Analysis and Optimization of Advanced Encryption Standard for the Internet of Things

Ishfaq Sultan, Bisma Javid Mir, and M. Tariq Banday*, *Senior Member, IEEE*
Department of Electronics and Instrumentation Technology
University of Kashmir, Srinagar, *India*
*sgrmtb@yahoo.com

***Abstract*—The Internet of Things creates advanced application systems requiring minimum human interaction by integrating entities having unique addressing mechanisms with digital processing platforms by communicating in a network. The Internet of Things devices being resource constrained are prone to attacks and security breaches. Conventional cryptographic procedures are not viable for Internet of Things and embedded applications as the computations involved with them require higher processing power, memory and energy. Lightweight cryptographic techniques can be used in resource constrained Internet of Things devices to mitigate the security issues. These lightweight cryptographic solutions use smaller key sizes, smaller block sizes and lesser number of rounds. This paper uses a lightweight cryptographic algorithm, AES (Advanced Encryption Algorithm) which is one of the efficient and robust algorithms, to secure communication in the wireless sensor networks. The experimentation uses UDP protocol for transmissions and Routing Protocol for Low-Power and Lossy networks (RPL) for packet delivery from source to destination. Later, the performance of different variants of AES is checked by performing the power analysis. The results demonstrate that AES-192 and AES-256 with 8 rounds show optimized results in terms of power, thus making it suitable for power constrained devices.**

Keywords—Lightweight Cryptography; UDP; User Datagram Protocol; RPL; Routing Protocol for Low-Power and Lossy Networks; Contiki; Cooja.

I. INTRODUCTION

The concept of the Internet of Things was introduced in 1998 by Kevin Ashton [1]. Internet of Things (IoT) is an interconnection of devices which have the capability of sensing, actuating, communicating, etc. through the Internet where human involvement is minimum. IoT is a term that is allied with the notion of "future Internet" [2].

The IoT architecture comprises of three basic layers: physical layer which consists of sensors and actuators, network layer which have communication capabilities and application layer at the user end [3][4]. IoT is presently an emergent technology that will cover every aspect of future to create a much easy life for us with smart devices all around. Few instances of the IoT systems are smart buildings and homes, smarter vehicles, e-health aids etc. [5]. These IoT systems are commonly used today and it is expected that the total number of nodes in 2020 will cross 212 billion [3].

Besides the heterogeneous and ubiquitous nature of IoT, huge number of nodes monitoring the environment makes security a key issue in IoT devices. Severe security challenges are growing in terms of data confidentiality, integrity, and authenticity. Conventional cryptographic algorithms are slow in speed, use large key size and consume huge power. Therefore, these techniques cannot be used in resource constrained IoT applications [6]. Since the execution time of traditional cryptosystems is enormous, several researchers have tried to decrease this time [7][8][9][10][11]. But these attempts resulted in another constraint i.e., cost. The hardware requirements for such a system are very costly because of the suggested integrated modules. The best recommended solution for the security in IoT is lightweight cryptography which reduces the total implementation cost related to many parameters, such as key size, power consumption, area, cycle rate and throughput rate. The techniques used in lightweight cryptography consume small memory and requires less computation, thus using least amount of necessary resources of the nodes.

The lightweight cryptography is broadly classified into two categories [12], symmetric lightweight ciphers and asymmetric lightweight ciphers. Symmetric lightweight ciphers are faster and consume less power but are less secure compared to asymmetric lightweight ciphers. Asymmetric ciphers on the other hand are slower and consume more power. Symmetric lightweight techniques comprise of lightweight hash algorithms and lightweight block ciphers. The two NIST-accepted lightweight block ciphers [7] include Advanced Encryption Standard (AES) and Triple Data Encryption Algorithm (TDEA) [13]. The Advanced Encryption Algorithm has three variants that comprise of AES-256, AES-192, and AES-128 with key sizes 256-, 192- and 128-bit, respectively. The block size of all the variants is same (128-bits).

