SN Computer Science (2024) 5:684
https://doi.org/10.1007/s42979-024-03026-6

ORIGINAL RESEARCH

RLCA-Robust Lightweight Cryptographic Paradigm for Secure Data Transfer Between IoT Devices

Shreyas Srinath[1] · **G. S. Nagaraja**[1]

Received: 10 May 2024 / Accepted: 3 June 2024

Abstract

IoT has grown in popularity and acceptance as a result of its numerous uses across different industries. They gather information from the live environment and some send it over networks. These devices may range in size from one cubic millimetre to a credit card size. When deploying in real-time, there are numerous obstacles to overcome, including those related to processing power, wireless capability, interoperability, and security. When billions of smart devices make a connected network in a diverse set of platforms, especially when shifting from the server world to sensors, it gives birth to various unprecedented challenges for their owners or users. These challenges constitute safety, durability, & privacy, interoperability, and technologies. In this paper, the need for secure data communication for low processing power devices is presented and a new lightweight block cipher cryptographic technique called RLCA (Robust Lightweight Cryptographic Algorithm) uses a combination of linear and non-linear mathematical techniques. Choosing the PRESENT algorithm as a benchmark, a comparative analysis based on the time taken to encrypt and decrypt for different data sizes is carried out. The experimental results have shown an improvement of 97% in execution time as compared with the standard PRESENT algorithm.

Keywords Energy-data security · Efficient cryptography · Feistel structure IoT security · Secure communication

Introduction

The Internet of Things (IoT) has now become a leading research period due to its applications in different area seg. smart healthcare, smart vehicles and transportation, smart manufacturing, smart infrastructure (smart buildings, smart water management, smart energy grids), and smart agriculture [11, 13]. The factors that are making significant contributions in making it challenging for data security in IoT are: The IoT devices interact directly with the other systems in the network so that confidential information can be collected or regulate physical environment variables. IoT devices are an easy target for attackers and therefore, are bound to be attacked [10]. Such devices are also easily available making them soft targets for attackers [9].

The combination of the above factors makes cyber security a challenging issue in IoT devices which requires consideration of confidentiality of sensitive information, data ethicality, protection, user authentication, authorization of information, accessibility, privacy, regulatory measures, and the need for frequent updates of systems [7, 8]. Therefore, cryptography plays a vital role in ensuring confidentiality, data validity and data verification& authorization so that the data remains unaffected while transferring before creation and after accessing it through IoT devices [12]. It could also be an answer for protecting the data which is either stored or transferred over the network. However, due to the resource-constrained nature of IoT devices, the traditional methods which were applicable for PC-based cryptography are not effective or applicable as they quest for large amounts of resources to operate. A lighter type of these solutions is lightweight cryptography which can solve these problems for protecting the communication in IoT devices which are limited in resources [15].

Cryptographic algorithms like Blowfish, RSA, DES, TRIPLE DES, AES etc.,that are in use today were developed to satisfy the demands of the desktop computing era [3]. But the IoT Devices which have low memory and processing

✉ Shreyas Srinath
shreyassrinath94@gmail.com

G. S. Nagaraja
nagarajags@rvce.edu.in

[1] Department of Computer Science, R.V College of Engineering, Bengaluru 560059, India

Published online: 28 June 2024

power, these devices require cryptographic solutions that are not only reliable for protecting data but also designed to function well under strict limitations. In the digital age, data protection is still of utmost importance, hence it is critical to build cryptographic algorithms that maintains the equilibrium between security and resource efficiency [16].

Major constraints IoT devices have are Restricted memory (registers, RAM, ROM) with limited computing efficiency as most of it works on low-power ARM-based processor segments, will have less physical area to implement the assembly and devices will have low battery power or no battery due to its size constraints. In these constraints, if conventional cryptography standards are applied to IoT devices their performance may not be acceptable [6, 14].

In this proposed work a new RLCA (Robust Lightweight Cryptographic Algorithm) is developed using a combination of linear and non-linear mathematical techniques.

RLCA has been enhanced for 64-bit key and block sizes. Because of its lightweight architecture and ability to balance security and efficiency, RLCA is a good choice for applications with constrained computational resources.

The Feistel cipher structure, a tried& tested cryptographic method renowned for its efficiency and simplicity, is adopted in the RLCA algorithm. RLCA uses a sequence of rounds in this structure, where each round consists of applying basic mathematical operations to the plaintext block. These operations consist of bit rotations and addition modulo bitwise XOR.

The approach taken here is to have the least amount of computing overhead possible while still having strong encryption capabilities. By carefully choosing cryptographic primitives and optimizing algorithmic parameters, this equilibrium is reached.

Literature Review

The necessity to safeguard data in cloud computing and Internet of Things (IoT) contexts is driving a rapid evolution in the development of cryptographic algorithms. The fundamental ideas of the Feistel structure, which emphasizes iterative rounds of substitution and permutation to generate durable encryption, are frequently the source of inspiration for these algorithms. The goal in the context of cloud and IoT is to develop algorithms that provide robust security guarantees and can function well in contexts with limited resources. A review of some of the feistal structure-based and Cloud -IoT integrated cryptographic designs is presented below.

