Hyperelliptic Curve Cryptosystems: Closing the Performance Gap to Elliptic Curves

Jan Pelzl, Thomas Wollinger, Jorge Guajardo, and Christof Paar

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

Abstract. For most of the time since they were proposed, it was widely believed that hyperelliptic curve cryptosystems (HECC) carry a substantial performance penalty compared to elliptic curve cryptosystems (ECC) and are, thus, not too attractive for practical applications. Only quite recently improvements have been made, mainly restricted to curves of genus 2. The work at hand advances the state-of-the-art considerably in several aspects. First, we generalize and improve the closed formulae for the group operation of genus 3 for HEC defined over fields of characteristic two. For certain curves we achieve over 50% complexity improvement compared to the best previously published results. Second, we introduce a new complexity metric for ECC and HECC defined over characteristic two fields which allow performance comparisons of practical relevance. It can be shown that the HECC performance is in the range of the performance of an ECC; for specific parameters HECC can even possess a lower complexity than an ECC at the same security level. Third, we describe the first implementation of a HEC cryptosystem on an embedded (ARM7) processor. Since HEC are particularly attractive for constrained environments, such a case study should be of relevance.

Keywords: Hyperelliptic curves, explicit formulae, comparison HECC vs. ECC, efficient implementation

1 Introduction

In 1976 Diffie and Hellman [DH76] revolutionized the field of cryptography by introducing the concept of public-key cryptography. Their key exchange protocol is based on the difficulty of solving the discrete logarithm (DL) problem over a finite field. Years later, [Mil86,Kob87] introduced a variant of the Diffie-Hellman key exchange, based on the difficulty of the DL problem in the group of points of an elliptic curve (EC) over a finite field. Since their introduction, elliptic curve cryptosystems (ECC) have been extensively studied not only by the research community but also in industry. In particular, there are several standards involving EC, such as the IEEE P1363 [P1399] standardization effort and the bank industry standards [ANS99]. It is important to point out that ECC

C.D. Walter et al. (Eds.): CHES 2003, LNCS 2779, pp. 351–365, 2003.

benefit from shorter operand sizes when compared to RSA or DL based systems. This fact makes ECC particularly well suited for small processors and memory constrained environments.

In 1988 Koblitz suggested for the first time the generalization of EC to curves of higher genus, namely hyperelliptic curves (HEC) [Kob88]. In contrast to the EC case, it has only been until recently that Koblitz's idea to use HEC for cryptographic applications, has been analyzed and implemented both in software [Kri97,SS98,SSI98,Eng99,SS00] and in more hardware-oriented platforms such as FPGAs [Wol01,BCLW02]. In 1999, [Sma99] concluded that there seems to be little practical benefit in using HEC, because of the difficulty of finding hyperelliptic curves and their relatively poor performance when compared to EC. However, quite recently the efficiency of the HEC group operation has been improved [Har00,MDM$^+$02,Tak02,Lan02a]. It is well known that the best algorithm to compute the discrete logarithm in generic groups such as the Jacobian of a HEC is Pollard's rho method or one of its parallel variants [Pol78,vOW99]. For curves of genus higher than four, [Gau00] showed that there exists an algorithm with complexity $O(q^2)$ where F_q is the field over which the HEC is defined. Thus, in this work, we only consider HEC of genus less than four, as curves of higher genus are potentially insecure from a cryptographic point of view.

It is widely accepted that for most cryptographic applications based on EC or HEC one needs a group order of size at least $\approx 2^{160}$. Thus, for HECC over $\mathbb{F}_q$ we will need at least $g \cdot \log_2 q \approx 2^{160}$, where g is the genus of the curve. In particular, for a curve of genus two, we will need a field $\mathbb{F}_q$ with $|\mathbb{F}_q| \approx 2^{80}$, i.e., 80-bit long operands. Similarly, for curves of genus three, our discussion above implies 54-bit long operands. These field sizes make HEC specially promising for use in embedded environments where memory and speed are constrained, and where the above operand sizes seem well suited to their *small* processor architectures.

Our Main Contributions

Genus 3 group operations: The work at hand presents for the first time generalized explicit formulae for genus-3 curves including fields of characteristic 2. For certain curves our group doubling formula saves more than 66% of the field multiplications compared to [KGM$^+$02].

New complexity metric for HECC and ECC: We introduce a new metric for HECC and ECC over characteristic two fields which is based on an atomic operation count rather than on the (theoretical) bit complexity or specific timings. The most interesting results: (a) under certain conditions genus-3 hyperelliptic curves are faster than ECC at the same level of security and (b) these HEC are faster than genus-2 curves.

