computing@computingonline.net
www.computingonline.net

ISSN 1727-6209
International Journal of Computing

CALCULATION OF GF (P) ELLIPTIC CURVES IN FPGA

Marek Aleksander [1], Mykola Karpinskyy [2], Grzegorz Litawa [1]

[1] State Higher Vocational School in Nowy Sacz, Poland,
aleksmar@pwsz-ns.edu.pl, http://www.pwsz-ns.edu.pl/it/
[2] University of Bielsko-Biala, Poland,
mkarpinski@ath.bielsko.pl, http://www.keia.ath.bielsko.pl/index.php?poz=14

Abstract: *The paper describes a hardware system carrying out point summation on elliptic curves. The implementation of basic function, which is modulo multiplication of huge integrals, is based on Krestenson's basis. Such a summing unit has been used for hardware implementation of Pollard rho-algorithm. The paper also presents the performance of the mentioned unit.*

Keywords: *Elliptic curve, rho-Pollard, Rademacher-Krestenson's bases, FPGA, GF(p), ECDLP.*

1. INTRODUCTION

The need for keeping stored, processed and distributed data secret provoked a growing interest in cryptographic and identification techniques. An essential issue is therefore the safety level provided by the cryptographic systems.

This article concerns gauging of elliptic curve (EC) based cryptosystems safety. Elliptic curves over finite field GF(2m) and p, GF(p) are commonly used for cryptography purposes. Recently it has become convenient to use hardware FPGA units (Field Programmable Gate Array) for extremely fast operations carried out on elliptic curves GF(2m) according to the nature of the calculations. Information concerning this issue may be found in e.g. [1]. The aim of our work was to device a hardware unit carrying out operations on elliptic curves over finite field GF(p). In order to accomplish our goal we have employed an unconventional approach: a technique relying on Rademacher's-Krestenson's basis allows multiplication operations to be converted into summing ones and taking advantage of previously generated tables. Such operations are carried out by hardware systems and offer a substantial simplification of calculations.

2. ELLIPTIC CURVES AND CORRESPONDING ARITHMETIC

Integrals of expressions in the form of:

$$\int R(x,\sqrt{ax^3+bx^2+cx+d})dx, \tag{1}$$

are called elliptic integrals. The name has been derived from an ellipse length calculation. Being more specific about elliptic integrals it should be stressed that the only integrals of that type are ones which may not be calculated in a finite form [2, 3]. A curve circumscribed by the following equation is a good example of an elliptic curve:

$$y^2 = 4x^3 - cx - d \tag{2}$$

with a point in infinity $\mathcal{O}$

A closer look at the most important grouping operation which is the point summation on an elliptic curve reveals an issue: assume that points P and Q belong to curve E and have respective coordinates $P(x_1, y_1)$ and $Q(x_2, y_2)$. Assume that point $P \neq Q$ and $P,Q \neq \mathcal{O}$. Sum of points $P + Q$ will be defined as point $R(x_3, y_3)$ such that a straight line drawn through points P and Q intersects with the curve in point X = –*R*, consequently being a opposite of *R*. Adding a point to itself (doubling) is clearly described as $X = P + P$. Point coordinates equation takes a form of $R = 2P = P + P$.

Taking advantage of this equation it is simple to define multiplication of a point by an integer. If τ is an integer then $\tau P = P + P + ... + P$, where the number of added points equals τ.

At the ending of the theoretical discussion consider some basic equations where subtraction is defined as addition of a negative point $Q - P = Q + (-P)$, and multiplication by a negative integer $-\tau$ as $\tau P = -(\tau P)$.

Further discussion will focus on elliptic curves over finite field *GF*(2^m) and *GF*(*p*) where *p* is a large prime number. Elliptic curves over finite field are characterized by a finite number of rational points belonging to the curve.

Contemporary cryptography relies on elliptic curves over finite field *GF*(2^m) and *GF*(*p*). Therefore these cases will be addressed further in the paper [2, 3, 4, 5, 6]. An elliptic curve over finite field *GF(p)* is a modulo *p* reduction of curve *E*.

Let curve *E* be defined by the equation

$$y^2 = x^3 + ax + b, \qquad (3)$$

then the elliptic curve [4] over finite field *GF*(*p*) will be described as:

$$y^2 \bmod p = g(x) \bmod p. \qquad (4)$$

The modulo *p* reduction of curve *E* coordinates has led to a presentation of the coordinates in field *GF(p)*.

