International Journal Of Engineering Research And Development
e- ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com
Volume 21, Issue 4 (April 2025), PP 67-74

An Efficient Real-time Cryptography Algorithm for IoT Devices

Fagbohunmi Griffin Siji [1] Uchegbu Chinenye. Eberechi [2]
[1] *Computers Engineering Department Abia State University Uturu Nigeria.*
[2] *Electrical and Electronic Engineering Department, Abia State University Uturu Nigeria.*

Abstract:
The use of cryptographic algorithms are to enable secure transmission of data between any two nodes on a network. The mechanism used by cryptographic algorithms where their action is based on processing of data blocks make them unsuitable to be deployed in IoT devices network This is because IoT devices are mostly embedded any thus energy and memory constrained. The aim of this paper is to design a cryptographic algorithm that has low memory requirement and real time ability. The algorithm is based on a variant of the AES cryptographic algorithm specifically designed for memory constrained IoT devices with additional features such as real time operation. The additional feature of real time operation is to enable the algorithm to be used in most IoT network system where response is usually needed in real time. The algorithms also makes use of threads that enable certain operation to be suspended if they are not immediately needed for the task at hand. This capability makes the algorithm to optimize the use of the IoT device's RAM for multitasking operations. The design proposed in this paper employs an AES algorithm using an updating process comprising a cipher key of 10KB of flash and 780 bytes of RAM with a real-time response of about 13.5µs

Keywords*: Computer Network Security, Cryptography, Embedded Systems, Internet of Things.*

Date of Submission: 02-04-2025 Date of acceptance: 12-04-2025

I. INTRODUCTION

Internet of things (IoT) comprises of connected groups of embedded devices, each having microcontrollers, microprocessors, specialized software and a number of sensors. These IoT devices always operate on the internet to transmit data to one another (Gehrmann and Gunnarsson, 2019). IoTs are responsible for providing distributed resources and services which are shared amongst themselves to different organisations, home or sites requiring them. It is therefore possible to send data to any device or anyone using IoT network (Latif et al.2020). The embedded devices have the similar functionality as the computer, however they are small as they are normally inside another device. They also have specialized functions with varieties of sensors for receiving different types of data and converting them to electrical signals using their transducers., these makes them different from general purpose computers (Rosero-Montalvo et al., 2021). With their varieties of sensors, it is therefore possible to get a large volume of data, from which certain information or knowledge can be adduced using machine learning algorithms for a better and effective utilization in real life scenario (Islam et al., 2019). Despite these good functionality of IoT devices, there is the need to protect them from adversarial threats and attacks. It therefore becomes imperative to protect IoT networks especially when there output can affect human life or industrial processes. Standard security techniques such as cryptography algorithms with end to end signatures must then be employed to safeguard IoT networks (Khan et al., 2021). Due to the amount of overhead required in cryptography algorithm processing, they usually involve a high volume of information (blocks of data) therefore, their compilation requires a lot of resources which cannot be done by the IoT devices due to their memory and computationally constrained nature. Moreover the block size of the IoT devices is much smaller than that used in cryptography algorithm, these leaves spaces in the cryptography blocks leading to wastage of memory and inefficient resource utilization.

Cryptography algorithms will therefore result in overheard for the IoT devices thereby causing increased computational requirement which cannot be carried out by the memory and computationally constrained IoT devices. It should be realized here that IoT devices can only compile small volumes of data due to its small RAM flash sizes, limited power consumption, processor speed and real time capabilities (Prakash et al., 2019). IoT devices uses different types of wireless network protocols to connect themselves together with a low rate of transfer to the cloud, their low transfer rates if combined with the cryptography algorithm to be compiled cause low response time for real time applications (Shah and Venkatesan, 2018).

