Weighted Coordinates on Genus 2 Hyperelliptic Curves

Tanja Lange

Information-Security and Cryptography,
Ruhr-University of Bochum,
Universitätsstr. 150,
44780 Bochum, Germany,
lange@itsc.ruhr-uni-bochum.de,
http://www.itsc.ruhr-uni-bochum.de/tanja

October 11, 2002

Abstract

This paper is the third in a line considering the arithmetic in the ideal class group of hyperelliptic genus two curves. The previous two papers deal with generalizations of affine and projective coordinates. Now we investigate how one can obtain inversion free formulae that are faster than projective by considering weighted coordinates. To that end we make an extensive case study to deal with different characteristic, equation of the curve, space requirement and situation of appliance.

1 Introduction

Although it is now more than a decade ago that Koblitz [4] suggested to use the ideal class group related to hyperelliptic curves for discrete logarithm based cryptography, their use in practice is just beginning to receive more attention. So far, the arithmetic in this group was usually performed using Cantor's algorithm (see Cantor [1] and Koblitz [4] for a generalization to even characteristic). However, for fixed genus one can make these steps explicit and a more clever ordering results in faster formulae for addition and doubling of classes. We now give a brief overview of the results obtained so far for genus 2 curves and sketch the situation for elliptic curves, which will lead to analogies in the case of $g = 2$.

1.1 Formulae for Genus 2 Curves

For genus 2 curves this was considered by Spallek [11] and by Krieger [5]. The first practical formulae were obtained by Harley [3], which were generalized to even characteristic by Lange [6]; an improvement of the former paper can be found in Matsuo, Chao, and Tsujii [9]. A significant improvement was obtained independently by Takahashi [13] and Miyamoto, Doi, Matsuo, Chao, and Tsuji [10]; this was generalized to even characteristic by Lange[7] (see also [12]). All these formulae involve (at least) 1 inversion per addition or doubling respectively. [10] proposes to trade this inversion for several multiplications and squarings in that they include an additional coordinate which contains the common denominator of all other coordinates. This corresponds to projective coordinates in the case of elliptic curves. This setting has been generalized and improved in [8] by Lange. A milestone on the road towards hyperelliptic curve cryptography in real life is Kim Nguyen's hardware implementation of Lange's projective formulae reported at *ECC 2002 - Workshop on elliptic curve cryptography, Essen.*

1.2 Situation for Elliptic Curves

For elliptic curves, i. e. curves of genus 1, one can choose from several systems of coordinates (see Cohen, Miyaji, and Ono [2]) such that depending on the requirement one can use the optimal system for the respective purpose. Affine points are tuples (x, y) satisfying $y^2 + (a_1x + a_3)y = x^3 + a_2x^2 + a_4x + a_6$. The homogenized equations leads to points (X, Y, Z) with the correspondence $x = X/Z, y = Y/Z$. The addition formulae for these points in projective coordinates avoid inversions. The idea to achieve formulae faster than the projective and still without using inversions is to allow *weighted projective* coordinates. In the elliptic setting one lets (X, Y, Z) correspond to $(x, y) = (X/Z^2, Y/Z^3)$. These coordinates are called Jacobian coordinates; in this system, additions are slightly more expensive while doublings get considerably cheaper than in projective coordinates. Some computations done in the process of adding or doubling can provide useful in the following operation. Thus, if the space is not too restricted, one can include them in the set of coordinates. The coordinates (X, Y, Z, Z^2, Z^3) are called *Chudnovsky Jacobian*; they are faster than projective coordinates with respect to both operations and allow faster additions and slower doublings compared to ordinary Jacobian coordinates. To obtain faster doublings it is useful to work with Cohen's modified Jacobian coordinates (X, Y, Z, aZ^4), where the elliptic curve is given by $y^2 = x^3 + ax + b$. This is very useful if one can store several precomputations and then use a binary (signed) window method to compute the scalar multiple as then there are much more doublings than additions. Furthermore, Cohen, Miyaji, and Ono investigate mixed coordinates, i. e. depending on the costs of inversions relative to multiplications they propose different sets of coordinates for the precomputations, the additions and the general doublings.
In this paper we try to mimic their approach to generalize it to hyperelliptic curves.

2 The New System of Coordinates

2.1 General Setting

Let us briefly recall the facts about arithmetic for genus two curves we will need here. Let $\mathbb{F}_q$ be a finite field of $q = p^r, p$ prime, elements, where p is the characteristic of the field. A genus two curve over $\mathbb{F}_q$ with at least one $\mathbb{F}_q$-rational Weierstraßpoint is given by

$$C : y^2 + (h_2x^2 + h_1x + h_0)y = x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0, h_i, f_i \in \mathbb{F}_q.$$

For short we write $y^2 + h(x)y = f(x)$. If $p \neq 5$ one easily gets $f_4 = 0$ by the substitution $x \mapsto x - f_4/5$. In odd characteristic one also obtains $h(x) \equiv 0$ by $y \mapsto y - h/2$. The group one uses is the ideal class group of the maximal order of the function field $\mathbb{F}_q(x, y)/(y^2 + h(x)y - f(x))$. Its size is $O(q^2)$. Each class can be represented by a pair of polynomials $[u, v]$, where $2 \geq \deg u > \deg v$, u is monic and $u|v^2 + h(x)v - f(x)$. Cantor's algorithm works with this representation as polynomials, however, if one uses the explicit formulae it is enough to consider a class as the quadruple $[u_1, u_0, v_1, v_0]$ with the interpretation as $[x^2 + u_1x + u_0, v_1x + v_0]$. To achieve that u is monic one needs one inversion per addition or doubling. We call these coordinates *affine* due to their similarity to elliptic affine coordinates.

For the complete investigation of affine addition formulae we refer to Lange [7].
In *projective coordinates* one lets $[U_1, U_0, V_1, V_0, Z]$ stand for $[x^2 + U_1/Z\, x + U_0/Z\, , V_1/Z\, x + V_0/Z]$. This allows to double and add without field inversions (see Lange [8] for details).

2.2 Weighted Coordinates

Now, we suggest to let $[U_1, U_0, V_1, V_0, Z_1, Z_2]$ correspond to the affine point $[x^2 + U_1/Z_1^2\, x + U_0/Z_1^2, V_1/Z_1^3 Z_2\, x + V_0/Z_1^3 Z_2]$. This means that now a point corresponds to a sextuple, thus one needs one more entry than for projective coordinates. If we compare this to the case of elliptic curves, for equal security the entries are of only half the size, thus the space requirements are similar.
We consider separately the cases $p = 2$ and $p \neq 2$. In the former case we also distinguish $h_2 = 0$ or not. Furthermore, we assume $f_4 = 0$ in any case. For each of the cases we also investigate the effects of enlarged sets of coordinates depending on the use of the system. We only consider the main cases of addition and doubling; for applications it is usually enough to implement these and check at the very end of every scalar multiplication if the result is a valid class, as the other cases occur with very low probability (only $O(1/q)$ and $q \sim 2^{80}$ in common use for cryptography) and one cannot run into division by zero as inversions are avoided. A complete study of all cases and their treatment (only in the affine case) can be found in [7] and the missing cases can easily be generalized from there.

3 Case of Odd Characteristic

In odd characteristic we assume that $h(x) \equiv 0$ and $f_4 = 0$. We do not list the pure approach but start with the enlarged set of coordinates $[U_1, U_0, V_1, V_0, Z_1, Z_0, z_1, z_2]$, where $z_1 = Z_1^2, z_2 = Z_2^2$. These additional entries are updated during each addition or doubling. z_2 is only used for the doublings but is computed for the new coordinates anyways. As additions occur at most half as often as doublings we do not include $Z_1 Z_2$ which would be useful for additions, because it is of less use to doublings and is not automatically updated. If space is more restricted such that we can only use the sextuple of coordinates, we need two extra squarings in the first step of the doubling or addition. We first list the algorithm to add two points in these coordinates and then consider doubling. Finally, we put both together these algorithms to compute scalar multiples, also paying attention to other systems of coordinates.

