Contemporary Engineering Sciences, Vol. 7, 2014, no. 4, 193 - 206
HIKARI Ltd, www.m-hikari.com
http://dx.doi.org/10.12988/ces.2014.3955

Fast Exponentiation in Galois Fields GF(2^m)

Using Precomputations

Akram A. Moustafa

Department of Computer Science
Al al-Bayt University, P.O. Box 922283, 11192, Mafraq, Jordan
hamarshi@aabu.edu.jo

Abstract

In **t**his paper, a new method proposed for fast exponentiation in Galois fields with processing of several bits of the exponent and with using two tables of precomputations. They also describe, analyze and illustrate in details with examples the proposed technology of exponentiation in Galois fields. Choice of an optimal computation complexity is the number of simultaneously processed bits of the exponent. Theoretical and experimental results demonstrated that the proposed method allows decreasing the number of needed processing operations for exponentiation in more than 7 times. The obtained results proved that along with decreasing the number of operations, the proposed method opens wide possibilities of parallelizing the process of exponentiation. The proposed method of fast exponentiation in Galois fields oriented for using in cryptographic systems for information protection.

Keyword: Galois Fields, Precomputations, exponentiation, cryptographic, complexity, discrete logarithm

1. Introduction

The operations in Galois fields (2^m) are very important in modern information technologies. They are used for coding the result of measurements, compressing data, and are primary for a wide class of correction codes [1]. One of the most important

spheres of application of the operations in Galois fields is systems of cryptographic protection of information [2]. The main advantage of the operations in Galois fields, compared with the ordinary modular arithmetic, is the absence of carries. Actually, when the summation is performed in Galois fields, every bit of numbers processed independently from each other allows accelerate computations and significantly simplifies the scheme for hardware implementation.

In recent years, expansion of using "cloud computing" technologies significantly exacerbated the problem of reliability of the cryptographic protection of information [2]. To counteract the threat, the security of all cryptographic protection systems should be reinforced, first of all, by increasing number of bits in the keys. For cryptographic algorithms with open key, where the basic operation is exponentiation, increasing the number of bits in the numbers, which these algorithms use, results the exponentiation increase of computational complexity. It causes significant deceleration of functions of protection of the information [3].

That is why the actual problem today is in increasing performance of realization of cryptographic algorithms. The possible ways of solving this important problem are:
(a) decreasing the computational complexity of basic operations by using precomputations, (b) searching for the possibilities of parallelizing the computations, expansion of using the cryptographic algorithms that based on Galois fields, which don`t have carries, (c) applying of specialized hardware – cryptoprocessors.

The basic operation of cryptographic algorithms with open key that defines the time for performing them is exponentiation [4]. So, for increasing the performance of realization of cryptographic algorithms, it is necessary to explore the possibilities of acceleration execution of the exponentiation operation.

Thus, the scientific task for increasing the speed of execution of the exponentiation in Galois fields is actual for modern level of development of the information technologies.

2. Analysis of the known ways to implement the exponentiation in Galois fields

The operation of exponentiation in finite fields has two important features which define the ability of efficient applying it in mechanisms of cryptographic protection of information:

(1) The operation of exponentiation is based on two computing functions: multiplication without carry and reduction – finding the remainder of the polynomial division of the result of multiplication to the formulated polynomial of the field P. This multiplication performs significantly easier and faster than an ordinary arithmetic multiplication. Similarly, the operation of the polynomial division performs

significantly easier and faster than arithmetic division. It allows using the operation of multiplication in finite fields like the measure of acceleration the procedures of the cryptographic protection of data in computer systems and networks.

(2) In the heart of any cryptography is the irreversible mathematical conversion, that is, such a conversion for which, in principle, analytically cannot be derived the inverse. The classical examples of analytically unsolvable problems are discrete logarithm and solving the systems of non-linear Boolean equations. It is obvious that the problem of discrete logarithm is unsolvable in ordinary algebra and in the algebra of finite fields. The practical confirmation of this provision is using of such conversions in the block encryption standard Rijndael. Obviously, the only way of solving the problem of discrete logarithm in ordinary algebra and in the algebra of finite fields is the brute force search.