There is a new technique which involves the use of Trusted Execution Environments (TEE) to leave out the cloud computing side's users. Consequently microcontrollers with a trusted platform module (TPM) are being designed to isolate certain software from the other system operations. It can be assumed that a microcontroller is trusted since it runs only one application and therefore does not have multi-user security issues (Szymanski 2017). With this assumption, the adversarial threats that can affect IoT devices are those external to it, which means that the communication channel is susceptible to attack since many devices transmit data on the wireless communication channel using the 802.11 protocols that are vulnerable to many weaknesses (Naru et al., 2017). With this in mind, IoT devices are now designed to overcome weakness in security that is due to authentication. In other words IoT devices are designed to authenticate the source of data, therefore if individual devices on the network is secured the entire IoT network is also secured. An example of microcontroller with Trusted Execution Environment (TEE) is the Arduino IoT products which has a crypto chip incorporated with specific libraries for the connection of cloud services (Sung et al., 2018). The main function of the crypto chip is to connect the IoT devices to the cloud. Generally in IoT devices wireless sensor networks, the devices repeatedly transmit data between themselves while the IoT acting as the central node in the star topology communicates the aggregated data to the cloud (Fotovvat et al., 2023), in this way, the information sent to the cloud will be secured since the individual IoT devices in the network has incorporated in them the Trusted Execution Environment. (TEE). However this mechanism cannot protect against malicious damage to the IoT devices or the injection of spam onto the network.

The inclusion of cryptography application in IoT devices increase the overhead resulting in an increase in latency which invariably affects the device's response time. In this type of situation, it is important to increase the algorithm efficiency by implementing parallel processing where a number of instructions which do not depend on one another can be executed simultaneously. This can be done using a rea time operating system where redundancy are inserted into the algorithm so as to increase the likelihood of running the program function in appropriate time slots. This will increase the efficiency of the program execution, the aim of inserting these redundancies is to simultaneously run the crypto library together with checking the remaining battery lifetime, while the sensors are being calibrated before transferring data to other IoT devices. With these method one can be sure that the data been transferred are authenticated.

The aim of this paper is therefore to propose an embedded cryptography algorithm using real time operating system with optimum memory utilization. With this method, an encrypted message will be transmitted in the IoT wireless sensor network within the memory and computational capability of the IoT devices with enough memory to compile the program. In order to achieve this, it was assumed that the IoT devices and sensors were trusted because of the Trusted Execution Environment (TEE) embedded in their microcontrollers while only the wireless communication medium channel (802.11 n) is not.

The average RAM and flash size needed to execute the algorithm for the crypto library of AES 128 is 8 bits data bus size, this is compatible with the Arduino specification. The application used above was deployed for the design a low latency response time software, by optimizing the parameters such as size of algorithm and overhead. In order to further secure the wireless network, a function to update the cryptographic keys was embedded in the TEE, this is necessary to secure the data transferred by the IoT devices. The size of the Real Time Operating System used in implementing the AES cryptography algorithm was 0.8KB of RAM and 10KB of flash, while the response time of the software was 075μS.

The remainder of the paper is organized as follows, a review of related works on implementation of IoT cryptographic algorithm was discussed in section 2, the experimental setup was described in section 3, where the requirements assumed for both RAM and flash size was designed. Section 4 discussed the result and analysis of the experiments carried out, here the schedule of the real time function and optimization used was designed. Section 5 concludes the paper.

II. REVIEW OF RELATED WORKS

Cryptographic algorithm of minimal size for IoT devices are currently new in the research field, this is not unconnected with the type of information being gathered by sensors which often times are related to human lives and other sensitive data in industries. So a lot of care and relevant advancement must be available before delving into such sensitive areas. The advantages of wireless communication is not without security challenges, and the IoT device being memory and computationally constrained cannot use the traditional cryptography algorithms. Therefore cryptographic algorithms which can be deployed in hostile environments and also consume little memory must be designed. According to (Khan et al., 2022); (Fotovvat et al., 2023), their work presented an exhaustive comparisons of cryptography algorithm with comparatively small sizes deployed in different environmental conditions Their result shows that the limited memory size of the IoT devices did not permit security of transmitted data in hostile environment. According to (Guo et al., 2019), the researchers proposed a reduction in the complexity of the block encryption while verifying their advantages on complex task like image encryption. In the work of the researchers in (Jalaly and jalaly 2019), a new chaining encryption algorithm was

presented for LPWAN IoT network while also providing statistical properties of the method. The aim of the work was to empirically give data involving their memory and power consumption. In the work of (Dang et al., 2022), an encryption algorithm for LORA wireless sensor network communication was presented. In the works of (Naru et al., 2017); (Prakash et al., 2019); (Ramesh and Govindarasu, 2020); (Prakash et al., 2019); (Hijawi et al., 2023), a new low memory design of cryptographic algorithm were proposed, where the method of traditional encryption algorithm were modified to be conformable to the memory constrained wireless sensor network and provide immediate response to query.. According to the works of researchers in (Gope and Hwang 2016); (Islamet al., 2019), they employed FGPA and dual ARM processors in the design of low memory encryption algorithm suited for IoT devices. In the work of (Tsai et al., 2018), the researchers designed an AES 128 secure channel using the LORAWAN communication channel for IoT devices.

