Efficient Arithmetic on Elliptic and Hyperelliptic Curves

Peter Birkner

Department of Mathematics and Computer Science
Eindhoven University of Technology

Tutorial on Elliptic and Hyperelliptic Curve Cryptography
4 September 2007, Dublin

Introduction

Main question of this talk:

How can we efficiently compute on elliptic and hyperelliptic curves?

Outline

- Part I : General Concepts
- Part II : Fast arithmetic on elliptic curves (characteristic > 3)
- Part III : Fast arithmetic on hyperelliptic curves (characteristic 2)

Part I : General concepts

Double-and-Add (1)

Algorithm 1 (Double-and-Add)

IN: An element $P \in G$ and a positive integer $n = (n_{l-1} \dots n_0)$ where $n_{l-1} = 1$.

OUT: The element $[n]P \in G$.

1. $R \leftarrow P$
2. `for` $i = l - 2$ `to` 0 `do`
    1. $R \leftarrow [2]R$
    2. `if` $n_i = 1$ `then` $R \leftarrow R \oplus P$
    3. $i \leftarrow i - 1$
3. `return` R

Double-and-Add (2)

The algorithm uses the following rule:

$$[(n_{l-1} \dots n_i)_2]P = [2]([(n_{l-1} \dots n_{i+1})_2]P) \oplus [n_i]P$$

Example: $45 = (101101)_2$, compute $[45]P$

P
$[2]P$
$[2]([2]P) \oplus P$
$[2]([2]([2]P) \oplus P) \oplus P$
$[2]([2]([2]([2]P) \oplus P) \oplus P)$
$[2]([2]([2]([2]([2]P) \oplus P) \oplus P)) \oplus P \quad = [45]P$

Double-and-Add (3)

Remarks

- $w(n)$ denotes the Hamming weight of n. This is the number of nonzero digits in the binary representation of n.
- Double-and-Add needs $\ell - 1$ doublings and $w(n)$ additions $\Rightarrow$ doubling is more important than addition!
- **Variant:** Use NAF instead of binary representation (Double-and-Add/Subtract). This is a signed bit representation using the digits $1, 0, -1$.
- In NAF no two consecutive digits are non-zero. The number of additions/subtractions is less than in binary form. We need to compute $-P$ efficiently.

Windowing Methods (1)

The 2^k-ary Method

- Use a larger basis to get sparse representations of n
- A common choice is 2^k as basis
- $S = \{0, 1, \ldots, 2^k - 1\}$ are the digits
- To perform scalar multiplication, first precompute $[s]P$ for all $s \in S$ and use a modified version of Algorithm 1

Example $k = 3$, $S = \{0, 1, 2, 3, 4, 5, 6, 7\}$

$n = 241 = (11|110|001)_2 = (361)_{2^3}$

Windowing Methods (2)

Algorithm 2 (Left–to–right 2^k-ary)

IN: An element $P \in G$ and a positive integer n in 2^k-ary representation $n = (n_{l-1} \dots n_0)_{2^k}$ Precomputed values $P, [2]P, \cdots, [2^k - 1]P$

OUT: The element $[n]P \in G$.

1. $R \leftarrow [n_{l-1}]P$
2. `for` $i = l - 2$ `to` 0 `do`
   1. $R \leftarrow [2^k]R$
   2. `if` $n_i \neq 0$ `then` $R \leftarrow R \oplus [n_i]P$
   3. $i \leftarrow i - 1$
3. `return` R

Windowing Methods (3)

Example $k = 3$, $S = \{0, 1, 2, 3, 4, 5, 6, 7\}$

$n = 241 = (361)_{2^3}$

Precompute the values $P, [2]P, \ldots, [7]P$

$R = [3]P$ (Start with $[3]P$)

$R = [2^3]R = [24]P$ (Multiply by 2^3 and add $[6]P$)
$R = R \oplus [6]P = [30]P$

$R = [2^3]R = [240]P$ (Multiply by 2^3 and add P)
$R = R \oplus [1]P = [241]P$

Sliding Window Method

- To reduce the number of precomputations, a **sliding window method** can be used!
- Digits are only the **odd** integers smaller than 2^k and 0
- $S = \{0, 1, 3, 5, \ldots, 2^k - 1\}$
- Consecutive zeros are skipped
- Scan from right to left $\Rightarrow$ block is odd
- **Example** $(k = 3)$

$$241 = (\underline{1}\ \underline{111}\ 000\ \underline{1})_2$$

- Sliding window is also possible with signed digits!
- Compute P, 2P, 4P, 8P, 15P, 30P, 60P, 120P, 240P, 241P

Part II : Fast arithmetic on elliptic curves (characteristic > 3)

Elliptic Curves in Characteristic $p > 3$

An elliptic curve over $\mathbb{F}_p$ can be given by an equation of the form

$$E : y^2 = x^3 + ax + b,$$

where $a, b \in \mathbb{F}_p$ and the discriminant $4a^3 + 27b^2 \neq 0$.

