Modular Matrix Multiplication for Cryptographic Conversions

Sakhybay Tynymbayev[1], Sergiy Gnatyuk[2,3], Rat Berdibayev[1], Kaiyrbek Makulov[2], and Tetiana Okhrimenko[3]

[1] Almaty University of Power Engineering and Telecommunication, 126/1 Baytursynuly str., Almaty, 050013, Kazakhstan
[2] Yessenov University, micro district 32, 130000, Aktau, Kazakhstan
[3] National Aviation University, 1 Lybomyra Huzara ave., Kyiv, 03058, Ukraine

Abstract

Today, three types of encryptors are most widely used for data encryption: hardware, software/hardware, and software. Their main difference is not only in the way encryption is implemented and the degree of data protection reliability but also in the price, which often becomes a determining factor for users. While the price of hardware encryptors is much higher than that of software encryptors, the price difference is not comparable to a significant increase in the quality of information security. Hardware encryption has several strong advantages over software encryption, one of which is faster performance. Hardware implementation guarantees the integrity of the encryption process. At the same time, the generation and storage of keys, as well as encryption, is carried out in the encryption board itself, and not in the computer's operating memory. In this regard, the development of high-performance hardware processor operating units for asymmetric encryption, despite their high cost, is an urgent scientific and applied task. This paper analyzes modern approaches to modular multiplication, highlighting their strengths and weaknesses. An algorithm for multiplication with stepwise formation of partial and intermediate remainders is investigated, which, in turn, does not require preliminary calculations, and all calculations do not exceed the range of the module's bit grid. As a result, a synchronous matrix multiplier containing n blocks of AND circuits, $n – 1$ FPRs, and a single FIR with an intermediate remainder register has been developed, which will be useful for cryptographic transformations in systems with increased requirements for performance and information security (for example, in critical information infrastructure).

Keywords

Multiplier; information security; hardware encryption; public key cryptosystem; cryptography; cryptographic algorithm; encryption.

1. Introduction

The vast majority of modern cryptographic systems perform transformations with integers. Large integers (not necessarily primes) act as keys to perform cryptographic transforxmations. To achieve the desired level of security, depending on the cryptosystem, integers range from several hundred to several thousand bits. Over time, to maintain the desired level of security, the size should increase. To work with integers in binary representation, the following arithmetic operations are widely used: addition, subtraction, left and right shifts, multiplication, squaring, modular multiplication, and division.

Specifically, in asymmetric cryptosystems, the data encryption and decryption procedure are carried out by raising the number a to a degree x by modular P (a x mod$P$), which can be realized by software, hardware, and software and hardware means [1, 2].

Hardware encryption has several significant advantages over software encryption, one of which (and probably the most significant) is faster performance [3, 4]. The hardware implementation guarantees the integrity of the encryption process. In this case, the generation and storage of keys, as

CQPC-2023: Classic, Quantum, and Post-Quantum Cryptography, August 1, 2023, Kyiv, Ukraine
EMAIL: s.tynym@gmail.com (S. Tynymbayev); s.gnatyuk@nau.edu.ua (S. Gnatyuk); r.berdybaev@aues.kz (R. Berdibayev); kaiyrbek.makulov@yu.edu.kz (K. Makulov); t.okhrimenko@npp.nau.edu.ua (T. Okhrimenko)
ORCID: 0000-0002-9326-9476 (S. Tynymbayev); 0000-0003-4992-0564 (S. Gnatyuk); 0000-0002-8341-9645 (R. Berdibayev); 0000-0002-0826-0371 (K. Makulov); 0000-0001-9036-6556 (T. Okhrimenko)

well as encryption, is carried out directly in the encryption board itself, and not in the computer's operating memory.

Today, the development of high-speed operating units of hardware processors for asymmetric encryption, despite their high cost, is an urgent scientific and applied task [5–7].

Given the above, the *main goal of the paper* is the development of a modular matrix multiplier for cryptographic transformations.

2. Analysis of Modular Multiplication Approaches

The multiplication operation takes a leading place among the operations in rings and number fields, which form the basis for cryptographic transformations with public keys. At the same time, multiplication is a pretty time-consuming operation [8].

