Indonesian Journal of Electrical Engineering and Computer Science
Vol. 20, No. 2, November 2020, pp. 894~902
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v20.i2.pp 894-902

Compare encryption performance across devices to ensure the security of the IOT

A.YU. Pyrkova, ZH.E. Temirbekova
Faculty of Information Technology, Al-Farabi Kazakh National University, Kazakhstan

Article Info

Article history:

Received Mar 3, 2020
Revised May 1, 2020
Accepted May 15, 2020

Keywords:

BLE nano microcontroller
mbed platform
Homomorphic encryption in a binary number ring
Smartcard ML3-36k-R1
cryptography

ABSTRACT

The Internet of Things (IoT) combines many devices with various platforms, computing capabilities and functions. The heterogeneity of the network and the ubiquity of IoT devices place increased demands on security and privacy protection. Therefore, cryptographic mechanisms must be strong enough to meet these increased requirements, but at the same time they must be effective enough to be implemented on devices with disabilities. One of the limited devices are microcontrollers and smart cards. This paper presents the performance and memory limitations of modern cryptographic primitives and schemes on various types of devices that can be used in IoT. In this article, we provided a detailed assessment of the performance of the most commonly used cryptographic algorithms on devices with disabilities that often appear on IoT networks. We relied on the most popular open source microcontroller development platform, on the mbed platform. To provide a data protection function, we used cryptography asymmetric fully homomorphic encryption in the binary ring and symmetric cryptography AES 128 bit. In addition, we compared run-time encryption and decryption on a personal computer (PC) with Windows 7, the Bluetooth Low Energy (BLE) Nano Kit microcontroller, the BLE Nano 1.5, and the smartcard ML3-36k-R1.

Corresponding Author:

Temirbekova Zhanerke Erlanovna,
Faculty of Information Technology,
Al-Farabi Kazakh National University,
Al-Farabi Avenue, 71, Almaty, Kazakhstan.
Email: temyrbekovazhanerke2@gmail.com

1. INTRODUCTION

The Internet of Things (IoT) has tightly entered our lives and billions of people around the world. However, an increase in the number of connected devices leads to an increase in security risks: from physical harm to people to downtime and damage to equipment - it can even be pipelines, blast furnaces and plants for generating electricity. Since a number of such IoT facilities and systems have already been attacked and significant damage has been done, ensuring their protection comes to the fore. As IoT technologies evolve, the number of devices connected to each other is also increasing. These devices provide an unprecedented ability to identify and control the environment. Sensor data can be transferred between devices via remote data to the analyzed center. The security of this technology is still a big question, because it is still unsafe by nature. For embedded systems, this is not a new security issue [1-3]. Security issues such as eavesdropping can illegally affect people's daily activities, even a -legitimate subject can collect data without user consent. The easiest way to ensure data privacy is to apply encryption to the data [4, 5].

The IoT, like any rapidly developing technology, is experiencing a number of "growth diseases", among which the most serious is the security problem. The more "smart" devices are connected to the network, the higher the risks associated with unauthorized access to the IoT system and the use of its capabilities by attackers.

The development of the concept of the IOT and its implementation in various fields provides for the presence of tens of billions of stand-alone devices. According to the Statista portal, in 2017 there are already more than 20 billion of them, and by 2025 at least 75 billion are expected. All of them are connected to the Network and transmit data corresponding to their functionality through it. Both data and functionality are targets for attackers, which mean they must be protected.

In addition to the basic security features, confidentiality must also be considered in the IoT. Many IoT services and applications provide confidential and personal information that may be disclosed by an attacker. Unprotected confidential data may be transferred to third parties. Many privacy solutions are designed for powerful computers and sites on the Internet. Confidentiality solutions are usually based on computationally expensive cryptographic primitives. In this regard, it remains an open task to develop a safe, effective and confidentiality solution for IoT, which works mainly with devices with limited access. The main goal of this work is to show how common cryptographic primitives are required on various devices, and to show the perspective of some methods of maintaining confidentiality in IoT.

In this article, we present the performance of widely used cryptographic algorithms on various devices and discuss their memory limitations. We implement and measure the cryptographic operations on a variety of platforms. The purpose of a scientific article is to performance encrypt and decrypt modern cryptographic algorithms, like fully homomorphic encryption in a binary number ring and block cipher on BLE Nano Kit and BLE Nano 1.5 microcontrollers on mbed platform, Smartcard ML3-36k-R1 on MULTOS platform and PC Windows 7. Then compare the execution time of various symmetric and asymmetric algorithms in the resource-limited IoT device and a high-performance IoT device.