HECC implementation on an embedded platform: We support our theoretical findings with a HECC implementation on an ARM7TDMI, which is one of the most popular embedded processors. Our implementation uses the best explicit formulae for genus-2 and genus-3 curves.

The remainder of the paper is organized as follows. Section 2 summarizes contributions dealing with previous implementations and comparisons of HECC and ECC. Section 3 gives a brief overview of the mathematical background related to HECC. Section 4 and 5 present our new explicit formulae for genus-3 curves and a theoretical comparison between ECC and HECC. Section 6 introduces the implementation of HECC on embedded processors. Finally, we end this contribution with a discussion of our results and some conclusions.

2 Previous Work

In this section, we summarize previous improvements of the group operation of genus-2 and genus-3 curves, earlier theoretical comparisons between ECC and HECC, and other HECC implementations.

Improvements to HECC Group Operations
Table 1 summarizes the efforts made to date to speed up genus-2 curves. I refers to inversion, M to multiplication, S to squaring, and M/S to multiplications or squarings, since squarings are assumed to be of the same complexity as a multiplication in these publications. For more details on previous improvements made to the explicit formulae the interested reader is referred to [PWGP03]

For genus-3 hyperelliptic curves of odd characteristic the only improvement over Cantor's algorithm was presented in [KGM$^+$02]. The authors adopted the methods from [MDM$^+$02,Har00] to obtain the speed-up. The operation complexity for genus-3 curves is summarized in Table 3.

Theoretical Comparisons
In [SSI98], the authors clarified practical advantages of hyperelliptic cryptosystems when compared to ECC and to RSA. To our knowledge this is the first and only contribution that investigates in detail the theoretical complexity of ECC and HECC. They estimated the cost of different cryptosystems based on the number of bit operations. In their work they used Cantor's formula and the cost of one multiplication in $\mathbb{F}_{2^n}$ was assumed to take n^2 bit operations. One of the estimated theoretical results shows that genus-3 curves needed three times as many bit operations as elliptic curves. We want to point out that this publication used supersingular curves[1] and curves of genus higher than 4 which today are believed to be insecure due to the attacks presented in [FR94,Gau00,Gal01].

In the following years further analyses of the complexity of HECC were published. A theoretical analysis of the computational efficiency of the arithmetic on hyperelliptic curves is derived in [Eng99]. In [SS00], the authors implemented hyperelliptic curve cryptosystems and analyzed the complexity of the group law on Jacobians $\mathbb{J}_C(\mathbb{F}_p)$ and $\mathbb{J}_C(\mathbb{F}_{2^n})$. Moreover, they verified their theoretical complexity estimates with a HECC implementation and with the theoretical analysis

[1] [Gal01] gives some arguments against using supersingular hyperelliptic curves in cryptographic applications.

done by Enge in [Eng99]. More recent papers present timings for HECC using explicit formulae and compared HECC to ECC [Lan02a]. However, these comparisons were based on the implementation timings.

Table 1. Speeding up group operations on hyperelliptic curves of genus two.

	field charac.	curve properties	cost	
			addition	doubling
Cantor [Nag00]	general		$3I + 70M/S$	$3I + 76M/S$
Nagao [Nag00]	odd	$h(x) = 0,\ f_i \in \mathbb{F}_2$	$1I + 55M/S$	$1I + 55M/S$
Harley [Har00]	odd	$h(x) = 0$	$2I + 27M/S$	$2I + 30M/S$
Matsuo et al. [MCT01]	odd	$h(x) = 0$	$2I + 25M/S$	$2I + 27M/S$
Miyamoto et al. [MDM$^+$02]	odd	$h(x) = 0,\ f_4 = 0$	$I + 26M/S$	$I + 27M/S$
Takahashi [Tak02]	odd	$h(x) = 0$	$I + 25M/S$	$I + 29M/S$
Lange [Lan02a]	general	$h_i \in \mathbb{F}_2,\ f_4 = 0$	$I + 22M + 3S$	$I + 22M + 5S$
	two	$h_i \in \mathbb{F}_2,\ f_4 = 0$	$I + 22M + 2S$	$I + 20M + 4S$
Lange [Lan02b]	general	$h_i \in \mathbb{F}_2,\ f_4 = 0$	$47M + 4S(40M + 3S)$[2]	$40M + 6S$
	two	$h_i \in \mathbb{F}_2,\ f_4 = 0$	$46M + 2S$	$33M + 6S$
Lange [Lan02c]	odd	$h_i \in \mathbb{F}_2,\ f_4 = 0$	$47M + 7S(36M + 5S)$[2]	$34M + 7S$
	even	$h_2 \neq 0,\ h_i \in \mathbb{F}_2,\ f_4 = 0$	$46M + 4S(35M + 5S)$[2]	$35M + 6S$
	even	$h_2 = 0,\ h_i \in \mathbb{F}_2,\ f_4 = 0$	$44M + 6S(34M + 6S)$[2]	$29M + 6S$

To our knowledge there is no theoretical complexity comparison between ECC and HECC published that uses the explicit formulae for HECC and compares HECC and ECC in terms of processor instructions, such as shift and XOR operations. Hence, this comparison is processor independent and can be adapted to any platform.

