RBFK function is ensured by the pseudo-random number. To compute the result of RBFK function, we need to compute the middle layer Hj = [H_0, H_1, H_2, H_3] as the following equations.

$$H_0 = x_0 \oplus x_2 \oplus R$$
$$H_1 = (x_0 \oplus R) \odot x_2$$
$$H_2 = (x_3 \odot R) \oplus x_1$$
$$H_3 = x_1 \odot x_3 \odot R$$

According to the Eqs. (3)–(6), the output layer $Y_k = [Y_0, Y_1, Y_2, Y_3]$ is calculated by feeding inputs from the middle layer H_j into the P Table and Q Table.

$$\mathrm{Y0} = \mathrm{Q}(\mathrm{H}_2) \tag{3}$$

$$\mathrm{Y1} = \mathrm{P}(\mathrm{H}_3) \tag{4}$$

$$\mathrm{Y2} = \mathrm{Q}(\mathrm{H}_0) \tag{5}$$

$$\mathrm{Y3} = \mathrm{P}(\mathrm{H}_1) \tag{6}$$

Table 1 (a) P Table and (b) Q Table

Input key(ki)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
(a) P Table																
Generated key(ki)	3	F	E	0	5	4	B	C	D	A	9	6	7	8	2	1
(b) Q Table																
Generated key(ki)	9	E	5	6	A	2	3	C	F	0	4	D	7	B	1	8

Substitution box

In key scheduling, substitution boxes are used to address nonlinearity in generated round keys. Two unique S-boxes are designed in such a manner that Shannon diffusion and confusion characteristics are met. Table 1(a) and (b) show the transformations performed by P Table and the Q Table (Rana et al. 2021), respectively. The provided P and Q tables are two different permutation boxes that provide linear transformation, i.e., diffusion properties in ciphertext and round keys. These boxes have been selected based on their avalanche effect. For example, in the P table, when the input $(0)_{16}$ or $(0000)_2$ is replaced by $(3)_{16}$ $(0011)_2$, the output bits are changed by 50% compared to the input. Moreover, the key scheduling considered in this paper has a nonlinear transformation as it uses a random number, R in the RBFK block. Besides, the P and Q tables are designed to achieve a high avalanche effect in generated round keys.

Key expansion of RBFK-64 and RBFK-128

The keys that are used to accomplish encryption and decryption are the most basic components of a cipher. If the key used to create ciphertext is revealed, security is compromised. Therefore, the key should be as difficult to uncover as possible. The key sensitivity must be sufficiently high to protect the information against different types of attacks, such as chosen ciphertext only, chosen plaintext only, differential attacks, and so on. Even if the attacker guesses a key that is only one bit different from the original, decryption with that supposed key should produce encrypted text. The proposed ciphers used the RBFK function to generate keys with an avalanche effect of more than 50% to guarantee high key sensitivity. The proposed key generation technique for RBFK-64 is illustrated in Fig. 3.

A 64-bit cipher key is used as input in RBFK-64 to produce 5 distinct round keys for the encryption process. This 64-bit cipher key is split into 4-bit groups to produce 16 networks. The key generation mechanism employs four RBFK blocks, each of which operates on four 4-bit (0–15) values. So, as stated in (7), each of the RBFK blocks selects four numbers from a permutation of 16 4-bit blocks of input cipher key (K).

$$RBFK_i = ||_{j=1}^{4} K_{(j-1)+i} \tag{7}$$

The range (i) is 1 to 4 for the first four produced round keys shown in Fig. 3. As seen in Eq. (7), each RBFK block accepts the input of four segments. The P and Q tables, which are permutations, as described in Table 1 (a) and (b), have been used to generate the final

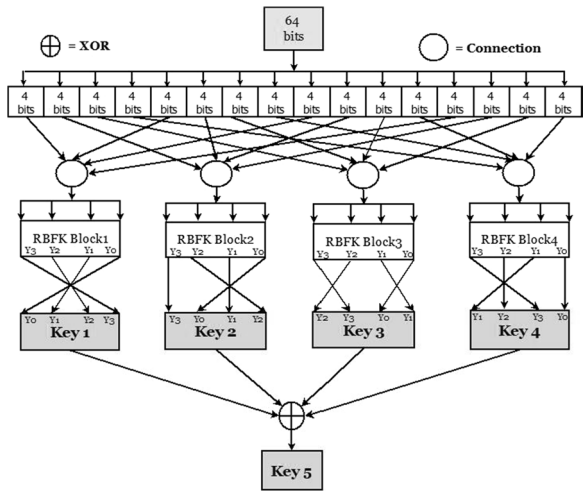


Fig. 3 RBFK based key scheduling for 64-bit cipher key

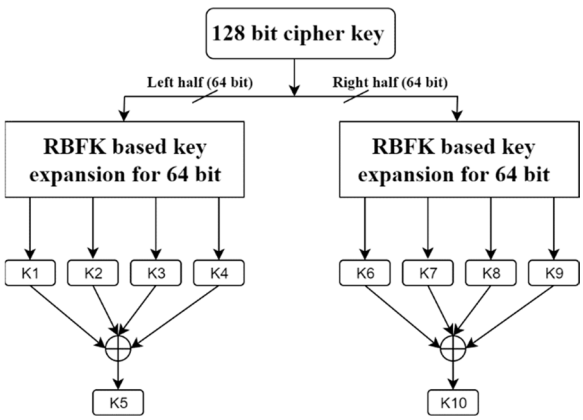


Fig. 4 RBFK based key scheduling for 128 bit cipher key

output (Y3, Y2, Y1, Y0) of RBFK blocks. Following that, K1, K2, K3, and K4 are produced by concatenating the various combinations of four 4-bit outputs of MBFK blocks.

$$K1 = Y0 \,\|\, Y1 \,\|\, Y2 \,\|\, Y3 \text{ received from RBFK block1}$$
$$K2 = Y3 \,\|\, Y0 \,\|\, Y1 \,\|\, Y2 \text{ received from RBFK block2}$$
$$K3 = Y2 \,\|\, Y3 \,\|\, Y0 \,\|\, Y1 \text{ received from RBFK block3}$$
$$K4 = Y1 \,\|\, Y2 \,\|\, Y3 \,\|\, Y0 \text{ received from RBFK block4}$$

The next step is to apply XOR operation on the first four generated round keys to produce a new unique 5th round key namely K5.

The RBFK-128 variant, on the other hand, uses a 128-bit cipher key as input and creates ten distinct round keys to complete the encryption process. The RBFK-128 variant employs two blocks of key expansions to generate ten round keys by separating two 64-bit keys from a 128-bit key to feed into the key expansion block of RBFK-64, as shown in Fig. 4.

Avalanche effect of RBFK based key expansion

The RBFK ciphers' key expansion technique yields round keys with a larger avalanche effect. We generated a large number of keys to test the avalanche effect of the RBFK key scheduling technique. In the best-case scenario, it assures an avalanche effect of up to 58.46%. However, the avalanche effect of RBFK-based key scheduling is more than 50% in most cases, meeting the Shannon confusion properties criteria. Table 2 displays 7 pairings of ciphertext for 7 different pairs of cipher keys, where each pair of keys differs only by a single bit. The plain text used in Table 2 is 0xabcd123487650135.

