


# Encryption Algorithms for WSN - IOT: A Software based Analysis with Contiki Cooja

Yves Frederic Ebobisse Djene<sup>1,2</sup><sup>a</sup>, Amine Berrazzouk<sup>1</sup>, Brahim El Bhiri<sup>2</sup> and Youssef Fakhri<sup>1</sup>

<sup>1</sup> LARIT- IBN Tofail University, Kenitra, Morocco

<sup>2</sup> SMARTiLab EMSI, Rabat, Morocco

Keywords: Wireless sensor network (WSN), IOT, simulation, cryptography, algorithms, contiki, analysis, evaluation

Abstract: IOT has increasingly become part of people's daily lives and is applied in an incredibly broad range of domains (transport, logistics, agriculture, etc...). Though initial applications of IOT did not take security into consideration, it has become an important field of research. This paper focuses on encryption algorithms evaluation using Contiki Cooja simulator. We mainly analyze the impact of the mode of operation, key length and blocks as we measure the number of clock ticks on sensor nodes.

## 1 INTRODUCTION

The IOT (Internet of Things) global market is meant to reach 1.6 trillion US dollar in 2025, with more devices interconnected and communicating through the internet (Tankovska, 2020). As we go through literature, there are several definitions of IOT (Ray, 2018):

– “3A concept: anytime, anywhere and any media, resulting into sustained ratio between radio and man around 1:1”

– “A global infrastructure for the information society enabling advanced services by interconnecting (physical and virtual) things based on, existing and evolving, interoperable information and communication technologies”

(Ray, 2018) identified several functional blocks required to build an IOT infrastructure. Among these are:

– Devices: sensors, actuators, monitoring devices, etc... that collect data and process it locally or send it to centralized servers. These devices usually have constrained capabilities in terms of storage and processing for instance.

– Communication: an important part of the architecture that involves protocols at the datalink, network, transport and application layer

– Services: such as device control and monitoring, as well as data publishing and analytics which need to be provided by a complete system

– Applications: in order to monitor the overall system, fields-oriented applications (agriculture, health, transport, smart grids) are required to keep track of the changes in the system and ease decision making.


– Security: copes with everything related to security in the system.

An IOT architecture is based on 3 important layers (Rwan, 2015):

- The perception layer “senses” the environment, in other words, it collects data from the real world (physical environment or nodes) and transmits it to the Network layer.
- The Network layer is in charge of data routing and transmission from one end to another end of the system (for example from a sensor to a sink, a gateway, or base station, etc..).
- Application layer guarantees confidentiality, authenticity and integrity of the collected data and helps create the smart environment.

If security was a neglected issue at the early stages of IOT development, it became a non-negligible aspect that needs to be properly addressed in IOT systems. Some IOT security issues relate to (Ray, 2018):

- Confidentiality: we need to secure data and deliver it to authorized users

 <https://orcid.org/0000-0003-4929-7362>

- Integrity: ensure that the accuracy of the data received and transmitted using end-to-end security in communications.
- Availability: data and devices must be accessible and reachable whenever required
- Authentication: be able to identify each IoT entity in the system
- Lightweight solutions: because of their constrained capabilities in terms of energy, computational power and storage, IoT devices require lightweight protocols and applications as we will later examine in this study.
- Heterogeneity: entities in an IOT system may come from different vendors, with different levels of specifications, but these specifications should still be able to seamlessly cooperate.
- Key Management System: whenever exchanged, data needs to be encrypted.

There are further security issues in IOT systems mentioned by (Sha, 2018) as :

- Integration with the physical world: a corrupted or wrong data transmitted by sensors in a train or a plane can have irremediable consequences. If the data is for example tampered by unauthorized access, it can endanger people's lives.
- Heterogeneity of devices and communications: depending on the domain of application, IOT systems do not have the same requirements when it comes to devices. Monitoring a farm, an industrial compound, a home or smart Grid comes with different costs in terms of physical architectures and communication schemes. While some may stress on data updates others may rely on more secure channels.
- Scalability: Deploying IoT systems increase interactions within the architecture, between nodes and outside the environment, with data servers, etc... As the number of entities grows, it is important to ensure the availability, the liability and the reliability of the system
- Resource constraints: as we previously mentioned, due to resource limitations, IoT devices are designed with low capabilities. Some features that need to be integrated in IoT architectures involve more challenges just as encryption, trust management and PKI usually require more powerful systems.

Addressing these issues and implementing solutions within IoT systems have an impact in several aspects of the behaviour of the environment in terms of energy consumption, reliability and availability.

