Cihan University-Erbil Scientific Journal (CUESJ)

REVIEW ARTICLE

Analysis the Structures of Some Symmetric Cipher Algorithms Suitable for the Security of Internet of Things Devices

Khalid F. Jasim*, Reem J. Ismail, Abdullah A. Nahi Al-Rabeeah, Soma Solaimanzadeh

Department of Computer Science, Cihan University-Erbil, Kurdistan Region, Iraq

ABSTRACT

In the past years, the internet of things (IoT) used in different applications and very wide range of fields (e.g. cloud services, smart environments, logistics, social and personal domains, and health-care services). The IoT included a variety of components and devices such as radio frequency identification devices, wireless sensors, actuators, and wireless networks. Furthermore, the IoT with smart devices adopted in various companies, organizations, and public services systems. For instance, some devices such as Notebooks and smartphones have been used to perform different management activities and duties. These smart devices relied on data exchange and data storage resources in clouds computing services. In this context, the saved data and exchanged data required protection against hacking operations, transferred with more secure communications channels, and safe storage environment in the clouds and local storage systems. In this paper, we proposed some encryption algorithms to tackle the issue of data confidentiality in the IoT applications. This research provided analysis and investigation of these encryption algorithms in light of components of the designs, versions of these algorithms, encryption keys, block size, round functions, and the techniques used in the designs.

Keywords: Internet of things applications, block ciphers, stream ciphers, encryption algorithms, data security

INTRODUCTION

In the previous years, the extensive growth of internet of things (IoT) applications reflected, positively, on the different activities of user's life. The basic concepts of IoT were presented in 1999, in which there was massive progress in some technologies such as radio frequency identification (RFID devices), Wireless Sensor Networks, and wireless devices. Furthermore, the IoT was intended to establish connections based on various objects (e.g. RFID devices, sensors, actuators, and readers).[1] The IoT presented important applications in different fields (e.g. smart environments, logistics, social and personal domains, and health-care services).[2]

The structure of IoT based on cloud services was presented in Zhou *et al.*,[3] and the requirements of user's privacy and data confidentiality have been covered. In this context, the security requirements focused on location privacy, identity privacy, cloud security, layer attack, and node attack. The IoT medical applications were used in the design of smart medication dispenser devices, in which can help the patients to receive their medications as specified in the treatment schedule. These smart dispenser devices designed based on IoT applications, smartphone devices, and cloud computing services for data storage.[4] Furthermore, some IoT used in the design of E-Health Database systems. In these E-Systems, the patient's status can be checked and monitored more efficiently. The researchers in Yang *et al.*[5] presented analysis study of privacy issues in these systems. Then, the researchers proposed a system relied on big data and IoT applications, in which this system offered data confidentiality of patients' records and protected communications. The system relied on authentication key technique and algorithm for secure data transfer through the medical network.

On the other hand, in IoT technologies, huge number of smart devices are accessed various services through the Internet and wireless networks and different types of sensitive information are exchanged through these networks. Thus, some security issues appeared related to data confidentiality and privacy of users and smart devices.[6-8] To overcome and mitigate the security problems, some symmetric ciphers have been proposed that provide data confidentiality and privacy of users and smart devices.[9]

Corresponding Author:
Khalid F. Jasim, Department of Computer Science, Cihan University-Erbil, Kurdistan Region, Iraq.
E-mail: khalid.jassim@cihanuniversity.edu.iq

Received: June 13, 2021
Accepted: July 18, 2021
Published: September 01, 2021

DOI: 10.24086/cuesj.v5n2y2021. pp 13-19

In recent years, a variety of devices (e.g., Sensors, smart devices, and RFID systems) have been adopted by organizations, companies, and public services. These smart devices are interconnected through IoT systems. There was data transfers and data storage through these systems and networks; therefore, there was a demand to protect these data. The cryptographic algorithms (e.g., lightweight cryptographic [LWC] algorithms, LWC) can provide data security in IoT systems and applications. The researchers in Alahdal and Deshmukh[10] conducted a survey study on LWC encryption algorithms that can support data confidentiality in IoT applications. This study focused on some LWC encryptions algorithms (e.g. Hash Functions and Block Ciphers Algorithms), and analyzed the features of these algorithms (e.g., energy consumption of devices, required area in designs, throughput of algorithms, and latency). Furthermore, the researchers in Thakor *et al*.[11] proposed the LWC encryption algorithms to protect and secure the communications between the smart devices which are used in IoT applications. The researchers concentrated on comparing the LWC encryption algorithms in light of cost of implementations, performance of these algorithms in software and hardware implementations, and various analysis attacks as well.

