Co-Z Divisor Addition Formulae in Jacobian of Genus 2 Hyperelliptic Curves over Prime Fields

Vladislav Kovtun
Chair of Information Security
National Aviation University
Kiev, Ukraine
vladislav.kovtun@nrjetix.com

Sergey Kavun
Department of Computer Systems and Technologies
Kharkov National University of Economic
Kharkov, Ukraine
kavserg@gmail.com

Abstract—**in this paper we proposed a new approach to divisor scalar multiplication in Jacobian of genus 2 hyperelliptic curves over fields with odd characteristic, without field inversion. It is based on improved addition formulae of the weight 2 divisors in projective divisor representation in most frequent case that suit very well to scalar multiplication algorithms based on Euclidean addition chains.**

Keywords-hyperelliptic curve, divisor, Jacobian, addition formulae, exponentiation, projective representation

I. INTRODUCTION

Since the introduction in publications of Miller and Koblitz the application of algebraic curves in cryptography was given its impulse to start [1, 2]. In these publications the usage of the property of the elliptic curve (EC) points to form the additive Abelelian group was suggested. In his later research [3], Koblitz has proved the possibility of using more complex curves – hyperelliptic (HEC). For HEC the group (Jacobian) of more complex structures should be considered – divisors instead of the curve points. As referred to in [3], HEC have a variety of advantages over EC: being richer source of the Abelian groups [3, 4] (forming the Abelian group, the size of which is defined by product of the base field size by a curve genus). Within the long time HEC cryptosystems were restricted to academic interest only, having no practical application due to high complexity of software and hardware implementation, low performance, absence of in-depth studies in the field of cryptoanalysis of such cryptosystems and absence of comprehensible algorithms of cryptosystem parameters generation [3, 4]. The active research [3-18] of HEC has allowed to overcome the majority of the described difficulties. The given research is devoted to the issues of further efficiency improvement of genus 2 HEC cryptosystems over prime fields.

The authors of publications [7, 9-18], offer variety of approaches which increase the performance of HEC cryptosystems essentially, making them highly competitive with EC cryptosystems.

II. BACKGROUND

Let's observe basic concepts of cryptosystems on HEC. More detailed information can be obtained from [3, 4].

Let K be a field and $\bar{K}$ be the algebraic closure of K. *Hyperelliptic curve* (HEC) C of genus $g \geq 1$ over K is a set of points (u, v) that satisfy the equation:

$$C: v^2 + h(u)v = f(u),\ k[u, v], \qquad (1)$$

where $h(u) \in k[u]$ is a polynomial of degree at most g, $f(u) \in k[u]$ is a monic polynomial of degree $2g+1$, and there are no solutions $(u, v) \in \bar{K} \times \bar{K}$ which simultaneously satisfy the equation (1) and the partial derivative equations $2u + h(u) = 0$, $h'(u)v - f'(u) = 0$.

In case of genus 2 HEC, polynomials $f(u)$ and $h(u)$ may be represented as $f(u) = u^5 + f_4u^4 + f_3u^3 + f_2u^2 + f_1u + f_0$ and $h(u) = h_2u^2 + h_1u + h_0$, $h_i, f_j \in K$.

Divisor D is a formal sum of points in C:

$$D = \sum_{P \in C} m_P P,\ m_P \in \mathbf{Z},$$

Where only a finite number of the m_P are non zero.

Divisor $D \in \mathbf{D}^0$ is a *principal divisor*, if $D = \mathrm{div}(R)$ for some rational function $R \in \bar{K}(C)^*$. The set of all principal divisors, denotes $P_C(\bar{K}) = \{\mathrm{div}(F) : F \in \bar{K}(C)\}$, in curve C over $\bar{K}$, moreover $P_C(\bar{K})$ is a subgroup of $\mathbf{D}^0$. Generally $P(C) = P_C(\bar{K})$ is called a group of principal divisors of *curve* C. The quotient group $J_C(\bar{K}) = \mathrm{div}_C^0(\bar{K}) / P_C(\bar{K})$ is called the *Jacobian of the curve* C *over* $\bar{K}$. The quotient group $J(C) = \mathrm{div}^0(C) / P(C)$ is called *Jacobian of the curve* C.

