Low Cost Security: Explicit Formulae for Genus 4 Hyperelliptic Curves

Jan Pelzl, Thomas Wollinger, and Christof Paar

Department of Electrical Engineering and Information Sciences
Communication Security Group (COSY)
Ruhr-Universitaet Bochum, Germany
{pelzl, wollinger, cpaar}@crypto.rub.de

Abstract. It is widely believed that genus four hyperelliptic curve cryptosystems (HECC) are not attractive for practical applications because of their complexity compared to systems based on lower genera, especially elliptic curves. Our contribution shows that for low cost security applications genus-4 hyperelliptic curves (HEC) can outperform genus-2 HEC and that we can achieve a performance similar to genus-3 HEC. Furthermore our implementation results show that a genus-4 HECC is an alternative cryptosystem to systems based on elliptic curves.
In the work at hand we present for the first time explicit formulae for genus-4 HEC, resulting in a 60% speed-up compared to the best published results. In addition we implemented genus-4 HECC on a Pentium4 and an ARM microprocessor. Our implementations on the ARM show that for genus four HECC are only a factor of 1.66 slower than genus-2 curves considering group order $\approx 2^{190}$. For the same group order ECC and genus-3 HECC are about a factor of 2 faster than genus-4 curves on the ARM. The two most surprising results are: 1) for low cost security application, namely considering an underlying group of order 2^{128}, HECC with genus 4 outperform genus-2 curves by a factor of 1.46 and has similar performance to genus-3 curves on the ARM and 2) when compared to genus-2 and genus-3, genus-4 HECC are better suited to embedded microprocessors than to general purpose processors.

Keywords: hyperelliptic curves, genus four, explicit formulae, efficient implementation, low cost security, embedded application, comparison HECC vs. ECC

1 Introduction

It is widely recognized that data security will play a central role in the design of future IT systems. One of the major tools to provide information security is public-key cryptography. Additionally, one notices that more and more IT applications are realized as embedded systems. In fact, 98% of all microprocessors sold today are embedded in household appliances, vehicles, and machines on factory floors [EGH00,BW00], whereas only 2% are used in PCs and workstations. This high percentage becomes more understandable if we bear in mind that a modern car has up to 80 microprocessors on board, to enable the functionality of the ABS, the engine or the entertainment electronic. We also should keep in mind that the number of microprocessors in most of our modern kitchens is higher than in our offices. Shortcomings of the embedded processors is that they possess CPUs with very low clock rates and relatively small amount of memory. Hence, embedded processors have a 100 – 1000 times lower computational power than conventional PCs. Embedded devices are already ubiquitous in our lives and will be even more in the very near future. In addition to many other challenges, the integration of security and privacy in the existing and new embedded applications will be a major one.

Since the invention of public-key (PK) cryptography in 1976, three different variants of PK cryptosystems of practical relevance have been introduced, namely cryptosystems based on the difficulty of integer factorization (e.g. RSA [RSA78]), solving the discrete logarithm problem in finite fields (e.g. Diffie-Hellman [DH76]), and the discrete logarithm problem (DLP) in the group of points of an elliptic curve (EC) over a finite field [Mil86,Kob87]. Hyperelliptic curve cryptosystems (HECC) are a generalization of elliptic curve cryptosystems (ECC) that were suggested in 1988 for cryptographic applications [Kob88].

Considering the implementation aspects of the three public-key variants, one notices that a major difference is the bit-length of the operands. It is widely accepted that for commercial applications one needs 1024-bit operands for RSA or Diffie-Hellman. In the case of ECC or HECC applications, a group order of size $\approx 2^{160}$ is believed to be sufficient for moderate long-term security. In this contribution we consider genus-4 HECC over $\mathbb{F}_q$ and therefore we will need at least $4 \cdot \log_2 q \approx 2^{160}$. In particular, for these curves, we will need a field $\mathbb{F}_q$ with $|\mathbb{F}_q| \approx 2^{40}$, i.e., 40-bit long operands. However, in many low cost and embedded applications lower security margins are adequate. In practice, if a group order of 2^{128} is sufficient, the operations can be performed with an operand length of 32-bit. Thus, the underlying field operations can be implemented very efficiently if working with 32-bit microprocessors (e.g. ARM). It is important to point out that the small field sizes and the resulting short operand size of HECC compared to other cryptosystems makes HECC specially promising for the use in embedded environments. We discuss the security of such curves in Section 4.2.

Our Contributions

The work at hand presents for the first time explicit formulae for genus-4 curves. Genus-4 HECC did not draw a lot of attention in the past because they seem to be far less efficient than genus-2 HECC, genus-3 HECC, and ECC. Our contribution is a major step in accelerating this kind of cryptosystem and contrary to common belief we were able to develop explicit formulae that perform the scalar multiplication 72% and 60% faster than previous work by Cantor [Can87] and Nagao [Nag00], respectively.

Genus-4 HECC are well suited for the implementation of public-key cryptosystems in constrained environments because the underlying arithmetic is performed with relatively small operand bit-lengths. In this contribution, we present our implementation of this cryptosystem on an ARM and a Pentium microprocessor. We were able to perform a 160bit scalar multiplication in 172 msec on the ARM@80MHz and in 6.9 msec on the Pentium4@1.8GHz. In addition, our implementations show, that genus-4 HECC are only a factor of 1.66 and 2.08 slower than genus-2 and genus-3 curves considering group order of $\approx 2^{190}$, respectively. Compared to ECC, the genus-4 HECC are a factor of 2 slower for the same group order .

Genus-4 HEC are well suited, especially for cryptographic applications with short term security. Performing arithmetic with 32-bit operands only, genus-4 HECC allow for a security comparable to of 128-bit ECC. We implemented genus-4 HECC with underlying field arithmetic for 32-bit. In this case one is able to perform arithmetic with only one word. Contrary to the general case, the implementation of genus-4 curves in groups of order $\approx 2^{128}$ outperform genus-2 curves by a factor of about 1.5. Furthermore, our implementation shows that, HECC with genus three and four have similar performance considering the group order $\approx 2^{128}$.

The remainder of the paper is organized as follows. Section 2 summarizes contributions dealing with previous implementations and efficient formulae of genus-4 HECC. Section 3 gives a brief overview of the mathematical background related to HECC and Section 4 considers the security of the implemented HECCs. Sections 5 and 6 present our new explicit formulae for genus-4 curves and methodology used for our implementation. Finally, we end this contribution with a discussion of our results and some conclusions.

2 Previous Work

We will first summarize previous improvements on genus-4 HEC group operations and second introduce implementations published in earlier contributions.

Improvements to HECC Group Operations of Genus-4 HECC

Cantor [Can87] presented algorithms to perform the group operations on HEC in 1987. In recent years, there has been extensive research being performed to speed up the group operations on genus two HECC [Nag00,Har00,MCT01,MDM+02,Tak02,Lan02a,Lan02b,Lan02c] and genus three [Nag00,KGM+02,PWGP03]. Only Nagao [Nag00] tried to improve Cantor's algorithm for higher genera. He mainly applied the following ideas:

- Division of polynomials without field inversion
- Computation of the gcd with one inversion.
- Interleaving superfluous calculations in the reduction part
- Expressing points on the Jacobian in a different form

Nagao evaluated the computational cost of the group operations by applying the stated improvements for genus $2 \leq g \leq 10$. The most efficient group addition for genus-4 curves needs 2I + 289M/S or 3I + 286M/S (depending on the cost of the field inversion compared to multiplications, one or the other is more efficient). I refers to field inversion, M to field multiplication, S to field squaring, and M/S to field multiplications or squarings, since squarings are assumed to be of the same complexity as multiplications in these publications. For the computation of a group doubling in genus-4 curves one has to perform 2I + 268M/S or 3I + 260M/S. Notice that the ideas proposed by [Nag00] are used to improve polynomial arithmetic.