Moreover, many users rely on some devices such as notebooks and Smartphones to perform their duties, and these devices are connected via cloud computing services, in which there are data transfers among these devices, clouds, and IoT applications. In such case, the exchanged data require protection and must be transferred in secure and safe communications. The LWC Block ciphers algorithms have been proposed to solve the problems of data security in IoT networks. For instance, some Block ciphers algorithms (e.g., advanced encryption standard [AES], PRESENT, LBlock, and Skipjack) have been proposed as security encryption algorithms, and the features of these algorithms were investigated such as performance of algorithms, consumption of energy, time of execution during encryption/decryption operations, required memory space of these algorithms, and throughputs of these algorithms.[12] Furthermore, a variety of stream ciphers and block ciphers encryption algorithms were proposed to provide data protection, secure data transmissions, and data integrity in wireless mobile communications and smartphones device as well (e.g., AES, A5, SNOW, and ZUC).[13,14]

The sections of our research paper will be presented as follows. The IoT backgrounds, concepts, devices, services, algorithms, and features are presented in section 1. Section 2 focused on analyzing AES cipher algorithm (e.g., secret keys, versions of AES, rounds, and math transformations). Section 3 investigated some components of RC6 cipher algorithm (e.g., Encryption keys, block size, the 4 registers, array S[], and encryption steps). Section 4 described various elements of Twofish cipher algorithm (e.g., encryption keys, function F, S-boxes, rounds, MDS matrix, and encryption process). Section 5 analyzed the components of SPECK cipher algorithm (e.g. Block size, round function, logic operations [ARX], parameters, and versions of SPECK cipher). Section 6 investigated the components of lightweight encryption algorithm (LEA) encryption algorithm (e.g., Versions of LEA, block size, encryption keys, round function with logic and rotation operations [ROR and ROL], and encryption process). Section 7 concentrated on various components of ChaCha20 encryption algorithm (e.g., Secret keys, state matrix, quarter round function [QRF], and process of key stream). Section 8 provides analysis and discussion of IoT applications and the proposed encryption algorithms that can offer data security in IoT systems and applications. Finally, the conclusions of our work are presented in section 9.

ANALYSIS OF AES CIPHER ALGORITHM

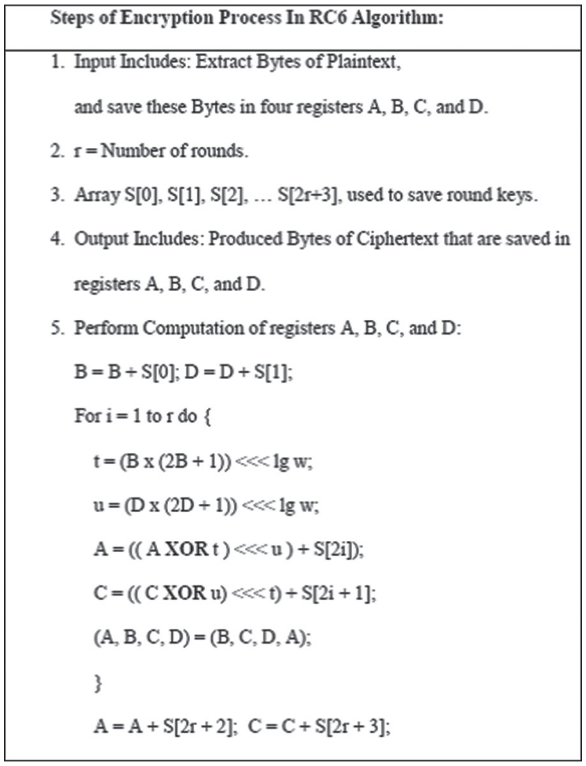
Daemen and Rijmen proposed the design of AES as encryption cipher algorithm for various types of data security applications. The design of AES relies on substitution-permutation network, linear layers, nonlinear layers, and add round key operation. This cipher system depends on three secret keys (128 bits, 192 bits, and 256 bits). Furthermore, the design of AES includes three types (e.g., AES-128 with 10 rounds, AES-192 with 12 rounds, and AES-256 with 14 rounds).[15] The AES used to encrypt input block of size (128 bits), the input block arranged as 16 bytes (B0, B1, B2,…, B15), and these 16 bytes saved in state array (4 rows × 4 columns). The design of AES relies on round function; the round function includes four operations. AddRoundKey operation, which is responsible for adding the bytes of round key to the bytes of state array. SubBytes operation, which is used to apply Substitution Box (S-Box) on each bytes in state array. ShiftRows, which shifts the bytes of Row(i) in state array according to defined numbers. MixColumns operation, in this operation each column of state array is multiplied by defined and constant matrix (MDS Matrix).[16]

ANALYSIS OF RC6 CIPHER ALGORITHM

The design of RC6 cipher algorithm identified by version (RC6-w/r/b), in which w represented the number of bits in word w (w=32 bits), r represented the number of rounds in this cipher, and b defines the number of bytes adopted for encryption secret key. The input block size in RC6 cipher determined by 128 bits. This algorithm relied on different lengths of secret encryption key (e.g., Key size equal to 128 bits, 192 bits, and 256 bits).[17,18] This algorithm is block cipher, and relies on Feistel cipher structure, in which the encryption depends on input of plaintext (with 2w bits and k key), process the input by sequence of substitution and permutation operations through r rounds, then produce the block of ciphertext. RC6 includes four registers (A, B, C, and D, length of each register equal to 32 bits), these registers used to save input of plaintext/cipher text blocks during encryption and decryption processes.[19] Furthermore, the array S[ ] (S[0], S[1], S[2],…, S[2r + 3], r=number of rounds) is used to save the rounds keys and these rounds keys are adopted in encryption/decryption operations. The (32 bits) words of array S[ ] are extracted from secret key bytes (128 bits, 192 bits, and 256 bits) according to key expansion algorithm.[20] In RC6, encryption process is shown in Figure 1:[20]

