SCIS 2004 The 2004 Symposium on Cryptography and Information Security
Sendai, Japan, Jan.27-30, 2004
The Institute of Electronics, Information and Communication Engineers

Improvements of addition algorithm on genus 3 hyperelliptic curves and their implementations*

Masaki Gonda† Kazuto Matsuo‡ Kazumaro Aoki§ Jinhui Chao¶ Shigeo Tsujii†

Abstract— Genus 3 hyperelliptic curve cryptosystems are able to carry out fast-encryption on a 64-bit CPU because a 56-bit field is enough for their definition fields. Recently, Kuroki et al. proposed an extension of the Harley algorithm, which had been known as the fastest addition algorithm of divisor classes on genus 2 hyperelliptic curves, on genus 3 hyperelliptic curves and Pelzl et al. improved the algorithm. This paper shows an improvement of the Harley algorithm on genus 3 hyperelliptic curves using Toom's multiplication. The proposed algorithm takes only $I + 70M$ $(I + 71M)$ for an addition (resp. a doubling) instead of $I + 76M$ (resp. $I + 74M$), which is the best possible of the previous works, where I and M denote the required time for an inversion and a multiplication over the definition field respectively. This paper also shows 2 variations of the proposed algorithm in order to adjust the algorithm to various platforms. Moreover the proposed algorithms are implemented on a 64-bit CPU. The implementation results show a 160-bit scalar multiplication can be done within 176μs on a 64-bit CPU Alpha EV68 1.25GHz.

Keywords: genus 3 hyperelliptic curves, hyperelliptic curve cryptosystems, addition algorithm, Cantor algorithm, Harley algorithm, Toom's multiplication

1 Introduction

Hyperelliptic curve cryptosystems have been able to use in practice since a fast addition algorithm of divisor classes on hyperelliptic curves (Cantor algorithm) was proposed by Cantor [Can87].

Recently, Harley [GH00, Har00] proposed a new fast addition algorithm on genus 2 hyperelliptic curves (Harley algorithm). In the same fashion as the Cantor algorithm, the Harley algorithm uses Mumford's representation [Mum84] for input and output divisor classes, but uses a method similar to chord-tangent law in an elliptic curve instead of quadratic form computations for polynomials used in the Cantor algorithm. Moreover it uses modern polynomial computation techniques i.e. Chinese remainder theorem, Newton's iteration, and Karatsuba's multiplication. Consequently the Harley algorithm is faster than the Cantor algorithm. Besides [MCT01] showed that hyperelliptic curve cryptosystems using the Harley algorithm are theoretically able to reach the same performance as elliptic curve cryptosystems and a lot of researches on the Harley algorithm has been proposed afterwords [MC01, MDM+02, Tak02, KGM+02, SMCT02, Lan02a, Lan02b, Lan02c, TM02, TMM03, PWGP03, SMCT03, PWP03b, MS03, PWP03a, KK03, Ava03]. Especially Kuroki et al. [KGM+02] proposed an extension of the Harley algorithm on genus 3 hyperelliptic curves over odd characteristic fields and Pelzl et al. [PWGP03] showed its improvement and generalization for arbitrary characteristic fields.

Genus 3 hyperelliptic curve cryptosystems can be constructed on 56-bit fields from the Hasse-Weil range [Sti93], even if Thériault's recent improvement [Thé03] of Gaudry's attack [Gau00] is taken into account. Therefore their implementation on 64-bit CPUs does not need multi-precision arithmetic and they are able to carry out fast-encryption.

This paper proposes improvements of the Harley algorithm on genus 3 hyperelliptic curves from [KGM+02] and [PWGP03] and shows implementation results of the proposed algorithms. Moreover this paper discusses finite field arithmetic suitable for genus 3 hyperelliptic curve cryptosystems to take the properties of the definition field and the platform chosen in this paper into consideration.

In this paper, A, I, M, denote the required time of computing $a + b$, $1/a$ and ab for the definition field elements a, b respectively. To take the properties of the definition field and the platform chosen in this paper into account, the required time for $-a$, $a - b$, $2a$ are also denoted as A, moreover, the required time for a^2 is also denoted as M.

2 Genus 3 hyperelliptic curves and their divisor class groups

Let n be a positive integer, $p \neq 2, 7$ be a prime number, and $q = p^n$. A genus 3 hyperelliptic curve C over $\mathbb{F}_q$ is defined as follows:

$$C \ : \ Y^2 = F(X)$$
$$F(X) = X^7 + f_5X^5 + f_4X^4 + \cdots + f_0 \in \mathbb{F}_q[X]$$

with $\mathrm{disc}\,(F) \neq 0$.

* This research was partially supported by Telecommunications Advancement Organization of Japan (TAO).
† Dept. of Information and System Engineering, Chuo University
‡ The Research and Development Initiative of Chuo University
§ NTT Information Sharing Platform Laboratories
¶ Dept. of Electrical, Electronic, and Communication Engineering, Chuo University

The divisor class group $\mathcal{J}_C(\mathbb{F}_q)$ of C forms a finite Abelian group and therefore cryptosystems based on discrete logarithm problems can be constructed on C. Any equivalent class $\mathcal{D}$ in $\mathcal{J}_C(\mathbb{F}_q)$ can be represented by Mumford's representation defined as follows [Mum84].