The points on the curve E are the pairs (x, y) which satisfy the curve equation. The set of $\mathbb{F}_p$-rational points (i.e. points with coeffs. in $\mathbb{F}_p$) form an additive group, denoted by $E(\mathbb{F}_p)$.

Arithmetic on Elliptic Curves

The most important operation (for cryptography) on $E(\mathbb{F}_p)$ is scalar multiplication $[n]P = \underbrace{P + \ldots + P}_{n-times}$ for an integer n and a point $P \in E(\mathbb{F}_p)$.

Use for instance double-and-add or windowing methods to compute $[n]P$.

We need fast addition and doubling functions on $E(\mathbb{F}_p)$!

(Doubling is the most important function in double-and-add algorithms.)

Choice of Coordinate System

An elliptic curve can be represented using several coordinate systems. For each such system, the speed of additions and doublings is different.

We consider affine, projective and Jacobian coordinates and give the appropriate formulas and the number of operations.

Affine Coordinates

Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two points on E

- (Addition) $R = P \oplus Q$ can be computed using the following formulas, where $x_1 \neq x_2$:

$$x_3 = \lambda^2 - x_1 - x_2, \quad y_3 = \lambda(x_1 - x_3) - y_1,$$

  where $\lambda = (y_2 - y_1)/(x_2 - x_1)$. **Complexity:** 1I+2M+1S

- (Doubling) $[2]P = P \oplus P$ can be computing using the formulas:

$$x_3 = \lambda^2 - 2x_1, \quad y_3 = \lambda(x_1 - x_3) - y_1$$

  where $\lambda = (3x_1^2 + a)/(2y_1)$. **Complexity:** 1I+2M+2S

Projective Coordinates (1)

Let $x = X/Z$ and $y = Y/Z$. $P = (X, Y, Z)$. We consider the homogenised curve equation

$$E_P : Y^2 Z = X^3 + aXZ^2 + bZ^3.$$

Let $P = (X_1, Y_1, Z_1)$ and $Q = (X_2, Y_2, Z_2)$ be on E_P and $R = P \oplus Q$

Addition: $R = P \oplus Q = (X_3, Y_3, Z_3)$

$$X_3 = vA,\ Y_3 = u(v^2 X_1 Z_2 - A),\ Z_3 = v^3 Z_1 Z_2,$$

where $u = Y_2 Z_1 - Y_1 Z_2$, $v = X_2 Z_1 - X_1 Z_2$ and $A = u^2 Z_1 Z_2 - v^3 - 2v^2 X_1 Z_2$

Complexity: 12M+2S

Projective Coordinates (2)

Let $P = (X_1, Y_1, Z_1)$.

Doubling: $[2]P = P \oplus P = (X_3, Y_3, Z_3)$

$$X_3 = hs,\ Y_3 = w(B-h) - 2RR,\ Z_3 = sss,$$

where $XX = X_1^2$, $ZZ = Z_1^2$, $w = aZZ + 3XX$, $s = 2Y_1Z_1$, $ss = s^2$, $sss = s \cdot ss$, $R = Y_1 s$, $RR = R^2$, $B = (X_1 + R)^2 - XX - RR$, $h = w^2 - 2B$

Complexity: 6M+6S (see Explicit-Formulas Database, http://www.hyperelliptic.org/EFD)

Jacobian Coordinates (1)

In Jacobian coordinates, we set $x = X/Z^2$ and $y = Y/Z^3$. The weighted homogenised curve equation is:

$$E_J : Y^2 = X^3 + aXZ^4 + bZ^6.$$

Let $P = (X_1, Y_1, Z_1)$ and $Q = (X_2, Y_2, Z_2)$ be on E_J.

