I.J. Intelligent Systems and Applications, **2018, 12, 27-34**
Published Online December 2018 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2018.12.03

Improved Method of López-Dahab-Montgomery Scalar Point Multiplication in Binary Elliptic Curve Cryptography

Zhengbing Hu
School of Educational Information Technology, Central China Normal University, China
E-mail: hzb@mail.ccnu.edu.cn

Ivan Dychka, Mykola Onai, Mykhailo Ivaschenko
Faculty of Applied Mathematics,
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Ukraine
E-mail: dychka@pzks.fpm.kpi.ua

Su Jun
School of Computer Science, Hubei University of Technology, Wuhan, China
Email: sjhosix@gmail.com

Received: 26 July 2018; Accepted: 23 September 2018; Published: 08 December 2018

Abstract—As elliptic curve cryptography is one of the popular ways of constructing an encoding and decoding processes, public-key algorithms as its basis provide people a comfortable way of exchanging pieces of encoded information. As the time goes by, a lot of algorithms have emerged, some of them are still in use today; some others are still being developed into new forms. The main point of algorithm innovation is to reduce the number of processed operations during every possible step to find maximum efficiency and highest speed while performing the calculations. This article describes an improved method of the López-Dahab-Montgomery (LD-Montgomery) scalar point multiplication in terms of working with binary elliptic curves. It is shown in the article that the possible improvement lies in reordering the set of operations which is used in LD-Montgomery scalar point multiplication algorithm. The algorithm is used to compute point multiplication results of the curves over binary Galois Fields featuring the following *m* values: $m \in \{32, 64, 128, 256, 512, 1024, 2048, 4096\}$. The article also presents the experimental results based on different scalars.

Index Term—Elliptic curve cryptography, binary elliptic curve, scalar point multiplication, finite field arithmetic, López-Dahab, LD-Montgomery scalar point multiplication algorithm.

I. Introduction

Nowadays information technologies develop and define the world's progress while we talk about the information flow, methods of IT processing and managing patterns. As a result, the issues of data security are becoming more and more important in aspects of the methods of transferring unencrypted data which is unsafe and unreliable to use. Thus, we find ourselves in the situation where data encryption is not simply a necessary option but it is vital, and if one wants his data to be transferred securely, he'd better to use data encryption methods more or less [1].

One of the efficient data encryption technologies is elliptic curve cryptography which is constructed by focusing on Galois Fields. One of the most important areas of applying elliptic curve over Galois Field operations is systems of cryptographic protection of information [2]. One of the most costly operations in terms of computing in implementing the methods is to cryptography data encoding which uses elliptic curves, it is multiplication of a point on an elliptic curve by a number. That is why the issue of speeding up the scalar point multiplication is relevant.

II. Galois Field Concepts. Binary Field Arithmetic

Field is an algebraic structure with two operations – addition (+) and multiplication ($\cdot$), which satisfies the following conditions:

- Closeness and associativity of the operations;
- Unity element existing (for adding the unity element is 0, for multiplication – 1);
- Inverse element existing (for adding the unity element is $-a$, for multiplication a^{-1});
- Commutative addition law;
- Commutative multiplication law for non-zero elements;

- Distributive law.

The real numbers, the rational numbers, and the complex numbers under addition and multiplication are examples of algebraic fields [3].

The field is called *finite* or *Galois field* if it contains a finite number of elements.

There are two kind of finite fields:

- prime Galois fields, denoted as $GF(p)$;
- Galois fields extension, denoted as $GF(p^m)$.

All the elements of a $GF(p)$ are integer numbers from 0 to $p-1$. The operations in this field are performed modulo p, where p is a prime number. A polynomial form representation or normal form representation is usually used for building a $GF(p^m)$. Let's take a closer look at the polynomial representation. It is followed by building a finite field, its elements are the polynomial of the power which are not more than $m-1$:

$$GF(p^m)=\left\{a_{m-1}x^{m-1}+a_{m-2}x^{m-2}+\ldots+a_2x^2+a_1x+a_0 : a_i\in\{0,1\}\right\}.$$

Meanwhile, cryptosystems over $GF(2^m)$ are widely used.

The elements of $GF(2^m)$ are the binary polynomials of degree value, the maximum value is $m-1$.

The operations exceeding $GF(2^m)$ are performed by modulo irreducible polynomial of power m.

A field element $a(x)=\sum_{i=0}^{m-1}a_ix^i$ is associated with the binary vector $a(x)=(a_{m-1},a_{m-2},\ldots,a_1,a_0)$ of length m. Elements addition in terms of $GF(2^m)$ is performed by bitwise modulo addition of two vectors (bitwise exclusive-or $A\oplus B$), which contains the elements of the fields. Elements multiplication in terms of $GF(2^m)$ is performed by modulo irreducible polynomial.

