SCIS 2002 The 2002 Symposium on Cryptography and Information Security
Shirahama,Japan, Jan.29-Feb.1,2002
The Institute of Electronics, Information and Communication Engineers

Fast Genus Three Hyperelliptic Curve Cryptosystems

Junichi KUROKI [*] Masaki GONDA [*] Kazuto MATSUO [†] Jinhui CHAO [‡]
Shigeo TSUJII [*]

Abstract— This paper presents a fast addition algorithm for the divisor class groups of genus three hyperelliptic curves. This algorithm improves the most recently proposed Harley algorithm for genus three hyperelliptic curves, which have brought up a noticeable progress since the well known Cantor algorithm. In this paper, we extend the Harley algorithm to genus three curves. The computational cost of the proposed algorithm is $I + 81M$ for an addition and $I + 74M$ for a doubling. (Here I and M denote the cost of an inversion and a multiplication on the definition fields.) We also show the implementation of the algorithm on Alpha 21264 / 667MHz, which takes 932 μs for a 160bit scalar multiplication on a divisor class group.

Keywords: Hyperelliptic curve cryptosystems, Hyperelliptic curves, Addition algorithms, Cantor algorithm, Harley algorithm

1 Introduction

In hyperelliptic curve cryptosystems, until recently the Cantor [Can87] algorithm has been known as the only fast algorithm for addition in the Jacobian varieties. This algorithm used the so-called Mumford's representation for divisors [Mum84] and based on a functional field analogy of composition and reduction of quadratic forms. Since the Cantor algorithm was applied in cryptosystems, numerous researches have been reported on improvement or speed-up of the original Cantor algorithm. [Kob89, PS98, SSI98, SS98, Sma99, Nag00].

A noticeable progress was made recently by the Harley algorithm for genus two hyperelliptic curves [GH00, Har00a]. This algorithm, while also with on Mumford's representation, calculates the divisor addition in a similar way to the chord-tangent law for addition of Mordel-Weil groups of elliptic curves rather than using the quadratic forms of polynomials. Furthermore, this algorithm made extensive use of fast algorithms such as the Chinese remainder theorem, the Newton iteration, and the Karatsuba multiplication for polynomial multiplication as well. As a result, the Harley algorithm costs much less computation time than the Cantor algorithm. It is known that the original Cantor algorithm costs $3I + 70M$ for an addition, $3I + 76M$ for a doubling [Nag00]. (Hereafter the I and M stand for the cost of an inversion and a multiplication on the definition field.) The Harley algorithm only needs $2I + 27M$ for an addition and $2I + 30M$ for a doubling. The Harley algorithm is improved in [MCT01] which costs $2I + 25M$ for an addition and $2I + 27M$ for a doubling.

In this paper, we extend the Harley algorithm for the genus two hyperelliptic curves to genus three hyperelliptic curves. It is known that when a genus three curve is used, to obtain a cryptographically secure Jacobian variety, e.g. in a size of 160bits, a definition field of 55bits will be enough. Therefore, if a 64bit CPU is used there is no need to use multiple precision calculations.

For a genus three hyperelliptic curve, the original Cantor algorithm needs $4I + 200M$ for an addition and $4I + 207M$ for a doubling [Nag00]. The algorithm proposed in this paper costs $I + 81M$ for an addition and $I + 74M$ for a doubling. Moreover, the algorithm is implemented on Alpha 21264 / 667MHz. The time for an addition is 4.27 μs, a doubling 4.09 μs and a 160bit scalar multiplication 932 μs.

2 Preliminary

Let p be an odd prime other than 7, n a positive integer, such that $q = p^n$. A genus three hyperelliptic curve C over $\mathbb{F}_q$ is defined as

$$\begin{aligned} C : Y^2 &= F(X), &(1)\\ F(X) &= X^7 + f_6X^6 + f_5X^5 + \cdots + f_0. &(2)\end{aligned}$$

Here, $f_i \in \mathbb{F}_q, \mathrm{disc}(F) \neq 0$.
By birational transformation

$$(X, Y) \mapsto \left(X + \frac{f_6}{7}, Y\right)$$

the C can be transformed into a hyperelliptic curve with $f_6 = 0$ which is isomorphic to C over $\mathbb{F}_q$. Hereafter, we assume $f_6 = 0$.

