UDC 004.021

MODIFIED CHANGE-OF-BASIS CONVERSION METHOD IN $GF(2^m)$

Dychka I. A. – Dr. Sc., Professor, Dean of the Faculty of Applied Mathematics, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.

Legeza V. P. – Dr. Sc., Professor of the Computer Systems Software Department, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.

Onai M. V. – PhD, Associate Professor of the Computer Systems Software Department, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.

Severin A. I. – student of the Computer Systems Software Department, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.

ABSTRACT

Context. When cryptographic applications and data transmission control systems are implementing, there is a need for quick methods for performing operations on finite field elements. The object of the study is the processes of encryption, decryption and transmission of information using the Galois fields. The subject of the study is the methods and algorithms for calculations in the Galois fields in polynomial and normal bases.

Objective. The purpose of this study is to analyze the methods of performing operations in the Galois field depending on the chosen basis (polynomial, normal) and modification of the element conversion method from the polynomial basis to the normal and vice versa, as well as the development of a new method for generating normal polynomials in order to improve the time characteristics.

Method. In this paper, a comparative analysis of the processes of performing basic operations in the polynomial and normal bases is performed (addition, multiplication, multiplicative inverse element calculation, division, exponentiation, Frobenius operation), and the process of conversion from one basis to another is considered and analyzed. The methods of conversion between bases depending on different input data, in particular, parameters p and m of the field, are investigated. A method for the finding normal polynomials among the irreducible and modified approach for constructing a conversion matrix between bases are proposed.

Results. Existing and proposed algorithms are implemented in the C# programming language in the Visual Studio 2015 development environment. For experimental research, a software has been developed that allows performing calculations using the polynomial and normal representation of $GF(p^m)$ elements, to specify different input parameters p and m, and also receive different sets of test data depending on the normal polynomials of the Galois field.

Conclusions. The obtained experimental results of the methods and algorithms for performing operations on the elements of $GF(2^m)$ in the given bases showed that the proposed method for finding normal polynomials for the conversion between bases of binary fields gives an increase in speed over 15 times for the parameter $m > 14$; the proposed approach for constructing a conversion matrix gives an increase in the speed of more than 5 times for the parameter $m > 12$.

KEYWORDS: finite field, Galois field, polynomial basis, normal basis, irreducible polynomial, normal polynomial.

ABBREVIATIONS

AES is an Advanced Encryption Standard;

QR code is a Quick Response Code.

NOMENCLATURE

$GF(p)$ is a finite field of order p, where p is prime;

$GF(p^m)$ is a Galois field of order p^m, where p is prime, $m \in N$;

$f(x)$ is an irreducible polynomial, a polynomial which is not equal to a constant and cannot be expanded into the factors in a given field;

mod is the modulo operation which gives the remainder after division of one number or polynomial by another;

A is a matrix that represents the elements of $GF(p^m)$ in a polynomial basis;

B is a matrix that represents the elements of $GF(p^m)$ in a normal basis;

S is a conversion matrix from normal basis to polynomial;

$GF(2^m)$ is a binary Galois field, where operations can be performed faster because of the field elements representation features;

M is an additional matrix for multiplying in normal basis;

$\oplus$ is a XOR operation (exclusive disjunction), a logical operation that outputs true only when inputs differ (one is true, the other is false);

O notation is a mathematical notation that describes the limiting behavior of a function when the argument tends towards a particular value or infinity;

$Tr(\mathbf{x})$ is a trace of polynomial;

C# is a multi-paradigm programming language.

INTRODUCTION

The finite fields are increasingly used in the systems of protection and transmission of information because they use arithmetic for integers which allow avoiding work with float numbers and allows you to represent data in a convenient way for a computer [1].

Galois fields are most commonly used in cryptography, correction of data transmission errors, wavelet transformations, and filters construction in the digital processing of signals. This is because every byte in a computer can be represented as a vector in a binary finite field, and operations using machine arithmetic are

DOI 10.15588/1607-3274-2020-2-12

very simple to implement. Let's analyze the usage of finite fields in processing data and consider the peculiarities of their application.

Calculations in modern block ciphers are performed according to the rules of finite fields arithmetic. For example, the international standard of encryption AES (Advanced Encryption Standard) [2–3] and national cipher "Kalyna" (from 2015) [4] use transformations based on the Galois fields arithmetic.

The elliptic curve cryptography [5–6], which is the basis for building a digital signature, cryptographic protocols, and certificates, is another important area of application of Galois fields. This area of asymmetric cryptography studies operations on points of elliptic curves whose coordinates are elements of finite fields. Adding and multiplying points on a scalar are the basic operations over an elliptic curve whose computational complexity varies depending on the basis in which the elements are represented.

One of the important applications of Galois fields is the error control coding [7–8]. This area studies the methods for detecting and fixing data transfer errors. For example, Reed-Solomon's cyclic codes use $GF(2^m)$, their arithmetic is used in reading QR codes.

The data conversion is an important part of image, video and audio processing. It can be implemented by the construction of digital filters that are based on computations in finite fields.

The object of study is the processes of encryption, decryption and transmission of information using Galois fields.

The subject of study is the methods and algorithms for performing calculations in Galois fields in polynomial and normal bases.

Operations over elements of finite fields are performed at different times, depending on the representation of the elements. So, the problem of using different bases depending on the frequency of operations is actual. It is also important to build efficient methods for converting between bases and modifying them.

The purpose of the work is to analyze the methods of performing operations in the Galois field depending on the chosen basis (polynomial, normal) and modification of the element conversion method from the polynomial basis to the normal and vice versa, as well as the development of a new method for generating normal polynomials in order to improve the time characteristics.

1 PROBLEM STATEMENT

Suppose the input data represented as element $\mathbf{x}_A$ in the polynomial basis $A=\left\{1,t,t^2,\ldots,t^{m-1}\right\}$ of Galois field $GF(p^m)$ with an irreducible polynomial $f(t)$ where $t\in GF(p)$, $\mathbf{x}_A,\mathbf{x}_B\in GF(p^m)$.

The problem converting given Galois field element $\mathbf{x}_A$ to element $\mathbf{x}_B$ of normal basis $B=\left\{\alpha_0=t,\alpha_1=t^p,\alpha_2=t^{p^2},\ldots,\alpha_{m-1}=t^{p^{m-1}}\right\}$ can be represented by finding the irreducible polynomial $g(t)$ for which the following statements hold:

1. S is a conversion matrix from normal basis to polynomial constructed using Galois field arithmetic with the irreducible polynomial $g(t)$.
2. The inverse matrix S^{-1} exist.
3. $\mathbf{x}_B=S^{-1}\cdot\mathbf{x}_A$.

2 REVIEW OF THE LITERATURE

Galois field $GF(q)$ is a field that consists of a finite number of elements. The number of its elements is called the order of the field. The order of the field is a power of a prime number, that is $q=p^m$, where p is a prime number, and m is any positive integer. The value of p is called the characteristic of the finite field [5, 10]. The simplest Galois field is obtained when $m=1$ – a field of residues by the modulus of a prime p: $GF(p)=\{0,1,\ldots,p-1\}$. In this case, 1 is an identity element relative to the multiplication operation, and 0 is an identity element relative to the addition. If $m>1$, then $GF(p^m)$ is called the extension of $GF(p)$.