A microcontroller is an integrated circuit (IC) that can be programmed to perform a set of functions to control a collection of electronic devices [6, 7]. Being programmable is what makes the microcontroller unique. Microcontroller can be easily adopted in various applications with a variety of peripherals due to its merits of small size, simple architecture. Bluetooth low energy (BLE) Nano is one kind of microcontroller with open source platform [8]. BLE is a wireless technology standard for personal area networks. BLE is targeted for very low power devices, devices that can run on a coin cell battery for months or years. Typical applications that uses BLE are health care, fitness trackers, beacons, smart home, security, entertainment, proximity sensors, Industrial and automotive. In this paper we use two kinds of microcontrollers: BLE Nano Kit and BLE Nano 1.5. BLE Nano Kit and BLE Nano 1.5 are the smallest BLE development board in the market. The core is Nordic nRF51822 (an ARM Cortex-M0 SoC plus BLE capability) running at 16MHz with ultra-low power consumption. You can quickly produce porotypes and demos target for IoT and other interesting projects. BLE Nano could operate under 1.8V to 3.3V, therefore it works with a lot of electronic components. In short, BLE Nano Kit and BLE Nano 1.5 have a broad developing future because of its low cost, cross-OS scalability, open source and easy usage features [9, 10]. As a result, various multifunctional applications can be developed on this platform. To perform the operation as a smart card, we chose ML3-36k-R1 smart card. Our card has an M3 genetic ML3-36k-R1 implementation by MULTOS.

The rest of the paper is organized as follows. In Section 2, we overview the main features and applicability of both block cipher and fully homomorphic encryption in a binary number ring. In Section 3 we present the running time of different algorithms in our microcontroller, smart card and PC and issues as well as discuss the adoption of the approach. Finally, in Section 4, we report the final conclusions of the paper.

2. RESEARCH METHOD

This section discusses in detail the method and process for this experimental encryption method and the briefly introduces the main characteristics and advantages of block cipher and fully homomorphic encryption in a binary number ring, the two proposed technologies.

2.1. Block cipher

To ensure security and maintain confidentiality of data, we use from the symmetric block cipher algorithm, we selected widely used Advanced Encryption Standard (AES) encryption. AES in addition recognized as Rijndael is used for analyze time securing data. In cryptography, a block cipher is a deterministic algorithm operating on fixed-length groups of bits, called blocks, with an unvarying transformation that is specified by a symmetric key. Block ciphers are important elementary components in the design of many cryptographic protocols, and are widely used to implement encryption of bulk data. AES is based on a design principle known as a substitution-permutation network, a combination of both substitution and permutation, and is fast in both software and hardware [11, 12]. In this article, data was encrypted and decrypted using AES with a 128-bit key. The AES-128 encryption and decryption data flow diagram is shown in Figure 1.