ANALYSIS OF TWOFISH CIPHER ALGORITHM

The Twofish algorithm classified as symmetric crypto algorithm. The design adopted block ciphers technique, the sender and receiver must possess the same software of this algorithm, and



Steps of Encryption Process In RC6 Algorithm:

```
1. Input Includes: Extract Bytes of Plaintext,
   and save these Bytes in four registers A, B, C, and D.
2. r = Number of rounds.
3. Array S[0], S[1], S[2], … S[2r+3], used to save round keys.
4. Output Includes: Produced Bytes of Ciphertext that are saved in
   registers A, B, C, and D.
5. Perform Computation of registers A, B, C, and D:
   B = B + S[0]; D = D + S[1];
   For i = 1 to r do {
      t = (B x (2B + 1)) <<< lg w;
      u = (D x (2D + 1)) <<< lg w;
      A = (( A XOR t ) <<< u ) + S[2i]);
      C = (( C XOR u) <<< t) + S[2i + 1];
      (A, B, C, D) = (B, C, D, A);
      }
      A = A + S[2r + 2];  C = C + S[2r + 3];
```

Figure 1: Encryption steps in RC6 cipher algorithm

same secret encryption keys to exchange cipher text messages. The Twofish cipher can operates with three types of secret encryption keys (e.g., 128 bits, 192 bits, and 256 bits), and one type of input block size which is equal to (128 bits). The inner structure of this algorithm includes 16 rounds of Feistel cipher networks and function F. The function F consists of (4) S-boxes (each with 8X8 bits, and key dependent mode) and Maximum Distance Matrix (MDS with 4X4 constant values). Furthermore, this function adopts bitwise rotation operations and math transformation with (Pseudo-Hadamard).[21,22]

In the encryption process [Figure 2], the input plaintext (128 bits) is partitioned into 4 words (each word with 32 bits). These 4 words are XORed with 4 key words (k#0, k#1, k#2, and k#3). The 16 rounds of this algorithm are performed. In each round of the 16 rounds, the 2 words in the left side are employed as input to function (g), in which function (g) relies on 4 S-boxes (S-box#0, S-box#1, S-box#2, and S-box#3) and MDS matrix. Each of the 4 S-boxes process input of (8 bits) and produce encoded output of (8 bits). The 4 produced outputs are multiplied by the MDS matrix. As shown in Figure 2, the outputs of MDS matrix are processed to be combined by using math transformation with (Pseudo-Hadamard). The results are added with values of 2 key words (k#2r+8 and k#2r+9, where r = round number). Then, these results processed by 2 XOR operations. The aforementioned processing steps are performed in each of 16 rounds and then produce the 128 bits output of cipher text.[23]

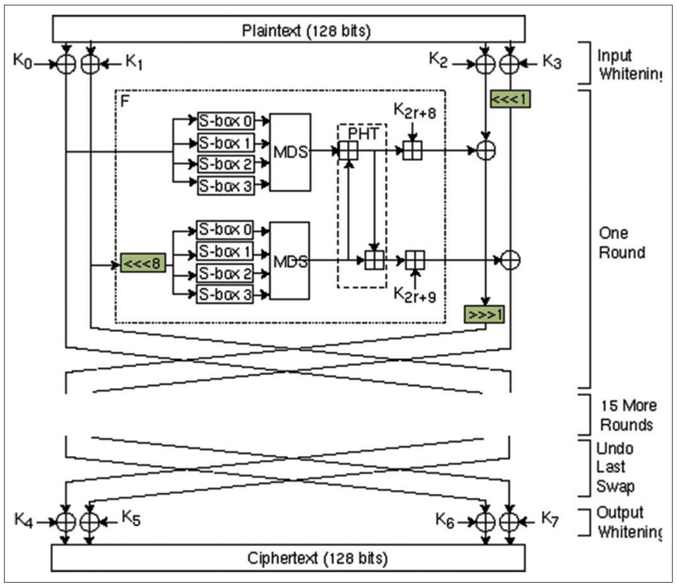


Figure 2: Structure of Twofish cipher algorithm

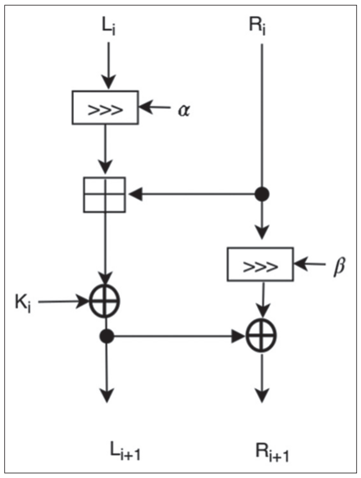


Figure 3: Structure of round function in SPECK cipher

ANALYSIS OF SPECK CIPHER ALGORITHM

The SPECK cipher algorithm designed based on block ciphers techniques. This cipher algorithm constructed for light weight smart devices with small hardware requirements. The components of this light block cipher relied on basic logic operations (i.e., ADDITION, ROTATION, and XOR logic operations, or ARX operations). In this context, these logic operations can be constructed efficiently in Light Weight and small hardware devices (e.g., FPGA, Means the Field Programmable Gates Arrays). The structure of SPECK cipher organized based on Feistel cipher structure, employed different types of input/output block sizes (e.g., 32 bits, 48 bits, 64 bits, 96 bits, and 128 bits), and possessed high performance when implemented using software on (64 bits) processors. The round function in SPECK cipher [Figure 3] adopted three basic logic operations (ARX: ADDITION, ROTATION, and XOR logic operations).[24]

