# Using FHE in a binary ring Encryption and Decryption with BLE Nano kit microcontroller

*Zhanerke* Temirbekova Erlanovna[1*], *Anna* Pyrkova [2]

[1]Faculty of Information Technology, Al-Farabi Kazakh national university, Almaty, Kazakhstan
[2]Faculty of Information Technology, Al-Farabi Kazakh national university, Almaty, Kazakhstan

**Abstract.** An integrated circuit (IC) that can be programmed to perform a series of functions to control a range of electronic devices is a microcontroller. What makes the microcontroller special is that it is programmable. In this article, we're going to try to rely on the mbed platform, the most common open source microcontroller development platform; we use completely homomorphic encryption in a binary number ring to ensure the data protection feature. Let us compare the time it takes to perform encryption and decryption on a Visual Studio C ++ and a Bluetooth Low Energy (BLE) Nano kit microcontroller. Experimental results show that the device can complete a fully homomorphic encryption in a binary number ring in 64.2 microseconds, which is reasonable in a real application scenario and illustrates the feasibility of implementing a more complex cryptographic system using a microcontroller.

## 1. Introduction

Microcontroller can be easily adopted in various applications with a variety of peripherals due to its merits of small size, simple architecture. One kind of microcontroller with an open source platform is the BLE Nano Kit [1-2]. The smallest BLE production board on the market is the BLE Nano.

In short, due to its low cost, cross-OS scalability, open source and easy use features, BLE Nano Kit has a wide developing future [3-4]. As a consequence, on this framework, different multifunctional applications can be created. The aim of a scientific article is to perform on the microcontroller of the BLE Nano Kit on a Windows block cipher and modern cryptographic algorithms on the mbed platform and Visual Studio C++, such as completely homomorphic encryption in a binary number ring. The execution time of various algorithms in the microcontroller and the personal computer is then compared.

As follows, the rest of the paper is organized. In Section 2, we summarize the key features and applicability of a binary number ring for both block cipher and completely homomorphic encryption. We present the running time of various algorithms in our microcontroller and PC (personal computer) and problems in Section 3, as well as address the adoption of the strategy. Finally, we are reporting the final findings of the paper in Section 4.

* Corresponding author: temyrbekovazhanerke2@gmail.com

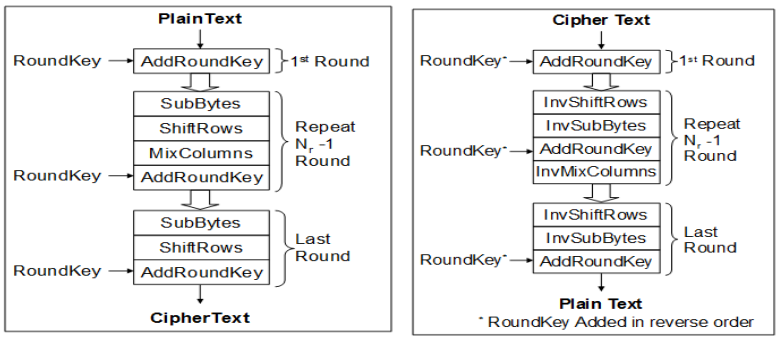
## 2. Cryptographic backgrounds

The key features and benefits of block cipher and completely homomorphic encryption in a binary number ring, both proposed technologies, are briefly discussed in this section.

### 2.1 Block cipher

A block cipher is a deterministic algorithm in cryptography that operates on fixed-length groups of bits, called blocks, with an unvarying transformation identified by a symmetric key. In the design of many cryptographic protocols, block ciphers are significant elementary components and are commonly used to enforce bulk data encryption.

We selected the commonly used Advanced Encryption Standard (AES) encryption[5] from the symmetric block cipher algorithm. In both software and hardware, AES is based on a design concept known as a substitution-permutation network, a combination of both substitution and permutation, and is fast (Fig.1).




(a) (b)

**Fig. 1.** a and b AES encryption and decryption steps

AES consists of two Main Parts:

1. Method for Encryption or Decryption:
   The block cipher uses the four following operations in each round:
   - SubBytes: A nonlinear substitution box called the AES S-Box transforms each byte of the sequence. The S-Box has been carefully designed in the Block cipher and the cipher uses only one S-Box in the encryption process.
   - ShiftRows: A transposition step that ensures that a different number of byte positions are transferred to the last three rows of the array.
   - MixColumns: To generate even more diffusion, mix every column in the series.
   - Addkey: Using bitwise XOR, each array byte is mixed with a sub-key material byte, often called round-key. The sub-key is generated by "key expansion" and is extracted using a Rijndael key-schedule from the main cipher key.