In this paper, different modes of AES i.e., AES-ECB, AES-CBC, and AES-CTR with variable key lengths have been implemented in *Contiki* OS using *Cooja* simulator. These modes have also been tested and implemented using 8 rounds instead of 12 and 14 in 192- and 256-bit key lengths respectively. The motes used in the simulation are *Sky* and *Z1* motes. *Sky* mote is based on Texas Instruments MSP430 F1611 microcontroller whereas *Z1* mote is based on Texas Instruments MSP430 F2617 microcontroller. Both the motes are used in low power IoT and wireless sensor networks and are based on IEEE 802.15.4 protocol for wireless communication.

II. RELATED WORK

Margi et al. [14] investigated the influence of operating systems (*TinyOS* and *Contiki*) on the security of IoT and embedded applications on the *TelosB* platform. Hyncica [15] the authors assessed the performance of 15 block ciphers on 8-, 16- and 32-bit microcontrollers. They adopted TomCrypt LTC library which is generally suited for 32-bit platforms and is not optimized for constrained IoT devices. They implemented the Electronic Codebook (ECB) mode of AES operation, which targets only confidentiality and permits a clear distinction in the performance of the plain block ciphers compared to other modes of operation. ECB mode is considered insecure in the modern world because the applications demand more advanced security measures including authentication and data integrity. Therefore, it is necessary to evaluate the performance assessment over resource constrained devices of other AES modes of operation. The authors did not address the power measurements which is an important aspect as far as constrained IoT devices are concerned.

978-1-7281-5475-6/20/$31.00 ©2020 IEEE

Law, Doumen and Hartel [16] implemented 8 different block ciphers in 4 encryption modes on the MSP430F149 platform. The platform is based on Texas Instruments 16-bit RISC microcontroller. The authors focused on encryption modes and did not investigate other conventional security principles required by almost all IoT devices. Geovandro, Pereira and Renan [17] presented energy and time benchmarks of encryption implementations for different platforms and operating systems. The authors used the Intel Edison platform for experimentation and gave benchmark results of symmetric cryptography, and also described the methodology for energy consumption measurement on the said platform. Peyrard et al [18] improved the security level in *Contiki* OS by demonstrating the usage of formal verification. The authors presented a case study on analytical verification of encryption and decryption modules of *Contiki* OS on AES-CCM* (CCM-star), which is a variant of AES in counter with CBC-MAC mode by employing software analysis platform: Frama-C for C code. Abdelmoghni [19] investigated the energy consumption and duration of the AES both in software and hardware with varying key and buffer size on two resources constrained IoT platforms.

6LoWPAN adaptation layer is one of the prevalent solutions for adopting transmitting messages over IEEE 802.15.4 networks. Rantos, et al [20] focused on the security delivered to LLN (low-power and lossy network) nodes by exploiting 6LoWPAN adaptation setup. The authors suggested a compression format for IPsec, which is able to offer end-to-end security. The format utilized AES-CCM* (CCM-Star), while contemplating the boundaries of the fundamental IEEE 802.15.4 protocol. The structure proposed by the authors offered authentication, confidentiality, and integrity of data and featured low packet overhead compared to similar approaches. Smeets et al. [21] proposed a symmetric key management structure for low power wireless sensor networks that have the capability to use damage proof hardware for key distribution and generation. The authors pointed out that the structure made use of a trusted central entity for key negotiation and no deployment knowledge was required before enrolling to offer end-to-end security. The assessment and implementation were performed on low power *Zolertia Z1* hardware platform running on the *Contiki* operating system.

III. ADVANCED ENCRYPTION STANDARD

National Institute of Standards and Technology (NIST) in 2001 confirmed the block cipher Rijndael as the new Advanced Encryption Standard (AES) and declared it as an absolute standard in FIPS Publication 197 [22]. Rijndael was given by two new Belgian scientists, Joan Daemen and Vincent Rijmen. AES is the most extensively used symmetric cipher which means that a single key is used for the operation of encryption and decryption. It is safe against all the known attacks. The key size in AES can be 128-, 192-, 256-bits and the block size is 128-bits. The number of rounds in AES with key size 128-, 192-, 256-bits are 10, 12 and 14 respectively as shown in Table 1 [23].

