CEUR-WS.org/Vol-3680/S4Paper14.pdf

Using Fully Homomorphic Encryption in IoT

Zhanerke E. Temirbekova[1,2], Gulzat Turken[1] and Manuel M. Barata[3]

[1]*Al-Farabi Kazakh National University, al-Farabi Ave. 71, Almaty, A15E3B9, Kazakhstan*
[2]*International Information Technology University, Manas St. 34/1, Almaty, A15M0F0, Kazakhstan*
[3]*Instituto Superior de Engenharia de Lisboa, Lizbon, Portugal*

Abstract
to protect the confidentiality of basic data. Recent advances in homomorphic encryption have made it possible to protect confidential and personal data in Internet of Things applications using schemes based on homomorphic encryption. However, being relatively young in this field of cryptography, standards and guidelines for the use of fully homomorphic encryption schemes are still evolving. The article analyzes the existing libraries in the field of homomorphic encryption. As a result of the analysis, the necessity of carrying out the operation of homomorphic encryption and division, as well as the relevance of developing the implementation of the library of homomorphic encryption of integers, is revealed. The method of homomorphic division is proposed, which allows performing the operation of separating homomorphic encrypted data. To ensure the secure storage and exchange of data between IoT constructs, a complete homomorphic encryption libraries architecture has been created and implemented, allowing all arithmetic operations to be performed on the data encrypted in various AtmelAVR microcontrollers.

Keywords
IoT system, homomorphic encryption, microcontroller, library, security

1. Introduction

IoT (Internet of things) devices are widely used in all sectors, from the health sector to manufacturing [1]. The main purpose of IoT technologies is to allow internet – connected devices to communicate with each other, exchange data, store data and perform calculations according to user requirements.

Although IoT devices are projected to reach 83 billion by 2024, the security of these devices is a major concern, with the risk of any IoT-enabled device stealing the functionality, as well as user data, if appropriate security measures are not taken [2]. According to a 2022 report by Palo Alto Networks [3], 98% of all IoT device traffic is unencrypted, indicating that personal and confidential data on the network is not stored in secret, and allows attackers to spy on unencrypted network traffic, collect personal or confidential information, and then use that data by the attacker for their personal purposes. According to SAM Seamless Network, more than 1.5 billion IoT devices were attacked in 2021, including about 900 million phishing attacks [4].
IoT devices are not by themselves an analogue of computers, they cannot perform any resource-intensive task from start to finish, they perform only some part of it, and the rest of the parts are completed by other IoT devices [5]. IoT work in a specific group or cluster, they jointly solve some problem. To protect the information transmitted between the IoT cluster of devices, they must be encrypted and we must be able to perform operations such as those with encrypted data in an unencrypted state so as not to compromise the overall result of the work.

We can create this capability through homomorphic encryption, which can be implemented in AtmelAVR microcontrollers (DFRobot Beetle BLUE, Atmega 328, Atmega 32u4, Atmega 2560) that control IoT devices in medicine, consumer electronics, and manufacturing.

DTESI 2023: Proceedings of the 8th International Conference on Digital Technologies in Education, Science and Industry, December 06–07, 2023, Almaty, Kazakhstan

temyrbekovazhanerke2@gmail.com (Zh. Temirbekova); turken.gulzat@gmail.com (G. Turken); manuel.barata@isel.pt (M. Barata)
0000-0003-3909-0210 (Zh. Temirbekova); 0000-0003-4981-514X (G. Turken); 0000-0002-8335-4052 (M. Barata)

© 2023 Copyright for this paper by its authors.
Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

One of the new directions of modern cryptosystems is homomorphic cryptography. Its distinctive feature is that this type of cryptography allows you to process encrypted data without a prior secret key, so that the result of operations with encrypted data is equal to the result of operations with open data after decryption [6]. This solves one of the problems of cryptography – the generation, storage and distribution of common session keys. This increases the level of data security – the server receives encrypted data, processes them and returns the encrypted result, while open data and encryption keys do not leave the secure segment during network interaction.

In recent years, there has been a lot of work on FHE (Full homomorphic encryption) for IoT devices in the world. S.R. Sujoy, P. Goyuri, N. Deepika (2021) show in their work that fully homomorphic encryption algorithms can be applied to IoT applications and devices, as well as aimed at ensuring higher computing speed while maintaining data confidentiality [7].

D. Goran, M. Milan, V. Pavle [8] in their work «Evaluating the implementation of homomorphic encryption in an IoT device» evaluated the features of the homomorphic encryption mechanisms of BFV and BGV and measured computational performance. The Raspberry Pi 4 evaluates the encryption schemes on the IoT platform based on the B model, showing that homomorphic encryption operations can be used in embedded devices and is primarily aimed at improving privacy and providing high bandwidth and low latency to speed up applications.

Among the representatives of the Russian scientific community, the works of the following scientists can be especially noted: I.B. Saenko, V.A. Desnitsky (Moscow), I.V. Kotenko (Sverlosk), P.D. Zegzhda (St. Petersburg).

Taking into the analyzes made, there is a need for methods, algorithms that effectively ensure the security of IoT applications and devices and determine whether the topic under consideration is an urgent problem.

The homomorphism property is used in many cryptographic systems, in particular in secure voting systems, in all kinds of collision-resistant hash functions, in the creation of closed information of search engines, and in cloud computing. The Homomorphism property guarantees the confidentiality of the processed data.

Currently, active research is being conducted in the field of homomorphic encryption. The following can be noted as the main directions of its development:

First, to create a symmetric full homomorphic encryption, you can use invariant matrix polynomials. Russian cryptographic scientist F.B. Burtyka works in this area, who proposed to carry out encryption in three rounds: in the first step, open texts are obtained that are elements of the ring, in the second step they are encoded into matrices using a secret vector, in the third step these matrices are displayed in matrix polynomials using a secret invariant matrix polynomial. Then, reverse encryption is implemented in both rounds [9, 10].