[*] Department of Information and System Engineering, Chuo University, 1-13-27 Kasuga, Bunkyo-ku, Tokyo, 112-8551 Japan
[†] Research and Development Initiative, Chuo University, 42-8 Ichigaya Honmura-cho, Shinjuku-ku, Tokyo, 162-8473 Japan
[‡] Department of Electrical, Electronic, and Communication Engineering, Chuo University, 1-13-27 Kasuga, Bunkyo-ku, Tokyo, 112-8551 Japan

We denote for a point $P = (x, y)$ on C, $-P = (x, -y)$, and the infinite point of C as P_∞, and $-P_\infty = P_\infty$. The points in the form of $(x, 0)$ will be called as a ramification point.
A divisor $\mathcal{D}$ of C is defined as following finite formal sum of the rational points of C.

$$\mathcal{D} = \sum_{P_i \in C} \mathrm{ord}_{P_i}(\mathcal{D}) P_i,\ \mathrm{ord}_{P_i}(\mathcal{D}) \in \mathbb{Z}. \tag{3}$$

Here, P_i is points on C. It is known that the set of divisors $\mathfrak{D}$ of C forms an abelian group.

The degree of a divisor $\mathcal{D}$ is defined as follows.

$$\deg \mathcal{D} = \sum_i \mathrm{ord}_{P_i}(\mathcal{D}) \tag{4}$$

The set of degree zero divisors $\mathfrak{D}^0$ forms a subgroup of $\mathfrak{D}$.
Let f be a rational function on C, the divisor (f) is defined as

$$(f) = \sum_i v_{P_i}(f) P_i. \tag{5}$$

Here P_i are zeros or poles of f on C with the multiplicity $v_{P_i}(f)$. The (f) is usually called as a principal divisor. The principal divisors $\mathfrak{D}^l$ form another subgroup of $\mathfrak{D}^0$.
The Jacobian variety $\mathcal{J}_\mathcal{C}$ of C is defined as

$$\mathcal{J}_\mathcal{C} = \mathfrak{D}^0 / \mathfrak{D}^l. \tag{6}$$

When two divisors $\mathcal{D}_1, \mathcal{D}_2 \in \mathfrak{D}^0$ are linearly equivalent in $\mathcal{J}_\mathcal{C}$, we denote $\mathcal{D}_1 \sim \mathcal{D}_2$.
An element $\mathcal{D}$ in $\mathcal{J}_\mathcal{C}$ can be expressed as follows.

$$\mathcal{D} = \sum_i \mathrm{ord}_{P_i}(\mathcal{D}) P_i - \big(\sum_i \mathrm{ord}_{P_i}(\mathcal{D})\big) P_\infty,\ \mathrm{ord}_{P_i}(\mathcal{D}) \geq 0 \tag{7}$$

Here for all i, j, $P_i \neq -P_j$.
Such divisors are called semi-reduced divisors.
$\sum_i \mathrm{ord}_{P_i}(\mathcal{D})$ is called as the weight of $\mathcal{D}$ [GH00].
In particular, a semi-reduced divisor whose weight is no more than the genus is called a reduced divisor. It is known that the elements of $\mathcal{J}_\mathcal{C}$ can be uniquely represented by reduced divisor. Therefore, we will use the reduced divisor in addition of $\mathcal{J}_\mathcal{C}$. Further, it is known that a semi-reduced divisor can be represented by the following pair of polynomials [Mum84].

Definition 2.1. (Mumford's representation)

$$\mathcal{D} = (U, V),\ U, V \in \bar{\mathbb{F}}_q[X], \tag{8}$$
$$U = \prod (X - x_i)^{\mathrm{ord}_{P_i}(\mathcal{D})},\ P_i = (x_i, y_i), \tag{9}$$
$$F - V^2 \equiv 0 \bmod U,\ \deg V < \deg U. \tag{10}$$

This polynomial representation of semi-reduced divisors is called Mumford's representation [GH00]. The genus three curves, a divisor $\mathcal{D}$ is reduced if $\deg U \leq 3$.

3 Harley algorithm

In this section, we give a brief outline of the Harley algorithm [GH00]. This algorithm is proposed for fast addition of divisor class groups of genus two hyperelliptic curves, which generalized the chord-tangent law of addition of the rational points on elliptic curves. A main feature of this algorithm is a detailed classification of the cases according to the properties of summand divisors. Then the computation procedures are optimized for each case respectively in order to avoid redundance and reach the highest efficiency. Besides, it makes extensive uses of fast algorithms such as the Chinese remainder theorem, the Newton iteration and the Karatsuba multiplication of polynomials.
Take an example of $\mathcal{D}_3 = (U_3, V_3) = \mathcal{D}_1 + \mathcal{D}_2$. Then in the first place, divisor additions are classified into different cases according to the degree of U_1, U_2 in the Mumford's representations of $\mathcal{D}_1 = (U_1, V_1)$ and $\mathcal{D}_2 = (U_2, V_2)$. Then subcases are classified according to whether U_1 and U_2 have a nontrivial common divisor by calculation of the resultant.
Below, we only describe the most frequent case when the U_1 and U_2 are coprime. In this case, one at first computes a divisor $\mathcal{D} = (U, -V)$ which is linearly equivalent to the sum divisor $\mathcal{D}_3$. Then $U = U_1 U_2$ and V can be found from (11) by the Chinese remainder theorem.