III. Elliptic Curve Arithmetic

Cryptographic mechanisms based on elliptic curves depend on how we operate the points of the used curve. Meanwhile, curve arithmetic is defined by some field (meaning the operations of this field). As the greater focus is placed on the efficiency, the operations of the chosen field must be efficient enough [4].

Elliptic curve over a field K is a set of points where lie in a plane and satisfy the affine equation:

$$y^2+a_1xy+a_3y=x^3+a_2x^2+a_4x+a_6,$$

In which, the points are at infinity.

The number of the curve points is also named by *the order* of the elliptic curve.

Variables x, y and the coefficients a_i, $i=\overline{1..6}$ take the values which are the elements of the field K.

Depending on the characteristics of the field K, the general equation of an elliptic curve can be expressed in a simplified form.

Let us review the form of the equation of an elliptic curve over the $GF(2^m)$. The characteristic of this field is 2. Depending on the value of the a_1, we get two types of elliptic curves over $GF(2^m)$:

- if $a_1\neq 0$, then we get the equation of a non-super singular curve: $y^2+xy=x^3+ax^2+b$;
- if $a_1=0$, then we get the equation of a super singular curve:

$$y^2+a_3y=x^3+a_2x^2+b,\ a_3\neq 0.$$

In cryptography, the usage of super singular curves is not recommended. Special properties of these curves allow reduce discrete logarithm problem over the elliptic curve to the discrete logarithm problem over the finite field. This fact causes strength reduction in modern cryptosystems built upon such curves [5].

Base operation over the points of an elliptic curve are point addition and point doubling.

Let $P=(x_1;y_1)$ and $Q=(x_2;y_2)$. Then $P+Q=(x_3;y_3)$, where $x_3=\lambda^2+\lambda+\mu$, $y_3=(1+\lambda)x_3+\nu$, and

$$\lambda=\begin{cases}(y_1+y_2)(x_1+x_2)^{-1}, P\neq\pm Q;\\ x_1+y_1x_1^{-1}, P=Q;\end{cases}$$

$$\mu=\begin{cases}x_1+x_2+a, P\neq\pm Q;\\ a, P=Q;\end{cases}\qquad \nu=\begin{cases}\lambda x_1+y_1, P\neq\pm Q;\\ x_1^2, P=Q.\end{cases}$$

In case $P = Q$ (elliptic curve point doubling), it substitute the values of λ, and μ, we get the following formula to get x_3: $x_3=\left(x_1+y_1x_1^{-1}\right)^2+x_1+y_1x_1^{-1}+a=$
$=x_1^2+2y_1+y_1^2x_1^{-2}+x_1+y_1x_1^{-1}+a.$

An elliptic curve is defined over $GF(2^m)$, then $2y_1 = 0$. From the elliptic curve equation of $x\neq 0$, the value of x_3 can be obtained as following:

$$y^2+xy=x^3+ax^2+b\Leftrightarrow y^2x^{-2}+yx^{-1}=x+a+bx^{-2}\Leftrightarrow$$
$$\Leftrightarrow bx^{-2}=y^2x^{-2}+yx^{-1}-x-a\Leftrightarrow$$
$$\Leftrightarrow bx^{-2}=y^2x^{-2}+x+yx^{-1}+a.$$

So , $x_3 = x_1^2 + bx_1^{-2}$.

Also, all the operations over the coordinates of elliptic curves points are performed by the rules of $GF(2^m)$ accordingly.

Sometimes a necessity appears to calculate the point which is inverse to the equation $P=(x;y)$. Such operation in terms of using a non-super singular curve over the field $GF(2^m)$ is performed by substituting the second coordinate's value to *x* + *y*, so the point *–P* locates the following coordinates: $(x;x+y)$.

IV. Existing Point Multiplication Methods

Presume *E* presents an elliptic curve defined by $GF(2^m)$. Our task is to compute s_P, where *s* is a positive integer and *P* is a point on an elliptic curve *E*. This operation is called *point multiplication* or *scalar multiplication*, and is widely used in the modern elliptic cryptography methods, which is one of the most important and costly operations in processing encryption/decryption. The core operations in elliptic-curve cryptography are based specifically on scalar multiplication (e.g. single-scalar multiplication $(s,P \mapsto sP)$, double-scalar multiplication $(s,n,P,Q \mapsto sP+nQ)$, etc.) [6]. So far, there have been presented a lot of methods for executing this operation. In this article, only single-scalar multiplication methods are going to be analyzed. Also, only those methods that perform sequential point multiplications will be reviewed further [7].