The works reviewed in this paper described the design of cryptographic algorithm for IoT networks, however none of the works considered the option of transmitting data to the cloud. In recent works the cloud are considered an extension of IoT network

Due to this development, cryptography algorithm must be designed to be conformable to this trend i.e. aggregated data from the IoT network must be transmitted to cloud for enhanced security and compatibility with current practice. For this reason, an IoT communication channel must be secured so as not to transmit corrupted message to the cloud. Also the response time of the IoT network must be instant so as not to compromise its purpose as well as the cryptography algorithm consuming little memory space.

III. Setup of Experiment

In this section, the Advanced Encryption Standard will be described together with its configuration The assumptions made is then presented together with part of the network that is trusted and which components are not

3.1 Advanced Encryption Standard

The Advanced Encryption Standard uses a symmetric block cipher for its encryption and decryption. In this encryption, a block of plaintext is encrypted to give a block of ciphertext having the same size as the block of plaintext. The encryption key sizes supported by AES are 128, 192 and 256 There are five modes supported by AES, these include (i) Electronic Code Block (ECB) , (ii) Cipher in the block chaining mode (CBC mode), (iii) Cipher Feedback mode (CFB mode), (iv) Output Feedback mode (OFB mode) and (v) Counter mode (CTR mode) (Serra et al.,2021). Among these five modes, the CBC mode is the most widely used because it encrypts the plaintext block to the ciphertext block while successive encryption blocks are XORed with the plaintext to get the ciphertext, this is done until the last plaintext block is encrypted. The security of the CBC mode stems from the fact that when the same plaintext is encrypted using this method at different times, it will obtain different ciphertext due to the different encrypting keys used (Borges al., 2023).

.3.2 Current Cryptographic Libraries

A good number of Cryptographic libraries use the AES for the encryption and decryption of large datasets. These libraries are not however compatible to the 8-bit/16-bit architecture of the IoT devices. In fact very few research work has been done on cryptographic algorithm using AES on IoT devices. Among the few works is that by (Rosero-Montalvo et al., 2021) where data received from some environmental conditions were encrypted using AES into a 16-bit data bus. In this work a real time cryptographic system is designed using two well-known libraries, Library 1 (Van Heesch, 2018) and Library 2 (Landman 2022)

3.3 Design of AES Encryption for IoT Devices

In order to design an AES encryption/decryption algorithm for IoT devises, the following simplification will be made, (i) It is assumed that the data acquired by the IoT devices from the sensors of other ioT devices are not corrupt or tampered with. (ii) The software used by the IoT devices does not contain virus. (iii) The program in the RAM are customized so as to run the computationally involved AES algorithm. (iv) The network topology of the IoT network is of star type where data is aggregated at a particular IoT device acting as the hub. (v) The IoT at the hub is equipped with the encryption keys prior to deployment and (vi) the IoT devices receive the 16 bytes data from the AES encryption algorithm in series of two consecutive transmission

It is a well-known fact that AES is a synchronous cipher algorithm where both sender and receiver must know the encryption/decryption key before initiating it. It is therefore a complex task to update the cipher keys, however this can be done by generating two random numbers, where the first determines the number of times the encryption key is used. After using the encryption key for that amount of time, the next cipher key will be generated by XORing the cipher key with the next random number. The position of the next random number (i.e. number of times the cipher key is repeated) in the ciphertext is only known to the IoT device at the hub of the star topology. This means that only the IoT device at the hub can update the cipher key. The cipher key update is stored in the

EEPROM so as to make the process unknown to other IoT devices on the network. In order to reduce the computation in the RAM, the Rijndael S-box is stored in the flash. . At the end, the real time operating system is used to reduce the response latency thereby using symmetric multiprocessing into the IoT device.
An IoT system used as an autonomous robot in greenhouses is shown in figure 1, it acquires various types of data from its environment. LT-35 temperature sensor is used to sense the temperature of its surroundings, VEML6075 was used for the measurement of Ultraviolet rays' level. Data were also gotten from the robot motors so as to determine its velocity and gyration