Secondly, the development of symmetric fully homomorphic linear cryptosystems based on the problem of factorization of numbers. Among the Russian scientists working in this direction, one can name A.V. Trepacheva [11], P.K. Babenka [12]. The cryptographic stability of these systems is justified by the use of complexity in solving the problem of factorization of large numbers.

Thirdly, the development of marginal systems of homomorphic encryption and information protection in cloud computing. Research in this direction is carried out by N.P. Varnovsky, S.A. Martishin, M.V. Khrapchenko, A.V. Shakurov. In conclusion, a system was obtained that does not require an additional public key and replaces the not very efficient and problematic re-encryption procedure (bootstrapping) performed on cryptographically dedicated servers [13].

2. Fully homomorphic encryption (FHE)

FHE – based data protection is a new type of security that allows you to calculate encrypted data without first re-encrypting it. However, the practical FHE solution is not available for implementation today. The most popular asymmetric and symmetric homomorphic encryption algorithms for analyzing process time, taking into account the effective and security component: Benalo [14], El Gamal [15], RSA [16], Paillier [17], Gentry's bit homomorphic encryption [18], A.

Abramov's homomorphic encryption in the field of polynomials with rational coefficients [19] and homomorphic encryption in the field of polynomials with a variable S.F. Krendelev [20] was studied. The homomorphic features and disadvantages and similarities of each of the algorithms were studied. In addition, comparisons were made with the calculation of encryption, reverse encryption times for each of the algorithms. The work carried out testing of homomorphic encryption algorithms on the microcontroller AtmelAVR (DFRobot Beetle BLUE, Atmega 328, Atmega 32u4, Atmega 2560, ESP 32). Testing to compare the performance of the encryption and reverse encryption operation calculated the average run time for 8 iterations. The work carried out testing of homomorphic encryption algorithms on the microcontroller AtmelAVR (DFRobot Beetle BLUE, Atmega 328, Atmega 32u4, Atmega 2560, ESP 32). The experimental result is shown in Table 1.

Table 1
Comparison by file size 40 MB

Cryptosystem	Benaloh	El-Gamal	RSA	Paillier	Gentry	Abramov	Krendelev
Key generation (ms)	23,67	10,808	12,25	23,31	8,51	25,03	20,12
Encryption time (ms)	6772,98	4983,26	2133,65	1672,12	13108,17	1048,69	751,84
Decryption time (ms)	2736,36	3412,96	1391,59	1336,23	4098,52	530,52	403,96
Data memory usage (bytes)	204	202	206	201	216	178	175
Use of program memory (bytes)	1689	1382	1802	1567	2976	992	867
Current strength (W)	1,15	1,03	1,21	1,12	2,24	0,63	0,57
Voltage (Volts)	2,584	2,508	2,607	2,520	2,897	2,361	2,336
Power (W)	1,15	1,03	1,21	1,12	2,24	0,63	0,57
HE categories	PHE	PHE	PHE	PHE	FHE	FHE	FHE
Homomorphic signs	Add	Multiplications	Multiplications	Add	Add, multiplications	Add, multiplications	Add, multiplications
Similarities	Homomorphism						

Based on comparisons, it was found that homomorphic encryption in the field of polynomials with a variable proposed by S.F. Krendelev is effective in terms of time in the AtmelAVR (DFRobot Beetle BLUE, Atmega 328, Atmega 32u4, Atmega 2560, ESP 32) microcontroller, in terms of the measure of memory use in the microcontroller, and in terms of current strength and voltage use. Due to the need for full homomorphic encryption algorithms for secure storage and transmission of data on IoT devices. It was found that the homomorphic encryption algorithm in the field of polynomials with a variable proposed by S.F. Krendelev requires all arithmetic operations.

In 2011, S.F. Krendelev proposed his fully homomorphic cryptosystem, the work of which is based on homomorphism in the field of polynomials with variables. Each a [illegible] Z_n number $a(x) = a_0 + a_1x + \ldots + a_kx^k$ is related to the polynomial, where k and a_i are randomly selected. For two polynomial images of numbers a_0 and b_0 in terms of structure, the free term of their sum $a(x) + b(x)$ and the product $a(x)*b(x)$ is $a(0) + b(0)$ and $a(0)*b(0)$.

Then $\varphi:Z_n[x] \to Z_n[y],\ x = c_0 + c_1y + \ldots + c_ty^t = \varphi(y)$ is a homomorphism that preserves addition and multiplication. The implementation of S.F. Krendelev is more profitable than Gentry. In addition, it has a number of disadvantages:

1. An infinite increase in the degrees of polynomials can lead to inefficiency in the calculation;

2. Although all operations are actually performed on empty terms, it is necessary to store in memory and perform calculations on polynomials of a large degree;

3. In the system of S. F. Krendelev, the operations of division and subtraction on encrypted data were not performed.

In the article, division and subtraction operations were added to the system proposed by S.F. Krendelev to encrypted data.

Implementation of the division operation into encrypted data in the system of S.F. Krendelev: Definition 1. Z – some kind of field. A polynomial in a single variable in the field Z is the formal sum of the form:

$$f(x) = a_n x^n + a_{n-1}x^{n-1} + a_1 x + a_0, \tag{1}$$

where, $a_i \in Z, i \in \{0,1,\dots,n\}\ n \in N$.

Any element of the field Z is considered a polynomial of zero degree, a polynomial of arbitrary degree with zero coefficients – a zero polynomial, a unit polynomial of the field Z – and they are denoted by $\vartheta(x)$ and $E(x)$, respectively.

In the set of all polynomials in one variable in the Z field, you can define the operations of addition and multiplication of polynomials according to the following rules. Suppose $f(x)$ – 2 is a polynomial and

$$g(x) = b_n x^n + b_{n-1}x^{n-1} + b_1 x + b_0, \tag{2}$$

where $b_i \in Z,\ j \in \{0,1,\dots,m\}\ m \in N$ is a type of the polynomial.

Now let's look at the problem of creating a homomorphism from a Z ring in A *Z[x]* ring.

