Converting AES Algorithm to be LWC_AES using LCPoAES for Ease Installation on IoT Devices

1st Khumbelo Difference Muthavhine
Department of Electrical and Mining Engineering
University of South Africa
johannesburg, South Africa
kdmuthavhine@gmail.com

Abstract—Security of the Internet of Things (IoT) and memory constraints are not highly considered during the design and implementation of IoT devices. Little has been done to convert memory constraints on cryptographic algorithms, mostly on AES, for ease of installation on IoT devices. In this study, the mathematical building blocks of Advanced Encryption Standard (AES) were mathematically converted to ease the installation of AES on IoT devices. A novel approach of converting building blocks of AES in such a way its memory was reduced for ease installation on IoT devices and its security strength kept approximately to same or increased was developed. It has been said that a novel approach is converting building blocks of AES, so this approach is named "Lightweight Conversion Process of AES (LCPoAES)". LCPoAES generated a new algorithm called Lightweight Conversion AES (LWC_AES). In this study, the encryption strength of LWC_AES should be approximately equal to or higher than the strength of AES after conversion. Additionally, the memory size of LWC_AES should be much lesser than AES’ memory size. In this study, the focus is to generate LWC_AES for ease installation on IoT devices compared to AES. In this study, LCPoAES was applied to AES to convert and solve the memory constraints of the AES during installation on IoT devices. The novel approach managed to reduce the memory constraint of AES and gave approximately the same strength of encryption as AES and even yielded more confusion to the attacker after LCPoAES was applied.

Index Terms—Internet of Thing (IoT), AES, LCPoAES, LWC_AES, LCPoAES.

I. INTRODUCTION

Security and memory are the main concerns on IoT devices [1] - [7], [21]. The security of information, data, and network should be equipped with a strong cryptographic algorithm like Advanced Encryption Standard (AES) on each IoT devices to achieve identification of users, the confidentiality of data, integrity (correctness) of exact information, and deniability of unauthorized users including connection of unwanted devices [6]. Many organizations are shifting their mode of operation to IoT devices [6] – [9]. Čolaković et al. [6] indicated that security is needed during services requirement of IoT devices than making a profit. Today, IoT devices are needed in many companies for enabling technologies, services and applications, business models, social or environmental impacts [7]. All services given by IoT devices need to be secured by a strong algorithm like AES.

Installing security software or solutions like (AES) and Blowfish algorithms on small devices is a big challenge. Most cryptographic algorithms, like AES and Blowfish, need a huge amount of memory before being installed on some IoT devices. The problem forces many developers and designers of IoT devices to reject most cryptographic algorithms like AES, regardless of security strength. Besides being rejected, some algorithms occupy a high percentage of memory after being installed on IoT devices, which leads to blocking other necessary software, like anti-viruses, to be further installed.

In this study, a novel approach called “LCPoAES” was developed in order to convert AES to a generated new converted algorithm called Lightweight Conversion AES (LWC_AES). LWC_AES and LCPoAES are terms first coined in this study. The definition of LWC_AES should not be confused with the definition of lightweight algorithms, which is an algorithm that is composed of one or few components, operators, or building blocks, which do not involve any conversion from existing (prominent) algorithms like AES [1] - [2] - [3]. This is a novel approach since no one has done or coin a term called LCPoAES in the cryptographic study, except in this study. According to the definition coined in this study, LCPoAES is also be defined as a cyclic procedure of studying, analyzing, and measuring constraints of AES to convert them in such a way that the memory is reduced for ease installation on IoT devices while encryption strength is kept approximately constant or increased compared to AES. The product of LCPoAES is a new converted AES, which is given a name in this study, called Lightweight Converted AES (LWC_AES). The name is coined after conversion of the generated LWC_AES, which has different functionality of encryption compared to an original AES.