A key architecture in cryptography is the Feistel structure, which divides data in half and undergoes a function flurry on each half as shown in Fig. 1. With a key's play, the structure shuffles bits round by round to create a strong cipher that gets stronger every iteration.

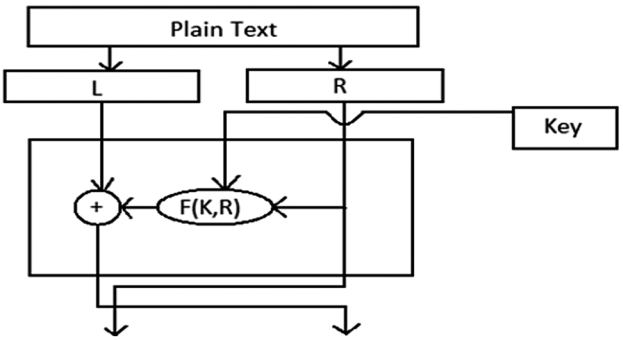

Fig. 1 Typical Feistal structure block

Using protocols like MQTT or CoAP for communication, cloud-based IoT data transmission algorithms route sensor readings through effective and secure data schemes, managing massive data streams, and guaranteeing dependable information dissemination as shown in Fig. 2.

Feistal Structure-Based Algorithms

An Innovative Effective Lightweight Homomorphic Cryptography Method for Data Protection in cloud computing by Thabit et al. [5]

This is a lightweight cryptography algorithm that works in two phases. In the first phase, the complexity of encryption is increased by combining Feistal and substitution or permutation methods using the following logical operations: shifting, XOR, swapping and XNOR. A secret key of 128-bit is generated using the above operations. In the second phase, the encryption is made stronger by implementing multiplicative homomorphic encryption. The multiplicative homomorphic property is based on the law of exponent as follows:

$$(\mathbf{a})^{\mathbf{n}}(\mathbf{b})^{\mathbf{n}} = (\mathbf{ab})^{\mathbf{n}}$$

In cryptography, this property indicates that the multiplication of two cipher texts is equivalent to the multiplication of two plaintexts which are then encrypted with the same secret key. The method presented in [5] uses 7 rounds for encryption.

The encryption is performed in two layers: Layer 1 and Layer 2.Encryption layer 1 involves the following steps: key expansion, encryption and decryption. In the key expansion step, the key is divided into 7 sub-keys. In the encryption block step, there are 7 rounds and the output of one round is sent as an input to the next round. The reduced number of rounds is chosen to reduce energy consumption. The decryption step consists of the decryption

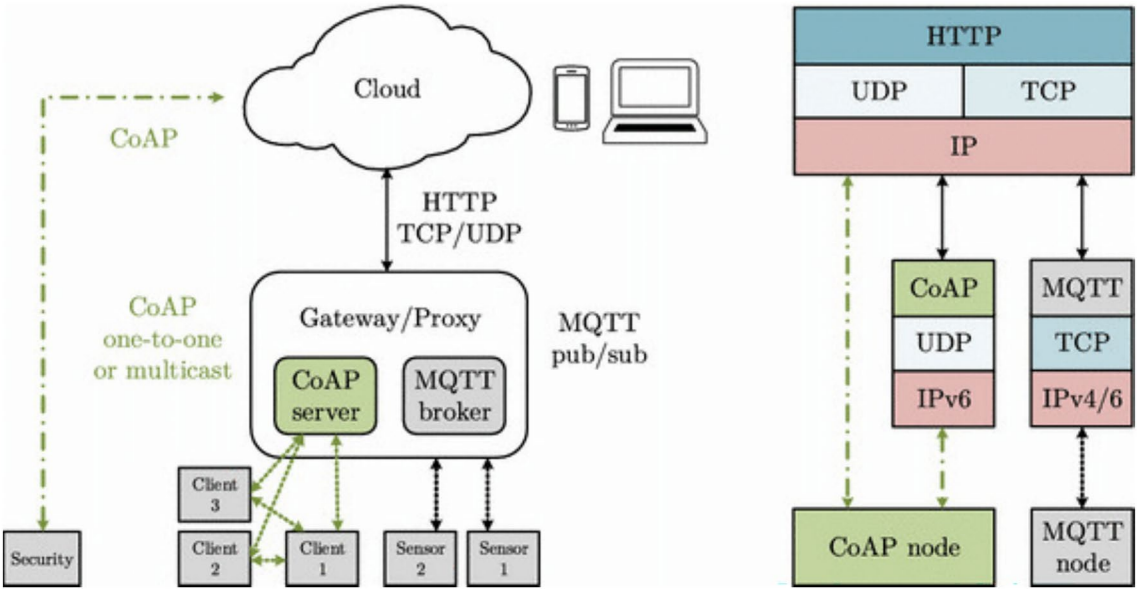

Fig. 2 Cloud -IoT integrated environment for data security

of ciphertext. The 128-bit Ci block is sub-divided into four blocks which use mixing of XOR and Sub operations.

The complete process has two layers, both layers at each of the sender and receiver end. At the sender's end, the encryption of plaintext takes place where plaintext undergoes an encryption process in layer 1, then ciphertext is converted to decimal form and then a multiplicative homomorphic property is used where ciphertexts (generated using secret key) are multiplied. The next step is uploading encrypted data on the cloud to be directed to the receiver. At the receiver's side, the encoded data is downloaded and converted to a binary system, then the cipher text undergoes the multiplicative homomorphic property of the RSA algorithm. The next step is the decryption of ciphertext using a decryption algorithm. The output is converted to a 128-bit binary system. In this way, the sensitive data is decrypted to the plaintext at the receiver's end.