Our focus in this paper is the evaluation of some encryption algorithms on nodes in IoT/WSN architectures.

The present study explores literature and related work defining encryption modes and types as well as some performance analysis. Simulations, methodologies, results and their analysis are also presented.

## 2 RELATED WORK

When it comes to security challenges in IOT and WSN architectures, there are several aspects that can be addressed. (Mardiana binti Mohamad, 2019) analysis of publications in IoT security from 2016 to June 2018 (Elsevier, IEEE, Hindawi and Springer) showed that most papers focused respectively on authentication, trust, encryption and secure routing.

It is important to note that this evaluation may suffer some flaws because cryptography (encryption and decryption) is transverse to the previously mentioned fields. We can simply consider encryption as the process of transforming a plain text into a cypher text using a hash. This process can be reverted by the receiver via a key (deciphering).

Cryptography helps achieving several security goals in information systems in general. Through encryption and decryption, the sender and the receiver can communicate with a certain level of security. Depending on the encryption method, messages are meant to be useless for any unauthorized third party. By using hashing and message digests, data's integrity can be verified, while digital signatures and certificates tackle authentication goals.

(Mardiana binti Mohamad, 2019) also pointed out that security can be applied at different layers of an IoT architecture (physical, network or application). As we examine encryption systems, we will gradually investigate their effect on IoT or WSN devices and networks.

### 2.1 Encryption Modes

Encryption can be either done through a stream cipher that is byte-by-byte, or with block cyphers of fixed lengths when you work on larger blocks (Meneghello, 2019). Different operating modes can be used to encode and decode larger blocks namely (Kowalczyk, 2020):

- ECB (Electronic CodeBook): each plain block is encrypted and decrypted separately with a key.

- CBC (Cipher Block Chaining): the initial block is encrypted with the initializing vector. A XOR operation is performed with the plaintext block before encryption with the key. Next block uses the previous cypher block in the same manner. In order to decipher the stream, the first encrypted block is decrypted, then a XOR operation is applied to the result to get one plaintext block. Next step involves deciphering the new block with the key and applying a XOR operation with the cyphertext of the previous step.
- CTR (Counter): a number used once (nonce) is added to a counter and encrypted using the key. The final cipher block is then obtained through a XOR operation with the plain text. If you have several blocks, the counter increases for each block. The same operation is done for decryption by switching the positions of the cipher and the plaintext.

More modes are available such as OFB (Output FeedBack), CFB (Cipher FeedBack), and PCBC (Propagating or Plaintext Cipher-Block Chaining) but were not part of our study.

It is important to mention that these modes are related to the way blocks/stream are treated when encryption/decryption is performed. When you consider encryption algorithms, there are two main techniques.

## 2.2 Encryption Techniques

- Asymmetric Encryption

Also referred as the public key cryptography, it requires two keys: one public key used by the sender to encrypt data and the corresponding private key used by the receiver to decrypt the data.

- Symmetric Encryption

The secret key cryptography or symmetric encryption involves a unique key that is used by the sender and the receiver. At one end of the communication, encryption is performed and at the other end, decryption, using the same key.

AES (Advanced Encryption Standard), DES (Data Encryption Signature), 3DES (Triple DES) and Blowfish are examples of symmetric encryption algorithms while Diffie Hellman Key agreement, RSA (Rivest Shamir Adleman), El Galmal, DSA (Digital Signature Algorithm) and ECC (Elliptic Curve Cryptography) are asymmetric. These algorithms usually necessitate heavy computational power and resources that IoT devices may not have.

The performance of four algorithms namely AES, DES, 3DES and Blowfish was compared with ECB and CBC modes respectively as shown in figures 1 & 2 (Tamimi, 2006). The amount of time required to perform encryption and decryption was then analyzed. According to their work, Blowfish was the fastest in both modes, and AES increased rapidly with the size of the block. They also stated that these results were different from other works in literature because of the size of data size used.

It is important to mention that these experiments were conducted on 3500+ AMD 64bits processor with 1Gb of RAM and using C# and visual studio. Even if this work seems to be outdated, it does not comply with the IoT/WSN environment that impose some limitations in terms of RAM, storage and computational power in order to evaluate the effect on IoT and WSN networks and devices.