The LWC_AES’s definition should also not be confused with the simplified algorithm’s definition, where algorithms are mathematically simplified for learning and understanding of the basics of cryptology [4]. The main aim of generating LWC_AES is to preserve or increase the strength of AES while the memory constraint problem is highly reduced (more than...
Little has been done to reduce the memory constraint of algorithms but not to convert AES to be LWC algorithms. Perhaps mostly is on the development of lightweight algorithms without reducing the memory of existing algorithms like AES to be LWC. Table 1 explains the difference between Lightweight Conversion algorithms like LWC and lightweight algorithms. Refer to Table 1.

### Table 1

| Lightweight Converted Algorithm | Lightweight algorithm | Simplified algorithm |
|--------------------------------|-----------------------|----------------------|
| Lightweight Converted Algorithm (LWC Algorithm) is constructed from or after the existing algorithm is known and has constraints used to prevent constraints but keeping the strength of encryption approximately equal to or more than the strength of the existing algorithm. | Lightweight algorithm is constructed with one or few building blocks. Or one or few operators [1], [2], [3]. Construction is independent to any existing algorithm and any existing algorithm’s strength [1] - [2]. Example one exclusive XOR operator is used as a lightweight algorithm on IoT camera [3]. Without any additional building block nor other operators. Therefore no concern about security strength. | A simplified algorithm is constructed from or after the existing algorithm by simplifying it mathematically for better understanding. Construction is independent to constraints and strength of existing algorithm [4] - [5]. Example, simplified DES algorithm [4] - [5]. |

**A. AES Algorithm**

Advanced Encryption Standard (AES) is developed by Vincent Rijmen and Joan Daemen. The aim to develop AES was to replace Data Encryption Standard (DES) algorithm. AES is a block cipher consists of four main functions, namely: *Substitute Byte* (*SubByte*), *Shift Rows* (*ShiftRows*), *Mix Columns* (*MixColumn*) and *Add Round Key* (*AddRoundKey*) [8] - [10]. Out of these four main functions, three functions have inverses, namely: *Inverse Mix Columns* (*InvMixColumn*), *Inverse Substitute Byte* (*InvSubByte*) and *Inverse Shift Rows* (*InvShiftRows*). *Add Round Key* is the only function without and inverse [11] - [14]. The main functions are used during encryption process and inverses are used during decryption process [8], [10] - [14]. Figure 3 shows the encryption and decryption processes. Refer to Fig. 1.

During encryption process, the first step is *SubByte*. In this step, AES algorithm uses a Substitution-Box (S-Box). S-Box is a look-up table composed of inputs and outputs in number of bytes [10] - [12]. In the *SubBytes* step, each input byte is substituted by another different byte using AES S-Box [12] - [14]. Referring to Fig. 2, suppose that input byte is c000 in hexadecimal notation, c0 = x which is row and 00 = y which is column. Checking from AES S-Box given on Fig. 2 where x and y intersect, c000 is substituted by ba. During decryption process, inverse AES S-Box is used. When inverse AES S-Box is used, then the step is called *InvSubBytes*, which is a step number three during decryption process. The *InvSubByte* is directly an inverse of the *SubByte*. Referring to Fig. 2. Again, suppose ba is the input byte of the *InvSubByte*. The *InvSubByte* substitutes byte b00a by assigning b0 = x and 0a = y, then, from AES inverse S-Box, the intersection of b = x and a = y gives an output of c0. ba is substituted by c0 in hexadecimal notation. AES converts a string of plaintext into 4×4 matrix; after substitution, the matrix is called the first state of AES. Note that a state is referred to output of each function of AES. Another critical function that operates the state is *MixColumns*. The mixing is the multiplication process of matrix rows and columns. Each 8-Bits entity of a row is multiplied by each 8-Bits entity of the state column using matrix transformation [12] - [13]. In simple terms, each row of matrix transformation is used to multiply every column of the state [8], [10]. The outputs of multiplication are XORed to produce a new state [10] - [14]. The reverse transformation of *MixColumn* is called (*InvMixColumns*). *InvMixColumns* is implemented during decryption process [11] - [13]. The size of states is always the same size, which is 4×4 matrix. Refer to Fig. 3.
The final function of AES during encryption process is called Add Round Key, also known as AddRoundKey. Unlike other functions, the AddRoundKey does not have an inverse. The procedure of the AddRoundKey is applied to both the encryption and decryption process. During the AddRoundKey operation, either the state generated after MixColumns or InvMixColumns are simply XORed with the state of key [8], [10], [14]. The process is noticed as a column bitwise between the four 8-Bytes of a state column and one of 32-bits round key. For detail, refer to Fig. 4.