Definition 1. (Ring homomorphism). let *f: A-> B*, where A and B are the rings of addition, multiplication, zero and one ring homomorphism if the following conditions implement [21]:

$$f(a +_A b) = f(a) +_B f(b) \tag{3}$$

$$f(a -_A b) = f(a) -_B f(b) \qquad 4)$$

$$f(a *_A b) = f(a) *_B f(b) \tag{5}$$

$$(a /_A b) = f(a) /_B f(b) \tag{6}$$

$$f(0_A) = 0_B \tag{7}$$

$$f(1_A) = 1_B \tag{8}$$

Let Z be given a ring of integers, a ring of polynomials *Z[x*], and *P(x)* compiled according to the 1 – algorithm. Then *P(x), P:Z→Z[x]* homomorphism.

Encryption and reverse encryption algorithm

Key generation. x_0 if the secret key of a given cipher is, say, $z_1, z_2 \in Z, z_1 > z_2$, let's show that the (6) property is executed, i.e.:

$$P(z_1 /_z z_2) = P(z_1) /_{z[x]} P(z_2)$$

$$P(z_1) = q^n f_1(x) - q^n f_1\left(\frac{p}{q}\right) + z_1 = q^n a_0 + q^n a_1 x + q^n a_2 x^2 + \cdots - q^n a_0 -$$

$$-q^{n-1}a_1 p - q^{n-2}a_2 p^2 - \cdots + z_1 = a_1 q^{n-1}(qx - p) + a_2 q^{n-2}(q^2x^2 - p^2) + \cdots z_1$$

$$\frac{P(z_1)}{P(z_1)} = \frac{a_1 q^{n-1}(qx - p) + a_2 q^{n-2}(q^2x^2 - p^2) + \cdots + z_1}{b_1 q^{n-1}(qx - p) + b_2 q^{n-2}(q^2x^2 - p^2) + \cdots + z_2}$$

$$P\left(\frac{z_1}{z_2}\right) = c_1 q^{n-1}(qx-p) + c_2 q^{n-2}(q^2x^2-p^2) + \cdots + \frac{z_1}{z_2}$$

For $\frac{P(z_1)}{P(z_1)} = P\left(\frac{z_1}{z_2}\right)$ you can select c_i as $c_0 = \forall$.

$$c_i = \frac{a_i z_2 - b_i z_1}{z_2 * P_{[x]}(z_2)}, i = \overline{1,n} \qquad \frac{a_i z_2 - b_i z_1}{z_2 * P_{[x=p/q]} z_2}$$

$$\frac{a_1 q^{n-1}(qx-p) + a_2 q^{n-2}(q^2x^2-p^2) + \cdots + z_1}{b_1 q^{n-1}(qx-p) + b_2 q^{n-2}(q^2x^2-p^2) + \cdots + z_2} =$$

$$\frac{a_1 z_2 - b_1 z_1}{z_2 * b_1 q^{n-1}(qx-p) + b_2 q^{n-2}(q^2x^2-p^2) + \cdots + z_2} * q^{n-1}(qx-p)$$
$$+ \frac{a_2 z_2 - b_2 z_1}{z_2 * b_1 q^{n-1}(qx-p) + b_2 q^{n-2}(q^2x^2-p^2) + \cdots + z_2} * q^{n-2}(q^2x^2-p^2) + \cdots$$
$$+ \frac{z_1}{z_2}$$

$$a_1 q^{n-1}(qx-p) + a_2 q^{n-2}(q^2x^2-p^2) + \cdots + z_1 = \frac{a_1 z_2 - b_1 z_1}{z_2} * q^{n-1}(qx-p) +$$
$$+ \frac{a_1 z_2 - b_1 z_1}{z_2} * q^{n-2}(q^2x^2-p^2) + \cdots + \frac{z_1}{z_2} * b_2 q^{n-2}(q^2x^2-p^2) + \cdots + z_2$$

$$a_1 q^{n-1}(qx-p) + a_2 q^{n-2}(q^2x^2-p^2) + \cdots + z_1$$
$$= q^{n-1}(qx-p)\left[\frac{a_1 z_2 - b_1 z_1}{z_2} + \frac{z_1}{z_2} * b_1\right]$$
$$+ q^{n-2}(q^2x^2-p^2)\left[\frac{a_1 z_2 - b_1 z_1}{z_2} + \frac{z_1}{z_2} * b_2\right] + \cdots + z_1$$

$$a_1 q^{n-1}(qx-p) + a_2 q^{n-2}(q^2x^2-p^2) + \cdots + z_1$$
$$= a_1 q^{n-1}(qx-p) + a_2 q^{n-2}(q^2x^2-p^2) + \cdots + z_1$$

Since $\frac{P(z_1)}{P(z_1)} = P\left(\frac{z_1}{z_2}\right)$ the image of P is a homomorphism.

Proof. let $z_1, z_2 \in Z$ show that the property (4) is executed, i.e.: $P(z_1 -_z z_2) = P(z_1) -_{z[x]} P(z_2)$,

$P(z_1) -_{z[x]} P(z_2) = (q^n f_1(x) - q^n f_1\left(\frac{p}{q}\right) + z_1) - (q^n f_2(x) - q^n f_2\left(\frac{p}{q}\right) + z_2 == q^n a_0 + q^n a_1 x + \cdots + q^n a_n x^n - q^n a_0 - q^{n-1} p a_1 - \cdots - p^n a_n + z_1 - q^n b_0 - q^n b_1 x - \cdots - q^n b_n x^n + q^n b_0 + q^{n-1} p b_1 + \cdots + p^n b_n - z_2 = (a_1 - b_1)(q^n x - q^{n-1} p) + (a_2 - b_2)(q^n x^2 - q^{n-2} p^2) + \cdots + (a_n - b_n)(q^n x^n - p^n) + z_1 - z_2$.