Further, we will operate with divisors in Mumford representation [4] $D = \left(x^2 + u_1x + u_0, v_1x + v_0\right)$, $\deg v < \deg u \leq 2$,

$u \mid f(u) - h(u)v - v^2$, where $\forall D_i \in J(C)$, $weight(D_i) = 2$, $i = \overline{1,2}$ the result $D_3 = D_1 + D_2$ will have a $weight(D_3) = 2$, this allows to save several additional verifications of common addition algorithm. Assume $\mathbf{GF}(p)$ is a base field, where p is an odd prime.

HECC uses a divisor scalar multiplication operation:

$$\underbrace{D + D + \cdots + D}_{k} = k \cdot D .$$

At the intermediate computation phase of most popular binary scalar multiplication algorithm performs divisor addition and doubling operation. The addition and doubling algorithms use field $\mathbf{GF}(p)$ inversion. Inversion is the most computationally intensive and space critical operation. Projective divisor representation [11, 12] is the most popular approach which allows to save a field inversion.

Divisor $D = \left(x^2 + u_1 x + u_0, v_1 x + v_0\right)$ in projective representation may be represented as $D = [U_1, U_0, V_1, V_0, Z]$, where $D = \left(x^2 + U_1/Z\, x + U_0/Z, V_1/Z\, x + V_0/Z\right)$.

At this moment, the most efficient types are arithmetic in projective representation, described in the paper [20] and in weighted representation, described in [17].

III. Co-Z Approach

Paper [22] gives a new impulse to the increase of efficiency of point scalar multiplication in EC over $\mathbf{GF}(p)$. The author has proposed transformation of EC points to projective and modified Jacobi representation with the same denominator and further operation with points of identical Z-coordinates. This approach is called a Co-Z in literature. For the implementation of scalar multiplication on the basis of idea [22], algorithms described in [22] should be used, based on Euclidian addition chains approach and scalar in Zeckendorf representation in order to replace doublings by Fibonacci numbers computations, refer to. Algorithm A.1.

Apply Co-Z approach to the divisor addition algorithm in projective representation. As a basis we should use divisor addition algorithm proposed in [15] and improved in [20] (Algorithm A.2).

Algorithm A.1 Fibonacci-and-add (k, P)

Input: $D \in J_C$, $k = (d_l, \ldots, d_2)_Z$

Output: $[k]D \in J_C$

begin
$(U, V) \leftarrow (D, D)$
for $i = l - 1$ downto 2
if $d_i = 1$ then $U \leftarrow U + D$ (add step)
$(U, V) \leftarrow (U + V, U)$ (Fibonacci step)
end
return U
end

On the assumption that $Z_1 = Z_2 = Z$, for D_1 and D_2, algorithm A.2 can be transformed to the algorithm A.3. Let's describe all modifications in A.2.