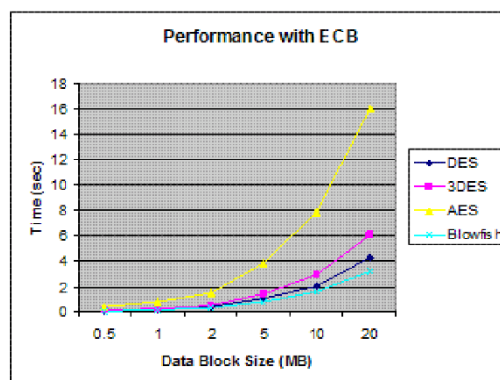


Figure 1 : ECB mode Performance

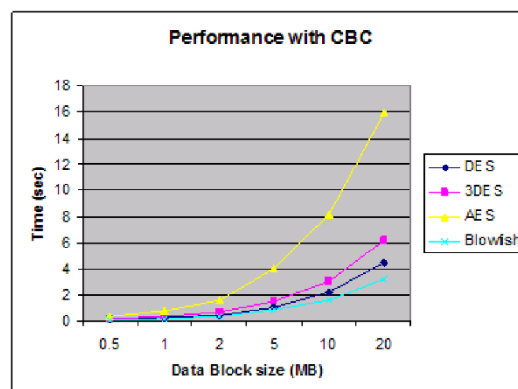


Figure 2 : CBC mode Performance

(Mansour, 2012) evaluated several algorithms (both symmetric and asymmetric) on sensors nodes with regard to energy consumption. They implemented their algorithms in nesC, a programming language for networked embedded systems using TelosB motes (sensors with 16-bit 8MHz

TI MSP430 micro-processor) while their asymmetric implementation of algorithms was based on TinyECC. Their results encompassed every stages of the encryption, from initialization to encryption/decryption phases (for symmetric keys) as well as the generation and exchange of private/public keys, the calculation of symmetric key between two nodes (for asymmetric encryption). They compared AES, CHAOS (CTR mode) - two symmetric algorithm on one hand - and ECIES (Elliptic Curve Integrated Encryption Scheme) coupled with AES and CHAOS with regard to execution time but also in terms of energy consumption. From their results 1 byte consumes the same amount of energy for encryption and decryption for AES and CHAOS while Asymmetric Algorithms show a higher consumption for the respective operations. Asymmetric algorithms tend to consume minimum 100 times amount of energy as illustrated in table I based on their work.

Table 1: Energy Consumption in mJ (Mansour, 2012)

	Algorithm	Encrypt	Decrypt
SYMMETRIC	AES-CTR	0.07	0.07
	CHAOS-CTR	0.01	0.01
ASYMMETRIC	ECIES/AES	2.1	1.4
	ECIES/CHAOS	1.9	1.3

Meanwhile, (Ghehioeche, 2019) evaluated the performance of AES, Trivium, ECIES, RSA and McEliece algorithms. TOSSIM and AVRORA simulators were used to evaluate computational and power consumption, using MICA2 sensor. McEliece was implemented using NesC without optimization. Their work came in line with the result obtained in (Mansour, 2012) meaning that private key encryption algorithms are faster and consume less energy than public shared algorithms. Table 2 provides a summary of the work by (Ghehioeche, 2019).

Table 2: Energy Consumption in mJ (Ghehioeche, 2019)

	Algorithm	Encryption	Decryption
SYMMETRIC	AES	0.013	0.016
	TRIVIUM	25.4391	25.3159
ASYMMETRIC	ECIES	33.535	20.658
	RSA	44.012	43.249
	Mc Eliece	110.5004	-

### 3 EXPERIMENTS AND SIMULATIONS

Based on related work, we explored several possibilities to establish our own simulations and tests. Our main objective was to first find a way of calculating the Energy consumption of an algorithm regardless of initialization phases on the nodes part of the system.

#### 3.1 Environment

There are several simulators that can be used to simulate WSN IOT. (Mardiana binti Mohamad, 2019) 's review of related work mentioned MATLAB, NS2 &3, AVISPA, OPTNET, MICA2 and Contiki. Some of them are meant to simulate networks without really emulating sensors. We ended up working with Contiki because we could program several types of nodes as well as using a Java graphical tool to simulate (Cooja) and collect data. Contiki (Contiki, 2019) programs are written in C while the simulator requires Java. Contiki does not only allow simulation of sensor nodes, but also IOT devices.

Our experiments were conducted on i-5 3400Ghz 8Go RAM computer with Windows 7 (64bytes). We used a VMware virtual Machine based on Ubuntu 18.04LTS 64bytes version and 8Gb of RAM. We were also able to install two versions of Contiki and compared the results obtained. We used the following implementations of AES(Kokke, 2019), DES and 3DES(Ibeatu, 2019). It is also important to note that although there are several models of sensors available in Contiki Cooja simulator, we mainly worked with Skymote.