Modular multiplication can be done in two ways. In the first case, the operation is divided into two stages. At the first stage, *n*-bit numbers A and B are multiplied and form a 2*n*-bit number *C*. In the second stage, the product $C = A*B$ is reduced modulo *P*.

To date, a lot of experience has been accumulated in the development of high-speed integer multipliers and tools for squaring.

Among them are: Brown's multiplier, Wallace's multiplier, Dadd's multipliers, systolic and vedic multipliers, and quadratus, where the complexity of the calculation is O (n^2) bit operations. However, these multipliers are very effective in calculating "low-bit" numbers, which are widely used in the construction of processing units of all kinds of computers [9].

Today, the following integer multiplication methods are known and used in cryptography:

- Column multiplication.
- Karatsuba-Ofman.
- Toom-Cook.
- Schönhage-Strassen.
- Comba.
- Führer (development of the Schönhage-Straussen method).

In cryptography, the Karatsuba method [10], which has a complexity of $O(n^2)$ steps (bit operations), and the Toom-Cook algorithm [11] with a complexity of $O(n^{\log_2 3})$ bit operations are widely used to multiply multibit numbers, allowing to calculation of the required product faster than $O(n^2)$ steps (bit operations). The Schönhage-Strassen algorithm [12] allows multiplying two n-bit numbers in $O(n\log n, \log n)$ bit operations.

The modulo reduction operation performed in the second step is to obtain the rest of the result of dividing $C = A*B$ by the modulo *P*. Paper [13] analyzes various ways to reduce numbers by modular. It is shown that the most effective means of building is a modular drive device based on a dividing device. This device includes a partial remainder maker. High-performance matrix and conveyor modulo conversion devices can be easily implemented based on partial residue formers [8, 14–18].

High-performance matrix and conveyor devices for modular multiplication can be easily implemented based on partial remainder formers [14–18].

In the second method, modular multiplication is performed by using an algorithm for dividing large numbers. For example, the Barrett algorithm [19] requires preliminary calculations of the constant

$$\mu = \left\lfloor \frac{d^{2m}}{N} \right\rfloor$$

where $d = 2^k$, *k* is word size in bits, *m* is the number of words in the module *N*. The effectiveness of the Barrett algorithm depends entirely on how efficiently the preliminary calculations are performed by the distribution of large numbers.

Montgomery's algorithm requires the preliminary calculation of the constant $r^2(\mathrm{mod}N)$, using division with remainder [20, 21].

In the third method, the process of modular multiplication is performed in a large number of steps, where the number of steps is determined by the bit depth of the multiplier.

This paper describes the modular multiplication of numbers, where multiplication begins with the analysis of the lowest bits. In such a multiplier, the following steps are performed at each step of the multiplication:

1. Partial remainder FPRi calculates the partial remainder r_i for which the previous partial remainder $r_i - 1$, shifted by one digit toward the higher digit, by modulo *P*, i.e., $r_i=2_{ri-1}$modP. When forming the first partial remainder r_i, *A* is taken as the previous partial remainder, i.e., $r_i = A_1$.

2. The partial remainder r_i is logically multiplied by the ***i***-bit of the multiplier *B* by the logic circuit block I_i.

3. The intermediate residual R_i is calculated by adding up the partial remainder r_i with the previous intermediate remainder R_{i-1} by module *P*, that is $R_i = (r_i + R_{i-1})$ mod*P*.

After performing n multiplication steps, the result is formed $R - R_{n-1}=(r_{n-1} + R_{n-2})$ mod$P$.

In work [22], the algorithm for multiplying numbers by modulo is implemented according to an asynchronous matrix system that consists of n block schemes of I, $n-1$ FPR, and $n-1$ FIR (intermediate residual shaper). The disadvantage of this scheme is the complexity of its hardware implementation. To eliminate this disadvantage, this paper proposes a synchronous matrix multiplier developed by the authors, which contains n block schemes of I, $n-1$ FPR, and a single FIR with an intermediate remainder register. The operation of the multiplier is synchronized using a level divider.

3. Developed Modular Matrix Multiplier

The functional scheme of a synchronous multiplier with a matrix structure is shown in Fig. 1. The multiplier consists of the n-bit multiplier register RgB, the multiple registers RgA, and the module register RgP, the synchronization block SYNCHRO unit that generates level signals for the block of schemes $I_0 \div I_{n-1}$. At the entrance SYNCHRO unit is supplied to the START signal, clock signals CLOCK and a binary code to the signal for the number of multiplier digits n. Following the START signal, the RECEPTION signal vibrates, after which the discharge of multiplier B is received into the RgP register through circuit block 3'.