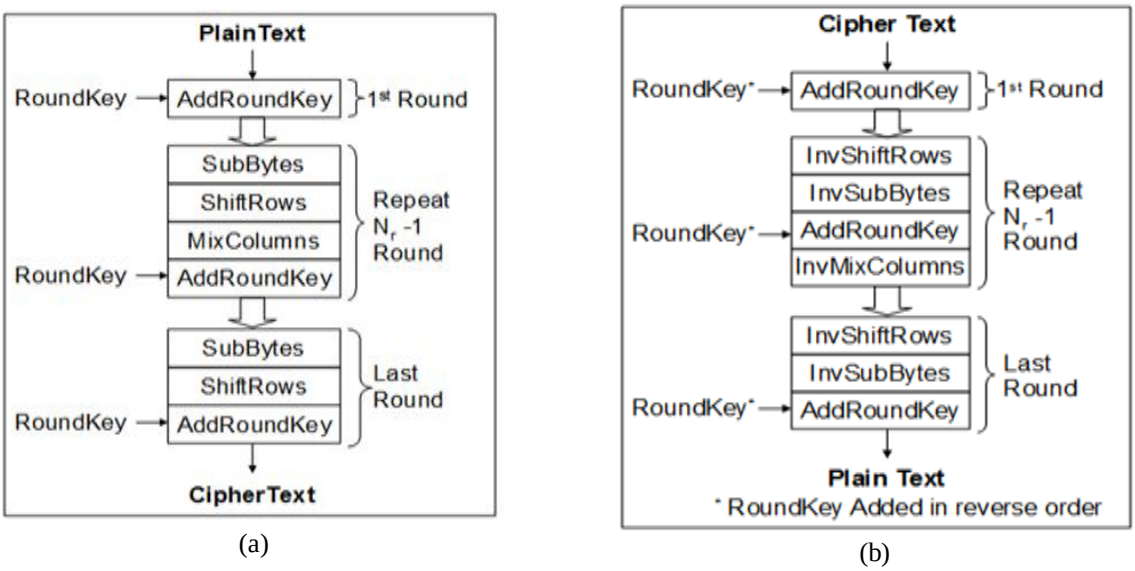

Figure 1. (a) and (b) AES encryption and decryption steps

AES consists of 2 Main Parts:

1) Encryption or Decryption Process

In each round, the block cipher uses the four following operations [13, 14]:

a) SubBytes: Each byte of the array is transformed using a nonlinear substitution box called the AES S-Box. The S-Box in the Block cipher has been carefully constructed and the cipher uses only one S-Box throughout the encryption.
b) ShiftRows: Is a transposition step which ensures that the last three rows of the array are shifted by a different number of byte positions.
c) MixColumns: Mixes each column in the array to create even more diffusion.
d) Addkey: Using bitwise XOR, each byte of the array is mixed with a byte of a sub-key material, also called round-key. The sub-key is made by "key expansion" and is derived from the main cipher key using a Rijndael key-schedule.
e) Encryption process: It starts with AddKey with Key0.

Then go to loop and do SubBytes, ShiftRows, MixColumns, Addkey in that order for 9 circles each circle with different circle key. Then go to final circle (circle 10) and repeat the same previous function in the loop except MixColumns. Decryption process: it is reverse of encryption process in every step which means the decryption first circle is the tenth circle of the encryption and it uses the invers functions of MixColumns, SubBytes, ShiftRows and us you can assume the keys arrangement and reversed too as it starts with Addkey10 instead of Addkey0 as it was in the encryption process.

2) Key generation

It involves RotWord, SubBytes and XOR bitwise operation to generate enough keys for each circle In the Encryption, Decryption process. As each circle works with different key generated from the key generation process. This encryption protects information and keys guarantees that they continue to be under the users influence and will be exposed in storage or in transit. Advanced encryption standard algorithm has replaced the data encryption standard algorithm as permitted general for an extensive variety of applications.

2.2. Fully homomorphic encryption in a binary number ring

In the present work fully homomorphic encryption in a binary number ring technique is used for data encryption. Here homomorphic encryption technique is used because that is most capable of carrying mathematical computations evolved on data readings. Homomorphic encryption is a form of encryption which performs arbitrary computations on encrypted data. In cloud computing we may keep our sensitive data in encrypted format, but if you want to do any calculation on cipher text, the key must be shared with cloud service providers which may cause to exploit our data. So that to avoid share the key to cloud service providers instead use the Homomorphic Encryption technique. The computations include searching, sorting, addition, multiplications performed on cipher text [15, 16].

Among so many cryptographies, homomorphic encryption has attracted widely attentions from scholars for its special performance [17]. Common cryptography can't directly do calculations on encrypted data, but homomorphic encryption can, meanwhile, the operation results of homomorphic encryption will be automatically encrypted. The application prospect of homomorphic encryption is widely and cheerful in the fields of secure multi-party computation, electronic voting, cipher text searching, encrypted mail filtering, mobile cipher. Finally, security analysis is tested and further research way is pointed out.

Homomorphic encryption seeks to aid in this encryption process by allowing specific types of computations to be carried out on cipher text which produces an encrypted result which is also in cipher text. Its outcome is the result of operations performed on the plaintext. Case in point, one person could add two encrypted numbers and then another person could decrypt the result, without either of them being able to find the value of the individual numbers.

Fully homomorphic encryption (FHE) enables secure computation over the encrypted data of a single party. Craig Gentry implemented fully homomorphic encryption based on bootstrapping over partially homomorphic encryption by using ideal lattices. It is limited because each cipher text is noisy in some sense, and this noise grows as one ad and multiplies cipher texts. Gentry showed that any boots trappable Somewhat Homomorphic Encryption scheme can be converted into a FHE through a self-eMbedding recursion. In case of Gentry's "noisy" scheme, the bootstrapping procedure effectively "refreshes" the cipher text by applying to it the decryption procedure homomorphically, there by obtaining a new cipher text that encrypts the same value as before but has lower instance of noise [18]. The cipher text is periodically "refreshed" whenever the noise grows too complex.

