Math-Net.Ru

Общероссийский математический портал

D. M. Dygin, S. V. Grebnev, Эффективная реализация схемы цифровой подписи ГОСТ Р 34.10 с помощью современных методов скалярного умножения на эллиптических кривых, *Матем. вопр. криптогр.*, 2013, том 4, выпуск 2, 47–57

DOI: <https://doi.org/10.4213/mvk82>

Использование Общероссийского математического портала Math-Net.Ru подразумевает, что вы прочитали и согласны с пользовательским соглашением

<http://www.mathnet.ru/rus/agreement>

Параметры загрузки:

IP: 52.6.47.48

26 апреля 2023 г., 13:49:09

УДК: 519.719.2+519.742.72

Efficient implementation of the GOST R 34.10 digital signature scheme using modern approaches to elliptic curve scalar multiplication

D. M. Dygin, S. V. Grebnev

Russian Technical Committee for Standardization (TC 26), Moscow

Получено 18.IX.2012

An approach to an efficient implementation of the Russian national digital signature scheme GOST R 34.10 in view of its new extensions is proposed. Modern algorithms for scalar multiplication and different representations of elliptic curves over prime finite fields are used. Results of numerical experiments and recommendations on the selection of parameters of algorithms are presented.

Key words: digital signature, GOST R 34.10, Edwards curves, Hessian curves, scalar multiplication, national standard, efficient implementation, performance evaluation

Эффективная реализация схемы цифровой подписи ГОСТ Р 34.10 с помощью современных методов скалярного умножения на эллиптических кривых

Д. М. Дыгин, С. В. Гребнев

Технический комитет по стандартизации (ТК 26), Москва

Аннотация. Предложен подход к эффективной реализации Российского национального стандарта цифровой подписи ГОСТ Р 34.10 в связи с его новыми модификациями. Используются современные алгоритмы скалярного умножения и различные представления эллиптических кривых над простыми полями. Приводятся результаты вычислительных экспериментов и рекомендации по выбору параметров описанных алгоритмов.

Ключевые слова: цифровая подпись, ГОСТ Р 34.10, кривые Эдвардса, кривые Хесса, скалярное умножение, национальный стандарт, эффективная реализация, оценивание эффективности

Citation: *Mathematical Aspects of Cryptography*, 2013, vol. 4, no. 2, pp. 47–57 (Russian).

© 2013 D. M. Dygin, S. V. Grebnev

1. Introduction

In 2011 Technical committee on standardization “Cryptography and security mechanisms” (TC 26) has proposed extensions¹ to the Russian national digital signature scheme standard GOST R 34.10-2001 [3] which allow to implement the scheme upon the groups of order about 2^{512} , while the standard currently in force [2] is built upon the groups of order about 2^{256} .

In this paper we show the possibilities to reduce the losses in performance connected with the increase of the group sizes by implementing modern approaches to representation of elliptic curves and algorithms for scalar multiplications.

2. GOST R 34.10 Digital Signature Scheme

GOST R 34.10 Digital Signature Scheme is a variant of generalized El-Gamal scheme built in a subgroup $\langle P \rangle_q$ of a prime order q of a group of points of an elliptic curve written in a (short) Weierstrass form

$$E_{a,b}(GF(p)) = \{(x, y) : y^2 = x^3 + ax + b\} \quad (1)$$

over a prime finite field of characteristic p , where $2^{254} < q < 2^{256}$ as required by the standard currently in force [2] or $2^{508} < q < 2^{512}$ as required by the proposed extension to the standard [3]. The signature of a message M is a pair (r, s) , where $s = dr + kh(M) \pmod{q}$, $r = x_{(kP)}$, d is the private key, $k \in_R \mathbb{Z}$, $h(\cdot)$ is the hash function defined by [4], $x_{(P)}$ defines an x -coordinate of a point P on an elliptic curve in (short) Weierstrass form. The signature verification process is defined by the equation $x_{(sh(M) \pmod{q}P - (rh(M) \pmod{q})Q)} \pmod{q} = r$, where $Q = dP$ is the public key; the signature is valid iff the equality holds.

The natural options to increase performance of an implementation of the scheme are:

- to use an efficient representation of the field elements ($p < 2^{256}$ or $p < 2^{512}$) in order to minimize the number of machine words required;
- to build special representations of the exponents which allow for faster scalar and multi-scalar multiplications, the main time-consuming processes of signature generation and verification, respectively;
- to treat an equivalent representation of the points internally, transforming them to affine coordinates for the (short) Weierstrass form as a final step.