The round function includes left half of the input (Li, i=round No. i) and right half of the input (Ri, i=round No. i), Ki which represents the (n bits) of round key specified for round (i), rotations operations defined by (>>> α, shift to right by α bits) and (<<< β, shift to left by β bits), modulo addition operation (), and the XOR operation (). The outputs

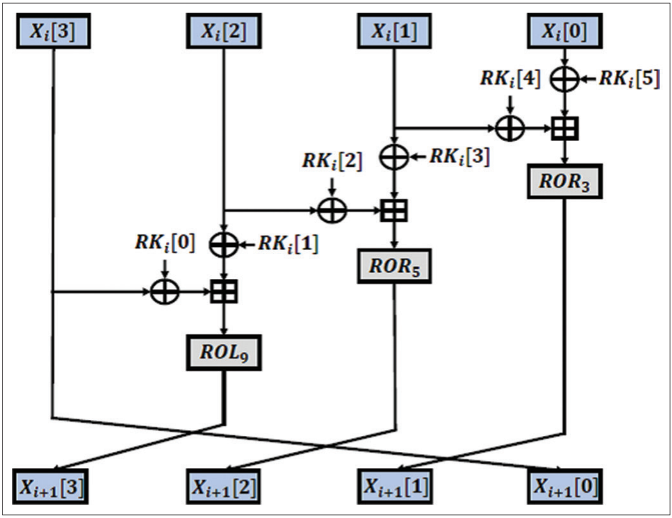


Figure 4: Structure of round function in lightweight encryption algorithm cipher

of this round function identified by the left half output (Li+1) and right half output (Ri+1), in which can be computed as shown in the following formulas: [25]

$$Li+1 = ((Li >>> \alpha) \boxplus R)) \oplus Ki, \text{ and } Ri+1 = (Ri <<< \beta) \oplus Li+1 \quad (1)$$

The SPECK cipher can operates with different versions, therefore the parameters (α and β) can be defined with different values (e.g. α = 7 or α = 8, and β = 2 or β = 3).

Various versions of SPECK cipher are depicted in Table 1. For instance, SPECK 32/64 represents the version that includes Block Size (32 bits), Secret Key (64 bits), parameters (α=7 and β=2), and number of rounds (=22) SPECK 128/256, this version adopts Block Size (128 bits), Secret Key (256 bits), parameters (α=8 and β=3), and number of rounds (=34)

ANALYSIS OF LEA CIPHER ALGORITHM

The LEA encryption algorithm designed based on the concepts of block ciphers, offered high speed of data encryption, and implemented in software applications. This algorithm possessed suitable security level, in which can withstand against various types of cryptanalysis attacks (e.g. linear analysis attack, and differential analysis attack), and also has been proved as secure and efficient encryption algorithm.[26] The LEA algorithm includes three versions (LEA-128, LEA-192, and LEA-256). The first version (LEA-128) relies on block size (128 bits), key size (128 bits), round number (24), and word size (32 bits). The second version (LEA-192) employs block size (128 bits), key size (192 bits), round number (28), and word size (32 bits). The third version (LEA-256) adopts block size (128 bits), key size (256 bits), round number (32), and word size (32 bits). The design of LEA constructed with round functions. The round function [Figure 4] consists of (6) XORs (XOR logic operation), (3) additions (Modular Addition operation, with mod value 2^{32}), and (3) rotations (Rotations with circular type operation, and identified by ROR3 rotation, ROR5 rotation, and ROL9 rotation operations).[27]

The LEA encryption algorithm[28] [Figure 5] was proposed to perform data confidentiality in different IoT applications

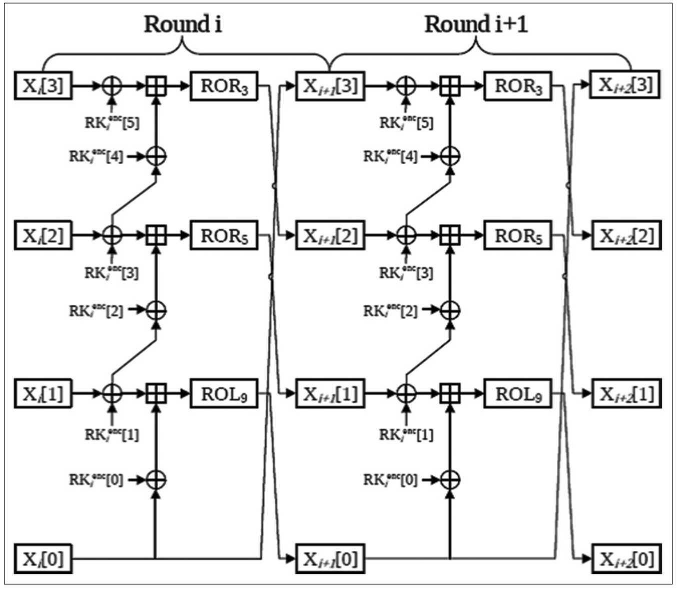


Figure 5: Lightweight encryption algorithm with encryption process

Table 1: Versions of SPECK cipher with different parameters

Block size of (Input/Output) In bits	Size of secret key in bits	Value of α	Value of β	Total no. of rounds
32	64	7	2	22
48	72	8	3	22
48	96	8	3	23
64	96	8	3	26
64	128	8	3	27
96	96	8	3	28
96	144	8	3	29
128	128	8	3	32
128	192	8	3	33
128	256	8	3	34