3. POLLARD RHO PARALLEL METHOD OF FINDING DISCRETE LOGARITHM

The concept of discrete logarithm. The crucial issue with reference to ECC (Elliptic Curve Cryptography) security is the Elliptic Curve Discrete Logarithm Problem (ECDLP) defined in the following way: Let *E* represent an elliptic curve defined over a finite field, point *P* of order *m* and point *Q* being a multiple of *P*. The found integer $l \in <0, m\text{-}1>$ such that $Q=l \cdot P$ [2, 7] is denoted as the discrete logarithm of *Q* to the base *P*. There are numerous approaches to the problem of the elliptic curve discrete logarithm one of which is the Pollard's rho Algorithm and its parallel version.

Pollard has invented several methods of calculating discrete logarithms in various groups. Pollard rho Method [1, 2] applies a singular random walk path until it falls into a cycle. Pollard rho Method complexity is of order $\sqrt{\pi n/2}$, having the possibility of limiting the demand for memory it is currently the best known method. In order to speed up the calculations a modified method is used. It is called Parallel Method as it uses many random walk paths realized simultaneously by many processors. At present it is the fastest way to attack an elliptic curve. In Pollard rho Parallel Method we seek an intersection of two paths rather than await the cycle completion. The use of multiple random walk paths results in a linear increase of efficiency. On average it takes as long as [7] $\sqrt{\pi n/2} + c$ to find a point, the expected value of memory demand equals $\sqrt{\pi n/2}\,/2^K$ which may be lowered through altering *K*. Lowering memory demand, however, means additional 2^K point computations ($c = 2^K$) done by the machine.

4. THEORETICAL BASIS FOR IMPLEMENTATION OF FPGA SYSTEM CARRYING OUT OPERATIONS ON ELLIPTIC CURVES GF(P)

The use of reprogrammable structures with implemented Pollard rho parallel algorithm is intended to increase effectiveness of calculations carried out on elliptic curves over finite field of higher orders *GF*(*p*). Coordinates calculation for consecutive points on elliptic curves forces multiplications of large modulo numbers. In such a case traditional methods of multiplication fail to be satisfactory even with the use of dedicated blocks. One solution to this problem is implementation of Rademacher-Krestenson's basis thanks to which modulo multiplication may be totally eliminated and replaced by adding operations based on previously generated tables [8].

Assume a multiplication of two numbers *x* and *y* modulo a number *p*.

A system of congruence classes (Krestenson's BTL) applied to multiplication operations allows to present the result of a multiplication in a matrix form. Therefore x and y must be presented as:

$$x = x_{r-1}2^{r-1} + x_{r-2}2^{r-2} + x_i 2^i + \ldots + x_1 2^1 + x_0 2^0\,, \qquad (5)$$

$$y = y_{r-1}2^{r-1} + y_{r-2}2^{y-2} + y_j 2^j + \ldots + y_1 2^1 + y_0 2^0\,, \qquad (6)$$

where *r* – positioning of *x* i *y* and x_i, y_j = 0,1. There is a matrix shown in table 1 which represents a product of *x* and *y* relative to module *p* where $m_{ij} = 2^{i+j} \bmod p$.

Table 1. Determining the transformation matrix module based on Rademacher–Krestenson.

	y_{r-1}	..	y_j	…	y_1	y_0
x_{r-1}	$m_{r-1\ r-1}$	..	$m_{r-1\ j}$	…	$m_{r-1\ 1}$	$m_{r-1\ 0}$
…	…	..	…	…	…	…
x_i	$m_{i\ r-1}$	..	m_{ij}	…	m_{i1}	m_{i0}
…	…	..	…	…	…	…
x_1	$m_{1\ r-1}$	..	m_{1j}	…	m_{11}	m_{10}
x_0	$m_{0\ r-1}$	..	m_{0j}	…	m_{01}	m_{00}

Product of numbers *x* and *y* is obtained from the following equation:

$$(x \cdot y) \bmod p = \left(\sum_{s,k=1}^{r-1} m_{sk} \right) \bmod p, \qquad (7)$$

where $x_s, y_k = 1$. Logically meaning that m_{sk} lies at the intersection of column and row for which corresponding x_i and y_j equal 1.

Employing Krestenson's basis for large number calculations allows efficient multiplication operations not only program-controlled but also hardware-controlled large number calculations. Obviously the above described algorithm allows to replace multiplication operations characterized by square complexity of calculation with summing operations which in contrast are of linear complexity of calculations. Such an implementation allows highly efficient calculations on elliptic curves over finite field of higher orders.

Hardware-based systems or parallel systems are a perfect opportunity to utilize algorithms capable of dividing any large numbers into segments operated directly by a processor, or into segments of size suitable for hardware structures for processing them. Special algorithms have been devised in order to represent any number in a form of a list

$$A=(a_1, a_2, \ldots, a_n), \qquad (8)$$

where each of n cells a_i includes a precisely sized fragment of a large number. This approach allows representation of large natural numbers in binary system or as digits sequences housed in cells a_i. In this way each cell may be simultaneously operated by independent processes. of others by parallel processes [9].