FHE in a binary number ring. The scheme of completely homomorphic encryption, which Gentry suggested, can be considered using the example of calculations in Z_2 [19, 20].

1) Encryption

The process of data encryption can be represented as follows:

a) We choose an arbitrary odd number $p = 2k + 1$, which is a secret parameter. Let $m \in \{0,1\}$.

b) The number $z \in Z_2$ is compiled such that $z = 2r + m$, where r is an arbitrary number. This means that $z = mmod2$.

c) In the process of encryption, each m is associated with the number $c = z + pq$, where q is chosen arbitrarily. Thus, $c = 2r + m + (2k + 1) * q$. It is easy to see that $cmod2 = (m + q)mod2$ and therefore an attacker can determine only the parity of the output of encryption.

2) Decryption

Let the encrypted number c and the secret p be known. Then the process of decrypting the data should contain the following actions:

a) Decoding using a secret parameter $p: r = cmodp = (z + pq)modp = zmodp + (pq)modp$ where $r = cmodp$ is called noise

b) Obtaining the original encrypted bit: $m = rmod2$.

3) Justification:

Let there be two bits $m_1, m_2 \in Z_2$ and they are associated with a pair of numbers $z_1 = 2r_1 + m_1$ and $z_2 = 2r_2 + m_2$. Let us take the secret parameter $p = 2k + 1$ and encrypt the data: $c_1 = z_1 + pq_1$ and $c_2 = z_2 + pq_2$. The sum of these numbers is calculated:

$c_1 + c_2 = z_1 + pq_1 + z_2 + pq_2 = z_1 + z_2 + p(q_1 + q_2) = 2r_1 + m_1 + 2r_2 + m_2 + (2k + 1)(q_1 + q_2)$

For the sum of these numbers, the decrypted message is the sum of the original bits $m_1 m_2$: $\big((c_1 + c_2)modp\big)mod2 = (2(r_1 + r_2) + m_1 + m_2)mod2 = m_1 + m_2$. But without knowing p, decoding the data is not possible: $\big((c_1 + c_2)modp\big)mod2 = m_1 + m_2 + q_1 + q_2$. Figure 2 shows the FHE in a binary ring is an addition operation.

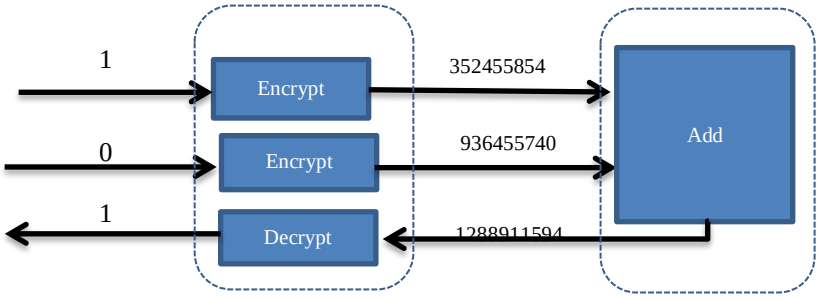

Figure 2. Result of homomorphic encryption with the addition operation in C++

Similarly, the operation of multiplication is checked:

$c_1c_2 = (z_1pq_1)(z_2pq_2) = z_1z_2 + p(z_1q_2 + z_2q_1) + p^2q_1q_2 = (2r_1 + m_1)(2r_2 + m_2) + (2k + 1)\big((2r_1 + m_1)q_2 + (2r_2 + m_2)q_1\big) = 4r_1r_2 + 2(r_1m_2 + r_2m_1) + m_1m_2 + 2k(2r_1q_2 + m_1q_2 + 2r_2q_1 + m_2q_1) + 2r_1q_2 + m_1q_2 + 2r_2q_1 + m_2q_1$

The decryption procedure must be applied to the results obtained, which will result in the following: $\big((c_1c_2)modp\big)mod2 = (4r_1r_2 + 2(r_1m_2 + r_2m_1) + m_1m_2)mod2 = m_1m_2$.

Figure 3 illustrates the fully homomorphic encryption in a binary ring is a multiplication operation.