(e.g. Mobile Based Environments, Cloud Computing Services, and Embedded Devices). The three versions of LEA encryption algorithm (LEA-128, LEA-192, and LEA-256) supported data confidentiality through data encryptions. For instance, encryption process starts with extracting 128 bits of plaintext, represents these 128 bits by 4 words (Xi[0], Xi[1], Xi[2], Xi[3], and each word=32 bits). In round number (i), the word Xi[0] transferred into word Xi+1[3] in next round (i+1). The word Xi[1] XORed with round key RKi[1], perform Modular Addition on this word, apply rotation operation (ROL9), then the result word transferred into word Xi+1[0] in next round (i+1). The word Xi[2] XORed with round key RKi[3], apply Modular Addition on this word, apply rotation operation (ROR5), then the result word transferred into word Xi+1[1] in next round (i+1). The word Xi[3] XORed with round key RKi[5], perform Modular Addition on this word, apply rotation operation (ROR3), then the result word transferred into word Xi+1[2] in next round (i+1). For example, in first version (LEA-128) the number of rounds equal (24), therefore the aforementioned encryption process is repeated on the plaintext words (X[0], X[1], X[2], and X[3]), then after round No. 24 the 128 bits of encrypted data are produced.

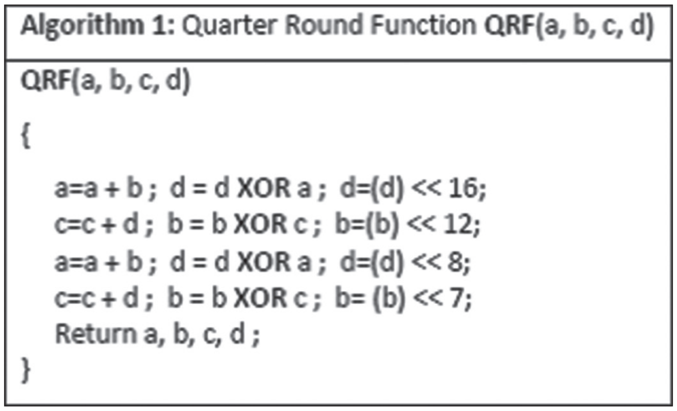
ANALYSIS OF CHACHA20 CIPHER ALGORITHM

The stream ciphers techniques relied on composing elements of key stream based on predefined and dynamic state with initial values. In the encryption process, the elements of key stream mixed with the elements of plaintext then the output of this process represent the elements of cipher text (e.g. Encrypted Data). Some encryption algorithms, in which relied on stream ciphers techniques, have been proposed to perform the data confidentiality in IoT applications.[29] In this context, the ChaCha20 encryption algorithm designed based on stream ciphers techniques, and supported data confidentiality in various data security applications (e.g. IoT devices, Google Chrome in smart devices, security of protocols in OpenSSL, and OpenSSH).[30] The design of this algorithm depends on a number of ARX logical operations (i.e. Addition operation, rotation operation, and XOR logical operation). In this design, the ARX logical operations constructed on (32 bits) words and it was found as efficient encryption algorithm in hardware and software implementations. For instance, each round in this algorithm constructed with 16 (XOR logic operations), 16 (Modulo Additions operations with Mod 2^{32}), and 16 (Rotations operations).[31]

The ChaCha20 cipher algorithm depends on secret key K with size (256 bits), K defined as (K=k0, k1, k2,…, k7), and it is organized to process words of (32 bits) in each time. The algorithm includes a state which is composed as matrix (4X4 matrix); this state matrix consists of 16 values and each value represented by (32 bits word). The 16 elements of state matrix defined as 4 rows (X0,X1,X2,X3; X4,X5,X6,X7; X8,X9,X10,X11; X12,X13,X14,X15). In the encryption process, the state matrix is loaded with predefined fixed values (e.g. X0=0x61707865, X1=0x3320646e, X2=0x79622d32, and X3=0x6b206574), the other elements are loaded by elements from secret key K (X4=k0, X5=k1, X6=k2, X7=k3 …, X11=k7), 32-bit counter (X12=C0), and 3 (32-bit) nonce-values (X13=n0, X14=n1, X15=n2).[32]