¹ On August 07, 2012 the proposed extensions were adopted by the order of Federal Agency on Technical Regulating and Metrology and put in force as a national standard GOST R 34.10-2012 since January 01, 2013.

3. Scalar and Multi-Scalar Multiplication

A common scheme of both scalar multiplication (for an exponent k and a point P , compute kP) and multi-scalar multiplication (for exponents k_1, k_2 and points P, Q , compute $k_1P + k_2Q$) algorithms may be described in the following way:

1. pre-compute a set of points iP (iP, jQ or $iP + jQ$);
2. build a special representation of the exponent k (a special joint representation of the pair (k_1, k_2));
3. set the initial value of result to an infinity point, then “scan” the representation of exponents and for each digit add a corresponding multiple of corresponding precomputed point to a result.

The problem is to minimize the number of additions, i.e. the weight of a representation, while the number of multiplications is generally defined by bit-length of exponents.

After a theoretical study and a series of experiments, we have chosen to use *wmbNAF* as in [10] for scalar multiplication and *jwmbNAF* as in [9] for multi-scalar multiplication. Several other methods examined, e.g. DBNS [7], appeared totally impractical in our setting.

Multibase non-adjacent form with window w (wmbNAF) of an integer k , denoted $(a_1, \dots, a_J)NAF_w(k) = \{d_1^{(a_1)} \dots d_m^{(a_m)}\}$, is a sequence of digits d_i , each one associated with a base a_i from a fixed finite set $A = \{a_1, \dots, a_J\}$ such that:

1. any positive d has unique representation of the form $(a_1, \dots, a_J)NAF_w(d)$ for the set of coprime bases A and window w ;
2. for any w consecutive digits, at most one is non-zero;
3. $d_i \in \{0, \pm 1, \pm 2, \dots, \pm \lfloor (a_1^w - 1)/2 \rfloor\} \setminus \{\pm a_1, \pm 2a_1, \dots, \pm \lfloor (a_1^{w-1} - 1)/2 \rfloor a_1\}$, $d_1 > 0$;
4. $k = (\dots((d_1 \cdot (a_2) + d_2) \cdot (a_3)) + \dots + d_{m-1}) \cdot (a_m) + d_m$; the latter formula, where (a_i) denotes the base associated with the i -th digit, gives a natural algorithm for scalar multiplication.

Please note that we keep the original notations from [10] hereafter, so the term $d_i = d^{(a_j)}$ should be read as “set the i -th digit to the value d and set the associated base to a_j ”. The reader should not be confused by the notation (a_i) , which denotes the base actually written at the i -th position of the wmbNAF, *not* its index in A .

A *joint multibase non-adjacent form with window w* (*jwmbNAF*) of a pair of integers k_1, k_2 is actually a pair of *wmbNAF*'s of length l , maybe left-padded with zeroes for a smaller integer, calculated for k_1, k_2 simultaneously in such a manner that the bases (a_i) associated with each position coincide:

$$(a_1, \dots, a_J) \overleftarrow{NAF}_w(k_1, k_2) = \begin{pmatrix} (d_1^{(a_1)}, \dots, d_l^{(a_l)}) \\ (e_1^{(a_1)}, \dots, e_l^{(a_l)}) \end{pmatrix}, \text{ and} \\ k_1P + k_2Q = (a_{l-1})(\dots(a_3)((a_2)(d_1P + e_1Q) + d_2P + e_2Q) + \dots \\ + d_{l-1}P + e_{l-1}Q) + d_lP + e_lQ;$$

the latter formula gives a natural algorithm for multi-scalar multiplication.

The paper [9] presents algorithms to compute *jwmbNAF* in two different ways, denoted $(a_1, \dots, a_J) \overrightarrow{NAF}_w(k_1, k_2)$ ("left-to-right") or $(a_1, \dots, a_J) \overleftarrow{NAF}_w(k_1, k_2)$ ("right-to-left") depending on the order in which the bases are treated. Since the paper is not widely available, we provide a sketch of the first algorithm (to obtain the second option, just reverse the order in which a_j 's are treated at the step 2.1).

Algorithm 1. Input: $k_1, k_2 \in \mathbb{N}$, $J \geq 1$, $A = \{a_1, a_2, \dots, a_J\}$ — an ascending set of different prime bases, $j = 1, \dots, J$, window $w \in \mathbb{N}$, $w \geq 2$.