Genus-4 HECC Implementations

Since HECC were proposed, there have been several software implementations on general purpose machines [Kri97,SS98,Sma99,SS00,MCT01,MDM+02,KGM+02,Lan02a] and publications dealing with hardware implementations of HECC [Wol01,BCLW02]. Only very recently work dealing with the implementation of HECC on embedded systems was published in [Pel02,PWGP03].

The results of previous genus-4 HECC software implementations are summarized in Table 1. The table entries are sorted in chronological order. All implementations use Cantor's algorithm with polynomial arithmetic. We remark that the contribution at hand is the first genus-4 HECC implementation based on explicit formulae.

3 Mathematical Background

The mathematical background described in this section is limited to the material that is required in our contribution. The interested reader is referred to [Kob89,Kob98] for more details.

Table 1. Execution times of recent HEC implementations in software

reference	processor	genus	field	$t_{scalarmult.}$ in ms
[Kri97]	Pentium@100MHz	4	$\mathbb{F}_{2^{31}}$	1100
[SS98]	Alpha@467MHz	4	$\mathbb{F}_{2^{41}}$	96.6
	Pentium-II@300MHz	4	$\mathbb{F}_{2^{41}}$	10900
[SS00]	Alpha21164A@600MHz	4	$\mathbb{F}_{2^{41}}$	43

3.1 HECC and the Jacobian

Let $\mathbb{F}$ be a finite field, and let $\overline{\mathbb{F}}$ be the algebraic closure of $\mathbb{F}$. A hyperelliptic curve C of genus $g \geq 1$ over $\mathbb{F}$ is the set of solutions $(u, v) \in \mathbb{F} \times \mathbb{F}$ to the equation

$$C : v^2 + h(u)v = f(u)$$

The polynomial $h(u) \in \mathbb{F}[u]$ is of degree at most g and $f(u) \in \mathbb{F}[u]$ is a monic polynomial of degree $2g + 1$. For odd characteristic it suffices to let $h(u) = 0$ and to have $f(u)$ square free.

A divisor $D = \sum m_i P_i$, $m_i \in \mathbb{Z}$, is a finite formal sum of $\overline{\mathbb{F}}$-points. The set of divisors of degree zero will be denoted by $\mathbb{D}^0$. Every rational function on the curve gives rise to a divisor of degree zero and is called principal. The the set of all principal divisors is denoted by $\mathbb{P}$. We can define the Jacobian of C over $\mathbb{F}$, denoted by $\mathbb{J}_C(\mathbb{F})$ as the quotient group $\mathbb{D}^0/\mathbb{P}$.

In [Can87] it is shown that the divisors of the Jacobian can be represented as a pair of polynomials $a(u)$ and $b(u)$ with $\deg b(u) < \deg a(u) \leq g$, with $a(u)$ dividing $v^2 + h(u)v - f(u)$ and where the coefficients of $a(u)$ and $b(u)$ are elements of $\mathbb{F}$ [Mum84]. In the remainder of this paper, a divisor D represented by polynomials will be denoted by $div(a, b)$.

3.2 Group Operations in the Jacobian

This section gives a brief description of the algorithms used for adding and doubling divisors on $\mathbb{J}_C(\mathbb{F})$. Algorithm 1 describes the group addition.

Algorithm 1 Group addition

Require: $D_1 = \mathrm{div}(a_1, b_1)$, $D_2 = \mathrm{div}(a_2, b_2)$
Ensure: $D = \mathrm{div}(a_3, b_3) = D_1 + D_2$
1: $d = \gcd(a_1, a_2, b_1 + b_2 + h) = s_1a_1 + s_2a_2 + s_3(b_1 + b_2 + h)$
2: $a'_0 = a_1a_2/d^2$
3: $b'_0 = [s_1a_1b_2 + s_2a_2b_1 + s_3(b_1b_2 + f)]d^{-1}(\mathrm{mod} a'_0)$
4: **while** $\deg a'_k > g$ **do**
5: $a'_k = \frac{f - b'_{k-1}h - (b'_{k-1})^2}{a'_{k-1}}$
6: $b'_k = (-h - b'_{k-1}) \bmod a'_k$
7: **end while**
8: Output $(a_3 = a'_k, b_3 = b'_k)$

Doubling a divisor is easier than general addition and therefore, Steps 1,2, and 3 of Algorithm 1 can be simplified as follows:

1: $d = \gcd(a, 2b + h) = s_1a + s_3(2b + h)$
2: $a'_0 = a^2/d^2$
3: $b'_0 = [s_1ab + s_3(b^2 + f)]d^{-1}(\mathrm{mod} a'_0)$

3.3 Harley's Algorithm

The algorithms given in the previous section for the group operations in the Jacobian of HEC require the use of polynomial arithmetic for polynomials with coefficients in the definition field. An alternative approach for genus-2 curves was proposed by Harley in [GH00]. Harley computes the necessary coefficients from the steps of Cantor's algorithm directly in the definition field without the use of polynomial arithmetic, resulting in a faster execution time.

In [GH00], the authors found out that it is essential to know the weight of the input divisor to determine explicit formulae. For each case, implementations of different explicit formulae are required. However, for practical purposes it is sufficient to only consider the most frequent cases[1] which occur with probability of $1-O(1/q)$, where q is the group order. General formulae for the most frequent case for genus-2 curves and arbitrary characteristic were presented in [Lan02a].

Algorithm 2 and 3 describe the most frequent case of group addition and doubling for genus-4 curves, respectively.

Algorithm 2 Most Frequent Case for Group Addition (g=4)

Require: $D_1 = \mathrm{div}(a_1, b_1)$, $D_2 = \mathrm{div}(a_2, b_2)$
Ensure: $D_3 = \mathrm{div}(a_3, b_3) = D_1 + D_2$
1: $k = \frac{f - b_1 h - b_1^2}{a_1}$ (exact division)
2: $s \equiv \frac{b_2 - b_1}{a_1} \bmod a_2$
3: $z = s a_1$
4: $a = \frac{k - s(z + h + 2b_1)}{a_2}$ (exact division)
5: $a' = a$ made monic
6: $b' \equiv -(h + z + b_1) \bmod a'$
7: $a_3 = \frac{f - b'h - (b')^2}{a'}$ (exact division)
8: $b_3 \equiv -(b' + h) \bmod a_3$

Algorithm 3 Most Frequent Case for Group Doubling (g=4)

Require: $D_1 = \mathrm{div}(a_1, b_1)$
Ensure: $D_2 = \mathrm{div}(a_2, b_2) = 2D_1$
1: $k = \frac{b^2 - bh - f}{a}$ (exact division)
2: $s \equiv \frac{k}{h + 2b} \bmod a$
3: $a' = s^2 + \frac{k - s(h + 2b)}{a}$ (exact division)
4: $a'' = a'$ made monic
5: $b'' \equiv -(h + sa + b) \bmod a'$
6: $a_2 = \frac{f - b'h - (b')^2}{a'}$ (exact division)
7: $b_2 \equiv -(b' + h) \bmod a_3$

In Section 5 we develop for the first time explicit formulae of Cantor's Algorithm for genus-4 curves.

[1] For addition the inputs are two co-prime polynomials and for doubling the input is a square free polynomial.