Algorithm A.2. Addition reduced divisors		
Input:	$[U_{11}, U_{10}, V_{11}, V_{10}, Z_1]$, $[U_{21}, U_{20}, V_{21}, V_{20}, Z_2]$	
Output:	$[U_1', U_0', V_1', V_2', Z'] = [U_{11}, U_{10}, V_{11}, V_{10}, Z_1] +$ $+ [U_{21}, U_{20}, V_{21}, V_{20}, Z_2]$, $weight(D_1) = weight(D_2) = 2$	
#	**Expression**	**Cost**
1	Precomputations: $Z = Z_1 \cdot Z_2$, $\tilde{U}_{21} = Z_1 \cdot U_{21}$, $\tilde{U}_{20} = Z_1 \cdot U_{20}$, $\tilde{V}_{21} = Z_1 \cdot V_{21}$, $\tilde{V}_{20} = Z_1 \cdot V_{20}$	5M
2	Compute resultant r for u_1 and u_2: $y_1 = U_{11} \cdot Z_2 - \tilde{U}_{21}$, $y_2 = \tilde{U}_{20} - U_{10} \cdot Z_2$, $y_3 = U_{11} \cdot y_1 + y_2 \cdot Z_1$, $r = y_2 \cdot y_3 + y_1^2 \cdot U_{10}$	1S, 6M
3	Compute almost inverse $inv = r/u_2 \bmod u_1$, $inv = inv_1 x + inv_0$: $inv_1 = y_1$, $inv_0 = y_3$	
4	Compute $s = (v_1 - v_2)\, inv \bmod u_1$, $s = s_1 x + s_0$: $w_0 = V_{10} \cdot Z_2 - \tilde{V}_{20}$, $w_1 = V_{11} \cdot Z_2 - \tilde{V}_{21}$, $w_2 = inv_0 \cdot w_0$, $w_3 = inv_1 \cdot w_1$, $s_0 = w_2 - U_{10} \cdot w_3$, $s_1 = (inv_0 + Z_1 \cdot inv_1) \cdot (w_0 + w_1) - w_2 - w_3 \cdot (Z_1 + U_{11})$ If $s_1 = 0$ then consider special case	8M
5	Precomputations: $R = r \cdot Z$, $s_2 = s_0 \cdot Z$, $s_3 = s_1 \cdot Z$, $\tilde{R} = R \cdot s_3$, $w_0 = s_1 \cdot s_0$, $w_1 = s_1 \cdot s_3$, $w_2 = s_0 \cdot s_3$, $w_3 = w_1 \cdot \tilde{U}_{21}$, $w_4 = R \cdot s_1$	9M
6	Compute $l = su_2$, $l = x^3 + l_2 x^2 + l_1 x + l_0$: $l_0 = w_0 \cdot \tilde{U}_{20}$, $l_2 = w_3 + w_2$, $l_1 = (w_1 + w_0) \cdot (\tilde{U}_{21} + \tilde{U}_{20}) - l_0 - w_3$	2M
7	Compute $u' = \left(s(l + h + 2v_1) - k\right) u_1^{-1}$, $k = \left(f - v_1 h - v_1^2\right)/u_1$, $u' = x^2 + u_1' x + u_0'$: $\tilde{U}_1' = 2w_2 - s_3 \cdot s_1 y_1 + h_2 \tilde{R} - R^2$ $\tilde{U}_0' = s_2^2 + s_1 \cdot y_1 \cdot (s_1 \cdot U_{11} - 2s_2) + y_2 \cdot w_1 + 2w_4 \cdot \tilde{V}_{21} + h_1 \tilde{R} +$ $+ R \cdot \left[h_2 (s_2 - s_1 U_{11}) + r \cdot (y_1 + 2\tilde{U}_{21} - f_4 Z)\right]$	2S, 8M
8	Adjust: $U_0' = \tilde{U}_0' \cdot \tilde{R}$, $U_1' = \tilde{U}_1' \cdot \tilde{R}$, $Z' = s_3^2 \cdot \tilde{R}$	1S, 3M
9	Compute $v' \equiv -(h + s_1 l + v_2) \bmod u'$, $v' = v_1' x + v_0'$: $V_1' = \tilde{U}_1' \cdot (l_2 - \tilde{U}_1' + h_2 \tilde{R}) + s_3^2 \cdot (\tilde{U}_0' - h_0 \tilde{R} - w_4 \tilde{V}_{21} - l_1)$, $V_0' = \tilde{U}_0' \cdot (l_2 - \tilde{U}_1' + h_2 \tilde{R}) - s_3^2 \cdot (l_0 + h_2 \tilde{R} + w_4 \cdot \tilde{V}_{20})$	5M
		4S, 46M

Step A.2.1. This step is to be omitted due to existence of the same denominator of all coordinates, which allows to save 5 multiplications in $\mathbf{GF}(p)$.

Step A.2.2. While computation of y_1 and y_2, reduction to common denominator of coordinates U_{1j} and U_{2j} is not required, saves 2 multiplications in $\mathbf{GF}(p)$.

Step A.2.4. While computation of w_1 and w_2, reduction to common denominator of coordinates V_{1j} and V_{2j} is also not required, saves 2 multiplications in $\mathbf{GF}(p)$.

