International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 12, Number 11 (2019), pp. 1950-1956

Secure Information Transmission Device Implemented on an Embedded System using 3DES and AES Algorithms

Ricardo Martínez Santa[1], Fernando Martínez Santa[2] and Holman Montiel Ariza[3]

[1]*Universidad Internacional de la Rioja en México, Ciudad de México.*

[2,3]*Facultad Tecnológica, Universidad Distrital Francisco José de Caldas, Bogotá D.C, Colombia.*

[1]*ORCID:* 0000-0001-5331-2954 [2]*ORCID*: 0000-0003-2895-3084
[3]*ORCID:* 0000-0002-6077-3510

Abstract

The main objective of this article is to implement a hardware-software prototype for sending and receiving secure information between a PC and an embedded system, using cryptographic algorithms 3DES (Triple DES) and 3 different versions of AES. Likewise, a comparative study is carried out between the implemented algorithms, in terms of the execution times of the encryption and decryption operations, both in the PC and in the embedded system, in order to select the cryptographic algorithm that shows the best performance measures. Finally, it is determined that the algorithm with the best relation between total execution time and security level is the AES128, this having a much lower execution time in the implementation carried out in the PSoC 6 embedded system than in the PC. However, the implementation of the 196-bit and 256-bit versions of the AES algorithm can be done without significantly increasing the total algorithm execution time on both devices.

Keywords: Cryptography, AES, 3DES, Embedded Systems.

1. INTRODUCTION

Currently, the area of cryptography has taken a new momentum given the rise of IoT (Internet of Things) and the possibility of providing security to data that are handled to this type of applications. Generally, these applications are implemented through embedded systems based on microcontrollers, low power consumption microprocessors and/or reconfigurable digital systems (basically FPGAs). Because such embedded systems have computational and speed limitations compared to a PC, new so-called "light" algorithms have been developed such as those presented by [1,2] and other "ultra-light" algorithms such as those shown in [3,4].

Such algorithms can run on embedded systems with a lower or even much lower computational load than standard cryptographic algorithms such as AES. Countless applications have been made on reconfigurable digital systems, some of them making use of softcores (processors described in languages such as VHDL or Verilog) such as the one shown in [5], but the applications that achieve greater performance in terms of execution times are those that are implemented directly on the device (generally an FPGA) without the use of a processor such as [6]. Given the speed levels reached by this type of implementations it is even feasible to develop classic algorithms such as DES [7], Triple DES or 3DES [8,9] and AES128 [10-12]. These implementations turn out to have great features, but they do not always turn out to be the most viable from the economic and development time point of view for IoT applications. On the other hand, there have also been applications of cryptographic algorithms implemented on microcontrollers such as those exposed in [13-17], which present lower performance in terms of execution times compared to FPGA applications, but sufficient for most applications of the IoT and at the same time economically viable. Another hybrid approach used recently is the use of SOC (System On Chip), which are devices that combine both worlds: microcontrollers and/or microprocessors, with reconfigurable logical devices such as FPGAs, using digital blocks as hardware to accelerate basic cryptographic operations managed from the central processor, some of the applications found make use of PSoC microcontrollers such as [18-20]. It is precisely this hybrid approach, which is the focus of the research developed in this work, which is based on a PSoC 6 microcontroller connected to a computer application developed in the Python programming language.

2. MATERIALS AND METHODS

Python's pyCryptodome library was used as it is supported and continuously updated. Then the cryptographic algorithms for blocks TDES, AES128, AES192 and AES256 were implemented by means of pyCryptodome verifying their correct operation. After this, versions of these algorithms including time measurement functions were implemented and time sampling was performed for statistical analysis. After this, the same algorithms were implemented in C language, making use of the acceleration hardware available in the PSoC microcontroller. Also included were the functions of time measurement and subsequent sampling. The next step was to analyze the statistical data to evaluate which of the evaluated algorithms to determine the algorithm to implement in the final application. Finally, the AES128 algorithm was implemented in an application that turns on and/or off digital outputs (in this case LEDs) of the embedded system sending encrypted commands.