All the methods of elliptic curve scalar point multiplication are based on the analysis of scalar's binary representation. The versions of algorithms which analyze each bit (one after another) per step of the algorithm are the classic ones. Depending on the order of the analysis of bits (from low to high and vice versa), there are two types of algorithms: RL (right-to-left) and LR (left-to-right).

A. Basic binary methods

Algorithm 1. Implementation of RL-binary method

Input: $s=(s_{l-1},\ldots,s_1,s_0)_2, P\in E\left(GF(2^m)\right)$

Output: *sP*

1. $Q \leftarrow \infty$
2. For *i* from 0 to *l* – 1 do
   2.1. If s_i = 1 then $Q \leftarrow Q+P$
   2.2. $P \leftarrow 2P$
3. Return *Q*

Algorithm 2. Implementation of LR-binary method

Input: $s=(s_{l-1},\ldots,s_1,s_0)_2, P\in E\left(GF(2^m)\right)$

Output: *sP*

1. $Q \leftarrow \infty$
2. For *i* from *l* – 1 downto 0 do
   2.1. $Q \leftarrow 2Q$
   2.2. If s_i = 1 then $Q \leftarrow Q+P$
3. Return *Q*

B. Methods that use a Non-Adjacent Form of a scalar

A *non-adjacent form* (NAF) of a positive integer *s* is an expression $s=\sum_{i=0}^{l-1} s_i 2^i$ where $s_i \in \{0,\pm 1\}, s_{l-1} \neq 0$, and no two consecutive digits s_i are nonzero [5].

Algorithm 3. Implementation of binary NAF RL-method

Input: $P\in E\left(GF(2^m)\right)$, *s*

Output: *sP*

1. Compute
   $\mathrm{NAF}(s)=\sum_{i=0}^{l-1} s_i 2^i,\ s_i \in \{0;\pm 1\};\ s_{l-1}\neq 0$
2. $Q \leftarrow \infty$
3. For *i* from 0 to *l* – 1 do
   3.1. If s_i = 1 then $Q \leftarrow Q+P$
   3.2. Else if s_i = –1 then $Q \leftarrow Q-P$
   3.3. $P \leftarrow 2P$
4. Return *Q*

Algorithm 4. Implementation of binary NAF LR-method

Input: $P\in E\left(GF(2^m)\right)$, *s*

Output: *sP*

1. Compute
   $\mathrm{NAF}(s)=\sum_{i=0}^{l-1} s_i 2^i,\ s_i \in \{0;\pm 1\};\ s_{l-1}\neq 0$
2. $Q \leftarrow \infty$
3. For *i* from *l* – 1 downto 0 do
   3.1. $Q \leftarrow 2Q$
   3.2. If s_i = 1 then $Q \leftarrow Q+P$
   3.3. Else if s_i = –1 then $Q \leftarrow Q-P$
4. Return *Q*

C. Window methods

The time of Algorithm 3 and 4 can be decreased by using a window method which allows us to process *w* digits of *s* at a time. Yet it requires some extra memory to be performed [8].

In case $w \geq 2$, it is a positive integer. A *width-w NAF* of a positive integer *s* is expressed by the following expression: $s=\sum_{i=0}^{l-1} s_i 2^i$ and where each nonzero coefficient s_i is odd, $|s_i| < 2^{w-1}, s_{l-1} \neq 0$, and at most one of any *w* consecutive digits is nonzero. Let's also put that the width-*w* NAF (NAF_w) has length *l* [6].

NAF_w of a positive integer *s* can be obtained using Algorithm 5.

Algorithm 5. Computing the width-*w* NAF of a positive integer

Input: window width *w*, *s*
Output: $\mathrm{NAF}_w(s)$

1. $i \leftarrow 0$
2. While $s \geq 1$ do
   2.1. If *s* is odd then
   $s_i \leftarrow s \bmod 2^w,\ s \leftarrow s - s_i$
   2.2. Else $s_i \leftarrow 0$
   2.3. $s \leftarrow s/2,\ i \leftarrow i+1$
3. Return $(s_{l-1}, s_{l-2}, \ldots, s_1, s_0)$

Algorithm 6. Implementation of window NAF LR-method

Input: $P \in E\left(GF(2^m)\right)$, window width *w*, *s*

Output: *sP*

1. Use Algorithm 5 to compute $\mathrm{NAF}_w(s) = \sum_{i=0}^{l-1} s_i 2^i$
2. Compute $P_i = iP$ for $i \in \left\{1, 3, 5, \ldots, 2^{w-1} - 1\right\}$
3. $Q \leftarrow \infty$
4. For *i* from *l* – 1 downto 0 do
   4.1. $Q \leftarrow 2Q$
   4.2. If $s_i \neq 0$ then
   If $s_i > 0$ then $Q \leftarrow Q + P_{s_i}$
   Else $Q \leftarrow Q - P_{-s_i}$
5. Return *Q*

Window NAF RL-method construction is inexpedient at each step of such algorithm. It is necessary to recalculate the table of precomputations which reduces the speed of the algorithm.

