UDC 004.056.5

ORGANIZATION OF FAST EXPONENTIATION ON GALOIS FIELDS FOR CRYPTOGRAPHIC DATA PROTECTION SYSTEMS

Al-Mrayt Ghassan Abdel Jalil Halil, O. Markovskyi, A. Stupak

The article proposes the organization of accelerated execution of the basic operation of a wide range of cryptographic algorithms with a public key - exponentiation on finite Galois fields GF(2^n). Acceleration of the computational implementation of this operation is achieved by organizing the processing of several bits of the code at once during squaring on Galois fields. This organization is based on the use of polynomial squared properties, Montgomery group reduction, and extensive use of previous calculations. Procedures for performing basic operations of exponentiation on Galois fields are developed in detail, the work of which is illustrated by numerical examples. It has been proved that the proposed organization can increase the computational speed of this operation by 2.4 times, which is significant for cryptographic applications.

Key words*: multiplication operation on Galois fields, cryptographic algorithms based on Galois Fields algebra, Galois Fields exponentiation, Montgomery reduction.*

Introduction

The algebra of finite Galois fields, whose fundamentals were developed in the first half of the 19th century, only gained widespread use in information technology at the beginning of the 21st century. Currently, the mathematical principles of this algebra are the basis for many of the most advanced modern technologies, including mobile communication, high-speed data transmission, mechanisms for restoring lost data, cryptographic data protection, and information security [1]. One of the most significant properties of Galois fields is that regardless of the choice of the generating polynomial, it is feasible to generate a set of algebraic bases whose results will be different [2]. Using this property, it was possible to implement the concept of mathematically distributing communications carried out on the same carrier frequency. The implementation of such a concept in mobile communication systems makes it possible to hold thousands of conversations simultaneously while ensuring their reliable separation. This property is the basis for the application of Galois finite field algebra in modern cryptographic data protection mechanisms. In particular, the algebraic properties of Galois fields are the basis for the implementation of nonlinear transformations in the AES algorithm, which is widely used in practice [3]. A number of protocols for asymmetric encryption, identification, and digital signature with a public key [4], and schemes for cryptographically strong identification of remote users, are based on the Galois field algebra.

It is widely known that the effectiveness of cryptographic data protection mechanisms is determined by the level of security achieved by their use. In addition, it is determined by the speed at which their computation is performed. The last criterion is critical for cryptographic algorithms with a public key, the main computing operation of which is exponentiation performed on huge numbers. When using traditional algebra, this basic operation has the form of modular exponentiation. In Galois field algebra, the result of exponentiation is reduced to the field formed by the fundamental polynomial. The computational complexity of exponentiating n-bit numbers is O(n^3) [5]. This means that with a doubling of the bit depth, the amount of computation increases by a factor of 8. In Galois field algebra, this operation is much faster due to the fact that each bit of numbers is processed independently. In modern conditions, when within the framework of cloud technologies, cybercriminals have remote access to high-powered computer systems, there is an objective need to improve the level of security of cryptographic tools. The only way to enhance protection is to increase the number of bits used. And this dramatically slows down the computational implementation of cryptographic protocols. One of the possible ways out of this situation may be to expand the use of the Galois field algebra and search for ways to speed up the exponentiation of multidigit numbers.

Therefore, the scientific problem of accelerating the computing implementation of the exponentiation operation on Galois fields, which is fundamental to cryptographic applications, is of current relevance to the current stage of development of information and computer technologies.

Problem statement and review of methods for its solution

The expansion of the use of Galois field algebra in modern cryptographic information security protocols, as well as the potential for achieving a higher speed of exponentiation compared to traditional algebra, has led to intensive study of the problem of efficient computational implementation of basic operations in this algebra using hardware and software [6].

When using the Galois field algebra, for each number $A=a_{n-1}\cdot 2^{n-1}+a_{n-2}\cdot 2^{n-2}+\ldots+a_1\cdot 2+a_0$, $\forall j\in\{0, 1, \ldots, n-1\}$: $a_j\in\{0, 1\}$ can be associated with the polynomial $A(x)= a_{n-1}\cdot x^{n-1}+a_{n-2}\cdot x^{n-2}+\ldots+a_1\cdot x+a_0$.