HECC Implementations

Since HEC cryptosystems were proposed, there have been several software implementations on general purpose machines and, only recently, publications dealing with hardware implementations of HECC. To our knowledge there has not been any work dealing with the implementation of HEC on embedded systems. The results of previous HECC software implementations are summarized in Table 2. Detailed information about previously made HECC implementations can be found in [PWGP03].

The first HECC hardware architectures were proposed in [Wol01]. The performance of a hardware-based genus two hyperelliptic curve coprocessor over $\mathbb{F}_{2^{113}}$ was presented in [BCLW02]. The FPGA was clocked at 45 MHz and required 4750 clock cycles for a group addition and 4050 clock cycles for a group doubling operation.

[2] mixed addition

Table 2. Execution times of recent HEC implementations in software.

reference	processor	genus	field	$t_{scalarmult.}$ in ms
[Kri97]	Pentium@100MHz	2	$\mathbb{F}_{2^{64}}$	520
		3	$\mathbb{F}_{2^{42}}$	1200
		4	$\mathbb{F}_{2^{31}}$	1100
[SS98]	Alpha@467MHz	3	$\mathbb{F}_{2^{59}}$	83.3
		3	$\mathbb{F}_{2^{89}}$	25700
		3	$\mathbb{F}_{2^{113}}$	37900
		4	$\mathbb{F}_{2^{41}}$	96.6
	Pentium-II@300MHz	3	$\mathbb{F}_{2^{59}}$	11700
		4	$\mathbb{F}_{2^{41}}$	10900
[SS00]	Alpha21164A@600MHz	3	$\mathbb{F}_p(\log_2 p = 60)$	98
		3	$\mathbb{F}_{2^{59}}$	40
		4	$\mathbb{F}_{2^{41}}$	43
[MCT01]	PentiumIII@866MHz	2	186-bit OEF	1.98
[MDM+02]	PentiumIII@866MHz	2	186-bit OEF	1.69
[KGM+02]	Alpha21264@667MHz	3	$\mathbb{F}_{2^{61}-1}$	0.932
[Lan02a]	Pentium-IV@1.5GHz	2	$\mathbb{F}_{2^{160}}$	18.875
		2	$\mathbb{F}_{2^{180}}$	25.215
		2	$\mathbb{F}_p(\log_2 p = 160)$	5.663
		2	$\mathbb{F}_p(\log_2 p = 180)$	8.162

3 Mathematical Background

In this section we present an elementary introduction to some of the theory of hyperelliptic curves over finite fields of arbitrary characteristic, restricting attention to material that is relevant for this work. For more details the reader is referred to [Kob89,Kob98].

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

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

Such a curve is said to be non-singular if there are no pairs $(u, v) \in \overline{\mathbb{F}} \times \overline{\mathbb{F}}$ which simultaneously satisfy the equation of the curve C and the partial differential equations $2v + h(u) = 0$ and $h'(u)v - f'(u) = 0$. 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. Its degree is the sum of the coefficients $\sum m_i$. The set of all divisors form an Abelian group denoted by $\mathbb{D}(C)$. The set of divisors of degree zero will be denoted by $\mathbb{D}^0 \subset \mathbb{D}(C)$.

Every rational function on the curve gives rise to a divisor of degree zero, consisting of the formal sum of the poles and zeros of the function. Such divisors are called principal and the set of all principal divisors is denoted by $\mathbb{P}$. If $D_1, D_2 \in \mathbb{D}^0$ then we write $D_1 \sim D_2$ if $D_1 - D_2 \in \mathbb{P}$; D_1 and D_2 are said to be equivalent divisors. Now, we can define the Jacobian of C as the quotient group $\mathbb{D}^0/\mathbb{P}$. If we want to define the Jacobian over $\mathbb{F}$, denoted by $\mathbb{J}_C(\mathbb{F})$, we say that a divisor $D = \sum m_i P_i$ is defined over $\mathbb{F}$ (sometimes also called a $\mathbb{F}$-divisor or rational divisor) if $D^\sigma = \sum m_i P_i^\sigma$ is equal to D for all automorphisms σ of $\overline{\mathbb{F}}$ over $\mathbb{F}$. Notice that this does not mean that each P_i^σ is equal to P_i, σ may permute the points.