2.1 Approach to the encryption algorithms to be evaluated

The prototype implemented consists of a secure information exchange system between a master PC and a slave embedded system connected by USB, designed for IoT applications and/or intelligent sensor networks. Data security is implemented through the Triple DES algorithm (Data Encryption Standard) and three different versions of AES (Advanced Encryption Standard), using the Python programming language on the PC and the pyCryptodome library. At the same time the embedded system used is the CY8CKIT-062-BLE of Cypress Semiconductor, card based on the series of microcontrollers PSoC 6 which has a hardware-cryptography software module.

2.2 pyCryptodome Library

The pyCrytodome library is a collection of standard cryptographic algorithms as well as functions usually used in cryptography, among which can be found block and data flow encryption algorithms, symmetric and asymmetric encryption algorithms, algorithms for password validation, algorithms for error detection and correction, generation of pseudo-random numbers, etc. This library is a modern version of pyCrypto included in the PyPI (Python Package Index), which was left some time ago to give support and provide updates.

2.3 Microcontroller PSoC 6

The PSoC family of microcontrollers from Cypress Semiconductor have the particularity of having, in addition to the microprocessor as such, reconfigurable analog and digital devices. These devices (specifically digital ones) behave in a similar way to SPLDs, CPLDs or FPGAs and can even be reconfigured by means of hardware description languages such as Verilog. This feature makes this family of microcontrollers very flexible, since not only the microprocessor but also the logic circuits can be programmed. Specifically, the PSoC 6 series has a dual processor core (an ARM Cortex M0+ and an ARM Cortex M4) and a large number of reconfigurable analog and digital modules within which there is a generic cryptography module, which accompanied by software libraries is able to implement standard cryptographic algorithms such as RSA, AES, SHA, and so on.

The implemented application consists in turning on or off the logic outputs of the embedded system microcontroller from the PC, sending secure commands (using AES128) through USB, having the PC and PSoC 6 sets the same password of 128 bits (16 Bytes). The simulation of these outputs is carried out by means of the LEDs connected to the embedded system: one red, one orange and one RGB. A total of 12 commands were generated with the format:

- LED {Orange, Red} {ON, OFF}
- RGB LED {Red, Green, Blue, Yellow, Cyan, Magenta, White, OFF}

The program written in Python, receives the command typed by the user and compares its extension in number of characters, since the AES algorithm uses data blocks of 128 bits (16 Bytes), the program completes or truncates the command entered to 16 ASCII characters. The entered command is then encrypted with AES128 and sent via USB to the embedded system. Once the message is received by the embedded system, it is decrypted and recognized, if it is a valid command: turn off, turn on and/or set the required color. Then it establishes a response for the PC depending on the action executed. The possible responses of the microcontroller are in the following format:

- LED {N., R.} {Off, on}
- RGB {off, in Red, in Yellow, in White, in Blue, in Magenta, in Green, in Cyan}
- invalid command

The response is encrypted using the same key as the received command and sent back to the PC. When the answer arrives, the program in Python decrypts it and displays it on the screen. The python program calculates the total execution time of a valid command including command encryption, command sending and receiving, command decryption, command execution, response encryption, response sending and receiving, and response decryption. The times obtained can be seen in the results analysis section.

3. EXPERIMENTAL RESULTS

In order to determine the effectiveness of the implementation of cryptographic algorithms in the CY8CKIT-062-BLE embedded system, it was compared with the implementation done entirely on the PC using Python and pyCryptodome. The comparative study is based on the measurement of execution times of four symmetrical cryptographic algorithms by blocks: 3DES (TDES), AES128, AES192 and AES256. For each algorithm, the initialization (initialization and key expansion), encryption and decryption times were measured, both in the PC implementation and in the embedded system (see characteristics in Table 1).

Table 1. Characteristics of the devices used

Characteristics of the processors used		
Platform	***Embedded System***	***PC***
Reference	CY8CKIT-062-BLE	ROG GL553VD
Processor:	CY8C6347BZI-BLD53 (ARM® Cortex® M0+/M4)	Intel® Core™ i7-7700HQ
Clock frequency:	100 MHz	2.8 GHz
RAM	288 KB	12 GB

For this purpose, a statistical experiment is proposed, obtaining samples of these execution times. Given that the total statistical population is not defined for the proposed experiment, a non-probability type of sampling is applied for each group of samples obtained, carrying out a univariate analysis.