3.1 Addition

The addition algorithm given in the following table takes as input two points $[U_{11}, U_{10}, V_{11}, V_{10}, Z_{11}, Z_{12}, z_{11}, z_{12}]$, $[U_{21}, U_{20}, V_{21}, V_{20}, Z_{21}, Z_{22}, z_{21}, z_{22}]$ and outputs their sum. We always include the polynomial f (and later h for even characteristic) in the input to show which assumptions are made on the curve, although the coefficients might not be used in the algorithm.
If one computes a scalar multiple of a point given in affine coordinates and has the intermediate results not-normalized, then in the addition the intermediate result enters in the new weighted coordinates whereas the other class enters always as $[U_{11}, U_{10}, V_{11}, V_{10}, 1, 1, 1, 1]$. The number in brackets refer to this (cheaper) case. The next subsection concentrates on this special case.

Addition, odd characteristic		
Input	$[U_{11}, U_{10}, V_{11}, V_{10}, Z_{11}, Z_{12}, z_{11}, z_{12}], [U_{21}, U_{20}, V_{21}, V_{20}, Z_{21}, Z_{22}, z_{21}, z_{22}]$ $f = x^5 + f_3x^3 + f_2x^2 + f_1x + f_0$	
Output	$[U_1', U_0', V_1', V_0', Z_1', Z_2', z_1', z_2'] = [U_{11}, U_{10}, V_{11}, V_{10}, Z_{11}, Z_{12}, z_{11}, z_{12}]+$ $[U_{21}, U_{20}, V_{21}, V_{20}, Z_{21}, Z_{22}, z_{21}, z_{22}]$	
Step	Expression	Operations
1	precomputations: $z_{13} = Z_{11}Z_{12}$, $z_{23} = Z_{21}Z_{22}$, $z_{12} = z_{11}z_{13}$, $z_{22} = z_{21}z_{23}$; $\tilde{U}_{21} = U_{21}z_{11}$, $\tilde{U}_{20} = U_{20}z_{11}$, $\tilde{V}_{21} = V_{21}z_{12}$, $\tilde{V}_{20} = V_{20}z_{12}$;	8M (2M)
2	compute resultant r and precomputations: $y_1 = U_{11}z_{21} - \tilde{U}_{21}$, $y_2 = \tilde{U}_{20} - U_{10}z_{21}$, $y_3 = U_{11}y_1 + y_2z_{11}$; $r = y_2y_3 + y_1^2U_{10}$; $Z_2' = Z_{11}Z_{21}$, $\tilde{Z}_2 = Z_{12}Z_{22}$, $Z_1 = Z_2'^2$, $\tilde{Z}_2 = \tilde{Z}_2Z_1$, $\tilde{Z}_2 = \tilde{Z}_2r$; $Z_2' = Z_2'\tilde{Z}_2$, $\tilde{Z}_2 = \tilde{Z}_2^2$, $z_2' = Z_2'^2$;	4S, 11M (3S, 8M)
3	compute almost inverse: $inv_1 = y_1$, $inv_0 = y_3$;	
4	compute s: $w_0 = V_{10}z_{22} - \tilde{V}_{20}$, $w_1 = V_{11}z_{22} - \tilde{V}_{21}$, $w_2 = inv_0w_0$, $w_3 = inv_1w_1$; $s_1 = (inv_0 + z_{11}inv_1)(w_0 + w_1) - w_2 - w_3(z_{11} + U_{11})$; $s_0 = w_2 - U_{10}w_3$; If $s_1 = 0$ different case	8M (7M)
5	precomputations: $S_1 = s_1^2$, $S_0 = s_0Z_1$, $Z_1' = s_1Z_1$, $S = Z_1'S_0$, $S_0 = S_0^2$; $R = rZ_1'$, $s_0 = s_0Z_1'$, $s_1 = s_1Z_1'$, $z_1' = Z_1'^2$;	3S, 6M
6	compute l: $l_2 = s_1\tilde{U}_{21}$, $l_0 = s_0\tilde{U}_{20}$, $l_1 = (s_0 + s_1)(\tilde{U}_{20} + \tilde{U}_{21}) - l_0 - l_2$; $l_2 = l_2 + S$;	3M
7	compute U': $V_1' = R\tilde{V}_{21}$; $U_0' = S_0 + y_1(S_1(y_1 + \tilde{U}_{21}) - 2s_0) + y_2s_1 + 2V_1' + (2\tilde{U}_{21} + y_1)\tilde{Z}_2$; $U_1' = 2S - y_1s_1 - z_2'$;	6M
8	precomputations: $l_2 = l_2 - U_1'$, $w_0 = l_2U_0'$, $w_1 = l_2U_1'$;	2M
9	compute V': $V_1' = w_1 - z_1'(l_1 + V_1' - U_0')$; $V_0' = w_0 - z_1'(l_0 + R\tilde{V}_{20})$;	3M
total		7S, 47M (6S, 37M)

3.2 Mixed Addition

Here we assume that one input is in affine coordinates while the other is in the new coordinates. The output is in new coordinates, too.

Mixed Addition, odd characteristic		
Input	$[U_{11}, U_{10}, V_{11}, V_{10}], [U_{21}, U_{20}, V_{21}, V_{20}, Z_{21}, Z_{22}, z_{21}, z_{22}]$ $f = x^5 + f_3x^3 + f_2x^2 + f_1x + f_0$	
Output	$[U'_1, U'_0, V'_1, V'_0, Z'_1, Z'_2, z'_1, z'_2] = [U_{11}, U_{10}, V_{11}, V_{10}] + [U_{21}, U_{20}, V_{21}, V_{20}, Z_{21}, Z_{22}, z_{21}, z_{22}]$	
Step	Expression	Operations
1	precomputations and resultant: $z_{23} = Z_{21}Z_{22}$, $z_{22} = z_{21}z_{23}$, $y_1 = U_{11}z_{21} - U_{21}$; $y_2 = U_{20} - U_{10}z_{21}$, $y_3 = U_{11}y_1 + y_2$, $r = y_2y_3 + y_1^2U_{10}$;	1S, 7M
2	compute almost inverse: $inv_1 = y_1$, $inv_0 = y_3$;	
3	compute s: $w_0 = V_{10}z_{22} - V_{20}$, $w_1 = V_{11}z_{22} - V_{21}$, $w_2 = inv_0w_0$, $w_3 = inv_1w_1$; $s_1 = (inv_0 + inv_1)(w_0 + w_1) - w_2 - w_3(1 + U_{11})$; $s_0 = w_2 - U_{10}w_3$; If $s_1 = 0$ different case	7M
4	precomputations: $\tilde{Z}_2 = rz_{23}$, $Z'_2 = \tilde{Z}_2Z_{21}$, $\tilde{Z}_2 = \tilde{Z}_2^2$, $Z'_1 = s_1Z_{21}$, $R = rs_1$; $\tilde{s}_0 = s_0z_{21}$, $S_0 = s_0s_1$, $S = S_0z_{21}$, $S_1 = s_1^2$, $z'_1 = Z'^2_1$, $z'_2 = Z'^2_2$;	4S, 7M
5	compute l: $l_2 = S_1U_{21}$, $l_0 = S_0U_{20}$, $l_1 = (S_1 + S_0)(U_{21} + U_{20}) - l_0 - l_2$; $l_2 = l_2 + S$;	3M
6	compute U': $V'_1 = RV_{21}$; $U'_0 = (s_0 - U_{11}s_1)(\tilde{s}_0 - y_1s_1) + l_1 - U_{10}z'_1 + 2V'_1 + (2U_{21} + y_1)\tilde{Z}_2$; $U'_1 = 2S - y_1S_1 - z'_2$;	7M
7	precomputations: $l_2 = l_2 - U'_1$, $w_0 = l_2U'_0$, $w_1 = l_2U'_1$;	2M
8	compute V': $V'_1 = w_1 - z'_1(l_1 + V'_1 - U'_0)$; $V'_0 = w_0 - z'_1(l_0 + RV_{20})$;	3M
total		5S, 36M

3.3 Doubling

The formulae for doubling make obvious why we include $z_2 = Z_2^2$ as well.