AES supports three original sizes of keys, namely: 192-Bits, 128-Bits, and 256-Bits. All subkeys are generated from an original key; the method of generating subkeys depends on the size of the original key. Subkeys are used during encryption and decryption processes [10] - [14]. The mathematical steps explaining the generation of subkeys are given in Fig. 5. Refer to Fig. 5.
B. Problem Statement

The memory constraints on the IoT devices are very narrow, along the lines that some strong and well-known cryptographic algorithms like AES cannot be easily installed on IoT devices for the security of data and information during communication. Consequently, algorithms are rejected if they exceed the memory limit of IoT devices [12], pp. 32], [15], pp. 252-260], [15], pp. 5, 17]. A well-known cryptographic algorithm like AES seems to require more memory occupation as compared to the memory allocated on IoT devices [16], pp. 1], [17], pp 1, 6], [19], pp. 1, 6, 17], [18], pp. 5, 17]. This problem of memory constraints can harm the entire security of the IoT system if it is not considered properly. Little has been done to convert AES to be LWC AES to avoid memory constraints [16], pp. 1], [17], pp. 1], [20], pp. 1, 6, 17], [18], pp. 2]. The novel approach, called LCPoAES was applied on AES to develop a new converted algorithm called LWC_AES in order to reduce memory constraints for ease installation on IoT devices.

II. RELATED WORK

Čolaković and M. Hadzialic [6] indicated that the exponential growth of the IoT devices attracted many companies to shift their businesses into the involvement of IoT. The main concerns were security and privacy. Abiodun et al. [7] indicated that security and privacy were the primary and continuous concerns on IoT devices as many challenges and developments were still upcoming on IoT devices, and new, strong algorithms like AES had to be developed for compatibility of installation on IoT devices.

Muthavhine and Sumbwanyambe [21] indicated that IoT devices were substantially tiny in size and memory allocation. Algorithms like AES were generically rejected due to memory constraints required before installation. Usman et al. [1] also indicated that many cryptologists, developed lightweight algorithms to resolve memory constraints problems, but lightweight algorithms nevertheless might compromise the security of IoT devices. The continuation of developing lightweight algorithms is still going on, even many researchers are discouraging.

Shifa et al. [3] developed a lightweight algorithm called EXPER, which consisted of five simple mathematical stages with one function applied on many rounds. On each round, the XOR substitution operator with shift operation was performed on ciphertext generated from the last rounds. EXPER was compared to AES when the speed and encryption strength were considered; EXPER gave good results on speed compared to AES since EXPER was a lightweight algorithm. Though EXPER was faster compared to AES, the encryption strength of AES was much better than EXPER. Shifa et al. [3] concluded that the speed of an algorithm did not depend on the strength of that algorithm. The results indicated that EXPER could not secure data than AES.

Buchanan and Asif [2] also indicated that lightweight algorithms were easily attacked by simple mathematical methods such as side-channel attacks. Furthermore, Buchanan and Asif
[2] indicated that there was a need for AES on IoT devices. The main concern was the security due to the lack of memory required by IoT devices before the installation of strong and prominent cryptographic algorithms [7]. Some researchers used lightweight algorithms instead of Lightweight Converted Algorithms. Most lightweight algorithms were not strength-tested compared to the well-known strong algorithms [2].

Shifa et al. [3] used one mathematical operator called XOR as an encryption algorithm for camera. Abiodun et al. [7] used lightweight algorithm called by-hop encryption to try to solve security and memory constraint issues. The encryption strength of by-hop and Xor algorithms were not compared to any strength of known algorithms like AES and others [10]. Usman et al. [1] proposed a lightweight algorithm called Secure IoT (SIT) without analyzing security strength compared to any strong and known algorithms.