The operations of AES are performed on bytes unlike DES where operations are carried out in bits. 128-bits of plain-text block is treated as 16 bytes in AES. These 16 bytes are arranged as a matrix with four rows and four columns. In these 16 bytes, first 4 bytes occupy the first column in the matrix of size 4x4. The next 4 bytes occupy the second column, the following 4 bytes occupy the third column and the last 4 bytes occupy the fourth column. This 4x4 matrix is called the state array. In each round of AES, processing is done on input state array to generate the output state array. The output state array produced by the last round is rearranged into an output block of size 128-bit.

TABLE 1. RELATIONSHIP BETWEEN KEY SIZE AND NUMBER OF ROUNDS

Algorithm	Key Size	Length of Key in words	Size of Block	Size of Block in words	Total Number of rounds
AES-256	256	8	128	4	14
AES-192	192	6	128	4	12
AES-128	128	4	128	4	10

The block ciphers are structures used for the process of encipherment and decipherment where a block of input data is treated as one block and produces a block of output data that is of the same size [24]. In AES, block size is 128-bits or 16 bytes. The block cipher algorithms should have the capability of enciphering an input data with different size than the defined block size of the algorithm. It can be provided by using different modes of operation. Some of these modes convert the block cipher into stream cipher thus making the algorithm strong. The different modes of operation standardized by NIST in 2001 are Cipher FeedBack (CFB), Cipher Block Chaining (CBC), Electronic Code Book (ECB), Counter mode (CTR) and Output Feed Back (OFB). All these modes have been approved for AES [25].

A. ECB (Electronic Code Book) Mode

The simplest mode of operation is Electronic Code Book (ECB) Mode. In this mode, the ciphertext is generated by dividing the plaintext in blocks and then encrypting each block separately using the same key. The resultant ciphertext is also generated in blocks. If the input message is not a multiple of block size, it is padded such that it becomes a multiple of block size. One of the advantages of this mode is that error is not propagated from one block to another [26][23][27]. But this mode has a weakness that similar plaintext blocks result in similar ciphertext blocks, thus making it unsuitable for encryption of data greater than one block [28][26].

B. CBC (Cipher Block Chaining) Mode

To overcome the disadvantage of ECB Mode, this mode implements two new ideas. The first idea is that the encryption of all the blocks is reliant on each other in such a way that the ciphertext depends not only on the current block but also on all the previous plaintext blocks. The second idea is that an initialization vector (IV) is used to create randomness in the ciphertext [23]. The size of the IV is equal to the block size. In this mode, the first plaintext block is x-ored with the IV and the output generated is encrypted using the key resulting in the first ciphertext. This ciphertext is then x-ored with the next plaintext message and the result is again encrypted using the same key, thus generating the second ciphertext. In the similar manner, the process continues. Thus, in this mode different output is generated for two similar blocks. Also, a different result can be obtained after encrypting two similar messages if the IV used is different. The disadvantage in this mode is that an error in one block will propagate to all other blocks which will be reflected in the process of decryption.

C. CTR (Counter) Mode

In this mode, the block cipher works like a stream cipher. An input block called counter (just like the initialization vector) is used. The size of the counter is equal to the block size. The counter is encrypted using the key and the output is x-ored with the first plaintext block to generate the first ciphertext block. For the next block, the counter is

incremented and encrypted using the same key and then again x-ored with the second plaintext block. For the following blocks the counter is incremented again and again and, in this way, the encryption of the data takes place. Just like ECB, CTR does not propagate error from one block to another [23][29]. As there is no feedback in this mode, thus an efficient implementation of this mode is possible in both software and hardware by exploiting the use of parallelism [28].

IV. IMPLEMENTATION

The work carried out in this paper has been done in an open source operating system called *Contiki*. This operating system is best suitable for embedded devices with less memory and is also used in wireless sensor networks. *Contiki* operating system is extremely portable, supports multitasking and is best suited for microcontrollers with small memory. It has 40 KBs of ROM and 2KBs of RAM in its typical configuration. The code in *Contiki* consumes memory of the order of kilobytes and it can be configured to be as low as tens of bytes. Both IPv6 and IPv4 are supported in *Contiki*. *Contiki*-OS presented the concept of using Internet Protocol communication in low-power sensor networks. Rime and *uIP* are the two communication stacks in *Contiki*. Rime has been aimed for low-power radios as it is a lightweight communication stack. It offers basic communication that ranges from local area broadcast to reliable multi-hop bulk data flooding. Similarly, *uIP* being a small TCP/IP stack is responsible for communication over the Internet in *Contiki*. *Contiki*-OS is supported by a wide range of platforms that include embedded microcontrollers like ARM, AVR, msp430, pic32, etc., old home computers, and many more.