Definition 1 (Mumford's representation).

$$\begin{aligned} \mathcal{D} &= (U, V),\ U, V \in \mathbb{F}_q[X],\ \textit{where} \\ U &= \prod (X - x_i)^{\mathrm{ord}_{P_i}(\mathcal{D})}, \\ y_i &= V(x_i) \end{aligned}$$

for $P_i = (x_i, y_i) \in C$ *with* $\mathrm{ord}_{P_i}(\mathcal{D}) > 0$, *and*

$$\deg V < \deg U,\ F - V^2 \equiv 0 \bmod U.$$

The degree of U is called the weight of $\mathcal{D}$ [GH00] and $\mathcal{D}$ a reduced divisor, if its weight equals 3. Any class in $\mathcal{J}_C(\mathbb{F}_q)$ is uniquely represented by a reduced divisor, i.e. each class includes unique reduced divisor.

3 Harley algorithm on genus 3 hyperelliptic curves

This section shows a brief overview of the Harley algorithm on genus 3 hyperelliptic curves according to [KGM+02].

The Harley algorithm uses reduced divisors represented by Mumford's representation for input and output divisor classes. In the algorithm, first one executes classification of the input divisor classes by using the weights of them. Then one executes another classification of the divisor classes by testing $\gcd(U_1, U_2) = 1$ for addition $\mathcal{D}_3 = \mathcal{D}_1 + \mathcal{D}_2$, $\mathcal{D}_1 = (U_1, V_1)$, $\mathcal{D}_2 = (U_2, V_2)$ or by testing $\gcd(U_1, V_1) = 1$ for doubling $\mathcal{D}_2 = 2\mathcal{D}_1$, $\mathcal{D}_1 = (U_1, V_1)$. These gcd computations are carried out by a resultant computation in practice. The cases classified above, i.e. the case satisfied $\deg U_1 = \deg U_2 = 3$ and $\gcd(U_1, U_2) = 1$ for addition and the case satisfied $\deg U_1 = 3$ and $\gcd(U_1, V_1) = 1$ for doubling, are called the most frequent cases. See [KGM+02] for details of the classification.

For genus 2 hyperelliptic curves, more precisely classification is executed and a different procedure is used for each case [Har00, MCT01, SMCT02]. However, for genus 3 hyperelliptic curves, there exist about 70 cases and therefore the classification needs large costs. On the other hand, the probability of occurring the cases except the most frequent case is $O(1/q)$ in both the addition and the doubling [Nag00], so that these cases can be negligible and an efficient strategy for genus 3 hyperelliptic curves is to take the Harley algorithm for only the most frequent cases and the Cantor algorithm for the other cases. Therefore we will consider only the most frequent cases hereafter.

```
Input: Genus 3 HEC $C : Y^2 = F(X)$, weight 3 reduced divisors $\mathcal{D}_1 = (U_1, V_1)$, $\mathcal{D}_2 = (U_2, V_2)$ with $\gcd(U_1, U_2) = 1$
Output: The weight 3 reduced divisor $\mathcal{D}_3 = (U_3, V_3) = \mathcal{D}_1 + \mathcal{D}_2$
1: Compute $\mathcal{D}_s \sim -\mathcal{D}_3$ such that $U_s = U_1 U_2$ from $\mathcal{D}_1$ and $\mathcal{D}_2$
2: Compute $\mathcal{D}_t \sim \mathcal{D}_3$ such that $\deg U_t = 4$ from $\mathcal{D}_s$
3: Compute $\mathcal{D}_3$ from $\mathcal{D}_t$
```

Algorithm 1: Harley addition algorithm on genus 3 HEC for the most frequent case

Algorithm 1 shows an overview of the Harley algorithm on genus 3 hyperelliptic curves for the most frequent case of addition $\mathcal{D}_3 = \mathcal{D}_1 + \mathcal{D}_2$.

Step 1 in Algorithm 1 is called composition part and Step 2 to 3 reduction part. The composition part is accomplished by using Chinese remainder theorem for polynomials and, in doubling, Newton's iteration is used instead of Chinese remainder theorem. Unlike for genus 2 hyperelliptic curves, the reduction part needs 2 steps for genus 3 hyperelliptic curves. The reduction part of the doubling is the same as the addition. In practice, Steps 1 and 2 are computed simultaneously and all detailed algorithm is written down into arithmetic over the definition fields. Moreover the multiplication and the inversion costs are reduced by using Karatsuba's multiplication [KO63] and Montgomery's multiple inversion technique [Coh93, Algorithm 10.3.4] respectively. In addition, [PWGP03] reduces the multiplication costs by applying Bézout's determinant to the resultant computations.

4 Improvements of the algorithm

This section shows improvements of the Harley algorithm on genus 3 hyperelliptic curves under the standard assumption $A \ll M \ll I$.

The essentials of the improvements are the following 3 points:

1. Using Toom's multiplication
2. Using virtual polynomial multiplication
3. Refining the details

The following discusses on using Toom's multiplication and virtual polynomial multiplication.

4.1 Toom's multiplication

Toom's multiplication is known as an efficient multiplication algorithm for high-degree polynomials [Too63, Ber01]. This algorithm is generally inefficient for low-degree polynomials such as appeared in the Harley algorithm. However, in certain cases of low-degree polynomial multiplication, the cost of Toom's multiplication is smaller than Karatsuba's one. For example, a multiplication of a degree 2 polynomial and a degree 1 polynomial can be done within $4M$ by Toom's multiplication as the following procedure in stead of $5M$ by Karatsuba's one.

```
Input: $R = r_2X^2 + r_1X + r_0$, $S = s_1X + s_0$
Output: $T = t_3X^3 + t_2X^2 + t_1X + t_0 = RS$
1: $w_1 = (r_2 + r_1 + r_0)(s_1 + s_0)$
2: $w_2 = (r_2 - r_1 + r_0)(-s_1 + s_0)$
3: $t_0 = r_0s_0$
4: $t_3 = r_2s_1$
5: $t_1 = (-2t_3 + w_1 - w_2)/2$
6: $t_2 = (-2t_0 + w_1 + w_2)/2$
```

Toom's multiplication is effective for at least 2 places in both the addition and the doubling procedure of the algorithm.