Table 2 Avalanche effect (AE) of key generation technique

SL	Main cipher keys (64 bits)	Cipher text (64 bits)	AE (%)
1	0x123456789abcdeb1	0x36f17d72c4030960	50.77
	0x123456789abcdeb2	0x5bea90f335ba4067	
2	0xaaaaaaaaaaaaaaaa	0x6dd51747962e525c	57.58
	0xaaaaaaaaaaaaaaab	0xee488abb17c2fde1	
3	0xaaaaaaaaaaaaaaaa	0x6dd51747962e525c	57.58
	0xaaaaaaabaaaaaaaa	0xee488abb17c2fde1	
4	0x555555555555555	0x6dd51747962e525c	58.46
	0x555555555555554	0xee488abb17c2fde1	
5	0x555555555555555	0x6dd51747962e525c	58.46
	0x555555545555555	0xee488abb17c2fde1	
6	0x1000000000000000	0x7eaf6aeccbbd511e	50.78
	0x1000000000000001	0x0409cc3c79611f90	
7	0x1000000000000000	0x7eaf6aeccbbd511e	56.92
	0x1000000000000008	0x31237913d05e3682	

Encryption process

The encryption process is a modified Feistel structure with a G-function, which is based on two genetic algorithm operators: crossover and mutation. Figures 5 and 6 depict the flow of operations for a single round of encryption for RBFK-64

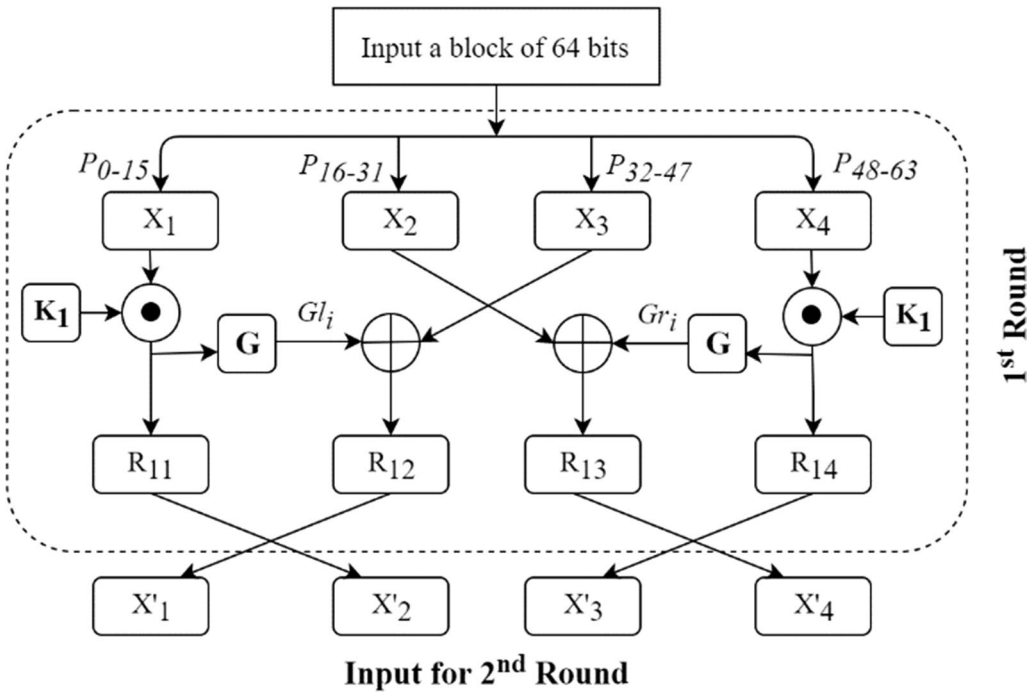


Fig. 5 The first round of encryption and input for 2nd round (RBFK-64 variant)

and RBFK-128, respectively. The encryption algorithm for both RBFK-64 and RBFK-128 consists of 5 rounds.

The RBFK cipher is tested for differential cryptanalysis, linear cryptanalysis, impossible differential cryptanalysis, and zero-correlation cryptanalysis. The testing is performed for a number of images. Generally, the round function of a cipher must include linear and non-linear transformations to provide diffusion and confusion. Therefore, in the design of the RBFK round function, two substitution boxes provide nonlinear transformation, and the scan pattern provides linear transformation. The RBFK is designed with a sufficiently complex round function for its application in edge IoT devices with limited computing power and memory. At a time, the encryption procedure accepts a 64-bit plaintext as input for both variants. In RBFK-64, the 1st round employs the 1st key (K1) produced using the key scheduling process of RBFK cipher (64 bit), followed by K2, K3, K4, and K5 for the second, third, fourth, and fifth rounds, in that order. In RBFK-128, the first round uses the key K1 and K6 generated using the key scheduling process of RBFK cipher (128 bit), followed by K2 and K7 for the second round, K3 and K8 for the third round, K4 and K9 for the fourth round, and K5 and K10 for the fifth round.

The 64-bit message is split into four 16-bit parts (X1, X2, X3, X4), as illustrated in Fig. 5. Swapping, XOR, and XNOR operations are undertaken among the divided blocks according to the Feistel structure to maximize the avalanche effect in encrypted text. The round key and the leftmost (X1) and rightmost (X4) blocks are each subjected to an XNOR operation. The output (R_{ij}) of the XNOR operation is then fed as an input to the G-function, resulting in the output Gl_i or Gr_i, where i indicates the round number. The third block (X3) and the left G-function's output Gl_i are XORed again, as are the second block (X2) and the right G-function's output Gr_i. Then, except for the last round, a swapping operation is done among the four blocks so that the four input segments of the following round, X1′, X2′, X3′, X4′, are R_{12}, R_{11}, R_{14}, and R_{13}, as illustrated in Fig. 5 and explained in Algorithm 1. The last round of encryption must avoid the swapping operation among the four blocks since symmetric key ciphers normally utilize the same algorithm for both encryption and decryption with reversal usage of round keys.

Finally, all four blocks are merged to form a 64-bit ciphertext. The decryption procedure is the inverse of the encryption procedure. With the same design as the encryption procedure, the final key is utilized first this time. In RBFK-128, all-around operations are the same as in RBFK-64 except that the two keys are used per block as shown in Fig. 6. The round keys K1 and K6 are used for 1st round of the encryption process of the RBFK-128 variant.

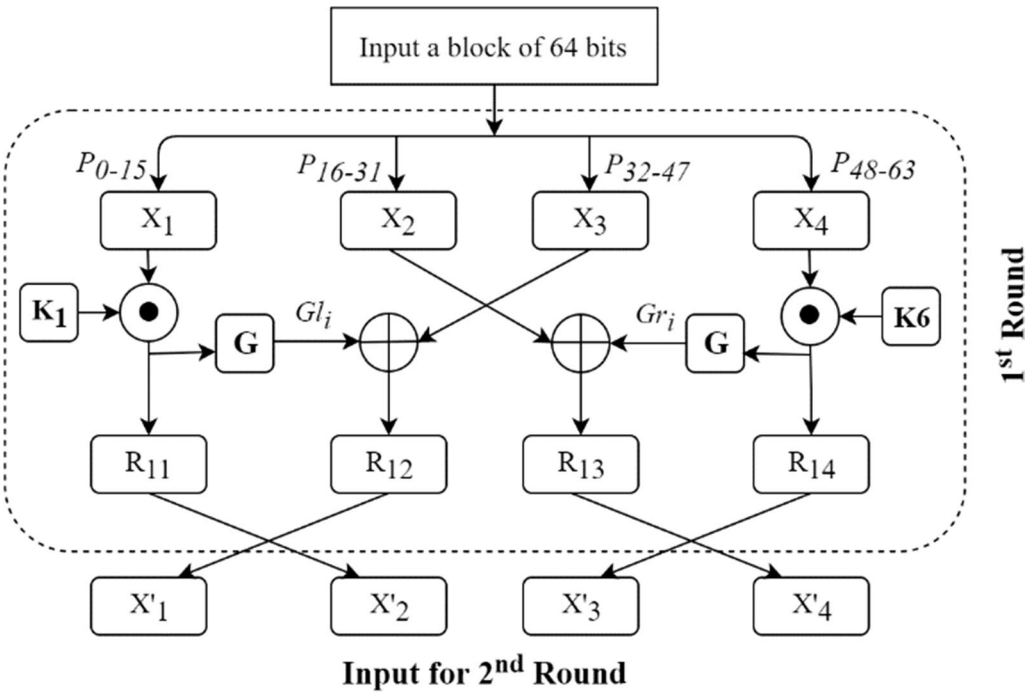


Fig. 6 The first round of encryption and input for 2nd round (RBFK-128 variant)