4 Security of the Implemented HECC

4.1 Security of HECC with High Genera

The DLP on $\mathbb{J}(\mathbb{F})$ can be stated as follows: given two divisors $D_1, D_2 \in \mathbb{J}(\mathbb{F})$, determine the smallest integer m such that $D_2 = mD_1$, if such an m exists.

The Pollard rho method and its variants [Pol78,Wie86,GLV00] solve the DLP with complexity $O(\sqrt{n})$ in generic groups of order n. In [FR94,Rüc99] attacks against special cases of HECC were discovered with complexity smaller than $O(\sqrt{n})$. An algorithm to compute the DL in subexponential time for sufficiently large genera and its variants were published in [ADH94,FS97,Eng99,Gau00,EG02]. However, the complexity of this algorithm is only better than the Pollard's rho method for $g > 4$. In [Gau00] it is shown that index-calculus algorithms in the Jacobian of HEC have a lower complexity than the Pollard rho method for curves of genus greater than 4. Thus, to our knowledge the best attacks against HEC of the form suggested in this contribution have a complexity of $O(\sqrt{n})$.

4.2 Security of 128-bit HECC

In [LV00], Lenstra and Verheul argue, that for commercial security in the year 2003, 136-bit ECC should be considered. Furthermore, the authors state that ECC using 136-bit keys are as secure as 1068-bit keys for RSA or DSS. This notion of commercial security is based on the hypothesis that a 56-bit block cipher offered adequate security in 1982.

Recently, Certicom announced that Chris Monico and his team of mathematicians from Notre Dame solved the Certicom ECCp-109 Challenge [Cor02]. The underlying curve is a randomly generated curves over prime fields of order p. The challenge was solved by utilizing 10,000 computers running 549 days. Note that in our HECC implementations we used group orders between 2^{128} and 2^{252}. Therefore, HECC with a group order of 2^{128} is by a factor of $\sqrt{2^{19}} \approx 724$ harder to break than the ECCp-109 challenge which would mean 1089 years, assuming the computational resources of the ECCp-109 attack.

It is also worth to point out that the factorization of the 512-bit RSA challenge took only about 2% of the time required to break the ECC2K-108 challenge (or to break DES). This implies that ECC or HECC in groups of order 2^{128} offer far more security than a 512-bit RSA system. Nevertheless, RSA with a 512-bit key is still in use, for example in fielded smart card applications.

5 First Explicit Formulae for Genus-4 HECC

The idea of describing the group operations of HECC as explicit formulae has its origin in the idea of distinguishing between possible cases according to the properties of the input divisors [GH00]. However, one can only consider the most frequent case which occurs with probability $1 - O(1/q)$, where q is the group order. For the derivation of the explicit formulae, all polynomial calculations have to be mapped onto field operations. For genus-2 HEC such a mapping can be calculated quite easily since the degree of the occurring polynomials is low. Determining explicit formulae for genus-3 HEC is more complicated since — with increasing genus — the degree of the polynomials rises. For higher genera, the problem becomes intractable by hand and the use of additional tools for the mathematical evaluation of all equations is inevitable.

The work at hand is the first approach using explicit formulae to optimize the group operations for genus-4 HEC. Table 7 presents the explicit formulae for a group addition and Table 8 those for a group doubling. The complexity of the formulae shown in the tables is based on the assumption that the coefficients h_i of $h(x) = h_4x^4 + h_3x^3 + h_2x^2 + h_1x + h_0$ are defined over $\mathbb{F}_2$. In addition, we can set f_8 to zero by substituting $x' = x + \frac{f_8}{9}$. Thus, all multiplications with the coefficients h_i for $i \in \{0, 1, 2, 3, 4\}$ and f_8 are neglected in the total operation count. A comparison of the computational complexity of our approach with the results of previously done work on genus-4 curves is illustrated in Table 2.

Our improvements of the original algorithm proposed by Cantor are mainly based on following techniques:

1. Chinese remainder theorem
2. Montgomery's trick of simultaneous inversions [Coh93, Algorithm 10.3.4]
3. Reordering of normalization step [Tak02]
4. Karatsuba multiplication [KO63]
5. Using Karatsuba to reduce complexity of polynomial reduction
6. Smart choice of HEC

An extensive description on how to apply the given techniques to reduce the complexity of the group operation can be found in [KGM+02,Lan02a,Pel02]. In the following we only stress the smart choice of HEC since the proposal of their use is new in the context of HECC of genus 4.

A detailed analysis of the explicit formulae give rise to certain types of curves with good properties, i.e. optimum performance regarding the number of required field operations for the execution of the group operations. As a result of this analysis, curves of the form $y^2 + y = f(x)$ over extension fields of characteristic two turn out to be ideal. Unfortunately this kind of curve is supersingular and therefore not suited for use in cryptography [Gau00,Gal01,SZ02]. Thus, we propose to use HEC of the form $y^2 + xy = f(x)$ over $\mathbb{F}_{2^n}$, which seem to be the best choice without any security limitations. With these curves, we can save 12 multiplications in the case of the group addition and 118 multiplications the for group doubling.

The following comparison is based on the assumption that a scalar multiplication with an n-bit scalar is realized by the sliding window method. Hence, the approximated cost of the scalar multiplication is $n \cdot$ doublings $+ 0.2 \cdot n \cdot$ additions for a 4-bit window size [BSS99]. For arbitrary curves over fields of general characteristic, we achieve a 24% improvement[2] compared to the results presented in [Nag00]. In the case of HEC over $\mathbb{F}_{2^n}$ with $h(x) = x$, we can reach an improvement up to 60% compared[2] to the best known formulae for genus-4 curves. When comparing our result to the original formulae presented by Cantor, the speed up is 72% and 47% for curves with $h(x) = x$ and general curves[2], respectively.

6 HECC Implementation

We implemented the derived explicit formulae for genus-4 HECC from Section 5 on a Pentium4 and on an ARM microprocessor. This contribution is the first to implement explicit formulae

[2] We assumed, that the computation time for one inversion is approximately the same as for 8 multiplications. This value is fortified by several implementations targeting HECC on 32-bit processors, see Appendix A.

Table 2. Complexity comparison of the group operations on HEC of genus four

	field characteristic	curve properties	cost addition	cost doubling
Cantor [Nag00]	general		$6I + 386M/S$	$6I + 359M/S$
Nagao [Nag00]	odd	$h(x) = 0,\ f_i \in \mathbb{F}_2$	$2I + 289M/S$	$2I + 268M/S$
This work (Tables 7, 8)	general	$h_i \in \mathbb{F}_2,\ f_8 = 0$	$2I + 160M + 4S$	$2I + 193M + 16S$
	two	$h(x) = x,\ f_8 = 0$	$2I + 148M + 6S$	$2I + 75M + 14S$

for genus-4 HECC and is also the first to run this cryptosystem on an embedded processor as the ARM7TDMI.

The task of implementing the genus-4 HEC group operation on an embedded device is quite challenging. Contrary to the case of curves with lower genera, the high multiplication and addition count requires careful handling of all available registers on the microprocessor. Testing and debugging such an implementation on an embedded processor becomes more complicated than on a Pentium machine.

6.1 Methodology

This subsection provides a short overview of the tools and the methodology used which finally lead to a successful implementation of the group operations.
The methodology is as follows:

1. Test the explicit formulae: NTL-based [Sho01] implementation of Cantor's algorithm and of the new explicit formulae. The *Microsoft Developer Studio 6* was used for compilation and debugging on the Pentium4.
2. Speeding up the implementation: We developed our own library for the required field and group operations. The routines are optimized for the use on embedded 32-bit processors.
3. Testing the code on the Pentium.
4. Portation to the ARM: The code was loaded into the ARM7TDMI@80MHz (ARMulator). Special functions to load the test vectors into the memory of the microprocessor and to time the routines were written. As programming and debugging platform, the *ARM Developer Suite 1.2* was used.
5. Running and testing genus-4 HECC on the ARM7TDMI (ARMulator).
6. Detailed timing analysis for different field sizes and curves.