To distinguish the operations in Galois fields from similar operations in ordinary arithmetic for designation of the operation of addition, the designations below should be used [2]. The operation of addition in Galois fields, which actually corresponds to the logical summation (XOR) below is indicated by symbol '⊕', whereas symbol '+' is used for ordinary summation with carry. The operation of multiplication without inter-bit carry below is denoted by '⊗' in contrast to '·', which is used for ordinary multiplication. The product without inter-bit carry $D=U\otimes V=\{d_{2\cdot n},\ldots,d_3,d_2,d_1\}$ $=d_1+2\cdot d_2+4\cdot d_3+\ldots+2^{2\cdot r}\cdot d_{2\cdot r}$, $\forall l\in\{1,..,2\cdot r\}$: $d_l\in\{0,1\}$ of two r-bit numbers $U=\{u_r,\ldots,u_2,u_1\}=u_1+2\cdot u_2+\ldots+2^r\cdot u_r$ and $V=\{v_r,\ldots,v_2,v_1\}=$ $=v_1+2\cdot v_2+\ldots+2^r\cdot v_r$, $\forall i\in\{1,..,r\}$: $v_i,u_i\in\{0,1\}$ is calculated by the following method: $D = U\cdot v_1 \oplus (U\cdot v_2)<<1\oplus \ldots \oplus (U\cdot v_r)<<(r\text{-}1)$, where $p<<q$ – the operation of the logical shift of the number p to the left by q bits. The operation of raising the number A to the exponent E without carry is denoted by $A|^E$. The operation of calculation of remainder of the polynomial division that corresponds to the number A to the polynomial corresponding to M is denoted as *A rem M.* Considering the above definitions the exponentiation operation in Galois fields may be expressed as $A|^E$ rem *M,* where M is the number corresponding to the formulated polynomial of the field.

By the present time a number of methods of performing the operation of exponentiation in Galois fields were proposed [5-8]. The analysis indicates, what authors of these methods considering the ability of parallelizing the execution of the basic for exponentiation operation of multiplication in Galois fields as a main reserve for increasing performance.

The procedure of modular exponentiation $A|^E$ rem M in Galois fields, like an ordinary modular exponentiation, reduces to consequent execution of m cycles, each of which has operation of squaring the result of previous cycle and, additionally, operation of multiplication (in dependence of current bit of the exponent E). Based on order of analyzing bits of the exponent E, there are two types of modular exponentiation: from right to left and from left to right (in practice, the analysis of bits of the exponent E from left to right is used more often).

Formally, this method can be represented as:

1. $R = 1$; $j = m$-1.
2. $R = (R \otimes R)$ rem M ;
3. if $e_j = 1$ then $R = (R \otimes A)$ rem M
4. $j = j$-1; if $j >= 0$ return to 2.

Result: $R = A|^E$ rem M.

To represent a fast procedure for the calculation of the exponent in Galois fields will be illustrated with the following example: $P(x) = x^9 + x^8 + x^7 + x^6 + x^5 + x + 1$, so $M = 1111100011_2 = 995$. The exponentiation $314|^{351}$ rem M in Galois fields (A=314, $E=351=101011111_2$) is executed in 9 cycles. The changing of each variable of the represented procedure is submitted in Table 1.

Table 1: The dynamic of changing variables in cycles during calculating the exponent $314|^{351}$ rem M in Galois field

Cycle	j	Initial R	Current bit of the exponent e_j	Final R
1	8	1	1	314
2	7	107	0	107
3	6	244	1	117
4	5	416	0	416
5	4	156	1	254
6	3	86	1	306
7	2	43	1	153
8	1	242	1	303
9	0	378	1	212

So, the result of the exponentiation $314|^{351}$ rem 995 is final R in last cycle – 212.

Assessment of calculating complexity of the represented procedure of the exponentiation in Galois fields can be performed by counting the needed operations of XOR and shifts. The exponentiation procedure consists of m cycles, in each of which the operation of squaring in Galois field is performed and, with probability of 0.5 – multiplication on constant A (if considering appearing of 0 and 1 as equiprobable). The operation of multiplication in Galois fields consists of two consequentially performed stages: polynomial multiplication and reduction, which is polynomial division.

The operation of polynomial multiplication also consists of m cycles, in each of which with probability of 0.5 (depends on current bit of multiplier) logical summation of multiplicand and intermediate result is performed. Also, in every cycle, carry of the multiplier code (or intermediate result) is performed. So, for performing the operation of polynomial multiply, in average, requires 0.5*m operations of XOR and m shift operations, which in total give 1.5*m elementary operations [6].

Similarly, the operation of polynomial division of 2*m-bit code to m-bit code of the formulated polynomial requires m shifts and, in average, 0.5*m operation of XOR. So, the operation of multiplication in Galois fields requires, in average, 3*m elementary operations of XOR and shifts. Consequently, the operation of exponentiation in Galois fields by the above described method requires, in average, $Q=4.5*m^2$ similar operations.