#### 3.2 Methodology

As we mentioned in the previous paragraph, we used two methods to run simulations. These methods are directly related to the version of Contiki used.

- Method 1 using Contiki 3.0: this method just requires to add or copy the header and c files (aes.h and aes.c for example), either at the root of the project or in the libraries' directory
- Method 2 using Contiki 2.7: here, there is need to first build a library using MSP430-GCC. This operation requires to build libraries for each type of node used

In the simulator, we created nodes implementing different encryption algorithms with different block sizes. We measured clock ticks at the beginning of the process (encryption or decryption) and calculated the difference (number of clock ticks executed to perform an operation). We did not set initialization phases (exchange of keys, vector) but rather considered that they were already available for the node. We tested different sizes of blocks (16, 32, 48 and 64 bytes) as well as various operational modes (ECB, CTR, CBC for AES in instance), and different key lengths (48, 128, 192, 256) depending on the parameters available for the algorithm. Each node implemented an encryption algorithm with specific parameters. For each type of node, 100 rounds were executed and the mean clock ticks of the round were used in order to compare the algorithms.

Table 3 summarizes the available algorithms, mode of operation, cipher block size and key length. Each of the entries of Table III has been implemented into a node and tested. We choose to present our results based on the number of clock ticks for each operation because, combined with the mote data sheet characteristics (voltage, current, and CLOCK TIME) the amount of energy consumed by encryption or decryption for a specific algorithm can easily be calculated.

Table 3: Encryption Algorithms (Mode, keylength, Size)

Algorithm	Mode	Key length	Block Size
AES	CBC	256,192,128	16,32,48,64
	ECB	256,192,128	16
	CTR	256,192,128	16,32,48,64
DES	Not available	48	16,32,48,64
3DES	Not available	48	16,32,48,64

### 3.3 Results and Discussion

It is important to note at this level that Method 1 is presented here because it generated quicker results. Method 2 gave similar observations with different values and will be presented at the end of the paper.

The first noticeable remark is the fact that using a specific mode and size of block, the key length has no effect on the calculation. It is illustrated with CBC Mode at 16bytes (figure 3) and 32 bytes (figure 4) block sizes, respectively. Similar observations were obtained at 48 and 64 bytes block sizes for CBC (not presented).

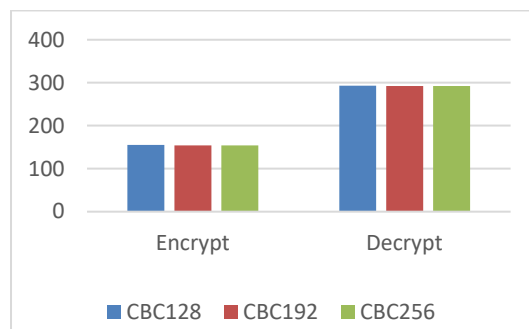


Figure 3: CBC results with 16bytes block size

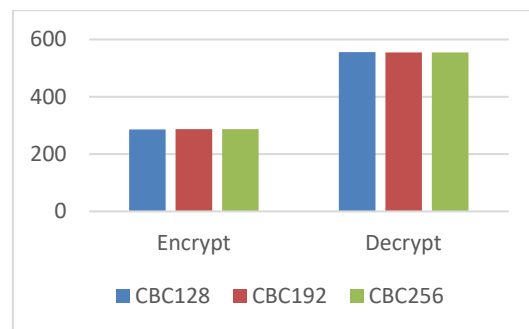


Figure 4: CBC results with 32bytes block size

Tables 4 and 5 show that similar measurements were obtained when we compared CTR and ECB modes with 128, 192 and 256 bytes key lengths respectively, for the same block size to encrypt and decrypt.

Table 4: AES CTR 32 bytes block size

	CTR128	CTR192	CTR256
ENCRYPT	290	291	291
DECRYPT	290	290	290

Table 5: Clock ticks (AES ECB 16 bits block size)

	ECB128	ECB192	ECB256
ENCRYPT	150	149	149
DECRYPT	286	286	286

It was also observed that for the same block size and key length, encryption with CBC, ECB and CTR use almost the same number of clock sticks (figure 5). On the other hand, when it comes to decryption of blocks, CTR Mode tends to use almost half of the value needed by CBC and ECB which were relatively close.