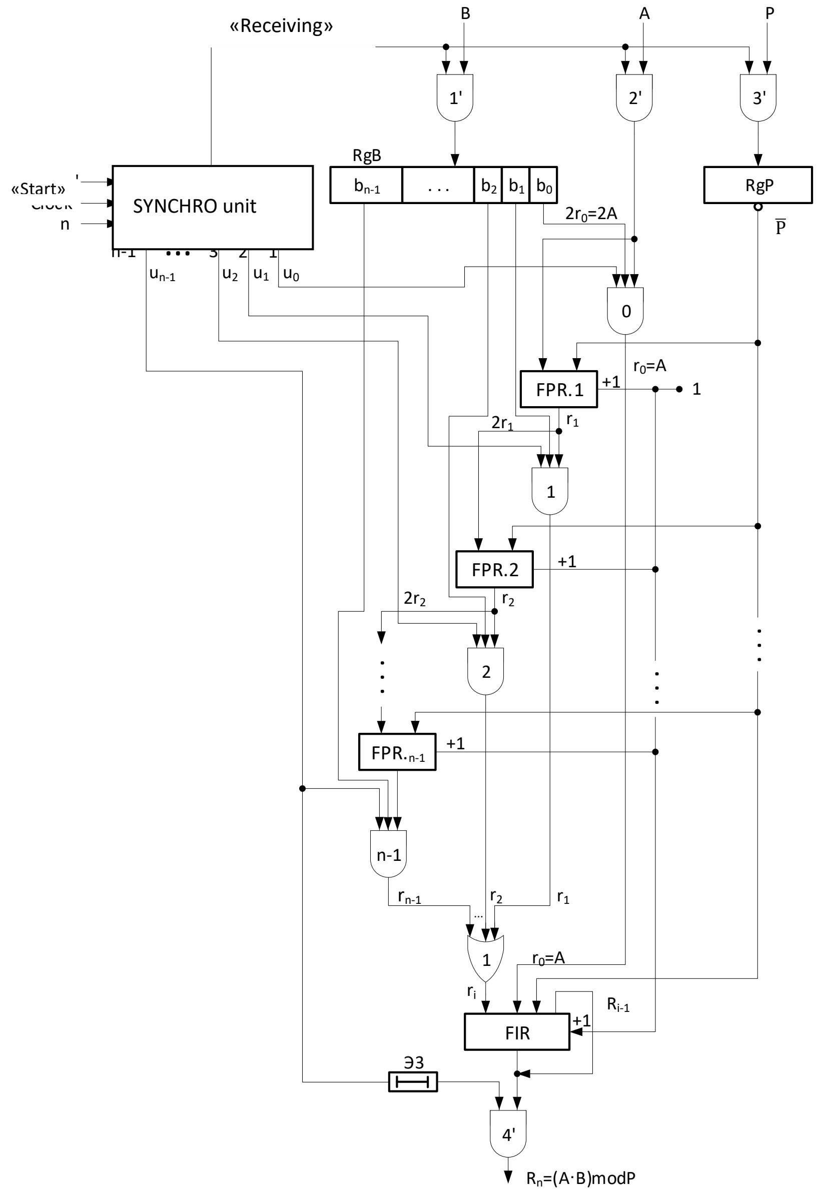

Figure 1: Functional scheme of the modular matrix multiplier

The SYNCHRO unit consists of a binary counter and a decoder. The state of the counter is decoded on their outputs generating signal levels for $I_0 \div I_{n-1}$. With the START signal, the binary code of the number n is also written to the binary counter and permits the clock signal Clock to pass to the counter input.

The multiplier also includes partial remainder generators $FPR_{.1} \div FPR_{.n-1}$ and circuits $I_0 \div I_{n-1}$, circuit OR_1, and intermediate remainder generator FIR with intermediate remainder register.

The results output of multiplication $I_{4'}$ after receiving the clock signal u_{n-1} from the SYNCHRO unit through the delay element EZ.