Algorithm 1: Encryption (Roundkeys[**K**], Plaintext[**P**])

Initialization: $X_1 \leftarrow P_{0-15}, X_2 \leftarrow P_{16-31}, X_3 \leftarrow P_{32-47}, X_4 \leftarrow P_{48-63}$;
$K_1, K_2, K_2, K_3, K_4, K_5$ *are the five round keys.*
Output : *A block of cipher text* $[C]$ *of 64 bits*

for $i \leftarrow 1$ **to** 5 **do**
  /* Start of the round i */
  $R_{i,4} = X_4 \odot K_i$;
  for $j \leftarrow 1$ **to** 3 **do**
    if $j == 1$ **then**
      $R_{i,j} \leftarrow X_j \odot K_i$;
    end
    if $j == 2$ **then**
      $Gl_i =$ **G-function** $(R_{i,j-1})$;
      $R_{i,j} \leftarrow Gl_i \oplus X_{j+1}$;
    else
      $Gr_i =$ **G-function** $(R_{i,j+1})$;
      $R_{i,j} \leftarrow Gr_i \oplus X_{j-1}$;
    end
  end
  /* End of inner for loop. */
  /* Swapping $R_{i,j}$ before starting the next round except 5^{th} round. */
  if $i \neq 5$ **then**
    $X_1 \leftarrow R_{i,2}$;
    $X_2 \leftarrow R_{i,1}$;
    $X_3 \leftarrow R_{i,4}$;
    $X_4 \leftarrow R_{i,3}$;
  end
end
/* End of outer for loop. */
/* Finally, Concatenated four 16 bit segments after completing five round to generate 64 bit ciphertext. */
Return Ciphertext , $C = R_{5,1} \nparallel R_{5,2} \nparallel R_{5,3} \nparallel R_{5,4}$;

G-function

The core of the G-function is designed based on the idea of a genetic algorithm operator i.e., mutation and a scan pattern as shown in Fig. 7. Algorithm 2 represents the pseudo-code for a graphical view of the G-function. This function accepts 16 bits as input and performed transposition by a scan pattern as shown in Fig. 8. After that, the output of the scan pattern is divided into two eight-bit pieces evenly (Higher 8 bits, Lower 8 bits). Figure 9 shows how the middle four bits of 8-bit data are changed by a substitution box. The S-Box accepts four bits as input and outputs four bits in such a way that the MSB of the input chooses the row and the remaining three bits determine the column. For example, if the input is

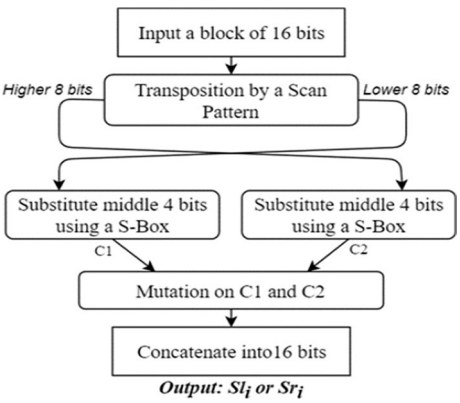


Fig. 7 The G-function for each round of RBFK cipher with 16 bit input and 16 bit output

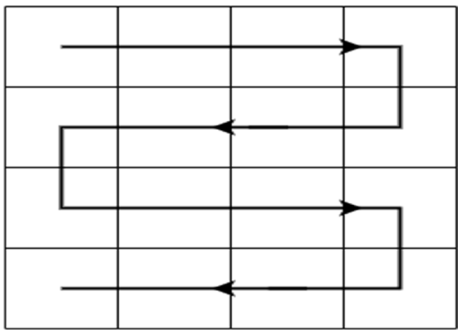


Fig. 8 Scan pattern with 16-bit input and 16-bit output

	000	001	010	011	100	101	110	111
0	A	E	D	C	B	F	9	8
1	7	6	0	4	3	2	1	5

Fig. 9 S-Box with 4 bits input and 4 bits output

$(0101)_2$, the output will be F_{16}, which is $(1111)_2$ as shown in Fig. 9. Because of its invertible characteristics, the same S-Box can be utilized for encryption and decryption. A coin flip mutation operation is applied on both S-Box outputs. After that, the 16-bit output is produced.

Scan-pattern

Figure 8 depicts a scan pattern in which the line comes from the left of the first row to the right, then from the right to the left of the second row. A similar process occurs in the following two rows. Each cell in the scan pattern represents a single bit, 0 or 1. As a result, the scan pattern can accept 16 bits as input and provide 16 bits as output.

For example, by following the lines of the scan pattern, the output would be 1011, 0011, 0010, 1010 for the supplied input 1011, 1100, 0010, 0100 as in Table 3. The reverse order properties of the second and fourth rows help to reduce the plaintext pattern to an unknown pattern.

Table 3 Sample input and output of scan pattern

	1st row	2nd row	3rd row	4th row
Input	1011	1100	0010	0101
Output	1011	0011	0010	1010

Algorithm 2: G-function(16 bits input [**R**])

Initialization: $X \leftarrow R_{0-15}$;
Output : *A block of 16 bit.*

1 $[X_1, X_2] \leftarrow$ **Scan-Pattern** (X);
/* X_1 receives higher byte of 16 bit output generated by Scan-Pattern and X_2 relates to lower byte. */
2 $M_1 \leftarrow$ middle 4 bits of X_2;
3 $M_2 \leftarrow$ middle 4 bits of X_1;
4 $C_1 \leftarrow$ **S-Box** (M_1);
5 $C_2 \leftarrow$ **S-Box**(M_2);
/* Apply **coin flip** Mutation to C_1 and C_2. */
6 $C_1 \leftarrow C_1 \oplus (81)_{16}$; /* Flipped MSB and LSB bits of C_1. */
7 $C_2 \leftarrow C_2 \oplus (81)_{16}$; /* Flipped MSB and LSB bits of C_2. */
/* Finally, Concatenated C_1 and C_2 to generate 16 bit output in statement 8. */
8 **Return** $C_1 \nparallel C_2$;

Table 4 Comparison of execution cycles taken by the different ciphers on AVR architecture

CIPHER	Device	Block size (bit)	Key size (bit)	Code size (Byte)	RAM (Byte)	Cycles (key generation)	Cycles (encryption)	Cycles (decryption)
AES (Stallings 2005)	AVR	128	128	23090	720	3274	5423	5388
DES (Kumar et al. 2016)	AVR	64	64	2580	2248	2218	7046	2580
HIGHT (Kim et al. 2019)	AVR	64	128	13476	288	1412	3376	3401
LEA (Jha 2011)	AVR	128	128	3700	432	4290	3723	3784
PRESENT (Papapagiannopoulos 2016)	AVR	64	80	1738	274	2570	7447	7422
Simon (Beaulieu et al. 2014)	AVR	64	96	1370	188	2991	1980	1925
Speck(Beaulieu et al. 2014)	AVR	64	96	2552	124	1509	1179	1411
SIT(Usman et al. 2017)	AVR	64	64	826	22	2130	876	851
G-cipher (Rana et al. 2018)	AVR	64	64	1228	34	1630	792	789
RBFK-64	AVR	64	64	1228	34	1483	792	789
RBFK-128	AVR	64	128	1566	34	2966	792	789

The proposed RBFK-64 and RBFK-128 bit variations were tested on AVR architecture-based devices with the Fair Evaluation of Lightweight Cryptographic Systems (FELICS) tool to generate execution cycles for key generation, encryption, and decryption, as well as to measure the memory usage. Both 64-bit and 128-bit variations of RBFK require fewer execution cycles than the existing ciphers that are presented later in Table 4. Because of requiring few execution cycles, the RBFK ciphers will fit well into edge IoT devices. A security evaluation was also carried out to validate the security strength of RBFK ciphers against various attacks. The following section presents a detailed evaluation.