Doubling, odd characteristic		
Input	$[U_1, U_0, V_1, V_0, Z_1, Z_2, z_1, z_2]$ $f = x^5 + f_3x^3 + f_2x^2 + f_1x + f_0$	
Output	$[U_1', U_0', V_1', V_0', Z_1', Z_2', z_1', z_2'] = 2[U_1, U_0, V_1, V_0, Z_1, Z_2, z_1, z_2]$	
Step	Expression	Operations
1	precomputation and resultant: $\tilde{U}_0 = U_0z_1$, $w_0 = V_1^2$, $w_1 = U_1^2$; $w_3 = V_0z_1 - U_1V_1$, $r = w_0U_0 + V_0w_3$; $\tilde{Z}_2 = Z_2r$, $\tilde{Z}_2 = \tilde{Z}_2z_1$, $Z_2' = 2\tilde{Z}_2Z_1$, $\tilde{Z}_2 = \tilde{Z}_2^2$;	3S, 8M
2	compute almost inverse: $inv_1 = -V_1$, $inv_0 = w_3$;	
3	compute k: $z_3 = z_1^2$, $w_3 = f_3z_3 + w_1$; $k_1 = z_2(2(w_1 - \tilde{U}_0) + w_3)$, $z_3 = z_3z_1$; $k_0 = z_2(U_1(4\tilde{U}_0 - w_3) + z_3f_2) - w_0$;	1S, 6M
4	compute $s = kinv \bmod u$: $w_0 = k_0inv_0$, $w_1 = k_1inv_1$; $s_1 = (inv_0 + inv_1)(k_0 + k_1) - w_0 - w_1(1 + U_1)$; $s_0 = w_0 - w_1\tilde{U}_0$; If $s_1 = 0$ different case	5M
5	precomputations: $S_0 = s_0^2$, $Z_1' = s_1z_1$, $z_1' = Z_1'^2$, $S = s_0Z_1'$; $R = rZ_1'$, $z_2' = Z_2'^2$, $s_0 = s_0s_1$, $s_1 = Z_1's_1$;	3S, 5M
6	compute l: $l_2 = s_1U_1$, $l_0 = s_0U_0$, $l_1 = (s_0 + s_1)(U_0 + U_1) - l_0 - l_2$; $l_2 = l_2 + S$;	3M
7	compute U': $V_1' = RV_1$; $U_0' = S_0 + 4(V_1' + 2\tilde{Z}_2U_1)$; $U_1' = 2S - z_2'$;	2M
8	precomputations: $l_2 = l_2 - U_1'$, $w_0 = l_2U_0'$, $w_1 = l_2U_1'$;	2M
9	compute V': $V_1' = w_1 - z_1'(l_1 + 2V_1' - U_0')$; $V_0' = w_0 - z_1'(l_0 + 2RV_0)$;	3M
total		7S, 34M

3.4 Different Sets of Coordinates

So far we have given algorithms to perform the computations within one system and also considered additions involving mixed coordinates. Now we are concerned with mixes of systems. To have suitable abbreviations, we denote by $\mathcal{C}_1 + \mathcal{C}_2 = \mathcal{C}_3$ the computation of an addition, where the first input is in coordinate system $\mathcal{C}_1$, the second in $\mathcal{C}_2$ and the output

Table 1: Addition and Doubling in Different Systems, Odd Characteristic

Doubling		**Addition**	
operation	costs	operation	costs
$2\mathcal{N} = \mathcal{P}$	7S, 38M	$\mathcal{N} + \mathcal{N} = \mathcal{P}$	7S, 51M
$2\mathcal{P} = \mathcal{P}$	6S, 37M	$\mathcal{N} + \mathcal{P} = \mathcal{P}$	4S, 51M
$2\mathcal{N} = \mathcal{N}$	7S, 34M	$\mathcal{N} + \mathcal{N} = \mathcal{N}$	7S, 47M
$2\mathcal{P} = \mathcal{N}$	6S, 34M	$\mathcal{N} + \mathcal{P} = \mathcal{N}$	5S, 45M
$2\mathcal{A} = \mathcal{P}$	5S, 25M	$\mathcal{P} + \mathcal{P} = \mathcal{P}$	5S, 45M
$2\mathcal{A} = \mathcal{N}$	5S, 21M	$\mathcal{P} + \mathcal{P} = \mathcal{N}$	5S, 41M
		$\mathcal{A} + \mathcal{N} = \mathcal{P}$	5S, 41M
		$\mathcal{A} + \mathcal{P} = \mathcal{P}$	3S, 39M
		$\mathcal{A} + \mathcal{N} = \mathcal{N}$	5S, 36M
		$\mathcal{A} + \mathcal{P} = \mathcal{N}$	3S, 35M
		$\mathcal{A} + \mathcal{A} = \mathcal{N}$	3S, 25M
$2\mathcal{A} = \mathcal{A}$	1I, 5S, 22M	$\mathcal{A} + \mathcal{A} = \mathcal{A}$	1I, 3S, 22M

is in $\mathcal{C}_3$. Similarly, $2\mathcal{C}_1 = \mathcal{C}_2$ denotes a doubling with input in system $\mathcal{C}_1$ and output in $\mathcal{C}_2$. We denote the affine systems by $\mathcal{A}$, the projective by $\mathcal{P}$ and the new by $\mathcal{N}$. In the following we estimate the costs of computing scalar multiples using various systems of coordinates. To have the figures in mind, the following Table 1 lists the costs for the most useful additions and doublings.

3.5 Scalar Multiples in Odd Characteristic

In this section we concentrate on the multiplication of k-folds kD, where k is an integer and D is an ideal class. For references how to compute the respective expansions of k see [2] and the references given therein.
Let $\ell = \lfloor \log_2 k \rfloor$, i.e. $k = \sum_{i=0}^{\ell+1} k_i 2^i$. The direct approach to compute kD for a given class D is to use binary double-and-add starting with the most significant bit of k. For every $k_i = 1$ we need to perform an addition as well as a doubling, for a 0 one only doubles. The density, i.e. the number of ones in the expansion divided by the length, is asymptotically $1/2$.
Here we deal with the ideal class group of hyperelliptic curves and the negative of a class is obtained by negating the coordinates V_i resp. v_i. Hence, signed binary expansions are useful. They have the lower density of $1/3$ if one uses a NAF (non-adjacent form) of the multiplier, and are approximately of the same length.
If we can afford some precomputations, windowing methods get interesting; we consider signed expansions here. Let the window be of width w. The expansions we consider are of the form

$$k = 2^{k_0}(2^{k_1}(\cdots 2^{k_{v-1}}(2^{k_v}W[v] + W[v-1])\cdots) + W[0]),$$

where $W[i]$ is an odd integer in the range $-2^w + 1 \leq W[i] \leq 2^w - 1$ for all i, $W[v] > 0, k_0 \geq 0$ and $k_i \geq w + 1$ for $i \geq 1$.
We now first consider systems without precomputations and then investigate good matches of coordinate systems for windowing methods. The reason for treating these cases separately

Table 2: Without Precomputations, Odd Characteristic

Systems	Cost
$2\mathcal{A} = \mathcal{A}$, $\mathcal{A} + \mathcal{A} = \mathcal{A}$	$\ell/3$(4I, 18S, 88M)
$2\mathcal{N} = \mathcal{N}$, $\mathcal{A} + \mathcal{N} = \mathcal{N}$	$\ell/3$(26S, 138M)
$2\mathcal{N} = \mathcal{N}$, $\mathcal{N} + \mathcal{P} = \mathcal{N}$	$\ell/3$(26S, 147M)

is that for precomputations the addition will involve the set of coordinates which is advantageous for the precomputations, whereas in the system without precomputations this choice depends on the efficiency of the mixed addition only.

3.5.1 No Precomputations

In this approach we perform ℓ doublings and $\ell/3$ additions per scalar multiple of length ℓ. The following Table 2 lists the number of operations depending on the coordinate system, details are given below. We assume ℓ to be large and therefore leave out the costs for moving from one system to the other as they occur only once. However, note that except for the first line, where inversions are assumed to be cheap, this conversion involves no divisions.