A private cloud Omnet C++ version 6 is set up on the system with the following configuration: 16 GB RAM, 2.5GHz, Intel core I7 processor. The implementation was performed on MATLAB and written in Dev. C++ language. The two images with different sizesare tested on the algorithm and are evaluated on the following parameters: throughput, time and correlation. This algorithm presented in [5] has a minimum encryption/decryption time of 1.66 ms as compared to LED, SEA, HIGHT, RC6 and NLCA algorithms. The integrity, confidentiality and authenticity are maintained by the proposed algorithm. All algorithms have used 128-bit key except SEA and RC6 which used 96-bit and 256-bitkeys respectively.

A novel lightweight cryptographic algorithm for improving data protection in cloud computing by Fursan et al. [1]

This was developed based on Feistal and substitution permutation architecture for improved security in cloud computing. It uses 128-bit key for the encryption of data on the cloud. It is a 128-bit block cipher with 5 rounds for maximizing the energy efficiency of the algorithm. The procedure consists of three stages: key creation, encryption and decryption. During the key generation process, the 128-bit key is divided into two 64-bit segments. Each of these two sections are additionally split into 4-bit segments.

During an encryption process, the plaintext is subdivided into 4 parts each of which will be 32 bits. The XNOR operation of the first set of plaintexts with the first key K1 is sent to the function. Similarly, the second set is performed XNOR with key K1. The third set of plaintext is XORed with the output of the first plaintext with K1. Similarly, the fourth part is XORed with the output of the second part of plaintext with key K1. Then the following operations are performed to generate a ciphertext: transformation, swapping function, F function, left shifting and XOR and XNOR operations. The decryption process uses ciphertext split into four parts and each part undergoes each of the same operations as used in encryption. Decryption is just the opposite of the encryption process.

Authors have claimed that the proposed NLCA algorithm can work efficiently against the following types of threats: Differential and linear cryptanalysis, weak-key attacks, related-key attacks and symmetric properties.

The PRESENT algorithm This is considered the most efficient algorithm in terms of both software and hardware. It is an ISO/IEC (29192-2P:2012) approved algorithm. It works on a Substitution-Permutation based network which uses a 64-bit block. This block is utilized on two key alternatives: 128-bit and 80-bit keys with the GE requirements of 1886 and 1570 respectively. Approx. 1000 GE (encryption only) is the minimum requisite for GE for the PRESENT cipher, where it takes 2520–3010 GE to offer a satisfactory level of protection. PRESENT employs S-boxes (substitution layer –only one S-box in exchange for 8 S-boxes) of 4 bits where as it provides large cycles in the permutation layer.[18]

Cloud—IoT Based Algorithms

Investigation of Applied Application of Lightweight Cryptographic Algorithm ASCON: The lightweight cryptography algorithm proposed by Jefferey et al. [4] was tested at the Airforce Academy of the United States. It was observed that the algorithm provides high-quality security and authenticity with low memory requirements. The test was performed with the MQTT (Message Queuing Telemetry Transport) protocol which is a set of rules for queuing of the messages. The experimental setup was done using a weather messaging system and real-world MQTT and message broker. The weather system sends data providing weather information in real-time. These messages were sent to the server of MQTT and are stored in it as plaintext along with the timestamp linked to the particular message. These messages are then encrypted and broadcast by the MQTT server. The Raspberry Pi extracts the encrypted messages among the broadcasted data and then performs decryption on them. These decrypted messages are then sent to the e-link way and display the plaintext messages. It is also applied in location tracking systems. The location tracking system is a wearable smart device which provides the information regarding location of the person wearing it. The information includes the proximity, availability, tracking location in the form of longitude and latitude, and timestamp for the message. The sending and receiving of messages are vulnerable to various types of attacks, thereby, encrypting these messages before sending, will provide security to data.

ASCON cryptographic system is applied and tested for 12 hours and cron job measures in every half second. The implementation was performed with different message lengths: 500, 100 or 10 bytes where two Raspberri Pi are connected with wi-fi system. The configuration of Raspberry Pi is RAM of 512 MB and a processor speed of 1 GHz. After the setup, a roundtrip time is measured to calculate the performance of the algorithm in IoT. ASCON algorithm is executed in three ways: only encryption, only hashing and both encryption and hashing.

The performance of the algorithm is measured with time as a parameter and CPU and RAM usage. The performance of AES cryptography and ASCON is compared with a controlled environment where no encryption was done. This comparison was done for different lengths of randomly generated messages. It was observed that AES has a minimum execution time varying between 5.3e–06 and 6.96e–06 as compared to the time taken by ASCON which varies from 1.85e–04 to 7.76e–04. However, AES performed better in terms of execution time but it can be used only on devices with additional memory of 512 bytes. A comparative analysis of RAM and CPU usage was performed with AES and ASCON-encryption, ASCON-hashing and ASCON-encryption-hashing. However, RAM usage is similar across all ASCON and AES algorithms but CPU usage is least for the AES algorithm as compared to all three execution variants of the ASCON algorithm i.e. hashing, encryption and hashing-encryption. During a run of 30-minute timespan, there is an impact on performance of the ASCON algorithm. It was observed that however, AES is an optimal solution in terms of application to IoT devices but ASCON can be executed on embedded devices which is difficult to achieve with the AES cryptography algorithm.