6.2 The ARM Processor

As primary target platform, we choose the popular embedded 32-bit ARM microprocessor which can be found in mobile communication devices and consumer electronics. ARM stands for *Advanced RISC Machine* and shows a typical RISC architecture with additional features like flag-depending instruction execution. ARM 7 is based on a von Neuman architecture. The breakthrough of ARM came along with the architecture version 4T and the *Thumb Instruction Set*. The Thumb instruction set is restricted to 16-bit which allows compact code running with low power consumption. Hence, it is well suited for small hand held devices such as PDAs. As a reference and for testing we implemented all cryptosystems as well on a Pentium4.

7 Results

In this section we present our implementation results for the ARM and the Pentium. In the first part we discuss our results for standard commercial security application, whereas the second part concentrates on low cost security with genus-4 curves.

7.1 Pentium and ARM Timings

Tables 3 and 4, present the execution times for genus-4 HECC operations on the ARM processor and the Pentium, respectively. We provide timings for the group addition, group doubling and the scalar multiplication for different group orders. In addition we present the timings for genus-3 HECC, genus-2 HECC, and ECC using the same underlying library running on the same processor for comparison [PWGP03]. The formulae used for HECC implementation of genera two [Lan02a] and three [PWGP03] are to our knowledge the most efficient. In the case of ECC projektive coordinates were used.

Table 3. Timings of group operations with ARMulator ARM7TDMI@80MHz (explicit formulae)

Genus	Field	Group order	Group addition in μs	Group doubling in μs	Scalar. mult. in ms
4	$\mathbb{F}_{2^{40}}$	2^{160}	1315	740	172.43
	$\mathbb{F}_{2^{41}}$	2^{164}	1304	734	174.80
	$\mathbb{F}_{2^{44}}$	2^{176}	1319	747	190.07
	$\mathbb{F}_{2^{46}}$	2^{184}	1323	752	199.49
	$\mathbb{F}_{2^{47}}$	2^{188}	1310	745	201.89
	$\mathbb{F}_{2^{63}}$	2^{252}	1372	797	286.51
3	$\mathbb{F}_{2^{63}}$	2^{189}	615	219	72.09
2	$\mathbb{F}_{2^{95}}$	2^{190}	511	504	121.49
1	$\mathbb{F}_{2^{191}}$	2^{191}	598	358	100

Table 4. Timings of group operations on the Pentium4@1.8GHz (explicit formulae)

Genus	Field	Group order	Group addition in μs	Group doubling in μs	Scalar. mult. in ms
4	$\mathbb{F}_{2^{40}}$	2^{160}	51.0	29.3	6.88
	$\mathbb{F}_{2^{41}}$	2^{164}	49.7	27.6	6.96
	$\mathbb{F}_{2^{44}}$	2^{176}	49.9	27.9	7.50
	$\mathbb{F}_{2^{46}}$	2^{184}	50.1	28.2	7.92
	$\mathbb{F}_{2^{47}}$	2^{188}	50.3	29.3	8.05
	$\mathbb{F}_{2^{63}}$	2^{252}	51.6	29.8	8.43
3	$\mathbb{F}_{2^{63}}$	2^{189}	23.4	8.6	2.91
2	$\mathbb{F}_{2^{95}}$	2^{190}	19.1	18.8	4.68
1	$\mathbb{F}_{2^{191}}$	2^{191}	15.4	8.7	2.78

Contrary to common believe, the results show that genus-4 curves are worth to implement on 32-bit processors, though their performance for group orders larger than 2^{160} is slightly worse than that of HECC with genus $g \leq 3$. Considering a group order of approximately 2^{190}, genus-2 and genus-3 curves are a factor of 1.66 and a factor of 2.08, respectively, better than the introduced formulae for genus-4 HECC on the ARM7TDMI. Similarly, we obtain speed-ups of a factor of 1.72 and factor of 2.77 when using genus-2 and genus-3, respectively, compared to genus-4 curves on the Pentium4 for the same group order. Compared to ECC, genus-4 HECC performs a factor of 2.90 and a factor of 2.02 worse on the Pentium and ARM7TDMI, respectively.

Notice that the relative performance of genus-4 HECC is always better on the ARM microprocessor compared to the Pentium. Hence, we conclude that genus-4 HECC are well suited for the encryption in constraint environments and we encourage the research community to put more effort into the further development of this system.

7.2 Low Cost Security

In contrast to the facts mentioned in Section 7.1, there is a clear benefit of using genus-4 HECC over a 32-bit finite field. For hyperelliptic curve cryptosystems with a group order of 2^{128}, genus-4 group operations only require 32-bit field arithmetic. Thus, the processor word is optimal utilized and the cryptosystem does not need additional multi-precision arithmetic.

Analyzing the results for a group order of around 2^{128} as presented in Table 5, a major advantage of genus-4 HEC is noticeable. The comparison of the timings yield to following facts (numbers in parenthesis are for the Pentium timings):

- Genus-4 HECC outperforms genus-2 HECC by a factor of 1.46 (1.24)
- HECC of genus 3 are slightly faster by a factor of 1.04 (1.08) than genus-4 HECC

Table 5. Timings on the ARM7TDMI@80MHz and Pentium4@1.8GHz for group order $\approx 2^{128}$ (explicit formulae)

Genus	Field	Group order	Group addition in μs	Group doubling in μs	Scalar. mult. in ms
ARMulator ARM7TDMI@80MHz					
4	$\mathbb{F}_{2^{32}}$	2^{128}	441	260	49.07
3	$\mathbb{F}_{2^{43}}$	2^{129}	603	199	47.13
2	$\mathbb{F}_{2^{63}}$	2^{126}	450	443	71.54
Pentium4@1,8GHz					
4	$\mathbb{F}_{2^{32}}$	2^{128}	17.3	11.6	2.14
3	$\mathbb{F}_{2^{43}}$	2^{129}	23.6	8.2	1.98
2	$\mathbb{F}_{2^{63}}$	2^{126}	16.8	15.9	2.66

As a result of this comparison, we suggest the use of genus-4 HECC for short term security applications. Despite the high number of required field operations compared to HEC with genus $g \leq 3$, group operations on genus-4 HEC are easier to implement. This fact relies on the relatively simple field arithmetic based on operands of length no longer than 32 bits.

8 Conclusions

The work at hand presents major improvements to hyperelliptic curve cryptosystems of genus 4. We were able to reduce the average complexity of a scalar multiplication by up to 60%

compared to the currently best known formulae. The steps of the explicit formulae used to compute the group operations are presented for the first time for the case of genus-4 curves.

Additionally, we showed with the first implementation of genus-4 HECC on an embedded microprocessor that this cryptosystem is better suited for the use in embedded environments, than to general purpose processors. Contrary to common believe, our timing results show the practical relevance of this system.

Especially for applications based on asymmetric algorithms with group orders around 2^{128}, genus-4 HECC can be consider a viable choice. Not only the underlying field operations consist of simple 32-bit operations, but also we get better performance than genus-2 curves and similar speed than genus-3 curves.