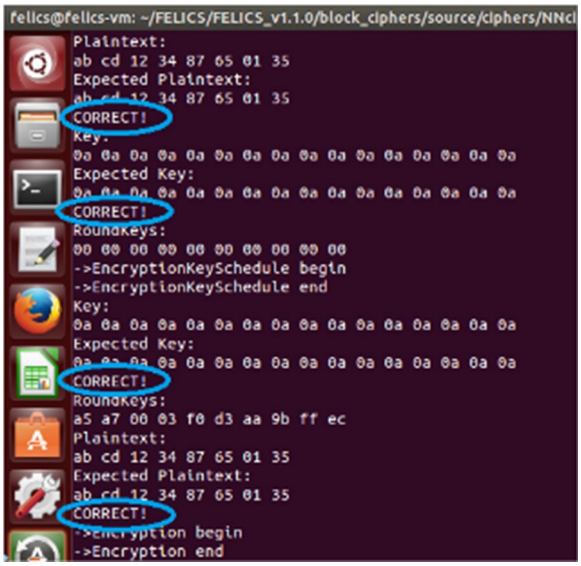


Fig. 10 The FELICS implementation to verify the RBFK ciphers

Results and discussion

Initially, the suggested technique was implemented in C, a widely used structural language. We also measured memory use as well as execution cycles on Linux Ubuntu using the benchmark FELICS. The FELICS utility is available for free download and use. The MATLAB 2021a program was used to assess the security of generated keys for the proposed RBFK techniques. In this work, we additionally investigated how much memory and clock cycles the RBFK cipher used to produce keys, ciphertext, and regenerate the original message.

FELICS implementation

For the FELICS benchmark (Dinu 2015), there are Command Line Interfaces (CLI) for implementing, testing and assessing a newly designed block cipher as shown in Fig. 10. It allows a cryptographer to test every round of encryption and decryption whether it works correctly or not. FELICS also have documentation support for the implementation of a newly designed cipher. It has various scenarios for different hardware architectures (e.g., AVR, PC), compiler options, and format of the generated report (binary, Excel, CSV, etc.). A cryptographer can test and evaluate his/her new cipher under his suitable choices.

RBFK algorithm compared with

The proposed RBFK lightweight block cipher was compared to AES (Federal Information Processing Standards Publication 197, 2001), DES, HIGHT (Kim et al. 2019), LEA (Jha 2011), PRESENT (Papapagiannopoulos 2016), Simon (Beaulieu et al. 2014), Speck (Beaulieu et al. 2014),

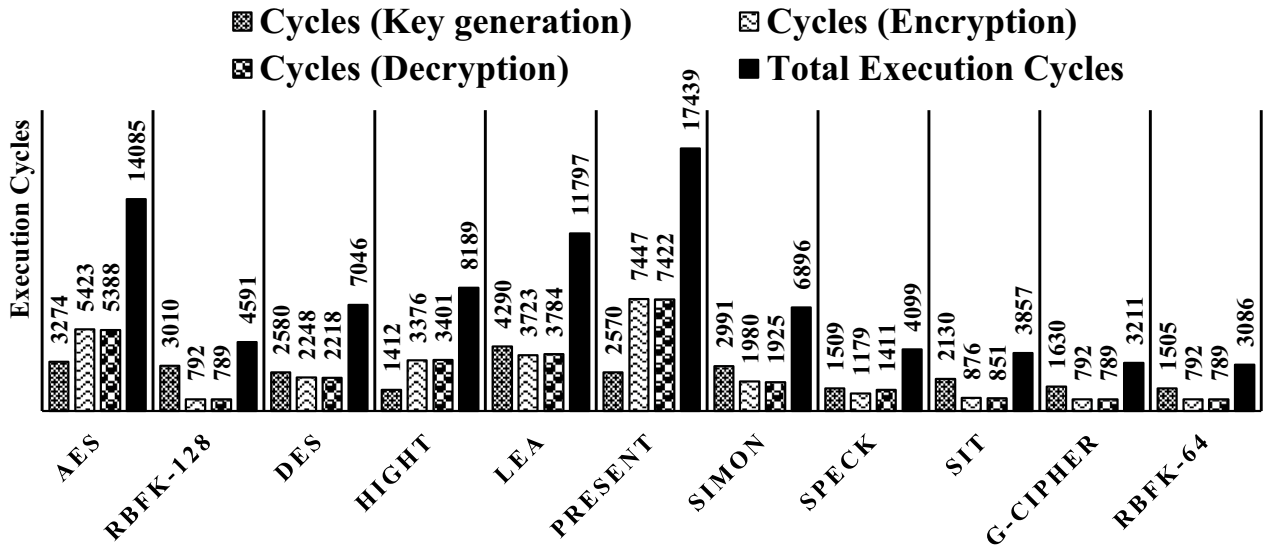


Fig. 11 Execution cycle of one block data for different ciphers

and SIT (Usman et al. 2017). Though AES is directly not a lightweight cipher in the NIST proposal (McKay et al. 2017b), it is reported to be used in edge devices (Gao et al. 2021; Usman et al. 2017). On the other hand, the DES considered here is a lightweight version of DES obtained by discarding the initial and final permutation and using one S-box instead of 8 S-boxes.

For the reported ciphers in the literature as well as the proposed RBFK ciphers, FELICS was used to extract clock cycles for key generation, encryption, and decryption. We also evaluate RAM, ROM, and the size of the code itself. The performance of notable ciphers for the AVR architecture is compared in Table 4. For the same key size ciphers, the RBFK ciphers need the fewest total execution cycles of all the algorithms reviewed. RBFK ciphers are memory efficient as these ciphers require a fewer amount of RAM to run. Since at present, the ROM of resource-limited devices is large enough in size, the cipher with a slightly large code size disturbs the performance less (Rana et al. 2019). However, a device's physical memory e.g., RAM is used mostly while running the algorithms and so the usage of RAM for a cipher should be as little as possible.

With the RBFK ciphers, Fig. 11 shows bar chart comparisons among several reported ciphers. The number of clock cycles required to produce keys, ciphertext from plaintext, plaintext from ciphertext, and total execution cycles are shown for each cipher. The figure clearly shows that the RBFK-64 encryption system uses significantly fewer total clock cycles than other ciphers, particularly SIT (Usman et al. 2017) and G-cipher (Rana et al. 2018) (e.g., 3086 cycles vs 3857 and 3211 cycles, respectively) which are proposed as lightweight cryptographic algorithms for use in IoT devices. As a result, the proposed RBFK cipher uses less power than other reported cryptographic algorithms. Similarly for RBFK-128, the total execution cycle is significantly fewer than the other 128-bit key size versions, namely AES (Stallings 2005) and LEA (Jha 2011) (e.g., 4591 cycles vs 14,085 and 11,797 cycles, respectively).

As an example, following Banerjee et al. 2015, execution cycles are measured for an AVR architecture-based microcontroller, particularly Atmega128 for a single block of plaintext, as shown in Table 5. According to the datasheet's absolute maximum rating (AMR), the Atmel Atmega128's maximum working voltage is usually 5 V, the maximum current is 40 mA, and the frequency is 16 MHz (Papapagiannopoulos 2016). AVR microcontrollers generally feature a two-stage pipeline, with an AVR machine cycle and a clock cycle having a one-to-one direct relationship. To calculate the time it takes for one machine cycle, the inverse of the clock frequency is taken. For a 16 MHz clock frequency, the clock period per machine cycle is 1/16 µs, or 0.06 µs.