Output:

$$(a_1, \dots, a_J) \overrightarrow{NAF}_w(k_1, k_2) = \begin{pmatrix} (d_1^{(a_1)}, \dots, d_l^{(a_l)}) \\ (e_1^{(a_1)}, \dots, e_l^{(a_l)}) \end{pmatrix}.$$

1. $i = 1$;
2. **while** $k_1 > 0$ **or** $k_2 > 0$ **do**
 - 2.1. **for** $j = 1$ **to** J **do** ("left-to-right")
 - 2.1.1. **if** $k_1 \bmod a_j == 0$ **and** $k_2 \bmod a_j == 0$ **then**
 - 2.1.1.1. $d_i = 0$, $k_1 = k_1/a_j$, $d_i = d_i^{(a_j)}$;
 - 2.1.1.2. $e_i = 0$, $k_2 = k_2/a_j$, $e_i = e_i^{(a_j)}$;
 - 2.1.1.3. **break**;
 - 2.1.2. **else if** $k_1 \bmod a_j == 0$ **and** $k_2 \bmod a_j \neq 0$ **then**
 - 2.1.2.1. $e_i = k_2 \bmod a_1^w$,
 - 2.1.2.2. $k_2 = k_2 - e_i$;
 - 2.1.2.3. **if** $k_2 \bmod a_j == 0$ **then**
 - 2.1.2.3.1. $d_i = 0$, $d_i = d_i^{(a_j)}$;
 - 2.1.2.3.2. $e_i = e_i^{(a_j)}$;
 - 2.1.2.3.3. $k_1 = k_1/a_j$, $k_2 = k_2/a_j$;
 - 2.1.2.3.4. **break**;

2.1.2.4. **else** $k_2 = k_2 + e_i$;
 2.1.3. **else if** $((k_1 \bmod a_j \neq 0) \text{ and } (k_2 \bmod a_j == 0))$ **then**
 2.1.3.1. $d_i = k_1 \bmod a_1^w$;
 2.1.3.2. $k_1 = k_1 - d_i$;
 2.1.3.3. **if** $(k_1 \bmod a_j == 0)$ **then**
 2.1.3.3.1. $e_i = 0, e_i = e_i^{(a_j)}$;
 2.1.3.3.2. $d_i = d_i^{(a_j)}$;
 2.1.3.3.3. $k_1 = k_1/a_j, k_2 = k_2/a_j$;
 2.1.3.3.4. **break**;
 2.1.3.4. **else** $k_1 = k_1 + d_i$;
 2.1.4. **else if** $((k_1 \bmod a_j \neq 0) \text{ and } (k_2 \bmod a_j \neq 0))$ **then**
 2.1.4.1. $d_i = k_1 \bmod a_1^w$;
 2.1.4.2. $k_1 = k_1 - d_i$;
 2.1.4.3. $e_i = k_2 \bmod a_1^w$;
 2.1.4.4. $k_2 = k_2 - e_i$;
 2.1.4.5. **if** $((k_1 \bmod a_j == 0) \text{ and } (k_2 \bmod a_j == 0))$ **then**
 2.1.4.5.1. $d_i = d_i^{(a_j)}$;
 2.1.4.5.2. $e_i = e_i^{(a_j)}$;
 2.1.4.5.3. $k_1 = k_1/a_j, k_2 = k_2/a_j$;
 2.1.4.5.4. **break**;
 2.1.4.6. **else**
 2.1.4.6.1. $k_1 = k_1 + d_i$;
 2.1.4.6.2. $k_2 = k_2 + e_i$;
 2.2. $i = i + 1$;
 3. **Stop; output** $(a_1, \dots, a_J) \overrightarrow{NAF}_w(k_1, k_2)$.

As was shown in [9], the digits of $(a_1, \dots, a_J) \overrightarrow{NAF}_w(k_1, k_2)$ satisfy

$$d_i, e_i \in \left\{ 0, \pm 1, \pm 2, \dots, \pm \left\lfloor \frac{a_1^w - 1}{2} a_1 \right\rfloor \right\} \setminus \left\{ \pm a_1, \pm 2a_1, \dots, \left\lfloor \frac{a_1^{w-1} - 1}{2} \right\rfloor \right\},$$

thus requiring either a_1^w or a_1^{2w} precomputed points, depending on their form, while the digits of $(a_1, \dots, a_J) \overrightarrow{NAF}_w(k_1, k_2)$

$$d_i, e_i \in \left\{ 0, \pm 1, \pm 2, \dots, \pm \left\lfloor \frac{a_1^w}{2} \right\rfloor \right\} \setminus \{\pm 6k, k \in n\},$$

requiring either $\lceil 5/6 a_1^w \rceil$ or $\lceil 5/6 a_1^w \rceil^2$ precomputed points.