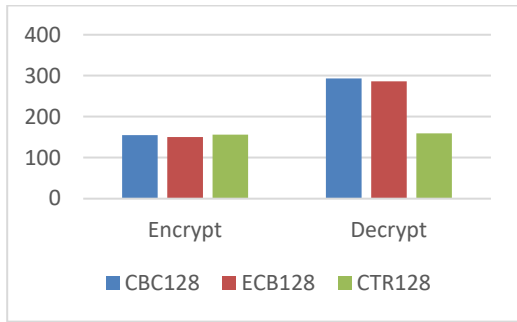


Figure 5: CBC, ECB, CTR results using 128 bytes key length and 16 block size.

As illustrated in figure 6, and conformingly with previous studies, DES and 3DES consume a far greater number of clock ticks than AES in various modes (16 bytes blocks). The tendency reported by studies using computers was clearly observed in sensor nodes.

Figure 7 illustrates the same conclusion using Method 2. The same huge difference between DES, 3DES and AES with the same size of block was observed. Thus, these two algorithms (DES, 3DES) are not suitable for Wireless Sensor Networks and IoT devices as they will consume more energy and subsequently reduce the lifespan of nodes in the network.

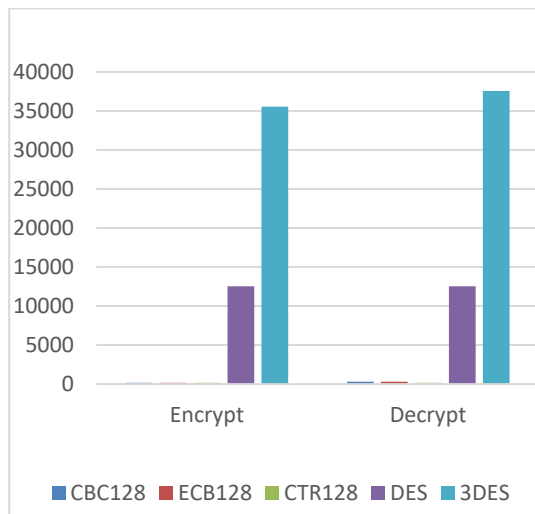


Figure 6: AES (CBC, ECB, CTR) vs DES/3DES for 16bytes block (Method 1)

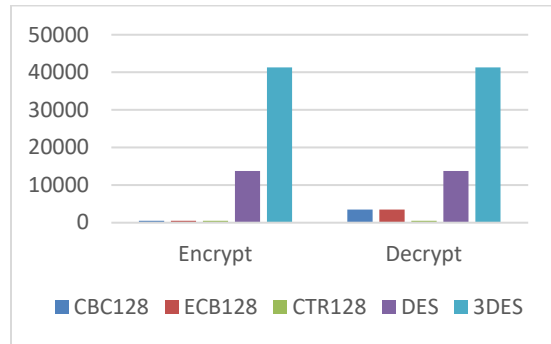


Figure 7: AES (CBC, ECB, CTR) vs DES/3DES for 16bytes block (Method 2)

Finally, comparing the two methods used in this study. It was noticed that Method 1 gave, for the same type of node (Skymote), better results compared with Method 2. The second method uses almost three times the number of clock ticks used in the first method (figure 8) and even more when deciphering a 16 block of data, except for the CTR Mode. Using a different type of node (for example wismote) also generated different results although the same pattern was maintained.

It was mentioned earlier in this study that CTR Mode consumed almost the same number of clock ticks for encryption and decryption. However, DES and 3DES results were not included in figures 8 and 9 as a conclusion was previously drawn compared to other algorithms.

At this point it is important to note that method 2 only works with targeted devices i.e a library has to be built for each targeted device and some incompatibilities might arise. For example, we were able to compile, simulate and take measurements with Wismote using method 1 while no solution was found for method 2.