The value of the modulus $\bar{P}$ from the inverted output RgP is delivered to the inputs of all $FPR_{.1} \div FPR_{.n-1}$ and FIR. On the RECEIVE signal, the multiplier A is delivered to the inputs of the $I_{2'}$ circuit block with a shift of one bit toward the high bit to the $FPR_{.1}$ input. The other inputs of the I_0 block receive the value of the lowest bit of the RgB – b_0 register and the control level u_0 from the SYNCHRO unit output.

The output of I_0 is connected to the FIR register. The output $FPR_{.1}$ is connected to the input of the circuit block I_1. The other inputs of the I_1 are connected to the outputs of the RgB and SYNCHRO unit, through which the value of the bit b_1 and the control level u_1 are received. The output of the I_1 block is connected to the input of the OR_1.

The code value from the output of $FPR_{.1}$ with a shift of one bit toward the lowest bit is applied to the input of $FPR_{.2}$. In turn, the output of $FPR_{.2}$. is connected to the inputs of the I_2 circuit block, to the other inputs of which the value of the bit b_2 from the RgB register and the control level u_2 from SYNCHRO unit are supplied. The output of the I_2 block is connected to the input of the OR_1 scheme. There are similar connections between $FPR_{.3} \div FPR_{.n-2}$ and blocks of schemes $I_3 \div I_{n-2}$.

The $FPR_{.n-1}$ inputs receive the value of the code from the $FPR_{.n-2}$ output with a shift of one bit toward the lowest bit and the code of the P module from the RgP outputs. The output of $FPR_{.n-1}$ is connected to the input of the I_{n-1} circuit block, which also receives the value of the high bit b_{n-1} of the RgB register and the control signal u_{n-1} from the SYNCHRO unit. The output of the I_{n-1} circuit block is connected to the OR_1 input. Fig. 2 shows the structure of the FPR, which is used to form a partial remainder r_i from the doubled previous remainder by modulo P: $r_i = 2_{ri-1} \mathrm{mod} P$.

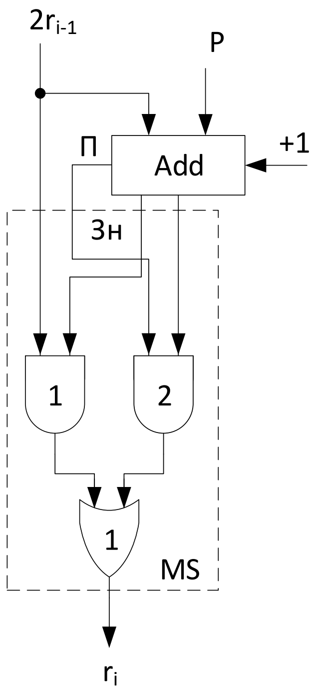

Figure 2: FPR structure

FPR consists of a binary adder Add and multiplexer MS, which contains blocks of schemes I_1, I_2, and scheme OR_1.

The doubled partial remainder $2_{ri-1,}$ module reverse code $\bar{P}$, and a single signal +1 is delivered to the adder inputs. As a result of performing the operation $r_i = 2_{ri-1} \mathrm{mod}\bar{P} + 1$, a difference with its sign ZN is formed at the output of the adder.

If ZN = 1, the following code 2_{ri-1} (2_{ri-1}<P) is transmitted to the FRP outputs. At the same time, transferring from the digit sign P = 0. If ZN = 0, then the result of subtraction $2_{ri-1} - P$ ($2_{ri-1} \geq P$) is transferred to the FPR outputs.

Fig. 3 shows the structure of the FIR intermediate remainder generator, which includes the Add adder, FPR, OR_2 circuits, and the RgR intermediate remainder register. The output of the RgR register is connected to the Add input, where the R_{i-1} value is transferred. It is easy to see that the circuit operates $R_i = (r_i + R_{i-1}) \mathrm{mod} P$.

Let's consider the operation of a modular matrix multiplier. After receiving the operands A, B, and P into the corresponding registers and the binary code, the number of bits multiplier in the SYNCHRO unit is the first clock impulse Clock 1 arrives and code 1 is written to the binary counter. At the same time, output 1 of the SYNCHRO unit produces a high level u_0, which is supplied to the input of the circuit block I_0.