The simulator for *Contiki* OS is called *Cooja*. The motes in *Contiki* are simulated in *Cooja* which is a real compiled and executing *Contiki* system. *Cooja* works by compiling *Contiki* for the native platform as a shared library and using JNI (Java Native Interface) loads the library into Java. Different kinds of sensor nodes can be a part of *Cooja* simulation wherein different nodes can incorporate different *Contiki* libraries. Few functions help *Cooja* to analyse and control *Contiki* OS. A tool called *powertrace* is used in *Contiki* application to measure the amount of power consumed. To use this tool in any *Contiki* application, simply edit the Makefile of the application using:

APPS += powertrace

Another change that needs to be done, is in the main program of the application:

This library is added in the beginning of the code:

#include "powertrace.h"

This function is added in the PROCESS_THREAD after the command PROCESS_BEGIN:

*powertrace_start (CLOCK_SECOND * 10);*

Where 10 is the runtime. It implies that the values of different parameters of power consumption will be printed after 10 seconds repeatedly [13].

The printed parameters include *clock_time* i.e., clock time; *rimeaddr* i.e., rime address; *seqno* i.e., sequence number; *all_cpu* i.e., collected CPU power consumption in active mode; *all_lpm* i.e., collected power consumption in Low Power Mode (LPM); *all_transmit* i.e., collected power consumption while transmission; *all_listen* i.e., collected power consumption while reception; *all_idle_transmit* i.e., collected idle transmission power consumption; *all_idle_listen* i.e., collected idle listen power consumption; *CPU* i.e., CPU power consumption for this particular cycle; *lpm* i.e., LPM power consumption for this particular cycle; *transmit* i.e., transmission power consumption for this particular cycle; *listen* i.e., listen power consumption for this particular cycle; *idle_transmit* i.e., idle transmission power consumption for this particular cycle; *idle_listen* i.e., idle listen power consumption for this particular cycle;

This implementation was done in *Cooja* using *Sky* and *Z1 motes*. *Tmote Sky* is an ultra-low power wireless module based on Texas Instruments MSP430 F1611 microcontroller extensively used in wireless sensor networks and rapid application prototyping. It consists of 2.4GHz IEEE 802.15.4 wireless transceiver having interoperability with other IEEE 802.15.4 devices. *Z1 mote* is a low power wireless hardware platform based on Texas MSP430 F2617 microcontroller complaint with IEEE 802.15.4 and Zigbee protocols used widely in wireless sensor networks and low power IoT applications. In this paper, the power consumption of different AES encryption algorithms is analyzed using *Sky* and *Z1* motes in *Cooja* simulator using the *Contiki* operating system.

The implementation uses a simple example of RPL-UDP present in *Contiki* operating system where transmission is done using UDP protocol and packet delivery from source to destination is done by RPL protocol. The implementation is done in *Cooja* simulator using *Sky* and *Z1 motes*. User Datagram Protocol, which is part of Internet Protocol suite (UDP/IP suite) is unreliable and connectionless protocol and therefore does not require a connection prior to data transfer. The example in our case simulates a client and a server node. The client node routes data and sends it to the server node using the RPL routing protocol. While sending the data from client, the data is sent to buffer first where the encryption operation takes place. The buffer is then transmitted to the server using the UDP protocol. The encrypted message in the buffer is decrypted using the decryption algorithm at the server end after it is received. The modification of the data in the buffer after encryption or decryption is done using the different modes of the AES algorithm. The same message of length 48 bytes is encrypted and transmitted using different key sizes and encryption modes including AES-ECB, AES-CBC, and AES-CTR. The message is also encrypted with a modified 192- and 256-bit key using only 8 rounds. The transmission power and total power consumption in each of the above encryption modes are computed and compared.