Let's show that one part is equal to the other:

$P(z_1 -_z z_2) = (q^n f_1(x) - q^n f_1\left(\frac{p}{q}\right) + z_1) - (q^n f_2(x) - q^n f_2\left(\frac{p}{q}\right) + z_2 = q^n(a_0 - b_0) + q^n(a_1 - b_1)x + q^n(a_2 - b_2)x^2 - q^n(a_0 - b_0) - q^{n-1}p(a_1 - b_1) - q^{n-2}p^2(a_2 - b_2) + z_1 - z_2 = (a_1 - b_1)(q^n x - q^{n-1} p) + (a_2 - b_2)(q^n x^2 - q^{n-2} p^2) + \cdots + (a_n - b_n)(q^n x^n - p^n) + z_1 - z_2$.

It is also clear that the representation of P compares 1 to 1 and 0 to 0, corresponding to the properties (3) and (4). Therefore, the representation of P is a homomorphism.

Key generation. $x_0 = \frac{p}{q}$ here $p \in Z, q \in N$ - any numbers.

Algorithm 2 (encryption). The encryption algorithm $Enc: Z \to Z[x]$ is displayed as follows.

1) $z \in Z$ — let be the number to be encrypted, $z_1 > z_2, z_2 \neq 0$;

2) We construct a polynomial so that the coefficients $a_0, a_1, \ldots a_n \in Z$ are chosen at random $f(x) = a_0 + a_1 x + \ldots + a_n x^n$;

3) $f(x_0) = f\left(\frac{p}{q}\right) = a_0 + a_1(\frac{q}{p}) + ... + a_n(\frac{q}{p})^n$ calculated. From this $q^n f\left(\frac{p}{q}\right) = q^n a_0 + q^{n-1} p a_1 + ... + p^n a_n$ it is obtained that, where $q^n f\left(\frac{p}{q}\right) \in Z$;

4) Then the encryption polynomial for *z* will look like this:

$g_z(x) = q^n f(x) - q^n f(x) + z \in Z[x]$

Algorithm 3 (decryption). Let's look at the reverse encryption algorithm $Dec: Z[x] \to Z$ this is reverse to the 2 algorithm.

1) Polynomial $g_{z'}^{''} \in Z[x]$- encrypted data $x_0 \in Q$ - is a secret key.

2) For decrypted x_0 at the point $g_{z'}^{''}$ is calculated.

3) Then $z' = g_{z'}^{''}(x_0) \in Z$ - decrypted data.

According to theorem 1, for $z_1, z_2 \in Z$ the encryption is homomorphic if the following properties are met:

$$z_1 + z_2 = Dec(Enc(z_1) + Enc(z_2)) \tag{9}$$

$$z_1 * z_2 = Dec(Enc(z_1) * Enc(z_2)) \tag{10}$$

In the work of S.F. Krendelev homomorphic encryption for the addition and multiplication operation was considered, in this work the method of full homomorphic encryption, which performs subtraction and division operations, is proposed.

Example of system operation:

Key: $x_0 = \frac{11}{7}$.

$Enc: Z \to Z[x]$.

1) z_1 =7, $z_2 = 5$;

2) $f_1(x) = 2 + 4x - 6x^2 + 3x^3$, $f_2(x) = 5 - 8x + 6x^2 + 9x^3$;

3) $q^n f_1\left(\frac{p}{q}\right) = 7^3 * 2 + 7^2 * 11 * 4 - 6 * 7 * 11^2 + 3 * 11^3 = 175$;

$q^n f_2\left(\frac{p}{q}\right) = 7^3 * 5 - 8 * 7^2 * 11 + 6 * 7 * 11^2 + 11^3 * 9 = 14464$;

4) Encryptable polynomial for $z_1$and z_2:

g_{z_1}=$1029x^3 - 2058x^2 + 1372x - 1060$, g_{z_2}=$3087x^3 + 2058x^2 - 2744x - 12744$;

Subtraction and division Homomorphism of the system:

$g_{z_1} - g_{z_2} = -2058x^3 - 4116x^2 + 4116x + 11684$;

$\frac{g_{z_1}}{g_{z_2}} = \frac{1}{3} + \frac{-2744x^2 + 2286\frac{2}{3}x + 3188}{3087x^3 + 2058x^2 - 2744x - 12744}$;

$Dec: Z[x] \to Z$.

$g_{z_1}x_0 - g_{z_2}x_0 = -2058\left(\frac{11}{7}\right)^3 - 4116\left(\frac{11}{7}\right)^2 + 4116 * \frac{11}{7} + 11684 = 2$;

$\frac{g_{z_1}x_0}{g_{z_2}x_0} = \frac{1}{3} + \frac{-2744\left(\frac{11}{7}\right)^2 + 2286\frac{2}{3} * \frac{11}{7} + 3188}{3087\left(\frac{11}{7}\right)^3 + 2058\left(\frac{11}{7}\right)^2 - 2744 * \frac{11}{7} - 12744} = 1{,}4.$

```
Algorithm 1
  $size \leftarrow RANDOM$
  $n \leftarrow RANDOM$
  $numbers1[size], numbers2[size]$
  $a[size][n+1]$
  $kopmushe1[size][n+1], kopmushe2[size][n+1]$
  $p, q \leftarrow RANDOM$
  $x0 \leftarrow p \div q$
  $ENCRYPTION(numbers1, numbers2, size, n, a, p, q)$
  $GOMOMORPHISM$
  -
  for $i \leftarrow 0$ to $size$ do
      for $j \leftarrow 0$ to $n+1$ do
          $gomAzaitu[i][j] \leftarrow (kopmushe1[i][j] - kopmushe2[i][j])$
      end for
  end for
  /
  for $i \leftarrow 0$ to $size$ do
      $resKopmushe \leftarrow kopmushe1[i]/kopmushe2[i]$
      $qaldyq \leftarrow kopmushe1[i]\ mod\ kopmushe2[i]$
      $bolu[i] \leftarrow resKopmushe + qaldyq \div kopmushe2[i]$
  end for
  $DECRYPTION(gomAzaitu, bolu, x0)$
```

Figure 1: Pseudo-code of the proposed subtraction and division operations

3. Development a library for fully homomorphic encryption

Building a library for full homomorphic encryption in a ring of polynomials with an advanced variable

In the Arduino IDE integrated development environment, in C++, a static library was created for full homomorphic encryption in a ring of polynomials with a homomorphic modified variable in a ring of polynomials with a variable for different microcontrollers.