Elements of $GF(p^m)$ can be represented in different bases [11]. The basic basis for computing in the Galois field is polynomial. In this basis, the elements of the field are represented by degrees of an element t: $A=\left\{1,t,t^2,\ldots,t^{m-1}\right\}$. Operations in such an algebraic structure are performed by the module of an irreducible polynomial of degree m (a polynomial which is not equal to a constant and cannot be factorized in a given field). For an extended Galois field, it is an analog of a prime number in $GF(p)$ where operations perform by modulo p. In this case, the field elements are represented by polynomials $a_{m-1}x^{m-1}+a_{m-2}x^{m-2}+\ldots+a_1x+a_0$, where $a_i\in GF(p)$. Elements of the field can be represented using different representations, in particular, the most common are the exponential, polynomial, vector and logarithmic. To construct $GF(p^m)$, first determine the generator (primitive) element α, and then by dividing the elements α^i by an irreducible polynomial (or substitutions from it) find the desired set.

Consider an example of constructing the extended Galois field. To obtain elements of $GF(p^m)$ in a polynomial basis it is necessary to choose an irreducible polynomial of degree m. Let $p=2, m=3$.

Let's find all irreducible polynomials for a given field. To do this, let's check the remainder of the division of the polynomials of degree m by all polynomials of lower powers (if the polynomial is irreducible, the remainder should not be equal to zero). There are 8 polynomials of degree 3. Let's check each of them.

For x^3:

$$x^3 \bmod x = 0,$$

DOI 10.15588/1607-3274-2020-2-12

hence, the polynomial isn't irreducible.

For x^3+1:

$$\left(x^3+1\right)\bmod x=1,$$

$$\left(x^3+1\right)\bmod\left(x+1\right)=0,$$

hence, the polynomial isn't irreducible.

For x^3+x:

$$\left(x^3+x\right)\bmod x=0,$$

hence, the polynomial isn't irreducible.

For x^3+x+1:

$$\left(x^3+x+1\right)\bmod x=1,$$

$$\left(x^3+x+1\right)\bmod\left(x+1\right)=1,$$

$$\left(x^3+x+1\right)\bmod x^2=x+1,$$

$$\left(x^3+x+1\right)\bmod\left(x^2+1\right)=1,$$

$$\left(x^3+x+1\right)\bmod\left(x^2+x\right)=1,$$

$$\left(x^3+x+1\right)\bmod\left(x^2+x+1\right)=x,$$

hence, the polynomial is irreducible.

For x^3+x^2:

$$\left(x^3+x^2\right)\bmod x=0,$$

hence, the polynomial isn't irreducible.

For x^3+x^2+1:

$$\left(x^3+x^2+1\right)\bmod x=1,$$

$$\left(x^3+x^2+1\right)\bmod\left(x+1\right)=1,$$

$$\left(x^3+x^2+1\right)\bmod x^2=1,$$

$$\left(x^3+x^2+1\right)\bmod\left(x^2+1\right)=x,$$

$$\left(x^3+x^2+1\right)\bmod\left(x^2+x\right)=1,$$

$$\left(x^3+x^2+1\right)\bmod\left(x^2+x+1\right)=x+1,$$

hence, the polynomial is irreducible.

For x^3+x^2+x:

$$\left(x^3+x^2+x\right)\bmod x=0,$$

hence, the polynomial isn't irreducible.

For x^3+x^2+x+1:

$$\left(x^3+x^2+x+1\right)\bmod x=1,$$

$$\left(x^3+x^2+x+1\right)\bmod\left(x+1\right)=0,$$

hence, the polynomial isn't irreducible.

Let's choose an irreducible polynomial – x^3+x+1. Then the field elements for $GF(2^3)$ are got in polynomial basis according to the algorithm:

1. Determine the primitive element. For a polynomial basis, it is traditionally equal to $\boldsymbol{\alpha}=x=(0;1;0)$.

2. For $i=\overline{0\ldots 2^3-2}=\overline{0\ldots 6}$ calculate α^i:

$$\boldsymbol{\alpha^0}=1=(0;0;1),$$

$$\boldsymbol{\alpha^1}=x=(0;1;0),$$

$$\boldsymbol{\alpha^2}=x^2=(1;0;0),$$

$$\boldsymbol{\alpha^3}=x^3=x^3+x^3+x+1=x+1=(0;1;1),$$

$$\boldsymbol{\alpha^4}=x^4=x^3\cdot x=(x+1)\cdot x=x^2+x=(1;1;0),$$

$$\boldsymbol{\alpha^5}=x^5=x^4\cdot x=\left(x^2+x\right)\cdot x=x^3+x^2=$$

$$=x^2+x+1=(1;1;1),$$

$$\boldsymbol{\alpha^6}=x^6=x^5\cdot x=\left(x^2+x+1\right)\cdot x=x^3+x^2+x=$$

$$=x^2+1=(1;0;1),$$

During the exponentiation of the primitive element to a greater degree, the elements are repeated, for example:

$$\boldsymbol{\alpha^7}=x^7=x^6\cdot x=\left(x^2+1\right)\cdot x=x^3+x=1=$$

$$=(0;0;1)=\boldsymbol{\alpha^0}.$$

Hence, elements of $GF(2^3)$ for an irreducible polynomial x^3+x+1 have the form shown in Table 1.

Table 1 – Elements of $GF(2^3)$ with an irreducible polynomial x^3+x+1 in a polynomial basis

x^2	x^1	1	Exponent of α
0	0	0	–
0	0	1	0
0	1	0	1
1	0	0	2
0	1	1	3
1	1	0	4
1	1	1	5
1	0	1	6

An alternative way of representing elements is the normal basis, where they are represented as follows $B=\left\{\alpha_0=t,\alpha_1=t^p,\alpha_2=t^{p^2},\ldots,\alpha_{m-1}=t^{p^{m-1}}\right\}$.

To construct $GF(p^m)$ in a polynomial basis, it is necessary to choose an irreducible polynomial of degree m. To represent the polynomial elements of this field in a normal basis, the chosen polynomial needs to be normal. Therefore, it is necessary to choose only polynomials that are normal from the set of irreducible polynomials.

There are different approaches to convert the elements of the Galois field from one basis to another. Consider a classic algorithm using an inverse matrix for converting from a polynomial basis to a normal [6]:

1. Express the basic elements of a normal basis $B=\left\{\alpha_0=t,\alpha_1=t^p,\alpha_2=t^{p^2},\ldots,\alpha_{m-1}=t^{p^{m-1}}\right\}$ through the basic elements of a polynomial basis $A=\left\{1,t,t^2,\ldots,t^{m-1}\right\}$.

DOI 10.15588/1607-3274-2020-2-12

2. Construct the change-of-basis conversion matrix S (elements of the normal basis are placed in the columns in inverse order).

3. Calculate S^{-1} (mathematical operations on elements of the matrix are performed according to the rules of GF(p)).

4. Convert an element in a polynomial basis to a normal basis using the formula $\mathbf{x}_B = S^{-1} \cdot \mathbf{x}_A$.

If the matrix does not exist in the third step of the algorithm, then the irreducible polynomial used in the polynomial basis is not normal. Hence, it is impossible to construct a normal basis. In this case, it is necessary to select another irreducible polynomial in the given field and return to the first step of the algorithm.

The conversion algorithm from the normal basis to the polynomial will differ only in the last step, where instead of the formula $\mathbf{x}_B = S^{-1} \cdot \mathbf{x}_A$ is used – $\mathbf{x}_A = S \cdot \mathbf{x}_B$.

Consider an example of a conversion from a polynomial basis to a normal one. Let it be necessary to convert the element $(0;1;1)$ of $GF(2^3)$ with an irreducible polynomial $x^3 + x^2 + 1$ to the normal basis.