Table 5 presents the comparative results of the time complexity of different ciphers in terms of key generation, encryption, and decryption. When compared to HIGHT, the proposed RBFK cipher (64-bit) has lower complexity. Although the key generation complexity of HIGHT is better than that of RBFK, the proposed RBFK outperforms HIGHT in terms of encryption, decryption, and total time complexity. So, the RBFK cipher is more efficient than HIGHT. On the other hand, the time complexity of RBFK is comparable to that of SIT and G-cipher for the case of encryption and decryption. However, RBFK outperforms SIT and G-cipher in terms

Table 5 Time complexity of proposed RBFK ciphers, as well as others, for a single block of plaintext on an AVR architecture-based microcontroller, notably Atmega128

Ciphers	Block size (bit)	Key generation (µs)	Encryption (µs)	Decryption (µs)	Total execution (µs)
AES (Stallings 2005)	128	204.62	338.94	336.75	880.31
DES (Kumar et al. 2016)	64	161.25	140.50	138.62	440.37
HIGHT (Kim et al. 2019)	64	88.25	211.00	212.56	511.81
LEA (Jha 2011)	128	268.12	232.69	236.50	737.31
PRESENT (Papapagiannopoulos 2016)	64	160.62	465.44	463.87	1089.9
Simon (Beaulieu et al. 2014)	64	186.94	123.75	120.31	431.00
Speck (Beaulieu et al. 2014)	64	94.31	73.68	88.187	256.19
SIT (Usman et al. 2017)	64	133.12	54.75	53.187	241.06
G-Cipher (Rana et al. 2018)	64	101.87	49.50	49.312	200.69
RBFK-64	64	94.06	49.50	49.312	192.87
RBFK-128	64	188.12	49.50	49.312	286.97

of key generation and the total overall complexity. Therefore, RBFK is more efficient than both SIT and G-cipher. Similarly, Table 5 indicates that RBFK has better overall time complexity than all other ciphers considered.

Security analysis

The proposed ciphers are also demonstrated in MATLAB 2021a to assess the key sensitivity of various encrypted images using the histogram, correlation graph, and entropy. The desired score of the histogram, correlation graph, entropy, number of changing pixel rate (NPCR), and unified averaged changed intensity (UACI) implies that the underlying scored ciphers are highly secure for edge IoT devices in the edge computing framework. The security strength of the ciphers is evaluated using the following criteria:

(i) Key sensitivity
(ii) Histogram and correlation of the image
(iii) Change of image entropy
(iv) Percentage score of NPCR and UACI
(v) Power consumption

Linear and differential cryptanalysis

Differential cryptanalysis is commonly applied to block ciphers and cryptographic hash functions. This type of attack in block cipher is a collection of techniques for detecting variations across a network of transformations, detecting non-random behaviour in the cipher, and exploiting such characteristics to extract the actual secret key. A pair of plaintexts with a specified difference is fed into an encryption process in differential cryptanalysis. If the difference between ciphertexts is identical to the difference between plaintexts, then this difference is utilized to retrieve the secret key. The goal of the linear analysis is to discover relationships between input bits, output bits, and key bits. The RBFK cipher is tested for differential cryptanalysis, linear cryptanalysis, impossible differential cryptanalysis, and zero-correlation cryptanalysis. The testing is performed for a number of images. Generally, the round function of a cipher must include linear and non-linear transformations to provide diffusion and confusion. Therefore, in the design of the RBFK round function, two substitution boxes provide nonlinear transformation, and the scan pattern provides linear transformation. The RBFK is designed with a sufficiently complex round function for its application in edge IoT devices with limited computing power and memory. In this aspect, the proposed cipher has a significant avalanche effect as well as non-linear operations.

Key sensitivity analysis

To provide a graphical representation of key sensitivities, many images are encrypted and then deciphered using the actual keys. The images are also decrypted using an erroneous key that differs from the original key by only one bit. In this way, the avalanche effect of a cipher can be assessed. Even if the hackers predict keys that are quite similar to the original keys, the cipher content remains unknown. Experiments were done with four images of child, bridge, baboon, and football, and their encryption and decryption outcomes are shown in Fig. 12. The results demonstrate that the pixels of encrypted images have a high level of randomness for both the proposed ciphers. The encrypted images of the MSBK algorithm can only be deciphered with the original key, and even a single-bit variation from the actual key produces a random image as the decryption result.

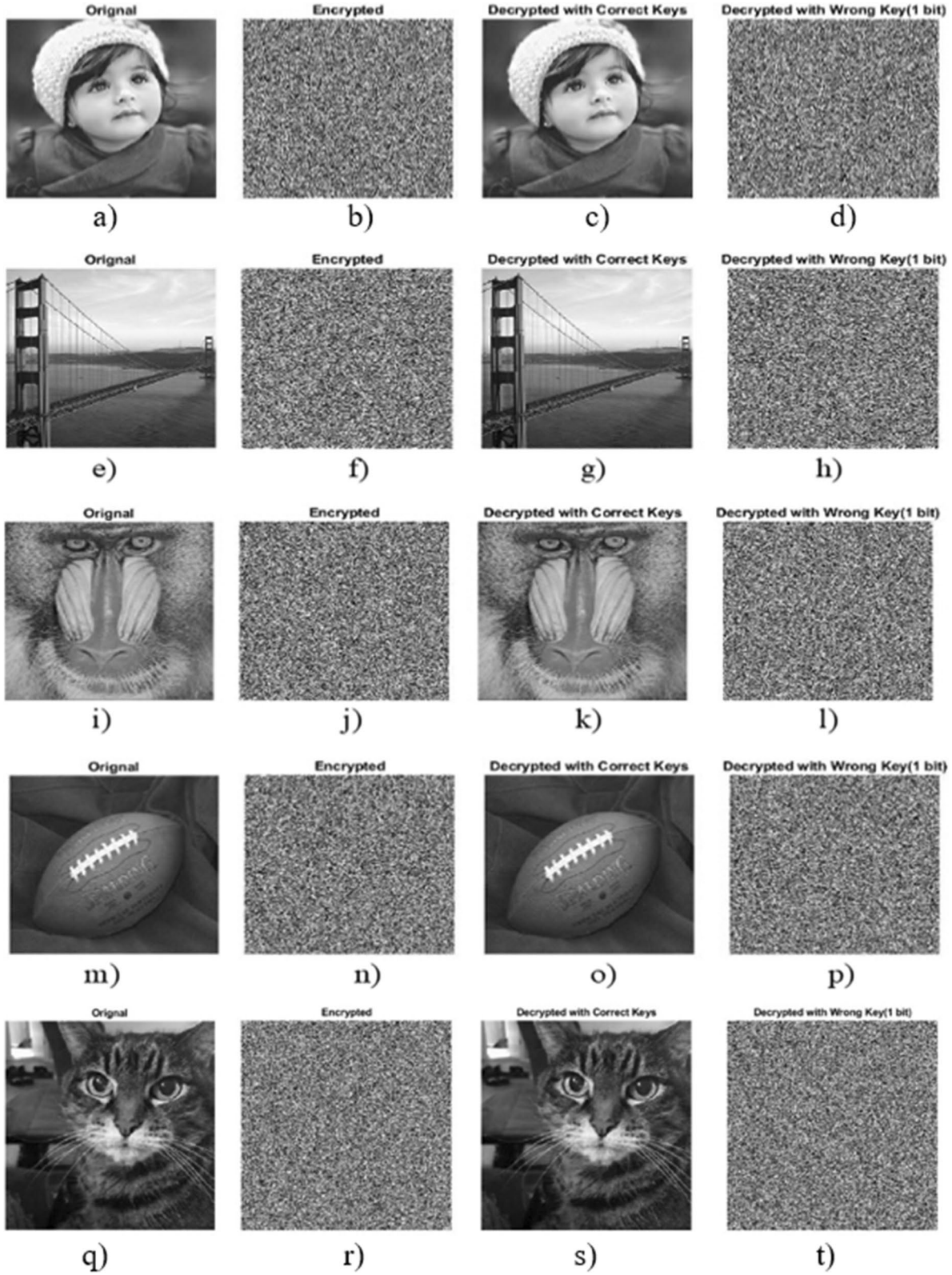
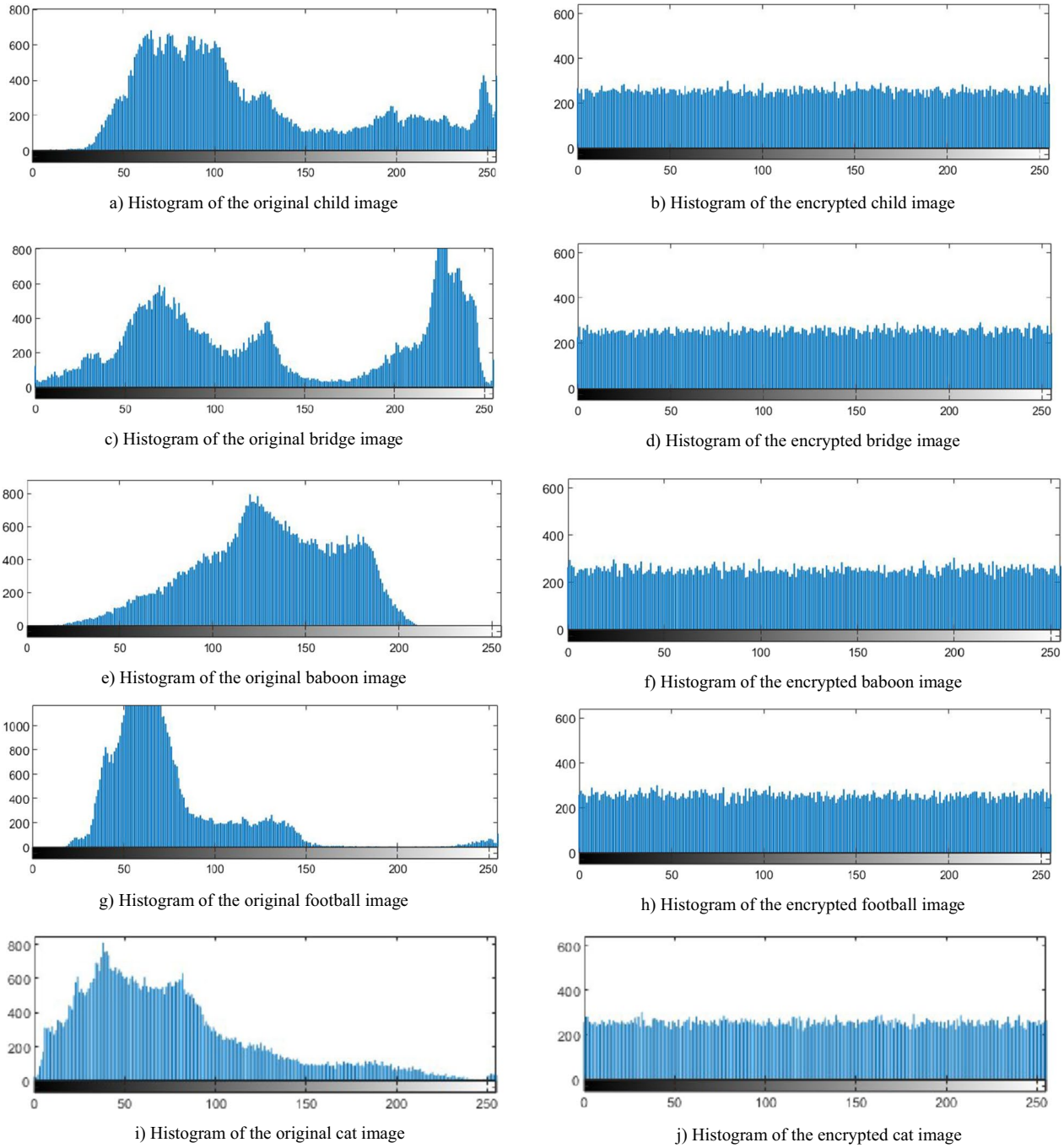


