Design and Implementation of a Privacy Framework for the Internet of Things (IoT)

P. Panagiotou [I], N. Sklavos [I, II], I. D. Zaharakis [II, III]

[I] SCYTALE Research Group, Computer Engineering & Informatics Dept., University of Patras, Hellas

[II] Computer Technology Institute & Press – "Diophantus" (CTI), Patras, Hellas

[III] Computer & Informatics Engineering Dept., Technological Educational Institute of Western Greece, Hellas

***Abstract*—Nowadays, Internet of Things (IoT) is gradually finding its way, into everybody's life. This paper, introduces both the design and the implementation of a privacy framework, for the Internet of Things (IoT). Taking the benefits from it, a privacy platform is developed with a twofold aim: firstly, to train in detail the fundamentals of cryptography and privacy, and secondly, to make applied cryptography more interactive and flexible. Through this work, the proposed architecture and the integration of this open platform is presented, in detail. In addition, an application of symmetric cryptography is introduced, based on the Advanced Encryption Standard(AES) in various modes of operation, for both encryption and decryption of texts, images and electronic data files.**

Keywords—cryptography; privacy; Internet of Things (IoT); embedded systems; UDOO Kit

I. INTRODUCTION

With each passing day, data exchange through the internet, grows exponentially. The growing trend of new technologies use, and the extensive usage of devices, that provide access to a lot of services, such as smart phones, makes it easier for users, to share information, than ever before. As a consequence, privacy is an imperative need, for an increasingly growing digital world, where adversaries exist. Internet of Things (IoT) aims to build a secure network for every device, that communicates with other ones. In addition, IoT can be used, to implement, lots of encryption schemes and security models [1].

Embedded systems are aimed to special purposes, mainly designed to implement a number of specific tasks and procedures. On the other hand, microcontrollers (μCs) are certainly the cornerstone of embedded systems. They do provide numerous benefits, for the majority of applications and services. Some of these benefits are the cost-effectiveness and portability due to their compact size. Furthermore, μCs can be used for various applications, due to their efficiency in programmability and their flexibility. All of these, lead to the conclusion that, an embedded system implementation, based on μCs can serve users, by improving the applications, that they need.

IoT could be widely used as a training model, for training and education purposes, replacing traditional and conventional education processes, with aim to a more interactive, flexible and future technology. Specifically, this paper takes advantage of the benefits that IoT technologies, currently offer. A training platform for privacy applications for IoT, is introduced. This platform attempts to reshape the practices of training, and is beneficial for students, researchers and industry experts. It provides an alternative training experience, by changing the traditional way of privacy applications, through data texts and images. It is based on an interactive way of training. Also, this work is focused on cryptography and security applications using, a full IoT implementation system [2]. A full IoT platform is proposed using the UDOO NEO board [3]. The proposed IoT system implements cryptographic primitives, such as symmetric encryption with different scenarios, that are provided, and at the same time, the way they are applied to different data services. It is based on Advanced Encryption Standard (AES), in different modes of operation [4].

Finally, the proposed platform could be enriched and assembled, by user's desires and expectations, in order to offer the ability to be customized and expanded.

The paper continues with a brief introduction to IoT & Education Primitives, in Section II. Section III provides the necessary background theory, regarding symmetric cryptosystems. Continuing to Section IV, a full IoT platform is introduced. In Section V, the architecture of the proposed system, is presented in detail. The designed architecture of this platform is analyzed and the peripherals are studied. Finally, in Section VI, the platform is introduced, with the use of Advanced Encryption Standard (AES). Conclusions and future work, are discussed in the last section.

II. INTERNET OF THINGS IN TRAINING & EDUCATION

Internet of Things (IoT), as mentioned previously, is connecting every physical object (e.g. an electronic device, like smart phones, printers, smart watches etc.), with devices of similar or alternative technologies. It can be used, in a number of various applications, from personal to industrial needs such as smart homes, agriculture, and smart cities. IoT's rapid expansion to a variety of fields, can help overcome problems

978-1-5386-7377-5/18/$31.00 ©2018 IEEE
DOI 10.1109/DSD.2018.00103

encountered in day to day life, by providing and improving solutions.