The addition operation on Galois fields is reduced to performing XOR and is further denoted by the symbol '⊕'. Reduction, or finding the remainder from the polynomial division A(x) by the Galois field polynomial P(x), is further denoted as A rem P to distinguish the operation of finding the remainder from dividing the number A by the number M in ordinary algebra: A mod M. Multiplication operation on the Galois fields A⊗B rem P, consists of two operations: polynomial multiplication, denoted by the symbol '⊗', and reduction of the polynomial product with respect to the generating polynomial of the field P. The operation of squaring the number A on the Galois field with the generating polynomial P is denoted as A⊗A rem P or $A|^2$ rem P. Accordingly, the operation of exponentiation on Galois fields, that is, the calculation of the remainder of the polynomial division of the result of raising the number A to the power of E by the polynomial P, is denoted as $A|^E$ rem P.

The existing technologies of exponentiation, both in traditional algebra and on Galois fields, are based on the classical algorithm that provides for the sequential analysis of the bits of the exponent code $E = \{e_{n-1}, e_{n-2}, \ldots, e_0\}$, $\forall j\in\{0,1, \ldots, n-1\}$; $e_i\in\{0, 1\}$. Each step performs a squaring operation on the Galois field and a multiplication operation on the field, depending on the current value of the exponent bit. As each step uses the results of the previous one, the algorithm cannot be parallelized at the bit level of the exponent code.

Currently, there are two versions of this algorithm, which differ in the direction the bits in the exponent code are analyzed. When exponentiating from the high-order digits of the exponent code, at each of n steps, the current result (which is initially equal to one) is squared and multiplied by A if the current bit of the exponent code is equal to one. Correspondingly, the average time t0 of exponentiation from the most significant bits is equal to $1.5\cdot n\cdot t_m$, where t_m is the multiplication time on the Galois fields. As a result of exponentiation from the least significant digits of the exponent, partial parallelization of calculations within a single step is possible. This makes it possible to speed up calculations by a factor of 1.5 [7].

It can be concluded from the above discussion that there is no way to accelerate exponentiation on Galois fields at the level of classical algorithms. This means that speeding up the operation of exponentiation on Galois fields can be achieved by reducing the time of performing the most multiplicative operations on Galois fields: multiplication and squaring [8].

Generally, these operations are divided into two phases: polynomial multiplication (polynomial squaring) and reduction, which involves finding the remainder of the polynomial division of the result of the first phase using the forming polynomial $P(x)$ of the Galois field. The operation of polynomial multiplication of n-bit numbers requires $0.5\cdot n$ logical addition operations and n shift operations and n bit value testing operations to calculate the product. Taking into account that the execution time of the logical addition command is approximately the same as the execution time of the shift command, it can be assumed that the implementation of polynomial multiplication is determined by the execution time of $2.5\cdot n$ logical operations.

During polynomial reduction, the number corresponding to the generating polynomial is added to the current remainder. This operation includes determining the position of the most significant digit of the current remainder, shifting the code of the forming polynomial, logically adding it to the current remainder. Thus, to perform the reduction, it is necessary to perform an average of n bit test operations, $2\cdot n$ shift operations (shifting the code of the generating polynomial and the test code containing one unit), as well as $0.5\cdot n$ logical addition operations. In general, the number of logical operations for performing reduction by dividing polynomials is $3.5\cdot n$. Thus, the total number of

logical operations required to implement the multiplication of n–bit numbers on the Galois fields formed by the polynomial $P(x)$ of degree n is $6 \cdot n$ [8]

The operation of polynomial multiplication is reduced to the logical addition of a maximum of n appropriately shifted multiplicand codes. In theory, the minimum time for this operation is determined by the number of $\log_2 n$ operations of logical addition. Considering the fact that in real applications the value of n is several thousand, the specified approach to accelerating polynomial multiplication can be applied only within the framework of hardware implementations [9].