Lightweight Revocable Hierarchical Attribute-Based Encryption for Internet of Things: The proposed paper by Ali et al. [2] executes a lightweight encryption mechanism based on attributes. The purpose of the work was to resolve the problem of single authority for key generation in an attribute-based encryption scheme. In large IoT networks, multiple users request for secret key to the single key authority which raisesproblems.

The system works in five parts as follows: initialization, key delegation, data outsourcing, decryption and user revocation. During system initialization, a central authority (CA) generates a universal set of attributes, security parameters and master secret keys for domain authority (DA). The master secret key is kept by the central authority confidentially and the master secret keys for DAsare assigned to them respectively. During the key delegation phase, DAs generate one key, each for users and two keys for data owners. For users, secret key is generated according to their attributes; and public keys and secret keys are generated for owners of the data. Before generating a secret key for the user, DAs check for the attribute associated with the user. If no such attribute is liked by the user then either DA rejects the request for a secret key or generates a new attribute associated with the user.

During the data outsourcing phase, the data owner creates a ciphertext by encrypting the data using the secret key and storing it with the cloud service provider (CSP).Then, an authorized user is identified using an access tree and a partial ciphertext is sent to the CSP. The ciphertext is generated by the CSP using the public key of the data owner and

the data file is outsourced. In the decryption phase, a user with associated attributes asks for access to the data file then it checks if the attribute of the user exists in the set of attributes earlier generated by the central authority. If not, then further processing is aborted. If the attribute exists in the set then the decryption token generator generates a secret key and decryption token where the secret key is kept by the data user confidentially and the decryption token is sent to the CSP. The CSP then runs an algorithm to output partially decrypted ciphertext using a decryption token and parameters and sends it to the user. The data user then deciphers the complete data from partial ciphertext using secret keys and parameters. In the user revocation phase, if the data user misses the attribute then it needs to be ensured that the user no longer has a valid secret key associated with the respective attribute. For this purpose, DAs generate a new secret and public parameter and according to them, CSP creates a new ciphertext by re-encrypting the cipher texts in the access trees. Accordingly, a new authorized secret key linked to the attribute is updated based on new parameters.

The experimental results have shown that encryption time of attribute-based encryption algorithm is very less as compared to other algorithms mentioned in two other papers. However, the decryption time of all algorithms was almost similar. Also, it was observed that communication overhead is reduced considerably for attribute-based cryptography algorithms as compared to other algorithms used in other works. For e.g. only 2Kb is used in the proposed work as compared to 12Kb and 26Kb in other works for 100 leaf nodes. The communication overhead has been shifted from data owners to the cloud service provider.

The literature review conducted shows the importance is not given to assessing cryptographic algorithms for hardware implementation and its suitability. This involves evaluating factors such as hardware efficiency, energy consumption, and performance on IoT devices with limited resources. Also, these devices vary significantly in terms of processing power, memory, communication capabilities, and energy constraints. As a result, cryptographic algorithms must be adaptable to diverse IoT environments [17].

Proposed Algorithm

The proposed BLOCK cipher RLCA (Robust Lightweight Cryptographic Algorithm) is designed which uses the Add function, Rotate Function and X-OR function and is a non-linear MOD addition that utilizes X-OR and rotation for mixing.

Feistal Structure adoption to the current RLCA algorithm.

Algorithm Notation:

- x and y: 32-bit words representing the plaintext block as shown in Fig. 3
- k_i: 32-bit round key
- rol32(x, n): Rotate left operation on a 32-bit word x by n bits
- ror32(x, n): Rotate right operation on a 32-bit word x by n bits as shown in Fig. 4.

Encryption Algorithm Initialization

1. Initialize
   a. Plain text – 64 bits
   b. Key – 128 bits
   2. Split the 128-bit Key into four 32-bit subkeys: KeyPart1, KeyPart2, KeyPart3, and KeyPart4.
   3. Initialize round keys.