. Figure 1 Robot in greenhouse environment

Operating Principle of the Autonomous Robot: The following series of operations are performed by the autonomous robot, they are: Motor control, reception of motor parameters, battery status, receiving environmental data, data encryption and transmitting information to neighbouring IoT devices Of all these steps, this paper only concentrates on the data encryption. The processes involved in the encryption of data are: (i) receiving plain text data input from the environment, (ii) converting the data into 16 hexadecimal bytes, (iii) Rotation of data, (iv) substitution of bytes (v) x-box and shift rows (vi) Round key substitution comprising of circular byte left shift and byte substitution, (vii) XORing the cipher text between either the rotation of data and round key substitution

IV. Results and Analysis

Here, the amount of memory used by each of the library used in this work is described. Together with the description of the real time operating system adopted by the encryption/decryption algorithm. Finally the amount of memory used by the RAM, flash and the real time response time was given

4.1 Used Memory

In order to calculate the resources required to compile the AES algorithm used fin this work, the memory consumption and response time for lib 1 and lib 2 were computed. The algorithm was designed using almost all the registers available in the RAM with optimization. The correct size for each register variable was then defined together with some constant variable size registers. The code blocks for making the threads were then defined, all these were inserted in the real time operating system (AES-RTOS v4) of the IoT devices. The response time to any query and the amount of memory consumed were determined at compile time of the cryptographic program. The keys used for encryption was stored in the ROM of the IoT device acting as the central hub so as not to be accessible to any other IoT devices on the network,

4.2 The Design of the Real Time Operating System

The advanced encryption system defined for the IoT network comprises of four threads, these are (i) plain text consisting of 16 hexadecimal bytes, (ii) data rotation which defines the byte substitution and number of rows to be shifted, (iii) round keys which defines the number of bytes for the cyclical shift bytes and byte substitution (x-box) and (iv) the cipher text which was obtained by XORing the data rotation key and the round key. It should be noted here that the number of consecutive keys required to encrypt the plain text is equal to the number of the cyclical bits. As said earlier, these keys are stored in the ROM of the IoT device acting as the central hub of the star topology network. In order to implement this design, the IoT devices acquire the plaintext from its environment using their sensors, these are stored in two consecutive 8-bit hexadecimal in the IoT data bus as it can be recalled that the data input from the AES encryption is 16 bit hexadecimal. In order to update the encryption keys, the following was done, The initial encryption key is stored in the ROM of the IoT device acting as the central hub. It should be realized here that the central node has the encryption keys of all the IoT devices on the network. Any IoT device on the network can transfer its sensed data to the central hub, for this transfer to be enabled, the encryption key flag on both the IoT device and the central hub must be high. It should be noted here that the encryption key flag of only one IoT device can be in high state, all other IoT device on the network will

have their encryption key file on low status. Now when the encryption key flag of any particular IoT device is high, it computes its first random number which will be used to decide the number of times the ciphertext will be decrypted using the same cipher key,. After completing the round of ciphertext / cipher key combination, an update will be performed on the cipher key by XORing the ciphertext with the second 8-bit number. The second 8-bit number is used to encrypt the plaintext using data rotation and then forwarded to the central hub IoT. In order to further update the cipher key, the random number that was generated the first time will again be XORed with the next plaintext to generate a new round of ciphertext. The ioT advanced encryption system generates the first random number as long as a hard reset is done irrespective of the flag status of the IoT device. A comparison between the standard Advanced Encryption Standard and the real time (AES-RTOS) encryption designed for the IoT network is given in Table 1. The used RAM, Flash and the time of response of each library is shown in Table 1. From the table it is realized that lib 1 consumes a lot of RAM, while lib 2 takes much time to perform text encryption. The library (AES V1 AND V2) defined for the IoT network, requires a much reduced resources for computation, coupled with a reduced response latency. On the other hand the flash memory consumed by Advanced Encryption Standard libraries designed for the IoT devices (lib AES-RTOS) is more than for the Standard AES The flowchart for the AES-RTOS is shown in figure 2

This is because the thread in the lib AES-RTOS consumes 65 bytes. Overall, the real time operating system designed for the IoT network manages the system's resources more efficiently and the fact that the threads has priorities for enabling them make their use in the encryption only when necessary. This means that threads based on their priorities may be enabled or disabled to free up more spaces for algorithm execution. The libraries used for the IoT encryption can be found in AES-RTOS lib. Figure 3 shows the pseudocode of the AES-RTOS.