Moreover, connecting recent technologies can enhance training, education, learning and pedagogy. The applications of ubiquitous computing, mobile computing and IoT, can contribute further in science, technology, and engineering training, by offering novel educational services and implement innovative ways [5]. Furthermore, a variety of tools can be developed, and be offered to professors, students, and industry parties, that serve quality content in any device, and anywhere. Moreover, connecting different technologies together, leads to scalable, customizable and expanded content.

More and more people have access to devices, that are widely connected and can be alternatively used. Training methodologies could contribute to the reduction of operating costs, in both schools and universities, since the use of stationary is minimized. The training process is completed through devices, that are reusable or already held by students (smartphones, tablets etc.). Along with this, developed platforms can be built with low-cost devices, and connected with current and future technologies. This novel practice can turn learners into creators and connect people all over the world, with a learning environment platform, that is compatible, stable and safe.

III. Cryptography and Privacy Aspects

Current section provides the background theory of symmetric cryptography, as well as the different modes of operation, concerning symmetric block ciphers, for both encryption and decryption processes.

A cryptosystem is called symmetric, when the same key is used for both encryption and decryption, processes. Figure 1 presents a communication model of a symmetric cryptosystem. The sender is encrypting the plaintext, using an encryption algorithm and a key. The receiver decrypts the ciphertext using the decryption algorithm and the same key. This model offers confidentiality in terms of data, provided that key exchange is done correctly before, using a secure channel. An adversary can monitor or alter the data, but is computationally unfeasible to decrypt the data with the known attack methods, assumed that a large key is used.

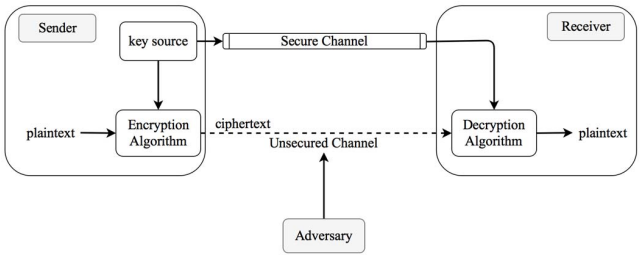

Fig. 1. *Communication Model of a Symmetric Cryptosystem*

A symmetric block cipher processes data, in a block-by-block, way of operation. Every block has a fixed size of n-bit (e.g. AES uses several values of n-bit, such as n=128-bit). Modes of operation connect symmetric block ciphers, if the plaintext contains more than one n-bit block(s). In this work, two modes of operation are presented and implemented; Electronic Code Book (ECB) and Cipher Block Chaining (CBC) mode [6]. Advanced Encryption Standard (AES) is implemented for the symmetric block cipher mode, that transforms data blocks of n=128-bit [4].

Electronic Code Book (ECB) mode, illustrated in the following Figure 2, breaks the plaintext into i-blocks. In the next step, it encrypts each one of them, individually. Each plaintext-block p_i is encrypted (E), independently from others, using a key and produces a ciphertext-block c_i. The decryption process replicates the inverted encryption process and each ciphertext-block c_i is decrypted (E^{-1}) using a key and produces the plaintext p_i'.

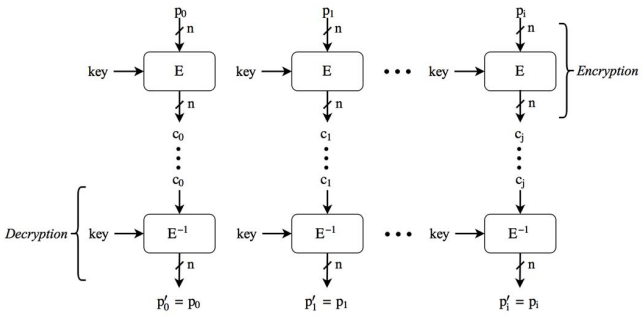

Fig. 2. *Encryption and Decryption process using ECB mode*

Cipher Block Chaining (CBC) mode is illustrated in Figure 3. At the beginning of encryption process, it uses an Initialization Vector (IV) of n-bit, which is XORed (exclusive OR function - XOR), with the plaintext-block p_0. Then this result is encrypted (E) producing the ciphertext-block c_0. Continuing the encryption process, each plaintext-block p_i is XORed with the encryption of the previous block p_{i-1}. The decryption process (E^{-1}) of a ciphertext-block c_i, is XORed with the previous ciphertext-block c_{i-1}, and the p_i' is produced. Again, the IV is used only once at the beginning of the decryption process, using the XOR operation, with the decryption of ciphertext-block c_0.