The **addition** $P \oplus Q = (X_3, Y_3, Z_3)$ works as follows:

$$X_3 = r^2 - J - 2V,\ \ Y_3 = r(V - X_3) - 2S_1J,\ \ Z_3 = ((Z_1 + Z_2)^2 - ZZ_1 - ZZ_2)H,$$

where $ZZ_1 = Z_1^2$, $ZZ_2 = Z_2^2$, $U_1 = X_1ZZ_2$, $U_2 = X_2ZZ_1$, $S_1 = Y_1Z_2ZZ_2$, $S_2 = Y_2Z_1ZZ_1$, $H = U_2 - U_1$, $I = (2H)^2$, $J = HI$, $r = 2(S_2 - S_1)$, $V = U_1I$

The **complexity** of the addition is: 11M+5S

Jacobian Coordinates (2)

Let $P = (X_1, Y_1, Z_1)$ be on E_J.

The **doubling** $[2]P = P \oplus P = (X_3, Y_3, Z_3)$ works as follows:

$$X_3 = T, \;\; Y_3 = M(S-T) - 8YYYY, \;\; Z_3 = (Y1 + Z1)^2 - YY - ZZ,$$

where $XX = X_1^2$, $YY = Y_1^2$, $YYYY = YY^2$, $ZZ = Z_1^2$,
$S = 2((X_1 + YY)^2 - XX - YYYY)$, $M = 3XX + aZZ^2$,
$T = M^2 - 2S$

The **complexity** of the doubling is: 2M+8S

Summary: Arithmetic on Elliptic Curves

Addition and doubling on elliptic curves using different coordinate systems (characteristic $p > 3$)

Coordinates	Addition	Doubling	Doubling*
Affine	1I+2M+1S	1I+2M+2S	14,6M
Projective	12M+2S	6M+6S	10,8M
Jacobian	11M+5S	2M+8S	8,4M

(Doubling* : As an example we assume $I/M = 11$ and $S/M = 0,8$.)

Part III : Fast arithmetic on hyperelliptic curves (characteristic 2)

Hyperelliptic Curves in Characteristic 2

A hyperelliptic curve C (of genus 2) over $\mathbb{F}_{2^d}$ can be given by an equation of the form

$$C : y^2 + h(x)y = f(x),$$

where $h \in \mathbb{F}_{2^d}[X]$ is of degree at most 2 and $f \in \mathbb{F}_{2^d}[X]$ is monic of degree 5.

The points on the curve C do **not** form a group! We use the divisor class group of C instead. (Recall, that a divisor is a formal sum of points.)

Goal: We want to have fast doubling in the divisor class group!

Mumford Representation

How to represent elements of the divisor class group?

Theorem (Mumford)

Let C be a hyperelliptic curve of genus g over an arbitrary field K. Each nontrivial divisor class of C over K can be represented by a unique pair of polynomials $u, v \in K[x]$, where

1. *u is monic,*
2. $\deg v < \deg u \leq g$,
3. $u \mid v^2 + vh - f$.

Example: ($C : y^2 + xy = x^5 + x^4 + x^2 + x$ over $\mathbb{F}_{2^7}$)
$D = [x^2 + z^{31}x + z^{61},\ z^{43}x + z^{90}]$ is a proper divisor class of C, where z is a generator of $\mathbb{F}_{2^7}^*$.

Group Law of the Divisor Class Group (1)

Algorithm 14.7 Cantor's algorithm

INPUT: Two divisor classes $\overline{D}_1 = [u_1, v_1]$ and $\overline{D}_2 = [u_2, v_2]$ on the curve $C : y^2 + h(x)y = f(x)$.
OUTPUT: The unique reduced divisor D such that $\overline{D} = \overline{D}_1 \oplus \overline{D}_2$.

1. $d_1 \leftarrow \gcd(u_1, u_2)$ $\quad [d_1 = e_1u_1 + e_2u_2]$
2. $d \leftarrow \gcd(d_1, v_1 + v_2 + h)$ $\quad [d = c_1d_1 + c_2(v_1 + v_2 + h)]$
3. $s_1 \leftarrow c_1e_1,\ s_2 \leftarrow c_1e_2$ and $s_3 \leftarrow c_2$
4. $u \leftarrow \dfrac{u_1u_2}{d^2}$ and $v \leftarrow \dfrac{s_1u_1v_2 + s_2u_2v_1 + s_3(v_1v_2 + f)}{d} \bmod u$
5. **repeat**
6. $\quad u' \leftarrow \dfrac{f - vh - v^2}{u}$ and $v' \leftarrow (-h - v) \bmod u'$
7. $\quad u \leftarrow u'$ and $v \leftarrow v'$
8. **until** $\deg u \leqslant g$
9. make u monic
10. **return** $[u, v]$