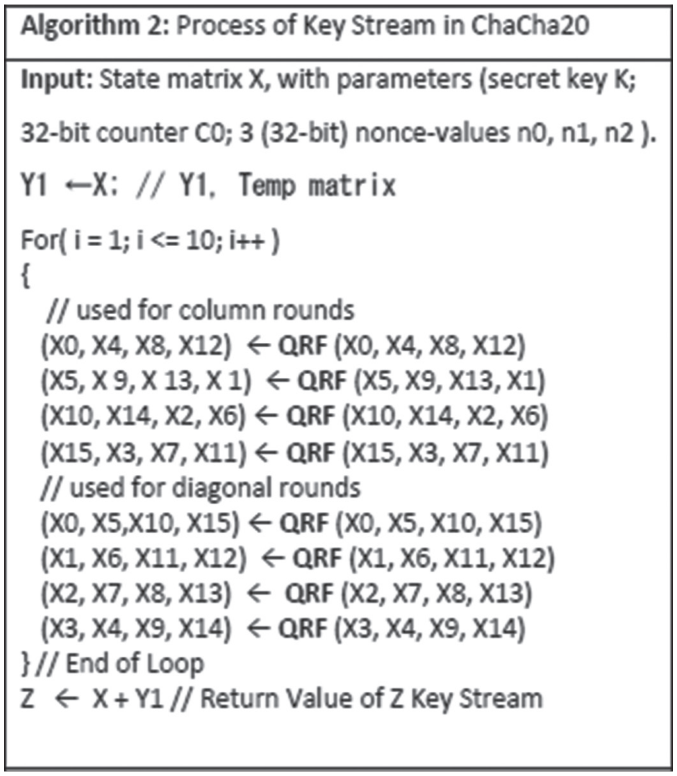
The QRF, used in ChaCha20 cipher algorithm, process input of 4 words (a, b, c, d, each with 32 bits), perform sequence of operations then the result includes updated 4 words (a, b, c, d, each with 32 bits) as shown in the following algorithm 1 QRF [Figure 6].

The process of key stream is shown in algorithm 2 [Figure 7]. The process starts with initialize the state matrix X with values from secret key K, counter value C0, nonce-values (n0, n1, n2), and constant values as shown in state matrix [Figure 8]. The 16 elements of state matrix X are processed for 20 rounds by using QRF function, and then produce the Z Key Stream. The Z Key Stream is used in encryption and decryption operations of input data.[33]



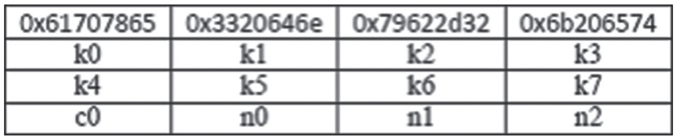
```
Algorithm 1: Quarter Round Function QRF(a, b, c, d)
QRF(a, b, c, d)
{
    a=a + b ;  d = d XOR a ;  d=(d) << 16;
    c=c + d ;  b = b XOR c ;  b=(b) << 12;
    a=a + b ;  d = d XOR a ;  d=(d) << 8;
    c=c + d ;  b = b XOR c ;  b= (b) << 7;
    Return a, b, c, d ;
}
```

Figure 6: Quarter round function in ChaCha20



```
Algorithm 2: Process of Key Stream in ChaCha20
Input: State matrix X, with parameters (secret key K;
32-bit counter C0; 3 (32-bit) nonce-values n0, n1, n2 ).
Y1 ←X: // Y1. Temp matrix
For( i = 1; i <= 10; i++ )
{
  // used for column rounds
  (X0, X4, X8, X12)  ← QRF (X0, X4, X8, X12)
  (X5, X 9, X 13, X 1)  ← QRF (X5, X9, X13, X1)
  (X10, X14, X2, X6) ← QRF (X10, X14, X2, X6)
  (X15, X3, X7, X11) ← QRF (X15, X3, X7, X11)
  // used for diagonal rounds
  (X0, X5,X10, X15) ← QRF (X0, X5, X10, X15)
  (X1, X6, X11, X12)  ← QRF (X1, X6, X11, X12)
  (X2, X7, X8, X13)  ←  QRF (X2, X7, X8, X13)
  (X3, X4, X9, X14)  ← QRF (X3, X4, X9, X14)
} // End of Loop
Z  ←  X + Y1 // Return Value of Z Key Stream
```

Figure 7: Process of key stream in ChaCha20



0x61707865	0x3320646e	0x79622d32	0x6b206574
k0	k1	k2	k3
k4	k5	k6	k7
c0	n0	n1	n2

Figure 8: Elements of (4X4) state matrix

DISCUSSION

IoT

The IoT applications designed to create connections based on different objects such as RFID devices, sensors, actuators, and readers. These applications covered vital areas such as smart environments, logistics, social and personal domains, health-care services, and relied on cloud services. The IoT systems with smart devices have been adopted in many organizations and public services. In these systems, the exchanged data required data protection and secure data transmissions through communication systems. Therefore, various encryption algorithms have been proposed that supported data confidentiality in IoT systems and applications.

Table 2: Analysis of the structures of encryption algorithms

Cipher algorithm	Type of cipher	Size of secret key (bits)	Block size (bits)	Total no. of rounds	Structure of cipher
AES	Block cipher	128/192/256	128	10/12/14	SPN
RC6	Block cipher	128/192/256	128	20	SPN
Twofish	Block cipher	128/192/256	128	16	SPN
SPECK	Block cipher	32-128	64-256	22-34	ARX
Lightweight encryption algorithm	Block cipher	128/192/256	128	24/28/32	ARX
ChaCha20	Stream cipher	256	512	20	ARX

AES Algorithm

This encryption algorithm supported data security in different IoT, Internet, and cloud services applications. This algorithm adopted block ciphers concepts, three types of secret keys (e.g., 128 bits, 192 bits, and 256 bits), block size (128 bits), three types of rounds (10, 12, and 14), and relied on substitutions and permutations networks (SPN technique) [Table 2]. Each round in the structure of this algorithm included four types of operations (i.e. Add-RoundKey, Sub-Bytes, Shift-Rows, and Mix-Columns). These operations enhanced the security of the AES algorithm.