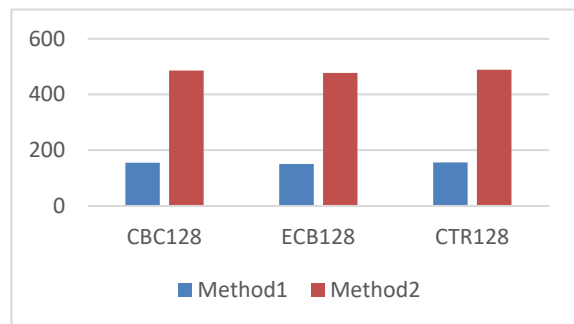


Figure 8: Comparison of Method 1 and Method 2 for (encryption)

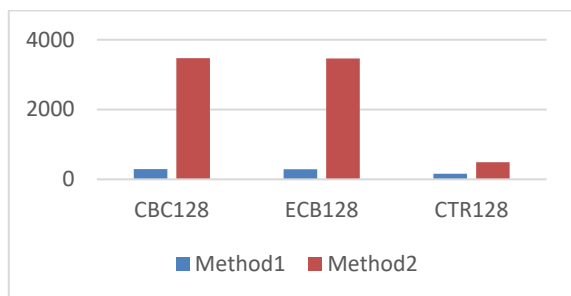


Figure 9: Comparison of Method 1 and Method 2 for decryption

## 4 CONCLUSIONS

Our work focused on evaluating the number of clock ticks required to perform encryption and decryption on wireless sensor nodes using different modes (ECB, CBC, CTR) for AES, DES and 3DES with different key lengths and block size of data (16, 32, 48 and 64). We observed that the length of the key does not have a significant impact on the consumed energy irrespective of the block size and the mode used or the method. As concluded by previous studies, DES and 3DES consume more energy compared with AES. CTR mode proved to be more efficient when it comes to decryption of data, compared with other AES modes. Depending on the method used (1 or 2), the results are different. Method 1 exhibited the capacity to be deployed on different sensor types of and consumed less energy while Method 2 requires the creation libraries for each type of nodes. We were unable to create libraries for other nodes other than skymote. Future works may include testing on real sensors, adding more encryption algorithms, introducing improvements made in IoT architectures compared with WSN and evaluating the performance of these methods on a network.

## ACKNOWLEDGEMENTS

The authors would like to thank SMARTiLab/ EMSI for support and material.

## REFERENCES

- Contiki Operating System. 2019 [C, Java, C++, Python]. Retrieved April 12, 2020, from <https://github.com/contiki-os/contiki/releases>
- Ghehiouche, A. A., Chikouche, N., Mezrag F., 2019. Performance Evaluation and Analysis of Encryption

- Schemes for Wireless Sensor Networks. In *International Conference on Digitization (ICD)*, Sharjah, United Arab Emirates, 2019, pp. 187-191
- Ibeatu, 2019. The TRIPLE-DES Algorithm Illustrated for C code. <https://github.com/lbeatu/The-TRIPLE-DES-Algorithm-Illustrated-for-C-code>
- Kokke, 2019. TinyAES Implementation (1.0). <https://github.com/kokke/tiny-AES-c>
- Kowalczyk, C., 2020. Block Ciphers Modes of Operation. CRYPTO-IT. <http://www.cryptoit.net/eng/theory/modes-of-block-ciphers.html>
- Mansour, I., Chalhoub, G. 2012. Evaluation of different cryptographic algorithms on wireless sensor network nodes. 2012 *International Conference on Wireless Communications in Underground and Confined Areas*. pp1-6.
- Mardiana binti Mohamad, N., Wan Haslina, H., 2019. Current research on Internet of Things (IoT) security: A survey. In *Computer Networks*, 148, pp 283-294.
- Meneghello, F., Calore, M., Zucchetto, D., Polese, M., Andrea Zanella, A., 2019. IoT: Internet of Threats? A survey of practical security vulnerabilities in real IoT devices. In *IEEE Internet of Things Journal*, 5(6), pp 8182-8201.
- Ray, P.P., 2018. A survey on Internet of Things architectures. *Journal of King Saud University – Computer and Information Sciences*, 30, pp 291-319.
- Rwan, M., Tasneem, Y., Fadi, A., Imran, Z., 2015. Internet of Things (IoT) Security: Current Status, Challenges and Prospective Measures. In *10th International Conference for Internet Technology and Secured Transactions (ICITST)*, pp. 336-341
- Sha, K., Wei, W., Yang, A., Wang, Z., Shi, W., 2018. On security challenges and open issues in Internet of Things. *Future Generation Computer Systems*, 83, pp 326-387.
- Tamimi, A., (2006). Performance Analysis of Data Encryption Algorithms. [https://www.cse.wustl.edu/~jain/cse567-06/ftp/encryption\\_per/](https://www.cse.wustl.edu/~jain/cse567-06/ftp/encryption_per/)
- Tankovska, H., 2020. Forecast end-user spending on IoT solutions worldwide from 2017 to 2025. Statistica. <https://www.statista.com/statistics/976313/global-iot-market-size/>