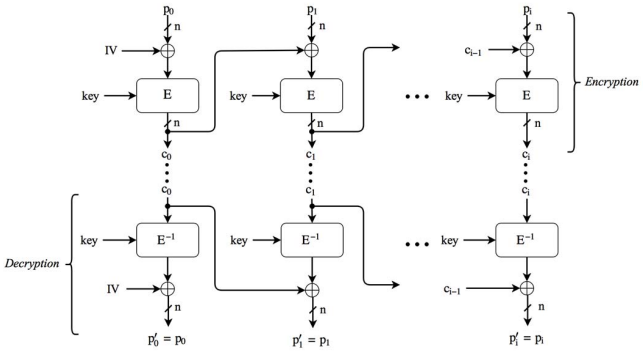

Fig. 3. *Encryption and Decryption process using CBC mode*

For both modes of operation, if p_i' is equal to p_i, for every plaintext block, then the decryption procedure is completed successfully, meaning the same key was used for both encryption and decryption processes. However, ECB mode has a major disadvantage. Since blocks are encrypted independently, the encryption of identical plaintext-blocks, using a key, produces identical ciphertext-blocks and vice versa. As a result, ECB mode must not be used for messages greater than one block, or if keys are reused. CBC mode overcomes this disadvantage, thanks to the chaining mechanism that causes ciphertext c_i to depend on p_i and all preceding plaintext blocks. Also changing the IV, same key and first plaintext block, results to a different ciphertext. An example using the proposed platform is provided in Section VI, in order to prove ECB disadvantages and the ways that CBC mode implementation, overcomes them.

IV. Selected IoT Platform

In order to integrate the training platform, a full IoT implementation, based on UDOO Neo board [3], is applied. Indeed, it is a single-board computer launched latest, in 2015, that is Arduino-powered with Android or Linux OS, enhanced with 9-axis motion sensors, Bluetooth 4.0 and a Wi-Fi module. It is primarily developed for training purposes, where new applications, services and projects can built-up with convenience and minimum knowledge requirements, using this low-cost system. Its use, may create a well-trained team of developers, designers and engineers. As a result, institutions and companies may have a useful tool for high-standards implementations [2], [5].

Specifically, UDOO Neo uses an NXP/Freescale iMX 6SoloX processor, that has two heterogeneous processors embedded on the same chip [7]. An ARM Cortex-A9 and an ARM Cortex-M4 embedded processor, that clock at 1 GHz and 200 MHz, respectively. The two processors communicate through a virtualized serial interface, that uses the shared memory to exchange data. The processors, also, share hardware implemented features, provided by the architecture. The iMX 6SoloX connected to peripherals such as 9-axis motion sensors (accelerometer, magnetometer and gyroscope), Bluetooth 4.0 and Wi-Fi module etc. The peripherals are connected to processors, through a high speed AXI bus, using different hardware interface (I2C's, SPI, GPIOs, UARTs and others).

All UDOO Neo hardware features, can be accessed and connected via processors pad, with an editable mixing. Thus, the functions are not fixed, but can be accessed on different pads. Some of these are connected to the external pins, for both processors, allowing the users to connect their own peripherals to expand the system as they wish. General Purpose Input/Output (GPIO) pins, can be dynamically shared, at boot time, between the Cortex-A9 and Cortex-M4 processors.

V. Proposed System Design

In this section, a proposed system using UDOO Neo board, is introduced in detail. The proposed system is using the operating system UDOObuntu 2, based on Linux Ubuntu 14.04, without any Graphical User Interface (GUI). Figure 4 illustrates the architecture. The Wi-Fi module of UDOO Neo board is configured, as an access point with protocol IEEE 802.11 b/g/n. The device can handle up to 10 clients to this mode.