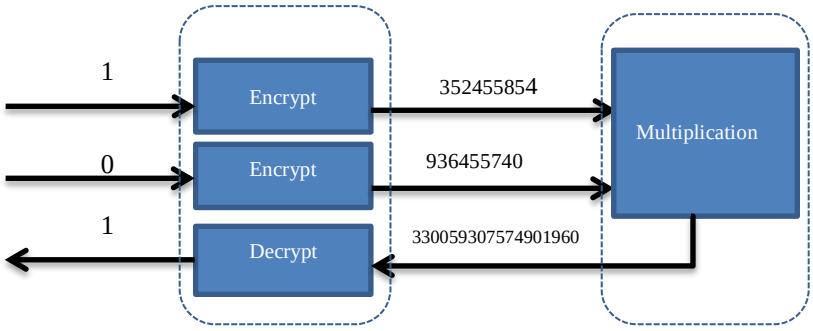

Figure 3. Result of homomorphic encryption with the multiplication operation in C++

3. RESULTS AND DISCUSSION

In this section, we implement measure and compare operations on various platforms, such as microcontrollers, chip cards, Windows 7 PCs, which are used in the IoT environment. We implement the most commonly used cryptographic primitives and schemes, such as fully homomorphic encryption in a binary ring and AES cryptography. These cryptographic primitives and operations are used in many IoT security solutions.

Resource-limited devices are considered the most used devices in the IoT infrastructure. We present the performance results of cryptographic primitives that are implemented on devices with limited resources, namely: BLE Nano kit microcontroller, BLE Nano 1.5 microcontroller, 33 MHz Multos card. Also, the performance results of cryptographic primitives, which are implemented by a high-performance device (PC with Windows 7).

The selected cryptographies are implemented in two programing languages that depend on the device. Implementations of cryptographic functions on microcontrollers are written in the programming language C. Our test application for the Multos card is written in C. C++ is used for the implementation and the tests on the PC device. All these devices provide the sufficient space of RAM and storage memory for privacy preserving schemes. Furthermore, the devices offer programmable platforms for loading own applications and libraries. Chosen privacy-preserving schemes are implemented and loaded on these devices. The technical characteristics of BLE Nano kit microcontroller, BLE Nano 1.5 microcontroller, 33 MHz Multos card devices and Windows 7 PCs are shown in Table 1.

Table 1. Technical characteristics of the devices used

Device	Processor	Frequency	RAME size	Storage size (ROM, Flash)
Microcontroller BLE Nano kit	16 bit CPU	16MHz	16 kB	256 kB
Microcontroller BLE Nano 1.5	16 bit CPU	16MHz	32 kB	256 kB
Smartcard ML3-36k-R1	16 bit CPU	33MHz	1088 + 960 B	280 + 60 kB
PC with Windows 7	64 bit Intel (R) Celeron (R) CPU	P4600 2.00 GHz	6 GB	16 GB

How programming works in microcontrollers?

The mbed platform is used for the implementation and the tests on the microcontroller's devices. Mbed is a platform and operating system for internet-connected devices based on 32-bit ARM Cortex-M microcontrollers [21]. Such devices are also known as Internet of Things devices [22]. The project is collaboratively developed by Arm and its technology partners. We connect the mbed microcontroller to our PC using USB, it looks like a USB flash drive. This small disk is represented by the mbed interface and allows you to save the hexadecimal files of the BLE Nano microcontrollers that we want to run directly on mbed without the need for drivers. When saving the .hex file to the mbed disk, it is not immediately loaded into the flash memory of the internal microcontroller.

When we hit reset, the mbed interface looks at the disk for the newest .hex file it can find. If there is a new file, it will load it in to the microcontroller's internal FLASH memory using the JTAG interface. If the newest binary is already loaded, it won't load it again. It then starts the microcontroller running.

How USB serial works?

The mbed interface also represents the USB serial / com interface. This is basically a UART-USB bridge that connects to the UART interface. Therefore, if we send characters from the UART of the target microcontroller, they will be read by the mbed interface and transferred via USB [23].

For programming devices, we use terminal emulator "Tera Term".

For the connection of microcontroller and terminal, there are two main points needed to be paid more attentions.

1) Serial Communications: The first point is serial communications. In microcontroller serial communications with a contact type UART interface with even parity bit and 2 stop bits for every 8 bits of data transfer.
2) Protocol Parameter Selection and Baud Rate: Serial baud rate of the contact type microcontroller is controlled by the clock signal of an external oscillator. BLE Nano microcontrollers have two external oscillators. The frequency of the slow clock is 32.768 kHz. The frequency of the main clock is 16 MHz and the baud rate is about 9,600 bps.