$$\begin{aligned} &V \equiv V_1 \bmod U_1, \\ &V \equiv S U_1 + V_1 \bmod U,\ S \in \mathbb{F}_q[X]. \end{aligned} \tag{11}$$

Here,

$$S \equiv \frac{V_2 - V_1}{U_1} \bmod U_2. \tag{12}$$

Now $\mathcal{D}$ is reduced to find $\mathcal{D}_3$. In doubling, the Newton iteration is used in the place of the Chinese remainder theorem.
This Harley algorithm in the most frequent case needs $2I + 27M$ for an addition, $2I + 30M$ for a doubling. Further details of the Harley algorithm are referred to [GH00, Har00a, MCT01].

4 Fast addition algorithm for genus three hyperelliptic curves

In this section, we show an extension of the Harley algorithm to genus three hyperelliptic curves.

4.1 Main Algorithm

As already mentioned in the section three, the Harley algorithm classifies the summand divisors into different cases according to the weights of both divisors and whether they have common factors. In that way, redundance is minimized for each subcase to raise the efficiency of whole computation. On the other hand, for genus three case, the same strategy will lead to 6 cases by the classification of weights. Further classification according to common divisors will result in about 70 subcases. To develop different procedures for all these subcases will be inefficient considering the size of the program code and the complexity of control functions for these conditional decisions.

Furthermore, in the additions on genus three hyperelliptic curves, the pairs of reduced summand divisors are almost always both with weight three and coprime to each other. In the doubling, the most frequent case is doubling of a reduced divisor with weight three and without ramification points. Hence, we will use Harley's strategy for such the most frequent cases. While for other cases, the Cantor algorithm will be used. The main flow of the algorithm for addition and doubling is shown in Table 1 and Table 2 respectively.

Input	Reduced-divisor $\mathcal{D}_1 = (U_1, V_1)$, $\mathcal{D}_2 = (U_2, V_2)$
Output	Reduced divisor $\mathcal{D}_3 = (U_3, V_3) = \mathcal{D}_1 + \mathcal{D}_2$

Step	Procedure
1	IF weights of $\mathcal{D}_1$,$\mathcal{D}_2$ are three then
2	Compute $r = \text{resultant}(U_1, U_2)$;
	IF $r \neq 0$ then
	Compute $\mathcal{D}_3$
	by frequent case addition algorithm.
3	Compute $\mathcal{D}_3$ by Cantor algorithm.

Table 1: Addition (Main flow)

Input	Reduced-divisor $\mathcal{D}_1 = (U_1, V_1)$
Output	Reduced divisor $\mathcal{D}_2 = (U_2, V_2) = 2\mathcal{D}_1$

Step	Procedure
1	If weight of $\mathcal{D}_1$ is three then
2	Compute $r = \text{resultant}(U_1, V_1)$;
	IF $r \neq 0$ then
	Compute $\mathcal{D}_2$
	by frequent case doubling algorithm.
3	Compute $\mathcal{D}_2$ by Cantor algorithm.

Table 2: Doubling (Main flow)

4.2 The most frequent case algorithm

Below, we show the algorithm for the most frequent case. This algorithm also combined the techniques used to improve the original Harley algorithm for genus two case [MCT01, MDMCT02]. The outline and computation costs for each step are shown in Table 3 and Table 4, for the addition and for the doubling respectively.

In the genus two case, both addition and doubling need once reduction to obtain the output divisor $\mathcal{D}_3 = (U_3, V_3)$. In the genus three case, however, both need twice reductions. To take the addition as an example, by the first reduction (in the Step 7, 8) one obtains $\mathcal{D}_t = (U_t, V_t)$. Since when $s_2 \neq 0$,

$$U_t = s_2^{-2}((S^2U_1 + 2SV_1)/U_2 - (F - V_1^2)/(U_1U_2)) \quad (13)$$

is a monic polynomial of the fourth degree. By definition $\mathcal{D}_t = (U_t, V_t)$ is just a semi-reduced divisor.
Thus another reduction (in the Step 9, 10) is needed to obtain reduced $\mathcal{D}_3 = (U_3, V_3)$.
Here the U_t is computed with the method shown in [MCT01]. Besides, in the step 6 of the Table 3 and the step 7 in the Table 4, the multiple inversion technique [Coh93] is used.
The overall cost for the most frequent case is then $I + 81M$ for an addition, $I + 74M$ for a doubling.