The boost library, which supports large numbers, is used to support performing multiple operations on encrypted numbers and reduce computational inaccuracies (value approximations) [22].

When implementing the modified library, it was faced with the following tasks:

- ability to edit integers;
- full homomorphic encryption;
- support for all mathematical operations, including the arithmetic division operation.

To support the separation operation, a library architecture was implemented based on the homomorphic separation method mentioned above. The library is represented by cryptographic, mathematical classes and a class responsible for basic information. The architecture of the created library is shown in Figure 2.

The Secret Key class works with data about the Secret Key used in the cryptographic algorithm. This provides the ability to create a new key, generate random and use it. The current library implementation uses a random number generator from the standard library with automatic randomization relative to the current time to generate keys and polynomial coefficients. A pseudo-random number generator from the standard library is recognized as cryptographically unreliable because it uses a linear congruent method. In this regard, an interface was introduced that allowed the use of random generator inputs. By agreement, it is recommended to use a random number generator from the Boost library. It uses non-deterministic random number generation and is cryptographically secure.

Encrypted Data is a module in which the main data type is a homomorphic encrypted number. In the encryption class, the possibilities of creating new cipher texts using open texts and secret keys (data encryption operation), obtaining open data from encrypted data based on the encryption key (data encryption operation) are implemented.

Homomorphism-performs the basic mathematical operations of subtraction, addition, division, multiplication on encrypted data. Encryption and reverse encryption are performed using pre-generated keys or by transmitting secret parameters. The class also implements all the mathematical operations necessary for polynomials, namely division, multiplication, subtraction and addition.

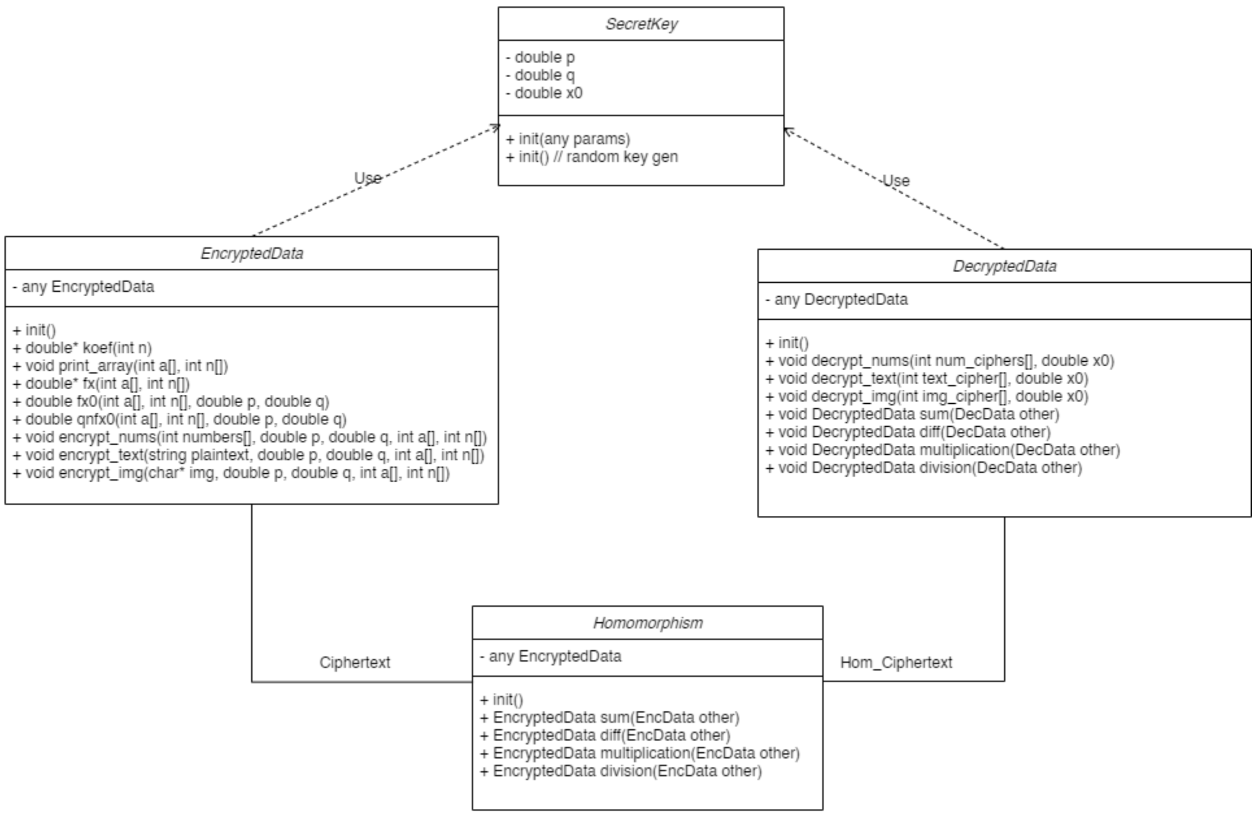

SecretKey
- double p
- double q
- double x0
+ init(any params)
+ init() // random key gen
Use
Use
EncryptedData
- any EncryptedData
+ init()
+ double* koef(int n)
+ void print_array(int a[], int n[])
+ double* fx(int a[], int n[])
+ double fx0(int a[], int n[], double p, double q)
+ double qnfx0(int a[], int n[], double p, double q)
+ void encrypt_nums(int numbers[], double p, double q, int a[], int n[])
+ void encrypt_text(string plaintext, double p, double q, int a[], int n[])
+ void encrypt_img(char* img, double p, double q, int a[], int n[])
DecryptedData
- any DecryptedData
+ init()
+ void decrypt_nums(int num_ciphers[], double x0)
+ void decrypt_text(int text_cipher[], double x0)
+ void decrypt_img(int img_cipher[], double x0)
+ void DecryptedData sum(DecData other)
+ void DecryptedData diff(DecData other)
+ void DecryptedData multiplication(DecData other)
+ void DecryptedData division(DecData other)
Ciphertext
Hom_Ciphertext
Homomorphism
- any EncryptedData
+ init()
+ EncryptedData sum(EncData other)
+ EncryptedData diff(EncData other)
+ EncryptedData multiplication(EncData other)
+ EncryptedData division(EncData other)

Figure 2: Library architecture of complete homomorphic encryption in a ring of polynomials with an advanced variable

Decrypted Data encrypts encrypted data using a secret key and reverse Homomorphism.