The best known method of acceleration of the exponentiation in Galois fields is combination of performing operations of multiplication and division $u=ab/c$, where a,b,c – polynomials in Galois fields GF(2^m). This method is based on the proposal made by Fitzpatrick P. [5]. Algorithm of combined performance of multiplication and division operations, which allows to find the result of $u=ab/c$ as minimal element of solutions of identity $uc + vf + wab \equiv 0 \bmod x^{2m-1}$. In practice, implementation of this method is efficient only in case of hardware implementation of exponentiation.

Efficiency of known methods of acceleration in Galois fields significantly limited because of authors of these methods who trying to solve the problem in general, excluding the specificity for different applications.

3. Method of accelerated computation of the exponentiation in Galois fields

The main sphere of using the exponentiation $A|^E$ rem M in Galois fields for protection of information in modern systems is cryptographic protocols with open key [1-8]. Specificity of such protocols is what public key is associated with its owner and can be changed rarely, so, it can be considered as permanent. Respectively, component M (the number corresponds to formulated polynomial of Galois field) of the exponentiation operation can be considered as constant. It allows distinguish the computations dependent on M, performs them once with saving its results in memory, and uses these results during every computation of the exponent. Such a method is used for acceleration the operation of exponentiation in ordinary algebra, which is the basis of such famous cryptographic algorithms as RSA, El-Gamal and DSA [9].

The theoretical base of the proposed method of exponentiation in Galois fields is represented by the following theorem.

Theorem: If code n of exponent in algebra of multiplication without carry is power of 2 ($n=2^k$), where k-integer, that is in polynomial presentation contains one component

$P(n) = x^{2^k}$, the result of the exponentiation $R|^E$ in Galois field of number R= $\{r_0, r_1,..., r_{m-1}\}$ can be presented as the sum:

$$R|^n = r_0 + r_1 \cdot 2^n + r_2 \cdot 2^{2 \cdot n} + r_3 \cdot 2^{3 \cdot n} + ... + r_{m-1} \cdot 2^{(m-1) \cdot n}.$$
$$P(R|^n) = r_0 + r_1 \cdot x^n + r_2 \cdot x^{2 \cdot n} + r_3 \cdot x^{3 \cdot n} + ... + r_{m-1} \cdot x^{(m-1) \cdot n}. \quad (1)$$

For n=2 this theorem transforms into the following property of square of the number A without carry: even bits of its binary code equal 0, and odd – coincide with binary bits of the number A. For example, if R=**1011**$_2$, then $R|^2$ = **1000101**$_2$.

The most important, from the side of computation organization, consequence of presented theorem is the fact, what computation of the exponent, which is power of 2 in algebra without carry doesn`t require any computational operations, and reduced only to permutation of bits of the original number. It means what the exponentiation to power of 2 in Galois fields requires only computations for reduction. With constant formulated polynomial of the field such computations can be performed in advance and the results can be saved in a Table. Truly, if for n which is power of 2, $R|^n = r_0 + r_1 \cdot 2^n + r_2 \cdot 2^{2 \cdot n} + r_3 \cdot 2^{3 \cdot n} + ... + r_{m-1} \cdot 2^{(m-1) \cdot n}$, then $R|^n$ rem $M = r_0$ rem $M + r_1 \cdot 2^n$ rem $M + r_2 \cdot 2^{2 \cdot n}$ rem $M + r_3 \cdot 2^{3 \cdot n}$ rem $M + ... + r_{m-1} \cdot 2^{(m-1) \cdot n}$ rem M. If values T[0]=1, T[1]=2^n rem M, T[2]=$2^{2 \cdot n}$ rem M, T[3]= $2^{3 \cdot n}$ rem M, ... T[m-1]= $2^{(m-1) \cdot n}$ rem M are computed in advance, then $R|^n$ rem M can be calculated as:

$$R|^n \text{ rem } M = r_0 \cdot \text{T}[0] + r_1 \cdot \text{T}[1] + r_2 \cdot \text{T}[2] + r_3 \cdot \text{T}[3] + ... + r_{m-1} \cdot \text{T}[m\text{-}1]. \quad (2)$$

In fact, formula (2) can be applied for squaring in Galois field of current value of the exponent $R|^2$ rem M, and that this operation which is performed in each cycle of exponentiation. Assuming that the values of the bits R: r_0, r_1,…,r_{m-1} with same probability equal 0 and 1, it is obvious that computation of $R|^n$ rem M with formula (2) requires, in average, m/2 operations of XOR.
This formula (2) allows calculating $R|^n$ rem M fast with n greater than 2, without computing smaller powers. For example, formula (2) allows to compute rapidly $R|^4$ rem M without computation of $R|^2$ rem M. As a result, the possibility of fast processing of multiple bits of the exponent code appears.