Almost all researchers consider the reduction operation as the primary source of acceleration for multiplication on Galois fields. This means that further reduction in the time for multiplication is achieved by speeding up the reduction operation. Most of the known methods [10-13] are based on the use of previous calculation depending on the constant polynomial P(x), which in cryptographic information protection systems is part of the public key and, accordingly, rarely changes.

In acceleration methods based on the use of this property of the generating polynomial, the remainders from the division of codes 2^{n+1}, …, $2^{2 \cdot n}$ by the generating polynomial $P(x)$ are pre-calculated: $P(x) : Q_1 = 2^{n+1}$ rem P, $Q_2 = 2^{n+2}$ rem P, …, $Q_n = 2^{2 \cdot n}$ rem P. The calculated codes are stored in the tabular memory of precalculations. The reduction is reduced to the addition of tabular codes that correlate with the units in the higher n digits of the code of the polynomial product. For this, it is necessary to perform an analysis of the higher n digits of the code of the polynomial product, which requires $2 \cdot n$ logical operations (n operations of testing the value of the bit and n operations of shifting the test code). Another $0.5 \cdot n$ operations are required, on the whole, to add the results of recalculations. Thus, due to the use of previous calculations, it is possible to reduce the average number of logical operations to implement the reduction to $2.5 \cdot n$. At the same time, the total average number of logical operations for multiplication on Galois fields is $5 \cdot n$.

There is another method of speeding up multiplication on Galois fields by combining both phases: polynomial multiplication and reduction using Montgomery technology [14]. In [15], a modification of the Montgomery technology, known in traditional algebra, to the peculiarities of the algebra of Galois fields is proposed. As a result of modifying Montgomery technology for the specifics of Galois fields, the number of logical operations for computing multiplication on Galois fields was reduced to $4.5 \cdot n$.

Purpose and objectives of research

In the current research, the objective is to accelerate the execution of the exponentiation operation on Galois fields, which is essential to the operation of cryptographic protocols. This will be accomplished through the application of precomputation, which facilitates the simultaneous execution of several operations.

In order to accomplish the set goal, the following scientific problems are solved:

- study of the specific properties of the squaring operation on Galois fields, which allow the execution time of several operations to be combined by using the results of previous calculations;
- development, on the basis of the specified specific properties, of the method of accelerated elevation to the square on Galois fields, which, due to the use of previous calculations, allows to combine the operation of adding a multiple and correcting the intermediate result, as well as to combine the processing of several adjacent digits of the multiplier in time;
- analyzing the performance of the developed organization of fast exponentiation on finite Galois fields and comparing it with other known methods designed to accelerate the calculation of exponents;
- study of the proposed organization of fast exponentiation on Galois fields based on software modeling.

Accelerated squaring method on Galois fields with Montgomery group reduction.

The main amount of calculations in exponentiation on Galois fields falls on the operation of squaring. As the main reserves for reducing the number of logical operations when squaring on Galois fields, we can consider:

- use of the property of a polynomial square;

- application of Montgomery reduction modified for Galois fields;
- group processing of discharges when performing the Montgomery reduction.

The basic property of a polynomial square that can be used to speed up calculations is that the polynomial square A⊗A of a binary number $A = a_{n-1}\cdot 2^{n-1} + a_{n-2}\cdot 2^{n-2} + \ldots + a_2\cdot 2^2 + a_1\cdot 2 + a_0$, де $\forall i \in \{0,1,\ldots,n\text{-}1\}$: $a_i \in \{0,1\}$ is equal to the number $A\otimes A = a_{n-1}\cdot 2^{2\cdot(n-1)} + a_{n-2}\cdot 2^{2\cdot(n-2)} + \ldots + a_2\cdot 2^{2\cdot 2} + a_1\cdot 2 + a_0$ [6] This means that polynomial squaring is reduced to inserting zeros between the binary digits of the number A. For example, if A= 14 = 11102, then A⊗A = 10101002 = 84.