4.2 Virtual polynomial multiplication

There is a lot of "multiply-and-add" operations in the Harley algorithm on genus 3 hyperelliptic curves. For example, if there exist sequences such that

$$\cdots + s_1 z_4 + \cdots ,$$
$$\cdots + s_1 z_3 + s_0 z_4 + \cdots ,$$
$$\cdots + s_0 z_3 + \cdots ,$$

then these can be regarded as the polynomial multiplication $(s_1X + s_0)(z_4X + z_3)$ and Karatsuba's multiplication is applicable. Therefore their computations can be done within $3M$ instead of $4M$.

This trick is effective for at least 2 places in the addition procedure of the algorithm.

4.3 Results

Tables 5 and 6 show the addition procedure and the doubling procedure of the proposed algorithm respectively. In these tables, "(Toom)" and "(Karatsuba)" denote the step using Toom's and Karatsuba's multiplication respectively. The proposed algorithm takes $I + 70M + 113A$ for an addition and $I + 71M + 107A$ for a doubling respectively.

In practical usage of genus 3 hyperelliptic curve cryptosystems, the assumption used here, i.e. $A \ll M \ll I$, is not always satisfied. In the case, i.e. $A \not\ll M$ case, it seems to be more efficient that Karatsuba's one is used instead of Toom's multiplication, moreover, the classical one is used instead of both Toom's and Karatsuba's multiplication. These algorithms are easily obtained from Tables 5 and 6. Consequently we obtain 3 algorithms which can be adjusted to various platforms. The costs for the proposed algorithms and the previous works appear in Table 1.

Obviously, which algorithm is the fastest depends on M/A. Therefore we show the fastest algorithm for the addition and the doubling with respect to M/A in Table 2.

Note that many other variations of the algorithm can be obtained by partially using another method for each polynomial multiplication in Tables 5 and 6.

5 Implementation on a 64-bit CPU

We implement the proposed algorithms on Alpha EV68, which is known as a standard 64-bit CPU. The implementation uses Compaq C++ with inline assembler. The inline assembler is only used for the `umulh` instruction and the `cttz` instruction.

The following discusses

1. Choice of the definition field
2. Arithmetic over the definition field

for 64-bit CPUs and shows implementation results.

	Addition	Doubling
Previous works		
[KGM+02]	$I + 81M + 125A$	$I + 74M + 125A$
[PWGP03]	$I + 76M + 95A$	$I + 75M + 97A$
This work		
Toom	$I + 70M + 113A$	$I + 71M + 107A$
Karatsuba	$I + 72M + 111A$	$I + 73M + 101A$
Classical	$I + 79M + 83A$	$I + 78M + 83A$

Table 1: Results and comparison of the addition algorithms on genus 3 hyperelliptic curves. "Toom" denotes the algorithm shown in Tables 5 and 6, "Karatsuba" the algorithm without Toom's multiplication, "Classical" the algorithm used only the classical multiplication.

5.1 Definition field

Most of recent CPUs including Alpha EV68 is able to multiply (half-)word-size integers fast. Therefore using a prime finite field $\mathbb{F}_p$ as the definition field is efficient in order to make the most of this multiplication capability.

On the other hand, the most efficient attack against genus 3 hyperelliptic curve cryptosystems over $\mathbb{F}_p$ is Thériault's improvement [Thé03] of Gaudry's attack [Gau00] and its complexity is $O(p^{10/7})$. Therefore a 56-bit field is enough for the definition field in order to carry out the same security as a 160-bit elliptic curve cryptosystem.

It is known that choosing a Mersenne prime as p leads to fast $\mathbb{F}_p$-arithmetic [BP98, AHK00] and the only Mersenne prime from 56- to 64-bit is $2^{61} - 1$, so that we choose $p = 2^{61} - 1$ for our implementation.

5.2 Arithmetic over the definition field

The implementation of $\mathbb{F}_p$-arithmetic is basically due to [BP98, AHK00]. However certain arithmetic can be improved to take the properties of both $p = 2^{61} - 1$ and the Harley algorithm on genus 3 hyperelliptic curves into account. Therefore the following discusses the $\mathbb{F}_p$-arithmetic suitable for genus 3 hyperelliptic curve cryptosystems using the property of $p = 2^{61} - 1$.

Note that the symbol "&" denotes the "bit-wise and" operation in the algorithms described below.

Addition Addition algorithm over $\mathbb{F}_p$ is not always implemented with a lot of care. Because the condition $A \ll M$ is satisfied in most of the public-key cryptosystems implementation. However implementation of genus 3 hyperelliptic curve cryptosystems is able to make the most of the multiplication capability of CPUs as described above, so that it is expected that M is small and M/A for genus 3 hyperelliptic curve cryptosystems are smaller than usual public-key cryptosystems. Therefore we must pay attention to the implementation of addition over $\mathbb{F}_p$.