Indeed, during processing of two bits of the exponent, their possible values equal: 00, 01, 10 and 11. In first case the value of the variable R transforms into $R|^4$ rem M, in second one - ($R|^4$ rem $M \otimes A$) rem M, in third one - ($R|^4$ rem $M \otimes A|^2$ rem M) rem M and in fourth case, the result of transformation is ($R|^4$ rem $M \otimes A|^3$ rem M) rem M. Calculation of $R|^4$ rem M can be organized efficiently with using the results of precomputations in accordance with formula (2). Before exponentiation of $A|^E$ rem M, the values of $A|^2$ rem M and $A|^3$ rem M can be pre-calculated. In this case, processing of 2 bits of the exponent requires computation of $R|^4$ rem M and multiplication of the result on A, $A|^2$ rem M or $A|^3$ rem M.

The Essence of the proposed method of acceleration of exponentiation in Galois fields is in organization of simultaneous processing of k bits of code of the exponent with using precomputations.

During processing k bits of code of the exponent simultaneously, their probable values will be in the range from 00..0 to 11..1. When values of k-bits of the exponent equal 00..0 , the current value of R transforms into $R\big|^{2^k}$ rem M. When k-bit fragment of the exponent equals 00..01, the value of R transforms into ($R\big|^{2^k}$ rem M⊗A) rem M, when k-bit fragment of the exponent equals 00..010, the current value of R transforms into ($R\big|^{2^k}$ rem M⊗A$\,\big|\,$2 rem M) rem M. Similarly, when k-bit fragment of the exponent consists of 1 (equals 11..1), the current value of R transforms into ($R\big|^{2^k}$ rem $M\otimes A\big|^{2^k-1}$ rem M) rem M.

Proceeding from the above, considering the base polynomial of the Galois field as constant, m values are calculated and placed into Table T: T[0]=1, T[1]= 2^{2^k} rem M, T[2]= $2^{2\cdot 2^k}$ rem M, T[3]= $2^{3\cdot 2^k}$ rem *M,* ... T[m-1]= $2^{(m-1)\cdot 2^k}$ rem M.
Before each exponentiation, $A\,\big|^{E}$ rem M, b values are calculated and placed into Table of W 2^k-2 values: $W[0] = A\,\big|^{2}$ rem M, $W[1]= A\,\big|^{3}$ rem M, …, $W[2^k\text{-}3] = A\big|^{2^k-1}$ rem M.

The whole procedure of exponentiation in Galois fields formally can be described in such way:

1. Current value $R=1$; number j of the beginning of the group of bits $j=m\text{-}1$.
2. Value q of the binary code is calculated, formed by k adjacent bits of code of the exponent, starting from j bit:

$$q = \sum_{l=0}^{k-1} e_{j-l} \cdot 2^{k-l-1}$$

3. Calculating $R\big|^{2^k}$ rem M :

$$G = \sum_{h=0}^{m-1} r_h \cdot T[h] \qquad (3)$$

4. Calculating the new value of R:

$$\begin{aligned} &\text{if } q = 0: \quad R = G\ ; \\ &\text{if } q = 1: \quad R = (\ G \otimes A\ ) \text{ rem } M; \\ &\text{if } q > 1: \quad R = (\ G \otimes W[q\text{-}2]\ ) \text{ rem } M. \end{aligned} \qquad (4)$$

5. If $j>k\text{-}1$, then $j=j\text{-}k$ and return to item 2, else – end.

The proposed method of fast computation of the exponent in Galois fields can be illustrated by the following example:

Let m=9 and formulated polynomial of the field equal:
$P(x) = x^9 + x^8 + x^7 + x^6 + x^5 + x + 1$, respectively $M = 1111100011_2 = 995$. If we suggest that 3 bits are processed simultaneously, then Table T contains m=9 values and formed as: T[0]=1, T[1] = 2^{2^k} rem $M = 2^8$ rem 995 = 256, T[2]= $2^{2\cdot 2^k}$ rem $M = 2^{2\cdot 8}$ rem 995 = 266 and so on. The calculated values T[0] … T[8] are in Table 2.