Assume extremely large integers X, Y, Z. Divide them into binary words of specified length m in base δ, where $\delta = p^m$:

$$X = x_n\delta^n + x_{n-1}\delta^{n-1} + \ldots + x_1\delta + x_0, \qquad (9)$$

$$Y = y_r\delta^r + y_{r-1}\delta^{r-1} + \ldots + y_1\delta + y_0, \qquad (10)$$

$$Z = z_k\delta^k + z_{k-1}\delta^{k-1} + \ldots + z_1\delta + z_0, \qquad (11)$$

and all coefficients $x_\alpha, y_\beta, z\lambda \in [0, \delta)$.

In further discussion numbers X, Y, Z are considered regardless of sign assuming $Y \le X$. In the case of addition

$$X + Y = Z, r \le k \le n+1. \qquad (12)$$

In a very simple way the above equation may be adapted to multiple number additions, minding an accurate estimation of number of cells k which may be occupied while summing a set of numbers.

In the case of subtraction the number of occupied cells is presented by

$$X - Y = Z, \quad 0 \le k \le n. \qquad (13)$$

Having a large number fragmented into excerpts in the form of lists it is possible to apply parallel summing and subtraction processes carried out by algorithms which will be explained at this point. Each of the processes initiates a double word, each component a_i is stored in a low-order word and component b_i – in a high-order word assuming that for every $i > r$ $b_i = 0$.

Summing algorithm for each of the processes looks as follows:

1. Summing i-th component $x_i + y_i$
2. Value of $\vartheta_i = (x_i + y_i) \bmod \delta$ gets written in the low-order word of adequate device
3. Position $i+1$ in the high-order word is replaced by $\omega_{i+1} = (x_i + y_i)\, div\, \delta$
4. Analysis of the high-order word of i-th position allows:
   a) algorithm termination in the case of lack of transfer
   b) return to point 1 in the case of transfer occurrence.

It turns out clear that the result Z is obtained after at most $n+1$ steps.

Subtracting algorithm for each of the processes looks as follows:

1. Subtracting i-th component $x_i - y_i$
2. A value ϑ_i is calculated according to the equation beneath and gets written in the low-order word of adequate device

$$\vartheta_i = \begin{cases} \delta + x_i - y_i, if\ x_i \prec y_i \\ x_i - y_i, if\ x_i \ge y_i \end{cases}$$

3. Position $i+1$ in the high-order word is replaced by $\omega_i = \begin{cases} 1, if\ x_i \prec y_i \\ 0, if\ x_i \ge y_i \end{cases}$
4. Analysis of the high-order word of i-th position allows:
   a) algorithm termination in the case of lack of transfer
   b) return to point 1 in the case of transfer occurrence.

Similarly to summing algorithm the result Z is obtained after at most n+1 steps.

Due to the fact that calculating reverse numbers in field are realized at vast expense the summation process is carried out in a mixed representation. For two reasons the mixed representation is favored: it does not require calculating reverse numbers in field, which is necessary in affine representation and allows reduction of multiplication operations from

16 in projective representation to 11. Table 2 shows calculations that need to be carried out in order to add two points: P1 expressed in affine coordinates and P2 expressed in projective coordinates. The result being obviously expressed in projective coordinates.

Table 2. Mixed summation point

$\lambda_1 = X_1 Z_2^2$	$\lambda_8 = \lambda_4 + Y_2$
$\lambda_3 = \lambda_1 - X_2$	$Z_3 = Z_2 - \lambda_3$
$\lambda_4 = Y_1 Z_2^3$	$X_3 = Z_6^2 - \lambda_7 + Z_3^2$
$\lambda_6 = \lambda_4 - Y_2$	$\lambda_9 = \lambda_7 + \lambda_3^2 - 2X_3$
$\lambda_7 = \lambda_1 + X_2$	$Y_3 = \left(\lambda_9\lambda_6 + \lambda_8\lambda_3^3\right)/2$

5. FPGA UNIT ARCHITECTURE

The construction of a hardware unit starts off with a creation of Krestenson matrix for numbers of definite length and a given module outside the system. In FPGA system the matrix is stored in ROM (*Read-Only Memory*) in such a form that is adequate to the size of a number or the number of integer words.