An addition procedure consists of two parts, i.e. an integer addition and a reduction modulo p. The reduction usually needs larger cost than the integer addition. However an efficient reduction procedure can be obtained from the properties of $p = 2^{61} - 1$. The following shows the addition algorithm over $\mathbb{F}_p$ used in our implementation.

```
Input: $a, b \in [0, p-1]$
Output: $c \in [0, p-1]$ such that $c \equiv a + b \bmod p$
1: $w = a + b + 1$
2: $c = \lfloor w/2^{61} \rfloor + (w \& p) - 1$
```

This algorithm reduces the cost for the reduction. Moreover this reduction trick makes multiple addition

M/A	≤ 3.2	3.3	3.4	≥ 3.5
Addition	Classical		Toom	
Doubling	Classical			Toom

Table 2: The fastest algorithm with respect to M/A.

efficiently. The following example shows an algorithm for multiple addition of 4 elements.

```
Input: $a, b, c, d \in [0, p-1]$
Output: $c \in [0, p-1]$ such that $c \equiv a+b+c+d \bmod p$
1: $w = a + b + c + d + 3$
2: $c = \lfloor w/2^{61} \rfloor + (w \& p) - 3$
```

Because there are many multiple addition in the Harley algorithm on genus 3 hyperelliptic curves, such kind of algorithm is efficient for speed-up the Harley algorithm. Moreover it is expected that these algorithms bring more efficient implementation on recent CPUs because these have a number of ALUs usually.

Multiplication The multiplication algorithm used in our implementation is basically due to [BP98]. In addition, to take 64-bit word boundaries into consideration, more efficient algorithm can be obtained. Moreover a reduction trick similar to the addition's one can be also applicable. The following shows the multiplication algorithm used in our implementation.

```
Input: $a, b \in [0, p-1]$
Output: $c \in [0, p-1]$ such that $c \equiv ab \bmod p$
1: $w = \lfloor \texttt{mulq}(2^3 a, b)/2^3 \rfloor$
2: $w = w + \texttt{umulh}(2^3 a, b)$
3: $c = \lfloor w/2^{61} \rfloor + (w \& p)$
```

In the algorithm, $\texttt{mulq}(a, b)$ ($\texttt{umulh}(a, b)$) denotes the function returns the low-order 64 bits (resp. the high-order 64 bits) of ab for 64-bit unsigned integers a, b.

Inversion As efficient inversion algorithm for single-precision arithmetic, there exist extended Euclidean algorithm (EGCD), extended binary GCD algorithm (EBGCD), and $(p-2)$-powering ($p-2$ method). We implemented each algorithm in order to decide which algorithm should be used and an experimental result shows the cost for EGCD > the cost for $p-2$ method $\approx 69M$ > the cost for EBGCD, so that we use EBGCD for the inversion.

The dominant part of the EBGCD is the "while-loop" shown as follows.

```
while $t_2$ is even do
    if $t_1$ is odd then
        $t_1 = t_1 + p$
    $t_1 = \lfloor t_1/2 \rfloor$
    $t_2 = \lfloor t_2/2 \rfloor$
```

This loop is called many times in an inversion. However the loop can be canceled using a property of $p = 2^{61} - 1$ as follows.

```
$w = \texttt{cttz}(t_2)$
$t_1 = \lfloor t_1/2^w \rfloor + 2^{61-w}(t_1 \& (2^w - 1))$
$t_2 = \lfloor t_2/2^w \rfloor$
```

In the above procedure, $\texttt{cttz}(a)$ denotes the function returns the position of the first bit to be "1" from the LSB to regard the bit-position of the LSB as 0. We use this trick in the inversion computation.

Arithmetic of multiple elements The Harley algorithm on genus 3 hyperelliptic curves needs not only multiple addition but also various arithmetic of multiple elements. The reduction trick used in the addition can be applied for these arithmetic. Therefore we make all the function that needs only at most 1 reduction with respect to the arithmetic appeared in the proposed algorithms. Table 3 shows the $\mathbb{F}_p$-arithmetic functions used in our implementation.

5.3 Implementation results

Table 4 shows implementation results. In the table, each of "Toom," "Karatsuba," "Classical" denotes the same algorithm shown in Table 1.

We use the signed sliding-window algorithm [BSS99, Algorithm IV.7] of the window-size 5 for the scalar multiplication and 160-bit random integers for the scalars, moreover, our implementation uses NTL/GMP [Sho03, Fou03] for the Cantor algorithm and the sliding-window algorithm. Each timing shows the average of every 1,000,000 operations on 100 random curves with F to be irreducible.

The results show the algorithm without both Toom's and Karatsuba's multiplication is the fastest. In practice, to regard an instruction as a step, the critical passes of the addition and the multiplication over $\mathbb{F}_p$ are 7 steps and 9 steps and therefore the assumption $A \ll M$ is obviously not satisfied. It is expected that such property increases with speed-up of the definition field arithmetic.

6 Conclusion

This paper showed improvements of the Harley algorithm on genus 3 hyperelliptic curves and obtained 3 algorithms which take $I + 70M$ ($I + 71M$) for an addition (resp. a doubling), $I + 72M$ (resp. $I + 73M$), and $I + 79M$ (resp. $I + 78M$) respectively. Under the standard assumptions, the cost $I + 70M$ ($I + 71M$) for an addition (resp. a doubling) is the best possible as far as we know.

Moreover this paper showed implementation results of the proposed algorithms on the 64-bit CPU Alpha EV68 1.25GHz. The implementation results show that an addition, a doubling, and a 160-bit scalar multiplication can be done within 888ns, 875ns, and 176μs respectively. This result is the fastest one for hyperelliptic curve addition implementation as far as we know.

$-a$	$2a$	$a/2$
$a+b$	$a-b$	$1/a$
a^2	ab	$a+b+c$
$2a+b$	$a+b+c+d$	$ab+c$
a^2+b	$ab+c+d$	a^2+b+c
a^2+2b	$ab+c+d+e$	$ab+cd$
$ab+cd+e$	$ab+cd+e+f$	$2ab+cd$
$2ab+cd+e$	a^2+2bc	$2(ab+c)+de$
$ab+cd+ef$	$ab+cd+ef+g$	$2ab+cd+ef$

Table 3: Implemented $\mathbb{F}_p$-arithmetic. $a, \ldots, g$ denote any elements in $\mathbb{F}_p$ or $\mathbb{F}_p^\times$ for the inverse.