It follows from the above that performing polynomial squaring comes down to shifts in software implementation and permutation of bits in hardware implementation. This means that when using the Montgomery reduction modified for the Galois field, the algorithm for squaring the number A reduces to the following sequence of actions:

1. The cycle counter j is set to zero: j=0, as well as the (n+1)-bit result code R: R=0.
2. Shift R is performed: R>>=1. If the value of j is even, j mod 2 =0, then the most significant digit of r_n is filled with the value of the least significant digit a_0 of the number A: $r_n = a_0$. Shift A: A>>=1. If the value of j is odd, then the most significant bit of r_n is filled with zero: r_n =0. Increment j: $j = j$+1. If $j < n$, return to repeat step 2. If $j > 2\cdot n$ go to step 4
3. If r_0=0, then code P is logically added to the current result P: R = R ⊕ P. Return to repeat step 2.
4. End of procedure. The value R = A⊗A⊗ U^{-1} rem P, U^{-1} is the multiplicative inversion of the polynomial $Q(x)=x^n$ on the Galois field formed by the polynomial P(x), i.e. U⊗U^{-1} rem P =1.

In order to obtain the correct value of the square of the number A on the Galois field, the result of the procedure should be multiplied by U: R′= R⊗U rem P. However, the specified correction is not performed during exposure.

The described procedure of squaring on the Galois field is illustrated by the example of squaring the number $A = 12_{10} = 1100_2$ on the Galois field, formed by the polynomial $P(x) = x^4+x^2+x+1$, which corresponds to the number $P=10111_2=23_{10}$; n =4, a $U=10000_2 = 32$, $U^{-1} = 8_{10} =1000_2$. Indeed, $U\cdot U^{-1}$ rem P = 32·8 rem 23 = 1. Real result R′=A⊗A rem P = 12⊗12 rem 23 = 12. Step-by-step change of variables R and A in the process of performing the above procedure of squaring A = 12 on the Galois field, with a generating polynomial $P(x) = x^4+x^2+x+1$ is shown in Table 1.

The result R is the product A⊗A⊗U^{-1} rem P = 12⊗12⊗8 rem 23 = 9.To obtain the correct value of the square of the number A=12 on the Galois field, multiply the result R by the value U: R′=R⊗U rem P = 9⊗16 rem 19 = 12.

The execution of the above procedure involves performing n shifts of the number A, $2\cdot n$ shifts of the number R, on average $0.5\cdot n$ logical addition operations (XOR), n bit value testing operations. Thus, the total number of logical operations required to implement the proposed squaring procedure on the Galois field is $3.5\cdot n$.

The main advantage of the proposed procedure is that it eliminates the testing of bits of the multiplier A. This opens up opportunities for group processing of several digits of the number and, thereby, reducing the amount of required calculations.

To theoretically substantiate the possibility of Montgomery group reduction, we prove that for any intermediate result code $R = r_n\cdot 2^n + r_{n-1}\cdot 2^{n-1}+\ldots+r_{k-1}\cdot 2^{k-1}+\ldots+r_1\cdot 2+r_0$, where $\forall j\in\{0,1,\ldots,n\}$: $r_j\in\{0,1\}$, there is a linear combination L(P) of no more than k shifted codes P: $L(P) = v_{k-1}\cdot 2^{k-1}\cdot P + v_{k-2}\cdot 2^{k-2}\cdot P+\ldots+v_1\cdot 2\cdot P+v_0\cdot P$, $\forall i\in\{0,1,\ldots,k\text{-}1\}$: $v_i\in\{0,1\}$, such that their k lower digits are equal to k least significant digits of R. The considered linear combination L(P) of shifted k codes P corresponding to the generating polynomial P(x) of the nth degree of the Galois field can be represented as an (n+k)-bit code D: $L(P) = v_{k-1}\cdot 2^{k-1}\cdot P + v_{k-2}\cdot 2^{k-2}\cdot P+\ldots+v_1\cdot 2\cdot P+v_0\cdot P = D = d_{n+k-1}\cdot 2^{n+k-1} + d_{n+k-2}\cdot 2^{n+k-2}+\ldots+d_1\cdot 2\oplus d_0$