(in: Handbook of Elliptic and Hyperelliptic Curve Cryptography, p. 308)

Group Law of the Divisor Class Group (2)

Remarks:

1. Cantor's algorithm is completely general and holds for all genera, all fields and all (valid) inputs.
2. From this algorithm all the explicit formulas are derived.
3. To get efficient formulas, we use Cantor but restrict to very special cases, e.g. restricting to genus 2, even characteristic and taking $\overline{D_1} = \overline{D_2}$ leads to efficient doubling formulae.
4. The efficiency of the formulas heavily depends on the degree of the polynomial h in the curve equation (see later).

Doubling in Genus 2 with General $h(x)$

Doubling, $\deg u = 2$			
Input Output	$[u,v], u = x^2+u_1x+u_0, v = v_1x+v_0$ $[u',v'] = 2[u,v]$		
Step	Expression	odd	even
1	compute $\tilde{v} \equiv (h+2v) \bmod u = \tilde{v}_1x+\tilde{v}_0$: $\tilde{v}_1 = h_1+2v_1-h_2u_1,\ \tilde{v}_0 = h_0+2v_0-h_2u_0;$		
2	compute resultant $r = \mathrm{res}(\tilde{v},u)$: $w_0 = v_1^2,\ w_1 = u_1^2,\ w_2 = \tilde{v}_1^2,\ w_3 = u_1\tilde{v}_1,\ r = u_0w_2+\tilde{v}_0(\tilde{v}_0-w_3);$	2S, 3M $(w_2 = 4w_0)$	2S, 3M (see below)
3	compute almost inverse $inv' = invr$: $inv'_1 = -\tilde{v}_1,\ inv'_0 = \tilde{v}_0-w_3;$		
4	compute $k' = (f-hv-v^2)/u \bmod u = k'_1x+k'_0$: $w_3 = f_3+w_1,\ w_4 = 2u_0,\ k'_1 = 2(w_1-f_4u_1)+w_3-w_4-h_2v_1;$ $k'_0 = u_1(2w_4-w_3+f_4u_1+h_2v_1)+f_2-w_0-2f_4u_0-h_1v_1-h_2v_0;$	1M	2M (see below)
5	compute $s' = k'inv' \bmod u$: $w_0 = k'_0inv'_0,\ w_1 = k'_1inv'_1,\ s'_1 = (inv'_0+inv'_1)(k'_0+k'_1)-w_0-w_1(1+u_1),\ s'_0 = w_0-u_0w_1;$	5M	5M
6	compute $s'' = x+s_0/s_1$ and s_1: $w_1 = 1/(rs'_1)(=1/r^2s_1),\ w_2 = rw_1(=1/s'_1),\ w_3 = s'^2_1w_1(=s_1);$ $w_4 = rw_2(=1/s_1),\ w_5 = w_4^2,\ s''_0 = s'_0w_2;$	I, 2S, 5M	I, 2S, 5M
7	compute $l' = s''u = x^3+l'_2x^2+l'_1x+l'_0$: $l'_2 = u_1+s''_0,\ l'_1 = u_1s''_0+u_0,\ l'_0 = u_0s''_0;$	2M	2M
8	compute $u' = s^2+(h+2v)s/u+(v^2+hv-f)/u^2$: $u'_0 = s''^2_0+w_4(h_2(s''_0-u_1)+2v_1+h_1)+w_5(2u_1-f_4),\ u'_1 = 2s''_0+h_2w_4-w_5;$	S, 2M	S, M
9	compute $v' \equiv -h-(l+v) \bmod u' = v'_1x+v'_0$: $w_1 = l'_2-u'_1,\ w_2 = u'_1w_1+u'_0-l'_1,\ v'_1 = w_2w_3-v_1-h_1+h_2u'_1;$ $w_2 = u'_0w_1-l'_0,\ v'_0 = w_2w_3-v_0-h_0+h_2u'_0;$	4M	4M
total			each I, 5S, 22M

Lange and Stevens (SAC 2004)

Restrict to the case $\deg(h) = 1$ (genus 2, characteristic 2)