	Addition	Doubling	Scalar mul.
Toom	930ns	933ns	187μs
Karatsuba	920ns	897ns	181μs
Classical	888ns	875ns	176μs

Table 4: Performance results on Alpha EV68 1.25GHz

In.	Genus 3 HEC $C : Y^2 = F(X)$, $F = X^7 + f_5X^5 + f_4X^4 + f_3X^3 + f_2X^2 + f_1X + f_0$; Reduced divisors $\mathcal{D}_1 = (U_1, V_1)$ and $\mathcal{D}_2 = (U_2, V_2)$, $U_1 = X^3 + u_{12}X^2 + u_{11}X + u_{10}$, $V_1 = v_{12}X^2 + v_{11}X + v_{10}$, $U_2 = X^3 + u_{22}X^2 + u_{21}X + u_{20}$, $V_2 = v_{22}X^2 + v_{21}X + v_{20}$;	
Out.	The reduced divisor $\mathcal{D}_3 = (U_3, V_3) = \mathcal{D}_1 + \mathcal{D}_2$, $U_3 = X^3 + u_{32}X^2 + u_{31}X + u_{30}$, $V_3 = v_{32}X^2 + v_{31}X + v_{30}$;	

Step	Procedure	Cost
1	Compute the resultant r of U_1 and U_2: $t_1 = u_{11}u_{20} - u_{10}u_{21}$; $t_2 = u_{12}u_{20} - u_{10}u_{22}$; $t_3 = u_{20} - u_{10}$; $t_4 = u_{21} - u_{11}$; $t_5 = u_{22} - u_{12}$; $t_6 = t_4^2$; $t_7 = t_3t_4$; $t_8 = u_{12}u_{21} - u_{11}u_{22} + t_3$; $t_9 = t_3^2 - t_1t_5$; $t_{10} = t_2t_5 - t_7$; $r = t_8t_9 + t_2(t_{10} - t_7) + t_1t_6$;	$14M + 12A$
2	If $r = 0$ then call the Cantor algorithm	–
3	Compute the pseudo-inverse $I = i_2X^2 + i_1X + i_0 \equiv r/U_1 \bmod U_2$: $i_2 = t_5t_8 - t_6$; $i_1 = u_{22}i_2 - t_{10}$; $i_0 = u_{21}i_2 - (u_{22}t_{10} + t_9)$;	$4M + 4A$
4	Compute $S' = s'_2X^2 + s'_1X + s'_0 = rS \equiv (V_2 - V_1)I \bmod U_2$ (Karatsuba, Toom): $t_2 = v_{10} - v_{20}$; $t_3 = v_{11} - v_{21}$; $t_4 = v_{12} - v_{22}$; $t_5 = t_3i_1$; $t_6 = t_2i_0$; $t_7 = t_4i_2$; $t_8 = u_{22}t_7$; $t_1 = t_5 + t_7 + t_8 - (t_3 + t_4)(i_1 + i_2)$; $t_3 = u_{20}t_1$; $t_4 = u_{20} + u_{22}$; $t_9 = (t_4 + u_{21})(t_1 - t_7)$; $t_{10} = (t_4 - u_{21})(t_1 + t_7)$; $s'_0 = -(t_3 + t_6)$; $s'_2 = t_7 - (s'_0 + t_5 + (t_2 + t_4)(i_0 + i_2) + (t_9 + t_{10})/2)$; $s'_1 = t_5 + t_6 + (t_{10} - t_9)/2 - (t_8 + (t_2 + t_3)(i_0 + i_1))$;	$10M + 31A$
5	If $s'_2 = 0$ then call the Cantor algorithm	–
6	Compute S, w and $w_i = 1/w$ s.t. $wS = S'/r$ and S is monic: $t_1 = (rs'_2)^{-1}$; $t_2 = rt_1$; $w = t_1s'^2_2$; $w_i = rt_2$; $s_0 = t_2s'_0$; $s_1 = t_2s'_1$;	$I + 7M$
7	Compute $Z = X^5 + z_4X^4 + z_3X^3 + z_2X^2 + z_1X + z_0 = SU_1$ (Toom): $t_6 = s_0 + s_1$; $t_1 = u_{10} + u_{12}$; $t_2 = t_6(t_1 + u_{11})$; $t_3 = (t_1 - u_{11})(s_0 - s_1)$; $t_4 = u_{12}s_1$; $z_0 = u_{10}s_0$; $z_1 = (t_2 - t_3)/2 - t_4$; $z_2 = (t_2 + t_3)/2 - z_0 + u_{10}$; $z_3 = u_{11} + s_0 + t_4$; $z_4 = u_{12} + s_1$;	$4M + 15A$
8	Compute $U_t = X^4 + u_{t3}X^3 + u_{t2}X^2 + u_{t1}X + u_{t0} = (S(Z + 2w_iV_1) - w_i^2((F - V_1^2)/U_1))/U_2$ (Karatsuba): $t_1 = s_0z_3$; $t_2 = (u_{22} + u_{21})(u_{t3} + u_{t2})$; $t_3 = u_{21}u_{t2}$; $t_4 = t_1 - t_3$; $u_{t3} = z_4 + s_1 - u_{22}$; $t_5 = s_1z_4 - u_{22}u_{t3}$; $u_{t2} = z_3 + s_0 + t_5 - u_{21}$; $u_{t1} = z_2 + t_6(z_4 + z_3) + w_i(2v_{12} - w_i) - (t_5 + t_2 + t_4 + u_{20})$; $u_{t0} = z_1 + t_4 + s_1z_2 + w_i(2(v_{11} + s_1v_{12}) + w_iu_{12}) - (u_{22}u_{t1} + u_{20}u_{t3})$;	$13M + 26A$
9	Compute $V_t = v_{t2}X^2 + v_{t1}X + v_{t0} \equiv wZ + V_1 \bmod U_t$: $t_1 = u_{t3} - z_4$; $v_{t0} = w(t_1u_{t0} + z_0) + v_{10}$; $v_{t1} = w(t_1u_{t1} + z_1 - u_{t0}) + v_{11}$; $v_{t2} = w(t_1u_{t2} + z_2 - u_{t1}) + v_{12}$; $v_{t3} = w(t_1u_{t3} + z_3 - u_{t2})$;	$8M + 11A$
10	Compute $U_3 = X^3 + u_{32}X^2 + u_{31}X + u_{30} = (F - V_t^2)/U_t$: $t_1 = 2v_{t3}$; $u_{32} = -(u_{t3} + v_{t3}^2)$; $u_{31} = f_5 - (u_{t2} + u_{32}u_{t3} + t_1v_{t2})$; $u_{30} = f_4 - (u_{t1} + v_{t2}^2 + u_{32}u_{t2} + u_{31}u_{t3} + t_1v_{t1})$;	$7M + 11A$
11	Compute $V_3 = v_{32}X^2 + v_{31}X + v_{30} \equiv V_t \bmod U_3$: $v_{32} = v_{t2} - u_{32}v_{t3}$; $v_{31} = v_{t1} - u_{31}v_{t3}$; $v_{30} = v_{t0} - u_{30}v_{t3}$;	$3M + 3A$
Total		$I + 70M + 113A$