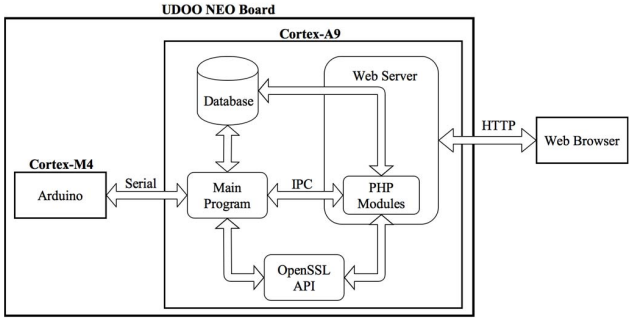

Fig. 4. *Software Architecture*

The UDOO Neo device is configured as a Web Server (Apache and PHP modules), that can handle Hypertext Transfer Protocol (HTTP) requests, from the user's Web Browser. A user is connected to the access point of the device, and communicates through Wi-Fi protocol. Through the web browser, a user is accessing the interface that is developed with web technologies such as HyperText Markup Language (HTML), Cascading Style Sheets (CSS) and JavaScript programming language.

The Application Programming Interface (API) is based on the user's HTTP requests that may contain information about the interface, encryption or decryption requests. All cryptographic implementations are produced using the OpenSSL API [8] with C/C++ programming language. An encryption or decryption request is first sent to OpenSSL module programs. Then, it waits for cryptographic results, either synchronous or asynchronous, which depend on the cryptographic execution time. A Python interpreter provides the main function of the tool, which is running a threaded program. It can handle encryption or decryption commands with the help of OpenSSL, saving and sampling data and other operations. The Python program is communicating through serial interface to Cortex-M4, and through Inter Process Communication (IPC), to Web Server PHP module. The Web Server and the python program use a database for data storage.

Cortex-M4 is running an Arduino code (a simplified version of C), responsible for handling all external peripherals. For example, in order to report sensor data and display the appropriate messages to screen, a button is pushed and change the status of RGB leds indications. All interaction and reporting, are then passed through bidirectional serial communication, to the main function of the tool. Then they are passed to a Web Server, if information must be sent to the user.

Figure 5 and 6 illustrate the current development of the training platform, using UDOO Neo board and peripherals. The board's peripheral is a TFT Color Display ILI9163 [9], connected through a Serial Peripheral Interface (SPI) bus and a DHT11 humidity and temperature sensor [10], connected through a serial interface using a single-wire two-way protocol.

In addition, an RGB led is connected to output pins with Pulse Width Modulation (PWM) mode. The two push-buttons are connected to input pins, with pull up resistors and handled by interrupts.

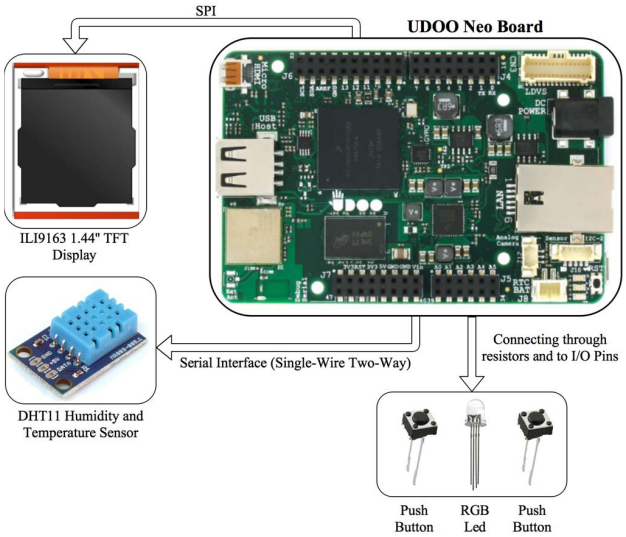

Fig. 5. *Proposed System*

The TFT display is used mainly to inform users about pending processes, such as the display a one-time password or to inform about account validation. Humidity and temperature sensors are sampled, by the system. These sampling data are stored as encrypted ones.

The proposed platform also integrates two-factor authentication login, by the user, each time.

The RGB led is handled by the user's web browser. The first push-button is used for validation, while the second for invalidation of the account.

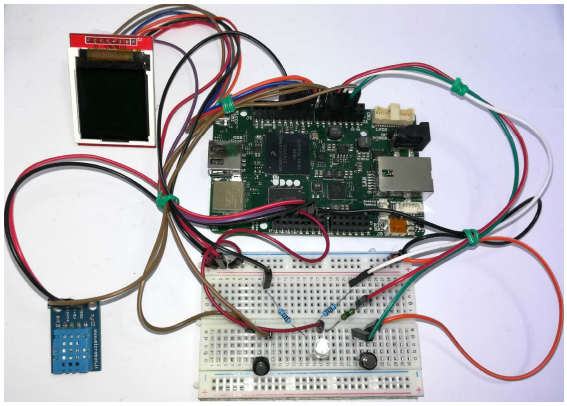

Fig. 6. *Proposed Platform Integration*

VI. Proposed IoT Privacy Implementation

In this section, the supported services of the proposed privacy implementation (IoT device), are presented. Firstly, a user device is connected to the IoT one, through the Wi-Fi access point and a passphrase. Then, the user uses a web browser to gain access to the services.