In [Can87], Cantor shows that each element of the Jacobian can be represented in the form $D = \sum_{i=1}^{r} P_i - r \cdot \infty$ such that for all $i \neq j$, P_i and P_j are not symmetric points. Such a divisor is called a semi-reduced divisor. Cantor concludes that from the Riemann-Roch Theorem follows that each element of the Jacobian can be represented uniquely by such a divisor, subject to the additional constraint $r \leq g$. Such divisors are referred to as reduced divisors. Finally, [Can87] shows 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] (notice that in our particular application $\mathbb{F}$ is a finite field). In the remainder of this paper, a divisor D represented by polynomials will be denoted by $div(a, b)$.

3.2 Group Operations on a Jacobian

This section gives a brief description of the algorithms used for adding and doubling divisors on $\mathbb{J}_C(\mathbb{F})$. These group operations will be performed in two steps. First we have to find a semi-reduced divisor $D' = \mathrm{div}(a', b')$, such that $D' \sim D_1 + D_2 = \mathrm{div}(a_1, b_1) + \mathrm{div}(a_2, b_2)$ in the group $\mathbb{J}_C(\mathbb{F})$. In the second step we have to reduce the semi-reduced divisor $D' = \mathrm{div}\ (a', b')$ to an equivalent divisor $D = (a, b)$. 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: $D = \mathrm{div}(a, b) = D_1 + D_2$
1: $d = \gcd(a_1, a_2, b_1 + b_2 + h) = s_1 a_1 + s_2 a_2 + s_3(b_1 + b_2 + h)$
2: $a'_0 = a_1 a_2 / d^2$
3: $b'_0 = [s_1 a_1 b_2 + s_2 a_2 b_1 + s_3(b_1 b_2 + f)] d^{-1} (\mathrm{mod} a'_0)$
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 $(a = a'_k, b = b'_k)$
```

Doubling a divisor is easier than general addition and therefore, Steps 1,2, and 3 of Algorithm 1 can be simplified. The formulae given for the group operation of HECC can be written explicitly as previously mentioned. In Section 4 we develop explicit formulae of Cantor's Algorithm for genus-3 curves. For security considerations of the HEC used see [PWGP03].

4 Speed-up for Genus-3 Curves

In this section we briefly outline the ideas of [GH00] and [KGM$^+$02] which are the starting point for our improvements. In [GH00], the authors noticed that one can reduce the number of operations required to add/double divisors by distinguishing between possible cases according to the properties of the input divisors. This technique is combined with the use of the Karatsuba multiplication algorithm [KO63] and the Chinese remainder theorem to further reduce the complexity of the overall group operations. The work of [GH00] was generalized by [KGM$^+$02] to genus-3 curves defined over odd characteristic fields. In particular, they notice that for genus-3 curves there are 6 possible choices for the degree of the input polynomials to Algorithm 1 and that further classification according to the common factors of the polynomials would lead to about 70 subcases. However, they only consider the most frequent cases[3] which occur with overwhelming probability of $1 - O(1/q) \approx 1 - 2^{-60}$ for genus-3 curves over $\mathbb{F}_{2^{60}}$. For the remaining cases, they use Cantor's algorithm.

In this work, we further optimize the formulae of [KGM$^+$02] and generalize them to arbitrary characteristic. Table 7 presents the explicit formulae for a group addition and Table 8 those for a group doubling. The formulae shown in the tables are based on the assumption that $h_i \in \{0, 1\}$, where $i = 0, 1, 2, 3$, and that f_6 is equal to zero. The latter can be achieved by substituting $x' = x + \frac{f_6}{7}$. The coefficient is still included in the algorithm for completeness.

Our improvements are based on the following techniques [PWGP03]:

1. Montgomery's trick of simultaneous inversions [Coh93, Algorithm 10.3.4]
2. Reordering of normalization step [Tak02]
3. Karatsuba multiplication
4. Calculation of the resultant using Bezout's matrix
5. Choice of HEC

Table 3. Comparing the complexity of the group operations on HEC of genus three.

	field	curve	cost	