As the sample is of a non-probability type, no specific valid sample size is defined, therefore an arbitrary sample of 50 data

is defined. For the experimental test, different time samples were taken on 15 non-consecutive days, under different processor load conditions (in the case of PCs). For the embedded system, time samples were taken for only 5 days due to the low standard deviation presented in the first tests (see section on analysis of results). Each day of sampling, 50 times samples were taken for each function (initialization, encryption and decryption) and for each device (PC and embedded system). In total 750 samples were taken for each function of each algorithm implemented in the PC and 250 samples for each function of each algorithm implemented in the embedded system.

3.1 Statistical analysis of the results

In order to determine the trend values of the execution times of the implemented algorithms, both in the embedded system and in the PC, for each sampling day (group of 50 samples) carried out, the mean, standard deviation and median were calculated. Finally, these data were totaled as the averages of all the sampling days. The measured execution times were initialization time (as the key is set and expanded), encryption time, and decryption time. First, we analyzed the times for the algorithms executed in the PC and implemented in Python, making use of the *timeit* function of the *time* library, executing only the required function. Fig. 1 shows the mean and standard deviations of the encryption time, for all implemented algorithms, totaling the 15 days of sampling.

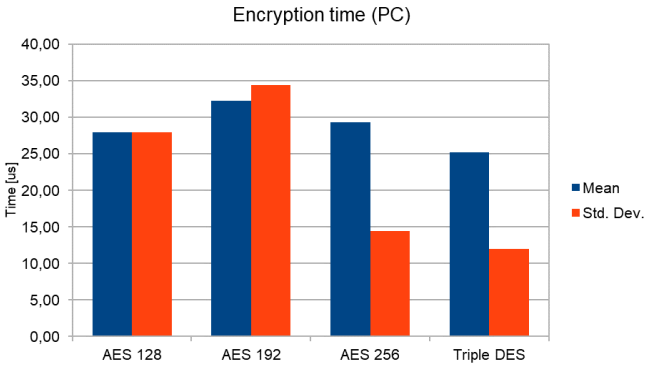

Fig. 1. Mean and standard deviation of the encryption time of the algorithms executed in the PC.

As can be seen in Fig. 1, for the encryption time in all algorithms, the standard deviation is too large with respect to the mean, the same happens with the initialization and decryption times. These values range from 33% for the AES192 initialization time to 111% for the Triple DES initialization time. These standard deviation values indicate that the samples show excessive variation, which implies the existence of samples well above or below the mean value. This is fully understandable, bearing in mind that each encryption algorithm tested runs depending on the operating system of the PC, i.e. the time measurements on the PC are not exclusive to the encryption algorithm. For this reason, the use of the mean as the central trend value of the sampled data is discarded, opting for the use of the median for this purpose.

Likewise, in the implementation on the embedded system, the times measured were the initialization, the encryption and the decryption. For this case, we used a hardware digital *timer* or counter of 16 bits working upwards at a sampling frequency Fs = 20MHz, which can count time periods of 0.5 µs, this time Ts, can be changed given the equation (1). The maximum time to measure depends on the number of bits of the *nbits* counter (16 but can be changed to 32) as shown in equation (2), which for the case of this application will be approximately 3.3 ms.

$$T_s = 1F_s \quad (1)$$

$$T_{max} = 1F_s(2^{nbits}) \quad (2)$$

Fig. 2 and Fig. 3 show the averages and standard deviations some of the times measured on the algorithms implemented this time on the embedded system based on microcontroller PSoC 6, totalizing the 5 days of sampling. In this case, the results of the AES algorithms (128, 192 and 256) and the Triple DES algorithm were separated, because in the latter the initialization time is immersed in both the encryption time and the decryption time.