Buchanan et al. [2] analyzed few algorithms including AES, RC5, ECC and CLEFIA. The results indicated that most algorithms suffered memory constraints and a disadvantageous lot of IoT devices due to resource-contraints like memory and power consumption [2]. Čolaković et al. [6] confirmed there was a need of conducting a study of reviews on IoT challenges, including converting algorithms that had high memory constraints. This was a gap that highly needed to be addressed for IoT security. Abiodun et al. [7] indicated there was a gap to put suitable algorithms on all architectural layers of IoT. In this study, the novel approach, called LCPoAES, was implemented on AES. After implementation, a new algorithm was generated and named LWCAES, which reduced memory constraints by more than 10% compared to AES.

III. RESEARCH METHODOLOGY

In this study, a novel approach called LCPoAES was implemented in C++ programming to reduce memory constraints found on AES during installation on IoT devices. Refer to Fig. 6. Installing AES on some IoT devices is very cumbersome when IoT devices have a shortage of memory storage. The method of LCPoAES was found to be working more successfully. This method also managed to reduce the memory constraints and preserve approximately the encryption strengths of AES. AES was converted to generate a newborn algorithm called LWCAES, which had less memory constraint and kept approximately the same encryption strength compared to an original AES. LCPoAES was conducted as follows:

1. AES was collected from IoT devices.
2. AES algorithms was validated, tested and verified using test vectors given by Kak [5]. Other vectors were given by Federal Information Processing Standards (FIBS) [8], pp. 35.
3. After verifying the validity of AES, LCPoAES was applied on AES to check if the memory was reduced by less than 10% compared to an original AES, then LCPoAES was applied again until more than 10% of memory reduction was achieved. Memory was measured using Memory C++ Software Tool.
4. After reducing the memory, the strength of LWC_AES was measured using the Avalanche effect. If the strength is approximately equal to or higher than an original AES, then a new LWC_AES was generated and accepted to be installed on IoT devices. Otherwise, step 3 is repeated.

The LCPoAES was applied to AES. Fig. 6 showed the flowchart of LCPoAES implemented on AES. Fig. 6 was chosen as a comprehensive explanation of how to conduct LCPoAES. Refer to Fig. 6.

![Flowchart of Research Methodology Using LCPoAES](image)

A. Conversion of AES to LWC_AES

On AES, 8×8 AES S-Box and it’s 8×8 inverse were removed from an original AES algorithm. Refer to Fig. 2 to revisit both AES S-Box and its inverse. Both AES S-Box and its inverse were found to be the functions or building blocks that limit AES to be installed on IoT devices due to memory constraints, since they were assigned as arrays of 256 elements each. Each element was found containing 8-Bits. AES S-Box and its inverse were removed and substituted with the input values shifted once, and XORed with the same inputs. Again, AES was found using a function called 256 Recon array, defined as $Recon[input]$. Refer to Fig. 5 to revisit $Recon[input]$ explanation. $Recon[input]$ was substituted by mathematical expression defined as $(input \ XOR \ (4 – input))$. The number of rounds was reduced from 10 to 8 rounds, then the memory of AES was reduced by more than 10%. Refer to Table VIII, IX and X. The strength became approximately the same compared to an original AES algorithm, since differences were between the range of (± 5%). Refer to Table VI and
VII. After conversion, the results gave an expected encryption strength and an algorithm that occupies less memory, and an expected new LWC_AES was generated which satisfies the criteria of LCPoAES. Then the new LWC_AES algorithm was accepted as the one recommended to replace the original AES.

IV. Measurement Tools

Encryption strength and memory were measured using tools explained in the next subsections.

A. Avalanche Effect Tool

There are many tools to measure the strength of cryptographic algorithms. Most of them are mathematical and software programs. In this study, Avalanche Effect was used as a tool to measure the strength of AES and LWC_Algorithm. Avalanche Effect method was given, explained by Verma and Sharma [9] and also discussed by Ariffin and Ashawesh [10].