We have found that for the set of prime bases $\{2, 3\}$ which allows for efficient exploitation of points doubling and tripling, optimal window sizes for 512-bit p are 9 for $wmbNAF$ and 6 for the $jwmbNAF$. It should be noted here that the sizes of L2 cache of modern top-end CPUs may be large enough to store and efficiently treat the required tables. Using these representations in an implementation of the GOST R 34.10, we achieve an average 25 % improvement in total performance over traditional NAF and JSF . Considering the weight of a representation itself, we observe an average reduction by 75 %.

We note, however, that for the practical setting of the signature verification process, where Q is arbitrary, it is preferable to use smaller than optimal window sizes.

4. Hessian and Edwards Curves

A huge database of representations of elliptic curves alongside with practice-oriented coordinate systems proposed during the 25 years of elliptic curve cryptography is collected at [1]. Among them, we have chosen for experiments the extended coordinates on Hessian curves [8], projective and inverted coordinates on Edwards curves [6] and extended coordinates on twisted Edwards curves [5], comparing them with traditional projective Weierstrass coordinates.

We suggest the reader to refer to [1] for the most complete and up-to-date information on the actual algorithms, giving here a brief illustration of the methodology of theoretical performance optimization.

Denote the total complexity of an algorithm by $m_1M + m_2S + m_3I$, where M, S, I denote the complexity of field multiplication, squaring and inversion, assuming that the complexity of field addition or multiplication by a “small” value, e.g. 2, is negligible. Next, for a given implementation of the field arithmetics, express the total complexity by the number of field multiplications required, estimating the complexity of squaring and inversion by experiment. The problem is to find a representation of the curve which minimizes the total complexity defined hereby.

4.1. Hessian curves

A curve in Hessian form $H_d(GF(p))$ where $p \equiv 2 \pmod{3}$, is given by

$$x^3 + y^3 + 1 = 3dxy, \quad x, y, d \in GF(p). \quad (2)$$

Addition and doubling is given by

$$(x_1, y_1) + (x_2, y_2) = \left(\frac{y_1^2 x_2 - y_2^2 x_1}{x_2 y_2 - x_1 y_1}, \frac{x_1^2 y_2 - x_2^2 y_1}{x_2 y_2 - x_1 y_1} \right),$$

$$2(x_1, y_1) = \left(\frac{y_1(1 - x_1^3)}{x_1^3 - y_1^3}, \frac{x_1(y_1^3 - 1)}{x_1^3 - y_1^3} \right).$$

Any Hessian curve is birationally equivalent to a (short) Weierstrass curve $E(GF(p)) : v^2 = u^3 - 27d(d^3 + 8)u + 54(d^6 - 20d^3 - 8)$, equivalence is given by $u = \mu - 9d^2$, $v = 3\mu(y - x)$, where $\mu = 12(d^3 - 1)/(d + x + y)$. The point $(0, 0)$ has order 3.

A curve in (short) Weierstrass form (1) with j -invariant $j(E)$ is birationally equivalent over $GF(p)$ to a Hessian curve (2) iff there exists $d \in GF(p)$ such that

$$d^3(d^3 + 216)^3 - j(E)(d^9 - 81d^6 + 2187d^3 - 19683) = 0.$$

A point $(x, y) \in H_d(GF(p))$ is represented in *projective coordinates* by a triple $(X : Y : Z)$ such that $X^3 + Y^3 + Z^3 = 3dXYZ$, $(x, y) = (X/Z, Y/Z)$. Neutral element of the group is represented as $(1 : -1 : 0)$. An inverse to $(X : Y : Z)$ is the point $(Y : X : Z)$.

Projective Hessian form possesses an interesting property which allows to use similar formulas for addition and doubling: $2(X : Y : Z) = (Z : X : Y) + (Y : Z : X)$, thus avoiding some kinds of side-channel attacks.

A point $(x, y) \in H_d(GF(p))$ is represented in *extended coordinates* by a set $(X : Y : Z : XX : YY : ZZ : XY : YZ : XZ)$ such that $X^3 + Y^3 + Z^3 = 3dXYZ$, $(x, y) = (X/Z, Y/Z)$, $XX = X^2$, $YY = Y^2$, $ZZ = Z^2$, $XY = 2X \cdot Y$, $XZ = 2X \cdot Z$, $YZ = 2Y \cdot Z$.