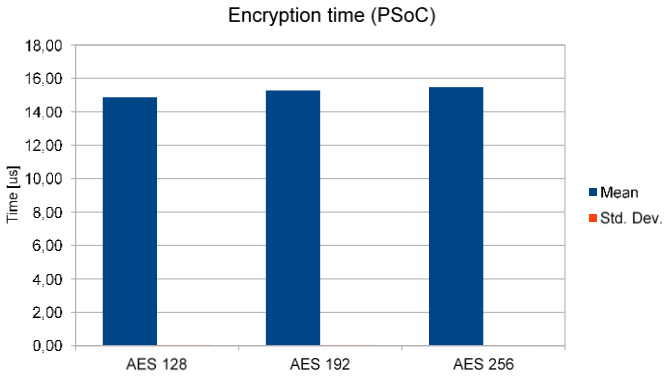

Fig. 2. Mean and standard deviation of the encryption time of the AES algorithms on the PSoC.

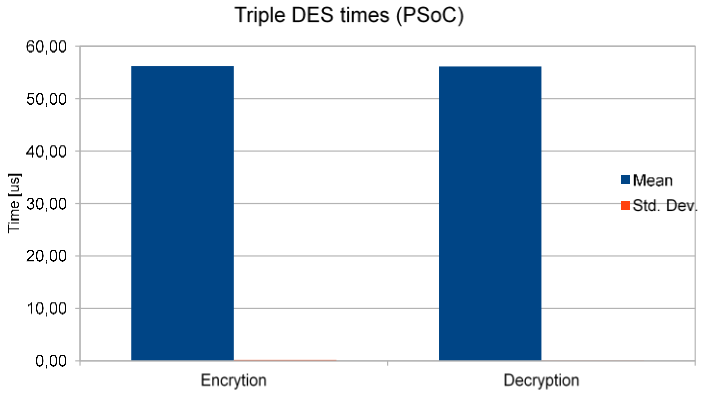

Fig. 3. Mean and standard deviation of the encryption and decryption time of the Triple DES algorithm on the PSoC.

As can be seen in Fig. 2 and Fig. 3, for all measured times and all algorithms, the opposite happens to the results of the implementation of the same algorithms in the PC, the standard deviation is too small with respect to the average, being almost imperceptible in the bar graphs shown in the figures, and reaching up to 0.018% in the case of decryption time Triple DES. This means that the sampled values have a minimum variability between them, which implies that the times are quite consistent. The latter validates the decision to take only 5 days of sampling instead of 15 as in the PC software implementation, given that the data obtained would remain "the same". This little variability of the data can be explained first, by the fact of using a hardware timer for time measurement, which avoids the use of a routine in the same processor for this measurement and second, because the implementation in the embedded system does not have any operating system and is dedicated exclusively to USB communication with the PC and the execution of the cryptographic algorithm, that is to say that there is no other task or programming thread that is executed while an encryption task is being performed modifying the execution times.

Returning to the execution times of the algorithms in the PC, the median of all the data of each sampling day was taken as a summary value of the measured time. Fig. 4 shows the execution times of the four cryptographic algorithms implemented in Python.

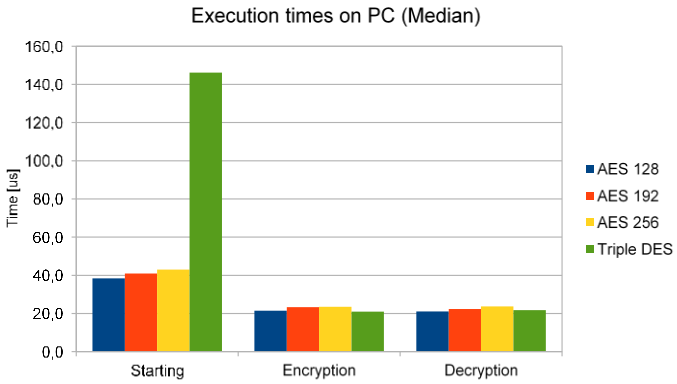

Fig. 4. Median of the execution times of the algorithms in the PC.

Fig. 4 shows that the encryption and decryption times of all the algorithms are similar, but the initialization time of the Triple DES algorithm is significantly longer, about three times the average time of the other three algorithms. Fig. 5 shows the execution times of only the three implemented AES algorithms, in order to appreciate the differences between them. As can be assumed, the fastest AES algorithm is the one with the shortest key, i.e. AES128, and the slowest is AES256, but the differences in execution times are not as marked as would be expected, for example, the difference between the encryption time of AES128 and AES192 is only 8.3%, and between the same time of AES128 and AES256 is only 9.4%. This last one means that it could have a key size of double bits increasing only a 9.4% of execution time in the Python implementation.