Fig. 12 Statistical analysis of key sensitivity by decrypting the same image with two keys: one is the actual key and another is a wrong key that differs by only a single bit from the actual key

Histogram analysis

The histogram of image pixels is also another effective way for testing the confusion and randomness of ciphertext (Baagyere et al. 2020) in what seems like a block cipher. This approach has been used to test both ciphers for several images in this section. In Fig. 13, the vertical line indicates the number of pixels available in an image, and the horizontal axis refers to image intensity. The figure presents the histograms of the four original images: (a) child, (c) bridge, (e) baboon, (g) football, (i) cat, and their respective encrypted images in (b), (d), (f), (h), and (j). The histogram of the encrypted images reveals a uniform distribution of pixels which ensures the strength of the encrypted image. As a result, without the correct



a) Histogram of the original child image

b) Histogram of the encrypted child image

c) Histogram of the original bridge image

d) Histogram of the encrypted bridge image

e) Histogram of the original baboon image

f) Histogram of the encrypted baboon image

g) Histogram of the original football image

h) Histogram of the encrypted football image

i) Histogram of the original cat image

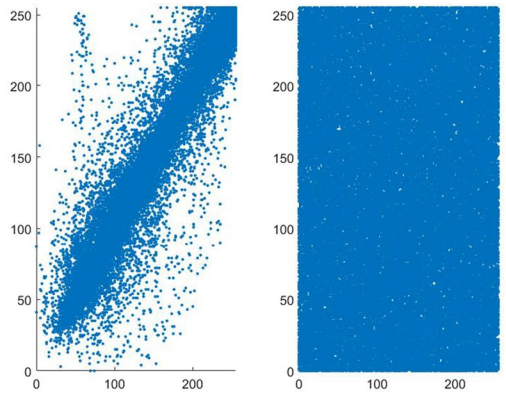
j) Histogram of the encrypted cat image

Fig. 13 Histograms of all original and encrypted images reported in Fig. 11

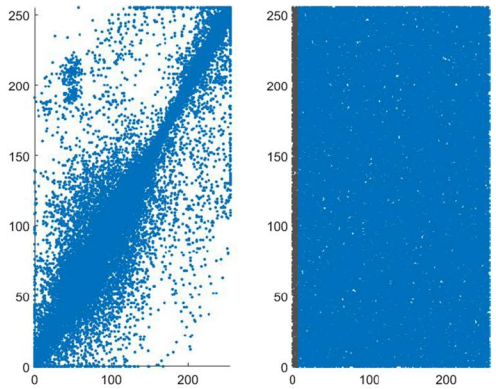
keys, statistical attacks must be ineffective in predicting the original image data from an encrypted image.

Correlation graph

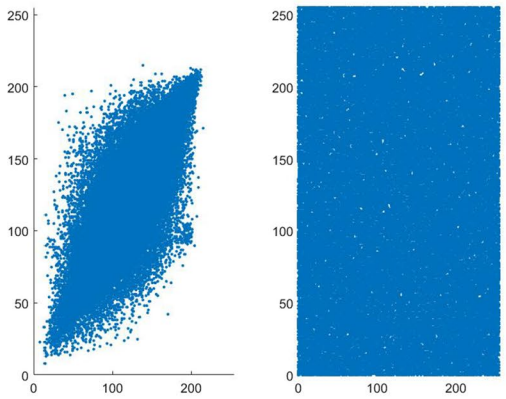
In the field of cryptography, a higher degree of visual randomness in the correlation graph of cipher images implies that the underlying cipher used for encryption has a higher level of security to protect the information. The visual randomization of pictures encrypted by the proposed RBFK-64 and RBFK-128 ciphers is represented in this section. Typically the original picture correlation plot indicates a linear relationship with a greater positive correlated value. Figure 14(a,b), (c,d), (e,f), (g,h), and (i,j) show correlation plots of original (left parts)



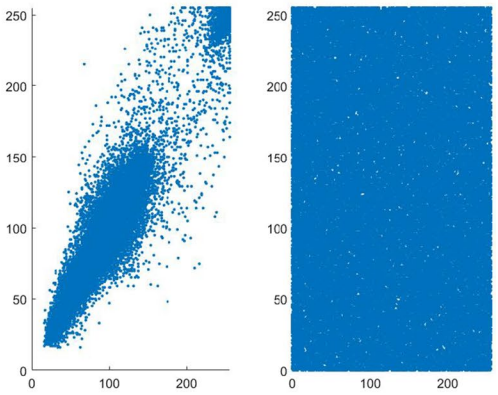
Correlations for (a) original and (b) encrypted 'Child' image