References

[ADH94] L.M. Adlemann, J. DeMarrais, and M.-D. Huang. A subexponential algorithm for discrete logarithms over the rational subgroup of the jacobians of large genus hyperelliptic curves over finite fields. In *The 1st Algorithmic Number Theory Symposium — ANTS I*, pages 28 – 40, Berlin, May 1994. Springer-Verlag. LNCS 877.

[BCLW02] N. Boston, T. Clancy, Y. Liow, and J. Webster. Genus Two Hyperelliptic Curve Coprocessor. In *Workshop on Cryptographic Hardware and Embedded Systems — CHES 2002*, New York, 2002. Springer Verlag. LNCS 2523.

[BSS99] I.F. Blake, G. Seroussi, and N.P. Smart. *Elliptic Curves in Cryptography.* London Mathematical Society Lecture Notes Series 265. Cambridge University Press, Reading, Massachusetts, 1999.

[BW00] G. Borriello and R. Want. Embedded computation meets the world wide web. *Communications of the ACM*, 43(5):59–66, May 2000.

[Can87] D.G. Cantor. Computing in Jacobian of a Hyperelliptic Curve. In *Mathematics of Computation*, volume 48(177), pages 95 – 101, January 1987.

[Coh93] H. Cohen. *A course in computational number theory.* Graduate Texts in Math. 138. Springer-Verlag, Berlin, 1993. Third corrected printing 1996.

[Cor02] Certicom Corperation. Certicom Announces Elliptic Curve Cryptosystem (ECC) Challenge Winner, 2002. http://www.certicom.com/about/pr/02/021106_ecc_winner.html.

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22:644–654, 1976.

[EG02] A. Enge and P. Gaudry. A general framework for subexponential discrete logarithm algorithms. *Acta Arith.*, 102:83 – 103, 2002.

[EGH00] D. Estrin, R. Govindan, and J. Heidemann. Embedding the Internet. *Communications of the ACM*, 43(5):39–41, May 2000.

[Eng99] A. Enge. Computing discrete logarithms in high-genus hyperelliptic jacobians in provably subexponential time. Preprint; Available at http://www.math.waterloo.ca/Cond0_Dept/CORR/corr99.html, 1999.

[FR94] G. Frey and H.-G. Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Mathematics of Computation*, 62(206):865–874, April 1994.

[FS97] R. Flassenberg and S.Paulus. Sieving in function fields. Available at ftp://ftp.informatik.tu-darmstadt.de/pub/TI/TR/TI-97-13.rafla.ps.gz, 1997. To appear in Experimental Mathematics.

[Gal01] S.D. Galbraith. Supersingular Curves in Cryptography. In *Advances in Cryptology — ASIACRYPT 2001*, pages 495–517, 2001. LNCS 2248.

[Gau00] P. Gaudry. An algorithm for solving the discrete log problem on hyperelliptic curves. In Bart Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, pages 19–34, Berlin, Germany, 2000. Springer-Verlag. LNCS 1807.

[GH00] P. Gaudry and R. Harley. Counting Points on Hyperelliptic Curves over Finite Fields. In W. Bosma, editor, *The 4th Algorithmic Number Theory Symposium — ANTS IV*, pages 297 – 312, Berlin, 2000. Springer Verlag. LNCS 1838.

[GLV00] R. Gallant, R. Lambert, and S. Vanstone. Improving the parallelized Pollard lambda search on anomalous binary curves. *Mathematics of Computation*, 69(232):1699–1705, 2000.

[Har00] R. Harley. Fast Arithmetic on Genus Two Curves. Available at http://cristal.inria.fr/~harley/hyper/, 2000.

[KGM+02] J. Kuroki, M. Gonda, K. Matsuo, Jinhui Chao, and Shigeo Tsujii. Fast Genus Three Hyperelliptic Curve Cryptosystems. In *The 2002 Symposium on Cryptography and Information Security, Japan - SCIS 2002*, Jan.29-Feb.1 2002.

[KO63] A. Karatsuba and Y. Ofman. Multiplication of Multidigit Numbers on Automata. *Sov. Phys. Dokl. (English translation)*, 7(7):595–596, 1963.

[Kob87] N. Koblitz. Elliptic Curve Cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[Kob88] N. Koblitz. A Family of Jacobians Suitable for Discrete Log Cryptosystems. In Shafi Goldwasser, editor, *Advances in Cryptology — Crypto '88*, pages 94 – 99, Berlin, 1988. Springer-Verlag. LNCS 403.

[Kob89] N. Koblitz. Hyperelliptic Cryptosystems. In Ernest F. Brickell, editor, *Journal of Cryptology*, pages 139 – 150, 1989.

[Kob98] N. Koblitz. *Algebraic Aspects of Cryptography.* Algorithms and Computation in Mathematics. Springer-Verlag, 1998.

[Kri97] U. Krieger. signature.c. Master's thesis, Universität Essen, Fachbereich 6 (Mathematik und Informatik), February 1997. (Diplomarbeit).

[Lan02a] T. Lange. Efficient Arithmetic on Genus 2 Hyperelliptic Curves over Finite Fields via Explicit Formulae. Cryptology ePrint Archive, Report 2002/121, 2002. http://eprint.iacr.org/.

[Lan02b] T. Lange. Inversion-Free Arithmetic on Genus 2 Hyperelliptic Curves. Cryptology ePrint Archive, Report 2002/147, 2002. http://eprint.iacr.org/.

[Lan02c] T. Lange. Weighted Coordinates on Genus 2 Hyperelliptic Curves. Cryptology ePrint Archive, Report 2002/153, 2002. http://eprint.iacr.org/.

[LV00] A. Lenstra and E. Verheul. Selecting Cryptographic Key Sizes. In Hideki Imai and Yuliang Zheng, editors, *Third International Workshop on Practice and Theory in Public Key Cryptography — PKC 2000*, Berlin, 2000. Springer-Verlag. LNCS 1751.

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

[MDM+02] Y. Miyamoto, H. Doi, K. Matsuo, J. Chao, and S. Tsuji. A Fast Addition Algorithm of Genus Two Hyperelliptic Curves. In *SCIS, IEICE Japan*, pages 497 – 502, 2002. in Japanese.

[Mil86] V. Miller. Uses of Elliptic Curves in Cryptography. In H. C. Williams, editor, *Advances in Cryptology — CRYPTO '85*, pages 417–426, Berlin, Germany, 1986. Springer-Verlag. LNCS 218.

[Mum84] D. Mumford. Tata lectures on theta II. In *Prog. Math.*, volume 43. Birkhäuser, 1984.

[Nag00] K. Nagao. Improving group law algorithms for Jacobians of hyperelliptic curves. In W. Bosma, editor, *ANTS IV*, pages 439 – 448, Berlin, 2000. Springer Verlag. LNCS 1838.

[Pel02] J. Pelzl. Hyperelliptic Cryptosystems on Embedded Microprocessors. Master's thesis, Fakultät für Elektrotechnik und Informationstechnik, Ruhr-Universität Bochum, September 2002. (Diplomarbeit).

[Pol78] J. M. Pollard. Monte Carlo methods for index computation mod p. *Mathematics of Computation*, 32(143):918–924, July 1978.

[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. http://eprint.iacr.org/.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[Rüc99] H.-G. Rück. On the discrete logarithm in the divisor class group of curves. *Mathematics of Computation*, 68(226):805–806, 1999.

[Sho01] V. Shoup. NTL: A libary for doing Number Theory (version 5.0c), 2001. http://www.shoup.net/ntl/index.html.

[Sma99] N.P. Smart. On the Performance of Hyperelliptic Cryptosystems. In *Advances in Cryptology - EUROCRYPT '99*, pages 165 – 175, Berlin, 1999. Springer-Verlag. LNCS 1592.