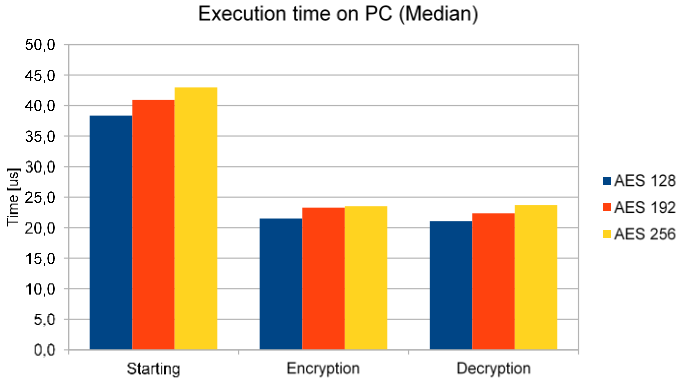

Fig. 5. Median of the execution times of the AES algorithms executed by the PC.

On the other hand, Fig. 6 and Fig. 7 show the same execution times taking the median of the sampled data, this time for the implementations in the embedded system based on PSoC 6. Again, the Triple DES algorithm has much longer execution times than the AES versions as shown in Figure 6. On the other hand, Fig. 7 shows again that the difference between the AES algorithm versions is not as marked (between 2.7% and 4%).

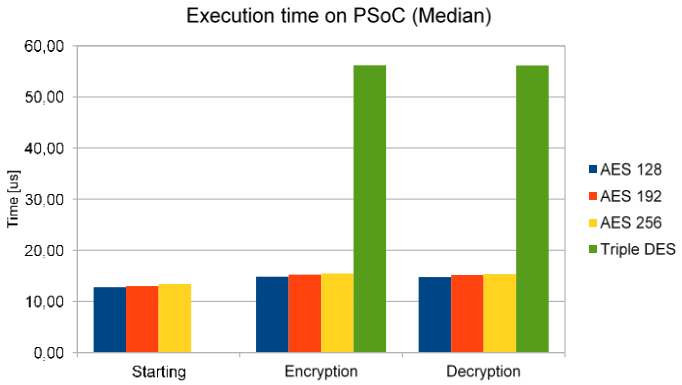

Fig. 6. Median of the execution times of the algorithms in the PSoC.

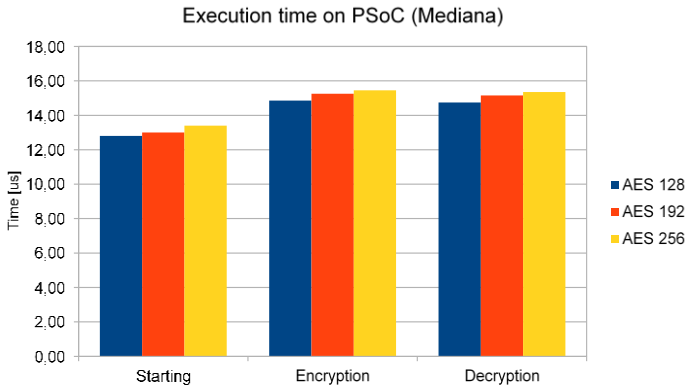

Fig. 7. Median of the execution times of the AES algorithms executed in the PSoC.

As can be seen in Fig. 6, the implementation of the Triple DES algorithm in the embedded system does not have an initialization routine; this implies that both in the measured encryption and decryption time, the initialization time is implicit. Therefore, in order to make a fair comparison of the four algorithms, a total encryption time and total decryption time were calculated, which include the initialization time. Tables 2 and 3 show comparisons of the four algorithms in terms of total encryption and decryption times.

Table 2. Measurements of total encryption and decryption times of the algorithms implemented in the PC.

T[ms]	*AES 128*	*AES 192*	*AES 256*	*Triple DES*
T. encryption	56,85	64,22	66,49	167,11
T. decryption	59,43	63,3	66,69	167,88

Table 3. Measurements of the total encryption and decryption times of the algorithms implemented in the PSoC.