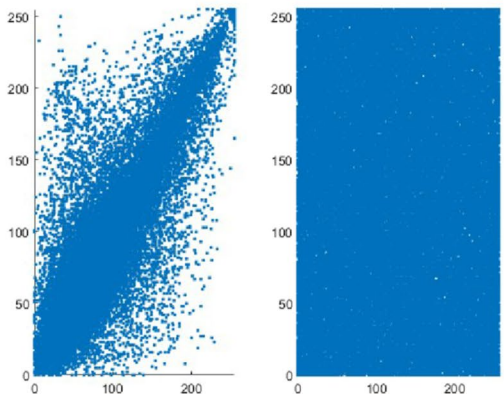
Correlations for (c) original and (d) encrypted 'Bridge' image



Correlations for (e) original and (f) encrypted 'Baboon' image



Correlations for (g) original and (h) encrypted 'Football' image



Correlations for (i) original and (j) encrypted 'Cat' image

Fig. 14 Correlation graphs of original, and encrypted images

and encrypted (right parts) photos of the child, bridge, baboon, and football, respectively. The correlation graph of encrypted pictures, on the other hand, displays considerable unpredictability, i.e., negative scores for both variants of RBFK ciphers. As a result, the negative correlation scores of encrypted pictures imply that the recommended RBFK ciphers have a higher level of security. As a consequence, without the necessary keys, all statistical attacks attempting to predict the plain picture from the encryption image would fail. In Table 6, we also report the correlation score of various pictures for proposed cryptosystems.

Table 6 NPCR, UACI, and correlation coefficients of RBFK ciphers

Images			RBFK-128	RBFK-64
Child	Correlation coefficient	Plain image	0.9750	0.9750
		Encrypted image	− 0.0008	− 0.0067
	Percentage of NPCR		99.6109	99.6109
	Percentage of UACI		16.7364	16.6811
	Entropy	Plain image	7.5597	7.5597
		Encrypted image	7.9975	7.9975
Bridge	Correlation coefficient	Plain image	0.9626	0.9626
		Encrypted image	− 0.0044	− 0.0046
	Percentage of NPCR		99.6292	99.5712
	Percentage of UACI		14.9606	15.0790
	Entropy	Plain image	7.5856	7.5856
		Encrypted image	7.9967	7.9977
Baboon	Correlation coefficient	Plain image	0.8198	0.8198
		Encrypted image	0.0015	− 0.0031
	Percentage of NPCR		99.6292	99.6307
	Percentage of UACI		13.2555	13.2777
	Entropy	Plain image	7.2316	7.2316
		Encrypted image	7.9967	7.9969
Football	Correlation coefficient	Plain image	0.9616	0.9616
		Encrypted image	0.0044	− 0.0004
	Percentage of NPCR		99.6521	99.6078
	Percentage of UACI		26.2973	25.9850
	Entropy	Plain image	6.6839	6.6839
		Encrypted image	7.9973	7.9973
Cat	Correlation coefficient	Plain image	0.8967	0.8967
		Encrypted image	0.0015	− 0.0021
	Percentage of NPCR		99.6075	99.5651
	Percentage of UACI		26.5610	26.4219
	Entropy	Plain image	7.4538	7.4538
		Encrypted image	7.9969	7.9977

For the RBFK-128 and RBFK-64 ciphers, we gathered NPCR, UACI, correlation coefficients, and image entropy in Table 6. These are very effective statistical theories to test a cipher's security strength by encrypting images. A percentage value of NPCR very close to 100% would indicate a strong cipher in encrypting an image. For pairs of adjacent pixels, the correlation coefficients of various encrypted images are always close to 0 or negative as the proposed cipher diminishes the statistical features of the plaintext. The entropy of multiple encrypted grayscale images is nearly 8, further indicating that our ciphers offer a higher level of security. As demonstrated in Table 6, the RBFK-128 and RBFK-64 ciphers provide the optimum security in terms of UACI, NPCR, and image entropy in the majority of cases.

Power consumption

To determine the overall power utilized by an algorithm on a certain device, we must first determine the algorithm's execution cycle. One can calculate the energy consumption of an algorithm on a certain device as in Eq. (8):

$$Power\ E = I * Vcc * T * N \tag{8}$$

where Vcc is the operating voltage and I (in Ampere) is the operating current used over T seconds. N is the needed number of execution cycles and corresponds to the clock period.

The highest operational voltage of the Atmel (Atmega88/168) is generally 6 V, according to the dataset's (http://www.farnell.com/datasheets/1807017.pdf) absolute maximum rating (AMR). The maximum current is 200 mA, and it runs at a frequency of 20 MHz. Figure 15 shows the energy usage of various existing ciphers as well as the proposed RBFK methods. The figure demonstrates that RBFK-64 ciphers consume the least power at only 19.29 mW. On the other hand, G-cipher and SIT have power consumptions of 20.07 and 24.11 mW, respectively. Hence, RBFK-64 is a better power-efficient system than others ciphers and this makes it suitable for IoT devices, specially those deployed in inaccessible areas where battery replacement is impossible. In addition, RBFK-128 is a much more power-efficient cipher than the other 128-bit key size variants of cipher investigated in this study.

Conclusion

This work introduces the RBFK cipher, a novel lightweight symmetric key cipher for edge IoT devices in the edge computing environment. The suggested method

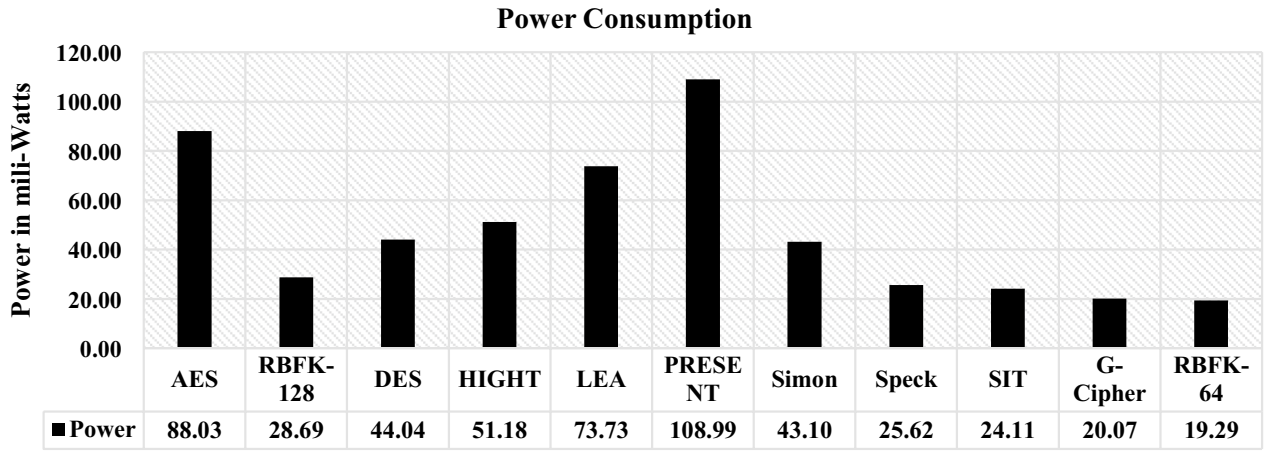


Fig. 15 Power consumption comparison of ciphers

employs butterfly architecture in its key scheduling process to produce secured round keys. The proposed ciphers use the RBFK function to generate keys with an avalanche effect of more than 50% to guarantee high key sensitivity. Besides, this new cipher provides randomness as it uses a random number R which is a function of the input. Furthermore, in the proposed design, each bit of ciphertext is dependent on a significant number of bits of the plaintext. In comparison to the ciphers reported in recent literature, the proposed ciphers have fewer key execution cycles and use less power. The RBFK cipher demonstrated strong encryption of images as measured by a number of analyses including NPCL, UACI, correlation coefficients and histogram of ciphers. Furthermore, key sensitivity results show that cipher images cannot be deciphered without the RBFK cipher's real keys. Results indicate that the proposed cipher consumes less power than existing algorithms including G-cipher, SIT, AES, Speck, PRESENT, and HIGHT. Hence, RBFK ciphers have the potential to be used as a cryptographic algorithm for edge IoT devices. The cipher works for 64-bit and 128-bit key sizes, with a block size of 64-bit.