Step A.2.5. The number of precomputations can be decreased by 3 multiplications in $\mathbf{GF}(p)$ due to reorder

computation of coefficients l_0, l_1 and l_2 of polynomial l on a step A.2.6.

Algorithm A.3. Co-Z reduced divisors addition	
Input:	$[U_{11}, U_{10}, V_{11}, V_{10}, Z]$, $[U_{21}, U_{20}, V_{21}, V_{20}, Z]$
Output:	$[U_1', U_0', V_1', V_2', Z'] = [U_{11}, U_{10}, V_{11}, V_{10}, Z] +$ $+[U_{21}, U_{20}, V_{21}, V_{20}, Z]$, $weight(D_1) = weight(D_2) = 2$

#	Expression	Cost
1	Compute resultant r of u_1, u_2: $y_1 = U_{11} - U_{21}$, $y_2 = U_{20} - U_{10}$, $y_3 = U_{11} \cdot y_1 + y_2 \cdot Z$, $r = y_2 \cdot y_3 + y_1^2 \cdot U_{10}$	1S, 4M
2	Compute almost inverse $inv = r/u_2 \bmod u_1$, $inv = inv_1 x + inv_0$: $inv_1 = y_1$, $inv_0 = y_3$	
3	Compute $s = (v_1 - v_2)\, inv \bmod u_1$, $s = s_1 x + s_0$: $w_0 = V_{10} - V_{20}$, $w_1 = V_{11} - V_{21}$, $s_0 = y_3 \cdot w_0 - U_{10} \cdot y_1 \cdot w_1$ $s_1 = y_2 Z \cdot w_1 + y_1 \cdot Z \cdot w_0$, If $s_1 = 0$ then consider special case	6M
4	Precomputations: $R = r \cdot Z$, $s_2 = s_0 \cdot Z$, $s_3 = s_1 \cdot Z$, $\tilde{R} = R \cdot s_3$, $w_3 = s_1 \cdot y_1$, $w_5 = w_3 + s_1 \cdot U_{21}$	6M
5	Compute $l = su_2$, $l = x^3 + l_2 x^2 + l_1 x + l_0$: $l_0 = s_0 \cdot U_{20}$, $l_2 = s_1 U_{21}$, $l_1 = (s_1 + s_0) \cdot (U_{21} + U_{20}) - l_0 - l_2$, $l_2 = l_2 + s_2$	2M
6	Compute $u' = (s(l + h + 2v_1) - k) u_1^{-1}$, $k = (f - v_1 h - v_1^2)/u_1$, $u' = x^2 + u_1' x + u_0'$: $\tilde{U}_1' = s_3 \cdot (2s_2 - w_3 + h_2 R) - R^2$ $\tilde{U}_0' = s_2^2 + w_3 \cdot (w_5 - 2s_2) + s_3 \cdot (y_2 \cdot s_1 + 2r \cdot V_{21} + h_1 R) +$ $+R \cdot [h_2 (s_2 - w_5) + r \cdot (U_{11} + U_{21} - f_4 \cdot Z)]$	2S, 8M
7	Adjust: $U_0' = \tilde{U}_0' \cdot \tilde{R}$, $U_1' = \tilde{U}_1' \cdot \tilde{R}$, $Z' = s_3^2 \cdot \tilde{R} (= s_3^3 \cdot r \cdot Z)$	1S, 3M
8	Compute $v' \equiv -(h + s_1 l + v_2) \bmod u'$, $v' = v_1' x + v_0'$: $V_1' = \tilde{U}_1' \cdot ((l_2 + h_2 R) \cdot s_3 - \tilde{U}_1') + s_3^2 \cdot (\tilde{U}_0' - s_3 \cdot (h_1 R + r V_{21} + l_1))$, $V_0' = \tilde{U}_0' \cdot ((l_2 + h_2 R) s_3 - \tilde{U}_1') - s_3^2 \cdot s_3 \cdot (l_0 + h_0 R + r \cdot V_{20})$	8M
		4S, 37M
9	Adjust: $Z_c = s_3^3 \cdot r$, $U_{20} = U_{20} \cdot Z_c$, $U_{21} = U_{21} \cdot Z_c$, $V_{20} = V_{20} \cdot Z_c$, $V_{21} = V_{21} \cdot Z_c$	5M
		4S, 42M

