UDC 004.056.5

Polina Buhaichenko,
Al-Mrayat Ghassan Abdel Jalil Halil

ONE APPROACH TO ORGANIZATION OF MODULAR EXPONENTIATION ON MULTI-CORE PROCESSORS

The article proposes an approach to the organization of the modular exponentation on multicore processors, which is based on the parallelization of the computational process due to the difference in the time of modular multiplication and modular elevation to the square. It is shown that the optimal number of cores that could be involved for the organization of parallel processing is two. The speed of modular exponentation increases by a 30%.

Keywords: modular exposure, Montgomery reduction, parallel calculations.

Fig.: 1. Bibl .: 8

Actual scientific researches and issues analysis. Virtually all public-key cryptographic algorithms are based on modular exposure operations on numbers whose bit rate (2048 or 4096) significantly exceeds the bit size of the processor. The classical method of breaking algorithms of this class consists in factorization of the module, ie its expansion into two prime numbers. The computational complexity of this problem is determined by the bit size of the numbers.

The emergence of cloud technologies has the effect of reducing the level of protection of algorithms. To restore the balance you need to increase the level of security, and the only way to achieve this for algorithms based on number theory is to increase the bit rate. However, for modular exposition, the computational complexity is cubic in nature, ie when the bit size doubles, the computational volume increases eightfold.

Therefore, the urgent task is to find ways to accelerate the calculation of the modular exponent.

Uninvestigated parts of general matters defining. The basic computational operation of a wide class of public key cryptographic algorithms is modular exponentation, which is performed on numbers whose bit size significantly exceeds the processor capacity. The classical algorithm of its execution has strictly consistent character and it cannot be parallelized. The only reserve for accelerating its execution is to reduce the execution time of its component - modular multiplication. It consists of multiplication and reduction, ie finding the remainder of the division of the product by the module.

One of the possible reserves for accelerating the computational implementation of modular multiplication is the combination in time of execution of several components of multiplication by reduction.

Analysis of recent research and publications. Both classical algorithms of modular exposition: from the senior and from the least significant digit of the code exponents have strictly consistent character and theoretically cannot be parallelized [3]. These algorithms consist of two basic operations - modular multiplication and modular squaring, and as is clear from the algorithms, the squaring accounts for 2/3 of the computational volume [4]. Barrett's technology [5], Montgomery technology [9] or precalculation technology [5] can be used for reduction both when multiplying and when squaring.

In paper [5] it was shown that in the absence of restrictions on memory resources on the critical path of the algorithm are only operations of modular elevation to the square. Therefore, a promising way to accelerate modular exposure through parallelization is to find approaches to reduce the implementation time of modular elevation to the square and the organization of the computing process on multiple processor cores with minimal synchronization and data exchange.

The research objective. The purpose of the work is to increase the speed of computational implementation of the basic operation of a wide range of cryptographic algorithms for information protection - modular exponentation due to the organization of parallel process processing on multi-core processors for cryptographic information protection systems.

The statement of basic materials. When implementing multiplication on an *r*-bit processor, *n*-bit numbers are divided into *k* fragments, where $k=n/r$. The number of processor multiplications of fragments is k^2. When calculating the modular square, the number of $K_к$ rocessor multiplications is determined by the formula:

$$K_к = \frac{k^2 + k}{2}. \quad (1)$$

Thus, due to the organization of modular squaring using two factors: savings on operations of multiplication of symmetric sections and the use of Montgomery group reduction using modulus-dependent preselections, the organization of modular squaring is proposed, which allowed to accelerate this operation four times. compared with modular multiplication.

This opens up the possibility for the computational process of modular exposure to be parallelized due to the fact that the basic procedures of this operation - modular

multiplication and modular squaring - have a significant difference in the time of computer implementation.

To implement this idea, a method of modular exposure on multicore processors has been developed.