Table 5: Explicit formula for addition on genus 3 HEC (most frequent case)

In.	Genus 3 HEC $C : Y^2 = F(X)$, $F = X^7 + f_5X^5 + f_4X^4 + f_3X^3 + f_2X^2 + f_1X + f_0$; A reduced divisor $\mathcal{D}_1 = (U_1, V_1)$, $U_1 = X^3 + u_{12}X^2 + u_{11}X + u_{10}$, $V_1 = v_{12}X^2 + v_{11}X + v_{10}$;	
Out.	The reduced divisor $\mathcal{D}_2 = (U_2, V_2) = 2\mathcal{D}_1$, $U_2 = X^3 + u_{22}X^2 + u_{21}X + u_{20}$, $V_2 = v_{22}X^2 + v_{21}X + v_{20}$;	

Step	Procedure	Cost
1	Compute the resultant r of U_1 and V_1: $t_1 = u_{11}v_{10} - u_{10}v_{11}$; $t_2 = u_{12}v_{10} - u_{10}v_{12}$; $t_3 = v_{11}^2$; $t_4 = v_{11}v_{10}$; $t_5 = v_{10} + u_{12}v_{11} - u_{11}v_{12}$; $t_6 = v_{10}^2 - v_{12}t_1$; $t_7 = v_{12}t_2 - t_4$; $r = t_5t_6 + t_2(t_7 - t_4) + t_1t_3$;	$14M + 9A$
2	If $r = 0$ then call the Cantor Algorithm	–
3	Compute the pseudo-inverse $I = i_2X^2 + i_1X + i_0 \equiv r/V_1 \bmod U_1$: $i_2 = t_3 - v_{12}t_5$; $i_1 = u_{12}i_2 + t_7$; $i_0 = u_{11}i_2 + u_{12}t_7 + t_6$;	$4M + 4A$
4	Compute $Z = z_2X^2 + z_1X + z_0 \equiv (F - V_1^2)/U_1 \bmod U_1$: $t_1 = 2u_{10}$; $t_2 = 2u_{11}$; $t_3 = u_{12}^2$; $t_4 = f_4 - (t_1 + v_{12}^2)$; $t_5 = f_5 + t_3 - t_2$; $t_{10} = 2v_{12}$; $z_2 = t_5 + 2t_3$; $z_1 = u_{12}(t_2 - t_5) + t_4$; $z_0 = f_3 + t_3(t_5 - u_{11}) + u_{12}(t_1 - t_4) + u_{11}(u_{11} - f_5) - t_{10}v_{11}$;	$7M + 18A$
5	Compute $S' = s'_2X^2 + s'_1X + s'_0 = 2rS \equiv ZI \bmod U_1$ (Karatsuba, Toom): $t_1 = i_1z_1$; $t_2 = i_0z_0$; $t_3 = i_2z_2$; $t_4 = u_{12}t_3$; $t_5 = (i_2 + i_1)(z_2 + z_1) - (t_1 + t_3 + t_4)$; $t_6 = u_{10}t_5$; $t_7 = u_{10} + u_{12}$; $t_8 = t_7 + u_{11}$; $t_9 = t_7 - u_{11}$; $t_7 = t_8(t_3 + t_5)$; $t_{11} = t_9(t_5 - t_3)$; $s'_2 = t_1 + t_6 + (i_2 + i_0)(z_2 + z_0) - (t_2 + t_3 + (t_7 + t_{11})/2)$; $s'_1 = t_4 + (i_0 + i_1)(z_1 + z_0) + (t_{11} - t_7)/2 - (t_1 + t_2)$; $s'_0 = t_2 - t_6$;	$10M + 28A$
6	If $s'_2 = 0$ then call the Cantor Algorithm	–
7	Compute S, w and $w_i = 1/w$ s.t. $wS = S'/(2r)$ and S is monic: $t_1 = 2r$; $t_2 = (t_1s'_2)^{-1}$; $t_3 = t_1t_2$; $w = t_2s'^2_2$; $w_i = t_1t_3$; $s_0 = t_3s'_0$; $s_1 = t_3s'_1$;	$I + 7M + A$
8	Compute $G = X^5 + g_4X^4 + g_3X^3 + g_2X^2 + g_1X + g_0 = SU_1$ (Toom): $t_1 = t_8(s_1 + s_0)$; $t_2 = t_9(s_0 - s_1)$; $t_3 = u_{12}s_1$; $g_0 = u_{10}s_0$; $g_1 = (t_1 - t_2)/2 - t_3$; $g_2 = u_{10} + (t_1 + t_2)/2 - g_0$; $g_3 = t_3 + u_{11} + s_0$; $g_4 = u_{12} + s_1$;	$4M + 12A$
9	Compute $U_t = X^4 + u_{t3}X^3 + u_{t2}X^2 + u_{t1}X + u_{t0} = ((G + w_iV_1)^2 - w_i^2F)/U_1^2$: $u_{t3} = 2s_1$; $u_{t2} = s_1^2 + 2s_0$; $u_{t1} = u_{t3}s_0 + w_i(t_{10} - w_i)$; $u_{t0} = s_0^2 + 2w_i((s_1 - u_{12})v_{12} + v_{11} + w_iu_{12})$;	$7M + 10A$