	characteristic	properties	addition	doubling
Cantor [Nag00]	general	$h(x) = 0,\ f_i \in \mathbb{F}_2$	$4I + 200M/S$	$4I + 207M/S$
Nagao [Nag00]	odd		$2I + 154M/S$	$2I + 146M/S$
Kuroki et al. [KGM$^+$02]	odd	$h(x) = 0,\ f_6 = 0$	$I + 81M/S$	$I + 74M/S$
This work (Tables 7, 8)	general	$h_i \in \mathbb{F}_2,\ f_6 = 0$	$I + 70M + 6S$	$I + 61M + 10S$
	two	$h_i \in \mathbb{F}_2,\ f_6 = 0$	$I + 65M + 6S$	$I + 53M + 10S$
	two	$h(x) = 1,\ f_6 = 0$	$I + 65M + 6S$	$I + 14M + 11S$

[3] For addition the inputs are two co-prime polynomials of degree 3, for doubling the input is a square free polynomial of degree 3

As a summary we include the computational cost of all the published results for genus-3 curves in Table 3. Compared to [KGM$^+$02], we save 5 multiplications in the addition algorithm and 3 multiplications in the doubling algorithm even though our formulae are more general.

5 Comparing ECC and HECC

In the past, providing complexity measures and, thus, comparisons between ECC and HECC was a difficult undertaking. The operations involved in both systems were very different (different field orders, field operations vs. operations with polynomials, etc.). Furthermore, measures such as the bit complexity often provide very little information about the *de facto* complexity in actual implementations. The underlying motivation for the work described in the following was the development of a more accurate metric for practical purposes. All operations which are computationally expensive will be expressed in terms of *atomic operations* (AOPS), such as processor word-SHIFTs and XORs. In particular, we will decompose field multiplications into AOPS. This provides a metric which allows a comparison of fields of different sizes which is crucial for comparing ECC and HECC with equal level of security. The approach possesses the advantage that it accurately counts the actual elementary processor operations (as opposed to the more theoretical bit complexity), while at the same time avoiding processor and implementation-dependent "tricks" which can skew comparisons that are merely based on timings. In summary, we believe we developed a method which allows accurate predictions of the performance on a given processor without the laborious task of actually implementing the cryptosystem. The accuracy of the new metric is demonstrated by a mere 12% difference between our theoretical and practical results.

The number of atomic operations is denoted as AOPS. In our comparison we make the following assumptions:

1. We only consider fields of characteristic two and thus neglect the cost of squaring.
2. We perform field multiplications with Algorithm 5 published[4] in [LD00]. This algorithm requires $3 + 2(w/4 - 1)$ word-SHIFTs and $s(11 + n/4) + 8(2s - 1)$ word-XORs, where w is the word size of processor and $s = \lceil \frac{n}{w} \rceil$ is the number of words needed to represent an element of the underlying field $\mathbb{F}_{2^n}$.
3. We express the cost of one field inversion as m field multiplications and denote the ratio of multiplications to inversions as *MI*-ratio.

Based on the assumptions stated above, the complexity of the group operations of HEC and EC are summarized. Referring to Tables 7 and 8, a divisor addition for a genus-3 curve requires $1I + 65M$ and doubling needs $1I + 53M$ (using a curve with $h = 1$, doubling needs only $1I + 14M$). Assuming that the cost

[4] To our knowledge this is the fastest published multiplication algorithm for finite fields of characteristic two.

of one field inversion is equivalent to m field multiplications, leads to $(65+m)M$ and $(53+m)M$ for addition and doubling, respectively. Due to the higher extension of the underlying field used for genus-2 curves, a different *MI*-ratio l is used. This leads to $(22+l)M$ for a divisor addition and $(20+l)M$ for a divisor doubling. The number of inversions and multiplications for a group operation on EC heavily depends on the chosen coordinate system. For completeness we summarize the number of required operations in Table 4.