After that, the user must register an account, using the preferred e-mail address and a desired password. At this phase of operation, a physical validation process occurs. The user must validate or invalidate the account physically, using the two push-buttons of the IoT device. After correct validation, a two-factor authentication (2FA) process must be performed, in order to gain access to the services. At the first stage, the user must login with its correct credentials. At the second stage, the IoT device generates randomly an One Time Password (OTP), equal to four characters long, that is being displayed on the device screen. Also, the sensor data (temperature and humidity) are shown as encrypted, using AES in CBC mode of operation. The user must provide this OTP in order to gain access to the main system and be able to have access to the decrypted data.

While accessing the main interface of the system, the user can perform symmetric encryption and/or decryption, as analyzed in Section III, using the applied Advanced Encryption Standard (AES) [4], on three different types of applications:

1. *Text: User input text.*
2. *Image: Image known file formats (bmp, png, jpeg etc.).*
3. *Files: All other data file formats.*

A user can select from a various set of privacy schemes and perform encryption and/or decryption, at its own choices of parameters and operations. The supported privacy schemes by the proposed system are the following ones:

1. *Modes of Operation (ECB, or CBC)*
2. *Key length (128-, 192-, 256- bit)*
3. *Padding (No padding, PKCS#7 padding [11])*
4. *Password (Key is generated using a Key Derivation Function – KDF)*
5. *Key – using hexadecimal notation. The size depends on key length*
6. *IV (only in CBC mode) – using hexadecimal notation and is equal to block size (16-byte).*

The following application scenario (based on the proposed system supported services) is presented in detail, in order to demonstrate and exploit the main disadvantages of ECB mode.

The encrypted process of an image is performed as follows. The image is converted into a BMP file format, lossless, and the header is extracted. Then the body is encrypted using AES and the user's selected privacy schemes. Last, the encrypted body is concatenated with the extracted header and the image is illustrated. A user can encrypt the example image, shown in Figure 7, represented on the left, using the two different modes of operation. This image is the official Linux image called "Tux". The converted image has a fixed header of 138 bytes and a body of 1866400 bytes. The tool using AES will encrypt 1866400/16 bytes (block size) = 1166500 blocks, thus no padding is needed for the selected image.

First, the image is encrypted using the following parameters: AES encryption algorithm, ECB mode of operation, and ey length equal to 128-bit. No padding is selected, the vector of the applied key in hexadecimal notation is equal to: "00000000000000000000000075BCD15". The encrypted image using ECB mode is illustrated on the right part of Figure 7.

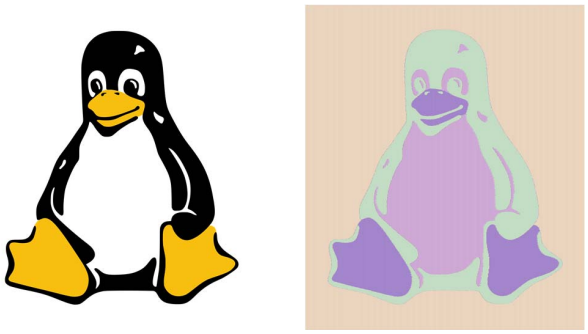

Fig. 7. *(left) "Tux" original image, (right) Encrypted image using AES-ECB mode*

Since the image has the same pattern of data, the encrypted blocks of data, follow this pattern and exploit the image. The key length and the password phrase, or key, are not codependent. If a key with larger vector length or a different key is applied, the image will be also exploited. In addition, multiple encryptions of the image will not encrypt the image correctly, as illustrated on the left part in Figure 8, where the image is encrypted three times using the previous AES parameters, $AES_{ECB}(AES_{ECB}(\,AES_{ECB}(image)))$. As a conclusion, an adversary can identify the image that is behind.