Table 2: Values of $T[j] = 2^{j\cdot 2^k}$ rem *M* for *m*=9, *k*=3 and *M*=995

J	T[*j*]
0	$1 = 000000001_2$
1	$256 = 100000000_2$
2	$266 = 100001010_2$
3	$163 = 010100011_2$
4	$334 = 101001110_2$
5	$309 = 100110101_2$
6	$403 = 110010011_2$
7	$504 = 111111000_2$
8	$458 = 111001010_2$

If we suggest that $314|^{351}$ rem 995 is calculated, that is A=314 and E=351=101011111_2, the second Table W of precomputations which depends on A, is formed as: $W[0] = A|^2$ rem $M = 314|^2$ rem 995 = 107, $W[1] = A|^3$ rem $M = W[0] \otimes A$ rem M = 107⊗314 rem 995 = 38 and so on. The values of Table *W* are in Table 3.

Table 3: Values of W[*j*]=$A|^{j+2}$ rem *M* for *m*=9, *k*=3 and *M*=995

J	W[*j*]
0	107
1	38
2	244
3	117
4	49
5	67

The process of calculating of the exponent in Galois fields for 9-bit number and with simultaneous processing of 3 bits requires 3 cycles. Changing of each variable on each of 3 cycles of calculating $314|^{351}$ rem 995 using proposed method is shown in Table 4.

The calculation of $G = R|^8$ rem M, for example, for second cycle (R=117_{10} =1110101_2) is performed by the formula (3) in such a way:

$G = T[0] \oplus T[2] \oplus T[4] \oplus T[5] \oplus T[6] = 000000001_2 \oplus 100001010_2 \oplus 101001110_2 \oplus 100110101_2 \oplus 110010011_2 = 011100011_2 = 227$.

Table 4: Dynamics of changing variables in proposed procedure of exponentiation in Galois fields (for example $314\big|^{351}$ rem 995 with simultaneous processing of 3 bits of the exponent)

Cycle	*J*	Initial *R*	$G=R\big\vert^{8}$ rem *M*	*q*	Final *R*
1	8	1	1	$5=101_2$	1⊗W[3] rem *M*=117
2	5	117	227	$3=011_2$	227⊗W[1] rem *M*= 306
3	2	306	177	$7=111_2$	177⊗W[5] rem *M* =212

In this way, the result of exponentiation $314\big|^{351}$ rem 995 is final value of *R* in last cycle = 212.

4. Assessment of efficiency

Assessment of complexity of computations of the exponent in Galois fields by the proposed method can be done by the following way. Table T depends only on formulated polynomial and calculated once.

With simultaneous processing of *k* bits of the exponent, forming of Table W requires 2^k-2 multiplication operations in Galois fields or in average $(2^k\text{-}2)\cdot 3\cdot m$ processor operations of XOR and shifts.

Exponentiation in Galois fields by the proposed method performs in *m/k* cycles. In each cycle according to proposed method calculated $R\big|^{2^k}$ rem *M* with using the results of precomputations from Table T, which requires according to formula (3) in average, *m/2* operations of XOR. Moreover, in each cycle the proposed method performs the operation of multiplication in Galois field of calculated value $R\big|^{2^k}$ rem *M* on value, chosen from Table W. That requires, in average, $3\cdot m$ elementary operations of XOR and shift. In this way, during one cycle, the proposed method performs, in average, $3.5\cdot m$ operations, and in m/k cycles of exponentiation in Galois field - $3.5\cdot m^2/k$. Considering forming of Table W, the overall number *N(k)* of processor operations of XOR and shifts which are required for exponentiation in Galois fields by the proposed method is:

$$N(k) = (2^k\text{-}2)\cdot 3\cdot m + 3.5\cdot m^2/k. \qquad (5)$$

It is obvious that function *N(k)* has minimum which can be reached by $\frac{dN(k)}{dk} = 0$. Therefore, the value of k, which provides minimal number of operations required for performing exponentiation by the proposed method, can be determined by the condition:

$$\frac{dN(k)}{dk} = 3\cdot 2^k \cdot \ln k \cdot m - \frac{3.5\cdot m^2}{k^2} = 0 \qquad (6)$$

$$\text{or } 2^k \cdot \ln k \cdot k^2 \approx m$$

Thus, when k=4, the numerical value $2^k \cdot \ln k \cdot k^2 = 335$, when k=5 the numerical value $2^k \cdot \ln k \cdot k^2 = 1287$ and when k=6 the numerical value $2^k \cdot \ln k \cdot k^2 = 4128$. It means that, for example, with typical for cryptographic system of protection of information with open key value of m=1024, the least volume of computations is reached when k=5.