Step A.2.6. Unlike algorithm A.2, the A.3 offers considering multiplier s_3, present in each coefficient l_0, l_1 and l_2 of polynomial l, when using coefficients l_i, $i = \overline{0,2}$ on the steps A.2.7 and A.2.9. This allows to factor out s_3, thus saves 3 multiplications in $\mathbf{GF}(p)$. Consider next the application of proposed algorithm for the mixed divisor addition D_1 and D_2, such as $[U_{11}, U_{10}, V_{11}, V_{10}, Z]$ and $[U_{21}, U_{20}, V_{21}, V_{20}, 1]$ (affine representation). Therefore it is necessary to reduce devisor D_2 to common Z-coordinate, i.e. $[U_{21} \cdot Z, U_{20} \cdot Z, V_{21} \cdot Z, V_{20} \cdot Z, Z]$ the action requires 4 multiplications in $\mathbf{GF}(p)$. Hereinafter the provided algorithm A.3 should be used for addition of (prior formed) divisors with the same Z-coordinate.

It is to be considered that after computing $D_3 = D_1 + D_2$ one of the items, for example D_2, should be transformed to the same Z-coordinate with a divisor D_3. For this purpose, at the step A.3.9, value Z_c should be computed, where $Z_3 = Z \cdot Z_c$, i.e. $Z_c = s_3^3 \cdot r$, which requires 1 field multiplication. In other words, divisor transformation to common Z-coordinate requires 5 field multiplications. Ultimately, for the divisor addition step in A.1 42M+4S (M - multiplication, S - squaring) field operations are required. For the Fibonacci step 46M+4S field operations are required. In accordance to the computational complexity estimation [20], the approach, described in this paper, is not effective, because complexity of mixed addition is 39M+4S. Thus the alternative approach to the divisor addition for the scalar multiplication implementation is proposed. Let's draw a computational complexity comparison between scalar multiplication algorithms described in [20] and those suggested in this paper, based on idea [22].

IV. Comparison with other methods

Assume that S=0,8M and scalar multiplier is a 80-bit integer and refer to [20, 22] for the complexity of scalar multiplication algorithms. In Table 1 the computational complexity of the compared algorithms is represented.

TABLE I. Computational complexity of scalar multiplication algorithms

#	Scalar multiplication algorithm	Cost, M
	Addition in mixed coordinates [20]	
1	Binary (left-to-right)	5192
2	NAF	4630
3	w-NAF, $w = 4$	4350
	Co-Z addition	
4	Fibonacci-and-add	6773
5	Window Fibonacci-and-add	5970

Computational complexity of Fibonacci-and-add scalar multiplication algorithm has 23% more than Binary left-to-right algorithm and 13% then Window Fibonacci-and-add .

V. Summary

In the paper new algorithm of weight 2 divisor addition with and identical (shared) Z-coordinates (by the Co-Z approach) has been proposed, which requires more $\mathbf{GF}(p)$ operations then algorithm [20], however allows to decrease computational complexity of Fibonacci-and-add scalar multiplication algorithm while approaching to the Binary left-to-right algorithm.

References

[1] N. Koblitz. Elliptic curve cryptosystems. Mathematics of Computation, 48(177), 1987, pp. 203–209.

[2] I. V. S. Miller. Use of elliptic curves in cryptography. In H. C. Williams, editor, Advances in Cryptology - CRYPTO'85, volume 218 of LNCS, Springer, 1985, pp. 417–426.

[3] N. Koblitz. Hyperelliptic cryptosystems. Journal of cryptology, No 1. 1989. pp.139-150.

[4] Menezes A., Wu Y.H., Zuccherato R. An elementary introduction to hyperelliptic curves / In: Koblitz N. ed. // Algebraic aspects of cryptography. –Berlin, Heidelberg, New York: Springer–Verlag, 1998. -pp. 28–63.