V. López-Dahab Coordinates

Besides the affine coordinate system, there also exist several coordinate systems of projective types. One of the systems is named *López-Dahab projective coordinate system* (LD projective coordinate system), it is very popular. In terms of using this coordinate system, point addition is performed in *mixed coordinates*, i.e., one point is given in affine coordinates while the other is given in projective coordinates [9].

In this system, a point (*X*; *Y*; *Z*) is called projective and represents points in affine coordinate system: $x = X \cdot Z^{-1}$ $y = X \cdot Z^{-2}$. According to this statement, we can get a new view for the elliptic curve equation in terms of using LD projective coordinates:

$$Y^2 + X \cdot Y \cdot Z = X^3 \cdot Z + a \cdot X^2 \cdot Z^2 + Z^4.$$

Points represented in LD projective coordinates satisfy the properties that differ from those that are valid for the points in affine coordinate system. Here are two of them:

1. The point at infinity is represented as the point *O* = (1; 0; 0). And for any point *P* on the curve is true that: *P* + *O* = *O* + *P* = *P*.
2. If *P* = (*X*; *Y*; *Z*) is a point on a curve, then the point –*P* = (*X*; *X* + *Y*; *Z*). This operation is also called *the inverse* of point *P*.

VI. López-Dahab Montgomery Scalar Point Multiplication Method

Presume *E* is a non-super singular elliptic curve over $GF(2^m)$, its coordinate satisfies the affine equation $y^2 + xy = x^3 + ax^2 + b,$ and the equation $Y^2 + X \cdot Y \cdot Z = X^3 \cdot Z + a \cdot X^2 \cdot Z^2 + Z^4$ is valid for LD projective coordinates.

Peter Laurence Montgomery introduced a method of speeding up performance of the modulo multiplication operation for big values of the modulo (more than 128 bit) in 1985. Before executing of the multiplication, the operands must be converted into Montgomery form. For the obtained result, an inverse transformation must be made. The last operation is computationally expensive because it contains the computation of multiplicative inverse [10]. Thus, Montgomery multiplication is used only in the algorithms of exponentiation.

Exponentiation algorithm that contain Montgomery arithmetic can be split into three phases [10]:

1. Operand's conversion is converted into Montgomery form.
2. Classic exponentiation algorithm execution in terms of Montgomery forms (a lot of multiplications).
3. Converting the result from Montgomery form.

Further Peter Montgomery in his work that was dedicated to speed up the algorithms of factorization of the elliptic curves introduced in a new algorithm of scalar point multiplication (Algorithm 7) [11].

Algorithm 7. Implementation of Montgomery point multiplication method

Input: $(s_{l-1}, s_{l-2}, \ldots, s_1, s_0)_2$ with $s_{t-1} = 1$,
$P \in E\left(GF(2^m)\right)$

Output: *sP*

1. $R_0 \leftarrow P,\ R_1 \leftarrow 2P$
2. For *i* from *l* – 2 downto 0 do
   $R_{1-s_i} \leftarrow R_{1-s_i} + R_{s_i}$
   $R_{s_i} \leftarrow 2R_{s_i}$
3. Return R_0

López and Dahab later introduced a modification (also for binary curves over $GF(2^m)$) that was actually an improved Montgomery algorithm [11]. The scientists presented the idea that the sum of two points can be

evaluated using only the operations that involve the *x*-coordinates of the points in procedures of doubling and adding per iteration.

In this algorithm that was called *MDouble*, the point $P = (x, y)$ is represented in the LD-projective form $P = \left(X \cdot Z^{-1}, Y \cdot Z^{-2}\right)$ (Algorithm 8).

Similarly, the point addition algorithm was called *MAdd* (Algorithm 9). Thus, following the modification designed by López and Dahab, we get Algorithm 10 [12, 13].