With m=1024 and k=5, the value of $N(k) = 826163$. When compared to given above assessment of the number of operations $Q=4.5 \cdot m^2 = 4718592$ for ordinary exponentiation, it shows that the proposed method allows to accelerate the operation of exponentiation in Galois fields in Q/N(5)=5.7 times.

5. Parallelization

Acceleration of exponentiation, which can be achieved by the proposed method, is based on decreasing of the number of performed operations by using precomputations.

Along with that an important advantage of the proposed method of exponentiation in Galois fields is the ability to parallelize the calculation of the exponent. This ability is based on the fact that in the proposed method, the calculation of $R\big|^{2^k}$ rem M requires the same number of operation as calculation $R\big|^2$ rem M. As shown above, calculating of $R\big|^2$ rem M with using Tables T requires $m/2$ operations, while computation $R\big|^2$ rem M by multiplication in Galois field requires $3 \cdot m$ processing operations.

The easiest method of parallelizing the calculation of the exponent is the separation of the code of index into h components $q_1, q_2, \ldots, q_h$, which represents groups of bits of code of the exponent. It means that they can be represented as $q_1 = E \wedge p_1$, $q_2 = E \wedge p_2, \ldots$, $q_h = E \wedge p_h$. Operation of XOR is denoted by symbol '$\wedge$', and $p_1, p_2, \ldots, p_h$ – unit masks of groups of bits of code of the exponent, that is p_1 - m-bit code, which consists of m_1 consecutive ones and (m-m_1) zeros: $p_1 = 2^{m-m_1} \cdot (2^{m_1} - 1)$, $p2$ - (m-$m1$)-bit code, which consists of m_2 consecutive ones and (m-m_1-m_2) zeros: $p_2 = 2^{m-m_1-m_2} \cdot (2^{m_2} - 1)$, and p_h – code which consists of m_h ones: $p_h = 2^{m_h - 1}$, herewith $m_1 + m_2 + \ldots + m_h = m$.

For example, if m=12, $E = 101101110101_2$, and h=3 and besides m_1=5, m_2 = 4, m_3 =3, then $p_1 = 111110000000_2$, $p_2 = 1111000_2$, $p_3 = 111_2$. Respectively, q_1=**10110**0000000_2, q_2=**1110**000_2, q_3= **101**$_2$.

The exponent in Galois field $A\,|^E$ rem M can be represented as:

$$A\,|^E \text{ rem } M = A\big|^{q_1+q_2+\ldots+q_h} \text{ rem } M = \qquad (7)$$

$(A\big|^{q_1} \text{ rem } M \otimes A\big|^{q_2} \text{ rem } M \otimes \ldots \otimes A\big|^{q_h} \text{ rem } M) \text{ rem } M$

Each of the components $A\big|^{q_1}$ rem M, $A\big|^{q_2}$ rem M and $A\big|^{q_h}$ rem M can be calculated independently and in parallel. After calculation the results are multiplied according to formula (7).

With traditional organization of exponentiation, the proposed method of parallel calculation of $A\,|^E$ rem M in Galois fields is ineffective. Indeed, time T_0 of calculation $A\,|^E$ rem M by a traditional method without parallelizing is $4.5 \cdot m^2 \cdot \tau$, where τ - time for performing Operations of XOR or shifts. Time of calculation T_1 of the first exponent $A\big|^{q_1}$ rem M according to formula (7) equals $T_1 = 3 \cdot m^2 \cdot \tau + 1.5 \cdot m_1^2 \cdot \tau$, of the second one - $T_2 = 3 \cdot m \cdot (m - m_1) \cdot \tau + 1.5 \cdot m \cdot m_2 \cdot \tau$ and so on. Finally, time for calculation the last h-component according to formula (7): $A\big|^{q_h}$ rem M equals $T_h = 4.5 \cdot m \cdot m_h \cdot \tau$. If considering the ratio of calculation time T_0 of $A\,|^E$ *rem* M without parallelizing to time T_1 of calculation of the first exponent $A\big|^{q_1}$ rem M, it is easy to show that:

$$\frac{T_1}{T_0} = \frac{3 \cdot m^2 \cdot \tau + 1.5 \cdot m_1 \cdot m \cdot \tau}{4.5 \cdot m^2 \cdot \tau} = 0.67 + 0.33 \cdot \frac{m_1}{m} > 0.67 \qquad (8)$$

From formula (8), it follows that $T_0/T_1 < (0.67)^{-1} = 1.5$. It means that with parallelizing, the calculation of the exponent in Galois field with traditional method does not allow reducing the required time more than in 1.5 times.