The multiplying unit is fed with numbers as binary vectors and produces results of the same type. As far as the structure of a multiplying unit is concerned there are some basic elements to be mentioned:

- Components responsible for picking Krestenson matrix from ROM and placing it in a 3-dimentional table inside the system.
- Components summing the matrix rows and realizing modulo operations in precisely the same way as it was described for the summing unit. All rows are summed in a parallel manner. As soon as the summing has finished the modulo operation is carried out in much the same way as in the case of the summing unit with the help of a parallel subtraction as the sum is smaller than a doubled module.
- Components responsible for summing the results of the prior operation, modulo operations done analogically to the above, outputting the result.

A multiplying unit structure is shown schematically in fig. 1.

In order to increase the pace of VHDL (*Very High Speed Integrated Circuits Hardware Description Language*) code creation, which is the native code of the unit, a special function has been written in C++ language allowing an automatic unit code generation in VHDL device description language.

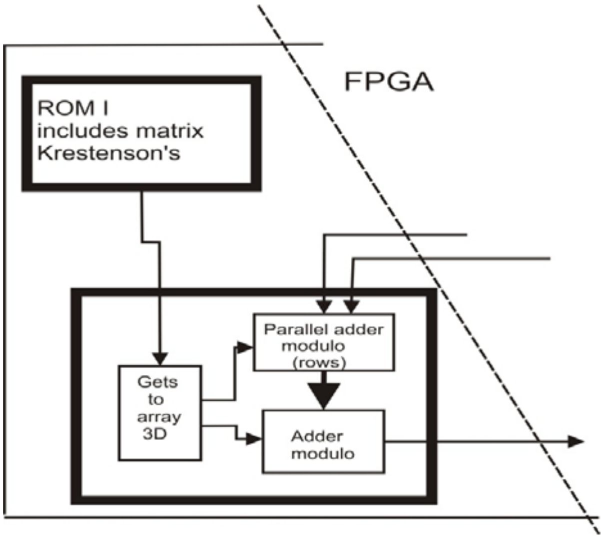


Fig. 1 – Multiplying unit

Having described the multiplying unit it is reasonable to move on to the nest unit which is responsible for Pollard rho algorithm realization. Structure of that unit is schematically shown in fig. 2.

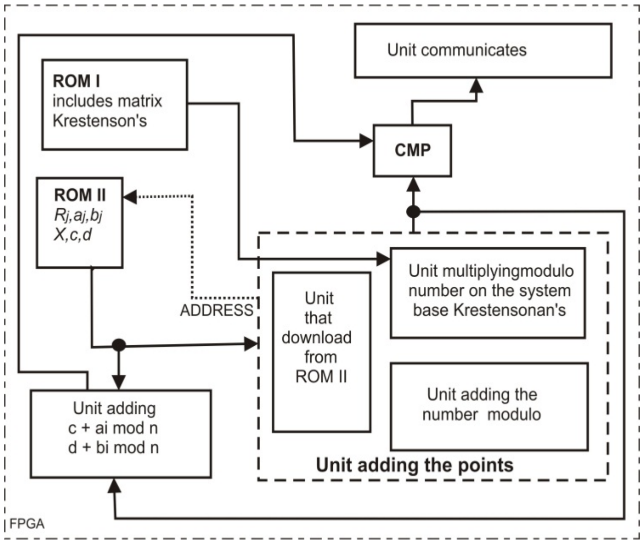


Fig. 2 – Realizing unit Pollard rho algorithm

As the schematic shows only marked points are sent over to the communication unit. Unit from figure 2 carries out a single random walk path. As it is shown in picture 2 the hardware unit's job is to realize a part of the algorithm, that is summing of points one of which is stored in R_j matrix and the other calculated by the system in a prior iteration or is the starting point of a random walk according to the rho Pollard algorithm.

Memory ROM II of R_i points basis and a_i b_i coordinates is generated accordingly to parameters of a curve being processed. ROM II memory is stored in VHDL language however generated by an exclusively dedicated program in C++ due to its size and necessity of adapting the content to calculations on various curves. Analogically are calculated coordinates of the starting point and the system structure adapted to various curve sizes. The other part of rho Pollard algorithm is carried out by PC run dedicated programs responsible for reading information in the from of distinctive points from FPGA system (practically from multiple such systems realizing independent random walks). This remaining part of the algorithm is also responsible for writing the information in a base and cross-comparing. Generally, a standard integer data type available in VHDL have been used in FPGA run summation processes. As it is natural for this sort of calculations to operate on huge numbers they are divided into words of integer type and subsequently added parallel as explained above.

6. RESULTS

Implementation of a unit presented in fig. 2 was carried out in Startix III Altery system. Summary of the system performance for various curve lengths are show in tab. 3.

Table 3. Results for various curve lengths.