   Encryption method: Begins with AddKey with Key0. Then go to the loop and do SubBytes, ShiftRows, MixColumns, Addkey, each circle with different circle keys in that order for 9 circles. Then go to the final circle (circle 10) and repeat, except for MixColumns, the same previous feature in the loop.

   Decryption method: in every stage it is reverse of the encryption process, which implies that the first circle of decryption is the tenth circle of the encryption and it uses the invers functions of MixColumns, SubBytes, ShiftRows and us. You should assume the arrangement of keys and reverse it as it begins with Addkey10 instead of Addkey0 as it was in the encryption phase.
2. Key generation.

In order to generate enough keys for each circle in the encryption, decryption process, RotWord, SubBytes and XOR bitwise operation are needed. Each circle operates with different keys created by the method of key generation.

## 2.2 Fully homomorphic encryption in a binary number ring

Homomorphic encryption is a type of encryption on encrypted data that performs arbitrary computations. We may keep our confidential data in encrypted format in cloud storage, but if you want to do some calculation on cipher text, the key must be shared with cloud service providers who may allow our data to be abused. Instead, the Homomorphic Encryption approach is used to prevent sharing the key with cloud service providers. Searching, sorting, addition, multiplications performed on cipher text involves the computations.

Homomorphic encryption has drawn widespread attention from scholars for its specific success among so many cryptographies [6-7]. Popular cryptography cannot explicitly measure encrypted data, but homomorphic encryption will automatically encrypt the operational results of homomorphic encryption. In the fields of secure multi-party computing, electronic voting, cipher text scanning, encrypted mail filtering, mobile cipher, the application prospect of homomorphic encryption is broad and cheerful. Finally, security analysis is reviewed and more testing methods are highlighted.

In this encryption method, homomorphic encryption seeks to support by enabling unique types of computations to be performed on cipher text that produces an encrypted result that is also in cipher text. The product of operations performed on the plaintext is the product. Case in point, one person might add two encrypted numbers and then another person might decrypt the outcome without the significance of the individual numbers being able to be identified by any of them.

By using ideal lattices, Craig Gentry introduced completely homomorphic encryption based on bootstrapping over partially homomorphic encryption. It is restricted because, in some way, any cipher text is noisy, and this noise grows as one ad and multiplies cipher texts. Gentry have shown that a self-eMbedding recursion can turn any boot trappable Somewhat Homomorphic Encryption scheme into a Completely Homorphic Encryption. The bootstrapping procedure effectively "updates" the cipher text in the case of Gentry's "noisy" scheme by applying the decryption procedure homomorphically to it, thereby obtaining a new cipher text that encrypts the same value as before but has a lower instance of noise [8]. Whenever the noise becomes too complicated, the cipher text is regularly "refreshed."

Fully homomorphic encryption in a binary number ring. The scheme of completely homomorphic encryption, which Gentry suggested, can be considered using the example of calculations in $Z_2$ [9-13].

**Encryption**

The data encryption method can be interpreted as follows:

1. We pick an arbitrary strange number $p = 2k + 1$, which is a secret parameter. Let $m \in \{0,1\}$.
2. The number $z \in Z_2$ is compiled such that $z = 2r + m$, where $r$ is an arbitrary number. This means that $z = mmod2$
3. In the encryption method, 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.

**Decryption**

Let it be known about the encrypted number $c$ and the secret $p$. The method of decrypting the information should then include the following actions:

1. Using a secret parameter decoding $p: r = cmodp = (z + pq)modp = zmodp + (pq)modp$ where $r = cmodp$ is called noise
2. Obtaining the original bit of encryption: $m = rmod2$.

Rationale:

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 illustrates that an addition operation is a completely homomorphic encryption in a binary ring. In visual studio C++ 2019 and in the BLE Nano kit microcontroller on the mbed platform using C++, completely homomorphic encryption in the binary ring was enforced.