Checking for Homomorphism. An array named resBuf is opened to check Homomorphism by the addition operation. Its measure is taken in accordance with which the length of the polynomial is greater. Next, all elements of the resBuf are taken as 0. Then the index polynomial values corresponding to each index are added throughout the cycle. To calculate the sum value, open the variable resu=0 and find resu+=resbuff[0]*x0. The same is done for the rest of the polynomial.

The Homomorphism check by subtraction operation is the same as checking the algorithm for adding the numbers of the first polynomial to the resBuf array and subtracting the numbers of the second polynomial from it.

Checking Homomorphism by the multiplication operation. The array named Kob will open. Its Dimension is [(first polynomial dimension)*(second polynomial dimension)]. Divided into two columns, in the first column are the coefficients that precede x, and in the second column are the degrees of that X. The Resu array is opened and the same ranks are added to it.

Checking Homomorphism by the division operation. The Gorner scheme was used to perform the division operation into encrypted data. When dividing a polynomial by a polynomial, the quotient and the remainder are obtained. The created library can be used in the client program in 3 Types: 1. HomomorphicControllerVersion_01.C, that is, the library is stored on the computer as a file, connecting the microcontroller to the computer through the Com port and calling the necessary functions. 2.Download The created library from GitHub in the form of a zip archive, and the user will use it by installing it in the form of a driver. 3.built-in library architecture written in microcontroller construction from Figures 3 can be seen.

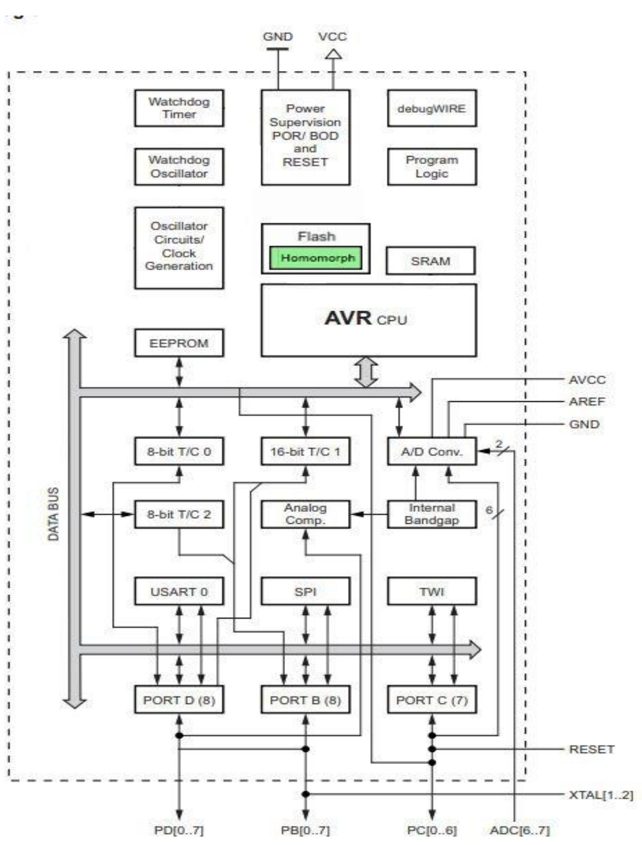


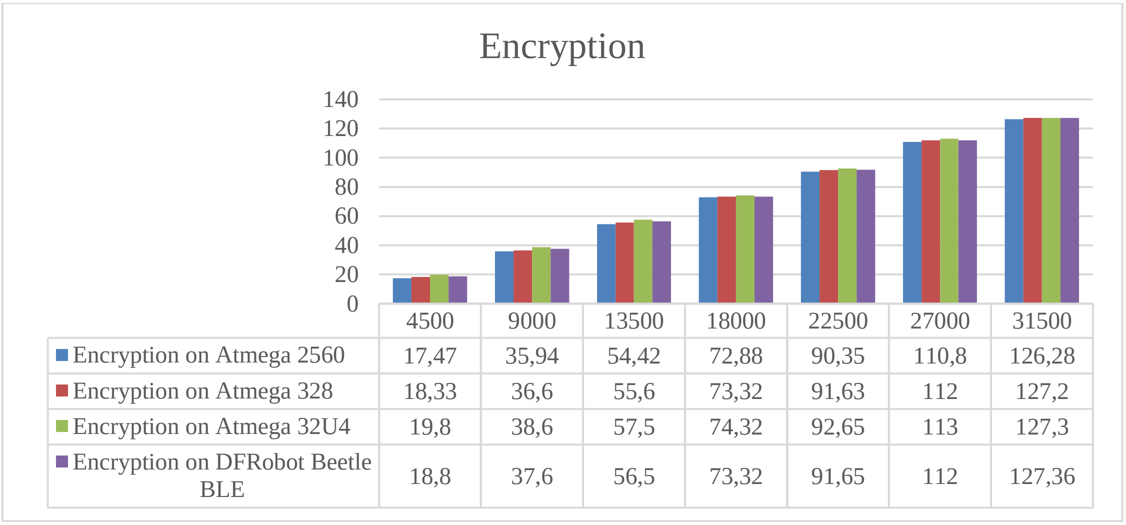
Figure 3: Library architecture installed on the Atmega 328 microcontroller