5 Implementation and comparison

The proposed algorithm is implemented on Alpha 21264 / 667MHz. The definition field is $\mathbb{F}_p$ where $p = 2^{61}-1$. The computation timings for addition, doubling and scalar multiplication are shown in Table 5. The scalar multiplication used sliding-window with window width 4 [MvOV97], the scalar is chosen as a random 160bit integer. We use NTL [Sho01] for the implementation of the Cantor algorithm.

addition	doubling	scalar mul.
4.27 μs.	4.09 μs.	932 μs.

Table 5: Performance results on Alpha 21264 / 667MHz

Acknowledgement

The authors wish to thank Dr. Kazumaro Aoki for valuable suggestion on this research.
A part of this research was supported by Telecommunications Advancement Organization of Japan (TAO).

Input	Weight three coprime reduced divisors $\mathcal{D}_1 = (U_1, V_1)$ and $\mathcal{D}_2 = (U_2, V_2)$	
Output	A weight three reduced divisor $\mathcal{D}_3 = (U_3, V_3) = \mathcal{D}_1 + \mathcal{D}_2$	
Step	Procedure	Cost
1	Compute the resultant r of U_1 and U_2.	$18M$
2	If $r = 0$ then $\mathcal{D}_1$ and $\mathcal{D}_2$ have a linear factor in common, and call Cantor algorithm.	—
3	Compute $I_1 = i_{12}X^2 + i_{11}X + i_{10} \equiv r/U_1 \bmod U_2$.	$3M$
4	Compute $rS = rs_2X^2 + rs_1X + rs_0 \equiv (V_2 - V_1)I_1 \bmod U_2$. (Karatsuba)	$11M$
5	If $rs_2 = 0$ then $\mathcal{D}_3$ should be weight two or one, and call the exclusive procedure.	—
6	Compute $S = s_2X^2 + s_1X + s_0 = r^{-1}(rs_2X^2 + rs_1X + rs_0)$.	$I + 7M$
7	Compute $U_t = s_2^{-2}((S^2U_1 + 2SV_1)/U_2 - (F - V_1^2)/(U_1U_2))$.	$19M$
8	Compute $V_t = -(SU_1 + V_1) \bmod U_t$. (Karatsuba)	$12M$
9	Compute $U_3 = (F - V_t^2)/U_t$.	$8M$
10	Compute $V_3 = -V_t \bmod U_3$. (Karatsuba)	$3M$
Total		$I + 81M$

Table 3: Addition for weight three coprime divisors on a genus three HEC (Frequent case addition algorithm)

Input	A weight three reduced divisor $\mathcal{D}_1 = (U_1, V_1)$ without ramification points	
Output	A weight three reduced divisor $\mathcal{D}_2 = (U_2, V_2) = 2\mathcal{D}_1$	
Step	Procedure	Cost
1	Compute the resultant r of U_1 and V_1.	$15M$
2	If $r = 0$ then $\mathcal{D}_1$ is with a ramification point, and call Cantor algorithm.	—
3	Compute $I_1 = i_{12}X^2 + i_{11}X + i_{10} \equiv r/V_1 \bmod U_1$.	$3M$
4	Compute $T_1 = t_{12}X^2 + t_{11}X + t_{10} \equiv (F - V_1^2)/U_1 \bmod U_1$.	$7M$
5	Compute $2rS = 2rs_2X^2 + 2rs_1X + 2rs_0 \equiv I_1T_1 \bmod U_1$. (Karatsuba)	$11M$
6	If $2rs_2 = 0$ then $\mathcal{D}_3$ should be weight two or one, and call the exclusive procedure.	—
7	Compute $S = s_2X^2 + s_1X + s_0 = (2r)^{-1}(2rs_2X^2 + 2rs_1X + 2rs_0)$.	$I + 7M$
8	Compute $U_t = s_2^{-2}(((SU_1 + V_1)^2 - F)/U_1^2)$.	$9M$
9	Compute $V_t = -(SU_1 + V_1) \bmod U_t$. (Karatsuba)	$12M$
10	Compute $U_2 = (F - V_t^2)/U_t$.	$7M$
11	Compute $V_2 = -V_t \bmod U_2$. (Karatsuba)	$3M$
Total		$I + 74M$

Table 4: Doubling for weight three reduced divisor on a genus three HEC (Frequent case doubling algorithm)

References

[BP98] D.V. Bailey and C.Paar, *Optimal extension fields for fast arithmetic in public-key algorithms*, Advances in Cryptology - CRYPTO'98 (H.Krawczyk, ed.), Lecture Notes in Computer Science, no. 1462, Springer-Verlag, 1998, pp.472–485.