Elements of $GF(2^3)$ for an irreducible polynomial $x^3 + x^2 + 1$ in a polynomial basis are given in Table 2.

Table 2 – Elements of $GF(2^3)$ with an irreducible polynomial $x^3 + x^2 + 1$ in a polynomial basis

x^2	x^1	1	Exponent α
0	0	0	–
0	0	1	0
0	1	0	1
0	1	1	2
1	0	0	3
1	0	1	4
1	1	0	5
1	1	1	6

The basic elements for a polynomial basis are 1, t, t^2.

According to the considered algorithm:

1. Expressing the basic elements of a normal basis through the basic elements of a polynomial.

Elements of a normal basis have the form:

$$B = \left\{ \alpha_0 = t, \alpha_1 = t^2, \alpha_2 = t^{2^2} = t^4 \right\}.$$

For elements of a normal basis having a power greater than $m-1$, the values in a polynomial basis can be found in two ways.

1st way. As a remainder of dividing an element in a normal basis by an irreducible polynomial:

$$\begin{array}{l|l} t^4 + 0t^3 + 0t^2 + 0t + 0 & t^3 + t^2 + 1 \\ \hline t^4 + t^3 + 0t^2 + t & t+1 \\ \hline t^3 + 0t^2 + t + 0 & \\ \hline t^3 + t^2 + 0t + 1 & \\ \hline t^2 + t + 1 & \end{array}$$

2nd way. By means of substitutions (from an irreducible polynomial):

$$t^4 = t \cdot t^3 = t \cdot (t^2 + 1) = t^3 + t = t^2 + t + 1.$$

Then $t^4 = t^2 + t + 1$.

Hence, $B = \left\{ \alpha_0 = t, \alpha_1 = t^2, \alpha_2 = t^2 + t + 1 \right\}$. In a vector representation, the basic elements of a normal basis have the form: $(0;1;0)$, $(1;0;0)$, $(1;1;1)$.

2. Constructing the change-of-basis conversion matrix from the basic elements of the normal basis:

$$S = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$

3. Calculating the inverse matrix.

Any method to calculate the inverse matrix can be used, however, it is necessary to perform all operations on the matrix elements in $GF(p)$.

Hence, $S^{-1} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$.

4. Obtaining the desired element $(0;1;1)$ in the normal basis:

$$\mathbf{x}_B = S^{-1} \cdot \mathbf{x}_A = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}.$$

The representation of all elements of the polynomial basis in normal is calculated by the formula $\mathbf{x}_B = S^{-1} \cdot \mathbf{x}_A$ (Table 3).

Table 3 – Elements representation in polynomial and normal basis of $GF(2^3)$

Element in a polynomial basis			Inverse matrix (S^{-1})	Element in the normal basis		
t^2	t^1	1		$t^4 = t^2 + t + 1$	t^2	t^1
0	0	0	$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$	0	0	0
0	0	1		1	1	1
0	1	0		0	0	1
0	1	1		1	1	0
1	0	0		0	1	0
1	0	1		1	0	1
1	1	0		0	1	1
1	1	1		1	0	0

Let's conduct a comparative analysis of the methods of performing operations on elements of $GF(2^m)$ in the polynomial and normal basis.

The main operations on the elements of the Galois field are: addition, multiplication, calculating the multiplicative inverse element, division, exponentiation, Frobenius operation.

The simplest operation in the Galois field is the addition. It is performed in a polynomial and normal basis by the formula:

$$c = (a + b) \bmod p.$$

DOI 10.15588/1607-3274-2020-2-12

That is, as the addition of two polynomials *a* and *b*, where operations over coefficients are performed by modulo *p*.

The computational complexity of this operation $O(m)$.

When $p = 2$ (in the binary field), the operation of the addition is greatly simplified. In this case, it's enough to perform the XOR bitwise operation:

$$c = a \oplus b .$$

Similarly to the addition, the subtraction operation is performed in both bases with the same computational complexity $O(m)$, by the formula:

$$c = (a - b) \bmod p .$$

Accordingly, in a binary field it is also sufficient to perform a bitwise operation XOR:

$$c = a \oplus b .$$

In a polynomial basis, the multiplication operation in a finite field is a multiplication by the module of an irreducible polynomial, that is [9, 12]:

1. Perform the multiplication of elements in a polynomial basis by the formula $c = a \cdot b$.

2. Calculate the remainder of the division of the result by an irreducible polynomial.

Consider an example. Let in $GF(2^3)$ with an irreducible polynomial $f(x) = x^3 + x^2 + 1$ it is necessary to multiply two polynomials $\mathbf{a} = x + 1$ and $\mathbf{b} = x^2 + x$.

Let us represent polynomials in the form of a coefficient vector: $\mathbf{a} = (0;1;1)$, $\mathbf{b} = (1;1;0)$.

Then according to the algorithm:

1. Multiplying polynomials:

$$\mathbf{a} \cdot \mathbf{b} = (x+1) \cdot (x^2 + x) = x^3 + x^2 + x^2 + x = x^3 + x .$$

2. Calculating the remainder of the division by an irreducible polynomial:

$x^3 + 0x^2 + x + 0$	$x^3 + x^2 + 1$
$x^3 + x^2 + 0x + 1$	1
$x^2 + x + 1$	

Hence, $(a \cdot b) \bmod f(x) = x^2 + x + 1$.

The multiplication in a polynomial basis based on the above algorithm has a computational complexity $O(m^2)$. There are modified multiplication algorithms that have computational complexity $O(m \log m \log\log m)$ [14].

A multiplication operation in the normal basis is performed using an additional multiplication matrix *M*. The algorithm in this case can be divided by two stages: the calculation of the matrix *M* and the actual multiplication. The first stage is executed only once, while for others, the already found matrix is used because it depends only on the Galois field.

The multiplication algorithm of elements $\mathbf{a} = (a_{m-1}; a_{m-2}; \ldots; a_0)$ and $\mathbf{b} = (b_{m-1}; b_{m-2}; \ldots; b_0)$ of $GF(p^m)$ [9, 13] based on the features of the normal basis where the following formula is true: $t^{p^i} \cdot t = \sum_{k=0}^{m-1} \mu_{i;k} t^{p^k}$, so that $t^{p^i} \cdot t^{p^j} = \sum_{k=0}^{m-1} \mu_{i-j;k-j} t^{p^k}$.

The algorithm for obtaining the multiplication matrix *M* for $p = 2$ [9]:

1. Construct the matrix L_1 (elements of the normal basis, expressed through the basic elements of a polynomial basis, are placed in the columns in ascending order, that is $t, t^p, t^{p^2}, \ldots, t^{p^{m-1}}$).

2. Construct the matrix L_2, where $t^{p^i} \cdot t$ elements ($i = \overline{0 \ldots m-1}$) are placed in the columns in ascending order.

3. Construct the matrix L with the following block structure:

$$L = (L_1 \mid L_2) .$$

4. Obtain the identity matrix in the left part of the *L* matrix by performing matrix transformations:

$$L' = (I \mid M) .$$

5. The multiplication matrix *M* is obtained in the right part of the matrix L'.

Multiplication algorithm of elements $\mathbf{a} = (a_{m-1}; a_{m-2}; \ldots; a_0)$ and $\mathbf{b} = (b_{m-1}; b_{m-2}; \ldots; b_0)$ of $GF(2^m)$ [9, 13]:

1. Initialize the variables: $\mathbf{x} = \mathbf{a}$, $\mathbf{y} = \mathbf{b}$.

2. For $k = \overline{m-1 \ldots 0}$:

$$\text{calculate } c_k = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i \cdot b_j \cdot \mu_{i-j;k-j} .$$

3. The result of the multiplication is $\mathbf{c} = (c_{m-1}; c_{m-2}; \ldots; c_0)$.

The computational complexity of the multiplication operation in the normal basis in the general case reaches $O(m^3)$, while in individual it improves to $O(m^2)$ [14].