Each i-th bit d_i from among the k least significant bits of the code D can be represented as a logical sum of pairwise products of bit components v_0, v_1, …, v_i and bit values p_0, p_1, …, p_i such that the sum of their indices is equal to i :

$$d_i = v_0 \cdot p_i \oplus v_1 \cdot p_{i-1} \oplus \ldots \oplus v_i \cdot p_0 = \bigoplus_{j=0}^{i} v_j \cdot p_{i-j}. \quad (1)$$

Table 1.
Dynamics of changes in variables R and A when performing the procedure of squaring A=12 on the Galois field formed by the polynomial $P(x) = x^4+x+1$

j	Transformation R	Transformation A
0	R=0 R>>1 = 00000	A=1100 A>>1 = 0110
1	R= 00000 R>>1 = 00000	A=0110
2	R= 00000 R>>1 = 00000	A=0110 A>>1 = 0011
3	R= 00000 R>>1 = 00000	A=0011
4	R= 10000 R>>1 = 01000	A=0011 A>>1 = 0001
5	R=01000 R>>1 = 00100	A=0001
6	R= 10100 R>>1 = 01010	A=0001 A>>1 = 0000
7	R= 01010 R>>1 = 00101 R⊕P = 10010	A=0000
8	R= 10010 R>>1 = 01001	

If we take into account that the generator polynomial $P(x)$ of the Galois field is prime, then p_0=1.. With this in mind, the expression for the i -th digit d_i of the number D can be represented as:

$$d_i = v_i \oplus \bigoplus_{j=0}^{i-1} v_j \cdot p_{i-1-j}. \quad (2)$$

In order to prove that for any of the 2^k-1 possible combinations (except zeros) of values of the k least significant digits of the number R, one can find a linear combination L(P) of codes P shifted by no more than k digits, it is necessary to show that for any code $r_{k-1},r_{k-2},\ldots,r_1,r_0$ (except zeros) there exists $v_{k-1},v_{k-2},\ldots,v_1,v_0$,such that $\forall i \in \{0,1,\ldots,k\text{-}1\}$: $r_i = d_i$.. This condition is satisfied if there is a solution for the following system of linear equations:

$$\begin{cases} r_0 = v_0 \\ r_1 = v_1 \oplus v_0 \cdot p_1 \\ r_2 = v_2 \oplus v_1 \cdot p_1 \oplus v_0 \cdot p_2 \\ \quad\quad \vdots \\ r_{k-1} = v_{k-1} \oplus v_{k-2} \cdot p_1 \oplus \ldots \oplus v_0 \cdot p_{k-1} \end{cases}. \quad (3)$$

An analysis of system (3) shows that it has a unique solution. Indeed, the value of v_0 is easily found from the first equation of systems (3): $v_0=r_0$. The second equation, taking into account the found value $v_0=r_0$, contains only one unknown component v_1, the value of which is uniquely found in the form: $v_1=r_1 \oplus r_0 \cdot p_1$. Similarly, the third equation of system (3), taking into account the found valuesv_0 and v_1, contains only one known value v3, which is uniquely in the form: $v_3=r_2 \oplus p_1 \cdot (r_0 \oplus r_1) \oplus r_0 \cdot p_2$. Thus, the analysis of system (3) shows that each of its following equations, including into account the previously identified unknowns, contains only one unknown component, which can be uniquely found from this equation. This means that system (3) always has a unique solution, that is, there

always exists a linear combination of numbers P shifted by no more than k-1 positions, such that its lower k digits are equal to the lower k digits of an arbitrary number R.

By the proved statement, one can perform Montgomery reduction by k digits of the current result simultaneously when squaring on Galois fields. This will significantly speed up the basic operation of exponentiation on Galois fields.