RC6

This algorithm relied on block ciphers, three types of secret keys (128 bits, 192 bits, and 256 bits), block size (128 bits), 20 rounds, and the design constructed with Feistel cipher structure [Table 2]. The algorithm employed 4 registers (A, B, C, and D, and each register loaded with 32 bits), state array S[ ] used for saving the values of round keys.

Twofish

The algorithm designed with block ciphers techniques, 3 secret keys (128 bits, 192 bits, and 256 bits), and block size equal to (128 bits). The structure of this algorithm involved 16 rounds, each round contains function F. Function F constructed with 4 S-boxes (S-box#0, S-box#1, S-box#2, and S-box#3), and MDS matrix with fixed values.

SPECK

This algorithm classified as light block cipher and suitable for light weight smart devices. The algorithm used three types of logic operations (ADDITION, ROTATION, and XOR logic operations [ARX])). This algorithm designed with Feistel cipher structure, and included various types of block size (32 bits - 128 bits), different secret keys (64 bits – 256 bits), and different number of rounds (22–34). Each round employed 2 ROTATION operation, 2 XOR operation, and 1 ADDITION operation.

LEA

This algorithm designed with three versions (LEA-128, LEA-192, and LEA-256). The LEA-128 algorithm used secret key (128 bits), block size (128 bits), and 24 rounds. The LEA-192 version adopted secret key (192 bits), block size (128 bits), and 28 rounds. The LEA-256 version included secret key (256 bits), block size (128 bits), and 32 rounds. The round function in this algorithm designed with (6) XORs, (3) additions, and (3) rotations (ROR3 rotation, ROR5 rotation, and ROL9 rotation operations).

ChaCha20

This algorithm designed based on stream ciphers techniques. The algorithm included secret key (256 bits), block size (512 bits), and 20 rounds. Each round employed (16) XOR, (16) Additions Mod (2^{32}), and (16) Rotations operations (ARX operations). Furthermore, the algorithm used state matrix X[4, 4] with 16 elements processed in QRF function during data encryption/decryption operations.

CONCLUSION

In previous years, the IoT applications deployed in various fields (e.g., cloud services, healthcare systems, environment of smart cities, smart logistic applications, transportations systems, and agriculture area with smart tools). The IoT devices used in these fields experienced data security issues. Some encryption algorithms proposed to overcome the problems of data confidentiality during the data transmissions and data storage environments such as cloud computing services. This research focused on analyzing the proposed encryption algorithms that can be used for data protections. In this paper, different features and components were investigated such as versions of algorithms, secret keys, block size, number of rounds, types of ciphers, logic operations, math transformations, and the structures of these algorithms as well.

REFERENCES

1. J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami. Internet of things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645-1660, 2013.
2. L. Atzori, A. Iera and G. Morabito. The internet of things: A survey. *Computer Networks*, vol. 54, no. 15, pp. 2787-2805, 2010.
3. J. Zhou, Z. Cao, X. Dong and A. V. Vasilakos. Security and privacy for cloud-based IoT: Challenges. *IEEE Communications Magazine*, vol. 55, no. 1, pp. 26-33, 2017.
4. M. H. Shukur. Design a Mobile Medication Dispenser Based on IoT Technology. In: 3rd *International Conference on Communication Engineering and Computer Science* (*CIC-COCOS'19*), 2019.
5. Y. Yang, X. Zheng, W. Guo, X. Liu and V. Chang. Privacy-preserving fusion of IoT and big data for e-health. *Future Generation Computer Systems*, vol. 86, pp. 1437-1455, 2018.