B. Memory C++ Software Tool

There are many procedures and tools to measure the memory of algorithms. In this study, a C++ tool was developed because of an accuracy that was very high and gave an exact value of Bytes without rounding it to the nearest KiloBytes compared to other available tools.

V. Results Found

In this study, the results found when LWC_AES was compared with existing AES regarding the strength and memory were discussed in this section. Table II showed the results of the Avalanche Effect using both key and plaintext on AES in order to find strength. The keys and plaintexts were the same when the experiment was conducted.

The same procedure was applied on LWC_AES. The same inputs used on AES were also used on LWC_AES. Table III showed Avalanche Effect using key and plaintext on LWC_AES.

The memory of AES and LWC_AES were measured using a software-developed tool. Table IV showed the memory of encryption, decryption, and full algorithm. Refer to Table IV.

Memories of LWC_AES were also measured. The size of keys and plaintext inputs used were the same as on AES when memory was measured. Table V showed the results of memories of LWC_AES. Refer to Table V.

VI. Results Analysis

The analysis of the results was conducted to check if LCPoAES was desirably achieved regarding strength and memory constraints. The strength of LWC_AES should be approximately equal to or more than the strength of AES. If the strength difference of LWC_AES is between the range of (±5%), therefore, LWC_AES and AES have approximately the same strength, then LCPoAES is achieved. If the strength difference of LWC_AES is less than (-5%), then LCPoAES is totally not achieved. If the strength difference of LWC_AES higher than (5%), then LCPoAES is highly achieved. That was also even confirmed by the definition of Avalanche Effect. To get the percentage difference of LWC_AES, the Avalanche Effect of AES was subtracted from LWC_AES using Equation “(1)”. The percentage difference of LWC_AES is denoted by $\Delta$W_AES. Refer to the Equation “(1)”.

$$\Delta W_{AES} = (W_{AES} - W_{AES \, strength})\% \quad (1)$$

The values of $LWC_{AES}$ and $AES \, strength$ were taken from Table II and Table III respectively.

For instance, considering the $Strength \, Difference \, of \, LWC_{AES}$ of Avalanche Effect applied using key was calculated using Equation “(2)” and the results found. Note that Equation “(2)” was derived from Equation “(1)” using Table II and Table III to get real values of strength substituted in the Equation “(1)”. Refer to Equation “(2)”.

$$\Delta W_{AES} = (49.7680 - 50.2075)\% = -0.4395\% \quad (2)$$
The values found from Equation “(2)” was put on Table VI. Therefore, the strength of LWC_AES and AES are approximately equal, since the difference is in the range of (± 5%). Then, LCPoAES is achieved. If the strength difference was zero percent, then LWC_AES and AES strength were exactly equal.

The analysis of memory results was conducted using LCPoAES if it was achieved or not. If the memory of LWC_AES was less than 10% compared to AES memory after LCPoAES applied, then LCPoAES was not achieved at all. If the memory of LWC_AES was between 10% and 30% compared to AES memory, then LCPoAES is achieved. If the memory of LWC_AES was greater than 30% compared to AES memory, then LCPoAES was highly achieved. To get the percentage memory of LWC_AES compared to the AES, Equation “(3)” was used. Equation “(3)” helped to calculate the percentage of memory constraints’ reduction when AES and LWC_AES were compared. If nothing was reduced, then the memory of AES and LWC_AES were considered to be equal, then the LWC_AES percentage would be zero percent (0%). If that happened, that showed that Lightweight Conversion was not achieved at all. The different percentage memory of LWC_AES and AES was denoted by ▲LWC_AES. Refer to Equation “(3)”.

\[
▲LWC_AES = (100 - \left( \frac{LWC_AES \text{ memory}}{AES \text{ memory}} \right) * 100)\% \tag{3}
\]

The values of LWC_AES memory and AES memory were taken from Table IV and Table V respectively.