Table 4. Field operations required in each coordinate system [HHM00]

Coordinate system	EC Addition		EC Doubling
	general	mixed coord.	
Affine coordinates	$1I+2M$		$1I+2M$
Standard projective coordinates [CC87,CMO98]	$13M$	$12M$	$7M$
Jacobian projective coordinates [CC87,CMO98]	$15M$		$5M$
New projective coordinates [LD99]	$14M$	$9M$	$4M$

Figure 1 illustrates the number of operations for a scalar multiplication on a 32-bit processor depending on the MI-ratios. The scalar multiplication with an n-bit scalar is realized by the sliding window method with an approximated cost of $n \cdot$doublings $+0.2 \cdot n \cdot$additions for a 4-bit window size [BSS99]. Figure 1 allows to estimate the efficiency of an ECC or a HECC built on top of a given field library by comparing the different MI-ratios.

In general we can draw the following conclusions from this comparison:

1. ECC with projective coordinates is in almost all cases the most efficient cryptosystem.
2. Scalar multiplication of genus-3 HEC with $h(x) = 1$ always outperforms genus-2 HEC.
3. Genus-3 HECC scalar multiplication is in most cases faster than ECC using affine coordinates.
4. For field libraries with very high MI-ratio, ECC using Jacobian projective is more efficient than genus-3 HEC. However, for low MI-ratios the HECC scalar multiplication becomes less expensive.

6 HECC on Embedded Systems

With the predicted advent of ubiquitous computing, embedded processors will play an increasingly important role in providing security functions. Due to their relatively short operand lengths, HECC are particularly well suited for embedded processors which are typically computationally constrained. We chose a representative of the popular ARM processor family for our implementation. The

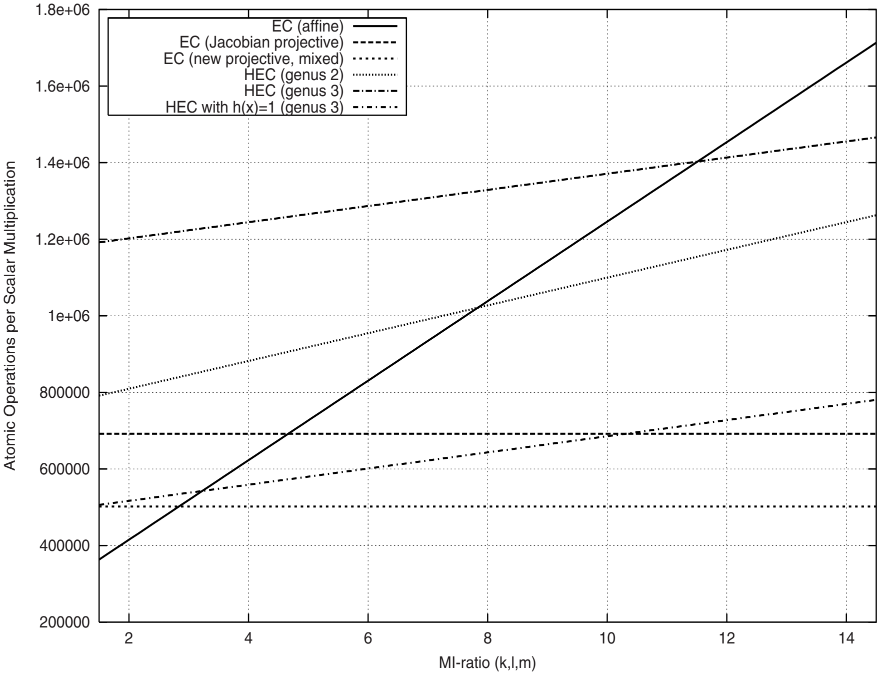

Fig. 1. Cost of a scalar multiplication for different MI-ratios and cryptosystems in AOPS (32-bit μP, group order $\approx\ 2^{190}$)

purpose was twofold. First, we wanted to provide actual timings of a highly optimized HEC implementation. Secondly, we wanted to validate our complexity metric.

The ARM7TDMI@80MHz[5] processor environment was chosen to implement both elliptic and hyperelliptic curve cryptosystems. For the elliptic curve case we used curves over $\mathbb{F}_{2^{191}}$ and Jacobian projective coordinates. The most efficient explicit formulae were implemented in the case of HECC. For genus-3 curves the polynomial $h(x)$ equals one. The group orders range from 2^{162} to 2^{190}. Table 5 presents timings for divisor addition, divisor doubling and scalar multiplication on the ARMulator. To our knowledge these are the first published timings for HECC on an embedded processor.

To theoretically determine the most efficient cryptosystem based on the timings given in Table 6, one can either use Figure 1 or calculate the necessary number of AOPS. Considering a finite field $\mathbb{F}_{2^{63}}$ for a genus-3 HEC, $619,402$ AOPS are needed to calculate one scalar multiplication. HECC of genus 2 with the underlying field $\mathbb{F}_{2^{95}}$ will take $1,049,028$ AOPS, and ECC over $\mathbb{F}_{2^{191}}$ using Jacobian projective coordinates requires $699,060$ AOPS. Thus, we expect

[5] Depending on the features of processor board, the performance numbers can differ.

HECC of genus-2 to be a factor of 1.5 slower and genus-3 HECC a factor of 1.1 faster than ECC. Genus-2 HECC is expected to be 1.5-times slower than genus-3 HECC.

Table 5. 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[6]
3	$\mathbb{F}_{2^{54}}$	2^{162}	914	317	90
	$\mathbb{F}_{2^{55}}$	2^{165}	917	319	91
	$\mathbb{F}_{2^{59}}$	2^{177}	1180	415	126
	$\mathbb{F}_{2^{60}}$	2^{180}	921	324	100
	$\mathbb{F}_{2^{61}}$	2^{183}	1183	417	130
	$\mathbb{F}_{2^{63}}$	2^{189}	925	329	106
2	$\mathbb{F}_{2^{81}}$	2^{162}	618	628	128
	$\mathbb{F}_{2^{83}}$	2^{166}	732	756	157
	$\mathbb{F}_{2^{88}}$	2^{176}	749	774	170
	$\mathbb{F}_{2^{91}}$	2^{182}	754	778	177