6. S. Singh and N. Singh. Internet of Things (IoT): Security Challenges, Business Opportunities and Reference Architecture for E-commerce. In: 2015 *International Conference on Green Computing and Internet of Things* (*ICGCIoT*), IEEE, pp. 1577-1581, 2015.
7. M. G. Samaila, M. Neto, D. A. B. Fernandes, M. M. Freire and P. R. M. Inácio. Challenges of securing internet of things devices: A survey. *Security and Privacy*, vol. 1, no. 2. e20, 2018.
8. Q. Jing, A. V. Vasilakos, J. Wan, J. Lu and D. Qiu. Security of the internet of things: Perspectives and challenges. *Wireless Networks*, vol. 20, no. 8, pp. 2481-2501, 2014.
9. H. Shin, H. K. Lee, H. Y. Cha, S. W. Heo and H. Kim. IoT Security Issues and Light Weight Block Cipher. In: 2019 *International Conference on Artificial Intelligence in Information and Communication* (*ICAIIC*), IEEE, pp. 381-384, 2019.
10. A. Alahdal and N. K. Deshmukh. A systematic technical survey of lightweight cryptography on IoT environment. *International Journal of Scientific and Technology Research*, vol. 9, no. 3, p. 16, 2020.
11. V. Thakor, M. A. Razzaque and M. R. A. Khandaker. Lightweight cryptography algorithms for resource-constrained IoT devices: A review, comparison and research opportunities. *IEEE Access*, vol. 9, pp. 77-93, 2021.
12. P. Panahi, C. Bayılmış, U. Çavuşoğlu and S. Kaçar. Performance evaluation of lightweight encryption algorithms for IoT-based applications. *Arabian Journal for Science and Engineering*, vol. 46, no. 4, pp. 4015-4037, 2021.
13. K. F. Jasim and I. F. Al-Shaikhli. Mobile Technology Generations and Cryptographic Algorithms: Analysis Study. In: 2015 4th *International Conference on Advanced Computer Science Applications and Technologies* (*ACSAT*), IEEE, pp. 50-55, 2015.
14. K. F. Jasim and I. F. Al-Shaikhli. Analysis of Confidentiality Algorithms in Different Mobile Generations. In: *Conference of Cihan University-Erbil on Communication Engineering and Computer Science*, p. 55, 2017.
15. A. M. Ruby, S. M. Soliman and H. Mostafa. Dynamically Reconfigurable Resource Efficient AES Implementation for IoT Applications. In: 2020 *IEEE International Symposium on Circuits and Systems* (*ISCAS*), IEEE, pp. 1-5, 2020.
16. G. Harcha, V. Lapôtre, C. Chavet and P. Coussy. Toward Secured IoT Devices: A Shuffled 8-Bit AES Hardware Implementation. In: 2020 *IEEE International Symposium on Circuits and Systems* (*ISCAS*), IEEE, pp. 1-4, 2020.
17. K. Aggarwal. Performance evaluation of RC6, blowfish, DES, IDEA, CAST-128 block ciphers. *International Journal of Computers and Applications*, vol. 68, no. 25, pp. 10-16, 2013.
18. R. E. J. Paje, A. M. Sison, R. P. Medina. 2019. Multidimensional Key RC6 Algorithm. In: *Proceedings of the* 3rd *International Conference on Cryptography, Security and Privacy* (*ICCSP 19*). Association for Computing Machinery, New York, USA, pp. 33-38, 2019.
19. A. I. Sallam, O. S. Faragallah and E. M. El-Rabaie. HEVC selective encryption using RC6 block cipher technique. *IEEE Transactions on Multimedia*, vol. 20, no. 7, pp. 1636-1644, 2018.
20. E. O. Agu, M. O. Ogar and A. O. Okwori. Formation of an improved RC6 (IRC6) cryptographic algorithm. *International Journal of Advanced Research in Computer Science*, vol. 10, pp. 34-39, 2019.
21. K. I. Santoso, M. A. Muin and M. A. Mahmudi. Implementation of AES cryptography and twofish hybrid algorithms for cloud. *Journal of Physics: Conference Series*, vol. 1517, no. 1, p. 012099. 2020.
22. V. Vatsala and T. Poongodi. Security and privacy of E-health data using two fish encryption algorithm. *International Journal of Recent Technology and Engineering*, vol. 9, no. 1, pp. 1906-1910, 2020.
23. M. Vedaraj. A hybrid data encryption technique using TWOFish and elgamal for cloud computing. *International Journal of Management, Technology And Engineering*, vol. 8, no. 1473, pp. 1473-1478, 2018.
24. K. Jang, S. Choi, H. Kwon, H. Kim, J. Park and H. Seo. Grover on Korean block ciphers. *Applied Sciences*, vol. 10, no. 18, p. 6407, 2020.
25. L. Sleem and R. Couturier. Speck-R: An ultra light-weight cryptographic scheme for Internet of things. *Multimedia Tools and Applications*, vol. 80, pp. 17067-17102, 2020.
26. ISO. *ISO/IEC 29192-2: 2019: Information Security Lightweight Cryptography Part 2: Block Ciphers*. Geneva, Switzerland: International Organization for Standardization, 2019.
27. J. Song and S. C. Seo. Efficient parallel implementation of CTR mode of ARX-based block ciphers on ARMv8 microcontrollers. *Applied Sciences*, vol. 11, no. 6, p. 2548, 2021.
28. Y. B. Kim, H. Kwon, S. W. An, H. Seo and S. C. Seo. Efficient implementation of ARX-based block ciphers on 8-Bit AVR microcontrollers. *Mathematics*, vol. 8, no. 10, p. 1837, 2020.
29. C. Manifavas, G. Hatzivasilis, K. Fysarakis and Y. Papaefstathiou. A survey of lightweight stream ciphers for embedded systems. *Security and Communication Networks*, vol. 9, no. 10, pp. 1226-1246, 2016.
30. Ianix. *ChaCha Usage and Deployment*, 2021. Available from: https://www.ianix.com/pub/chacha-deployment.html. [Last accessed on 2021 May 27].
31. M. Coutinho and T. C. S. Neto. *Improved Linear Approximations to ARX Ciphers and Attacks Against ChaCha*. IACR Cryptol, p. 224, 2021.
32. P. McLaren, W. J. Buchanan, G. Russell and Z. Tan. Deriving ChaCha20 key streams from targeted memory analysis. *Journal of Information Security and Applications*, vol. 48, p. 102372, 2019.
33. S. M. S. Reza, A. Ayob, M. M. Arifeen, N. Amin, M. H. M. Saad and A. Hussain. A lightweight security scheme for advanced metering infrastructures in smart grid. *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 2, pp. 777-784, 2020.