[5] Cantor D.G. Computing in Jacobian of a Hyperelliptic curve // Mathematics of Computation. –Vol. 48. -No.177. -1987. -pp.95–101.

[6] Gaudry P., Harley R. Counting points on hyperelliptic curves over finite fields / In W. Bosma, ed. // ANTS IV. –LNCS 1838. –Berlin: Springer–Verlag, 2000. –pp.297–312.

[7] 140. Harley R. Fast arithmetic on genius two curves. –2000. Available at: http://cristal.inria.fr/harley/hyper/, adding.txt and doubling.c

[8] Jacobson M. (Jr), Menezes A., Stain A. Hyperelliptic curves and cryptography // Report of Fields Institute Communications. –Vol.7. –2002. –28p.

[9] Lange T. Efficient arithmetic on hyperelliptic curves: PhD thesis: Mathematics and Informatics. –University of Essen: Institute for experimental mathematics. – Germany: Essen, 2001. –122p.

[10] 143. Matsuo K., Chao J., Tsujii S. Fast genius two hyperelliptic curve cryptosystem // Technical report IEICE. –ISEC2001–31. –IEICE`2001. –2001. –8p.

[11] Miyamoto Y., Doi H., Matsuo K., Chao J., Tsujii S. A fast addition algorithm of genius two hyperelliptic curve // In the 2002 Symposium on cryptography and information security. –SCIS`2002. Japan: IEICE, 2002. –pp.497–502. (In Japanese)

[12] Takahashi M. Improving Harley algorithms for jacobians of genius 2 hyperelliptic curves // In the 2002 Symposium on cryptography and information security. –SCIS`2002. Japan: IEICE, 2002. –pp.155–160. (In Japanese)

[13] Sugizaki H., Matsuo K., Chao J., Tsujii S. An extension of Harley addition algorithm for hyperelliptic curves over finite fields of characteristic two // Technical report IEICE. –ISEC2002–09. –IEICE`2002. –2002. –8p.

[14] Lange T. Efficient arithmetic on genius 2 hyperelliptic curves over finite fields via explicit formulae // Cryptology ePrint Archive. –Report 2002/121. –2002. –13p. Available http://eprint.iacr.org.

[15] Lange T. Inversion–free arithmetic on genius 2 hyperelliptic curves // Cryptology ePrint Archive. –Report 2002/147. –2002. –7p. Available http://eprint.iacr.org.

[16] Lange T. Formulae for arithmetic on genius 2 hyperelliptic curves. September 2003. Available http://www.ruhr–uni–bochum.de/itsc/tanja/preprints/expl_sub.pdf.

[17] Lange T. Weighted coordinates on genius 2 hyperelliptic curves // Cryptology ePrint Archive. –Report 2002/153. –2002. –20p. Available http://eprint.iacr.org.

[18] Wollinger T. Software and hardware implementation of hyperelliptic curve cryptosystems: PhD dissertation: Electronics and informatics. –Worchester Polytechnic Institute. –Germany: Bochum, 2004. –218p.

[19] Chudnovsky D.V., Chudnovsky G.V. Sequence of number generated by addition in formal group and new primality and factorization test // Advanced in Applied Math. –№8. –1986. –pp.385–434.

[20] Ковтун В.Ю., Збитнев С.И. Арифметические операции в якобиане гиперэллиптической кривой рода 2 в проективных координатах с уменьшенной сложностью // Восточно–Европейский журнал передовых технологий. –2004. –Вып. №½ (13). –С. 14–22.

[21] Cohen H., Miyaji A., Ono T. Efficient elliptic curve exponentiation using mixed coordinates // Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security: Advances in Cryptology. –CRYPTO`98. –LNCS 1514. –Berlin: Springer–Verlag, 1998. –pp. 51–65.

[22] N. Meloni. New point addition formul. for ECC applications. In C. Carlet and B. Sunar, editors, Arithmetic of Finite Fields (WAIFI 2007), LNCS 4547, Springer, 2007, pp. 189–201.