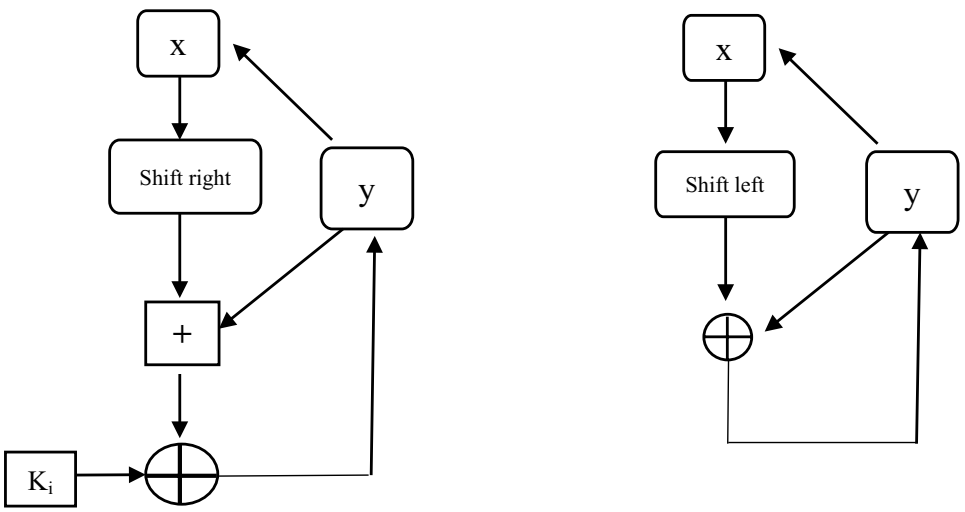

Fig. 3 Representation of a Round function

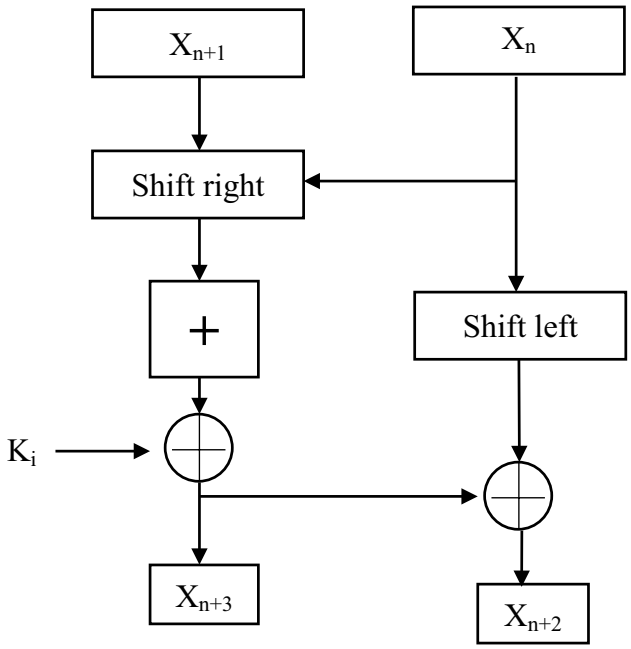

Fig. 4 Representation of sub-cipher after n steps of encryption

Encryption Algorithm

1. Initialize two 32-bit words, Left and Right, with the PlainText.

2. Perform a series of rounds.

a. Apply a round function to Left and Right using the current round key.

b. Update Left and Right using the round function results.

3. The final CipherText is the concatenation of Left and Right.

Key Schedule Update (for 128-bit key):

1. Rotate the subkey (KeyPart1) left by a fixed number of bits (e.g., 8 bits).

2. XOR the subkey (KeyPart1) with the current round number.

3. Rotate the subkey (KeyPart2) left by a fixed number of bits (e.g., 3 bits).

4. XOR the subkey (KeyPart2) with the current round number.

Round Function (for 128-bit key):

1. XOR Right with the Left rotated right by a fixed number of bits (e.g., 8 bits).

2. Add the current subkey to Right.

3. XOR Right with the Left rotated left by a fixed number of bits (e.g., 3 bits)

$$R_i = \left(L_i - 1 + ROR_r\left(L_i - 1\right)\right) \oplus K_i \quad (1)$$

where r is the rotation parameter, K_i is the round key for the ith round derived from the key schedule.

Decryption Algorithm

The inverse of the round function is used in the decryption process. This function undoes the operations performed by the round function in reverse order.

1. XOR with x: XOR is the 32-bit word y with the original value of x.

2. Rotate Right (ror32): Rotate the 32-bit word y to the right by CIPHER_B(fixed number of bits) bits.

3. XOR with Key: XOR the 32-bit word xwith the round key k.

4. Subtraction: Subtract the rotated y from the 32-bit word x.

5. Rotate Left (rol32): Rotate the 32-bit word x to the left by CIPHER _A(fixed number of bits) bits.

This proposed RLCA is implemented in C language. The necessary libraries included for building the cipher are stdint.h, stdio.h,time.h, stdlib.h. The helper functions used in the code are: ror64, rol32, and ror32 are helper functions for rotating bits. The typedef is used in C to create aliases for other data types 8, 16, 32 and 64-bit unsigned integers. The macros are made for the cipher round function with the name round Function and invert Round Function—the function clock_gettime to measure the encryption and decryption time.

Decoding Proposed Algorithm

The step-by-step process of the RLCA algorithm is detailed below. In the encryption phase, there are three subroutines called key scheduling, Initialization and computation phases. In the decryption phase, the same three subroutines will be there but will be processed in the reverse order as that of encryption.

Demonstration of the Process with Plain Text 0×0123456789ABCDEF as an Example

Encryption Phase

- PlainText: 64-bit value—0×0123456789ABCDEF
- Key: 128-bit value—0 × 0FED-CBA98765432100123456789ABCDEF

Key Scheduling:

- Split the 128-bit Key into four 32-bit subkeys:
- KeyPart1: 0xFEDCBA98; KeyPart2: 0×76,543,210; KeyPart3: 0×01234567;

KeyPart4: 0x89ABCDEF

Initialization:

- PlainText: 0×0123456789ABCDEF
- Divide it into two 32-bit words:
- Left: 0×01234567
- Right: 0×89ABCDEF

After all rounds, swap Left and Right:

- Left: 0×89ABCDEF
- Right: 0×01234567
- The CipherText is the concatenation of Left and Right:
- CipherText: 0×89ABCDEF01234567

Decryption Phase

CipherText: 0×89ABCDEF01234567.