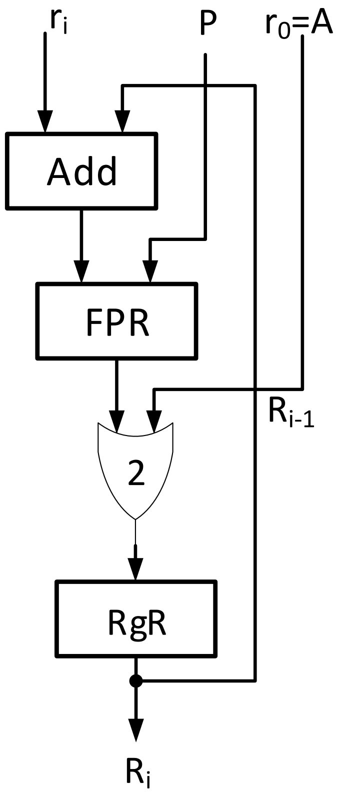

Figure 3: FIR structure

The other inputs of I_0 are supplied with the lowest bit of the multiplier b_0 and the bits of the multiplier A. With $b_0 = 1$, a partial remainder $r_0 = R_0$ is generated at the output of the I_0 block, which is written to RgR FIR. After that, the clock signal Clock 2 is delivered to the SYNCHRO unit, and the control level u_2 is generated at the output of the SYNCHRO unit, which is sent to the input of I_2. The other inputs of I_2 are supplied with the value of the bit b_2 of the RgB register and the value r_2 from the output of $FPR_{.2}$. The outputs of the I_2 circuit block are sent to the input of the OR_1 scheme.

Partial remainders $r_3 \div r_n$ are formed in a similar way, which also through schemes OR_1 are supplied to the entrance FIR. FIR, receiving partial remainder r_i, creates R_i according to the formula $R_i = (r_i + R_{i-1})\text{mod}P$.

Table 1 shows an example of performing a modular multiplication operation in a synchronous matrix multiplier, where A = 2710; B = 2310 = 101112; P = 3510. For convenience, all arithmetic operations are performed in the decimal system.

Verification:

R = (27×23) mod 35 = 621 mod 35 = 2610

Table 1
The order of multiplying numbers by modulo

	u_0	u_1	u_2	u_3	u_4
r_i	$r_0 = A*b_0 =$ $= 27_{10}$	$r_1 = 2r_0\text{mod}P =$ $= 54 - 35 = 19_{10}$	$r_2 = 2r_1\text{mod}P =$ $= 38 - 35 = 3_{10}$	$r_3 = 2r_2\text{mod}P =$ $= 6\text{mod}35 = 6_{10}$	$r_4 = 2r_3\text{mod}P =$ $= 12\text{mod}35 = 12_{10}$
RgR	$R_0 = r_0 =$ $= A = 27_{10}$	$R_1 =$ $= (r_1b_1 + R_0)\text{mod}P =$ $= (19+27)\text{mod}35 =$ $= 11_{10}$	$R_2 =$ $= (r_2b_2 + R_1)\text{mod}P =$ $= (3 + 11)\text{mod}35 =$ $= 14_{10}$	$R_3 =$ $= (r_3b_3 + R_2)\text{mod}P =$ $= (6*0 + 14)\text{mod}35 =$ $= 14_{10}$	$R_4 =$ $= (r_4b_4 + R_3)\text{mod}P =$ $= (12 + 14)\text{mod}35 =$ $= 26_{10}$

5. Conclusions

This paper analyzes modern approaches to modular multiplication and highlights their strengths and weaknesses. A multiplication algorithm with step-by-step formation of partial and intermediate remainders was studied, which, in turn, does not require preliminary calculations, and all calculations do not go beyond the range of the bit grid of the module. As a result, a synchronous matrix multiplier has been developed that contains n blocks of circuits I, $n - 1$ FPR, and a single FIR with an intermediate remainder register. The obtained results will be useful for cryptographic transformations in systems with increased requirements [23] for performance and information security (for example, in critical information infrastructure).

References

[1] S. Tynymbayev, et al., Development of Pipelined Polynomial Multiplier Modulo Irreducible Polynomials for Cryptosystems, Eastern-European Journal of Enterprise Technologies 1(4-115) (2022) 37–43.