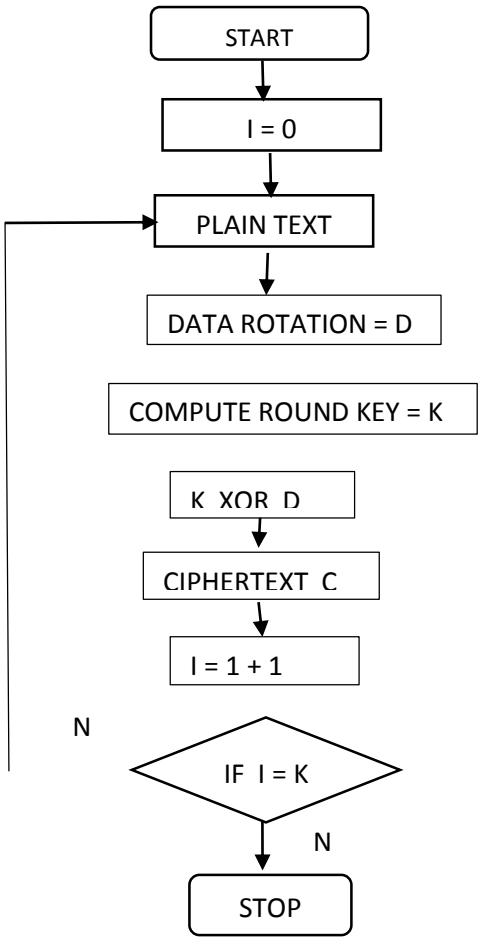

FIGURE 2 IMPLEMWBTATION OF AES-RTOS

Algorithm Implementation of Advanced Encryption System

```
Needed Queue s-box  input  [6][6]   output 36
    Cipher [6][6]  key 24 random
    Procedure  PLAIN TEXT  priority 4  stack size 96
    For k = 1 to 36 do
        If k < sensor reading  then
            Input[x_k]  < sensed data
        Elseif
            input [x_k]  <  00x0H
        elseif
            r  = random key then
            input[sensor reading] < flag status
            initiate random procedure
       end if
  end for
end procedure
function DATA ROTATION  priority  2, stack size 96
     for j = 1 to 10 do
     input < s-box[:j]
          for m = 1 to 5 do
          input [:] < input [:i]  >> n
          input round constant [:]
      end for
     initiate Queue ( )
          end for
  end procedure
function ROUND KEY priority   2,  stack size 96
      if cipher < key
          for m = 1 to 10 do
          g(ciphe[r3])
          s.box cipher([3])
     elseif cipher[i]  <  cipher [i]  XOR  s.box(cipher[3])
                for h  1 to 4 do
                else  cipher[h] < cipher [h – 1] XOR cipher[k]
       end if
                end for
          end for
   end procedure
function CIPHER TEXT  (priority    1,  stack size 96)
       if queue != 0 and QUEUE != 10 then
            output[:] < input [:]  XOR cipher[:]
       elseif QUEUE == 10 then
            transmit > out[:]
      endfi
  end procedure
```

Figure 3 Algorithm Implementation of AES-RTOS

Table 1 Comparison of the computational resources used up by each AES library

LIBRARIES	FLASH	RAM	RESPONSE TIME
Lib 1	6880	1070	3.45µs
Lib 2	7460	805	14.5µs
IIB AES Version 1	5050	602	10.54µs
Lib AES Version 2	4621	358	5.45µs
Lib AES-RTOS Version 3	11220	790	7.93µs

4.3 The Proposed Environment

The algorithm proposed in this paper for IoT encryption was validated in an actual environment by transmitting data from the sensors in greenhouses (controlled environmental conditions**).** In order to get the data from the environment, the sensors were installed on autonomous robots which moves around the greenhouse vicinity to obtain environmental data. Based on this, the following assumptions are made: (i) The secured nature

of the IoT devises are guaranteed based on the AES installed on it. (ii) The architecture of the IoT devices are known by the IoT device acting as the central hub, this includes the capture of environmental parameters using the LoRa protocol having a preamble of 3 bytes , a header of 3 bytes with an end of frame of 2 bytes, while 1 byte was used for the detection of error. The library of the IoT AES uses 8 bits to compile data which makes it compatible with most microcontrollers. In order to capture data sent from the autonomous robots, the IoT AES uses 8 bytes to receive the following: 1 byte to obtain the robots position, encryption key update (1 byte), sensing of environmental temperature (1 byte), Humidity sensing (1 byte), Ultraviolet ray sensing (1 byte), status of battery (1 byte), forward movement of robots (1 byte), backward movement of robot (1 byte) It should be noted here that the IoT sensor was designed to capture three different types of environmental condition, however in this paper, the environmental temperature was captured by the autonomous robots and their readings were sent to the IoT devices.