If inversions are relatively cheap, affine coordinates cannot be beaten, thus, if the class is given in a non-normalized system we first normalize it. This takes 1I, 4M for $\mathcal{P} \to \mathcal{A}$ and 1I, 7M for $\mathcal{N} \to \mathcal{A}$. Then we double ℓ and add $\ell/3$ times by the algorithms given in [7].
Otherwise, for affine input the new system is best, as the most common operation (doubling) is cheaper than in any other fixed system and the mixed addition is also fast.
If the input is in $\mathcal{P}$ and inversions are very expensive, we need to find two systems $\mathcal{C}_1$ and $\mathcal{C}_2$ such that $2\mathcal{C}_1 = \mathcal{C}_1$, $2\mathcal{C}_1 = \mathcal{C}_2$ and $\mathcal{P} + \mathcal{C}_2 = \mathcal{C}_1$ are as cheap as possible, the first being the most frequent operation. By Table 1 we choose $\mathcal{C}_1 = \mathcal{N}$. For $\mathcal{C}_2$ it is equal to choose $\mathcal{N}$ or $\mathcal{P}$, therefore we use $\mathcal{N}$ to save some bookkeeping. Thus the first doubling is done as $2\mathcal{P} = \mathcal{N}$ and all further as $2\mathcal{N} = \mathcal{N}$. There are approximately ℓ doublings $2\mathcal{N} = \mathcal{N}$ and $\ell/3$ additions $\mathcal{N} + \mathcal{P} = \mathcal{N}$.
If the input is in new coordinates we do the same except that the first doubling is $2\mathcal{N} = \mathcal{N}$ needing 1 more S and we use 4M for $\mathcal{N} \to \mathcal{P}$.
To compare, using only projective coordinates we would need $\ell/3$(23S, 156M) and only new coordinates results in $\ell/3$(28S, 149M).

3.5.2 Windowing Methods

To obtain the table of precomputed values, i.e. all odd multiples $W[i]D$ for $1 \leq W[i] \leq 2^w - 1$, we need $w - 1$ doublings and $2^{w-1} - 1$ additions.
Like in the previous case we distinguish cases depending on the relative cost of inversions. If inversions are not too expensive, the precomputations are performed in affine coordinates. To still trade off some inversions for multiplications, we make use of Montgomery's trick of

Table 3: Precomputations, Odd Characteristic

System	I	S	M
$\mathcal{A}$	$2^{w-1}+w-2$	$3\cdot 2^{w-1}+5w-8$	$22(2^{w-1}+w-2)$
$\mathcal{A}$	w	$3\cdot 2^{w-1}+5w-8$	$25\cdot 2^{w-1}+22w-50$
$\mathcal{P}$		$5\cdot 2^{w-1}+6w-11$	$45\cdot 2^{w-1}+37w-82$

Table 4: Windowing Method, Odd Characteristic

Systems	I	S	M
$2\mathcal{A}=\mathcal{A}$, $\mathcal{A}+\mathcal{A}=\mathcal{A}$	$n+\theta+\frac{n-\theta}{w+2}$	$5(n+\theta)+3\frac{n-\theta}{w+2}$	$22(n+\theta+\frac{n-\theta}{w+2})$
$2\mathcal{N}=\mathcal{N}$, $\mathcal{A}+\mathcal{N}=\mathcal{N}$		$7(n+\theta)+5\frac{n-\theta}{w+2}$	$34(n+\theta)+36\frac{n-\theta}{w+2}$
$2\mathcal{N}=\mathcal{N}$, $\mathcal{N}+\mathcal{P}=\mathcal{N}$		$7(n+\theta)+5\frac{n-\theta}{w+2}$	$34(n+\theta)+45\frac{n-\theta}{w+2}$

simultaneous inversions. Like in [2] we first compute $2D$, then $(3D, 4D)$, then $(5D, 7D, 8D)$, $\ldots$,$((2^{w-2}+1)D, \ldots, (2^{w-1}-1)D, 2^{w-1}D)$, and finally $((2^{w-1}+1)D, \ldots, (2^w-1)D)$. Computing m inversions simultaneously is done by 1I, $3(m-1)$M. Thus we need

$$w\text{I},\ w-1 \text{ class-doublings},\ 2^{w-1}-1 \text{ class-additions, and } 3(2^{w-1}-2) \text{ extra M}.$$

As most of the operations for the precomputations are additions, we choose projective coordinates $\mathcal{P}$ if we want to perform the precomputations avoiding inversions. The costs for these two approaches leading to $\mathcal{A}$ and also for precomputations in $\mathcal{P}$ are listed in Table 3.
If inversions are cheap we stick to the affine system to compute the scalar multiplication. If we can afford the w inversions (or one more for non-affine input) to do the precomputations in affine, we use the new system for doublings, and perform the additions using the new mixed system $\mathcal{A}+\mathcal{N}=\mathcal{N}$. Finally, if inversions are very expensive, the best match is obtained if one uses projective coordinates for the precomputations, and the doublings are performed as $2\mathcal{N}=\mathcal{N}$. Then the addition is done as $\mathcal{N}+\mathcal{P}=\mathcal{N}$. This approach is equal to that of the previous subsection with non-affine input. Again, here we did not need a second system $\mathcal{C}_2$ for doublings.
In the main loop we perform $K=\sum_{i=0}^{v} k_i$ doublings and v additions. To ease the formulae, we assume $K=\ell-w/2+\theta$, $v=(\ell-w/2-\theta)/(w+2)$, where $\theta=1/2-1/(w+2)$. Let $n=\ell-w/2$. The costs are listed in Table 4. Some more computations can be saved using the tricks of [2]. Again we leave out the costs for conversions. For the precomputations see Table 3.

4 Case of Even Characteristic with $h_2 \neq 0$

In this section we first state the algorithms to compute with the new system in even characteristic. Here, we assume $h_2 \neq 0$, however, the formulae hold universally. The more special case $h_2 = 0$ is considered separately in the following section. Like before it is interesting to include some precomputations in the coordinates which are updated during each iteration. In this approach we let:

$$z_1 = Z_1^2, z_2 = Z_2^2, z_3 = Z_1 Z_2.$$

This turns out to be useful for both additions and doublings. While the costs remain unchanged for doublings if one additionally puts $z_4 = z_1 z_3$ one saves 1M in the addition. Thus, here $\mathcal{N}$ denotes $[U_1, U_0, V_1, V_0, Z_1, Z_2, z_1, z_2, z_3, z_4]$. We first consider addition and mixed addition, then doubling and then turn our attention to the computation of scalar multiples.

4.1 Addition

Here we assume that both classes are in $\mathcal{N}$. If one is in $\mathcal{A}$ the costs are given in brackets.