The proposed method of implementing modular exposure on m processor cores is based on the idea that the n-bit code of the exponent $E = \{e_1,e_2,\ldots,e_n\}$, $\forall l \in \{1,2,\ldots,n\}$: $e_l \in \{0,1\}$ is forming m partial exponents ε_1, $\varepsilon_2,\ldots,\varepsilon_m$. Each i-th,, $i \in \{1,2,\ldots,m\}$, partial exponent ε_i contains a group of n_i adjacent bits of the exponent E at the same bit positions as the codes of the exponent E. All other bits of each of the partial exponents ε_1, $\varepsilon_2,\ldots,\varepsilon_m$ are equal to zero. It is obvious that $n_1+n_2+\ldots+n_m = n$. This means that subsets of groups of digits of exponent E in partial exponents do not intersect. In other words, the following condition is met:

$$\varepsilon_1 \cup \varepsilon_2 \cup \Lambda \cup \varepsilon_m = E \,. \tag{2}$$

For each partial exponents ε_1, $\varepsilon_2,\ldots,\varepsilon_m$ the initial numbers η_1, $\eta_2,\ldots,\eta_m$ are determined, starting from which the partial exponents contain a group of digits of the complete exponent E. In other words, the initial numbers η_1, $\eta_2,\ldots,\eta_m$ determine the number of high zeros in the codes of partial exponents. The numerical values of these numbers are determined by the formula:

$$\eta_1 = 0,\ \forall l \in \{2,3,\ldots,m\} : \eta_l = \sum_{h=1}^{l-1} n_h \,. \tag{3}$$

Technologically, as described, the partial exponents ε_1, $\varepsilon_2,\ldots,\varepsilon_m$ are formed when calculating the modular exponent as follows. Based on the condition of equal load of each of the processor cores, the values of n_1, $n_2,\ldots,n_m$ are determined - the number of significant binary digits in each of the partial exponents. It is obvious that the values of $n_1,n_2,\ldots,n_m$ do not depend on the number A, which is the operation of modular exposition, or on the code of the exponent E, nor on the module M, but only on the bit size n of the module and the number m of processor cores the modular exposure operation is performed. Therefore, the determination of the values of $n_1,n_2,\ldots,n_m$ can be done in advance. The results obtained can be used for hundreds of thousands of calculations of the modular exponent. According to the results of preliminary calculations of the values $n_1,n_2,\ldots,n_m$, the mask codes μ_1, $\mu_2,\ldots,\mu_m$ are formed. Each i-th, $i \in \{1,2,\ldots,m\}$, mask μ_i contains a group of ni adjacent units, starting with the ηi-th digit. All other bits of each mask μ_1, $\mu_2,\ldots,\mu_m$ are equal to zero. Thus,

we can say that each *i*-th mask with its single bits localizes a group of significant bits of the corresponding partial exponent.

Before the start of the calculations, the pre-formed masks and values of $n_1,n_2,\ldots,n_m$, as well as the values of $\eta_1, \eta_2,\ldots,\eta_m$ are transmitted to the processor cores. Direct calculation of the modular exponent A^E mod M begins with the fact that all processor cores are transmitted the same codes of the exponent E, the number A over which the exposition is performed, as well as the mask M. Subsequently, on each i-volume, $i \in \{1,2,\ldots,m\}$, the processor core performs operations in the following sequence:

1. The code of the partial exponent ε_i is calculated by the bit conjunction of the exponent code and the corresponding mask in the form: $\varepsilon_i = E \,\&\, \mu_i$.

2. The index j of the current bit of the exponent is set in η_i : $j = \eta_i$.

3. The counter with the number of operations is set to zero: c = 0.

4. The value of the current result R is set to one: R = 1.

5. If the value of the current j-th bit of the partial exponent ε_i is equal to one, then perform a modular multiplication: $R = R \cdot A \bmod M$.