One of the most important tasks of the research work is to evaluate the performance of the created libraries in various AtmelAVR microcontrollers. The Atmel ATmega328P provides the following features: 32K bytes of in-system programmable flash with read-while-write capabilities, 1K bytes EEPROM, 2K bytes SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, a byte oriented 2-wire serial interface, an SPI serial port, a 6-channel 10-bit ADC (8 channels in TQFP and QFN/MLF packages), a programmable watchdog timer with internal oscillator, and five software selectable power saving modes. The idle mode stops the CPU while allowing the SRAM, Timer/Counters, USART, 2-wire serial interface, SPI port, and interrupt system to continue functioning. The power-down mode saves the register contents but freezes the oscillator, disabling all other chip functions until the next interrupt or hardware reset. Initially, on the Atmega 328 microcontroller TGSH in a ring of polynomials with a modified variable, comparisons were made with different pairs of two-digit numbers: key generation, encryption, reverse encryption, (addition, subtraction, multiplication, division) times, and the complexity of the algorithm was considered. The result can be seen in Table 2.

Compared to the multiplication operation with the addition operation, the fact that the multiplication operation is performed worse than the addition operation shows that the complexity of the algorithm is higher. It was also found that memory operations take up most of the processor time. It can be concluded that these schemes may be suitable for use in specific software products. The table shows linear growth, which is a consequence of the increase in the amount of memory divided by polynomial coefficients as a result of addition, as well as the complication of the addition operation on data of a larger length. However, since the operation of adding polynomials $O(n)$ has algorithmic complexity, this increase can be considered Insignificant. When multiplying, there is a quadratic increase and, unlike adding, there is a significant decrease in speed. This is because when multiplying polynomials, their coefficients are multiplied, which requires more memory allocation than adding coefficients, and the operation of multiplying polynomials has a higher algorithmic complexity - $O(n^2)$.

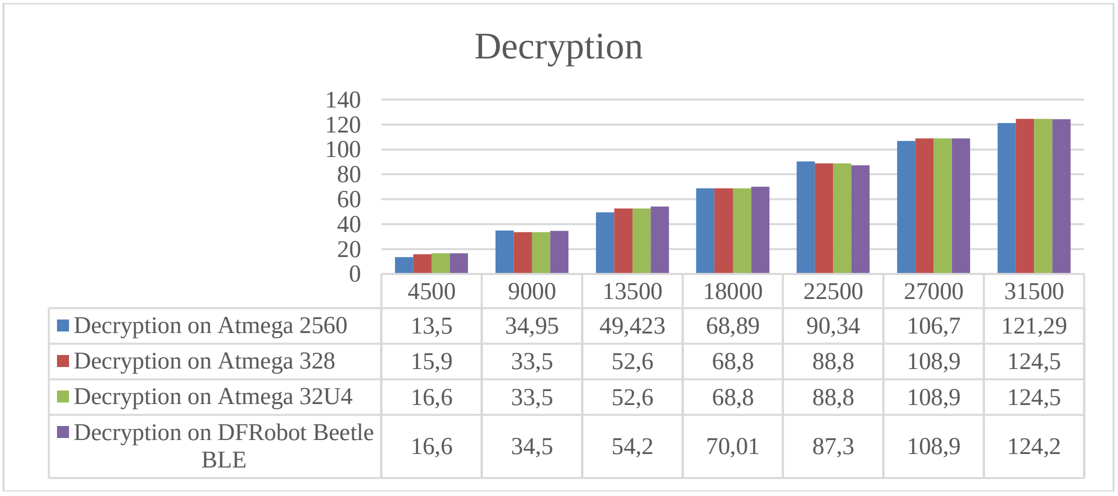
Table 2
Execution time of the created library on the Atmega 328 microcontroller

2-digit, different even numbers	Key generation, encryption, addition, decryption, s.	Key generation, encryption, subtraction, decryption, s.	Key generation, encryption, multiplication, decryption, s.	Key generation, encryption, divide, decryption, s.
100	1,95	1,92	2,52	2,44
200	2,625	2,605	3,44	3,33
300	3,736	3,727	4,94	4,77
400	5,032	5,027	6,68	6,67
500	6,931	6,947	9,28	9,2



	4500	9000	13500	18000	22500	27000	31500
🟦 Encryption on Atmega 2560	17,47	35,94	54,42	72,88	90,35	110,8	126,28
🟥 Encryption on Atmega 328	18,33	36,6	55,6	73,32	91,63	112	127,2
🟩 Encryption on Atmega 32U4	19,8	38,6	57,5	74,32	92,65	113	127,3
🟪 Encryption on DFRobot Beetle BLE	18,8	37,6	56,5	73,32	91,65	112	127,36

a-data encryption by the number of different characters



	4500	9000	13500	18000	22500	27000	31500
🟦 Decryption on Atmega 2560	13,5	34,95	49,423	68,89	90,34	106,7	121,29
🟥 Decryption on Atmega 328	15,9	33,5	52,6	68,8	88,8	108,9	124,5
🟩 Decryption on Atmega 32U4	16,6	33,5	52,6	68,8	88,8	108,9	124,5
🟪 Decryption on DFRobot Beetle BLE	16,6	34,5	54,2	70,01	87,3	108,9	124,2

b – decryption by the number of different characters

Figure 4. FH encryption in a ring of polynomials with variables and evaluation of the reverse encryption speed on a different microcontroller

When testing open data on various AtmelAVR microcontrollers, the Atmega 2560 microcontroller showed good calculations in terms of performance. Mega is designed in such a way that before writing a new code, the reboot is carried out not by pressing a button on the platform, but by the program itself. One of the ATmega8U2 data flow control lines is connected to the ATmega2560 recovery PIN via a 100 NF capacitor. This is network activation, i.e. a low-level signal resets the microcontroller. The Arduino program, using this function, loads the code with one click of the Download button in the programming environment. Low-level signaling in the data flow control network is coordinated with the start of code writing, which reduces bootstrap timeout.