T[ms]	*AES 128*	*AES 192*	*AES 256*	*Triple DES*
T. encryption	27,65	28,25	28,85	56,2
T. decryption	27,55	28,15	28,75	56,15

As it can be appreciated in Tables 2 and 3, the execution times are better in the embedded system than in the PC, but in both devices these times keep a relative proportion, being only a little higher the Triple DES times with respect to the AES in the implementation in PC compared with the one done on the embedded system. Based on the totalized data of encryption and decryption times carried out in both the PC and the PSoC, we proceed to make a one-to-one comparison of the four algorithms implemented in both devices. Fig. 8 and Fig. 9 show these comparisons for AES128 and 3DES algorithms.

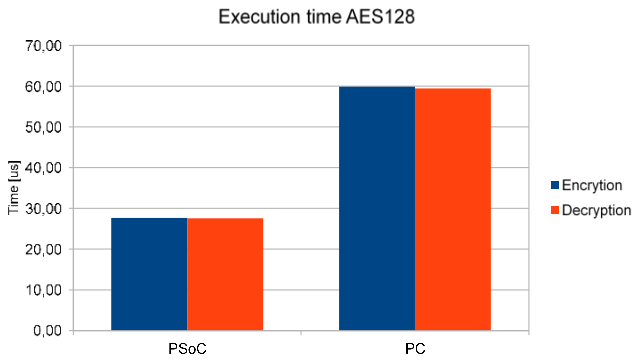

Fig. 8. Comparison between the execution times of the AES128 algorithm implemented in the PSoC and the PC.

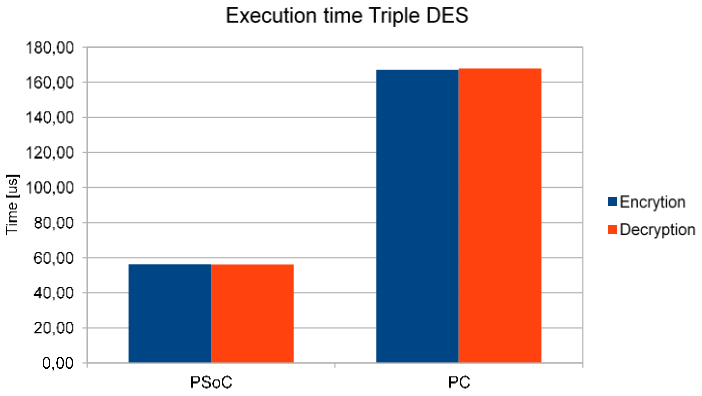

Fig. 9. Comparison between the execution times of the 3DES algorithm implemented in the PSoC and the PC.

As seen in Fig. 8 and Fig. 9 for the AES128 and 3DES algorithms, as well as for AES196 and AES256 (not shown, but with similar behavior), the execution times achieved by the PSoC-based embedded system range from 46% to 33% of the time made by the PC for the same algorithms. This can be explained, bearing in mind that the processor of the embedded system is dedicated solely to the execution of the cryptographic algorithm and that a large part of the cryptographic operations are carried out by means of a hardware acceleration module specific for cryptography available in the embedded system.

3.2 Application runtimes

The AES128 cryptographic algorithm was used for the application that turns on and off the logic outputs of the microcontroller of the embedded system from the PC, by means of secure commands via USB. For this application, a time sampling was also carried out, this time from the PC by means of the program developed in Python. The idea is to have an estimated value of the total execution time of a secure command sent from the PC and received by the embedded system. The steps for the complete execution of a command are as follows:

- Encryption in the PC of the command typed by the user.
- Sending the encrypted command by USB to the embedded system.
- Reception of the encrypted command in the embedded system.
- Decryption of the command in the embedded system.
- Command recognition and execution.
- Selection of an answer for the PC.
- Encryption in the embedded response system.
- Sending the encrypted response via USB to the PC.
- Reception of the encrypted response on the PC.
- Decryption of the response on the PC.
- PC display of the response of the embedded system.

A total of 50 command execution times were sampled for each of the 5 days of testing, for a total of 250 samples. Based on these data, the medians of each day's data were calculated, and these data were then averaged for the five days of sampling,

obtaining a total execution time for each command of 36.28 ms.