Addition, even characteristic, $h_2 \neq 0$		
Input	$[U_{11}, U_{10}, V_{11}, V_{10}, Z_{11}, Z_{12}, z_{11}, z_{12}, z_{13}, z_{14}]$, $[U_{21}, U_{20}, V_{21}, V_{20}, Z_{21}, Z_{22}, z_{21}, z_{22}, z_{23}, z_{24}]$ $h = h_2 x^2 + h_1 x + h_0,\ f = x^5 + f_3 x^3 + f_2 x^2 + f_1 x + f_0$	
Output	$[U_1', U_0', V_1', V_0', Z_1', Z_2', z_1', z_2', z_3', z_4'] = [U_{11}, U_{10}, V_{11}, V_{10}, Z_{11}, Z_{12}, z_{11}, z_{12}, z_{13}, z_{14}] +$ $[U_{21}, U_{20}, V_{21}, V_{20}, Z_{21}, Z_{22}, z_{21}, z_{22}, z_{23}, z_{24}]$	
Step	Expression	Operations
1	precomputations: $\tilde{U}_{21} = U_{21} z_{11}$, $\tilde{U}_{20} = U_{20} z_{11}$, $\tilde{V}_{21} = V_{21} z_{14}$, $\tilde{V}_{20} = V_{20} z_{14}$; $Z_1 = z_{11} z_{21}$, $Z_3 = z_{13} z_{23}$;	6M (–)
2	compute resultant r of U_1, U_2: $y_1 = U_{11} z_{21} + \tilde{U}_{21}$, $y_2 = U_{10} z_{21} + \tilde{U}_{20}$, $y_3 = U_{11} y_1 + y_2 z_{11}$; $r = y_2 y_3 + y_1^2 U_{10}$, $\tilde{Z}_2 = r Z_3$, $Z_2' = \tilde{Z}_2 Z_1$;	1S, 8M (1S, 7M)
3	compute almost inverse: $inv_1 = y_1$, $inv_0 = y_3$;	
4	compute s: $w_0 = V_{10} z_{24} + \tilde{V}_{20}$, $w_1 = V_{11} z_{24} + \tilde{V}_{21}$, $w_2 = inv_0 w_0$, $w_3 = inv_1 w_1$; $s_1 = (inv_0 + inv_1 z_{11})(w_0 + w_1) + w_2 + w_3(z_{11} + U_{11})$; $s_0 = w_2 + w_3 U_{10}$; If $s_1 = 0$ different case	8M (7M)
5	precomputations: $\tilde{s}_0 = s_0 Z_1$, $S_0 = \tilde{s}_0^2$, $Z_1' = s_1 Z_1$, $R = r Z_1'$; $t = y_1(y_1 + \tilde{U}_{21})$, $U_1' = y_1 s_1$, $s_1 = s_1 Z_1'$, $s_0 = s_0 Z_1'$; $z_1' = Z_1'^2$, $z_2' = Z_2'^2$, $z_3' = Z_1' Z_2'$, $z_4' = z_1' z_3'$;	3S, 9M
6	compute l: $l_2 = s_1 \tilde{U}_{21}$, $l_0 = s_0 \tilde{U}_{20}$, $l_1 = (s_0 + s_1)(\tilde{U}_{20} + \tilde{U}_{21}) + l_0 + l_2$;	3M
7	compute U': $U_0' = S_0 + tU_1' + y_2 s_1 + Z_2'(h_2(\tilde{s}_0 + t) + h_1 Z_1' + y_1 \tilde{Z}_2)$; $U_1' = U_1' Z_1' + h_2 z_3' + z_2'$;	5M
8	precomputations: $l_2 = l_2 + Z_1' \tilde{s}_0 + h_2 z_3' + U_1'$, $w_0 = l_2 U_0'$, $w_1 = l_2 U_1'$;	3M
9	compute V': $V_1' = w_1 + z_1'(l_1 + R\tilde{V}_{21} + U_0') + h_1 z_4'$; $V_0' = w_0 + z_1'(l_0 + R\tilde{V}_{20}) + h_0 z_4'$;	4M
total		4S, 46M (4S, 38M)

4.2 Mixed Addition

If one knows in advance that $\mathcal{A}+\mathcal{N}=\mathcal{N}$ shall be computed, it is more useful to apply the following algorithm to save one more squaring.

Mixed Addition, even characteristic, $h_2 \neq 0$		
Input	$[U_{11},U_{10},V_{11},V_{10}],[U_{21},U_{20},V_{21},V_{20},Z_{21},Z_{22},z_{21},z_{22},z_{23},z_{24}]$ $h=h_2x^2+h_1x+h_0,\ f=x^5+f_3x^3+f_2x^2+f_1x+f_0$	
Output	$[U_1',U_0',V_1',V_0',Z_1',Z_2',z_1',z_2',z_3',z_4']=[U_{11},U_{10},V_{11},V_{10}]+$ $[U_{21},U_{20},V_{21},V_{20},Z_{21},Z_{22},z_{21},z_{22},z_{23},z_{24}]$	
Step	Expression	Operations
1	precomputations and resultant: $y_1=U_{11}z_{21}+U_{21},\ y_2=U_{20}+U_{10}z_{21},\ y_3=U_{11}y_1+y_2;$ $r=y_2y_3+y_1^2U_{10},\ \tilde{Z}_2=rz_{23},\ Z_2'=\tilde{Z}_2Z_{21};$	1S, 7M
2	compute almost inverse: $inv_1=y_1,\ inv_0=y_3;$	
3	compute s: $w_0=V_{10}z_{24}+V_{20},\ w_1=V_{11}z_{24}+V_{21},\ w_2=inv_0w_0,\ w_3=inv_1w_1;$ $s_1=(inv_0+inv_1)(w_0+w_1)-w_2-w_3(1+U_{11});$ $s_0=w_2+U_{10}w_3;$ If $s_1=0$ different case	7M
4	precomputations: $Z_1'=s_1Z_{21},\ R=rs_1,\ \tilde{s}_0=s_0Z_{21},\ S_0=s_0s_1,\ S_1=s_1^2;$ $\tilde{Z}_2=\tilde{Z}_2^2,\ z_1'=Z_1'^2,\ z_2'=Z_2'^2,\ z_3'=Z_1'Z_2',\ z_4'=z_1'z_3';$	4S, 6M
5	compute l: $l_2=S_1U_{21},\ l_0=S_0U_{20},\ l_1=(S_1+S_0)(U_{21}+U_{20})-l_0-l_2;$	3M
6	compute U': $U_1'=y_1S_1;$ $U_0'=\tilde{s}_0(\tilde{s}_0+h_2Z_2')+U_{11}(h_2z_3'+U_1')+S_1y_2+h_1z_3'+y_1\tilde{Z}_2;$ $U_1'=U_1'+z_2'+h_2z_3';$	5M
7	precomputations: $l_2=l_2+Z_1'\tilde{s}_0+h_2z_3'+U_1',\ w_0=l_2U_0',\ w_1=l_2U_1';$	3M
8	compute V': $V_1'=w_1+z_1'(l_1+RV_{21}+U_0')+h_1z_4';$ $V_0'=w_0+z_1'(l_0+RV_{20})+h_0z_4';$	4M
total		5S, 35M

4.3 Doubling

Now finally we consider doublings.

Doubling, even characteristic, $h_2 \neq 0$		
Input	$[U_1, U_0, V_1, V_0, Z_1, Z_2, z_1, z_2, z_3, z_4]$ $h = h_2x^2 + h_1x + h_0,\ f = x^5 + f_3x^3 + f_2x^2 + f_1x + f_0$	
Output	$[U_1', U_0', V_1', V_0', Z_1', Z_2', z_1', z_2', z_3', z_4'] = 2[U_1, U_0, V_1, V_0, Z_1, Z_2, z_1, z_2, z_3, z_4]$	
Step	Expression	Operations
1	precomputation and resultant: $\tilde{V}_1 = h_1z_1 + h_2U_1$, $\tilde{V}_0 = h_0z_1 + h_2U_0$, $z_5 = z_1^2$; $w_0 = V_1^2$, $w_1 = U_1^2$, $w_2 = h_1^2z_5 + h_2^2w_1$; $w_3 = z_1(h_1U_1 + h_2U_0 + h_0z_1) + h_2w_1$; $r = w_2U_0 + \tilde{V}_0w_3$, $\tilde{Z}_2 = z_3r$, $Z_2' = \tilde{Z}_2z_4$;	3S, 5M
2	compute almost inverse: $inv_1 = \tilde{V}_1$, $inv_0 = w_3$;	
3	compute k: $w_3 = f_3z_5 + w_1$; $k_1 = w_3z_2 + V_1h_2z_3$; $k_0 = U_1k_1 + w_0 + z_4(V_1h_1 + V_0h_2 + f_2z_4)$;	6M
4	compute $s = kinv \bmod u$: $w_0 = k_0inv_0$, $w_1 = k_1inv_1$; $s_1 = (inv_0 + inv_1)(k_0 + k_1) + w_0 + w_1(1 + U_1)$; $s_0 = w_0 + U_0w_1z_1$; If $s_1 = 0$ different case	6M
5	precomputations: $t = h_2s_0 + s_1(h_2U_1 + h_1z_1)$, $Z_1' = s_1z_1$, $S_0 = s_0^2$, $z_1' = {Z_1'}^2$; $S = s_0Z_1'$, $R = \tilde{Z}_2Z_1'$, $s_0 = s_0s_1$, $s_1 = Z_1's_1$; $z_2' = {Z_2'}^2$, $z_3' = Z_1'Z_2'$, $z_4' = z_1'z_3'$;	3S, 8M
6	compute l: $l_2 = s_1U_1$, $l_0 = s_0U_0$, $l_1 = (s_1 + s_0)(U_1 + U_0) - l_0 - l_2$; $l_2 = l_2 + S + h_2z_3'$;	3M
7	compute U': $U_0' = S_0 + Z_2't$; $U_1' = z_2' + h_2z_3'$;	1M
8	precomputations: $l_2 = l_2 + U_1'$, $w_0 = l_2U_0'$, $w_1 = l_2U_1'$;	2M
9	compute V': $V_1' = w_1 - z_1'(l_1 + RV_1 + U_0') + z_4'h_1$; $V_0' = w_0 - z_1'(l_0 + RV_0) + z_4'h_0$;	4M
total		6S, 35M

4.4 Different Sets of Coordinates

Using the same abbreviations as above, we state the costs for the operations in different coordinate systems in Table 5.