6. Perform the operation of modular elevation to the square of the code of the current result: $R = R2 \bmod M$.

7. Perform the increment of the index j: $j = j + 1$ and the counter $c = c + 1$.

8. If the counter c is less than n_i : $c < n_i$, then proceed to re-execution of item 5

9. Perform the operation of modular elevation to the square of the code of the current result: $R=R^2$ mod M.

10. Perform the increment of the index j: $j = j + 1$ and the counter $c = c + 1$.

11. If the counter c is less than n: $c < n$, then go to the re-execution of paragraph 9, otherwise - the end.

An important element of the proposed method is the choice of values of n_1, $n_2,\ldots,n_m$ – the number of significant binary digits in each of the partial exponents based on the condition of equal load on each of the processor cores.

The maximum effect of acceleration of modular exposure is achieved with such a ratio between the number of bits of the exponent processed on the second and first processors:

$$\frac{n_2}{n_1} = 1 + 2 \cdot \gamma \,. \qquad (4)$$

The coefficient β of the acceleration of the calculation of the modular exponent is determined as follows:

$$\beta = \frac{t_0}{t_1} = \frac{n \cdot (\gamma + 0.5) \cdot t_{mm}}{(n \cdot \gamma + n_1 \cdot 0.5) \cdot t_{mm}} = \frac{\gamma + 0.5}{\gamma + 0.5 \cdot \frac{n_1}{n}} = \frac{\gamma + 0.5}{\gamma + \frac{1}{4 \cdot (1 + \gamma)}}. \quad (5)$$

In particular, if the same procedure is used for modular squaring and modular multiplication, then $t_{ms} = t_{mm}$, the value of γ=1 and, according to formula (5) the numerical value of acceleration β=1.33. This means that the use of two processors by the proposed method, accelerates the calculation of the modular exponent by 30%.

Conclusions. The method of parallelization of the modular exposition operation is theoretically substantiated, developed and investigated, characterized in that the exponent code is divided into a number of partial exponent codes containing fragments of the main exponent and zeros, due to which the calculation of modular partial exponents can be performed independently the final result is formed as a modular product of partial results, which allows to accelerate the computer implementation of modular exposure on multicore processors by parallelization.

References

1. Menezes Alfred, Handbook of Applied Cryptography. / Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone // CRC Press. – 2001. – 780 p.

2. Bardis N. Secure Implementation of Modular Exponentiation on Cloud Computing Resources / Bardis N., Markovskyi O. // Proceeding of International Conference Applied Mathematics, Computational Science and Systems Engineering. Athens, Greece, October 6-8, 2017. P.90-96.

3. Montgomery P. Modular multiplication without trial division. // Mathematics of Computation. – 44(170). – 1985. – P. 519–521.

4. 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.- P.44-49.

5. Brickell E.F. Fast exponentiation with precomputation / E.F. Brickell, D.M. Gordon, K.S. McCurlay, D.B. Wilson // Advances in cryptography –Proceeding of EUROCRYPT'12, LNCS-2059, Springer-Verlag. – 2012. – P. 200-207.

6. Boroujerdi N. Cloud Computing: Changing Cogitation about Computing / N. Boroujerdi, S. Nazem // IJCSI International Journal of Computer Science Issues. – Vol. 9. – Issue 4. – 2012. – №3. – PP. 169-180.

7. Kawamura S. A fast modular exponentiation algorithm / S. Kawamura, K. Takabayashi, A. Shimbo // IEEE Transaction on Information Theory. – Vol. 94. – № 6. – 2015. – P.2136-2142.

8. Xiaofeng Chen New Algorithms for Secure Outsourcing of Modular Exponentiations / Xiaofeng Chen, Jin Li, Jianfeng Ma3, Qiang Tang, Wenjing Lou // ESORICS 2012, LNCS 7459. – 2012. – PP. 541–556.

AURHORS

Buhaichenko Polina – student of Department of Computer Engineering, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”. E-mail: pbuhaichenko@gmail.com

Al-Mrayat Ghassan Abdel Jalil Halil – PhD student, Department of Computer Engineering, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”. E-mail: grayatjo@gmail.com