V. RESULTS AND DISCUSSION

The experimentation shows the analysis and optimization of the AES algorithm on the basis of power consumption which is done using the *powertrace* application. The *powertrace* application is embedded in the *Contiki* operating system which generates different power parameters like CPU, LPM, transmission, reception, idle transmission, idle reception, etc. The simulation has been carried out in *Cooja* simulator where the different AES types with appropriate rounds and key sizes have been analysed. The power analysis for all types of AES has been done on the payload of 48 bytes transmitted using the RPL/UDP protocol. The five configurations include AES with 128-, 192-, 256-bit keys, and AES with 192- and 256-bit keys using only 8 rounds instead of 12 and 14 respectively. The experimental implementation is done by using *Sky* and *Z1* motes as the target devices in the simulation environment. The input voltage of the *Sky* mote is taken as 3V, and the transmission and reception peak currents are taken as 21mA and 23mA respectively. The active mode and the low power mode

currents are taken as 600µA and 3µA respectively. Similarly, for *Z1* mote the input voltage is taken as 3V, and the transmission and reception peak currents are taken as 17.4mA and 18.8mA respectively. The active mode and the low power mode currents are taken as 0.5mA and 0.5µA respectively.

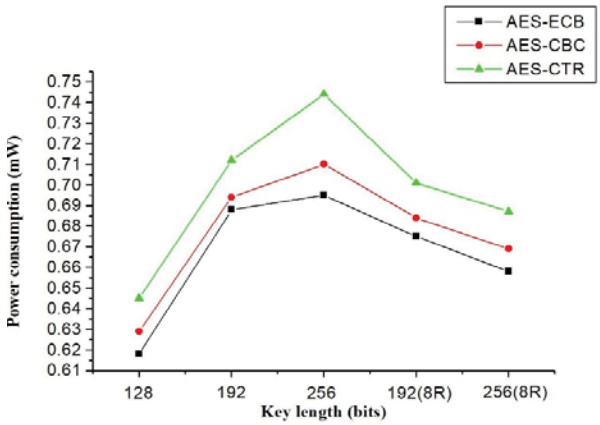

FIGURE 1: TOTAL POWER CONSUMPTION OF DIFFERENT ENCRYPTION TYPES USING VARIABLE KEY LENGTHS IN *COOJA* SIMULATOR USING *SKY* MOTE.

Figure 1 shows the power consumption (in milliwatts) for *Sky* mote in different AES encryption types using five key configurations. As evident from the figure AES-ECB consumes least power followed by AES-CBC and AES-CTR modes. The power consumption is least at 128-bit key length in all the three modes. The modified key versions of 192- and 256-bit with only 8 rounds consume less power compared to their standard versions using 12 and 14 rounds respectively. The power consumption in modified AES-256 is comparable to standard 128-bit key length in all three encryption types.

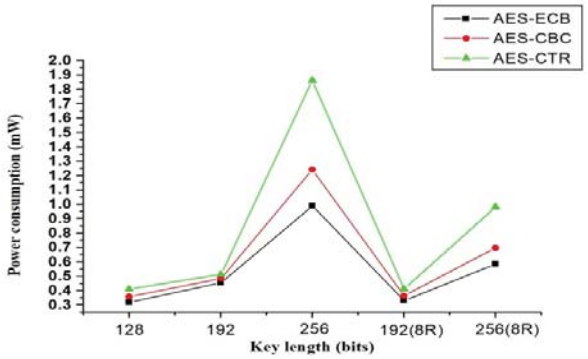

FIGURE 2: TOTAL POWER CONSUMPTION OF DIFFERENT ENCRYPTION TYPES USING VARIABLE KEY LENGTHS IN *COOJA* SIMULATOR USING *Z1* MOTE.

Figure 2 shows the power consumption (in milliwatts) for *Z1 mote* in different AES encryption types using five key configurations. AES-ECB consumes the least power followed by AES-CBC and AES-CTR modes. The power consumption is least at 128-bit key length in all the three modes. As shown in Figure 2 last two key versions of 192- and 256-bit with only 8 rounds consume less power compared to their normal versions using 12 and 14 rounds respectively. The power consumption in modified AES-256 is comparable to standard 128-bit key length in all three encryption types. Table II shows the comparison of power consumption values in three encryption types using different key configurations in *Cooja* between *Sky* and *Z1 motes*. As is evident from the table *Z1 Mote* is more efficient than *Sky Mote* for all the encryption modes.