[SS98] Y. Sakai and K. Sakurai. Design of Hyperelliptic Cryptosystems in small Characteristic and a Software Implementation over $\mathbb{F}_{2^n}$. In *Advances in Cryptology — ASIACRYPT '98*, pages 80 – 94, Berlin, 1998. Springer Verlag. LNCS 1514.

[SS00] Y. Sakai and K. Sakurai. On the Practical Performance of Hyperelliptic Curve Cryptosystems in Software Implementation. In *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, volume E83-A NO.4, pages 692 – 703, April 2000. .

[SZ02] J. Scholten and J. Zhu. Hyperelliptic curves in characteristic 2. *International Mathematics Research Notices*, 2002(17):905 – 917, 2002.

[Tak02] M. Takahashi. Improving Harley Algorithms for Jacobians of Genus 2 Hyperelliptic Curves. In *SCIS, IEICE Japan*, 2002. in Japanese.

[Wie86] D. H. Wiedemann. Solving Sparse Linear Equations Over Finite Fields. *IEEE Transactions on Information Theory*, IT-32(1):54–62, January 1986.

[Wol01] T. Wollinger. Computer Architectures for Cryptosystems Based on Hyperelliptic Curves. Master's thesis, ECE Department, Worcester Polytechnic Institute, Worcester, Massachusetts, USA, May 2001.

A Timings for Field Operation

In this section we present the timings for field operations for selected field orders.

Table 6. Timings field operations for genus-4 HECC

ARMulator ARM7TDMI@80MHz				
Field	Group order	Field inversion in μs	Field multiplication in μs	inversion / multiplication
$\mathbb{F}_{2^{32}}$	2^{128}	26.8	2.6	10.16
$\mathbb{F}_{2^{40}}$	2^{160}	49.2	7.3	6.73
$\mathbb{F}_{2^{47}}$	2^{188}	77.5	7.3	10.5
Pentium4@1.8GHz				
Field	Group order	Field inversion in ns	Field multiplication in ns	inversion / multiplication
$\mathbb{F}_{2^{32}}$	2^{128}	1650	168	9.82
$\mathbb{F}_{2^{40}}$	2^{160}	2519	413	6.04
$\mathbb{F}_{2^{47}}$	2^{188}	3752	402	9.33

B Explicit Formulae Genus-4 HECC

Table 7. Explicit formulae for adding on a HEC of genus four

Input	Weight three reduced divisors $D_1 = (a_1, b_1)$ and $D_2 = (a_2, b_2)$ where: $a_1 = x^4 + ax^3 + bx^2 + cx + d$; $b_1 = ix^3 + jx^2 + kx + l$ $a_2 = x^4 + ex^3 + fx^2 + gx + h$; $b_2 = mx^3 + nx^2 + ox + p$ $h = h_4x^4 + h_3x^3 + h_2x^2 + h_1x + h_0$, where $h_i \in \mathbb{F}_2$ $f = x^9 + f_7x^7 + f_6x^6 + f_5x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0$
Output	A weight four reduced divisor $D_3 = (a_3, b_3) = D_1 + D_2$ where: $a_3 = x^4 + \tilde{a}x^3 + \tilde{b}x^2 + \tilde{c}x + \tilde{d}$; $b_3 = \tilde{i}x^3 + \tilde{j}x^2 + \tilde{k}x + \tilde{l}$