The calculation of a multiplicative inverse element in the general case in both bases is carried out in two ways: according to the extended Euclidean algorithm or by Euler's theorem [15].

The division operation is implemented by multiplying a dividend to the multiplicative inverse element of a divisor $\mathbf{a}/\mathbf{b} = \mathbf{a} \cdot \mathbf{b}^{-1}$. Given that the multiplication and calculation of the multiplicative inverse of the element can be performed by different methods, the computational complexity of the division operation will be different.

The exponentiation in the normal and polynomial bases can be performed according to the same algorithm [9]:

1. Convert the power *k* to the *p*-th numeral system:

$$k = k_r k_{r-1} \ldots k_1 k_0 .$$

DOI 10.15588/1607-3274-2020-2-12

2. Initialize the variable $\mathbf{x} = \boldsymbol{\alpha}$, where $\boldsymbol{\alpha}$ is the element of $GF(p^m)$, which is raised to k-th power.

3. For $i = \overline{r-1\ldots 0}$:

a. perform squaring $\mathbf{x} = \mathbf{x}^2$;

b. if $k_i = 1$, then $\mathbf{x} = \boldsymbol{\alpha} \cdot \mathbf{x}$.

4. The result of the operation – $\mathbf{x}$.

This operation has a different computational complexity depending on the basis used, since multiplication, depending on the representation of the elements performed in different ways. In this case, the rise to a square in a normal basis can be replaced by the Frobenius operation for $GF(p^m)$.

Let's consider in detail the Frobenius operation – raising of the field element to the power p^k. Time costs, depending on the choice of basis differ significantly enough.

In a normal basis, Frobenius operation is very simple, it does not require significant time consumption. Due to the representation of elements of this basis $B = \left\{\alpha_0 = t, \alpha_1 = t^p, \alpha_2 = t^{p^2}, \ldots, \alpha_{m-1} = t^{p^{m-1}}\right\}$, it is enough to perform a cyclic shift on k digits. Depending on the representation of the finite field elements in a normal basis, the shift must be performed in a different direction. If an element is applied as the power of the generating element increases $\mathbf{a} = \left(a_0; a_1; \ldots; a_{m-1}\right)$, then the cyclic shift should be executed to the right by k positions $a^{p^k} = a_{m-k} a_{m-k+1} \ldots a_{m-1} a_0 a_1 \ldots a_{m-k-1}$; otherwise, if $\mathbf{a} = \left(a_{m-1}; a_{m-2}; \ldots; a_0\right)$, the shift is executed to the left $a^{p^k} = a_{m-k-1} a_{m-k-2} \ldots a_1 a_0 a_{m-1} \ldots a_{m-k}$.

Consider an example of Frobenius operation in a normal basis. Let the element $\mathbf{a} = \left(a_2; a_1; a_0\right) = \left(0; 1; 1\right)$ of $GF(2^3)$ with an irreducible polynomial $x^3 + x^2 + 1$ need to be raised to a power $z = p^k = 2^2 = 4$. Then the result of the operation:

$$\mathbf{a}^{2^2} = \mathbf{a} << 2 = \left(a_0; a_2; a_1\right) = \left(1; 0; 1\right).$$

In a polynomial basis for Frobenius operation there is no fast algorithm, it is performed as exponentiation.

Based on the analysis of the algorithms for performing operations in both bases and the algorithms for the conversion between bases, the following conclusions can be defined:

1. The computational complexity of algorithms for performing operations depends on the chosen basis.

2. The operations of multiplication and calculation of a multiplicative inverse element are executed faster in a polynomial basis, and the Frobenius operation – in a normal.

3. The usage of different bases is advisable when performing certain types of tasks, in particular, due to change-of-basis conversion, it is possible to speed up the execution of cryptographic operations.

3 MATERIALS AND METHODS

The prerequisite for constructing a normal basis is choosing a normal polynomial in Galois field, so reducing the computational complexity of the problem of finding a normal polynomial is actual.

The proposed method for finding normal polynomials with m degree in $GF(2^m)$ consists of the following steps:

1. Find all prime numbers in the range $\left[2^m; 2^{m+1}\right)$ (for example, by the algorithm of the sieve of Eratosthenes).

2. Convert the resulting prime numbers to the binary numeral system. The result of this step is the polynomials given by its coefficients.

3. Remove from the set of polynomials obtained in step 2 of the algorithm, the polynomials with the trace, which is equal to 0. The result of this step will be irreducible polynomials.

4. Calculate the determinant of the change-of-basis conversion matrix between the bases for a set which contains irreducible polynomials found in step 3. Note: all operations on matrix elements should be performed in $GF(p)$.

5. If the determinant is not equal to zero, then the irreducible polynomial is normal.

It is worth noting that in step 3, not all irreducible polynomials are obtained, but the number of missing irreducible polynomials is insignificant in relation to their total number.

Consider an example that illustrates the proposed method for constructing normal polynomials for $GF(2^4)$:

1. Prime numbers in range $\left[2^4; 2^5\right)$: 17, 19, 23, 29, 31.

2. Converting numbers from a decimal numeral system to binary:

$$17_{10} = 10001_2,$$
$$19_{10} = 10011_2,$$
$$23_{10} = 10111_2,$$
$$29_{10} = 11101_2,$$
$$31_{10} = 11111_2.$$

3. Determining the trace for polynomials found in step 2 of the algorithm by the formula $Tr(\mathbf{x}) = \sum_{i=0}^{m-1} a_i \left(\mathrm{mod}\, 2\right)$:

$$Tr\left(10001\right) = 2 \bmod 2 = 0,$$
$$Tr\left(10011\right) = 3 \bmod 2 = 1,$$
$$Tr\left(10111\right) = 4 \bmod 2 = 0,$$
$$Tr\left(11101\right) = 4 \bmod 2 = 0,$$
$$Tr\left(11111\right) = 5 \bmod 2 = 1.$$

Polynomials 10011 and 11111 are irreducible.

4. Calculating the determinant of the change-of-basis conversion matrix for each irreducible polynomial from step 3.

DOI 10.15588/1607-3274-2020-2-12

For polynomial 10011: $\det\begin{pmatrix}0&0&0&0\\1&0&1&0\\0&1&0&1\\1&1&0&0\end{pmatrix}=0$.

For polynomial 11111: $\det\begin{pmatrix}1&1&0&0\\0&1&1&0\\0&1&0&1\\0&1&0&0\end{pmatrix}=1$.

Hence, normal polynomial – $x^4+x^3+x^2+x+1$.

In the considered example, the polynomial 11001, that is irreducible, is not present in the result set of irreducible polynomials, in comparison with the classic method of finding irreducible polynomials (by checking whether the remainder of the division of the current polynomial by all irreducible with lesser degrees is equal to 0), which finds the whole set of irreducible polynomials. This is because its binary representation does not correspond to the prime number in the decimal system.