Divide it into two 32-bit words:

- Left: 0×89ABCDEF
- Right: 0×01234567

For each round:

Apply the round function in the reverse order using the current round key.

- Update Left and Right using the round function results.
- After all rounds, swap Left and Right:
- Left: 0×01234567
- Right: 0×89ABCDEF

The PlainText is the concatenation of Left and Right:

PlainText: 0×0123456789ABCDEF

Results and Discussion

RLCA has been tested on different sizes of data. The size of plaintext test data files are as follows: 1 byte, 10 bytes, 1 kb, 100 kb and 100 mb. The encryption and decryption time is measured for all ciphers.

1. CIPHER 64-bit plain text and 128-bit key length:

It is evident from Fig. 5 that the proposed algorithm is taking approx. same time for encoding and decoding, Also the time of encoding and decoding linearly increases with the increase in data size, this shows the anticipated algorithm is linear in terms of implementation time which is one of the functional requirements of a cryptographic technique [8].

Algorithm Evaluation The developed RLCA (Robust Lightweight Cryptographic Algorithm)is based on the Feistel structure and makes use of both linear and nonlinear mathematical functions, this combination strengthens the algorithm's defences against a variety of attacks, especially cryptanalysis. The Feistel structure introduces confusion and diffusion by nature, which are necessary characteristics to hinder cryptanalytic attempts. Furthermore, the addition of both linear and nonlinear mathematical functions strengthens the algorithm's defence against attacks by making the encryption process more complex and making it harder for adversaries to use mathematical analysis to decrypt encrypted data. This combination of strategies guarantees that it withstands malicious efforts to compromise critical information.

For a varied throughput, the PRESENT algorithm's encryption and decryption time is measured and plotted in Fig. 6. For 1 byte of data, it takes 15 microseconds for encryption and decryption. On the contrary, the proposed algorithm takes 0.4 microseconds for the encryption-decryption cycle.

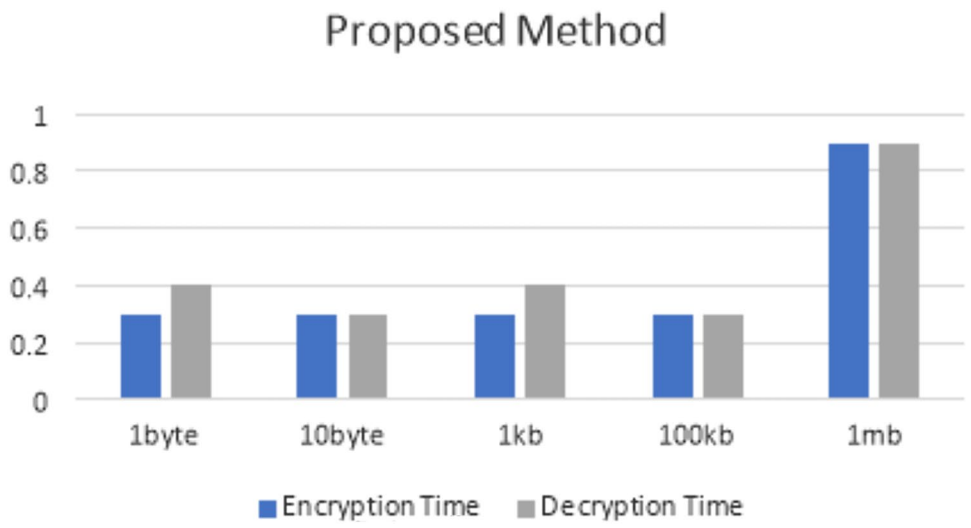

Fig. 5 Encryption time and Decoding time for various data sizes of the proposed algorithm