[2] E. Aitkhozhaeva, S. Tynymbaev, Aspects of Hardware Modulo Conversion in Asymmetric Cryptography, Bulletin of the National Academy of Sciences of Kazakhstan 5 (2014) 88–93.

[3] Y. Sadykov, et al., Technology of Location Hiding by Spoofing the Mobile Operator IP Address, in: IEEE International Conference on Information and Telecommun. Technologies and Radio Electronics (2021) 22–25. doi: 10.1109/UkrMiCo52950.2021.9716700

[4] A. Carlsson, et al., Sustainability Research of the Secure Wireless Communication System with Channel Reservation, in: 2020 IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (2020). doi:10.1109/tcset49122.2020.235583

[5] I. Kuzminykh, et al., Investigation of the IoT Device Lifetime with Secure Data Transmission, Internet of Things, Smart Spaces, and Next Generation Networks and Systems, vol. 11660 (2019) 16–27. doi: 10.1007/978-3-030-30859-9_2

[6] M. Vladymyrenko, et al., Analysis of Implementation Results of the Distributed Access Control System. in: 2019 IEEE International Scientific-Practical Conference Problems of Infocommun., Sci. and Technology (2019). doi: 10.1109/picst47496.2019.9061376

[7] V. Sokolov, P. Skladannyi, H. Hulak, Stability Verification of Self-Organized Wireless Networks with Block Encryption, in: 5th International Workshop on Computer Modeling and Intelligent Systems, vol. 3137 (2022) 227–237.

[8] A. Okhrimenko, V. Kovtun, Experimental Research of the Developed Methods of Arithmetic Operations in Cryptographic Transformations According to the ECDSA Scheme, in Cyber Hygiene 2654 (2019) 827–837.

[9] P. G. Comba, Exponentiation Cryptosystems on the IBM PC, IBM Systems Journal 29(4) (1990) 526–538.

[10] A. Karatsuba, Y. Ofman, Multiplication of Many-Digital Numbers by Automatic Computers, DAN USSR 145 (1962) 293–314.

[11] S. A. Cook, S. O. Aanderaa, On the Minimum Computation Time of Functions, Trans. AMS 142 (1969) 291–314.

[12] A. Schönhage, W. Strassen, Fast Multiplication of Large Numbers, Cybernetic Compendium 2 (1973) 87–98.

[13] R. Brumnik, et al., Techniques for Performance Improvement of Integer Multiplication in Cryptographic Applications, Mathematical Problems in Engineering (2014) 1–7.

[14] V. Petrenko, A. Chipiga, Combination Recurrent Residue Generator, Patent 2029435, MPK N03M7/18, no. 5032302/24 (1995).

[15] V. Petrenko, A. Sidorchuk, Y. Kuzminov, Device for Forming Residues according to an Arbitrary Modulus, Patent 2368942, MPK N03M7/18, no. 02101066858/08, Bulletin. No. 21 (2009).

[16] S. Tynymbayev, Y. Aitkhozhayeva, S. Adilbekkyzy, High-Speed Device for Modular Reduction, Bulletin of National Academy of Sciences of the Republic of Kazakhstan 6(376) (2018) 147–152.

[17] S. Tynymbaev, E. Aitkhozhaeva, Residue Generator using an Arbitrary Modulus, Patent of the Republic of Kazakhstan, no. 30983 (2016).

[18] S. Tynymbaev, et al., High-Speed Devices for Reducing Numbers Modulo, in 4th Int. Asian School Seminar “Problems of optimization of complex systems,” part 2 (2018) 273–279.

[19] P. Berrett, Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor (1987). doi: 10.1007/3-540-47721-7_24

[20] P. L. Montgomery, Modular Multiplication without Trial Division, Math. Compulation 44(170) (1985) 519–521. doi: 10.20307/2007970.

[21] P. Eran, T. M. Henige, Method and Apparatus for Efficient Modulo Multiplication, Patent US no. 8Y17756B2 (2013).

[22] B. Kaliski, Moore’s Law, in van Tilborg H.C.A., Jajodia S. (eds) Encyclopedia of Cryptography and Security (2011).

[23] B. Kuzma, et al., Fast Matrix Multiplication viaCompiler-only Layered Data Reorganizationand Intrinsic Lowering. doi: 10.1002/spe.3214