The implementation of the AES algorithm on top of the IoT devises has the following performance measures: (i) time to encrypt input data, (ii) error detecting function to determine if correct encrypted data was decrypted at the receiver. In order to validate the results obtained from the experiments, the algorithm was tested in two ways (i) using the real time operating system on the IoT, and (ii) without using the RTOS. This was used to ascertain if the time response for data aggregation at the central hub was good. From the experiments, it was observed that the response time for the AES implementation with real time operating system was 14.2μs, while the corresponding time without the RTOS was 34μs. Also the time taken for the transmission of the encrypted data to the receiver with the rea time operating system was 21μs while the corresponding time without the RTOS was 43μs. The reduction in time for the RTOS system was due to the parallel programming employed where multiple instructions which do not depend on one another can be

run simultaneously Also operation which are not needed for certain concurrent operations can be disabled by setting higher priorities to those operations needed while setting low priorities to operations that are not needed, this can help shot off unneeded operations thereby making the RTOS algorithm run faster and increasing the battery lifetime of the IoT devices. It was observed that the battery lifetime was increased by 34%, while the RAM consumption was reduced by 43% when compared to the non RTOS implementation. The time of response was also reduced by 45% when compared to the non RTOS implementation. For the experiments, where the ESP8255 board was used to implement the AES-RTOS, it consumed 23% of flash and 30% of RAM, with response times in the range of 4 μs and 7 μs. This was an improvement of 28% over the non RTOS implementation. It was observed that the central hub IoT device was able to efficiently update the encryption keys with no loss of data between it and the other IoT devices on the network. The design in this paper serves a good background for further studies inti the use of machine learning on AES-RTOS system implementation

V. CONCLUSION

The aim of cryptography is to ensure that transmitted data gets to its destination without being corrupted. Cryptographic algorithms however presents overhead in IoT devices thereby reducing their efficiency. It is therefore pertinent that a cryptographic algorithm with less memory consumption be designed for IoT devices due to their memory and computational restraint. Cryptographic libraries are good for normal network security but it is too heavy for network involving IoT devices. This brings about the design of a real time operating system with certain optimization property such as pipelining and parallel operation, which will help in the design of secure network for IoT network. In this paper a reduced RTOS was designed for IoT network with improvements in device's response time with conserved memory consumption. The optimization technique used also helped in the reduction of RAM usage in thread suspension. The advantage of this is an efficient and secure IoT network, the device's flash and RAM usage was reduced, so as to be used for other operations in the network such as data aggregation. From the experiments conducted, on the Arduino Uno, the encryption library used 30% of flash and 37% of RAM while the time of response was between 4 and 7 us. When ESP8266 board was used in place of the Arduino microcontroller, it consumed 23% and 30% of flash memory for the AES-RTOS and RAM respectively. The response time in this case was between 6 and 10.5 μs. The IoT device acting as the central hub could effectively utilize the change of encryption keys without loss of data packets as a result of regular encryption key update. This makes the future improvement of the use of machine learning on secure data packet transmission in IoT network possible.

REFERENCES

[1]. Borges, M., Paiva, S., Santos, A., Gaspar, B., and Cabral, J. (2023). Azure RTOS thread design for low-end robot device. In 2023 2nd International Conference on Societal Automation (SA), pp 1–8.

[2]. Dang, T.-P., Tran, T.-K., Bui, T.-T., and Huynh, H.-T. (2022). Lora gateway based on soc-fpga platforms. In 2021 International Symposium on Electrical andbElectronics Engineering (ISEE), pp 48–52.

[3]. Fotovvat, A., Rahman, G. M. E., Vedaei, S. S., and Wahid, K. A. (2023). Comparative performance analysis of lightweight cryptography algorithms for IoT sensor nodes. In IEEE Internet of Things Journal, Vol 8 No 10 pp 8279–8290.