Table 5: Different Systems, Even Characteristic, $h_2 \neq 0$

Doubling		**Addition**	
operation	costs	operation	costs
$2\mathcal{N} = \mathcal{P}$	6S, 37M	$\mathcal{N} + \mathcal{N} = \mathcal{P}$	4S, 48M
$2\mathcal{P} = \mathcal{P}$	6S, 36M	$\mathcal{N} + \mathcal{P} = \mathcal{P}$	4S, 48M
$2\mathcal{P} = \mathcal{N}$	6S, 35M	$\mathcal{N} + \mathcal{N} = \mathcal{N}$	4S, 46M
$2\mathcal{N} = \mathcal{N}$	6S, 35M	$\mathcal{N} + \mathcal{P} = \mathcal{N}$	4S, 47M
$2\mathcal{A} = \mathcal{P}$	5S, 24M	$\mathcal{P} + \mathcal{P} = \mathcal{P}$	4S, 46M
$2\mathcal{A} = \mathcal{N}$	5S, 20M	$\mathcal{P} + \mathcal{P} = \mathcal{N}$	4S, 45M
		$\mathcal{A} + \mathcal{P} = \mathcal{P}$	3S, 39M
		$\mathcal{A} + \mathcal{N} = \mathcal{P}$	5S, 37M
		$\mathcal{A} + \mathcal{P} = \mathcal{N}$	3S, 38M
		$\mathcal{A} + \mathcal{N} = \mathcal{N}$	5S, 35M
		$\mathcal{A} + \mathcal{A} = \mathcal{N}$	4S, 29M
$2\mathcal{A} = \mathcal{A}$	1I, 5S, 20M	$\mathcal{A} + \mathcal{A} = \mathcal{A}$	1I, 3S, 21M

Table 6: Without Precomputations, Even Characteristic, $h_2 \neq 0$

Systems	Cost
$2\mathcal{A} = \mathcal{A}$, $\mathcal{A} + \mathcal{A} = \mathcal{A}$	$\ell/3$(4I, 18S, 82M)
$2\mathcal{N} = \mathcal{N}$, $\mathcal{A} + \mathcal{N} = \mathcal{N}$	$\ell/3$(23S, 140M)
$2\mathcal{N} = \mathcal{N}$, $\mathcal{N} + \mathcal{P} = \mathcal{N}$	$\ell/3$(22S, 152M)

4.5 Computation of Scalar Multiples

We follow the same lines as in the previous section.

4.5.1 No Precomputations

For cheap inversions one again uses the affine system alone. If one wants to avoid inversions and has an affine input (or can allow 1I to achieve this) we perform the doublings as $2\mathcal{N} = \mathcal{N}$ and the addition as $\mathcal{A} + \mathcal{N} = \mathcal{N}$. For non-normalized input we work as in odd characteristic $\mathcal{C}_1 = \mathcal{C}_2 = \mathcal{N}$, $\mathcal{C}_3 = \mathcal{P}$.

4.5.2 Windowing Methods

To obtain the table of precomputed values we need $w-1$ doublings and $2^{w-1}-1$ additions. Like before we choose either $\mathcal{C}_3 = \mathcal{A}$ or $\mathcal{C}_3 = \mathcal{P}$

In the main loop we perform $K = \sum_{i=0}^{v} k_i$ doublings and v additions. The costs are listed in Table 8 for the most useful matches of sets of coordinates. Again we leave out the costs for conversions and mention that some constant number of operations can be saved if one considers in more detail the first doubling and the final addition/doubling like in [2].

Table 7: Precomputations, Even Characteristic, $h_2 \neq 0$

System	I	S	M
$\mathcal{A}$	$2^{w-1} + w - 2$	$3 \cdot 2^{w-1} + 5w - 8$	$21 \cdot 2^{w-1} + 20w - 41$
$\mathcal{A}$	w	$3 \cdot 2^{w-1} + 5w - 8$	$24 \cdot 2^{w-1} + 20w - 47$
$\mathcal{P}$		$2^{w+1} + 6w - 10$	$47 \cdot 2^{w-1} + 40w - 87$

Table 8: Windowing Method, Even Characteristic, $h_2 \neq 0$

Systems	I	S	M
$2\mathcal{A} = \mathcal{A}$, $\mathcal{A} + \mathcal{A} = \mathcal{A}$	$n + \theta + \frac{n-\theta}{w+2}$	$5(n+\theta) + 3\frac{n-\theta}{w+2}$	$17(n+\theta) + 21\frac{n-\theta}{w+2})$
$2\mathcal{N} = \mathcal{N}$, $\mathcal{A} + \mathcal{N} = \mathcal{N}$		$6(n+\theta) + 5\frac{n-\theta}{w+2}$	$35((n+\theta) + \frac{n-\theta}{w+2})$
$2\mathcal{N} = \mathcal{N}$, $\mathcal{N} + \mathcal{P} = \mathcal{N}$		$6(n+\theta) + 4\frac{n-\theta}{w+2}$	$35(n+\theta) + 47\frac{n-\theta}{w+2}$

Compared to the results in odd characteristic this case is a bit more expensive. On the other hand the arithmetic in binary fields is easier to implement and usually faster. Thus in the end this still might turn out to be better. In the next section we consider the sub-case $h_2 = 0$ in more detail. There the complexity is lower than for odd characteristic in any case.

5 Case of Even Characteristic with $h_2 = 0$

Obviously this case can be considered as a special case of the previous section. However, if one restricts the algorithms to this still very frequent case one can save some operations. Like before we suggest to enlarge the set of coordinates by $z_1 = Z_1^2, z_2 = Z_2, z_3 = Z_1Z_2, z_4 = Z_1^3Z_2$. To save space we can discard Z_2.

5.1 Addition

Addition, even characteristic, $h_2 = 0$	
Input	$[U_{11}, U_{10}, V_{11}, V_{10}, Z_{11}, Z_{12}, z_{11}, z_{12}, z_{13}, z_{14}], [U_{21}, U_{20}, V_{21}, V_{20}, Z_{21}, Z_{22}, z_{21}, z_{22}, z_{23}, z_{24}]$ $h = h_1 x + h_0,\ f = x^5 + f_3 x^3 + f_2 x^2 + f_1 x + f_0$
Output	$[U'_1, U'_0, V'_1, V'_0, Z'_1, Z'_2, z'_1, z'_2, z'_3 z'_4] = [U_{11}, U_{10}, V_{11}, V_{10}, Z_{11}, Z_{12}, z_{11}, z_{12}, z_{13}, z_{14}] +$ $[U_{21}, U_{20}, V_{21}, V_{20}, Z_{21}, Z_{22}, z_{21}, z_{22}, z_{23}, z_{24}]$