4. Conclusion

The article investigated the problems of full homomorphic encryption modified in the AtmelAVR microcontroller and obtained the following results:

1. Analysis of data protection methods and devices in the IoT device cluster has been developed;
2. Improved homomorphic encryption library used in AtmelAVR microcontroller;
3. A library architecture has been created on the AtmelAVR microcontroller to ensure the security of the IoT device cluster;

Currently, Data Protection during the exchange of information is one of the most important tasks not only for traditional networks, but also for the rapidly developing segment of the Internet of things. In order to ensure data security for users of AtmelAVR microcontrollers, the homomorphiccontroller_version01 library was implemented in the work. It contains encryption library files compiled for different AtmelAVR microcontroller families.

As part of this work, the problem of homomorphic division of integers is considered, a method for implementing homomorphic division is proposed and described, and practical examples of using the above method are given. The practical value of the work lies in solving one of the problems of homomorphic encryption - the implementation of homomorphic separation makes it possible to expand the scope of practical application of homomorphic encryption in such areas as cloud computing, solving information protection problems, and machine learning.

5. References

[1] Syed A.S., Sierra-Sosa D., Kumar A. et al. (2021). IoT in Smart Cities: A Survey of Technologies, Practices and Challenges. *Smart Cities.* Vol.4, Issue 2. P. 429-475.

[2] Smith S. (2024). IoT Connections to reach 83 billion by 2024 driven by maturing industrial use cases. URL: https://www.juniperresearch.com/press/iot-connections-to-reach-83-bn-by-2024.

[3] Divy T.S., Akash P., Aishwariya B., et al. (2022). Recent Advancements of Internet of Medical Things (IoMT): Challenges and Future Opportunities with Emerging Technologies. ICACRS. Vol. 3, Issue 2. P. 278-283.

[4] Iliar Ch., Shea Sh.. IoT security (internet of things security). URL: https://www.techtarget.com/iotagenda/definition/IoT-security-Internet-of-Things-security. 12.02.2023.

[5] Dong D.C., Peng H.T. (2020). Secured Data Transmission in IoT using Homomorphic Encryption. Computers and Electrical Engineering. Vol. 4, № 3, P. 845-867.

[6] Brakerski, Z.; Gentry, C.; Vaikuntanathan, V. (2012). Fully homomorphic encryption without bootstrapping. In Proceedings of the Conference on Innovations in Theoretical Computer Science (ITCS), Cambridge: MA USA, P. 309 – 325.

[7] Goran D., Milan M., Pavle V. (2021). Evaluation of Homomorphic Encryption Implementation in IoT Device. Journal of Electrical Systems and Information Technology. Vol. 8, № 3. P. 568-592.

[8] Sujoy S.R., Goyuri P., Deepika N. (2021). Interoperability in IoT for Smart Systems. *SMIT*. Vol.2, Issue 1. P. 364-402.
[9] Burtyka F.B. (2014). Symmetric fully homomorphic encryption using irreducible matrix polynomials. Izvestiya SFU. Technical sciences. No. 8. P. 107-122.
[10] Burtyka F.B. (2014). Batch symmetric fully homomorphic encryption based on matrix polynomials. Proceedings of the Institute of System Programming of the Russian Academy of Sciences. Vol. 26. No. 5. P. 99-116.
[11] Trepacheva A.V. (2015). Cryptanalysis of symmetric fully homomorphic linear cryptosystems based on the number factorization problem. Izvestiya SFU. Technical sciences. № 5 (166). Pp. 89-102.
[12] Trepacheva A.V., Babenko L. K. (2015). Formal cryptanalysis of completely homomorphic systems using the problem of factorization of numbers. Information counteraction to threats of terrorism. No. 24. P. 283-286.
[13] Varnovsky N. P., Martishin S. A., Khrapchenko M. V., Shakurov A.V. (2015). Threshold systems of homomorphic encryption and information protection in cloud computing. Programming. No. 4. P. 47-51.
[14] Benaloh B., Lindell Y. (2007). Introduction to Modern Cryptography: Principles and Protocols. Chapman & Hall/CRC. P. 385-395.
[15] Evaluation of the strength of the El Gamal cryptosystem. Technical Sciences in Russia and abroad: materials of the IV International Scientific Conference. Moscow: Buki Vedi, 2015. P. 14-16.
[16] Naehrig, Michael, Kristin Lauter, and Vinod Vaikuntanathan (2018). Can homomorphic encryption be practical? International Journal of Electrical & Computer Engineering. Vol. 8, Issue 3. P. 1720-1730.
[17] Paillier P. (2016). Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. *Appl. Sci.* Vol. *6. P.* 162-1-162-17.
[18] Gentry, Craig (2009). Fully homomorphic encryption using ideal lattices. In Proceedings of the forty-first annual ACM symposium on Theory of computing, pp. 169-178.
[19] Abramov A. (2009). Homomorphic encryption. News of the SFU. Technical sciences. № 5 (166). P. 89-102.
[20] A.O. Zhirov, O.V. Zhirova, S.F. Krendelev (2011). Secure cloud computing using homomorphic cryptography. Izvestiya MSU. Technical sciences. № 3 (106). P. 45-67.
[21] Smart N.P., Vercauteren F. (2010). Fully homomorphic encryption with relatively small key and ciphertext sizes, public Key Cryptography. PKC Springer Berlin Heidelberg, Vol. 6056, P. 420-443.
[22] Temirbekova Zh.E., Pyrkova A.Yu. (20220. Improving teachers' skills to integrate the microcontroller technology in computer engineering education. Education and information technology. Vol. 6. P. 656-692.
[23] Temirbekova Zh.E., Pyrkova A.Yu. (2020). Using FHE in a binary ring Encryption and Decryption with BLE Nano kit microcontroller. E3S Web of Conferences 202 (ICENIS 2020), P. 687-712.