For instance, considering LWC memory (percentage) when AES encryption memory was considered, then Equation “(4)” derived from Equation “(3)” was used when real values were substituted from able IV and Table V. Refer to Equation “(4)”.

\[
▲LWC_AES = (100 - \left( \frac{4760 \text{ Bytes}}{9080 \text{ Bytes}} \right) * 100)\% = 47.5770\% \tag{4}
\]

Therefore, the percentage of memory reduction was highly achieved since it was more than 30%. Equation “(1)” was used to calculate the results on Table VI. On LWC_AES and AES, the results show achievement of getting approximately the same strength which is (<± 5%) after Lightweight conversion was conducted. That was regarded as achievement according to the definition of LCPoAES. Refer to Table VI.

Table VII was constructed using Equation “(1)”. On LWC_AES, the results showed that approximately the same strength was achieved since the percentage was (<± 5%) after LCPoAES was conducted. That was regarded as an achievement according to the definition of LCPoAES. Refer to Table VII.

**Table VII**

**Analysis of Strength Difference when Avalanche Effect using Plaintext was Considered**

| Symbol of Procedure Name of Algorithms | Strength Difference considering Avalanche Effect using Plaintext (in Percentage) | Achievement Remarks of LCPoAES when strength is considered |
|----------------------------------------|---------------------------------------------------------------------------------|----------------------------------------------------------|
| ▲LWC_AES                              | 2.2339                                                                          | Achieved                                                 |

Table VIII was constructed using Equation “(3)”. On both encryption sides of LWC_AES and AES, the approach sufficiently managed to reduce AES memory constraints by more than 30%. According to Equation “(3)”, this is regarded as highly achieved since more than 30% of memory reduction was achieved on LWC_AES. Refer to Table VIII.

**Table VIII**

**Memory Reduction when Encryption Sides of Algorithms were Considered**

| Symbol of Procedure Name of Algorithms | Memory Constraint Reduction on Encryption (in Percentage) | Achievement Remarks of LCPoAES when Memory on Encryption is considered |
|----------------------------------------|----------------------------------------------------------|------------------------------------------------------------------------|
| ▲LWC_AES                              | 47.5770                                                  | Highly Achieved                                                          |

Table IX was also constructed using Equation “(3)” on the decryption side of LWC_AES, the approach method managed to reduce memory of AES by more than 30%. According to Equation “(3)”, that was regarded as highly achieved since more than 30% of memory reduction was found on LWC_AES.

**Table IX**

**Memory Reduction when Decryption Sides of Algorithms were Considered**

| Symbol of Procedure Name of Algorithms | Memory Constraint Reduction on Decryption (in Percentage) | Achievement Remarks of LCPoAES when Memory on Decryption is considered |
|----------------------------------------|----------------------------------------------------------|------------------------------------------------------------------------|
| ▲LWC_AES                              | 56.5213                                                  | Highly Achieved                                                          |

Table X was also constructed using Equation “(3)”. On full algorithms, results showed that memory was reduced by more than 30%. According to Equation “(3)”, that was regarded as highly achieved since more than 30% of memory reduction was obtained on LWC_AES.
TABLE X
MEMORY REDUCTION WHEN FULL ALGORITHMS WERE CONSIDERED

| Symbol of Procedure Name of Algorithms | Memory Constraint Reduction on Full Algorithm (in Percentage) | Achievement Remarks of LCPoAES when Memory on Full Algorithm is considered. |
|----------------------------------------|-----------------------------------------------------------------|--------------------------------------------------------------------------|
| LWC_AES                                | 47.8909                                                         | Highly Achieved                                                           |

VII. CONCLUSION AND FUTURE WORK

It has been proven in this study that LCPoAES method can reduce memory constraints by more than 40% found on AES for ease installation on IoT devices by converting AES to LWC_AES, while encryption strength is kept approximately equals to the strength of the original AES, since differences were between the range of (± 5%). This novelty of LCPoAES managed to reduce memory constraint problems on AES and IoT devices. It has to be noted that if memory constraint problems are reduced by using LCPoAES that does not mean time, speed, and power constraint problems are automatically solved. Future work will be done using LCPoAES to solve other constraints like time, speed, and power/energy consumption.