Algorithm 8. MDouble operation in López-Dahab Montgomery point multiplication method

Input: $b \in GF(2^m)$, $P(X : Y : Z) \in E\left(GF(2^m)\right)$

Output: $X \cdot Z^{-1}$ is affine *x*-coordinate of the point $2P$

1. $X \leftarrow X^2$
2. $Z \leftarrow Z^2$
3. $T \leftarrow Z^2$
4. $Z \leftarrow Z \cdot X$
5. $X \leftarrow X^2$
6. $T \leftarrow b \cdot T$
7. $X \leftarrow X + T$
8. Return $X \cdot Z^{-1}$

Algorithm 9. MAdd operation in López-Dahab Montgomery point multiplication method

Input: $P(x, y)$, $P_1(X_1 : Y_1 : Z_1)$, $P_2(X_2 : Y_2 : Z_2) \in$
$\in E\left(GF(2^m)\right)$

Output: affine *x*-coordinate $X_1 \cdot Z_1^{-1}$ of the point $P_1 + P_2$

1. $T_1 \leftarrow Z_1 \cdot X_2$
2. $T_2 \leftarrow Z_2 \cdot X_1$
3. $Z_1 \leftarrow T_1 + T_2$
4. $X_1 \leftarrow x \cdot Z_1$
5. $T_1 \leftarrow T_1 \cdot T_2$
6. $X_1 \leftarrow X_1 + T_1$
7. Return $X_1 \cdot Z_1^{-1}$

Algorithm 10. Implementation of López-Dahab Montgomery point multiplication method

Input: $(s_{l-1}, s_{l-2}, \ldots, s_1, s_0)_2$ with $s_{t-1} = 1$,
$P = (x, y) \in E\left(GF(2^m)\right)$

Output: sP

1. $X_1 \leftarrow x, Z_1 \leftarrow 1, X_2 \leftarrow x^4 + b, Z_2 \leftarrow x^2$
2. For i from $l - 2$ downto 0 do
   2.1. If $s_i = 1$, then

$$\left.\begin{array}{l} T \leftarrow Z_1 \\ Z_1 \leftarrow \left(X_1 Z_2 + X_2 Z_1\right)^2 \\ X_1 \leftarrow xZ_1 + X_1 X_2 T Z_2 \end{array}\right\} MAdd$$

$$\left.\begin{array}{l} T \leftarrow X_2 \\ X_2 \leftarrow X_2^4 + bX_2^4 \\ Z_2 \leftarrow T^2 Z_2^2 \end{array}\right\} MDouble$$

   2.2. Else

$$\left.\begin{array}{l} T \leftarrow Z_2 \\ Z_2 \leftarrow \left(X_1 Z_2 + X_2 Z_1\right)^2 \\ X_2 \leftarrow xZ_2 + X_1 X_2 Z_1 T \end{array}\right\} MAdd$$

$$\left.\begin{array}{l} T \leftarrow X_1 \\ X_1 \leftarrow X_1^4 + bZ_1^4 \\ Z_1 \leftarrow T^2 Z_1^2 \end{array}\right\} MDouble$$

3. $x_3 \leftarrow X_1 \cdot Z_1^{-1}$
4. $y_3 \leftarrow (x + x_3) \left[\begin{array}{l} \left(X_1 + xZ_1\right) \cdot \\ \cdot \left(X_2 + xZ_2\right) + \\ + \left(x^2 + y\right) \cdot \\ \cdot \left(Z_1 Z_2\right) \end{array}\right] \left(xZ_1 Z_2\right)^{-1} + y$
5. Return (x_3, y_3)

The algorithm is considered to be a faster approach of the basic Montgomery method which thanks to a fewer numbers of operations to be performed per iteration, it secondly contains no inversion which is once used in the basic version.

VII. Improved MDouble

Let $c = \sqrt{b} \Rightarrow c = b^{2^{m-1}}$, where m is the order of the curve. Thus, we can achieve speed-up of the algorithm by precomputing c and getting rid of squaring, as we know that the following statement is true: $(a+b)^2 = a^2 + b^2$ for $GF(2^m)$. Using this, we can say that the $Q = X^4 + b \cdot Z^4$ expression could be reformed as: $Q = \left(X^2 + c \cdot Z^2\right)^2$.

So, we suggest a new modification of the MDouble algorithm (Algorithm 11).

Algorithm 11. Improved MDouble

Input: $c \in GF(2^m)$, where $c = \sqrt{b}$, $b \in GF(2^m)$,
$X \cdot Z^{-1}$ is affine *x*-coordinate of the point
$P \in E\left(GF(2^m)\right)$

Output: $X \cdot Z^{-1}$ is affine *x*-coordinate of the point $2P$

1. $X \leftarrow X^2$
2. $Z \leftarrow Z^2$
3. $T \leftarrow c \cdot Z$
4. $Z \leftarrow Z \cdot X$

5. $X \leftarrow X + T$
6. $X \leftarrow X^2$
7. Return $X \cdot Z^{-1}$

So, we can see that the number of operations in Algorithm 11 is fewer than the number of operations in Algorithm 8 (table 1).