TABLE 2. RELATIONSHIP BETWEEN KEY SIZE AND NUMBER OF ROUNDS

Key Length (bits)	Power consumption (mW) for different modes of AES in *Sky* mote			Power consumption (mW) for different modes of AES in *Z1* mote		
	ECB	*CBC*	*CTR*	*ECB*	*CBC*	*CTR*
128	0.618	0.629	0.645	0.317	0.358	0.410
192	0.688	0.694	0.712	0.454	0.484	0.512
256	0.695	0.710	0.744	0.989	1.241	1.861
192(8R)	0.658	0.669	0.687	0.331	0.364	0.412
256(8R)	0.675	0.684	0.710	0.584	0.696	0.981

CONCLUSION

Lightweight block ciphers play a key role as far as security of resource-constrained IoT devices is concerned. In this paper, we have implemented AES block cipher in a simulation environment using *Cooja* simulator of the *Contiki* operating system. Three different modes of AES are implemented which include AES-ECB, AES-CBC, and AES CTR on *Zolertia Z1* and *Sky motes* in *Cooja*. The power consumption is calculated for five variable key configurations on the said modes of AES operation. The results show that AES-ECB is best-suited mode as far as power consumption is concerned followed by AES-CBC and AES-CTR modes. The modified key lengths, i.e. 192- and 256-bit with 8 rounds show a reduction in power consumption compared to their counterparts. Hence, these modifier keys can be used in resource-constrained IoT devices and can also be helpful against brute-force and differential cryptanalytic attacks because of the larger key size.

ACKNOWLEDGEMENT

This work has been supported by the Ministry of Electronics and Information Technology, Govt. of India under its project grant no.12 (2)/2017-CSRD.

REFERENCES

[1] Ashton, K. (2009). That 'internet of things' thing, *RFID Journal*, Vol. 22, no. 7, pp. 97-114.

[2] Bhumi, N., Tushar, C. (2015). Study of various Internet of Things platforms, *International Journal of Computer Science & Engineering Survey (IJCSES) Vol.6, No.6*, December 2015.

[3] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications, *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347-2376.

[4] Sethi P., Sarangi, S. R. (2017). Internet of Things: Architectures, Protocols, and Applications, *Hindawi Journal of Electrical and Computer Engineering*, Article ID 9324035, 25 pages, 2017.

[5] Arsalan, M. N., Niraj, K. J. (2017). A Comprehensive Study of Security of Internet-of-Things, *IEEE Transactions on emerging topics in computing, vol. 5, no. 4, pp. 586-602*, 1 Oct. – Dec. 2017.

[6] Tapalina B. (2013). LICRYPT: Lightweight Cryptography Technique for Securing Smart Objects in Internet of Things Environment, *CSI Communications*, May 2013.

[7] McKay, K. A., Bassham, L., Turan, M. S., Mouha, N. (2016). Report on lightweight cryptography. *NIST DRAFT*, NISTIR, pp 1–29.

[8] Elbirt, A. J. (2007). Fast and efficient implementation of AES via instruction set extensions, *Proc. of 21st International Conference on Advanced Information Networking and Applications Workshops AINAW'07*, vol. 1, pp. 396-403.

[9] Hodjat, Alireza, and Verbauwhede, I. (2004). Interfacing a high-speed crypto accelerator to an embedded CPU. *In Signals, Systems, and Computers, Proc. of Thirty-Eighth IEEE Asilomar Conference*, vol. 1, pp. 488-492.

[10] Grabher, P., Großschädl, J., Page, D. (2008). Light-weight instruction set extensions for bit-sliced cryptography, *Cryptographic Hardware and Embedded Systems–CHES*, pp. 331-345.

[11] O'Melia, Sean, Elbirt, A. J. (2008). Instruction Set Extensions for Enhancing the Performance of Symmetric-Key Cryptography, *Proceedings of Computer Security Applications Conference*, pp. 465-474.