MULTOS platform for Smartcard ML3-36k-R1

The MULTOS platform is used for the implementation and the tests on the Smartcard's devices. The purpose of the MULTOS platform is to provide a secure, device-independent platform for executing smart cards [24]. To this end, they developed a specification for execution and memory models, explained in more detail below, which all MULTOS implementations should provide. In addition to this mandatory part of the specification, there are also a number of additional elements, mainly related to cryptographic functions, which may or may not be available on a particular hardware platform [25, 26].

Table 2 presents our experimental results for selected resource-limited devices and PC techniques. We measure the performance overhead on the devices used and we estimate memory/communication overhead. We focus on the time of main operations/phases such as the encryption time and the decryption time. A performance cryptographic operation on microcontrollers is measured as the number of cycles, and the number is converted at run time (1 cycle takes 1 microsecond on a processor with a clock frequency of 1 MHz). Smart card, BLE Nano Kit, BLE Nano 1.5 and PC Windows 7 transaction times are average values calculated over 10 iterations.

Table 2. Running time of different algorithms in resource-limited devices and pc

Cipher	Microcontroller BLE Nano kit	Microcontroller BLE Nano 1.5	Smartcard ML3-36k-R1	PC Windows 7
AES 128 bit Encrypt	1,52 ms	1,48 ms	14,63 ms	0,10 ms
AES 128 bit Decrypt	1,75 ms	1,72 ms	16,07 ms	0,12 ms
FHE in a binary ring Encrypt	391,8 ms	390,7 ms	288,9 ms	21 ms
FHE in a binary ring Decrypt	625,5 ms	623,5 ms	546,9 ms	46 ms

The encryption was verified for microcontrollers by using terminal tera term software to send data and cypher keys from a computer. We then perform a series of tests. Figure 4 shows the implementation time of AES-128b operations on devices with limited resources. This operation takes only a few milliseconds on these devices (it is suitable healthcare system real-time health monitoring). The AES encryption operation of one 128-bit data block is more efficient on microcontrollers than on smart cards. Initializing the card operating system and APDU communication between the chip card and the reader causes a time delay. Therefore, chip cards with higher processor frequencies than microcontrollers require longer execution time for AES cryptographic operations than microcontrollers. From the results, it can be shown that the performance of microcontroller is comparable to the performance. However, measurements show that AES features are effective and can be implemented in IoT security solutions that run on devices with disabilities.

Figure 5 shows the implementation time of FHE in a binary ring encryption and decryption operations on devices with low performance. FHE in binary ring public key operations take hundreds of milliseconds on microcontrollers. FHE in a binary ring operations with private keys take several seconds on microcontrollers. The Multos cards provide direct FHE in a binary ring APIs that are optimized. In this regard, FHE in binary ring operations on these smart cards take from tens to hundreds of milliseconds.

Figure 6 shows AES 128 bit, FHE in a binary ring encryption time and decryption time on PC Windows 7. PC Windows 7 is a high-performance device. PC Windows 7 devices provide adequate space and RAM memory circuits for confidentiality. In this devices offer programmable platforms for loading their own applications and libraries. FHE in binary ring and AES 128 bit encryption and decryption time take tens of milliseconds on PC Windows 7.

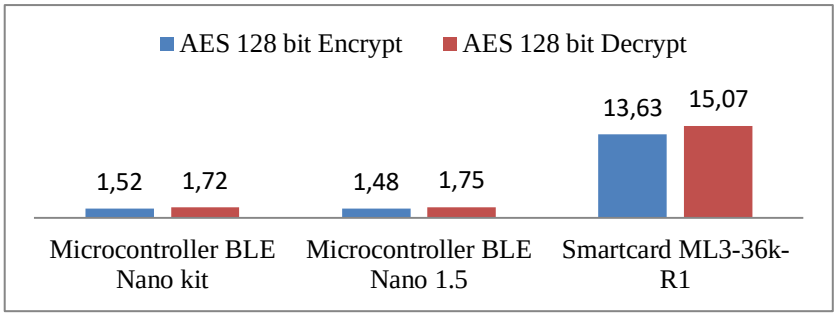

Figure 4. The operating time of AES 128bit on microcontrollers and smartcards devices