With the proposed organization of calculation of the exponent in Galois field using precomputations time for involution to the power of 2 by using precomputations from Table T, which is $0.5 \cdot m \cdot \tau$ is significantly less than the time of multiplication on number from Table W, which is $3 \cdot m \cdot \tau$.

With parallel calculation of the exponent on k processors by the proposed method using precomputations and simultaneous processing of k bits of code of the exponent on h processors, Table W is filled for all processors at once. Time T_1' of calculation on first processor of the component $A\big|^{q_1}$ rem M according to formula (7) equals $T_1' = 3.5 \cdot \tau \cdot m \cdot m_1/k + 0.5 \cdot \tau \cdot m \cdot (m - m_1)/k$. Similarly, time T_2' of calculation on second processor of the component $A\big|^{q_2}$ rem M equals $T_2' = 3.5 \cdot \tau \cdot m \cdot m_2/k + 0.5 \cdot \tau \cdot m \cdot (m - m_1 - m_2)/k$. Time T_3' of calculating on third processor of the component $A\big|^{q_3}$ rem M equals $T_3' = 3.5 \cdot \tau \cdot m \cdot m_3/k + 0.5 \cdot \tau \cdot m \cdot (m - m_1 - m_2 - m_3)/k$. Finally, time T_h' of calculation on the last h-component according to formula (7): $A\big|^{q_h}$ rem M on h processor equals $T_h' = 3.5 \cdot \tau \cdot m \cdot m_h/k$.

It is obvious that the best efficiency can be achieved if all of the processors are working concurrently. This implies that maximal effect for accelerating calculations with parallel work of h processors is achieved with equality of times T_1', T_2',…,T_h':

$$T_1' = T_2' = T_3' = \ldots = T_{h-1}' = T_h'. \qquad (9)$$

Condition formula (9) along with equation $m_1+m_2+...+m_h = m$ forms the system of equations, which can be used for defining the numerical values of $m_1,m_2,...,m_h$ with which the maximum efficiency of parallelizing is provided as:

$$\begin{cases} m_1 + m_2 + ... + m_h = m \\ 3 \cdot m_1 + 0.5 \cdot m = 3.5 \cdot m_h \\ 3 \cdot m_2 + 0.5 \cdot m - 0.5 \cdot m_1 = 3.5 \cdot m_h \\ 3 \cdot m_3 + 0.5 \cdot m - 0.5 \cdot m_1 - 0.5 \cdot m_2 = 3.5 \cdot m_h \\ ..... \\ 3 \cdot m_{h-1} + 0.5 \cdot m - 0.5 \cdot m_1 - ... - 0.5 \cdot m_{h-2} = 3.5 \cdot m_h \end{cases} \quad (10)$$

By subtracting from second equation of the system (10), the third one easily gets: $3.5 \cdot m_1 = 3 \cdot m_2$. By subtracting from third equation of the system (10), the fourth one can be obtained: $3.5 \cdot m_2 = 3 \cdot m_3$. Similarly, it is not difficult to show that for any i of m_i value, where $i \in \{2,3,..,h\}$ the equality $m_{i-1} = m_i \cdot 3/3,5 = 0.857 \cdot m_i$ is true:

$$m_i = m_1 \cdot \left(\frac{3.5}{3}\right)^{i-1} \approx m_1 \cdot (1.167)^{i-1} \quad (11)$$

From the first equation (10) including (11), the value of m_1 can be obtained as:

$$m_1 = \frac{m}{\sum_{j=0}^{h-1} (1.167)^j} \quad (12)$$

For example, implementing exponentiation of 1024 –bit numbers (m=1024) in Galois fields with simultaneous processing of 5 bits (k=5) by the proposed method on four processors (h=4) formation of 30 values of Table W is performed on one processor and requires $(2^k-2) \cdot 3 \cdot m = 92160$ operations.

The exponentiation itself is performed simultaneously on four processors. The value m_1 is defined from formula (12) $m_1=m/(1+1.167+(1.167)^2+(1.167)^3)$ $=1024/5.12= 200$, values m_2, m_3 and m_4 are defined from formula (11) as: $m_2 = m_1 \cdot 1.167= 233$, $m_3 = m_2 \cdot 1.167= 272$. $m_4=m_3 \cdot 1.167 = 319$. The overall number of operations of XOR and shifts, performed on first processor is calculated as: $3.5 \cdot m \cdot m_1/k + 0.5 \cdot m \cdot (m-m_1)/k =$ 227738, the number of operations performed on second processor: $3.5 \cdot m \cdot m_2/k + 0.5 \cdot m \cdot (m-m_1-m_2)/k = 227532$, the number of operations performed on third processor: $3.5 \cdot m \cdot m_3/k + 0.5m \cdot (m-m_1-m_2-m_3)/k =$ 227636. On fourth processor $3.5 \cdot m \cdot m_h/k = 228659$ operations are performed.