4. CONCLUSIONS

Precise measurement of PC execution times of implemented cryptographic algorithms is difficult because the PC processor executes an operating system, which is ultimately who gives execution priority to the programs and/or processes active at the time. This last one makes that the measured times not only include the execution of the cryptographic algorithm but of any other process with greater priority in that moment. This problem was minimized, taking 50 samples each time in cycle, and taking as valid value of time the median of these samples.

On the other hand, the measurement of the execution times of the algorithms implemented in the embedded system were quite consistent, because the processor (ARM Cortex M4) was dedicated exclusively to the execution of the cryptographic algorithm, without having to attend interruptions, programming threads or RTOS tasks that could affect the measurement of the execution times. Additionally, the implemented hardware time measurement methodology minimizes the error produced when the same processor oversees measuring the times.

The Triple DES algorithm implemented in the Python programming language and executed in the PC, presents an initialization time of almost three times that presented by each of the three AES algorithms, which makes this algorithm an uninteresting option compared to its security level and execution time.

The three different versions of the implemented AES algorithm showed a much better response time than the Triple DES algorithm, being the fastest the AES128 algorithm (128 bits password length), however, the implementations of AES192 and AES256 presented execution times very close to AES128. It is concluded that, if you want to have an application with the best possible Throughput, the recommended algorithm (within the four evaluated) is the AES128, but if you want to increase security, the AES256 has the best relationship between security (256 bits of password) and execution time (increases between 4% and 9.3% with respect to AES128).

Despite the high performance of the PC used in the experimental tests, the processor of the embedded system can execute the cryptographic algorithms more quickly, having a processing speed 28 times lower. This is due to the fact that the chip used is not only dedicated exclusively to the execution of the cryptographic algorithm (there is no operating system or any other task running at the same time), but also has an acceleration hardware module that facilitates a large part of the operations performed when executing a cryptographic algorithm.

A practical application of sending secure commands from a PC to a CY8CKIT-062-BLE embedded system was obtained, obtaining a total command execution time of 36.28 ms, which would imply a sampling frequency of 27.5 Hz, in case this prototype was used for an application in intelligent sensor systems, just to give an example.

5. FUTURE WORK

The communication medium used in the implemented prototype can be changed with relative ease to Bluetooth (native in the CY8CKIT-062-BLE embedded system), making use of the BLE-USB Dongle module (for PC) included with the embedded system, which gives the PC the ability to communicate with Bluetooth devices with low power consumption. You can also switch to WIFI communication by purchasing an external module (WIFI Shield) from the embedded system and using a standard WIFI network.

The complete application of sending secure commands was implemented with the AES128 algorithm but can be easily changed to AES192 and AES256. It is projected that the execution times that would be obtained by making this change would not increase the execution time beyond 10% (based on the previous study of the data obtained).

Given the scope of this work, only symmetrical block ciphers such as AES and TDES were evaluated, but both the pyCriptodome library and the PDL (Peripheral Device Library) of PSoC 6 support other types of ciphers and/or encryption algorithms, such as data flow ciphers, asymmetrical ciphers, and so on. It would be quite interesting to explore at least one standard asymmetric encryption algorithm such as RSA.

REFERENCES

[1] Roffe, I. G., Alvarado-Nava, O., Martínez, E. R., & Ramírez, A. F. (2018). Implementación Del Algoritmo De Cifrado Trivium En Un Sistema Embebido (An Implementation Of The Trivium Encryption Algorithm In An Embedded System). Pistas Educativas, 40(130).

[2] Gong, Z., Nikova, S., & Law, Y. W. (2011, June). KLEIN: a new family of lightweight block ciphers. In International Workshop on Radio Frequency Identification: Security and Privacy Issues (pp. 1-18). Springer, Berlin, Heidelberg.

[3] Engels, D., Fan, X., Gong, G., Hu, H., & Smith, E. M. (2010, January). Hummingbird: ultra-lightweight cryptography for resource-constrained devices. In International Conference on Financial Cryptography and Data Security (pp. 3-18). Springer, Berlin, Heidelberg.