Table 1. Number of operations performed per one doubling

	Squaring	Multiplication	Addition
1. MDouble	4	2	1
2. Improved MDouble	3	2	1

May we have an elliptic curve E, where: $a = z^3$, $b = z^3 + 1$, $c = z^3 + z + 1$, and *irreducible polynomial* $z^4 + z + 1$. May the scalar $s = 5_{10} = 101_2$.

May the point $P = \{z^3 + 1 : z^3 + z^2 + z + 1 : 1\}$.

Let's compare the results of calculations during the doubling of a point of an elliptic curve while using Algorithm 8 and Algorithm 9 (table 2, 3). It calculates the value of Z-coordinate in step 4 of both algorithms that is further used for computing Z^{-1}. It calculates the X value in the last step of both algorithms.

Table 2. The first doubling results during the process of multiplication. $P_i = (\{1\ 0\ 0\ 1\} : \{0001\})$

Step	MDouble		Improved MDouble	
	Var.	Value	Var.	Value
1	X	{1 1 0 1}	X	{1 1 0 1}
2	Z	{0 0 0 1}	Z	{0 0 0 1}
3	T	{0 0 0 1}	T	{1 0 1 1}
4	Z	{1 1 0 1}	Z	{1 1 0 1}
5	X	{1 1 1 0}	X	{0 1 1 0}
Step	Var.	Value	Var.	Value
6	T	{1 0 0 1}	X	{0 1 1 1}
7	X	{0 1 1 1}		–
$X \cdot Z^{-1}$	$\left(z^2+z+1\right)\cdot\left(z^3+z^2+1\right)^{-1}$		$\left(z^2+z+1\right)\cdot\left(z^3+z^2+1\right)^{-1}$	

Table 3. The second doubling results during the process of multiplication. $P_i = (\{1\ 1\ 1\ 1\} : \{1\ 0\ 0\ 0\})$

Step	MDouble		Improved MDouble	
	Var.	Value	Var.	Value
1	X	{1 0 1 0 }	X	{1 0 1 0}
2	Z	{1 1 0 0 }	Z	{1 1 0 0}
3	T	{1 1 1 1 }	T	{1 1 0 1}
4	Z	{0 0 0 1}	Z	{0 0 0 1}
5	X	{1 0 0 0 }	X	{0 1 1 1}
6	T	{1 1 1 0 }	X	{0 1 1 0}
7	X	{0 1 1 0 }		–
$X \cdot Z^{-1}$	$\left(z^2+z\right)\cdot(1)^{-1}$		$\left(z^2+z\right)\cdot(1)^{-1}$	

VIII. The Experimental Research

The software for performing experimental researches was designed in Visual Studio 2017 by using C# programming language. The hardware characteristics of the program are the following: CPU Intel Core I7-6500U @2.5 – 2.6 GHz, random access memory 8 Gb. The research was mainly based on comparing the timings of elliptic curves points' multiplication with different lengths of scalars and different orders of elliptic curves (table 4, 5).

Table 4. Lengths of scalars used for multiplication, bits

128	256	512	1024	2048	4096

Table 5. Orders of elliptic curves

32	64	128	256	512	1024	2048	4096

Using these orders enable us to understand if the algorithm is viable for cryptography. Further researches are presented on the tables which contains the comparison of timings of the LD-Montgomery scalar point multiplication method and its modification with the improved *MDouble* (tables 6 – 11).

Table 6. Point multiplication execution time with $m = 32$

Scalar length, bits	LD-Montgomery method, ms	LD-Montgomery method with improved MDouble, ms
128	0,95	0,7
256	1,17	1,04
512	1,97	1,8
1024	3,59	3,47
Scalar length, bits	LD-Montgomery method, ms	LD-Montgomery method with improved MDouble, ms
2048	7,05	6,4
4096	14,01	13,82

Table 7. Point multiplication execution time with $m = 64$

Scalar length, bits	LD-Montgomery method, ms	LD-Montgomery method with improved MDouble, ms
128	0,77	0,74
256	1,24	1,15
512	2,15	2,01
1024	4,07	3,86
2048	7,6	7,3
4096	14,93	14,33

Table 8. Point multiplication execution time with $m = 128$

Scalar length, bits	LD-Montgomery method, ms	LD-Montgomery method with improved MDouble, ms
128	0,93	0,89
256	1,49	1,32
512	2,55	2,39
1024	4,79	4,63
2048	8,76	8,45
4096	17,01	16,7