To do this, it is proposed once for a given generating polynomial P(x) of the Galois field for each of the possible 2^k-1 (except for zeros) combinations of the k -bit code $r_{k-1},r_{k-2},\ldots,r_1,r_0$ to calculate the values of the sums $L(P) = v_{k-1}\cdot 2^{k-1}\cdot P + v_{k-2}\cdot 2^{k-2}\cdot P+\ldots+v_1\cdot 2\cdot P+v_0\cdot P$, in which the values of k least significant digits are equal to the above combination. For given values of $r_{k-1}, \ldots, r_0$, the corresponding values $v_{k-1}, v_{k-2}, \ldots, v_1, v_0$ are found as a result of solving the system of equations (3). The calculation results are presented in the form of 2^k-1 tabular values T(1), T(2), …, T(2^k-1).

The value of k is chosen to be even and such that n is evenly divisible by it.

The foregoing is illustrated by the following example. Let n=8 and the Galois field is formed by the polynomial $P(x)=x^8+x^4+x^3+x^2+1$ For n=8, the number $U = 2^n = 256$, and its multiplicative inversion U^{-1} with the above generating polynomial P(x) is equal to $U^{-1}=127$; indeed 256⊗127 rem P(x) =1.

This polynomial corresponds to the number $P=100011101_2 = 285_{10}$. The lower four digits (for k=4) of this number are: p_0=1, p_1=0, p_2=1 и p_3=1 and p_3=1. In order to determine the values of v_0, v_1, v_2 and v_3 at which the lower three digits of the linear combination $v_3\cdot 2^3\cdot P \oplus v_2\cdot 2^2\cdot P \oplus v_1\cdot 2\cdot P \oplus v_0\cdot P$ are equal to 1010, i.e. r_3=1, r_2=0, r_1=1, r_0=0it is necessary to solve the system of equations (4) which, in the framework of the example, has the following form:

$$\begin{cases} 0 = v_0 \\ 1 = v_1 \\ 0 = v_2 \oplus v_0 \\ 1 = v_3 \oplus v_1 \oplus v_0 \end{cases} \quad (4)$$

Substituting the found value v_0=0 into the third equation, it is easy to determine v_2=0. Similarly, v_3=0 is deduced from the fourth equation. The found values determine the linear combination: $4\cdot P \oplus P = 2\cdot 285 = 570_{10} = 0010\ 0011\ 1010_2$. Thus, the table value T[1010] = T[10] = 570. The four least significant bits of this linear combination are equal to 1010. Similarly, linear combinations can be constructed for all possible 4-bit codes from 0001 to 1111, the values of which are summarized in Table 2.

In addition, to quickly form k-bit fragments of a polynomial square from k / 2-bit fragments of a number by inserting zeros between their bits, it is proposed to create and use a Z table. Such a table contains polynomial squares obtained by inserting zeros for each of $2^{k/2}$-1 k/2-bit codes. In particular, k=4 table Z consists of three rows: $Z[1] = Z[01_2] = 0001_2$, $Z[10_2] = 0100_2$ and $Z[11_2] = 0101$.

The actions outlined above, depending only on the generating polynomial P(x) and the number k of simultaneously processed bits, are carried out only once for cryptographic data protection systems, since the polynomial is part of the public key.

Calculation of the square A⊗A rem P of the number A on the Galois field is proposed to be performed in the following sequence:

1. The cycle counter j is set to zero: j=1, as well as the (n+k)-bit result code R: R=0.
2. R is shifted by k bits: R>>=k. The upper k digits of R are filled with a table code, the number of which is determined by the lower k/2 digits of A: $Z(a_{k/2-1},a_{k/2-2},\ldots,a_1,a_0)$.
3. If the lower k bits of R: $r_{k-1},r_{k-2},\ldots,r_0$ are equal to zero, go to step 4. Otherwise, the code $T[r_{k-1},r_{k-2},\ldots,r_0]$ is logically added to R: $R = R \oplus T[r_{k-1},r_{k-2},\ldots,r_0]$.
4. A is shifted by k/2 bits: A>>=k/2.. Increment j: j=j+1. If $j \le 2\cdot n/k$, return to repeat step 2.

The following example illustrates the proposed procedure for accelerated squaring on Galois fields. Let it be necessary to square the number $A=172_{10} = 1010\ 1100_2$ on the Galois field with the generating polynomial $P(x)=x^8+x^6+x^4+x^3$ for which table 2 is constructed for k=4. The true value of the result A⊗A rem P = 172 ⊗ 172 rem 285 = 11111_2=31.