Step	Expression	Operations
1	precomputations: $\tilde{U}_{21} = U_{21} z_{11}$, $\tilde{U}_{20} = U_{20} z_{11}$, $\tilde{V}_{21} = V_{21} z_{14}$, $\tilde{V}_{20} = V_{20} z_{14}$; $Z_1 = z_{11} z_{21}$, $Z_3 = z_{13} z_{23}$;	6M (–)
2	compute resultant r of U_1, U_2: $y_1 = U_{11} z_{21} + \tilde{U}_{21}$, $y_2 = U_{10} z_{21} + \tilde{U}_{20}$, $y_3 = U_{11} y_1 + y_2 z_{11}$; $r = y_2 y_3 + y_1^2 U_{10}$, $\tilde{Z}_2 = r Z_3$, $Z'_2 = \tilde{Z}_2 Z_1$, $\tilde{Z}_2 = \tilde{Z}_2^2$, $\tilde{Z}_2 = \tilde{Z}_2 Z_1$;	2S, 9M (2S, 8M)
3	compute almost inverse: $inv_1 = y_1$, $inv_0 = y_3$;	
4	compute s: $w_0 = V_{10} z_{24} + \tilde{V}_{20}$, $w_1 = V_{11} z_{24} + \tilde{V}_{21}$, $w_2 = inv_0 w_0$, $w_3 = inv_1 w_1$; $s_1 = (inv_0 + inv_1 z_{11})(w_0 + w_1) + w_2 + w_3(z_{11} + U_{11})$; $s_0 = w_2 + w_3 U_{10}$; If $s_1 = 0$ different case	8M (7M)
5	precomputations: $S_1 = s_1^2$, $Z'_1 = s_1 Z_1$, $R = r Z'_1$, $S_0 = s_0 Z_1$, $S = S_0 Z'_1$; $S_0 = S_0^2$, $z'_1 = Z_1'^2$, $z'_2 = Z_2'^2$, $z'_3 = Z'_1 Z'_2$, $z'_4 = z'_1 z'_3$, $s_1 = s_1 Z'_1$, $s_0 = s_0 Z'_1$;	4S, 8M
6	compute l: $l_2 = s_1 \tilde{U}_{21}$, $l_0 = s_0 \tilde{U}_{20}$, $l_1 = (s_0 + s_1)(\tilde{U}_{20} + \tilde{U}_{21}) + l_2 + l_0$; $l_2 = l_2 + S$;	3M
7	compute U': $U'_0 = S_0 + y_1(S_1(y_1 + \tilde{U}_{21}) + \tilde{Z}_2) + y_2 s_1 + h_1 z'_3$; $U'_1 = y_1 s_1 + z'_2$;	4M
8	precomputations: $l_2 = l_2 + U'_1$, $w_0 = l_2 U'_0$, $w_1 = l_2 U'_1$;	2M
9	compute V': $V'_1 = w_1 + z'_1(l_1 + R\tilde{V}_{21} + U'_0) + z'_4 h_1$; $V'_0 = w_0 + z'_1(l_0 + R\tilde{V}_{20}) + z'_4 h_0$;	4M
total		6S, 44M (6S, 36M)

5.2 Mixed Addition

Now we deal with the special case $\mathcal{A}+\mathcal{N}=\mathcal{N}$.

Mixed Addition, even characteristic, $h_2=0$		
Input	$[U_{11},U_{10},V_{11},V_{10}],[U_{21},U_{20},V_{21},V_{20},Z_{21},Z_{22},z_{21},z_{22},z_{23},z_{24}]$ $h=h_1x+h_0,\ f=x^5+f_3x^3+f_2x^2+f_1x+f_0$	
Output	$[U_1',U_0',V_1',V_0',Z_1',Z_2',z_1',z_2',z_3',z_4']=[U_{11},U_{10},V_{11},V_{10}]+$ $[U_{21},U_{20},V_{21},V_{20},Z_{21},Z_{22},z_{21},z_{22},z_{23},z_{24}]$	
Step	Expression	Operations
1	precomputations and resultant : $y_1=U_{11}z_{21}+U_{21},\ y_2=U_{20}+U_{10}z_{21},\ y_3=U_{11}y_1+y_2;$ $r=y_2y_3+y_1^2U_{10},\ \tilde{Z}_2=rz_{23},\ Z_2'=\tilde{Z}_2Z_{21},\ \tilde{Z}_2=\tilde{Z}_2^2;$	2S, 7M
2	compute almost inverse: $inv_1=y_1,\ inv_0=y_3;$	
3	compute s: $w_0=V_{10}z_{24}+V_{20},\ w_1=V_{11}z_{24}+V_{21},\ w_2=inv_0w_0,\ w_3=inv_1w_1;$ $s_1=(inv_0+inv_1)(w_0+w_1)-w_2-w_3(1+U_{11});$ $s_0=w_2+U_{10}w_3;$ If $s_1=0$ different case	7M
4	precomputations: $Z_1'=s_1Z_{21},\ R=rs_1,\ S_0=s_0Z_{21},\ S=S_0Z_1',\ S_0=S_0^2;$ $s_0=s_0s_1,\ s_1=s_1^2,\ z_1'=Z_1'^2,\ z_2'=Z_2'^2,\ z_3'=Z_1'Z_2',\ z_4'=z_1'z_3';$	4S, 7M
5	compute l: $l_2=s_1U_{21},\ l_0=s_0U_{20},\ l_1=(s_1+s_0)(U_{21}+U_{20})-l_0-l_2;$ $l_2=l_2+S;$	3M
6	compute U': $U_0'=S_0+y_1(s_1U_{11}+\tilde{Z}_2)+y_2s_1+h_1z_3';$ $U_1'=y_1s_1+z_2';$	4M
7	precomputations: $l_2=l_2+U_1',\ w_0=l_2U_0',\ w_1=l_2U_1';$	2M
8	compute V': $V_1'=w_1+z_1'(l_1+RV_{21}+h_1z_3'+U_0');$ $V_0'=w_0+z_1'(l_0+RV_{20}+h_0z_3');$	4M
total		6S, 34M

5.3 Doubling

Doubling, even characteristic, $h_2 = 0$		
Input	$[U_1, U_0, V_1, V_0, Z_1, Z_2, z_1, z_2, z_3, z_4]$ $h = h_1 x + h_0,\ f = x^5 + f_3 x^3 + f_2 x^2 + f_1 x + f_0$	
Output	$[U_1', U_0', V_1', V_0', Z_1', Z_2', z_1', z_2', z_3', z_4'] = 2[U_1, U_0, V_1, V_0, Z_1, Z_2, z_1, z_2, z_3, z_4]$	
Step	Expression	Operations
1	resultant: $r = h_1(h_1 U_0 + U_1 h_0) + h_0^2 z_1,\ \tilde{Z}_2 = r z_4,\ Z_2' = \tilde{Z}_2 z_4;$	2M
2	compute almost inverse: $inv_1 = h_1,\ inv_0 = h_1 U_1 + h_0 z_1;$	
3	compute k: $w_0 = V_1^2,\ w_1 = U_1^2,\ z_5 = z_1^2;$ $k_1 = z_2(f_3 z_5 + w_1);$ $k_0 = U_1 k_1 + w_0 + z_4(f_2 z_4 + V_1 h_1);$	3S, 5M
4	compute s: $w_0 = k_0 inv_0,\ w_1 = k_1 inv_1;$ $s_1 = (inv_0 + inv_1)(k_0 + k_1) + w_0 + (1 + U_1) w_1;$ $s_0 = w_0 + U_0 w_1 z_1;$ If $s_1 = 0$ different case	6M
5	precomputations: $Z_1' = s_1 z_1,\ S_0 = s_0^2,\ S = s_0 Z_1',\ R = \tilde{Z}_2 Z_1';$ $z_1' = Z_1'^2,\ z_2' = Z_2'^2,\ z_3' = Z_1' Z_2',\ z_4' = z_1' z_3';$ $s_0 = s_0 s_1,\ s_1 = Z_1' s_1;$	3S, 7M
6	compute l: $l_2 = s_1 U_1,\ l_0 = s_0 U_0,\ l_1 = (s_1 + s_0)(U_1 + U_0) - l_0 - l_2;$ $l_2 = l_2 + S;$	3M
7	compute U': $U_0' = S_0 + h_1 z_3';$ $U_1' = z_2';$	
8	precomputations: $l_2 = l_2 - U_1',\ w_0 = l_2 U_0',\ w_1 = l_2 U_1';$	2M
9	compute V': $V_1' = w_1 + z_1'(l_1 + R V_1 + U_0') + z_3' h_1;$ $V_0' = w_0 + z_1'(l_0 + R V_0) + z_3' h_0;$	4M
total		6S, 29M

5.4 Different Sets of Coordinates

Finally, we state the number of operations in the case of even characteristic and with $h_2 = 0$ in Table 9.