Consider the proposed method for a field of greater cardinality, in particular for $GF(2^5)$:

1. Prime numbers in range $\left[2^5;2^6\right)$: 37, 41, 43, 47, 53, 59, 61.

2. Converting numbers from a decimal numeral system to binary:

$$37_{10}=100101_2,$$
$$41_{10}=101001_2,$$
$$43_{10}=101011_2,$$
$$47_{10}=101111_2,$$
$$53_{10}=110101_2,$$
$$59_{10}=111011_2,$$
$$61_{10}=111101_2.$$

3. Determining the trace for polynomials found in step 2 of the algorithm by the formula $Tr(\mathbf{x})=\sum_{i=0}^{m-1}a_i\left(\mathrm{mod}\,2\right)$:

$$Tr\left(100101\right)=3\bmod 2=1,$$
$$Tr\left(101001\right)=3\bmod 2=1,$$
$$Tr\left(101011\right)=4\bmod 2=0,$$
$$Tr\left(101111\right)=5\bmod 2=1,$$
$$Tr\left(110101\right)=4\bmod 2=0,$$
$$Tr\left(111011\right)=5\bmod 2=1,$$
$$Tr\left(111101\right)=5\bmod 2=1.$$

Hence, polynomials 100101, 101001, 101111, 111011, 111101 are irreducible.

4. Calculating the determinant of the conversion matrix for each irreducible polynomial from step 3.

For polynomial 100101: $\det\begin{pmatrix}1&0&1&0&0\\1&1&0&0&0\\0&1&0&1&0\\1&0&0&0&1\\1&1&0&0&0\end{pmatrix}=0$.

For polynomial 101001: $\det\begin{pmatrix}0&1&1&0&0\\1&1&0&0&0\\1&0&0&1&0\\0&1&0&0&1\\0&0&0&0&0\end{pmatrix}=0$.

For polynomial 101111: $\det\begin{pmatrix}1&0&1&0&0\\1&1&0&0&0\\1&0&0&1&0\\1&0&0&0&1\\1&1&0&0&0\end{pmatrix}=0$.

For polynomial 111011: $\det\begin{pmatrix}1&0&1&0&0\\1&1&0&0&0\\0&1&0&1&0\\0&1&0&0&1\\0&1&0&0&0\end{pmatrix}=1$.

For polynomial 111101: $\det\begin{pmatrix}0&1&1&0&0\\1&1&0&0&0\\0&1&0&1&0\\1&0&0&0&1\\1&0&0&0&0\end{pmatrix}=1$.

Hence, normal polynomials – $x^5+x^4+x^3+x+1$, $x^5+x^4+x^3+x^2+1$.

The computational complexity of a proposed method of finding normal polynomials is $O\left(n\log\left(\log n\right)\right)$, which is less than the complexity of the traditional method – $O\left(n^3\right)$.

The construction of the change-of-basis conversion matrix is based on the expression of the basic elements of a normal basis $B=\left\{\alpha_0=t,\alpha_1=t^p,\alpha_2=t^{p^2},\ldots,\alpha_{m-1}=t^{p^{m-1}}\right\}$ through a polynomial basis. This stage is performed in two ways: the division of an element t^{p^i} by an irreducible polynomial or substitutions from an irreducible polynomial. Let's consider these methods in more detail for $GF(2^5)$ with an irreducible polynomial $f\left(x\right)=x^5+x^4+x^3+x^2+1$.

For $p^i=2^i<m$ no additional calculations are required (elements are expressed by definition): $\alpha_0=x^{2^0}=x$, $\alpha_1=x^{2^1}=x^2$, $\alpha_2=x^{2^2}=x^4$. Other

DOI 10.15588/1607-3274-2020-2-12

elements should be calculated using the below approaches.

Approach 1. Dividing on an irreducible polynomial

$$\alpha_3 = x^{2^3} = x^8 \bmod f(x).$$

$$\begin{array}{l|l} x^8+0x^7+0x^6+0x^5+0x^4+0x^3+0x^2+0x+0 & x^5+x^4+x^3+x^2+1 \\ \hline x^8+x^7+x^6+x^5+0x^4+x^3 & x^3+x^2 \\ \quad x^7+x^6+x^5+0x^4+x^3+0x^2 & \\ \quad x^7+x^6+x^5+x^4+0x^3+x^2 & \\ \quad\quad x^4+x^3+x^2+0x+0 & \end{array}$$

Hence, $\alpha_3 = x^4 + x^3 + x^2$.

In the same way:

$$\alpha_4 = x^{2^4} = x^{16} \bmod f(x) = x^3 + x + 1.$$

In this approach, to calculate the remainder of the element's division by an irreducible polynomial, it is necessary to allocate memory that is proportional to the value $n \cdot p^{m-1}$, where n is the amount of memory required to save $GF(p)$ element, $n = \lceil \log_2 p \rceil$.

The computational complexity of a given algorithm for an i-th element is equal to $O(i \cdot m^p)$.

Approach 2. Substitutions from an irreducible polynomial

$$\begin{aligned}
\alpha_3 = x^{2^3} &= x^8 = x^3 \cdot x^5 = x^3 \cdot (x^4 + x^3 + x^2 + 1) = \\
&= x^7 + x^6 + x^5 + x^3 = (x^2 + x + 1) \cdot x^5 + x^3 = \\
&= (x^2 + x + 1) \cdot (x^4 + x^3 + x^2 + 1) + x^3 = \\
&= x^6 + x + x^4 + 1 + x^3 = x \cdot x^5 + x^4 + x^3 + x + 1 = \\
&= x \cdot (x^4 + x^3 + x^2 + 1) + x^4 + x^3 + x + 1 = \\
&= x^5 + 1 = x^4 + x^3 + x^2, \\
\alpha_4 = x^{2^4} &= x^{16} = x \cdot (x^5)^3 = x \cdot (x^4 + x^3 + x^2 + 1)^3 = \\
&= x \cdot \Big((x^4 + x^3)^3 + (x^4 + x^3)^2 \cdot (x^2 + 1) + \\
&+ (x^4 + x^3) \cdot (x^2 + 1)^2 + (x^2 + 1)^3 \Big) = \\
&= x \cdot \Big((x^{12} + x^{11} + x^{10} + x^9) + (x^{10} + x^6) + \\
&+ (x^8 + x^7 + x^4 + x^3) + (x^6 + x^4 + x^2 + 1) \Big) = \\
&= x \cdot (x^{12} + x^{11} + x^9 + x^8 + x^7 + x^3 + x^2 + 1) = \ldots = \\
&= x^3 + x + 1.
\end{aligned}$$

This approach is difficult for software implementation, since as the parameter m increases, the number of additional calculations will increase substantially.

Modified approach. An element of a normal basis $t^{p^{i+1}}$ can be calculated using a recursive formula $\alpha_{i+1} = t^{p^{i+1}} = t^{p^i \cdot p} = \prod_{j=1}^{p} t^{p^i} = (\alpha_i)^p$ this will reduce the amount of allocated memory and speed up the execution of the algorithm. In this method, squaring is performed like usual multiplication and then the result of this operation is divided by an irreducible polynomial.