Table 9. Point multiplication execution time with $m = 256$

Scalar length, bits	LD-Montgomery method, ms	LD-Montgomery method with improved MDouble, ms
128	1,11	1,01
256	2,35	1,64
512	3,84	2,94
1024	6,11	5,9
2048	5,33	5,19
4096	10,82	10,58

Table 10. Point multiplication execution time with $m = 512$

Scalar length, bits	LD-Montgomery method, ms	LD-Montgomery method with improved MDouble, ms
128	2,32	1,77
256	2,9	2,35
512	4,96	4,3
1024	8,53	7,82
2048	14,34	14,19
4096	29,44	28,29

Table 11. Point multiplication execution time with $m = 1024$

Scalar length, bits	LD-Montgomery method, ms	LD-Montgomery method with improved MDouble, ms
128	2,98	2,64
256	4,59	4,33
512	7,27	6,69
1024	12,14	12,14
2048	23,11	23,04
Scalar length, bits	LD-Montgomery method, ms	LD-Montgomery method with improved MDouble, ms
4096	46,78	45,82

Table 12. Point multiplication execution time with $m = 2048$

Scalar length, bits	LD-Montgomery method, ms	LD-Montgomery method with improved MDouble, ms
128	5,57	5,46
256	11,91	8
512	22,31	22,06
1024	21,77	21,28
2048	40,82	37,77
4096	76,72	77,39

Table 13. Point multiplication execution time with $m = 4096$

Scalar length, bits	LD-Montgomery method, ms	LD-Montgomery method with improved MDouble, ms
128	9,8	9,58
256	26,63	25,52
512	41,08	38,36
1024	38,73	37,57
2048	70,44	68,85
4096	145,7	142,95

Thus, we can see that speed performance has been slightly improved for every scalar length presented here and for every order of elliptic curve from what have been reviewed. Even more, with the order growth, the delta between the time-values slowly rises, it comes up with the thought that MDouble-method and the presented modification are effective for cryptography purposes [14].

IX. Conclusions

Considering the methods used to perform point multiplication, the most powerful performance statistic is the MDouble-method created by López-Dahab through modifying the initial Montgomery double-and-add method [15].

The method contains two operations, it is obvious that improving the operation is working in some way to speed up the algorithm overall. Being used in cryptography and taking into account the fact that big data encryption is widely using multiple processors, a slight performance improvement could even lead to big one in general.

Thus, a modification of the *MDouble* operation was introduced. Increasing performance was achieved by using precomputations. The result gave an algorithm modification with fewer operations and performance time improvements.

Review the multiplication results of different orders curves (from 32 to 4096) by the scalars of different byte length (from 16 to 512), the following results are obtained: the average increase value is 4-5% (20 points per scalar were used). As mentioned above, considering all the processors working at the same time, multi-processor systems that the encryption technologies based on Galois Fields theory will obtain even bigger improvements [16].

Using improved operations potentially in terms of working with polynomials over Galois Field (addition, multiplication etc.), operations can be also improved in terms of working with scalar point multiplication, a better result could be obtained in future.

References

[1] Omar A. Dawood, Abdul Monem S. Rahma, Abdul Mohsen J. Abdul Hossen,"The New Block Cipher Design (Tigris Cipher)", IJCNIS, vol.7, no.12, pp.10-18, 2015.DOI: 10.5815/ijcnis.2015.12.02

[2] Bardis N.G. Fast implementation zero knowledge identification schemes using the Galois Fields arithmetic / Bardis N.G., Markovskyi O.P., Doukas N., Drigas F. // International Symposium IEEE on Telecommunications, Proceeding of IX – 2012

[3] Ç. K. Koç Cryptographic Engineering // Springer : ISBN – 2009 – 75 p.

[4] Ritu Goyal, Mehak Khurana,"Cryptographic Security using Various Encryption and Decryption Method", International Journal of Mathematical Sciences and Computing(IJMSC), Vol.3, No.3, pp.1-11, 2017.DOI: 10.5815/ijmsc.2017.03.01

[5] Henri Cohen Handbook of Elliptic and Hyperelliptic Curve Cryptography / Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, Frederik Vercauteren // Taylor & Francis Group – Pp. 43-44.

[6] Daniel J. Bernstein, Tanja Lange Faster addition and doubling on elliptic curves // Chicago, USA. Eindhoven, Netherlands – 2007 – Pp. 26-28.

[7] Nagaraja Shylashree, Venugopalachar Sridhar,"Hardware Realization of Fast Multi-Scalar Elliptic Curve Point

Multiplication by Reducing the Hamming Weights Over GF(p)", IJCNIS, vol.6, no.10, pp.57-63, 2014. DOI: 10.5815/ijcnis.2014.10.07

[8] Darrel Hankerson Guide to Elliptic Curve Cryptography / Hankerson Darrel, Menezes Alfred, Vanstone Scott // Springer-Verlag New York, USA. — 2004. — Pp. 98-99.