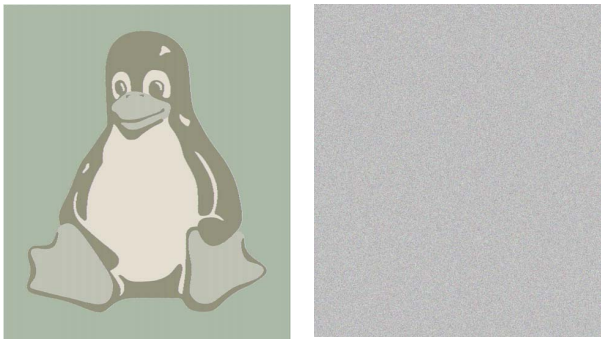

Fig. 8. *(left) Multiple encryptions using AES-ECB mode, (right) Encrypted using AES-CBC mode*

The user can encrypt the image and select the same parameters set, by changing only the mode of encryption operation from ECB to CBC. The key is the same as before and the IV is selected to be all zeroes. On the right side of Figure 8, the encrypted image of CBC encryption mode is illustrated. CBC mode encrypts the image successfully by removing all previous patterns, that exploit the image due to the chaining dependency mechanism, as mentioned in Section III. An adversary cannot identify the image, that has been encrypted.

Using the proposed training platform, another high-quality image is encrypted as second example, with lots of different pixels, illustrated on the left part, of Figure 9. Encryption process is performed with AES in ECB encryption mode, using the same applied key, as in the previous example. The encrypted image of this high-quality image is demonstrated on the right part, in Figure 9. As observed with the naked eye, the image that it is encrypted behind, cannot be identified. Examined more closely in the encrypted image, a number of patterns exist too. If the encrypted image is observed in byte level, an adversary can discover same ciphertext blocks that finally lead to identical blocks of original image. Again, CBC encryption mode can give a solution to this security issue, as presented in previous example.

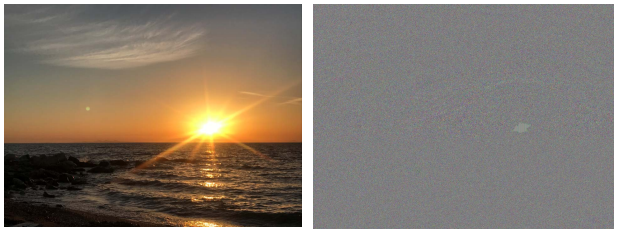

Fig. 9. *(left) original high quality image, (right) encrypted image using AES-ECB mode*

Both encryption and the decryption of ECB mode, can be easily parallelized using multiple AES cores. On the other hand, the encryption of CBC mode cannot be done in parallel, because in order to encrypt a plaintext block, depends on the encryption of previous plaintext blocks. This chaining dependency cannot be done in parallel, thus only the AES cipher can be parallel. In contrast, the decryption process of CBC mode can be parallelized due to the non-dependency of previous decrypted blocks.

Figure 10 demonstrates the number of clock cycles, that the UDOO Neo board needs on encrypting blocks, between non-parallel versions of ECB versus CBC mode. In x-axis, the number of blocks that will be encrypted is shown. In y-axis the clock cycles that spend on encrypting those blocks are presented. As the graph illustrates, the CBC mode needs more clock cycles for encrypting more blocks, versus the ECB mode that spends a fewer number of clock cycles. The CBC mode performs, for every encryption of a plaintext block, one more function, the XOR, thus more clock cycles are needed.

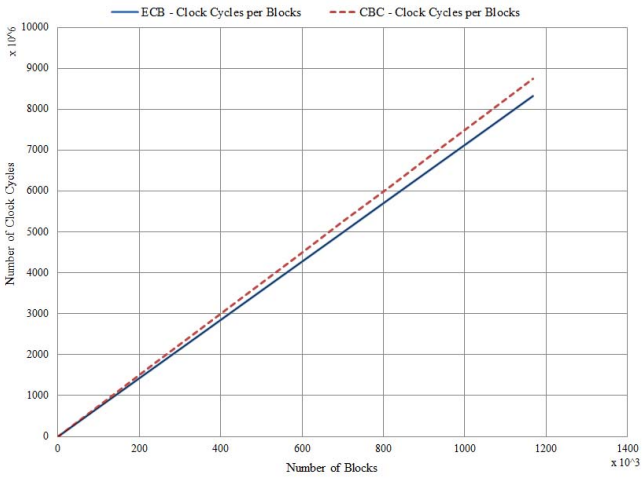

Fig. 10. *ECB vs CBC non parallel encryption – Clock Cycles per Blocks*

If we parallelize the encryption of ECB mode, then the results will be different. Figure 11 illustrates a parallel version of ECB versus CBC mode (that cannot be parallel in encryption process). As the graph presents, for a small number of blocks, ECB is being paralleled successfully and the clock cycles of encrypting these blocks are constant. However, CBC mode cannot be parallel and spends much more clock cycles as before.