Consider an example:

$$\begin{aligned}
\alpha_3 = x^{2^3} &= \alpha_2 \cdot \alpha_2 = x^4 \cdot x^4 = \\
&= x^8 \bmod f(x) = x^4 + x^3 + x^2, \\
\alpha_4 = x^{2^4} &= \alpha_3{}^2 = (x^4 + x^3 + x^2)^2 = \\
&= (x^8 + x^6 + x^4) \bmod f(x),
\end{aligned}$$

$$\begin{array}{l|l} x^8+0x^7+x^6+0x^5+x^4+0x^3+0x^2+0x+0 & x^5+x^4+x^3+x^2+1 \\ \hline x^8+x^7+x^6+x^5+0x^4+x^3 & x^3+x^2+x+1 \\ \quad x^7+0x^6+x^5+x^4+x^3+0x^2 & \\ \quad x^7+x^6+x^5+x^4+0x^3+x^2 & \\ \quad\quad x^6+0x^5+0x^4+x^3+x^2+0x & \\ \quad\quad x^6+x^5+x^4+x^3+0x^2+x & \\ \quad\quad\quad x^5+x^4+0x^3+x^2+x+0 & \\ \quad\quad\quad x^5+x^4+x^3+x^2+0x+1 & \\ \quad\quad\quad\quad x^3+0x^2+x+1 & \end{array}$$

Hence, $\alpha_4 = x^3 + x + 1$.

To calculate all elements of the change-of-basis conversion matrix using this approach, it is necessary to allocate memory proportional $n \cdot p$, where n is the amount of memory needed to save $GF(p)$ element. This is because it is this maximum number of coefficients that will be the result of exponentiation of the previous basic element of the field t^{p^i} to the power p. Since in the expression of elements of a normal basis through the elements of the polynomial by the indicated method, it will be necessary to calculate the remainder from the division of an element $t^{p^{i+1}}$, whose maximum length does not exceed $2m-1$, then the computational complexity of the implementation of this approach is equal $O(m^p)$.

4 EXPERIMENTS

In order to conduct experimental research, a software was developed. The development has been done using the C# programming language in the Visual Studio 2015 development environment.

The developed system has a modular architecture (the block diagram is shown in Fig. 1), which consists of the following components: implementation of change-of-basis conversion; performing calculations in a polynomial

DOI 10.15588/1607-3274-2020-2-12

basis; performing calculations in a normal basis; carrying out time measurements of execution of algorithms; generation of test data; finding normal polynomials; interaction with the user.

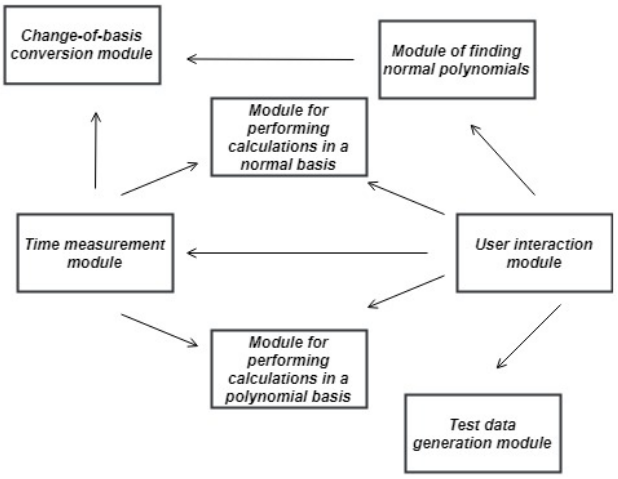

Figure 1 – Structural scheme of the developed software

Experimental research was conducted on the Acer ES1-311 personal computer, which has Windows 7 operating system with the following characteristics: CPU 1.83 GHz, Intel Celeron N2940, 4GB of RAM.

The investigation of the speed of performing addition, multiplication, calculating the multiplicative inverse element and division operations was conducted using the following input: $GF(2^{24})$, normal polynomial $x^{24}+x^{23}+x^8+x^5+x^4+x+1$, number of generated elements – 100000.

Experimental investigations of the performing Frobenius operation in different bases were conducted using the above input data with an additional parameter *k*, which took values from 1 to 5, 7, from 9 to 13.

Experimental investigations of the exponentiation operation were performed on the same input data as the time measurements for basic operations, with an additional parameter *k*, which took values from 1 to 8, and from 10 to 13.

5 RESULTS

The execution time of addition, multiplication, calculating the multiplicative inverse element and division operations are shown in Table 4.

Table 4 – Execution time of basic operations in $GF(2^{24})$, ms

Basis	Operation			
	Addition	Multiplication	Division	Calculation of the multiplicative inverse element
Normal	0.0020	1.2188	10.3941	9.2621
Polynomial	0.0019	0.1092	3.4224	3.2990

As can be seen from the experimental results, the addition is an operation independent of the basis representation of the element and is executed at the same time. The multiplication operation is performed 10 times faster in a polynomial basis than in the normal basis. The calculation of a multiplicative inverse element takes 2.5 times more time for a normal basis, and, as a result, the division operation is performed 3 times faster in a polynomial basis.

Summing up the experimental results, which are given in Table 5, the Frobenius operation in the normal basis is performed at a constant time, which is 47 times smaller than the time of this operation in the polynomial basis for $k=1$. It is worth noting that the execution time in a polynomial basis varies linearly with respect to the parameter *k*.

Table 5 – Execution time of Frobenius operation in $GF(2^{24})$ for different *k*. ms

k	Basis	
	Normal	Polynomial
1	0.002436	0.114450
2	0.001679	0.225372
3	0.000997	0.339942
4	0.000994	0.451851
5	0.000992	0.557715
7	0.000982	0.777917
9	0.000972	0.989613
10	0.000990	1.095348
11	0.000996	1.205277
12	0.000999	1.310586
13	0.001014	1.417930

The graph of the dependence of the execution time on the parameter *k* for the exponentiation operation is shown in Fig. 2.

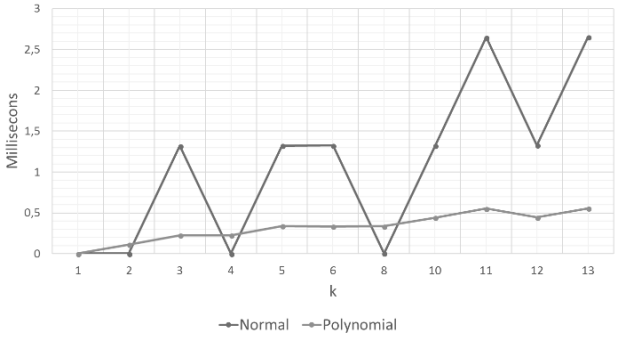

Figure 2 – Execution time of the exponentiation operation in $GF(2^{24})$

Experimental results of the method of finding normal polynomials for the parameter $p = 2$ is given in Table 6.

Table 6 – Execution time of finding normal polynomials, ms

m	Method	
	Classic	Proposed
8	8.24601	10.81466
10	47.79605	31.92607
14	18103.12345	315.03410
16	243103.12345	1514.77778

DOI 10.15588/1607-3274-2020-2-12

Experimental results of the construction a change-of-basis conversion matrix are given in Table 7.

Table 7 – Execution time of construction a change-of-basis conversion matrix

m	Irreducible polynomial	Time of construction, ms	
		Approach 1. (dividing on an irreducible polynomial)	Modified approach
8	$x^8+x^7+x^2+x+1$	0.0877433	0.1332861
10	$x^{10}+x^9+x^4+x+1$	0.1923009	0.2208144
14	$x^{14}+x^{13}+x^3+x^2+1$	2.0167270	0.4689929
16	$x^{16}+x^{15}+x^4+x+1$	8.7543370	0.6960259
18	$x^{18}+x^{17}+x^5+x+1$	37.6739521	0.8904218
20	$x^{20}+x^3+1$	164.2269162	1.1983284

As can be seen from the experimental results, the proposed approach gives worse time results for fields with a parameter $m<10$. However, for fields with a parameter $m>12$ the increase in speed is significant. So, in $GF(2^{14})$ the construction time of the conversion matrix is less than 5 times; in $GF(2^{16})$ – at 11, and for $m=18$–40 times.