[4] Beaulieu, R., Treatman-Clark, S., Shors, D., Weeks, B., Smith, J., & Wingers, L. (2015, June). The SIMON and SPECK lightweight block ciphers. In 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC) (pp. 1-6). IEEE.

[5] González, R. G., & López, A. T. (2014). Implementación Del Esquema De Firma Digital Ecdsa Sobre El Procesador Embebido Microblaze. Revista Telem@tica, 13(1), 31-45.

[6] Kavun, E. B., & Yalcin, T. (2011, November). RAM-based ultra-lightweight FPGA implementation of PRESENT. In 2011 International Conference on Reconfigurable Computing and FPGAs (pp. 280-285). IEEE.

[7] Taherkhani, S., Ever, E., & Gemikonakli, O. (2010, June). Implementation of non-pipelined and pipelined data encryption standard (DES) using Xilinx Virtex-6 FPGA technology. In 2010 10th IEEE International Conference on Computer and Information Technology (pp. 1257-1262). IEEE.

[8] Wee, C. M., Sutton, P. R., & Bergmann, N. W. (2005, August). An FPGA network architecture for accelerating 3DES-CBC. In International Conference on Field Programmable Logic and Applications, 2005. (pp. 654-657). IEEE.

[9] Ren, F., Chen, L., & Zhang, T. (2011, September). 3DES implementation based on FPGA. In International Conference on Web Information Systems and Mining (pp. 218-224). Springer, Berlin, Heidelberg.

[10] Liberatori, M., Otero, F., Bonadero, J. C., & Castineira, J. (2007, February). Aes-128 cipher. high speed, low cost fpga implementation. In 2007 3RD Southern Conference on Programmable Logic (pp. 195-198). IEEE.

[11] Renuka, G., Shree, V. U., & Reddy, P. C. S. (2018). Comparison of AES and DES Algorithms Implemented on Virtex-6 FPGA and Microblaze Soft Core Processor. International Journal of Electrical and Computer Engineering, 8(5), 3544.

[12] Navatha, K., Kumar, J. T., & Ganguly, P. (2017). An efficient FPGA Implementation of DES and Triple-DES Encryption Systems. Communication and Power Engineering, 348.

[13] Weeks, B., & Wingers, L. (2015, March). The SIMON and SPECK Block Ciphers on AVR 8-Bit Microcontrollers. In Lightweight Cryptography for Security and Privacy: Third International Workshop, LightSec 2014, Istanbul, Turkey, September 1-2, 2014, Revised Selected Papers (Vol. 8898, p. 3). Springer.

[14] Cazorla, M., Gourgeon, S., Marquet, K., & Minier, M. (2015). Survey and benchmark of lightweight block ciphers for MSP430 16-bit microcontroller. Security and Communication Networks, 8(18), 3564-3579.

[15] Kim, M., & Kwon, T. (2016). A Study of Implementing Efficient Rotation for ARX Lightweight Block Cipher on Low-level Microcontrollers. Journal of the Korea Institute of Information Security and Cryptology, 26(3), 623-630.

[16] Najm, Z., Jap, D., Jungk, B., Picek, S., & Bhasin, S. (2018, October). On Comparing Side-channel Properties of AES and ChaCha20 on Microcontrollers. In 2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS) (pp. 552-555). IEEE.

[17] Majid, A. H., Anwar, M., & Ashraf, M. W. (2018). Classified Structures and Cryptanalysis of Wg-7, Wg-8 and Wg-16 Stream Ciphers. Technical Journal, 23(02), 56-62. Darwish, H. Cryptographic Algorithms. Pomona: California State.

[18] Zhang, Z., Glaser, S. D., Watteyne, T., & Malek, S. (2016). Long-term monitoring of the Sierra Nevada snowpack using wireless sensor networks. IEEE Internet of Things Journal.

[19] Van Antwerpen, H., & van de Waerdt, J. W. (2015). U.S. Patent Application No. 14/580,753.

[20] Kitayama, R., Takenaka, T., Yanagisawa, M., & Togawa, N. (2016). A Highly-Adaptable and Small-Sized In-Field Power Analyzer for Low-Power IoT Devices. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 99(12), 2348-2362.