Numbers of operations for Hessian and Weierstrass curves (see [1])

	Projective (Hessian)	Extended (Hessian)	Projective (Weierstrass)
Addition	12M	6M + 5S	12M + 2S
Doubling	6M + 3S	3M + 6S	5M + 6S
Tripling	8M + 6S	—	—
Reduction to scalar coords.	1I + 2M	1I + 2M	1I + 1M
x in Weierstrass form	1I + 1M	1I + 1M	—

4.2. Edwards and Twisted Edwards curves

An Edwards curve $E_{Edw,c,d}(GF(p))$ is given by

$$x^2 + y^2 = c^2(1 + dx^2y^2), \text{ where } c, d \in GF(p), cd(1 - c^4d) \neq 0. \quad (3)$$

Group operation is given by

$$(x_1, y_1) + (x_2, y_2) = \left(\frac{x_1y_2 + x_2y_1}{c(1 + dx_1x_2y_1y_2)}, \frac{y_1y_2 - x_1x_2}{c(1 - dx_1x_2y_1y_2)} \right),$$

neutral element of the group is $(0, c)$. An inverse to (x, y) is the point $(-x, y)$. The point $(0, -1)$ has order 2, the points $(\pm 1, 0)$ have order 4.

A twisted Edwards curve $\bar{E}_{Edw,a,d}(GF(p))$, where $a \neq 1$, is given by

$$ax^2 + y^2 = 1 + dx^2y^2,$$

where $a, d \in GF(p)$, $a, d \neq 0$.

Group operation on a twisted Edwards curve is given by

$$(x_1, y_1) + (x_2, y_2) = \left(\frac{x_1y_2 + x_2y_1}{1 + dx_1x_2y_1y_2}, \frac{y_1y_2 - ax_1x_2}{1 - dx_1x_2y_1y_2} \right),$$

neutral element of the group is $(0, 1)$. An inverse to (x, y) is the point $(-x, y)$.

The possibilities of (birationally) expressing a Weierstrass curve in Edwards form are given by the following conjecture.

Lemma 1 (see [5] for proofs and explicit formulas). *Consider Montgomery curves:*

$$M_{A,B} = \{(x, y) : By^2 = x^3 + Ax + x\}. \quad (4)$$

We have:

- if $p \equiv 3 \pmod{4}$, then any Montgomery curve is birationally equivalent to a Edwards curve over $GF(p)$;
- any twisted Edwards curve $\bar{E}_{Edw,a,d}$ is birationally equivalent to a Montgomery curve $M_{A,B}$ over $GF(p)$.

A point $(x, y) \in E_{Edw,c,d}(GF(p))$ is represented in projective coordinates by a triple $(X : Y : Z)$ such that $(X^2 + Y^2)Z^2 = Z^4 + dX^2Y^2$, $(x, y) = (X/Z, Y/Z)$. Neutral element has coordinates $(0 : 1 : 1)$. An inverse to $(X : Y : Z)$ is the point $(-X : Y : Z)$.

A point $(x, y) \in E_{Edw,c,d}(GF(p))$ is represented in *inverted coordinates* by a triple $(X : Y : Z)$ such that $(X^2 + Y^2)Z^2 = c(Z^4 + dX^2Y^2)$, $(x, y) = (Z/X, Z/Y)$, $XYZ \neq 0$. Neutral element has coordinates $(c : 0 : 0)$. An inverse to $(X : Y : Z)$ is the point $(-X : Y : Z)$. This representation has four exceptional points: $(0, 1)$, $(0, -1)$, $(1, 0)$, $(-1, 0)$.

A point $(x, y) \in \overline{E}_{Edw,a,d}(GF(p))$ is represented in *extended coordinates* by a quadruple $(X : Y : Z : T)$ such that $(aX^2 + Y^2)Z^2 = Z^4 + dX^2Y^2$, $(x, y) = (X/Z, Y/Z)$, $x \cdot y = T/Z$. Neutral element has coordinates $(0 : 1 : 1 : 1)$. An inverse to $(X : Y : Z : T)$ is the point $(-X : Y : Z : -T)$.