6 DISCUSSION

The software architecture is developed, the feature of which is the encapsulation of the element basis of the finite field, which allows, using "Strategy" template, to replace a particular implementation in an instance of the class of an element of the Galois field without the use of inheritance.

The approach of constructing a change-of-basis conversion matrix, which gives an increase in the speed for the parameter $m>12$ in more than 5 times, is proposed. So, for $m=16$ the time of construction of the conversion matrix, it is 11 times smaller than the execution time using the standard approach (dividing the element by an irreducible polynomial).

The method of finding normal polynomials for change-of-basis conversion of binary finite fields is proposed, which gives an increase in speed over 15 times for a parameter $m\geq 14$. So, for $m=16$ the time of finding normal polynomials, there are more than 150 times less. It should be noted that the set of normal polynomials that can be found by the proposed method is less than by the classic method.

CONCLUSIONS

A polynomial and a normal basis can be used to perform operations on elements of the extended Galois field. Depending on the chosen basis, the execution time of operations is different, in particular, in the polynomial basis, the multiplication operation is performed 4 times faster, and the calculating of a multiplicative inverse element is faster in 2 times. At the same time, in the normal basis Frobenius operation is performed at the constant time regardless of the parameter k, with $k=1$ the time for execution less than 10 times, compared with the polynomial basis.

The scientific novelty of the obtained results is that:

1. The method of finding normal polynomials is proposed, which differs from the existing by using of prime numbers in a decimal representation instead of polynomials, which reduces the computational cost of the algorithm for finding normal polynomials from $O\left(n^3\right)$ to $O\left(n\log\left(\log n\right)\right)$.

2. A modified approach is proposed for constructing a change-of-basis conversion matrix, which consists of using a recursive formula $\alpha_{i+1}=t^{p^{i+1}}=\left(\alpha_i\right)^p$ instead of computing the remainder of the element's $t^{p^{i+1}}$ division by an irreducible polynomial, which reduces the amount of memory used from to $n\cdot p$, and also the computational complexity from $O\left(i\cdot m^p\right)$ to $O\left(m^p\right)$. An increase in speed more than 5 times is observed using this approach for the parameter $m>12$. For example, for $m=16$ the construction time of the conversion matrix, it is 11 times smaller than the execution time using the standard approach (dividing an element by an irreducible polynomial).

The practical significance of the obtained results is that the software has been developed for experimental research. It allows using polynomial and normal representation of $GF(p^m)$, setting different input parameters p and m, and obtaining different sets of test data depending on the chosen irreducible polynomial of Galois field.

Thus, the obtained results can be used to solve the practical problems of elliptic cryptography and algorithms for correction of data transmission errors.

The prospect for further research is developing modified methods for the implementation of multiplicative operations (multiplication, division, calculation of multiplicative inverse element) on elements of the finite field, which are represented in a normal basis.

REFERENCES

1. Lidl R., Niederreiter H. Finite Fields. Cambridge: Cambridge University Press, 1996, 755 p. DOI: 10.1017/CBO9780511525926.
2. Benvenuto C. J. Galois field in cryptography, University of Washington, 2012.
3. Advanced Encryption Standard (AES), *Federal Information Processing Standards*, 2001, DOI: 10.6028/NIST.FIPS.197.
4. Oliynykov R., Gorbenko I., Kazymyrov O. et. al. A New Encryption Standard of Ukraine: The Kalyna Block Cipher, *IACR Cryptology ePrint Archive*, 2015, Vol. 2015, №650.
5. Bolotov A. A., Gashkov S. B., Frolov A. B., Chasovskih A. B. Algoritmicheskie osnovy ellipticheskoj kriptografii. Moscow, Izd-vo RSGU, 2004, 499 p.
6. Bolotov, A. A., Gashkov S. B., Frolov A. B., Chasovskih A. A. Elementarnoe vvedenie v ellipticheskuyu kriptografiyu: algebraicheskie i algoritmicheskie osnovy [Text]. Moscow, KomKniga, 2006, 328 p. ISBN 5-484-00443-8.

DOI 10.15588/1607-3274-2020-2-12

7. Shrivastava P., Singh U. P. Error Detection and Correction Using Reed Solomon Codes, *International Journal of Advanced Research in Computer Science and Software Engineering*, 2013, Vol. 3, № 8, pp. 965–969 ISSN: 2277-128X.
8. Westall J., Martin J. An Introduction to Galois Fields and Reed-Solomon Coding, *School of Computing Clemson University Clemson*, SC 29634-1906, 2010.
9. Cohen H., Frey G., Avanzi R. et. al. Handbook of Elliptic and Hyperelliptic Curve Cryptography, 2005, 842 p. (Discrete Mathematics and Its Applications) ISBN 1-58488-518-1.
10. Algebraic structures [Electronic resource]. Access mode: http://faculty.bard.edu/belk/math332/AlgebraicStructures.pdf.
11. Gao S. Normal Bases over Finite Fields, University of Waterloo, 1993.
12. Gashkov S. B., Sergeev I. S. Complexity of computation in finite fields, *Journal of Mathematical Sciences*. – 2013. – Vol. 191, P. 661–685 DOI: 10.1007/s10958-013-1350-5.
13. Bolotov A. A., Gashkov S. B. On a quick multiplication in normal bases of finite fields, *Discrete Mathematics and Applications,* 2001, Vol. 11, №4, pp. 327–356 DOI 10.1515/DMA.2001.
14. Zindros D. A Gentle Introduction to Algorithm Complexity Analysis [Electronic resource]. Access mode: https://discrete.gr/complexity/.
15. Lassak M., Porubsky S. Fermat-Euler Theorem in Algebraic Number Fields, *Journal of number theory*, 1996. №60, pp. 254–290.

Received 25.02.2020.
Accepted 01.04.2020.

УДК 004.021

МОДИФІКОВАНИЙ МЕТОД МІЖБАЗИСНИХ ПЕРЕТВОРЕНЬ У ПОЛІ $GF(2^m)$

Дичка І. А. – д-р техн. наук, професор, декан факультету прикладної математики Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна.

Легеза В. П. – д-р техн. наук, професор кафедри програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна.

Онай М. В. – канд. техн. наук, доцент кафедри програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна.

Северін А. І. – студент кафедри програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна.

АННОТАЦІЯ

Актуальність. При реалізації криптографічних додатків та систем контролю передачі даних виникає потреба у швидких методах виконання операцій над елементами скінченних полів. Об'єктом дослідження є процеси шифрування, дешифрування та передачі інформації з використанням полів Галуа. Предметом дослідження є методи та алгоритми виконання обчислень у полях Галуа в поліноміальному й нормальному базисах.

Мета роботи. Метою даного дослідження є аналіз методів виконання операцій у полі Галуа залежно від обраного базису (поліноміальний, нормальний) та модифікація методу перетворення елементів з поліноміального базису у нормальний і навпаки, а також розроблення нового методу генерування нормальних поліномів для поліпшення часових характеристик.

Метод. У даній статті виконано порівняльний аналіз процесів виконання основних операцій у поліноміальному і нормальному базисах (додавання, множення, обчислення мультиплікативно оберненого елемента, ділення, піднесення до степеня, операція Фробеніуса), а також розглянуто та проаналізовано процес переходу з одного базису в інший. Досліджено способи переходу між базисами залежно від різних вхідних даних, зокрема, параметрів p та m поля. Запропоновано метод пошуку нормальних поліномів серед незвідних та модифікований підхід для побудови матриці переходу між базисами.