5.5 Computation of Scalar Multiples

Table 9 reveals that for this case the situation is different – additions involving $\mathcal{N}$ are less expensive than those involving $\mathcal{P}$. The reason for this might as well be that this special case was not considered in too much detail in Lange [8].

Table 9: Different Systems, Even Characteristic, $h_2 = 0$

Doubling		**Addition**	
operation	costs	operation	costs
$2\mathcal{P} = \mathcal{P}$	6S, 33M	$\mathcal{N} + \mathcal{P} = \mathcal{P}$	4S, 48M
$2\mathcal{N} = \mathcal{P}$	6S, 31M	$\mathcal{N} + \mathcal{N} = \mathcal{P}$	6S, 46M
$2\mathcal{P} = \mathcal{N}$	6S, 32M	$\mathcal{N} + \mathcal{P} = \mathcal{N}$	4S, 47M
$2\mathcal{N} = \mathcal{N}$	6S, 29M	$\mathcal{P} + \mathcal{P} = \mathcal{P}$	4S, 46M
$2\mathcal{A} = \mathcal{P}$	6S, 21M	$\mathcal{N} + \mathcal{N} = \mathcal{N}$	6S, 44M
$2\mathcal{A} = \mathcal{N}$	6S, 19M	$\mathcal{P} + \mathcal{P} = \mathcal{N}$	4S, 45M
		$\mathcal{A} + \mathcal{P} = \mathcal{P}$	3S, 39M
		$\mathcal{A} + \mathcal{N} = \mathcal{P}$	6S, 36M
		$\mathcal{A} + \mathcal{P} = \mathcal{N}$	3S, 38M
		$\mathcal{A} + \mathcal{N} = \mathcal{N}$	6S, 34M
		$\mathcal{A} + \mathcal{A} = \mathcal{N}$	5S, 25M
$2\mathcal{A} = \mathcal{A}$	1I, 5S, 17M	$\mathcal{A} + \mathcal{A} = \mathcal{A}$	1I, 3S, 21M

Table 10: Without Precomputations, Even Characteristic, $h_2 = 0$

Systems	Cost
$2\mathcal{A} = \mathcal{A}$, $\mathcal{A} + \mathcal{A} = \mathcal{A}$	$\ell/3$(4I, 18S, 73M)
$2\mathcal{N} = \mathcal{N}$, $\mathcal{A} + \mathcal{N} = \mathcal{N}$	$\ell/3$(24S, 121M)
$2\mathcal{N} = \mathcal{N}$, $\mathcal{N} + \mathcal{N} = \mathcal{N}$	$\ell/3$(24S, 131M)

5.5.1 No Precomputations

For cheap inversions one again uses the affine system alone. If one wants to avoid inversions and has an affine input (or can allow 1I to achieve this) we do the same as in the general case and perform the doublings as $2\mathcal{N} = \mathcal{N}$ and the addition as $\mathcal{A} + \mathcal{N} = \mathcal{N}$. For non-normalized input we here suggest to use $\mathcal{C}_1 = \mathcal{C}_2 = \mathcal{C}_3 = \mathcal{N}$.

5.5.2 Windowing Methods

To obtain the table of precomputed values we need $w - 1$ doublings and $2^{w-1} - 1$ additions. Here we choose either $\mathcal{C}_3 = \mathcal{A}$ or $\mathcal{C}_3 = \mathcal{N}$
Table 12 states the number of operations for the most useful matches of sets of coordinates.

6 Conclusion and Outlook

As mentioned before one can save some constant number of operations in considering the first and last additions separately. This is worthwhile for an implementation where the system of coordinates of the input and output are fixed.
Since the field size for hyperelliptic curve cryptography is of only half the bit size of that of

Table 11: Precomputations, Even Characteristic, $h_2 = 0$

System	I	S	M
$\mathcal{A}$	$2^{w-1}+w-2$	$3\cdot 2^{w-1}+5w-8$	$21\cdot 2^{w-1}+17w-38$
$\mathcal{A}$	w	$3\cdot 2^{w-1}+5w-8$	$24\cdot 2^{w-1}+17w-44$
$\mathcal{N}$		$6(2^{w-1}+w-2)$	$44\cdot 2^{w-1}+29w-73$

Table 12: Windowing Method, Even Characteristic, $h_2 = 0$

Systems	I	S	M
$2\mathcal{A}=\mathcal{A}$, $\mathcal{A}+\mathcal{A}=\mathcal{A}$	$n+\theta+\frac{n-\theta}{w+2}$	$5(n+\theta)+3\frac{n-\theta}{w+2}$	$20(n+\theta)+22\frac{n-\theta}{w+2})$
$2\mathcal{N}=\mathcal{N}$, $\mathcal{A}+\mathcal{N}=\mathcal{N}$		$6((n+\theta)+\frac{n-\theta}{w+2})$	$29(n+\theta)+34\frac{n-\theta}{w+2}$
$2\mathcal{N}=\mathcal{N}$, $\mathcal{N}+\mathcal{N}=\mathcal{N}$		$6((n+\theta)+\frac{n-\theta}{w+2})$	$29(n+\theta)+44\frac{n-\theta}{w+2}$

elliptic curve cryptography the operations are approximately half as slow. Comparing the results with the similar ones for elliptic curves one can assume that the complexity of scalar multiplications is similar. It would be very interesting to have a hardware implementation comparing these two systems.

A further possibility to choose coordinates is to allow even more entries to have a finer distinction. One notices that U_1' and V_1' are divisible by Z_1. Perhaps this can lead to further savings.

References

[1] D.G. Cantor. Computing in the Jacobian of a hyperelliptic curve. *Math. Comp.*, 48:95–101, 1987.

[2] H. Cohen, A. Miyaji, and T. Ono. Efficient elliptic curve exponentiation using mixed coordinates. In *Proceedings of Asiacrypt'98*, volume 1514 of *Lecture Notes in Comput. Sci.*, pages 51–65. Springer, New–York, 1998.

[3] R. Harley. Fast arithmetic on genus 2 curves.
available at `http://cristal.inria.fr/~harley/hyper`, 2000.

[4] N. Koblitz. Hyperelliptic cryptosystems. *J. of Cryptology*, 1:139–150, 1989.

[5] U. Krieger. Anwendung hyperelliptischer Kurven in der Kryptographie. Master's thesis, Universität Gesamthochschule Essen, 1997.

[6] T. Lange. *Efficient Arithmetic on Hyperelliptic Curves.* PhD thesis, Universität Gesamthochschule Essen, 2001.

[7] 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/ or http://www.itsc.ruhr-uni-bochum.de/tanja.

[8] T. Lange. Inversion-Free Arithmetic on Genus 2 Hyperelliptic Curves. Cryptology ePrint Archive, Report 2002/147, 2002. http://eprint.iacr.org/ or http://www.itsc.ruhr-uni-bochum.de/tanja.

[9] K. Matsuo, J. Chao, and S. Tsujii. Fast genus two hyperellptic curve cryptosysytems. Technical Report ISEC2001-23, IEICE, 2001. pages 89-96.

[10] Y. Miyamoto, H. Doi, K. Matsuo, J. Chao, and S. Tsuji. A fast addition algorithm of genus two hyperelliptic curve. In *Proc. of SCIS2002, IEICE Japan*, pages 497–502, 2002. in Japanese.

[11] A.M. Spallek. *Kurven vom Geschlecht 2 und ihre Anwendung in Public-Key-Kryptosystemen.* PhD thesis, Universität Gesamthochschule Essen, 1994.

[12] H. Sugizaki, K. Matsuo, J. Chao, and S. Tsujii. An Exntension of Harley algorithm addition algorithm for hyperelliptic curves over finite fields of characteritic two. Technical Report ISEC2002-9(2002-5), IEICE, 2002. pages 49-56.

[13] M. Takahashi. Improving Harley Algorithms for Jacobians of genus 2 Hyperelliptic Curves. In *Proc. of SCIS2002, IEICE Japan*, 2002. in Japanese.