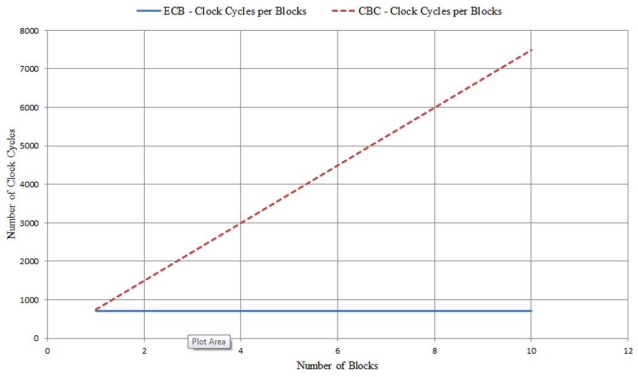

Fig. 11. *ECB vs CBC parallel encryption – Clock Cycles per Blocks*

VII. Conclusions & Future Work

To conclude, a training platform using IoT technology can facilitate users to learn in a more interactive way, connecting the physical and virtual world together. As it has been explained in detail through this work, this is achieved through the use of a full IoT platform, that offers numerous advantages.

The proposed system could be extended to a more complicated design; the objective could be to expand the system by implementing other cryptographic primitives, like Public Key Cryptography, Hashing, Digital Signatures and Cryptographic Pseudo-Random Number Generator (CPRNG). Additionally, this system should provide a connection to multiple boards using the same platform technology, for information exchange and suggest solutions to different security problems.

Acknowledgment

This work is under the UMI-Sci-Ed project. This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 710583.

References

[1] N. Sklavos, I. D. Zaharakis, "Cryptography and Security in Internet of Things (IoTs): Models, Schemes, and Implementations", IEEE proceedings of the 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS'16), Larnaca, Cyprus, November 21-23, 2016.

[2] N. Sklavos, I. D. Zaharakis, A. Kameas, A. Kalapodi, "Security & Trusted Devices in the Context of Internet of Things (IoT)", IEEE proceedings of 20th EUROMICRO Conference on Digital System Design, Architectures, Methods, Tools (DSD'17), Vienna, Austria, August 30 – September 1, 2017.

[3] UDOO NEO Board, 2018, Full Specifications, Available At: https://www.udoo.org/ .

[4] Advanced Encryption Standard (AES), FIPS Publication 197, National Institute of Standards and Technology, November 2001.

[5] I. D. Zaharakis, N. Sklavos, A. Kameas, "Exploiting Ubiquitous Computing, Mobile Computing and the Internet of Things to Promote Science Education", IEEE proceedings of the 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS'16), Larnaca, Cyprus, November 21-23, 2016.

[6] NIST Special Publication 800-38A, 2001 ED, Version 1, Recommendation for Block Cipher Modes of Operation—Methods and Techniques, December 2001, Natl. Inst. Stand. Technol.

[7] NXP/Freescale iMX 6SoloX, "i.MX 6Solo/6DualLite Applications Processor Reference Manual", 2017.

[8] OpenSSL - Cryptography and SSL/TLS Toolkit 2018, available at: https://www.openssl.org/.

[9] ILI9163 1.44 inch TFT LCD Display, a-Si TFT LCD Single Chip Driver 132RGBx162 Resolution and 262K color, Documentation, 2018, available at: https://www.rockbox.org/wiki/pub/Main/SonyNWZE370/ILI9163.pdf

[10] DHT11 Humidity and Temperature Digital Sensor, Documentation, 2018, available at: https://www.microbot.it/documents/mr003-005_datasheet.pdf

[11] "RFC 5652", 6.3. Content-encryption Process, September, 2009.