Doubling $\deg h = 1$**,** $\deg u = 2$				
Input Output	$[u, v], u = x^2 + u_1 x + u_0, v = v_1 x + v_0$; h_1^2, h_1^{-1} $[u', v'] = 2[u, v]$			
Step	Expression	$h_1 = 1$	h_1^{-1} small	h_1 arbitrary
1	compute rs_1: $z_0 = u_0^2$, $k_1' = u_1^2 + f_3$; $w_0 = f_0 + v_0^2 (= rs_1'/h_1^3)$;	3S	3S	3S
2	compute $1/s_1$ and s_0'': $w_1 = (1/w_0) z_0 (= h_1/s_1)$; $z_1 = k_1' w_1$, $s_0'' = z_1 + u_1$;	I, 2M	I, 2M	I, 2M
3	compute u': $w_2 = h_1^2 w_1$, $u_1' = w_2 w_1$; $u_0' = s_0''^2 + w_2$;	2S	S, 2M	S, 2M
4	compute v': $w_3 = w_2 + k_1'$; $v_1' = h_1^{-1}(w_3 z_1 + w_2 u_1' + f_2 + v_1^2)$; $v_0' = h_1^{-1}(w_3 u_0' + f_1 + z_0)$;	S, 3M	S, 3M	S, 5M
total		I, 6S, 5M	I, 5S, 7M	I, 5S, 9M

Inversion-free Doubling Formulae

Why do we need inversion-free formulae?

- In hardware applications an inverter might be very expensive (compared to other components).
- I/M-ratio depends on the CPU or platform.
- **Example:** IBM Thinkpad X41 with Linux and GMP has an I/M ratio between 10 and 13 for $\mathbb{F}_{2^d}$, where $7 \leq d \leq 113$.

  Asuming I/M = 13 and S/M = 0.8, the affine doubling forumulae 1I+5M+6S correspond to 13M+5M+4.8M=22.8M.

  Can we do better in this setting?

Inversion-free Doubling with Weighted Coordinates

In **projective coordinates** a divisor class is now represented by the quintuple $[U_1, U_0, V_1, V_0, Z]$, corresponding to the divisor class $[x^2 + U_1/Zx + U_0/Z,\ V_1/Zx + V_0/Z]$ in Mumford representation.

In **recent coordinates** $[U_1, U_0, V_1, V_0, Z, z]$ corresponds to $[x^2 + U_1/Zx + U_0/Z,\ V_1/Z^2x + V_0/Z^2]$ and $z = Z^2$.

New coordinates: $[U_1, U_0, V_1, V_0, Z_1, Z_2]$ corresponds to $[x^2 + U_1/Z_1^2x + U_0/Z_1^2,\ V_1/(Z_1^3Z_2)x + V_0/(Z_1^3Z_2)]$.

Inversion-free Doubling in Projective Coordinates

Doubling			
Input	$[U_1, U_0, V_1, V_0, Z]$		
Output	$[U_1', U_0', V_1', V_0', Z'] = 2[U_1, U_0, V_1, V_0, Z]$		
Step	Expression	odd	even
1	compute resultant and precomputations: $\tilde{h}_1 = h_1 Z, \tilde{h}_0 = h_0 Z, Z_2 = Z^2, \tilde{V}_1 = \tilde{h}_1 + 2V_1 - h_2 U_1;$ $\tilde{V}_0 = \tilde{h}_0 + 2V_0 - h_2 U_0, w_0 = V_1^2, w_1 = U_1^2, w_2 = \tilde{V}_1^2;$ $w_3 = \tilde{V}_0 Z - U_1 \tilde{V}_1, r = \tilde{V}_0 w_3 + w_2 U_0;$	3S, 4M $(w_2 = 4w_0)$	4S, 6M
2	compute almost inverse: $inv_1 = -\tilde{V}_1, inv_0 = w_3;$		
3	compute k: $w_3 = f_3 Z_2 + w_1, w_4 = 2U_0;$ $k_1 = 2w_1 + w_3 - Z(w_4 + 2f_4 U_1 + h_2 V_1);$ $k_0 = U_1(Z(2w_4 + f_4 U_1 + h_2 V_1) - w_3)$ $+Z(Z(f_2 Z - V_1 h_1 - V_0 h_2 - 2f_4 U_0) - w_0)$	5M (see below)	5M (see below)
4	compute $s = k inv \bmod u$: $w_0 = k_0 inv_0, w_1 = k_1 inv_1;$ $s_3 = (inv_0 + inv_1)(k_0 + k_1) - w_0 - (1 + U_1) w_1$ $s_1 = s_3 Z, s_0 = w_0 - Z U_0 w_1;$ If $s_1 = 0$ different case	7M	7M
5	precomputations: $R = Z_2 r, \tilde{R} = R s_1, S_1 = s_1^2, S_0 = s_0^2, t = h_2 s_0;$ $s_1 = s_1 s_3, s_0 = s_0 s_3, S = s_0 Z, \tilde{\tilde{R}} = \tilde{R} s_1;$	2S, 6M	2S, 6M
6	compute l: $l_2 = U_1 s_1, l_0 = U_0 s_0, l_1 = (s_1 + s_0)(U_1 + U_0) - l_2 - l_0;$	3M	3M
7	compute U': $U_0' = S_0 + R(s_3(2V_1 - h_2 U_1 + \tilde{h}_1) + t + Zr(2U_1 -$ $f_4 Z));$ $U_1' = 2S + h_2 \tilde{R} - R^2;$	1S, 4M	1S, 2M
8	precomputations: $l_2 = l_2 + S - U_1', w_0 = U_0' l_2 - S_1 l_0, w_1 = U_1' l_2 +$ $S_1(U_0' - l_1);$	4M	4M
9	adjust: $Z' = S_1 \tilde{R}, U_1' = \tilde{R} U_1', U_0' = \tilde{R} U_0';$	3M	3M
10	compute V': $V_0' = w_0 + h_2 U_0' - \tilde{\tilde{R}}(V_0 - \tilde{h}_0);$ $V_1' = w_1 + h_2 U_1' - \tilde{\tilde{R}}(V_1 - \tilde{h}_1);$	2M	2M
total		6S, 38M	7S, $\leq$38M