Step	Procedure (For simplicity, only the implemented case is given.)	Cost	
1	Almost inverse $inv = r/a_1 \bmod a_2 = inv_3x^3 + inv_2x^2 + inv_1x + inv_0$: $r_{23} = e + a;\ r_{22} = f + b;\ r_{21} = g + c;\ r_{20} = h + d;\ t_{20} = 1;\ r_{33} = r_{23} \cdot a + r_{22};$ $r_{32} = r_{23} \cdot b + r_{21};\ r_{31} = r_{23} \cdot c + r_{20};\ r_{30} = r_{23} \cdot d;\ t_{31} = 1;\ t_{30} = r_{23};$ $r_{42} = r_{33} \cdot r_{22} + r_{23} \cdot r_{32};\ r_{41} = r_{33} \cdot r_{21} + r_{23} \cdot r_{31};\ r_{40} = r_{33} \cdot r_{20} + r_{23} \cdot r_{30};$ $t_{41} = r_{23};\ t_{40} = r_{33} + r_{23}^2;\ r_{52} = r_{42} \cdot r_{32} + r_{33} \cdot r_{41};\ r_{51} = r_{42} \cdot r_{31} + r_{33} \cdot r_{40};$ $r_{50} = r_{42} \cdot r_{30};\ t_{52} = r_{33} \cdot t_{41};\ t_{51} = r_{42} + r_{33} \cdot t_{40};\ t_{50} = r_{42} \cdot r_{23};$ $r_{61} = r_{52} \cdot r_{41} + r_{42} \cdot r_{51};\ r_{60} = r_{52} \cdot r_{40} + r_{42} \cdot r_{50};\ t_{62} = r_{42} \cdot t_{52};$ $t_{61} = r_{52} \cdot t_{41} + r_{42} \cdot t_{51};\ t_{60} = r_{52} \cdot t_{40} + r_{42} \cdot t_{50};\ r_{71} = r_{61} \cdot r_{51} + r_{52} \cdot r_{60};$ $r_{70} = r_{61} \cdot r_{50};\ t_{73} = r_{52} \cdot t_{62};\ t_{72} = r_{61} \cdot t_{52} + r_{52} \cdot t_{61};\ t_{71} = r_{61} \cdot t_{51} + r_{52} \cdot t_{60};$ $t_{70} = r_{61} \cdot t_{50};\ r_{80} = r_{71} \cdot r_{60} + r_{61} \cdot r_{70};\ inv_3 = r_{61} \cdot t_{73};\ inv_2 = r_{71} \cdot t_{62} + r_{61} \cdot t_{72};$ $inv_1 = r_{71} \cdot t_{61} + r_{61} \cdot t_{71};\ inv_0 = r_{71} \cdot t_{60} + r_{61} \cdot t_{70};$	$45M + 1S$	$46M$
2	$s' = rs \equiv (b_2 - b_1)inv \bmod a_2 = s'_3x^3 + s'_2x^2 + s'_1x + s'_0$ (Karatsuba): $t_a = m + i;\ t_b = n + j;\ t_c = o + k;\ t_d = p + l;\ t_e = inv_3;\ t_f = inv_2;\ t_g = inv_1;$ $t_h = inv_0;\ t_0 = t_c \cdot t_g;\ t_1 = t_b \cdot t_f;\ t_2 = t_a \cdot t_e;\ t_3 = t_b \cdot t_g;\ t_4 = t_c \cdot t_f;\ t_{10} = t_d \cdot t_h;$ $t_{11} = (t_c + t_d) \cdot (t_g + t_h) + t_0 + t_{10};\ t_{12} = (t_b + t - d) \cdot (t_f + t_h) + t_{10} + t_1 + t_0;$ $t_{13} = (t_a + t_d) \cdot (t_e + t_h) + t_{10} + t_2 + t_3 + t_4;\ t_{14} = (t_a + t_c) \cdot (t_e + t_g) + t_2 + t_0 + t_1;$ $t_{15} = (t_a + t_b) \cdot (t_e + t_f) + t_2 + t_1;\ t_{16} = t_2;\ t_{17} = t_{15} + e \cdot t_{16};\ t_{18} = e \cdot t_{17} + t_{16} \cdot f + t_{14};$ $s'_3 = e \cdot t_{18} + f \cdot t_{17} + g \cdot t_{16} + t_{13};\ s'_2 = f \cdot t_{18} + g \cdot t_{17} + h \cdot t_{16} + t_{12};$ $s'_1 = g \cdot t_{18} + h \cdot t_{17} + t_{11};\ s'_0 = h \cdot t_{18} + t_{10};$	$23M$	$23M$
3	$s = x^3 + s_2x^2 + x_1x + s_0 = s'$ made monic: $t_1 = r_{80} \cdot s'_3;\ w_6 = t_1^{-1};\ w_7 = r_{80} \cdot w_6;\ w_4 = r_{80} \cdot w_7;\ w_3 = s_3'^2 \cdot w_6;$ $w_5 = w_4^2;\ s_0 = s'_0 \cdot w_7;\ s_1 = s'_1 \cdot w_7;\ s_2 = s'_2 \cdot w_7;$	$I + 7M + 2S$	$I + 7M + 2S$
4	$z = sa_1 = x^7 + z_6x^6 + z_5x^5 + z_4x^4 + z_3x^3 + z_2x^2 + z_1x + z_0$ (Karats.): $t_0 = c \cdot s_1;\ t_1 = b \cdot s_2;\ z_0 = s_0 \cdot d;\ z_1 = (c + d) \cdot (s_1 + s_0) + t_0 + z_0;$ $z_2 = (b + d) \cdot (s_2 + s_0) + z_0 + t_1 + t_0;\ z_3 = (a + d) \cdot (1 + s_0) + z_0 + a + b \cdot s_1 + c \cdot s_2;$ $z_4 = (a + c) \cdot (1 + s_1) + a + t_0 + t_1 + s_0;\ z_5 = (a + b) \cdot (1 + s_2) + a + t_1 + s_1;\ z_6 = a + s_2;$ $z_7 = 1;$	$10M$	$10M$
5	$a' = [s(z + w_4(h + 2b_1)) - w_5((f - b_1h - b_1^2)/a_1)]/a_2$ $= x^6 + u'_5x^5 + u'_4x^4 + u'_3x^3 + u'_2x^2 + u_1x + u_0$: $t_1 = s_2 \cdot w_4;\ t_2 = s_1 \cdot w_4;\ diff_4 = s_2 \cdot z_6 + z_5 + s_1 + f;\ diff_3 = g + z_4 + s_0 + s_2 \cdot z_5 + s_1 \cdot z_6;$ $diff_2 = h + z_3 + s_2 \cdot z_4 + s_1 \cdot z_5 + s_0 \cdot z_6;\ diff_1 = s_2 \cdot z_3 + s_1 \cdot z_4 + s_0 \cdot z_5 + z_2 + w_5;$ $diff_0 = w_4 + s_2 \cdot z_2 + s_1 \cdot z_3 + s_0 \cdot z_4 + w_5 \cdot a + z_1;\ u'_5 = z_6 + s_2 + e;\ u'_4 = diff_4 + e \cdot u'_5;$ $u'_3 = diff_3 + e \cdot u'_4 + f \cdot u'_5;\ u'_2 = diff_2 + e \cdot u'_3 + f \cdot u'_4 + g \cdot u'_5;$ $u'_1 = diff_1 + e \cdot u'_2 + f \cdot u'_3 + g \cdot u'_4 + h \cdot u'_5;\ u'_0 = diff_0 + e \cdot u'_1 + f \cdot u'_2 + g \cdot u'_3 + h \cdot u'_4;$	$29M$	$34M$
6	$b' = -(w_3z + h + b_1) \bmod a' = v'_5x^5 + v'_4x^4 + v'_3x^3 + v'_2x^2 + v'_1x + v'_0$: $t_1 = u'_5 + z_6;\ v'_5 = w_3 \cdot (+u'_5 \cdot t_1 + u'_4 + z_5);\ v'_4 = w_3 \cdot (+u'_4 \cdot t_1 + u'_3 + z_4);$ $v'_3 = w_3 \cdot (+u'_3 \cdot t_1 + u'_2 + z_3) + i;\ v'_2 = w_3 \cdot (+u'_2 \cdot t_1 + u'_1 + z_2) + j;$ $v'_1 = w_3 \cdot (+u'_1 \cdot t_1 + u'_0 + z_1) + 1 + k;\ v'_0 = w_3 \cdot (+u'_0 \cdot t_1 + z_0) + l;$	$12M$	$12M$
7	$a'_3 = (f - b'h - b'^2)/a' = u_{34}x^4 + u_{33}x^3 + u_{32}x^2 + u_{31}x + u_{30}$: $diff_3 = 1;\ diff_2 = v_4'^2;\ diff_1 = f_7;\ diff_0 = f_6 + v'_5 + v_3'^2;\ u_{34} = v_5'^2;$ $u_{33} = diff_3 + u_{34} \cdot u'_5;\ u_{32} = diff_2 + u_{34} \cdot u'_4 + u_{33} \cdot u'_5;$ $u_{31} = diff_1 + u_{34} \cdot u'_3 + u_{33} \cdot u'_4 + u_{32} \cdot u'_5;$ $u_{30} = diff_0 + u_{34} \cdot u'_2 + u_{33} \cdot u'_3 + u_{32} \cdot u'_4 + u_{31} \cdot u'_5;$	$10M + 3S$	$16M + 2S$
8	$a_3 = x^4 + \tilde{a}x^3 + \tilde{b}x^2 + \tilde{c}x + \tilde{d} = a'_3$ made monic: $t_0 = u_{34}^{-1};\ \tilde{a} = u_{33} \cdot t_0;\ \tilde{b} = u_{32} \cdot t_0;\ \tilde{c} = u_{31} \cdot t_0;\ \tilde{d} = u_{30} \cdot t_0;$	$I + 4M$	$I + 4M$
9	$b_3 = -(b' + h) \bmod a_3 = \tilde{i}x^3 + \tilde{j}x^2 + \tilde{k}x + \tilde{l}$: $t_0 = v'_4 + v'_5 \cdot a_3;\ \tilde{i} = \tilde{a} \cdot t_0 + \tilde{b} \cdot v'_5 + v'_3;\ \tilde{j} = \tilde{b} \cdot t_0 + \tilde{c} \cdot v'_5 + v'_2;$ $\tilde{k} = \tilde{c} \cdot t_0 + \tilde{d} \cdot v'_5 + v'_1 + 1;\ \tilde{l} = \tilde{d} \cdot t_0 + v'_0;$	$8M$	$8M$
Total	in fields of arbitrary characteristic in fields of characteristic 2 and $h(x) = x$	 $2I + 148M + 6S$	$2I + 160M + 4S$