Future research will focus on theoretical cryptoanalysis to formally assess the strength of the RBFK cipher. In addition, hardware implementation of the proposed cipher and testing on miniature IoT devices continues to be a significant future research interest.

Acknowledgements
Institute of Information and Communication Technology, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh.

Author contributions
The first author developed the initial idea, coded the algorithm, ran the experiments and prepared the initial draft of the paper. The second author assisted in formulating the fully developed idea, advised on experimentation and reviewed the manuscript. The last author advised on some aspects of the algorithm and experimentation, and suggested improvement of the manuscript. All authors read and approved the final manuscript.

Funding
This research has received no funding.

Availability of data and materials
Experimental data, source code and related materials will be provided on request.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 25 October 2022 Accepted: 26 December 2022
Published online: 01 February 2023

References

Alwarafy A, Al-Thelaya KA, Abdallah M, Schneider J, Hamdi M (2021) A survey on security and privacy issues in edge-computing-assisted internet of things. IEEE Internet Things J 8(6):4004–4022. https://doi.org/10.1109/JIOT.2020.3015432

Amin SU, Hossain MS (2021) Edge intelligence and internet of things in healthcare: a survey. IEEE Access 9:45–59. https://doi.org/10.1109/ACCESS.2020.3045115

Baagyere EY, Agbedemnab PA-N, Qin Z, Daabo MI, Qin Z (2020) A Multi-layered data encryption and decryption scheme based on genetic algorithm and residual numbers. IEEE Access 8:100438–100447. https://doi.org/10.1109/ACCESS.2020.2997838

Banerjee U, Ho L, Koppula S (2015) Power-based side-channel attack for AES key extraction on the Atmega328 microcontroller. Massachusetts Institute of Technology, Cambridge

Beaulieu R, Shors D, Smith J, Treatman-Clark S, Weeks B, Wingers L (2014) The Simon and Speck block ciphers on AVR 8-bit microcontrollers. National Security Agency

Cao K, Hu S, Shi Y, Colombo AW, Karnouskos S, Li X (2021) A survey on edge and edge-cloud computing assisted cyber-physical systems. IEEE Trans Ind Inf 17(11):7806–7819. https://doi.org/10.1109/TII.2021.3073066

Dinu D, Biryukov A, Großschädl J, Khovratovich D, Corre YL, Perrin L (2015) FELICS—fair evaluation of lightweight cryptographic systems. University of Luxembourg

Federal Information Processing Standards Publication 197 (2001) Announcing the advanced encryption standard (AES), NIST

Ferreira G, Paim G, Rocha LM, Santana GM, Neuenfeld RH, Costa EA, Bampi S (xxxx) Low-power fast Fourier transform hardware architecture

combining a split-radix butterfly and efficient adder compressors. https://doi.org/10.1049/cdt2.12015
Gao R, Li S, Gao Y et al (2021) A lightweight cryptographic algorithm for the transmission of images from road environments in self-driving. Cybersecurity 4:3. https://doi.org/10.1186/s42400-020-00066-2
http://www.farnell.com/datasheets/1807017.pdf
Jha VK (2011) Cryptanalysis of lightweight block ciphers. Master's Thesis, Aalto University School of Science Degree Programme of Computer Science and Engineering
Kim B, Cho J, Choi B, Park J, Seohindawi H (2019) Compact implementations Of HIGHT block cipher on IoT platforms. Secur Commun Netw 2019:5323578
Komal T, Ashutosh R, Roshan R, Nalawade SM (2015) Encryption and decryption using artificial neural network. Int Adv Res J Sci Eng Technol 2(4):81–83
Kumar SA, Vealey T, Srivastava H (2016) Security in internet of things: challenges, solutions and future directions. In: 2016 49th Hawaii international conference on system sciences (HICSS). IEEE, pp 5772–5781
Li S, Tryfonas T, Li H (2016) The internet of things: a security point of view. Internet Res 26(2):337–359
Mckay KA, Bassham L, Turan MS, Mouha N (2017a) Report on lightweight cryptography. National Institute of Standards and Technology, USA
McKay K, Bassham L, Sonmez Turan M, Mouha N (2017b) Report on lightweight cryptography, NIST interagency/internal report (NISTIR). National Institute of Standards and Technology, Gaithersburg, MD. https://doi.org/10.6028/NIST.IR.8114
Narayanan A et al (2020) Key advances in pervasive edge computing for industrial internet of things in 5G and beyond. IEEE Access 8:206734–206754. https://doi.org/10.1109/ACCESS.2020.3037717
Pan J, McElhannon J (2018) Future edge cloud and edge computing for internet of things applications. IEEE Internet Things J 5(1):439–449. https://doi.org/10.1109/JIOT.2017.2767608
Papapagiannopoulos K (2016) High throughput in slices: the case of PRESENT, PRINCE and KATAN64 ciphers. Department of Digital Security, Radboud University Nijmegen, Nijmegen
Preneel B (1998) Understanding cryptography, a textbook for students and practitioners. Springer, Heidelberg. https://doi.org/10.1007/978-3-642-04101-3 (**ISBN 978-3-642-04100-6 E-ISBN 978-3-642-04101-3**)
Rafique W, Qi L, Yaqoob I, Imran M, Rasool RU, Dou W (2020) Complementing iot services through software defined networking and edge computing: a comprehensive survey. IEEE Commun Surv Tutor 22(3):1761–1804. https://doi.org/10.1109/COMST.2020.2997475
Rana S, Hossain S, Shoun HI, Kashem DMA (2018) An effective lightweight cryptographic algorithm to secure resource-constrained devices. Int J Adv Comput Sci Appl (IJACSA). https://doi.org/10.14569/IJACSA.2018.091137
Rana S, Mondal MRH, Parvez AHMS (2021) A new key generation technique based on neural networks for lightweight block ciphers. Int J Adv Comput Sci Appl (IJACSA). https://doi.org/10.14569/IJACSA.2021.0120623
Rana S, Wadud AH, Azgar A, Kashem DMA (2019) A survey paper of lightweight block ciphers based on their different design architectures and performance metrics. Int J Comput Eng Inf Technol 11(6):112–118
Sachdev R (2020) Towards security and privacy for edge AI in IoT/IoE based digital marketing environments. In: 2020 fifth international conference on fog and mobile edge computing (FMEC), 2020, pp 341–346.https://doi.org/10.1109/FMEC49853.2020.9144755
Stallings W (2005) Cryptography and network security principles and practices, 4th edn. Prentice Hall, Hoboken
Usman M, Ahmed I, Aslam MI, Khanand S, Shah UA (2017) SIT: a lightweight encryption algorithm for secure internet of things. Iqra University, Defence View and Department of Electronic Engineering (IJACSA). Int J Adv Comput Sci Appl 8(1)
Volna E, Kotyrba M, Kocian V, Janosek M (2012) Cryptography based on neural network. In: Proceedings 26th European conference on modelling and simulation ©ECMS Klaus G. Troitzsch, Michael Möhring
Wang S, Zhang Z, Ye Z, Wang X, Lin X, Chen S (2013) Application of environmental internet of things on water quality management of urban scenic river. Int J Sustain Dev World Ecol 20(3):216–222
Xiao Y, Jia Y, Liu C, Cheng X, Yu J, Lv W (2019) Edge computing security: state-of-the-art and challenges. Proc IEEE PP(99):1–24. https://doi.org/10.1109/JPROC.2019.2918437
Zhang J, Chen B, Zhao Y, Cheng X, Hu F (2018) Data security and privacy-preserving in edge computing paradigm: survey and open issues. IEEE Access 6:18209–18237. https://doi.org/10.1109/ACCESS.2018.2820162

Publisher's Note