[BSS99] I.Blake, G.Seroussi, and N.Smart, *Elliptic curves in cryptography*, London Mathematical Society Lecture Note Series, no. 265, Cambridge U. P., 1999.

[Can87] D.G. Cantor, *Computing in the Jacobian of hyperelliptic curve*, Math. Comp. **48** (1987), no.177, 95–101.

[CF96] J.W.S. Cassels and E.V. Flynn, *Proglegomena to middlebrow arithmetic of curves of genus 2*, London Mathematical Society Lecture Note Series, no. 230, Cambridge U. P., 1996.

[Coh93] H.Cohen, *A Course in Computational Algebraic Number Theory*, GTM 138, Springer, 1993, pp.481

[GCL92] K.O. Geddes, S.R. Czapor, and G.Labahn, *Algorithms for computer algebra*, Kluwer Academic Pub., 1992.

[GG99] J.Gathen and J.Gerhard, *Modern computer algebra*, Cambridge U. P., 1999.

[GH00] P.Gaudry and R.Harley, *Counting points on hyperelliptic curves over finite fields*, ANTS-IV (W.Bosma, ed.), Lecture Notes in Computer Science, no. 1838, Springer-Verlag, 2000, pp.297–312.

[Har00a] R.Harley, *adding.text , doubling.c*, `http://cristal.inria.fr/harley/hyper/`, 2000.

[Knu98] D.E. Knuth, *The art of computer programming*, 3rd ed., vol. 2 Seminumerical Algorithms, Addison Wesley, 1998.

[Kob89] N.Koblitz, *Hyperelliptic curve cryptosystems*, J. Cryptology **1** (1989), no.3, 139–150.

[Kob98] N.Koblitz, *Algebraic aspects of cryptography*, Algorithms and Computation in Mathematics, no.3, Springer-Verlag, 1998.

[MCT01] K.Matsuo,J.Chao,S.Tsujii, *Fast Genus Two Hyperelliptic Curve Cryptosystems*, ISEC2001-31,IEICE 2001.

[MDMCT02] Y.Miyamoto,H.Doi,K.Matsuo,J.Chao, S.Tsujii, *A Fast Addition Algorithm of Genus Two Hyperelliptic Curves*, SCIS, 2002.

[Mum84] D.Mumford, *Tata lectures on theta II*, Progress in Mathematics, no.43, Birkhäuser, 1984.

[MvOV97] A.Menezes, P.van Oorschot, and S.Vanstone, *Handbook of applied cryptography*, CRC Press, 1997.

[Nag00] K.Nagao, *Improving group law algorithms for Jacobians of hyperelliptic curves*, ANTS-IV (W.Bosma, ed.), Lecture Notes in Computer Science, no. 1838, Springer-Verlag, 2000, pp.439–448.

[PS98] S.Paulus and A.Stein, *Comparing real and imaginary arithmetics for divisor class groups of hyperelliptic curves*, ANTS-III, Lecture Notes in Computer Science, no. 1423, Springer-Verlag, 1998, pp.576–591.

[Sho01] V.Shoup, *A tour of NTL*, `http://www.shoup.net/NTL/`, 2001.

[Sma99] N.Smart, *On the performance of hyperelliptic cryptosystems*, Advances in Cryptology - EUROCRYPTO'99, Lecture Notes in Computer Science, no. 1592, Springer-Verlag, 1999, pp.165–175.

[Spa94] A.M. Spallek, *Kurven vom geshlcht 2 und ihre anwendung in public-key-kryptosystemem*, Ph.D. thesis, GH Essen, 1994.

[SS98] Y.Sakai and K.Sakurai, *Design of hyperelliptic cryptosystems in small characteristic and a software imprementation over F_{2^n}*, Advances in Cryptology - ASIACRYPT'98 (K.Ohta and D.Pei, eds.), Lecture Notes in Computer Science, no. 1514, Springer-Verlag, 1998, pp.80–94.

[SSI98] Y.Sakai, K.Sakurai, and H.Ishizuka, *Secure hyperelliptic cryptosystems and their performance*, Public Key Cryptography (H.Imai and Y.Zheng, eds.), Lecture Notes in Computer Science, no. 1431, Springer-Verlag, 1998, pp.164–181.