Table 8. Explicit formulae for doubling on HEC of genus four

Input	A weight four reduced divisor $D_1 = (a_1, b_1)$ where: $a_1 = x^4 + ax^3 + bx^2 + cx + d$; $b_1 = ix^3 + jx^2 + kx + l$ $h = h_4x^4 + h_3x^3 + h_2x^2 + h_1x + h_0$, where $h_i \in \mathbb{F}_2$; $f = x^9 + f_7x^7 + f_6x^6 + f_5x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0$;		
Output	A weight three reduced divisor $D_2 = (a_2, b_2) = [2]D_1$ where: $a_2 = x^4 + \tilde{a}x^3 + \tilde{b}x^2 + \tilde{c}x + \tilde{d}$; $b_2 = \tilde{i}x^3 + \tilde{j}x^2 + \tilde{k}x + \tilde{l}$		
Step	Procedure (For simplicity, only the implemented case is given.)	Cost	
1	Almost inverse $inv = r/(h + 2b_1) \bmod a_1$ $= inv_3x^3 + inv_2x^2 + inv_1x + inv_0$: $inv_3 = 1;\ inv_2 = a;\ inv_1 = b;\ inv_0 = c;$	–	$45M$
2	$z = ((f - hb_1 - b_1^2)/a_1) \bmod a_1 = x^4 + z_3x^3 + z_2x^2 + z_1x + z_0$: $diff_2 = i^2 + c;\ diff_0 = j^2 + i;\ z_4' = a;\ z_3' = b + a^2 + f_7;\ z_2' = diff_2 + a \cdot (z_3' + b) + f_6;$ $z_1' = d + a \cdot (z_2' + c) + b \cdot z_3' + f_5;\ z_0' = diff_0 + a \cdot (z_1' + d) + b \cdot z_2' + c \cdot z_3' + f_4;$ $z_3 = b + z_3';\ z_2 = c + z_2';\ z_1 = d + z_1';\ z_0 = z_0';$	$6M + 3S$	$16M + 2S$
3	$s' = zinv \bmod a_1 = s_3'x^3 + s_2'x^2 + s_1'x + s_0'$ (Karatsuba): $t_1 = z_1 \cdot inv_1;\ t_2 = z_2 \cdot inv_2;\ t_{10} = z_0 \cdot inv_0;\ t_{16} = z_3 \cdot inv_3;$ $t_{15} = (z_3 + z_2) \cdot (inv_3 + inv_2) + t_{16} + t_2;\ t_{14} = (z_3 + z_1) \cdot (inv_3 + inv_1) + t_{16} + t_1 + t_2;$ $t_{13} = (z_3 + z_0) \cdot (inv_3 + inv_0) + t_{10} + t_{16} + z_2 \cdot inv_1 + z_1 \cdot inv_2;$ $t_{12} = (z_2 + z_0) \cdot (inv_2 + inv_0) + t_{10} + t_2 + t_1;\ t_{11} = (z_1 + z_0) \cdot (inv_1 + inv_0) + t_1 + t_{10};$ $t_3 = t_{15} + a \cdot t_{16};\ t_4 = a \cdot t_3 + b \cdot t_{16} + t_{14};\ s_0' = d \cdot t_4 + t_{10};\ s_1' = c \cdot t_4 + d \cdot t_3 + t_{11};$ $s_2' = b \cdot t_4 + c \cdot t_3 + d \cdot t_{16} + t_{12};\ s_3' = a \cdot t_4 + b \cdot t_3 + c \cdot t_{16} + t_{13};$	$23M$	$23M$
4	$s = x^3 + s_2x^2 + x_1x + s_0 = s'$ made monic: $t_1 = d \cdot s_3';\ w_6 = t_1^{-1};\ w_7 = d \cdot w_6;\ w_4 = d \cdot w_7;\ w_3 = s_3'^2 \cdot w_6;$ $w_5 = w_4^2;\ s_0 = s_0' \cdot w_7;\ s_1 = s_1' \cdot w_7;\ s_2 = s_2' \cdot w_7;$	$I + 7M + 2S$	$I + 7M + 2S$
5	$G = sa_1 = x^7 + g_6x^6 + g_5x^5 + g_4x^4 + g_3x^3 + g_2x^2 + g_1x + g_0$ (Karat.): $t_0 = c \cdot s_1;\ t_1 = b \cdot s_2;\ g_0 = s_0 \cdot d;\ g_1 = (c + d) \cdot (s_1 + s_0) + t_0 + g_0;$ $g_2 = (b + d) \cdot (s_2 + s_0) + g_0 + t_1 + t_0;\ g_3 = (a + d) \cdot (1 + s_0) + g_0 + a + b \cdot s_1 + c \cdot s_2;$ $g_4 = (a + c) \cdot (1 + s_1) + a + t_0 + t_1 + s_0;\ g_5 = (a + b) \cdot (1 + s_2) + a + t_1 + s_1;\ g_6 = a + s_2;$	$10M$	$10M$
6	$a' = a_1^{-2}[(G + w_4b_1)^2 + w_4hG + w_5(hb_1 - f)]$ $= x^6 + u_5'x^5 + u_4'x^4 + u_3'x^3 + u_2'x^2 + u_1x + u_0$: $t_0 = a^2;\ t_1 = b^2;\ t_2 = c^2;\ t_3 = g_6^2;\ t_4 = g_5^2;\ u_5' = 0;\ u_4' = t_3 + t_0;\ u_3' = 0;$ $u_2' = t_4 + t_1 + t_0 \cdot (t_3 + t_0);\ u_1' = w_5;\ u_0' = g_4^2 + w_4 + t_2 + t_0 \cdot u_2' + t_1 \cdot (t_0 + t_3);$	$3M + 6S$	$52M + 10S$
7	$b' = -(w_3z + h + b_1) \bmod a' = v_5'x^5 + v_4'x^4 + v_3'x^3 + v_2'x^2 + v_1'x + v_0'$: $v_5' = w_3 \cdot (u_4' + g_5);\ v_4' = w_3 \cdot (u_4' \cdot g_6 + g_4);\ v_3' = w_3 \cdot (u_2' + g_3) + i;$ $v_2' = w_3 \cdot (u_2' \cdot g_6 + u_1' + g_2) + j;\ v_1' = w_3 \cdot (u_1' \cdot g_6 + u_0' + g_1) + 1 + k;$ $v_0' = w_3 \cdot (u_0' \cdot g_6 + g_0) + l;$	$10M$	$12M$
8	$a_2' = (f - b'h - b'^2)/a' = u_{24}x^4 + u_{23}x^3 + u_{22}x^2 + u_{21}x + u_{20}$: $diff_3 = 1;\ diff_2 = v_4'^2;\ diff_1 = f_7;\ diff_0 = f_6 + v_5' + v_3'^2;\ u_{24} = v_5'^2;$ $u_{23} = diff_3;\ u_{22} = diff_2 + u_{24} \cdot u_4';\ u_{21} = diff_1 + u_{23} \cdot u_4';$ $u_{20} = diff_0 + u_{24} \cdot u_2' + u_{22} \cdot u_4';$	$4M + 3SQ$	$16M + 2SQ$
9	$a_2 = x^4 + \tilde{a}x^3 + \tilde{b}x^2 + \tilde{c}x + \tilde{d} = a_2'$ made monic: $t_0 = u_{24}^{-1};\ \tilde{a} = u_{23} \cdot t_0;\ \tilde{b} = u_{22} \cdot t_0;\ \tilde{c} = u_{21} \cdot t_0;\ \tilde{d} = u_{20} \cdot t_0;$	$I + 4M$	$I + 4M$
10	$b_2 = -(b' + h) \bmod a_2 = \tilde{i}x^3 + \tilde{j}x^2 + \tilde{k}x + \tilde{l}$: $t_0 = v_4' + v_5' \cdot a_2;\ \tilde{i} = \tilde{a} \cdot t_0 + \tilde{b} \cdot v_5' + v_3';\ \tilde{j} = \tilde{b} \cdot t_0 + \tilde{c} \cdot v_5' + v_2';$ $\tilde{k} = \tilde{c} \cdot t_0 + \tilde{d} \cdot v_5' + v_1' + 1;\ \tilde{l} = \tilde{d} \cdot t_0 + v_0';$	$8M$	$8M$
Total	in fields of arbitrary characteristic in fields of characteristic 2 with $h(x) = x$	 $2I + 75M + 14S$	$2I + 193M + 16S$