10	Compute $V_t = v_{t3}X^3 + v_{t2}X^2 + v_{t1}X + v_{t0} \equiv wG + V_1 \bmod U_t$: $t_1 = u_{t3} - g_4$; $v_{t0} = w(t_1u_{t0} + g_0) + v_{10}$; $v_{t1} = w(t_1u_{t1} + g_1 - u_{t0}) + v_{11}$; $v_{t2} = w(t_1u_{t2} + g_2 - u_{t1}) + v_{12}$; $v_{t3} = w(t_1u_{t3} + g_3 - u_{t2})$;	$8M + 11A$
11	Compute $U_2 = X^3 + u_{22}X^2 + u_{21}X + u_{20} = (F - V_t^2)/U_t$: $t_1 = 2v_{t3}$; $u_{22} = -(u_{t3} + v_{t3}^2)$; $u_{21} = f_5 - (u_{t2} + u_{22}u_{t3} + t_1v_{t2})$; $u_{20} = f_4 - (u_{t1} + v_{t2}^2 + u_{22}u_{t2} + u_{21}u_{t3} + t_1v_{t1})$;	$7M + 11A$
12	Compute $V_2 = v_{22}X^2 + v_{21}X + v_{20} \equiv V_t \bmod U_2$ (Karatsuba): $v_{22} = v_{t2} - u_{22}v_{t3}$; $v_{21} = v_{t1} - u_{21}v_{t3}$; $v_{20} = v_{t0} - u_{20}v_{t3}$;	$3M + 3A$
Total		$I + 71M + 107A$

Table 6: Explicit formula for doubling on genus 3 HEC (most frequent case)

References

[AHK00] K. Aoki, F. Hoshino, and T. Kobayashi, *The fastest ECC implementations*, Proc. of SCIS2000, B05, 2000, in Japanese.

[Ava03] R. M. Avanzi, *Aspects of hyperelliptic curves over large prime fields in software implementations*, Cryptology ePrint Archive, Report 2003/253, 2003.

[Ber01] D. Bernstein, *Multidigit multiplication for mathematicians*, 2001, manuscript, `ftp://koobera.math.uic.edu/pub/papers/m3.dvi`.

[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.), LNCS 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.

[Coh93] H. Cohen, *A course in computational algebraic number theory*, Graduate Text in Mathematics, no. 138, Springer-Verlag, 1993.

[Fou03] Free Software Foundation, *GNU MP*, `http://www.swox.com/gmp/manual/`, 2003.

[Gau00] P. Gaudry, *An algorithm for solving the discrete log problem on hyperelliptic curves*, Advances in Cryptology - EUROCRYPT2000 (B. Preneel, ed.), LNCS 1807, Springer-Verlag, 2000, pp. 19–34.

[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.), LNCS 1838, Springer-Verlag, 2000, pp. 313–332.

[Har00] R. Harley, *adding.text, doubling.c*, `http://cristal.inria.fr/~harley/hyper/`, 2000.

[KGM$^+$02] J. Kuroki, M. Gonda, K. Matsuo, J. Chao, and S. Tsujii, *Fast genus three hyperelliptic curve cryptosystems*, Proc. of SCIS2002, 2002, pp. 503–507.

[KK03] I. Kitamura and M. Katagi, *Efficient implementation of genus three hyperelliptic curve cryptography over $\mathbb{F}_{2^n}$*, Cryptology ePrint Archive, Report 2003/248, 2003.

[KO63] A. Karatsuba and Y. Ofman, *Multiplication of multidigit numbers on automata*, Soviet Physics Doklady **7** (1963), 595–596.

[Lan02a] T. Lange, *Efficient arithmetic on genus 2 hyperelliptic curves over finite fields via explicit formulae*, Cryptology ePrint Archive, Report 2002/121, 2002.

[Lan02b] ———, *Inversion-free arithmetic on genus 2 hyperelliptic curves*, Cryptology ePrint Archive, Report 2002/147, 2002.

[Lan02c] ———, *Weighted coordinates on genus 2 hyperelliptic curves*, Cryptology ePrint Archive, Report 2002/153, 2002.