The dynamics of changes in R and A over steps j of the described procedure for accelerated squaring on Galois fields is shown in Table 3.

The result R=66 differs from the true one and is the product $A \otimes A \otimes U^{-1}$ rem P = 172⊗172⊗147 rem 285. To obtain the real square R′ of the number A=172 on the Galois field, it is necessary to perform the Montgomery correction, that is, multiply the result R by the value U: R′=R⊗U rem P = 66⊗256 rem 285 = 31.

Table 2.

Tabular values of the results of precomputations for the Galois field with generating polynomial $P(x)= x^8+x^4+x^3+x^2+1$ for k=4

r_3,r_2,r_1,r_0	T	r_3,r_2,r_1,r_0	T
		1 0 0 0 (8)	2280_{10} = 1000 1110 1000_2
0 0 0 1 (1)	1425_{10} = 0101 1001 0001_2	1 0 0 1 (9)	1385_{10} = 0101 0110 1001_2
0 0 1 0 (2)	2850_{10} = 1011 0010 0010_2	1 0 1 0 (10)	570_{10} = 0010 0011 1010_2
0 0 1 1 (3)	1875_{10} = 0111 0101 0011_2	1 0 1 1 (11)	4027_{10} = 1111 1011 1011_2
0 1 0 0 (4)	1140_{10} = 0100 0111 0100_2	1 1 0 0 (12)	3228_{10} = 1100 1001 1100_2
0 1 0 1 (5)	2565_{10} = 1111 0011 0101_2	1 1 0 1 (13)	285_{10} = 0001 0001 1101_2
0 1 1 0 (6)	3990_{10} = 1111 1001 0110_2	1 1 1 0 (14)	1710_{10} = 0110 1010 1110_2
0 1 1 1 (7)	855_{10} = 0011 0101 0111_2	1 1 1 1 (15)	3135_{10} = 1100 0011 1111_2

Table 3

Step by step changes of variables R and A in each step execution of the procedure when squaring A⊗A rem P for A=172 and P=285 for k=4.

j	Transformation R		Transformation A (A>>=2)
	XOR	Shift (R>>=4)	
0	0000 0000	0000 0000 0000	1010 1100
1	-	0000 0000 0000	0010 1011
2	-	0011 0000 0000	0000 1010
3	-	0010 0011 0000	0000 0010
4	R=R⊕T[3] = 547⊕1875 = 0101 0100 0000	0000 0101 0100	0000 0000
5	R=R⊕T[4] = 84⊕1140= 0100 0010 0000	0000 0100 0010	

Analysis of the obtained results

The main advantage of the proposed method of performing the exponentiation operation on Galois fields is to speed up its computer implementation. This makes it possible to accelerate the implementation of a wide range of cryptographic data protection protocols accordingly.

When exponentiation on Galois fields is utilized in information security systems, the real length n (typically 2048 or 4096) of operands is 1-2 orders of magnitude greater than the capacity of the processor. Therefore, when estimating the number of operations required for squaring, one can

neglect operations on operands whose size is less than the processor capacity and take into account only operations on “long”, that is, n-bit operands.

The execution of the procedure described above includes performing n/k shifts of the number A, $2 \cdot n/k$ shifts of the number R, n/k operations of logical addition (XOR). Thus, the total number of logical operations required to implement the proposed squaring procedure on the Galois field is $4 \cdot n/k$. This means that the use of the group Montgomery reduction with processing of k digits at once makes it possible to speed up squaring on Galois fields by $0.75 \cdot k$ times.

Conclusion

Conducted research aimed at speeding up the computational implementation of the exponentiation operation on Galois fields, which is basic for elliptic cryptography, yielded the following results.A method of accelerated squaring on Galois fields is proposed and studied, which is distinguished by the fact that it uses the algebraic properties of this operation in combination with the application of group reduction, which allows to speed up this operation. The technology of implementation and application of the proposed method is described in detail. Theoretically and experimentally, it has been proven that the method provides acceleration of the square operation by 6-8 times, depending on the number of digits in the group. The exposition is illustrated by numerical examples.