(in: Tanja Lange: Formulae for Arithmetic on Genus 2 Hyperelliptic Curves, AAECC 2004)

Inversion-free Doubling in Recent Coordinates

Algorithm 14.51 Doubling in recent coordinates ($g = 2$, $h_2 = 0$, and q even)

INPUT: A divisor class $[U_1, U_0, V_1, V_0, Z, z]$ and the precomputed values h_1^2 and h_1^{-1}.
OUTPUT: The divisor class $[U_1', U_0', V_1', V_0', Z', z'] = [2][U_1, U_0, V_1, V_0, Z, z]$.

1. ***precomputations*** $[10\mathrm{M} + 4\mathrm{S}]$
   $Z_4 \leftarrow z^2$, $y_0 \leftarrow U_0^2$, $t_1 \leftarrow U_1^2 + f_3 z$ and $w_0 \leftarrow Z_4 f_0 + V_0^2$
   $\bar{Z} \leftarrow z w_0$, $w_1 \leftarrow y_0 Z_4$, $y_1 \leftarrow t_1 y_0 z$ and $s_0 \leftarrow y_1 + U_1 w_0 Z$
   $w_2 \leftarrow h_1^2 w_1$ and $w_3 \leftarrow w_2 + t_1 w_0$
2. ***compute U'*** $[2\mathrm{M} + \mathrm{S}]$
   $U_1' \leftarrow w_2 w_1$, $w_2 \leftarrow w_2 \bar{Z}$ and $U_0' \leftarrow s_0^2 + w_2$
3. ***compute V'*** $[11\mathrm{M} + 3\mathrm{S}]$
   $Z' \leftarrow \bar{Z}^2$ and $V_1' \leftarrow h_1^{-1}\big(w_2 U_1' + (w_3 y_1 + f_2 Z' + (V_1 w_0)^2) Z'\big)$
   $V_0' \leftarrow h_1^{-1}\big(\bar{Z}(w_3 U_0' + y_0 w_0 Z')\big)$, $z' \leftarrow Z'^2$
4. **return** $[U_1', U_0', V_1', V_0', Z', z']$ [total complexity: $23\mathrm{M} + 8\mathrm{S}$]

(in: Handbook of Elliptic and Hyperelliptic Curve Cryptography, p. 347)

If $h_1 = 1$ one saves 3M. Hence, for $h(x) = x$ a doubling costs 20M+8S.

Summary: Doubling on HEC

Doubling on hyperelliptic curves using different coordinate systems in characteristic 2

Coordinates	DBL with general $h(x)$	DBL with $h(x) = x$
Affine	1I+22M+5S	1I+5M+6S
Projective	38M+7S	22M+6S
New	39M+6S	n/a
Recent	n/a	20M+8S

Conclusion: If we restrict to special cases in terms of genus, field (characteristic) and degree of h, and choose the right coordinates, we get the best performance for scalar multiplication on HEC.

Thank you for your attention!