[4]. Gehrmann, C. and Gunnarsson, M. (2019). An Identity Privacy Preserving IoT Data Protection Scheme for Cloud Based Analytics. In Proceedings 2019 IEEE International Conference on Big Data, Big Data pp 5744–5753.

[5]. Gope, P. and Hwang, T. (2016). A Realistic Lightweight Anonymous Authentication Protocol for Securing Real-Time Application Data Access in Wireless Sensor Networks. In IEEE Transactions on Industrial Electronics, Vol 63 No 11 pp 7124–7132.

[6]. Guo, X., Hua, J., Zhang, Y., and Wang, D. (2019). A complexity-reduced block encryption algorithm suitable for internet of things. In IEEE Access, Vol 7 pp 54760–54769.

[7]. Hijawi, U., Unal, D., Hamila, R., Gastli, A., and Ellabban, O. (2023). Lightweight kpabe architecture enabled in mesh networked resource-constrained IoT devices. In IEEE Access, Vol 9 pp 5640–5650.

[8]. Islam, M. S., Verma, H., Khan, L., and Kantarcioglu, M. (2019). Secure real-time heterogeneous IoT data management system. In Proceedings 1st IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications, TPS-ISA 2019,pp 228–235.

[9]. Jalaly Bidgoly, A. and Jalaly Bidgoly, H. (2019). A novel chaining encryption algorithm for LPWAN IoT network. In IEEE Sensors Journal, Vol 19 No 16 pp 7027–7034.

[10]. Khan, M. N., Rao, A., and Camtepe, S. (2022). Lightweight Cryptographic Protocols for IoT-Constrained Devices: A Survey. In IEEE Internet of Things Journal, Vol 8 No 6 pp 4132–4156.

[11]. Landman, D. Latif, M. A., Ahmad, M. B., and Khan, M. K. (2022). A Review on Key Management and Lightweight Cryptography for IoT. 2020 Global Conference on Wireless and Optical Technologies, GCWOT 2020.

[12]. Naru, E. R., Saini, H., and Sharma, M. (2017). A recent review on lightweight cryptography in IoT. In 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), pp 887–890.

[13]. Prakash, V., Singh, A. V., and Kumar Khatri, S. (2019). A New Model of Light Weight Hybrid Cryptography for Internet of Things. In Proceedings of the 4th International Conference on Electronics and Communication and Aerospace Technology, ICECA 2019, pp 282–285.

[14]. Ramesh, S. and Govindarasu, M. (2020). An efficient framework for privacy-preserving computations on encrypted IoT data. In IEEE Internet of Things Journal, Vol 7 No 9 pp 8700–8708.

[15]. Rosero-Montalvo, P. D., L´opez-Batista, V. F., Arciniega-Rocha, R., and Peluffo-Ord´o˜nez, D. H. (2021). Air Pollution Monitoring Using WSN Nodes with Machine Learning Techniques: A Case Study. Logic Journal of the IGPL.

[16]. Serra, L. F. D., Gonc¸alves, P. G. B., Lopes Fraz˜ao, L. A., and Antunes, M. J. G. (2021). Performance analysis of AES encryption operation modes for IoT devices. In 2021 17th Iberian Conference on Information Systems and Technologies (CISTI), pp 1–6.

[17]. Shah, T. and Venkatesan, S. (2018). Authentication of IoT Device and IoT Server Using Secure Vaults. In Proceedings - 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications and 13th IEEE International Conference on Big Data Science and Engineering, Trustcom/BigDataSE 2018, pp 819–824.

[18]. Sung, B.-Y., Kim, K.-B., and Shin, K.-W. (2022). An AES-GCM authenticated encryption crypto-core for IoT secu-IoTBDS 2022 - 6th International Conference on Internet of Things, Big Data and Security In 2018 International Conference on Electronics, Information, and Communication (ICEIC), pp 1–3.

[19]. Szymanski, T. H. (2017). Security and Privacy for a Green Internet of Things. IT Professional, Vol 19 No 5) pp 34–41.

[20]. Tsai, K.-L., Huang, Y.-L., Leu, F.-Y., You, I., Huang, Y.- L., and Tsai, C.-H. (2018). AES-128 based secure low power communication for LORAWAN IoT environments. In IEEE Access, Vol 6 pp 45325–45334.

[21]. Van Heesch, D. (2018). Arduino cryptography library.