[12] Hammad, B. T., Jamil, N., Rusli, M. E., Reza Z'aba, M., Ahmed, I. T. (2007). Implementation of Lightweight Cryptographic Primitives, *Journal of Theoretical and Applied Information Technology*, vol. 95, no. 20.

[13] Barker, W. C., Barker, E. (2012). Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, *NIST Special Publication* (SP) 800-67.

[14] Margi, C. B., De Oliveira, B. T., De Sousa, G. T., et al., (2010). Impact of operating systems on Wireless Sensor Networks (security) applications and testbeds, *Proceedings of the 2010 19th International Conference on Computer Communications and Networks, (ICCCN '10)*, August 2010.

[15] Hyncica, O., Kucera, P., Honzik, P., Fiedler, P. (2011). Performance evaluation of symmetric cryptography in embedded systems, *Proceedings of the 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS'2011*, pp. 277–282, September 2011.

[16] Law, Y.W., Doumen, J., Hartel, P. (2006). Survey and benchmark of block ciphers for wireless sensor networks, *ACM Transactions on Sensor Networks*, vol. 2, no. 1, pp. 65–93.

[17] Geovandro, C. C. F., Pereira, Renan, C. A., Alves, Felipe L. da Silva, Roberto M., Azevedo, Bruno C., Albertini, Cíntia B. Margi. (2017). Performance Evaluation of Cryptographic Algorithms over IoT Platforms and Operating Systems, *Security and Communication Networks*, vol. 2017, Article ID 2046735, 16 pages.

[18] Peyrard, A., Kosmatov, N., Duquennoy, S., Lille, I., Raza, S. (2018). Towards Formal Verification of *Contiki*: Analysis of the AES-CCM* Modules with Frama-C. *In Proceedings of the 2018 International Conference on Embedded Wireless Systems and Networks (EWSN '18)*. Junction Publishing, USA, 264-269.

[19] Abdelmoghni, T., Mohamed, O. Z., Billel, B., Mohamed, M., Sidahmed, L. (2018). Implementation of AES Coprocessor for Wireless Sensor Networks, *2018 International Conference on Applied Smart Systems (ICASS)*, Medea, Algeria, 2018, pp. 1-5.

[20] Rantos, K., Papanikolaou, A., Manifavas, C. (2013). IPsec over IEEE 802.15.4 for low power and lossy network, *proceedings of the 11th ACM International Symposium on Mobility Management and Wireless Access (MobiWac '13)*. ACM, New York, NY, USA, 59-64.

[21] Smeets, Ruben, Aerts, Kris, Mentens, Nele, Braeken, An, Segers, Laurent, and Touhafi, Abdellah. (2014). Cryptographic Key Management Architecture for Dynamic 6LowPan Networks, *proceedings of the 9th International Conference on Applied Informatics 1* (2014): 219-29.

[22] FIPS. (2002). Advanced Encryption Standard, *Federal Information Processing Standard (FIPS) Publication 198,* 2002.

[23] Paar, C., Pelzl, J. (2010). Understanding Cryptography: Textbook for Students and Practitioners, London, *Springer*, 2010.

[24] Blazhevski, D., Bozhinovski, A., Stojchevska, B., Pachovski, V. (2013). Modes of Operation of The AES Algorithm, *Proc. of 10th Conference for Informatics and Information Technology* (CIIT 2013).

[25] Dworkin, N. (2001). Recommendation for Block Cipher Modes of Operation, Methods, and Techniques, *NIST Special Publication 800-38A* Edition 2001.

[26] Stallings, W. (2005). Cryptography and Network Security: Principles and Practices, Fourth Edition. *NJ, Prentice-Hall*, 2005.

[27] Rogaway, P. (2011). Evaluation of Some Blockcipher Modes of Operation. *Cryptography Research and Evaluation Committees (CRYPTREC), 2011*.

[28] Douglas, R., S. (2006). Cryptography Theory and Practice, *Chapman and Hall/CRC*. 2006.

[29] Henriquez, F. R., Saqib, N. A., Perez, A. D., Koc, C. K. (2006). Cryptographic Algorithms on Reconfigurable Hardware, *Springer*, 2006.