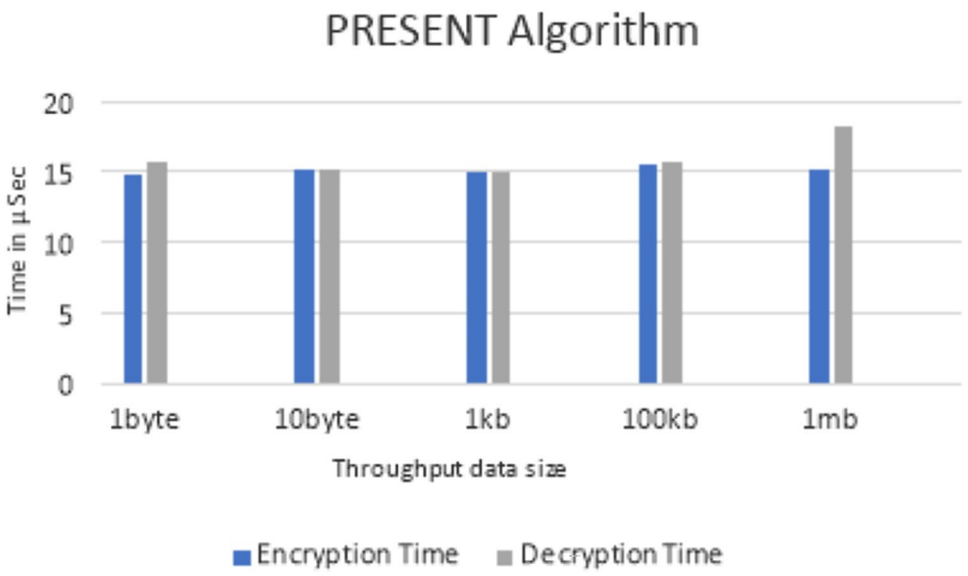

Fig. 6 Encryption and Decryption time of PRESENT algorithm

From Table 1, it was observed that the proposed Cipher has consumed less time during the encryption and decryption process. It was found that for 1 MB of data, the encryption-decryption time by the proposed Cipher_64_128 is 98% reduced as compared to NLCA algorithm. It is evident from the results that the proposed Cipher performed better as compared to the previous NLCA algorithm.

From Table 1, it can be derived that the proposed algorithm is taking 15µsec less as compared with the PRESENT algorithm for the data size of 1byte to 100 k bytes, on performing multiple iterations the results show that the algorithm developed is 97% efficient as compared with PRESENT algorithm, which is a benchmark algorithm in lightweight cryptography, graphical representation of time taken by RLCA and PRESENT is shown in Fig. 7.

Comparative Study: A comparative analysis is performed based on the time taken for encoding and the time taken for decryption. Also, the algorithm is evaluated for its efficiency over different data sizes.

Our proposed RLCA performance is compared with the NLCA (New lightweight cryptographic algorithm) Thabit *et al*. [1]. After careful analysis of NLCA, it is understood that 1 Mb data size is common between both, and therefore, is considered for comparison.

Cryptanalysis Using a 128-bit key and a 64-bit block size, it provides a lot of key space, which protects against brute force assaults. With 2^{128}potential combinations that would need to be looked through in order for the attacker to identify the right key. With today's technology, brute force attacks are computationally impossible due to their sheer number. Bitwise XOR, bit rotations, and addition modulo $2^{64}$are the operations carried out in each cycle of the RLCA algorithm. These actions add complexity to the encryption process, making it more difficult for attackers to take advantage of algorithmic faults, especially when repeated across several rounds.

The RLCA algorithm creates round keys from the original encryption key through an iterative key scheduling. This procedure increases the encryption's complexity and unpredictability, making it harder to carry out a successful brute force assault.Overall, when implemented appropriately and with good key management procedures, the RLCA variant is very resistant to brute force assaults due

Table 1 Time duration in encoding and decoding process by Cipher_64_128

Size	Proposed Method			PRESENT Algorithm		
	Time in Encryption(in µs)	Time in Decryption(in µs)	Overall time (in µs)	Time in Encryption (in µs)	Time in Decryption (in µs)	Overall time (in µs)
1byte	0.3	0.4	0.7	14.8	15.7	30.5
10byte	0.3	0.3	0.6	15.1	15.1	30.2
1 kb	0.3	0.4	0.7	15	15	30
100 kb	0.3	0.3	0.6	15.6	15.7	31.3
1000 kb(1mb)	0.9	0.9	1.8	15.1	18.2	33.3

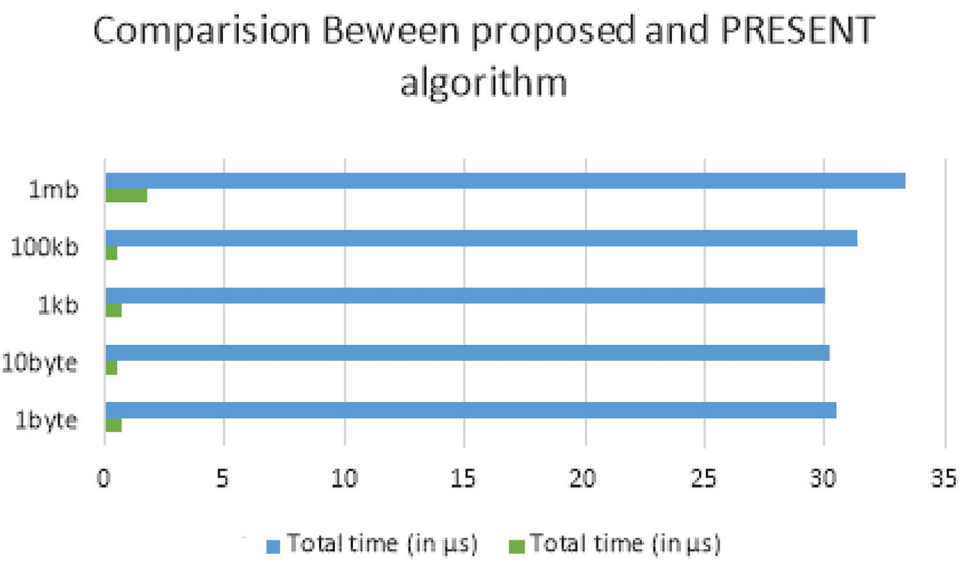

Fig. 7 Encryption and Decryption time comparison between proposed RLCA and PRESENT algorithm

to its huge key space, efficient cryptographic operations, and lack of known flaws.