	$\mathbb{F}_{2^{95}}$	2^{190}	641	650	155
1	$\mathbb{F}_{2^{191}}$	2^{191}	598	358	100

The timings for a scalar multiplication of genus-3 curves over $\mathbb{F}_{2^{63}}$ and of genus-2 curves over $\mathbb{F}_{2^{95}}$ are compared with the performance of the ECC scalar multiplication over $\mathbb{F}_{2^{191}}$. HECC of genus 3 is a factor of 1.1 and HECC of genus 2 is a factor of 1.5 slower than ECC. Furthermore, a divisor scalar multiplication on a HEC of genus 2 performs a factor of 1.5 worse than a genus-3 HECC. The deviation of our implementation and the theoretical findings is at most 12%. Thus, we can conclude that our theoretical estimates were quite accurate.

Table 6. Timings of the field library and corresponding MI-ratios. All timings in μs assuming a 80MHz clock rate.

Field	Multiplication	Inversion	MI-ratio
$\mathbb{F}_{2^{63}}$	11.5	73.7	6.4
$\mathbb{F}_{2^{95}}$	19.3	157.2	8.2
$\mathbb{F}_{2^{191}}$	50.7	469.9	9.3

[6] A further speed-up can be achieved by the use of special reduction routines targeting a fixed irreducible polynomial.

7 Conclusions

In this contribution, we were able to close the gap between the performance of HECC and ECC. In particular, an improvement of the explicit formulae for arbitrary characteristic for the case of genus-3 hyperelliptic curves was presented. For certain curves over fields of characteristic 2, the efficiency of the doubling algorithm could be enhanced drastically. This increased the performance of a scalar multiplication by over 50% compared to [KGM$^+$02].

A theoretical comparison of ECC to HECC with coefficients in $\mathbb{F}_{2^m}$ assuming the currently fastest algorithms for field operations was also presented. An important finding is that HECC can reach about the same throughput than ECC and that genus-3 HECC with $h(x) = 1$ are always faster than genus-2 HECC. However, the properties of the field libraries are the key to determine the overall performance of ECC and HECC.

The theoretical results are confirmed by the first implementation of genus-2 and genus-3 curves on an embedded processor.

References

[ANS99] ANSI X9.62-1999. The Elliptic Curve Digital Signature Algorithm. Technical report, ANSI, 1999.

[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.

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

[CC87] D.V. Chudnovsky and G.V. Chudnovsky. Sequences of numbers generated by addition in formal groups and new primality and factorization tests. In *Advances in Applied Mathematics*, volume 7, pages 385–434, 1987.

[CMO98] H. Cohen, A. Miyaji, and T. Ono. Efficient elliptic curve exponentiation using mixed coordinates. In K. Ohta and D. Pei, editors, *Advances in Cryptology — ASIACRYPT 98*, pages 51–65. Springer Verlag, 1998. LNCS 1514.

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

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

[Eng99] A. Enge. The extended Euclidean algorithm on polynomials, and the computational efficiency of hyperelliptic cryptosystems, November 1999. Preprint.

[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.

[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*, volume LNCS 1807, pages 19–34, Berlin, Germany, 2000. Springer-Verlag.

[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.

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

[HHM00] D. Hankerson, J. López Hernandez, and A. Menezes. Software Implementation of Elliptic Curve Cryptography Over Binary Fields. In Çetin K. Koç and Christof Paar, editors, *Workshop on Cryptographic Hardware and Embedded Systems — CHES 2000*, pages 1–24. Springer Verlag, August 2000. LNCS 1717.

[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] Uwe Krieger. signature.c, February 1997. Diplomarbeit, Universität Essen, Fachbereich 6 (Mathematik und Informatik).

[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/.

[LD99] J. Lopéz and R. Dahab. Improved algorithms for elliptic curve arithmetic in GF(2^n). In *Selected Areas in Cryptography – SAC '98*, pages 201–212, 1999. LNCS 1556.