[MC01] K. Matsuo and J. Chao, *Fast genus two hyperelliptic curve cryptosystems*, Talk at the 2nd workshop on cryptology and algebraic curves, Chuo U., `http://www.tsujii-lab.ise.chuo-u.ac.jp/hyper/pdf/mats_ohp0108.pdf`, 2001, in Japanese.

[MCT01] K. Matsuo, J. Chao, and S. Tsujii, *Fast genus two hyperelliptic curve cryptosystems*, Technical Report ISEC2001-31, IEICE Japan, 2001.

[MDM$^+$02] Y. Miyamoto, H. Doi, K. Matsuo, J. Chao, and S. Tsujii, *A fast addition algorithm of genus two hyperelliptic curves*, Proc. of SCIS2002, 2002, in Japanese, pp. 497–502.

[MS03] P. K. Mishra and P. Sarka, *Parallelizing explicit formula for arithmetic in the Jacobian of hyperelliptic curves*, Advances in Cryptology - ASIACRYPT2003 (C. S. Laih, ed.), LNCS 2894, Springer-Verlag, 2003, pp. 93–110.

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

[Nag00] K. Nagao, *Improving group law algorithms for Jacobians of hyperelliptic curves*, ANTS-IV (W. Bosma, ed.), LNCS 1838, Springer-Verlag, 2000, pp. 439–448.

[PWGP03] J. Pelzl, T. Wollinger, J. Guajardo, and C. Paar, *Hyperelliptic curve cryptosystems: Closing the performance gap to elliptic curves (update)*, Cryptology ePrint Archive, Report 2003/026, 2003.

[PWP03a] J. Pelzl, T. Wollinger, and C. Paar, *High performance arithmetic for hyperelliptic curve cryptosystems of genus two*, Cryptology ePrint Archive, Report 2003/212, 2003.

[PWP03b] ———, *Low cost security: Explicit formulae for genus 4 hyperelliptic curves*, Cryptology ePrint Archive, Report 2003/097, 2003.

[Sho03] V. Shoup, *A tour of NTL*, `http://www.shoup.net/ntl/`, 2003.

[SMCT02] H. Sugizaki, K. Matsuo, J. Chao, and S. Tsujii, *An extension of Harley addition algorithm for hyperelliptic curves over finite fields of characteristic two*, Technical Report ISEC2002-9, IEICE Japan, 2002.

[SMCT03] ———, *A generalized Harley algorithm for genus two hyperelliptic curves*, Proc. of SCIS2003, 2003, pp. 917–921.

[Sti93] H. Stichtenoth, *Algebraic function fields and codes*, Universitext, Springer-Verlag, 1993.

[Tak02] M. Takahashi, *Improving Harley algorithms for Jacobians of genus 2 hyperelliptic curves*, Proc. of SCIS2002, 2002, in Japanese, pp. 155–160.

[Thé03] N. Thériault, *Index calculus attack for hyperelliptic curves of small genus*, Advances in Cryptology - ASIACRYPT2003 (C. S. Laih, ed.), LNCS 2894, Springer-Verlag, 2003, pp. 75–92.

[TM02] N. Takahashi and M. Miyaji, *Efficient exponentiation on genus two hyperelliptic curves*, Technical Report ISEC2002-102, IEICE Japan, 2002, in Japanese.

[TMM03] N. Takahashi, H. Morimoto, and M. Miyaji, *Efficient exponentiation on genus two hyperelliptic curves (ii)*, Technical Report ISEC2002-145, IEICE Japan, 2003, in Japanese.

[Too63] A. L. Toom, *The complexity of a scheme of functional elements realizing the multiplication of integers*, Soviet Mathematics Doklady **3** (1963), 714–716.

[War03] H. S. Warren, Jr., *Hacker's delight*, Addison Wesley, 2003.

Improvements of addition algorithm on genus 3 hyperelliptic curves and their implementations — Update and Eratta —

M. Gonda, K. Matsuo, K. Aoki, J. Chao, and S. Tsujii

January 25, 2004

Update We made further improvements in our implementation, so please replace Table 4 by the following.

	Addition	Doubling	Scalar mul.
Toom	919ns	916ns	180μs
Karatsuba	920ns	897ns	177μs
Classical	888ns	875ns	172μs

Table 4: Performance results on Alpha EV68 1.25GHz

Eratta Please replace Step 4 in Table 5 by the following.

Step	Procedure	Cost
4	Compute $S' = s_2'X^2 + s_1'X + s_0' = rS \equiv (V_2 - V_1)I \bmod U_2$ (Karatsuba, Toom): $t_1 = v_{10} - v_{20}$; $t_2 = v_{11} - v_{21}$; $t_3 = v_{12} - v_{22}$; $t_4 = t_2 i_1$; $t_5 = t_1 i_0$; $t_6 = t_3 i_2$; $t_7 = u_{22} t_6$; $t_8 = t_4 + t_6 + t_7 - (t_2 + t_3)(i_1 + i_2)$; $t_9 = u_{20} + u_{22}$; $t_{10} = (t_9 + u_{21})(t_8 - t_6)$; $t_9 = (t_9 - u_{21})(t_8 + t_6)$; $s_0' = -(u_{20} t_8 + t_5)$; $s_2' = t_6 - (s_0' + t_4 + (t_1 + t_3)(i_0 + i_2) + (t_{10} + t_9)/2)$; $s_1' = t_4 + t_5 + (t_9 - t_{10})/2 - (t_7 + (t_1 + t_2)(i_0 + i_1))$;	$10M + 31A$

Table 5: Explicit formula for addition on genus 3 HEC (most frequent case)