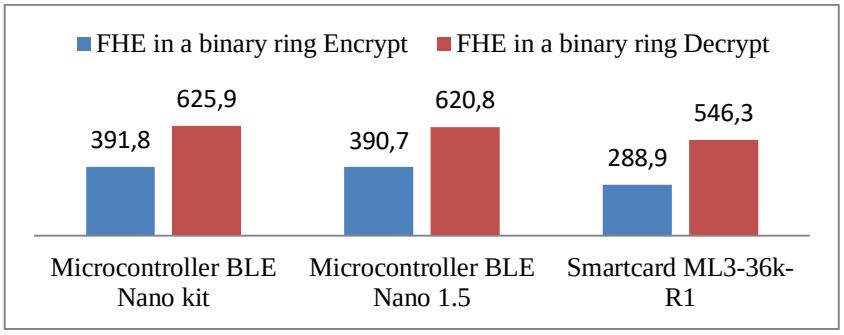

Figure 5. The operating time of FHE in a binary ring on microcontrollers and smartcards devices

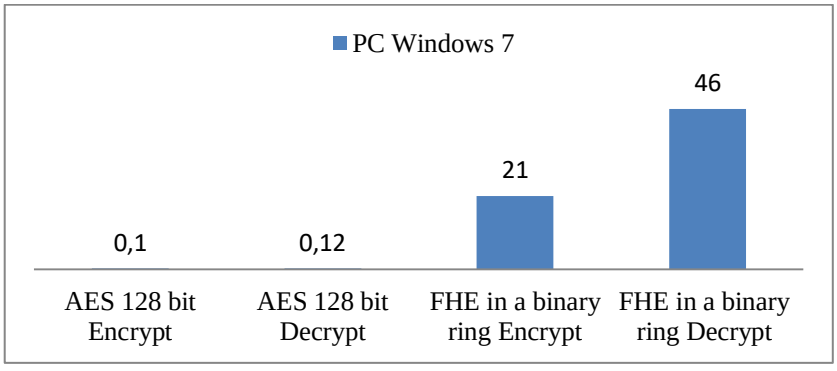

Figure 6. The operating times of AES 128 bit, FHE in a binary ring on PC Windows 7

4. CONCLUSION

This paper presents the performance and memory limitations of modern cryptographic primitives and schemes on various types of devices that can be used in IoT. Currently, symmetric ciphers and asymmetric ciphers

can be easily implemented in IoT services that use limited devices. In this article, the use of a hardware platform for creating real-life applications was satisfactory and promising in terms of speed, versatility, and security. In our research work, we used the good compatibility of the mbed platform with the microcontroller. As one of the most common microcontrollers, the BLE Nano Kit and BLE Nano 1.5 can be used in the application to provide various security functions. However, compared to Windows 7 x64, due to the limited processing power and memory of the BLE Nano Kit, BLE Nano 1.5 microcontrollers, and smartcard ML3-36k-R1 give rather low performance, especially when FHE is in the binary ring. Many symmetric ciphers, such as AES 128 bit, are fast enough to be implemented in security solutions that run on BLE Nano Kit, BLE Nano 1.5 microcontroller devices in the IoT infrastructure. On the other hand, security solutions based on asymmetric cryptographic operations, such as binary ring FHEs, require more time to execute. FHE in the binary ring takes tens to hundreds of milliseconds on devices with disabilities. For example, FHE in a binary ring takes several hundred seconds on microcontrollers, not suitable for real-time (health care) IoT applications. On the other hand, smart meters with limited microcontrollers, which usually send energy consumption data, can use the FHE signature in the binary ring, since the data is sent only a few times a day.