Matching the given data, it shows the load of each 4 processors during exponentiation is approximately equal. Time of exponentiation with using 4 processors is defined by sum of time for preparing Table W and time of work of one of the processors. In particular, for the considered example, time $t_{5,4}$ of calculation A^E rem M on four processors with simultaneous processing of 5 bits is defined by sum: $t_{4,5} = (2^k-2) \cdot 3 \cdot m \cdot \tau + 3.5 \cdot \tau \cdot m \cdot m_3/k = 320819 \cdot \tau$.

Time of calculation A^E rem M by the proposed method on one processor for the considered example is estimated as $N(5) \cdot \tau = 826163 \cdot \tau$. Comparing the estimates given

above, it shows that for the given example, the exponentiation on 4 processors by the proposed method allows to accelerate calculation in $N(5)\cdot\tau/t_{5.4} = 826163\cdot\tau/320819\cdot\tau =$ 2.6 times. In comparison with ordinary exponentiation on one processor, mixed use of simultaneous processing of 5 bits of the exponent and 4 processors for this example accelerates calculations in $4.5\cdot m^2\cdot\tau/t_{5,4} = 4718594\cdot\tau/320819\cdot\tau = 14.7$ times.

6. Conclusions

New research proposes a new method of accelerating exponentiation in Galois fields in cryptographic systems for information protection. In this research, the formulated polynomial of Galois field, which is a part of open key, is nearly constant, and allowing the use of precomputations. The proposed method allows reducing the calculating complexity of exponentiation in Galois fields by using simultaneous processing of some bits of code of the exponent. The method proved reducing the number of required operations in more than 5 times and, as a result, reducing the time for exponentiation.

The proposed method opens wide abilities for parallelizing the exponentiation in Galois fields on multiprocessor computer systems.

References

[1] Menezes A.J., Blake I.F., Gao S., Mullin R.C., Vanstone S.A., Yactobian T. Application of Finite Fields, N.Y. Knuwer Academic Published.-1993,387 p.

[2] Bardis N.G., Markovskyi O.P., Doukas N., Drigas F. Fast implementation zero knowledge identification schemes using the Galois Fields arithmetic, Proceeding of IX. International Symposium IEEE on Telecommunications - BIHTEL-2012, October 25-27, 2012, Sarajevo, Bosnia and Herzegovina. 978-1-4673-4876-8/12/$31.00 ©2012 IEEE.

[3] Bardis N., Doukas N. and Markovskyi O., Fast subscriber identification based on the zero knowledge principle for multimedia content distribution, International Journal of Multimedia Intelligence and Security.- 2010 - Vol.1, № 4. pp. 363 - 377.

[4] Soonhak Know, Chang Hoon Kim, Chun Pyo Hong. Efficient Exponentiation for Class of Finite Fields GF(2^n) Determined by Gauss Periods, Cryptographic Hardware and Embedded Systems ,CHES 2003, Proceeding 5-th International Workshop Cologne, Germany, Sept. 2003.- pp.228-242.

[5] Popovici E.M., Fitzpatrick P. Algorithm and Architecture for a Galois Field Multiplicative Arithmetic Processor, IEEE Transaction on Information theory. Vol. 49.- № 12,-2003.-pp.3303-3307.

[6] Paar C., Soria-Rodriguez P. Fast Arithmetic Architectures for Public-Key Algorithms over Galois Fields GF((2^n)^m) , Lecture Note in Computer Science - 1233: Advances of Cryptology - Eurocrypt-97.- pp.363-378.- Springer-Verlag.- Berlin.-1997.

[7] Ballet S. An improvement of the construction of the D.V. and G.V.Chudnovsky algorithm for multiplication in finite fields, Theoretical Computer Science. - vol.352.- pp.293-305.-2006.

[8] Wu H., Hasan M.A., Blake I.F., Gao S. Finite field multiplier using redundant representation, IEEE Trans. Computers, Vol.51, № 51,- 2002.- pp. 1306-1316.

[9] Peter de Rooij. Efficient exponentiation using precomputation and vector additional chains, Lecture Note in Computer Science - 950: Advances of Cryptology - Eurocrypt-94.- pp.389-399.- Springer-Verlag.-Berlin.-1994.

Received: September 1, 2013