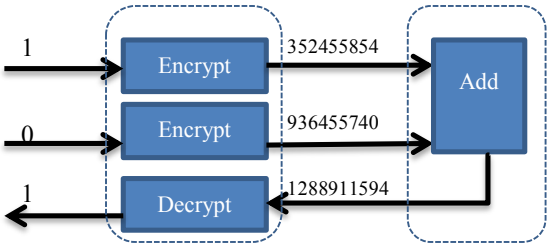



**Fig. 2.** The result of homomorphic encryption with the operation of adding.

Similarly, the operation of multiplication is checked:

$c_1 c_2 = (z_1 pq_1)(z_2 pq_2) = z_1 z_2 + p(z_1 q_2 + z_2 q_1) + p^2 q_1 == (2r_1 + m_1)(2r_2 + m_2) + (2k + 1)\big((2r_1 + m_1)q_2 + (2r_2 + m_2)q_1\big) = 4r_1 r_2 + 2(r_1 m_2 + r_2 m_1) + m_1 m_2 + 2k(2r_1 q_2 + m_1 q_2 + 2r_2 q_1 + m_2 q_1) + 2r_1 q_2 + m_1 q_2 + 2r_2 q_1 + m_2 q_1$

The decryption procedure must be applied to the results obtained, which will result in the following: $\big((c_1 c_2)modp\big)mod2 = (4r_1 r_2 + 2(r_1 m_2 + r_2 m_1) + m_1 m_2)mod2 = m_1 m_2$.

Figure 3 shows that a multiplication operation is a completely homomorphic encryption in a binary ring.

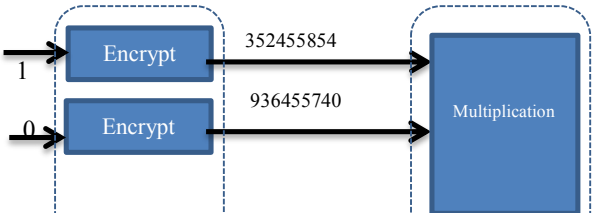



**Fig. 3.** The result of the multiplication operation of homomorphic encryption.

## 3. Performance

Here, on the mbed platform, we will implement fully homomorphic encryption in a binary ring on a BLE Nano Kit microcontroller.

How does programming operate?

The mbed microcontroller is connected to our PC via USB (Universal Serial Bus), it looks like a USB flash drive. This small disk is represented by the mbed interface and allows you to save the BLE Nano kit microcontroller hexadecimal files that we want to run directly on mbed without the need for a driver. It is not automatically loaded into the flash memory of the internal microcontroller when saving the .hex file on the mbed disk.

The mbed searches at the disk for the most recent .hex file it can find when we hit reset. If a new file exists, it uses the JTAG interface to load it into the internal FLASH memory of the microcontroller. If the current binary has already been loaded, it will not be loaded again. It then begins working the microcontroller.

How does a serial USB work?

The USB serial / com interface is also reflected by the mbed interface. Basically, this is a UART-USB bridge which connects to the UART interface. Therefore, if we send characters from the target microcontroller's UART, they will be read and transferred via USB using the mbed interface.

We use the terminal emulator "Tera Term" for programming devices.

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

1) Serial Communications: Serial Communications is the first stage. For each 8 bits of data transfer, in microcontroller serial communications with a contact style UART interface with even parity bit and 2 stop bits.

2) Protocol Parameter Selection and Baud Rate: The serial baud rate of the microcontroller contact form is regulated by an external oscillator's clock signal. There are two external oscillators for the BLE Nano Kit microcontroller. The slow clock has a frequency of 32,768 kHz. The frequency of the main clock is 16 MHz and the baud rate is about 9,600 bps.

Table 1 presents our experimental findings for selected strategies to protect privacy. On the devices used, we calculate the performance overhead and we estimate overhead memory/communication. We concentrate on the duration of primary activities/phases, such as the duration of encryption, the time of decryption. All time values are determined from 10 iterations as mean values.

**Table 1.** Running time of different algorithms in our microcontroller and pc

| Algorithms | Microcontroller BLE Nano Kit | Windows 7 |
|---|---|---|
| AES 128 bit Encrypt | 42.2 ms | 26.358 ms |
| AES 128 bit Decrypt | 39.5 ms | 29.013 ms |
| FHE in a binary ring Encrypt | 64.2 ms | 56.7363 ms |
| FHE in a binary ring Decrypt | 262.8 ms | 250.9675 ms |

The efficiency overhead of selected privacy protection techniques on the microcontroller and PC is shown in Figure 4.