Результати. Існуючі та запропоновані алгоритми реалізовано мовою програмування C# у середовищі розробки Visual Studio 2015. Для проведення експериментальних досліджень розроблено програмну систему, яка дозволяє виконувати обчислення, використовуючи поліноміальне й нормальне представлення елементів поля $GF(p^m)$, задавати різні вхідні параметри p та m, а також отримувати різні множини тестових даних залежно від нормальних поліномів поля Галуа.

Висновки. Отримані експериментальні результати роботи методів та алгоритмів виконання операцій над елементами поля $GF(2^m)$ у заданих базисах показали, що запропонований метод пошуку нормальних поліномів для міжбазисних перетворень бінарних полів дає приріст швидкодії у понад 15 разів для параметра $m > 14$; запропонований підхід побудови матриці переходу дає приріст швидкодії у понад 5 разів для параметра $m > 12$.

КЛЮЧОВІ СЛОВА: скінченне поле, поле Галуа, поліноміальний базис, нормальний базис, незвідний поліном, нормальний поліном.

УДК 004.021

МОДИФИЦИРОВАННЫЙ МЕТОД МЕЖБАЗИСНЫХ ПРЕОБРАЗОВАНИЙ В ПОЛЕ $GF(2^m)$

Дичка И. А. – д-р техн. наук, профессор, декан факультета прикладной математики Национального технического университета Украины «Киевский политехнический институт имени Игоря Сикорского», Киев, Украина.

Легеза В. П. – д-р техн. наук, профессор кафедры программного обеспечения компьютерных систем Национального технического университета Украины «Киевский политехнический институт имени Игоря Сикорского», Киев, Украина.

Онай Н. В. – канд. техн. наук, доцент кафедры программного обеспечения компьютерных систем Национального технического университета Украины «Киевский политехнический институт имени Игоря Сикорского», Киев, Украина.

DOI 10.15588/1607-3274-2020-2-12

Северин А. И. – студент кафедры программного обеспечения компьютерных систем Национального технического университета Украины «Киевский политехнический институт имени Игоря Сикорского», Киев, Украина.

АННОТАЦИЯ

Актуальность. При реализации криптографических приложений и систем контроля передачи данных возникает потребность в быстрых методах выполнения операций над элементами конечных полей. Объектом исследования являются процессы шифрования, дешифрования и передачи информации с использованием полей Галуа. Предметом исследования являются методы и алгоритмы выполнения вычислений в полях Галуа в полиномиальном и нормальном базисах.

Цель работы. Целью данного исследования является анализ методов выполнения операций в поле Галуа в зависимости от выбранного базиса (полиномиальный, нормальный) и модификация метода преобразования элементов с полиномиального базиса в нормальный и наоборот, а также разработка нового метода генерирования нормальных полиномов для улучшения временных характеристик.

Метод. В данной статье выполнен сравнительный анализ процессов выполнения основных операций в полиномиальном и нормальном базисах (сложение, умножение, вычисление мультипликативно обратного элемента, деление, возведение в степень, операция Фробениуса), а также рассмотрено и проанализировано процесс перехода с одного базиса в другой. Исследованы способы перехода между базисами в зависимости от различных входных данных, в частности, параметров p и m поля. Предложен метод поиска нормальных полиномов среди неприводимых и модифицированный подход для построения матрицы перехода между базисами.

Результаты. Существующие и предложенные алгоритмы реализованы на языке программирования C# в среде разработки Visual Studio 2015. Для проведения экспериментальных исследований разработано программную систему, которая позволяет выполнять вычисления, используя полиномиальное и нормальное представление элементов поля $GF(p^m)$, задавать различные входные параметры p и m, а также получать различные множества тестовых данных в зависимости от нормальных полиномов поля Галуа.

Выводы. Полученные экспериментальные результаты работы методов и алгоритмов выполнения операций над элементами поля $GF(2^m)$ в заданных базисах показали, что предложенный метод поиска нормальных полиномов для межбазисных преобразований бинарных полей дает прирост быстродействия более чем в 15 раз для параметра $m > 14$; предложенный подход построения матрицы перехода дает прирост быстродействия более чем в 5 раз для параметра $m > 12$.

КЛЮЧЕВЫЕ СЛОВА: конечное поле, поле Галуа, полиномиальный базис, нормальный базис, неприводимый полином, нормальный полином.

ЛІТЕРАТУРА / ЛИТЕРАТУРА

1. Lidl R. Finite Fields / R. Lidl, H. Niederreiter. – Cambridge : Cambridge University Press, 1996. – 755 p. DOI: 10.1017/CBO9780511525926.
2. Benvenuto C. J. Galois field in cryptography / Christoforus Juan Benvenuto. // University of Washington. – 2012.
3. Advanced Encryption Standard (AES). // Federal Information Processing Standards. – 2001. – DOI: 10.6028/NIST.FIPS.197.
4. A New Encryption Standard of Ukraine: The Kalyna Block Cipher / [R. Oliynykov, I. Gorbenko, O. Kazymyrov et. al.]. // IACR Cryptology ePrint Archive. – 2015. – Vol. 2015, №650.
5. Болотов А. А. Алгоритмические основы эллиптической криптографии [Text] / А. А. Болотов, С. Б. Гашков, А. Б. Фролов, А.Б. Часовских. – М. : Изд-во РСГУ, 2004. – 499 с.
6. Болотов А. А. Элементарное введение в эллиптическую криптографию: алгебраические и алгоритмические основы [Text] / А. А. Болотов, С. Б. Гашков, А. Б. Фролов, А. А. Часовских. – Москва : КомКнига, 2006. – 328 с. ISBN 5-484-00443-8.
7. Shrivastava P. Error Detection and Correction Using Reed Solomon Codes / P. Shrivastava, U. P. Singh. // International Journal of Advanced Research in Computer Science and Software Engineering. – 2013. – Vol. 3, № 8. – P. 965–969 ISSN: 2277-128X.
8. Westall, J. An Introduction to Galois Fields and Reed-Solomon Coding / J. Westall, J. Martin // School of Computing Clemson University Clemson, SC 29634–1906. – 2010.
9. Handbook of Elliptic and Hyperelliptic Curve Cryptography / [H. Cohen, G. Frey, R. Avanzi et. al.], 2005. – 842 p. – (Discrete Mathematics and Its Applications) – ISBN 1-58488-518-1.
10. Algebraic structures [Electronic resource]. – Access mode: http://faculty.bard.edu/belk/math332/AlgebraicStructures.pdf.
11. Gao S. Normal Bases over Finite Fields / Shuhong Gao // University of Waterloo. – 1993.
12. Gashkov S. B. Complexity of computation in finite fields / S. B. Gashkov, I. S. Sergeev // Journal of Mathematical Sciences. – 2013. – Vol. 191. – P. 661–685 DOI: 10.1007/s10958-013-1350-5.
13. Bolotov A. A. On a quick multiplication in normal bases of finite fields / A. A. Bolotov, S. B. Gashkov. // Discrete Mathematics and Applications. – 2001. – Vol. 11, № 4. – P. 327–356 DOI 10.1515/DMA.2001.
14. Zindros D. A Gentle Introduction to Algorithm Complexity Analysis [Electronic resource] / Dionysis Zindros. – Access mode: https://discrete.gr/complexity/.
15. Lassak M. Fermat-Euler Theorem in Algebraic Number Fields / M. Lassak, S. Porubsky // Journal of number theory. – 1996. – № 60. – P. 254–290.

DOI 10.15588/1607-3274-2020-2-12