GF(*p*)	iterations/second
72	349650
96	255681
120	207070
168	157024

The efficiency of the constructed unit is expressed as number of iterations per second, namely number of summations done by one unit in one second – shown in tab. 3. As it is known, an average number of iterations needed for working out discrete logarithm equals [1, 2, 7]

$$\sqrt{\pi n/2}\Big/D \qquad (12)$$

where D is number of units Thus it is possible to estimate time needed for finding discrete logarithm.

Let's compare performance of a single FPGA system containing a single processor responsible for realizing one random walk utilizing Krestenson's basis with performance of a Pentium4 processor which operates on huge numbers using one of the commonly available libraries. An FPGA system realizing calculations on curve *GF*(96) yields 256 000 summations while Pentium4 73 000 iterations per second.

7. CONCLUSION

Utilizing Krestenson's basis in FPGA obtain increase rate calculations on the elliptic curves *GF*(*p*). Following the presented idea, a code realizing single random walk may be used, after some modification, in an FPGA cluster containing a greater number of systems working parallel. This would linearly increase the efficiency of rho Pollard algorithm realization.

8. REFERENCES

[1] Majkowski P., Wojciechowski T., Wojdynski M., Rawski M. Implementation of module for cryptanalysis of elliptic curve ciphers in reprogrammable structures. *Measurements, Automation and Monitoring*, (53) 7 (2007). pp. 24-26. (in Polish)

[2] Blade I., Seroussi G., Smart N. *Elliptic curves in cryptography*. TAO, Warsaw, 2004. (in Polish)

[3] Fichtenholz G.M. *Differential and integral calculation*. Vol. 2, Polish Scientific Publishers PWN, Warsaw 2011. – 696 p. (in Polish)

[4] Chocianowicz W.: *Cryptology and advanced methods of cryptography* (http://uznam.net.pl/~blondasek/iuz/materialy/Krypto-2007-2008-2MUDZ.pdf) (in Polish)

[5] IEEE PI363. *Standard Specyfications for Public Key, Cryptography*. Draft 13. 1999.

[6] Hankerson D., Menezes A., Vanstone S. *Guide to elliptic curve cryptography*. Springer, NY 2004. – 332 p. (http://math.boisestate.edu/~liljanab/Crypto2 Spring10/GuideToECC.pdf)

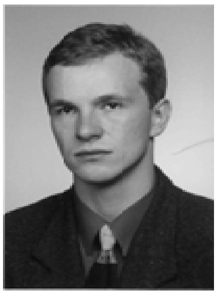
[7] Bulens P., Meurice G., Quisquater J.-J. Collision Search for Elliptic Curve Discrete Logarithm over GF(2^m) with FPGA, 2007 (http://www.springerlink.com/content/p274683181434n28/)

[8] Yakymenko I., Kasyanchuk M., Nykolajchuk Y. Matrix algorithms of processing of the information flow in computer systems based on theoretical and numerical Krestenson's basis, TCSET'2010, February 23-27, 2010, Lviv-Slavske, Ukraine. – P. 241.

[9] Makoha A.H., Zuj B.U. *Arithmetic of very large integers in parallel computer systems*, 20.03.2007 (http://revolution.allbest.ru/mathematics/00011260_0.html) (in Russian)

Marek Aleksander *graduated from the AGH University of Science and Technology, Cracow, Poland, in 2000. He received the Ph.D. degree from the Military Technical Academy, Warsaw, in 2004. He has been working in the State Higher Vocational School in Nowy Sacz, Poland, since 2001. Currently he is a director of Institute of Engineering. Author and co-author of over 40 publications. His main areas of interest are cryptology, mathematic modeling, wireless network security.*

Mykola Karpinskyy *graduated from the Lviv Polytechnic Institute in 1980. He has been working in the University of Bielsko-Biala since 2002. Currently he is a professor in Department of Electrical Engineering and Automatic. He obtains doctor's and habilitation's degrees in Science Electrical and Magnetic Instrumentation in 1989 and 1996, respectively, and full professor's title in Security of Information Technologies in 2001. Author and co-author of over 100 publications. His main areas of interest are computer systems and wireless networks, especially their security, in particular cryptographic methods of information defense, lighting engineering and electric and photometric measurements.*



Grzegorz Litawa *graduated from the University of Rzeszow in 2000 and Pedagogical University of Cracow in 2001. He has been working in the State Higher Vocational School in Nowy Sacz since 2002. Currently he is an assistant professor in Institute of Engineering. Author and co-author of over 10 publications. His main areas of interest are cryptography, cryptoanalysis, calculations distracted, design of digital circuits with the use in cryptograph and computer network security.*