Conclusion

This study has provided valuable insights into the challenges of designing a new algorithm for IoT-based devices. A new algorithm developed RLCA is presented which uses fiestal structure, and linear and non-linear mathematical techniques. A 64-bitblock size and a key length of a 128-bit key are chosen for the algorithm. The shifting and XOR operations are used for each round in generating the ciphertext. For the decryption of ciphertext, the key is generated in reverse order for each round and the round function is applied in the reverse order. The encryption time and decryption time taken by the proposed RLCA is noted for different input data sizes. A comparative analysis is done with the NLCA algorithm, where 1 Mb of the data file is chosen. NLCA and the proposed work both were tested on 1 Mb of data file, the encryption-decoding time by the anticipated method is 98% less as compared to other NLCA algorithms. It was concluded that the proposed RLCA is time-efficient during encoding and decryption processes. Results also show that as RLCA compared to the PRESENT algorithm, the performance is 97.2% better both in terms of encryption and decryption time. Future work includes the algorithm's resilience to new security risks and weaknesses before implementing it in the real world incorporating the algorithm into real-world applications and evaluating how well it works to secure sensitive data in various contexts.

Acknowledgements The research work was made possible with the support of R.V College of Engineering, Bengaluru, India which provided the necessary facilities.

Author Contributions This research work owes its success to the invaluable collaboration and collective efforts of all contributing authors. Each individual brought forth their unique expertise and unwavering dedication, thereby enriching the work with a wealth of diverse perspectives and insightful contributions.

Funding There are no particular grants from funding organizations for this research.

Data Availability The corresponding author can provide the datasets upon request.

Declarations

Conflict of Interest Hereby we are declaring that there is No Conflict of Interest related to this manuscript.

References

1. Thabit F, Alhomdy S, Al-Ahdal AHA, Jagtap S. A new lightweight cryptographic algorithm for enhancing data security incloud computing. 2021. https://doi.org/10.1016/j.gltp.2021.01.013.
2. Ali M, Sadeghi M-R, Liu X. Lightweight revocable hierarchical attribute-based encryption for Internet of Things. IEEE Access. 2020. https://doi.org/10.1109/access.2020.2969957.
3. He Q, Zhang N, Wei Y, Zhang Y. Lightweight attribute-basedencryption scheme for mobile cloud-assisted cyber-physical systems. 2018.
4. Avery, J, Fraelich B, Duran W, Lee A, Sullivan A, Mechalke Z, Bobby Birrer M, Dick S, Cochran J. Analysis of Practical Application of Lightweight Cryptographic Algorithm ASCON
5. Thabit F, Can O, Alhomdy S, Al-Gaphari GH, Jagtap S. A novel effective lightweight homomorphic cryptographic algorithm for

data security in cloud computing. 2022. https://doi.org/10.1016/j.ijin.2022.04.001.
6. Viswanath G, Krishna PV. Hybrid encryption framework for securing big data storage in a multi-cloud environment. Evol Intell. 2020. https://doi.org/10.1007/s12065-020-00404-w.
7. Bhardwaj I, Kumar A, Bansal M. A review on lightweight cryptography algorithms for data security and authentication in IoTs. In: 2017 4th International Conference on signal processing, computing and control (ISPCC). IEEE, 2017.
8. Thakor VA, Razzaque MA, Khandaker MR. Lightweight cryptography algorithms for resource-constrained IoT devices: a review, comparison and research opportunities. IEEE Access. 2021;9:28177–93.
9. McKay K, et al. Report on lightweight cryptography. No. NIST Internal or Interagency Report (NISTIR) 8114 (Draft). National Institute of Standards and Technology, 2016.
10. Hamsha K, Nagaraja GS. Threshold cryptography based light weight key management technique for hierarchical WSNs. Ubiquit Commun Netw. 2019.
11. Khan MN, Rao A, Camtepe S. Lightweight cryptographic protocols for IoT-constrained devices: a survey. IEEE Internet of Things J. 2020;8(6):4132–56.
12. Bhattasali T. Licrypt: lightweight cryptography technique for securing smart objects in internet of things environment.CSI Commun. 2013; pp. 26–36.
13. Chaudhary, Kumar RR, Chatterjee K. An efficient lightweight cryptographic technique for IoT based E-healthcare system. In: 2020 7th International Conference on Signal Processing and Integrated Networks (SPIN). IEEE, 2020.
14. Hassan A. Implementation of lightweight cryptographic algorithms in IoT devices and sensor networks. In: Proceedings of the Future Technologies Conference. Cham: Springer International Publishing, 2022.
15. Rana M, Mamun Q, Islam R. Lightweight cryptography in IoT networks: a survey. Futur Gener Comput Syst. 2022;129:77–89.
16. Thakor VA, Razzaque MA, Khandaker MR. Lightweight cryptography algorithms for resource-constrained IoT devices: a review, comparison and research opportunities. IEEE Access. 2021;19(9):28177–93.
17. Khan MN, Rao A, Camtepe S. Lightweight cryptographic protocols for IoT-constrained devices: a survey. IEEE Internet Things J. 2020;8(6):4132–56.
18. Dutta IK, Ghosh B, Bayoumi M. Lightweight cryptography for internet of insecure things: A survey. In: 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC) 2019 Jan 7; pp. 0475–0481. IEEE.