[LD00] J. Lopez and R. Dahab. High-speed software multiplication in $\mathbb{F}_{2^m}$. In *INDOCRYPT*, pages 203–212, 2000.

[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 Curve. 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*, volume LNCS 218, pages 417–426, Berlin, Germany, 1986. Springer-Verlag.

[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*, volume 1838 of *Lecture Notes in Computer Science*, pages 439–448, Berlin, 2000. Springer Verlag.

[P1399] *IEEE P1363 Standard Specifications for Public Key Cryptography*, November 1999. Last Preliminary Draft.

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

[PWGP03] Jan Pelzl, Thomas Wollinger, Jorge Guajardo, and Christof Paar. Hyperelliptic Curve Cryptosystems: Closing the Performance Gap to Elliptic Curves (Update). Cryptology ePrint Archive, Report 2003/026, 2003. `http://eprint.iacr.org/`.

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

[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. IEICE Trans.

[SSI98] Y. Sakai, K. Sakurai, and H. Ishizuka. Secure Hyperelliptic Cryptosystems and their Performance. In *Public Key Cryptography*, pages 164–181, Berlin, 1998. Springer-Verlag. LNCS 1431.

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

[vOW99] P. C. van Oorschot and M. J. Wiener. Parallel collision search with cryptanalytic applications. *Journal of Cryptology*, 12(1):1–28, Winter 1999.

[Wol01] T. Wollinger. Computer Architectures for Cryptosystems Based on Hyperelliptic Curves, 2001. Master Thesis, Worcester Polytechnic Institute.

A Explicit Formulae for Genus Three HEC

The explicit formulae for the group operations on HEC of genus 3 and arbitrary characteristic as well as the most efficient formulae for doubling on a HEC with $h(x) = 1$ for characteristic two is presented in Tables 7 and 8.

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

Input	Weight three reduced divisors $D_1 = (u_1, v_1)$ and $D_2 = (u_2, v_2)$ $h = x^3 + h_2x^2 + h_1x + h_0$, where $h_i \in \mathbb{F}_2$; $f = x^7 + f_5x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0$;
Output	A weight three reduced divisor $D_3 = (u_3, v_3) = D_1 + D_2$

Step	Procedure	Cost
1	Resultant r of u_1 and u_2 (Bezout)	$12M + 2S$
2	Almost inverse $inv = r/u_1 \bmod u_2$	$4M$
3	$s' = rs \equiv (v_2 - v_1)inv \bmod u_2$ (Karatsuba)	$11M$
4	$s = (s'/r)$ and make s monic	$I + 6M + 2S$
5	$z = su_1$	$6M$
6	$u' = [s(z + w_4(h + 2v_1)) - w_5((f - v_1h - v_1^2)/u_1)]/u_2$	$15M$
7	$v' = -(w_3z+h+v_1) \bmod u'$	$8M$
8	u', i.e. $u_3 = (f - v'h - v'^2)/u'$	$5M + 2S$
9	$v_3 = -(v' + h) \bmod u_3$	$3M$
Total	in fields of arbitrary characteristic in fields of characteristic 2	$I + 70M + 6S$ $I + 65M + 6S$

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

Input	A weight three reduced divisors $D_1 = (u_1, v_1)$ $h = x^3 + h_2x^2 + h_1x + h_0$, where $h_i \in \mathbb{F}_2$; $f = x^7 + f_5x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0$;
Output	A weight three reduced divisor $D_2 = (u_2, v_2) = [2]D_1$

Step	Procedure	Cost
1	Resultant r of u_1 and $h + 2v_1$ (Bezout)	$6M + 2S$
2	Almost inverse $inv = r/(h + 2v_1) \bmod u_1$	$4M$
3	$z = ((f - hv_1 - v_1^2)/u_1) \bmod u_1$	$7M + 2S$
4	$s' = zinv \bmod u_1$ (Karatsuba)	$11M$
5	$s = (s'/r)$ and make s monic	$I + 6M + 2S$
6	$G = su_1$	$6M$
7	$u' = u_1^{-2}[(G + w_4v_1)^2 + w_4hG + w_5(hv_1 - f)]$	$5M + 2S$
8	$v' = -(Gw_3 + h + v_1) \bmod u'$	$8M$
9	u', i.e. $u_2 = (f - v'h - v'^2)/u'$	$5M + 2S$
10	$v_2 = -(v' + h) \bmod u_2$	$3M$
Total	in fields of arbitrary characteristic in fields of characteristic 2	$I + 61M + 10S$ $I + 53M + 10S$
I	$d = gcd(u_1, 1) = 1 = s_1a + s_3h(s3 = 1, s1 = 0)$	–
II	$u' = u_1^2$	$3S$
III	$v' = v_1^2 + f \bmod u'$	$3S$
IV	$u'' = ((f - hv' - v'^2)/u')$	$3M + 3S$
V	$u_2 = u''$ made monic	$1I + 2M$
VI	$v_2 = -(v' + h) \bmod u_2$ (Karatsuba)	$5M$
VII	$u_3 := (f - v_2 * h - v_2^2)/u_2$	$1M + 2S$
VIII	$v_3 := -(v_2 + h) \bmod u_3$	$3M$
Total	in fields of characteristic 2 and with $h(x) = 1$	$I + 14M + 11S$