REFERENCES

[1] M. Usman, I. Ahmed, M. I. Aslam, S. Khan and U. A. Shah, “SIT: A Lightweight Encryption Algorithm for Secure Internet of Things,” International Journal of Advanced Computer Science and Applications, 2017.
[2] W. J. Buchanan, S. Li and R. Asif, “Lightweight cryptography methods,” Journal of Cyber Security Technology, pp. 187–201, 2018.
[3] A. Shiha, M. N. Asghar, N. Batool and M. Fluery, “Efficient Lightweight Encryption Algorithm for Smart Video Applications,” FIT-2018 Conference, 2019.
[4] Geeks for Geeks, “Simplified Data Encryption Standard Key Generation,” geeksforgeeks, 2021.
[5] A. Kak, “AES: The Advanced Encryption Standard,” Engineering Pudue, pp. 1–92, 2021.
[6] A. Čolaković and M. Hadzialic, “Internet of Things (IoT): A Review of Enabling Technologies, Challenges, and Open Research Issues,” Computer Networks, pp. 11–39, 2018.
[7] O. I. Abiodun, E. O. Abiodun, M. Alawida, R. S. Alkhawaldeh and H. Arshad, “A Review on the Security of the Internet of Things: Challenges and Solutions,” Springer, 2021.
[8] Federal Information Processing Standards (FIPS), Announcing the Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, 2001.
[9] R. Verma and A. K. Sharma, “Cryptography: Avalanche effect of AES and RSA,” International Journal of Scientific and Research Publications, 2020.
[10] N. A. M. Ariffin and A. Y. A. Ashawesh, “Enhanced AES Algorithm Based on 14 Rounds in Securing Data and Minimizing Processing Time,” The Electrochemical Society, pp. 1–9, 2021.
[11] R. Thomas, “Advanced Encryption Standard (AES): What It Is and How It Works,” 2020.
[12] A. Berent, “Advanced Encryption Standard by Example,” AB Software Development, 2019.
[13] W. Stallings and L. Brown, “Computer Security: Principles and Practice,” Information Security, Spring 2021, 2021.
[14] A. M. Abdullah, “Advanced Encryption Standard (AES) Algorithm to Encrypt and Decrypt Data,” pp. 1—12, 2017.
[15] A. Haroon, W. Naeem, M. A. Shah, M. Kamran, Y. Asim and Q. Javaid, “Constraints in the IoT: The World in 2020 and Beyond,” International Journal of Advanced Computer Science and Applications (IJACSA), pp. 255–271, 2016.
[16] G. Mustafa, R. Ashraf, R. Ashraf, M. Ayzed, M. Muhammad, A. Mirza and et al., “A Review of Data Security and Cryptographic Techniques in IoT Based Devices,” The 2nd International Conference, pp. 1–8, 2018.
[17] K. Deshpande and P. Singh, “Performance Evaluation of Cryptographic Ciphers on IoT Devices,” Arxiv, pp. 1–6, 2018.
[18] M. El-Hajj, A. Fadlallah, M. Channoun and A. Serhrouchni, “A Survey of Internet of Things (IoT) Authentication Schemes,” Sensors 2019, pp. 1–43, 2019.
[19] A. A. Milad, H. Z. Muda, Z. A. B. M. Noh and M. A. Algaet, “Comparative Study of Performance in Cryptography Algorithms (Blowfish and Skipjack),” Journal of Computer Science, pp. 191–1197, 2012.
[20] S. Monir, “A Lightweight Attribute-Based Access Control System for IoT”, University of Saskatchewan, pp. 1–74, 2017.
[21] K. D. Muthavhine and M. Sumbwanyambe, “An Analysis and a Comparative Study of Cryptographic Algorithms used on the Internet of Things (IoT) Based on Avalanche Effect,” International Conference on Information and Communications Technology (ICOIACT), 2018.