[9] F. Rodríguez-Henríquez Cryptographic Algorithms on Reconfigurable Hardware / F. Rodríguez-Henríquez, Nazar A. Saqib, A. Díaz-Pérez and Çetin K. Koç // Springer; ISBN – 2007 – 84 p.

[10] Peter l. Montgomery Modular Multiplication Without Trial Division // Ameriacn Mathematicl Society – 1985

[11] Peter L. Montgomery Speeding the Pollard and Elliptic Curve Methods of Factorization // American Mathematical Society – 1987 – Pp. 243-264.

[12] J. López, R. Dahab Fast Multiplication on Elliptic Curves over GF(2m) without Precomputation // Ontario, Canada. Campinas, Brazil – 1999.

[13] Darrel Hankerson Guide to Elliptic Curve Cryptography / Hankerson Darrel, Menezes Alfred, Vanstone Scott // Springer-Verlag New York, USA. — 2004. — 103 p.

[14] G. Seroussi Table of Low-Weight Binary Irreducible Polynomials from HP Labs Technical Reports http://www.hpl.hp.com/techreports/98/HPL-98-135.pdf // 1998.

[15] Zhen Huang,Shuguo Li,"Design and Implementation of a Low Power RSA Processor for Smartcard", IJMECS, vol.3, no.3, pp.8-14, 2011.

[16] Shivashankar S., Medha Kudari, Prakash S. Hiremath, " Galois Field-based Approach for Rotation and Scale Invariant Texture Classification", International Journal of Image, Graphics and Signal Processing(IJIGSP), Vol.10, No.9, pp. 56-64, 2018.DOI: 10.5815/ijigsp.2018.09.07

Authors' Profiles

Zhengbing Hu: Ph.D., Associate Professor of School of Educational Information Technology, Central China Normal University, M.Sc. (2002), Ph.D. (2006) from the National Technical University of Ukraine "Kyiv Polytechnic Institute". Postdoc (2008), Huazhong University of Science and Technology, China. Honorary Associate Researcher (2012), Hong Kong University, Hong Kong. Major interests: Computer Science and Technology Applications, Artificial Intelligence, Network Security, Communications, Data Processing, Cloud Computing, Education Technology.

Ivan Dychka: D.S., Professor, Dean of Faculty of Applied Mathematics, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Ukraine. Research Interests: Computer Systems and Networks Software, Automated Control Systems, Intelligence and Expert Systems, Databases and Knowledge Bases, Information Security Software for Computer Systems and Networks.

Mykola Onai was born on December 06, 1986. He received his Bachelor's Degree in Computer Engineering (June 2008) and his Master of Science Degree in Computer Systems and Networks (June 2010), both from the Department of Special Purpose Computer Systems at National Technical University of Ukraine "Kyiv Polytechnic Institute", Kyiv, Ukraine and the PhD degree in Computer Systems and Components in February 2018 from the Computer Systems Software Department at the National Technical University "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine. He is currently an Associate Professor in the Computer Systems Software Department at National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine. His main research interests are Finite Field Arithmetic, Public Key Cryptography, Elliptic Curve Cryptography, Computer Security, Network Security and Hardware Algorithms for Cryptography. Mykola Onai has authored and co-authored more than 40 scientific publications, and is inventor of 4 patents.

Mykhailo Ivashchenko was born in Dnipro, Ukraine on September 13, 1998. He graduated from "Dnipro Lyceum of Information Technologies at DNU" in Dnipro in June 2015, and is currently studying in the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" at Kyiv on a Bachelor program. The author's main scientific fields of interest are Mathematics and Programming.

Su Jun, born in Hubei China, received Ph.D. degree in Computer Systems and Components from Ternopil National Economic University, Ukraine in 2013. He is associate professor in school of computer science, Hubei University of technology, Wuhan, China. He has published more than 20 papers in the area of Computer Network and wireless communication. His research interests include Wireless Sensor Network, Self-organizing network technology, Wave propagation and electromagnetic interference. Dr. Su is a member of IEEE and ACM.

How to cite this paper: Zhengbing Hu, Ivan Dychka, Mykola Onai, Mykhailo Ivaschenko, Su Jun, "Improved Method of López-Dahab-Montgomery Scalar Point Multiplication in Binary Elliptic Curve Cryptography", International Journal of Intelligent Systems and Applications(IJISA), Vol.10, No.12, pp.27-34, 2018. DOI: 10.5815/ijisa.2018.12.03