The application of the proposed method for the computational implementation of squaring on Galois fields, which takes 2/3 of the calculations of the exponentiation operation on Galois fields, allows to speed up the execution of this basic operation of a wide range of cryptographic algorithms by 2.4 times.

The developed method is oriented for use in information protection systems based on high-speed public key cryptography.

References

1. Nikolajchyk J. Galose Fields code: theory and application / J.M. Nikolajchyk // Ternopil.- Edition TNU. —2012. – P.576.
2. Rahma, M. Principles of construction and design of operational nodes for Galois fields used in cryptographic protection of information based on elliptic curves / Mohhamed Rahma, V.S. Gluhov, I.M. Jolubak // Cyber-physical systems: multi-level organization and design. – Lviv: Edition Magnolia-2006.- 2019. – pp. 58-131.
3. Kalmykov I.A. Development of a method of nonlinear information encryption using the exponentiation operation for a finite Galois field / I.A. Kalmykov, E.C. Stepanov, K.T. Tincherov// Modern science-intensive technologies. - 2019.- № 9. - pp.84—89.
4. Schneier B. Appllied Cryptography. Protocols, Algorithms and Source Code in C. Wiley.- 2015.-pp.784.
5. Zahariydakis L. Using of Galois fields algebra for implement the zero knowledge identification and authentication of remote users / Lefteris . Zahariydakis, Maksimuk V.R. // Electronic modeling - 2017.- T.6, №39. - pp.33-45.
6. Moustafa A.A. Fast Exponentiation in Galois Fields GF(2m) Using Precomputations. / A.A.Moustafa// Contemporary Engineering Sciences, Vol. 7, - 2014, no. 4, - pp.193 – 206.
7. Markovskyi O. Method of acceleration of exponentiation using precalculations / O. Markovskyi, O. Rusanova, A. Olievskyi, V. Cherevik // Telecommunications and information technologies. - 2018.- № 1(58). – pp.31-39.
8. Rahma, M. Computing Square Roots and Solve Equations of ECC over Galois Fields /M. Rahma, V. Hlukhov / International Youth Science Forum ”Litteris Et Artibus”, November 23-25, 2017, Lviv, Ukraine, pp. 437-440.
9. Francis N. Closed formulas for exponential sums of symmetric polynomials over Galois fields / N.Francis, A.Luis, M. Castro // Journal of Algebraic Combinatorics, — 2019. — V. 50. - P. 73-98.

10. Fitzpatrick P. Algorithm and Architecture for a Galois Fiels multiplicative Arithmetic Processor./ P. Fitzpatrick, Popovici E. M. // IEEE Trans. on Information Theory. — 2003— V.49, - № 12, — pp. 3303— 3307.
11. Wu H. Finite field multiplier using redundant representation./ H. Wu, M.A. Hasan, I.F. Blake, S.Gao // IEEE Trans. Computers. —2002 — V.51,- № 51. — pp. 1306 — 1316.
12. Samofalov K.G. A method of accelerated implementation of exponentiation on Galois fields in information protection systems / K.G. Samofalov, A.S. Sharshakov // Problems of informatization and control: K.,NAU.- 2011. - v.2, № 33 - pp.143-151.
13. Osadchyy V. The Order of Edwards and Montgomery Curves «, / V.Osadchyy // WSEAS Transactions on Mathematics, - 2020.- Vol. 19.- № 25, - pp. 253-264.
14. Markovskyi Oleksandr The Employment of Montgomery reduction for acceleration of exponent on Galoise fields calculation / O. Markovskyi, V. Masimyk, O.Kot // Proceeding of International Conference on Security, Fault Tolerance, Intelligence" (ICSFTI2020), 13-14 may 2020, Kyiv .- pp.44-49.
15. Elfard S. Justification of Montgomery Modular Reductions / S. Elfard //Advanced Computing.- 2012. - № 11. – pp.41-45.