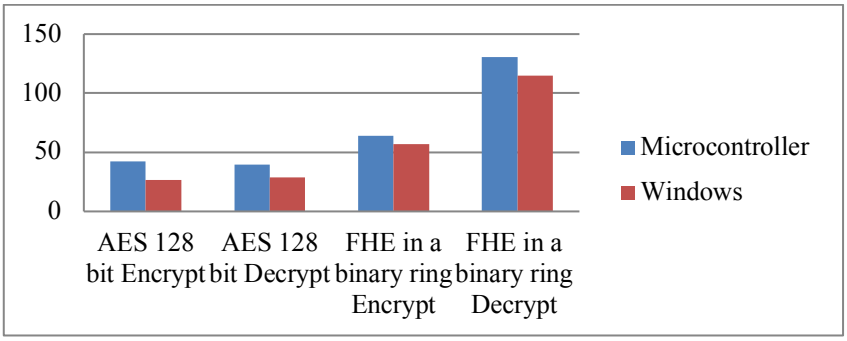



**Fig. 4.** Running Time of Different Algorithms in microcontroller BLE Nano Kit and PC

Using terminal Tera Term to submit data and cypher keys from a computer, the encryption was checked. We would then run a series of exams. We compare the findings with the Visual Studio C++ operations directly invoked from a Windows 7 x64 device environments and the BLE Nano microcontroller package using C++ on the mbed framework. Table 1 and figure 4 display the effects. From the observations, it can be shown that the output of the microcontroller is comparable to the output. The only variation in the transmission protocol that is not well configured in our implementation is caused by overhead.

## 4. Conclusion

This work presents the performance and memory limitations on different types of devices that can be used in IoT for existing cryptographic primitives and schemes. Symmetric ciphers and hash functions can now be easily integrated into IoT services using restricted devices. In this paper, in terms of speed, flexibility and protection, the use of hardware platforms to create a real-life application has proved satisfactory and promising. We have used the mbed platform's strong compatibility with the microcontroller in our research work. As one of the most popular BLE Nano Kit microcontrollers, different security features can be used in the application. However, compared to Visual Studio C++ from a Windows 7x64 computer environment due to the limited processing power and memory of the microcontroller BLE Nano Kit, BLE Nano Kit yields a fairly poor performance, especially when AES algorithms are involved with FHE in a binary ring.

## References

1. Jose Angel, BLE Nano hardware development kit for Bluetooth Low Energy (2015)
2. S. Aguilar, R. Vidal, C. Gomez, Opportunistic Sensor Data Collection with Bluetooth Low Energy. Sensors (2017)
3. C. Gomez, J. Oller, J. Paradells, Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology. Sensors (2012)
4. Atmel Corporation,"Integrating the Internet of Things: Necessary building blocks for broad market adoption", San Jose, USA: Atmel, 0776 Corporate IOT WhitePaper US 102014.
5. Jessica Chani Cahuana, Chalmers, A Search for a Convenient Data Encryption Algorithm For an Internet of Things Device. (2016)
6. Suhad Shakir, Hilal Adnan Fadhil, Zahereel I. Abdul Khabib, Rasim Azeez Kadhim. Cloud computing Data security: AES Encryption algorithm and PRT-PVD Steganography Technique. Australian Journal of Basic and Applied Sciences, **9**(19) Special (2015)
7. Daemen, Joan; Rijmen, Vincent. "AES Proposal: Rijndael". National Institute of Standards and Technology. - P. 10-17, (2003)
8. Ashokkumar C.; Ravi Prakash Giri; Bernard Menezes. IEEE European Symposium on Security and Privacy (EuroS&P). pp. 261–275, (2016)
9. 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**, P. 420-443. (2010)
10. Gentry A Fully Homomorphic Encryption Scheme. PhD thesis, Stanford University, P. 199 (2009)
11. Gentry, C. A Fully Homomophic Encryption Scheme. Ph.D. Thesis, Standford University, Stanford, CA, USA, (2009)
12. Craig Gentry. . In the 41st ACM Symposium on Theory of Computing (*STOC)*, (2009)
13. W. Lv, F. Meng, C. Zhang, Y. Lv, N. Cao, and J. Jiang. A general architecture of iot system. In 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), volume **1**, pages 659–664, July (2017)