Numbers of operations for Edwards and twisted Edwards curves (see [1])

	Projective	Inverted	Extended (twisted curves)
Addition	$10M + 1S$	$9M + 1S$	$9M$
Doubling	$3M + 4S$	$3M + 4S$	$4M + 4S$
Tripling	$9M + 4S$	$9M + 4S$	—
Reduction to scalar coords.	$1I + 1M$	$1I + 1M$	$1I + 1M$
x in Weierstrass form	$1I + 2M$	$1I + 2M$	$1I + 2M$

4.3. GOST R 34.10 and special representations of elliptic curves

It should be noticed that GOST R 34.10 strictly requires to produce a component r of a signature as an X -coordinate of a point represented in affine (short) Weierstrass form, therefore, we should take into account an overhead caused by the two changes of coordinates. For the curve shapes considered in this paper, these are birational equivalences, each with the complexity at most $2(1I + 5M)$ per point for a transformation from affine Weierstrass form, and most $1I + 5M$ for the calculation of an x -coordinate. Moreover, the possibility of re-implementation of existing practical cryptosystems by using Hessian and Edwards curves is limited because of the non-trivial small cofactors of their orders (as pointed out before, any Hessian curve has a point of order 3, while any Edwards curve has a point of order 4).

5. Experiments and Analysis

We have built a C++ implementation of the GOST R 34.10 scheme using all the improvements described above and measured its performance on an ordinary CPU.

We have experimented with the following elliptic curves, both over the fields of characteristics of bit-lengths 256 and 512:

- Edwards curves of orders $m = 4q$, where q is of bit-length 255 and 509, respectively;
- Weierstrass curves, equivalent to those above;
- twisted Edwards curves of orders $m = 4q$, where q is of bit-length 255 and 509, respectively;
- Hessian curves of orders $m = 3q$, where q is of bit-length 255 and 510, respectively.

We have used algorithms of the *wmbNAF* and *jwmbNAF* classes with optimal parameters, as described in section 3.

In the following table we list the best timings in milliseconds acquired for our implementation (signature generation/verification), built with Intel C++ Composer XE 2011 using gmp 5.0.2 multi-precision arithmetics library, on a single core of an Intel Xeon 3.0GHz CPU.

Curve representation	256 bit- p	512bit- p
Projective, Weierstrass	1.12/1.43	4.23/5.23
Extended, Hessian	0.7/0.98	2.65/3.54
Projective, Edwards	0.66/0.98	2.36/3.37
Inverted, Edwards	0.7/0.98	2.56/3.48
Extended, Twisted Edwards	0.7/0.98	2.52/3.38

Thus, both Hessian and Edwards representations show a 40 % improvement over Weierstrass form. Nevertheless, practical differences in performance of various forms of Edwards curves are marginal. While Hessian representation is generally slower, it may perform better in a restricted environment because of lower total degree of the terms of addition formulas.

We conclude that the relative deterioration in performance of the extended GOST R 34.10 scheme compared to the standard currently in force may be made as small as 3.5 – 4, considering optimal implementations.

References

1. Explicit-formulas Database. — <http://hyperelliptic.org/EFD>, 2012.
2. GOST R 34.10-2001. Cryptographic data security. Signature and verification processes of [electronic] digital signature (in Russian). — 2001.
3. GOST R 34.10-____. Cryptographic data security. Signature and verification processes of [electronic] digital signature (draft, edition 1) (in Russian). — <http://www.infotecs.ru/laws/gost/proj/gost3410.pdf>, 2011.

4. GOST R 34.11-____. Cryptographic data security. Hash function (draft, edition 1) (in Russian). — <http://www.infotecs.ru/laws/gost/proj/gost3411.pdf>, 2011.
5. *Bernstein, D., Birkner, P., Joye, M., Lange, T. and Peters, C.* Twisted Edwards curves. — <http://eprint.iacr.org/2008/013>, 2008.
6. *Bernstein, D. and Lange, T.* Inverted Edwards coordinates. — <http://eprint.iacr.org/2007/410>, 2007.
7. *Doche, C., Kohel, D. R., Sica, F.* Double-base number systems for multi-scalar multiplications // Proc. EUROCRYPT 2009. — Lect. Notes Comput. Sci. — 2009. — V. 5479. — P. 502–519.
8. *Joye, M. and Quisquater, J.-J.* Hessian elliptic curves and side-channel attacks // Proc. CHES 2001. — Lect. Notes Comput. Sci. — 2001. — V. 2162. — P. 402–410.
9. *Kalinin, M.* An Implementation of the GOST R 34.10 Digital signature scheme using novel algorithms of elliptic curve scalar multiplication. Diploma paper (in Russian) // Moscow State Univ., 2010. — 50 p.
10. *Longa, P.* Accelerating the scalar multiplication on elliptic curve cryptosystems over prime fields. — <http://eprint.iacr.org/2008/100>, 2008.