REFERENCES

[1] M. Aljarah, et al., *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, pp. 296-307, 2020.
[2] A. Muneer, et al., "Smart health monitoring system using IoT based smart fitness mirror," *TELKOMNIKA Telecommunication, Computing, Electronics and Control*, vol. 18, pp. 317-331, 2020.
[3] C. Ebert and A. Dubey, "Convergence of Enterprise IT and Embedded Systems," in *IEEE Software*, vol. 36, pp. 92-97, 2019.
[4] A. Ukil, et al., "Embedded security for Internet of Things," *2nd National Conference on the Emerging Trends and Applications in Computer Science* (NCETACS), pp. 87-92, 2011.
[5] Z. Kasiran, et al., "An advance encryption standard cryptosystem in iot transaction," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 17, pp. 1548-1554, 2020.
[6] J. Angel, "BLE Nano hardware development kit for Bluetooth Low Energy," 2015. Available: http://blog.blecentral.com/2015/05/28/ble-nano-hardware-development-kit-for-bluetooth-low-energy/.
[7] S. Aguilar, et al., "Opportunistic sensor data collection with bluetooth low energy," *Sensors*, vol. 17, p. 159, 2017.
[8] O. Abdelatty, et al., "A Low Power Bluetooth Low-Energy Transmitter with a 10.5nJ Startup-Energy Crystal Oscillator," *ESSCIRC 2019 – IEEE 45th European Solid State Circuits Conference (ESSCIRC)*, pp. 377-380, 2019.
[9] M. Mimoso, "Move Over Web Security, Embedded Devices Are Darling of Black Hat." Available: http://threatpost.com/move-over-web-security-embedded-devices-are-darling-of-black-hat/.
[10] F. Shi, et al., "A Survey of Data Semantization in Internet of Things," *Sensors*, vol. 18, p. 313, 2018.
[11] R. Atiqur, et al., "Mobile edge computing for internet of things (IoT): security and privacy issues," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 18, pp. 1486-1493, 2020.
[12] C. J. C Cahuana, "A Search for a Convenient Data Encryption Algorithm for an Internet of Things Device," pp. 102-106, 2016.
[13] A. Biryukov and D. Khovratovich, "Related-key Cryptanalysis of the Full AES-192 and AES-256," *The original on 2009-09-28.* Retrieved 2020-02-16.
[14] Z. Wang, et al., "Dynamically Reconfigurable Encryption and Decryption System Design for the Internet of Things Information Security," *Sensors (Basel)*, vol. 1, p. 143, 2019.
[15] N. P. Smart and F. Vercauteren, "Fully homomorphic encryption with relatively small key and ciphertext sizes, public Key Cryptography," *PKC Springer Berlin Heidelberg*, vol. 6056, pp. 420-443, 2010.
[16] Y. NarasimhaRao, et al., "Providing enhanced security in IoT based smart weather system," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 18, no. 1, pp. 9-15, 2020.
[17] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pp. 169-178, 2009.
[18] C. Gentry and D. Boneh, "A fully homomorphic encryption scheme," PhD thesis, Stanford, Stanford University, 2009.
[19] M. Van Dijk, et al., "Fully homomorphic encryption over the integers," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, vol. 12, pp. 24-43. 2010.
[20] M. Naehrig, et al., "Can homomorphic encryption be practical?" in *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pp. 113-124. 2011.
[21] Arm Holdings, "Mbed os developers website," 2020. Available: https://os.mbed.com/.
[22] Arm Holdings, "Mbed os product description," 2020. Available: https://www.mbed.com/en/development/mbed-os/.
[23] Mbed Team, "Example implementation to connect an mbed device to mbed device connector," 2020. Available: https://github.com/ARMmbed/mbed-os-example-client.
[24] E. Rescorla and N. Modadugu, "Datagram transport layer security version" *Pim Vullers, Gergely Alpár. Efficient Selective Disclosure on Smart Cards Using Idemix. 3rd Policies and Research in Identity Management* (*IDMAN)*, pp. 53-67, 2013.
[25] K. Hussain, et al., "Using a systematic framework to critically analyze proposed smart card based two factor authentication schemes," *Journal of King Saud University - Computer and Information Sciences*, pp. 56-64, 2019.
[26] Pandey J. G., et al., "A High-Performance and Area-Efficient VLSI Architecture for the PRESENT Lightweight Cipher," *Proceedings of the 17th International Conference on Embedded Systems; Maharashtra, India*, pp. 392-397, 2018.

BIOGRAPHIES OF AUTHORS

Pyrkova Anna Yurevna was born in Kazakhstan. She received the B.S. degree from the Kazakh National University named after al-Farabi in 1997 and the M.S. degree from the Kazakh National University named after al-Farabi in 1999, both in computer science. Main publications over the past 5 years: Number of scientific articles - 102, in publications recommended by the authorized body - 26, in scientific journals having, according to the information base of the company Thomson Reuters (Web of Science, Thomson Reuters) a non-zero impact factor - 8, in journals from the base of Scopus or Jstore - 5. Professor, Department of Informatics, Faculty of Information Technology, Al-Farabi Kazakh National University. She has an honorary diploma of the Ministry of Education and Science of the Republic of Kazakhstan "For a great contribution to the prosperity of Kazakhstan". Winner of the grant "The best teacher of the university -2018".

Temirbekova Zhanerke Erlanovna was born in Jambyl Region, Kazakhstan in 1989. She received the B.S. degree from the Kazakh National University named after al-Farabi in 2011 and the M.S. degree from the Kazakh National University named after al-Farabi in 2013, both in computer science. She is currently studying at the PhD of the Faculty of Information Technology of Al-Farabi Kazakh National University. Hers research interests include: microcontroller security, microcontroller protection algorithms.