Efficient Doubling on Genus Two Curves over Binary Fields (SAC 2004)

Marc Stevens
Eindhoven University of Technology

Tanja Lange
Ruhr-Universität Bochum

Overview

- Elliptic Curves
- Hyperelliptic Curves
- HEC of Genus 2
- Comparisons

Elliptic Curves

- Koblitz/Miller (1985)

  Use additive group of points on an elliptic curve:

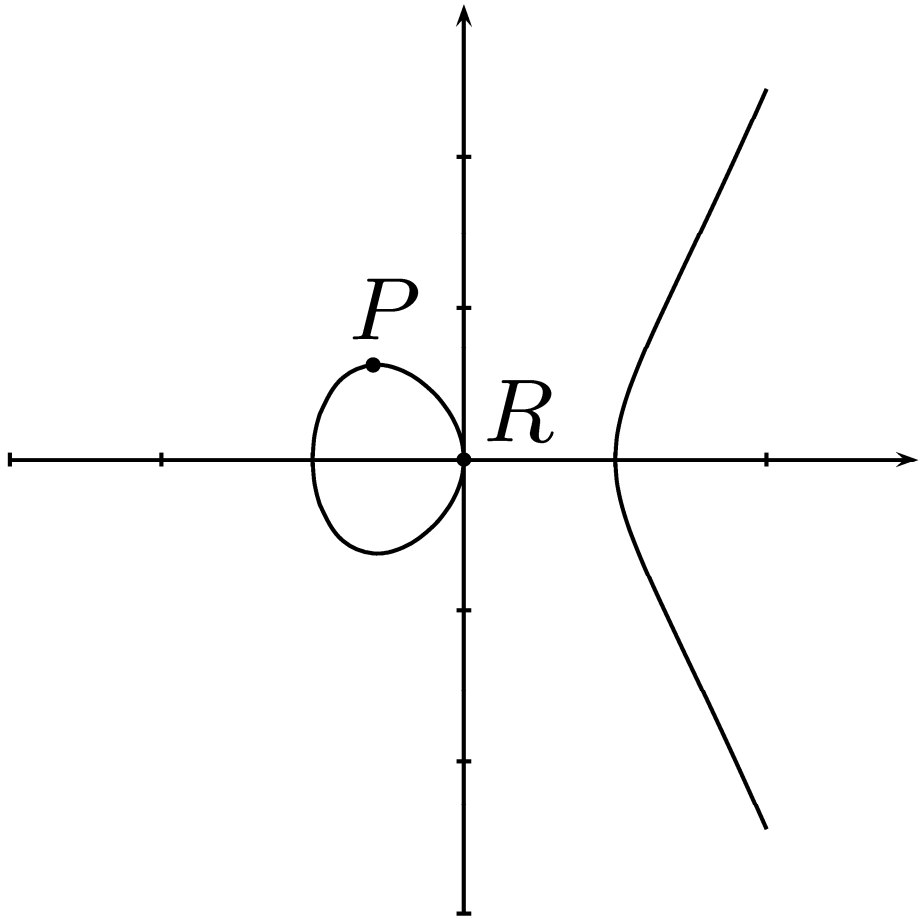
$$G = \{ (x, y) \in \mathbb{F}_q^2 \, : \, y^2 = x^3 + f_4 x + f_6 \} \cup \{ \infty \}$$

$$G = \{ (x, y) \in \mathbb{F}_q^2 \, : \, y^2 + xy = x^3 + f_2 x^2 + f_6 \} \cup \{ \infty \}$$

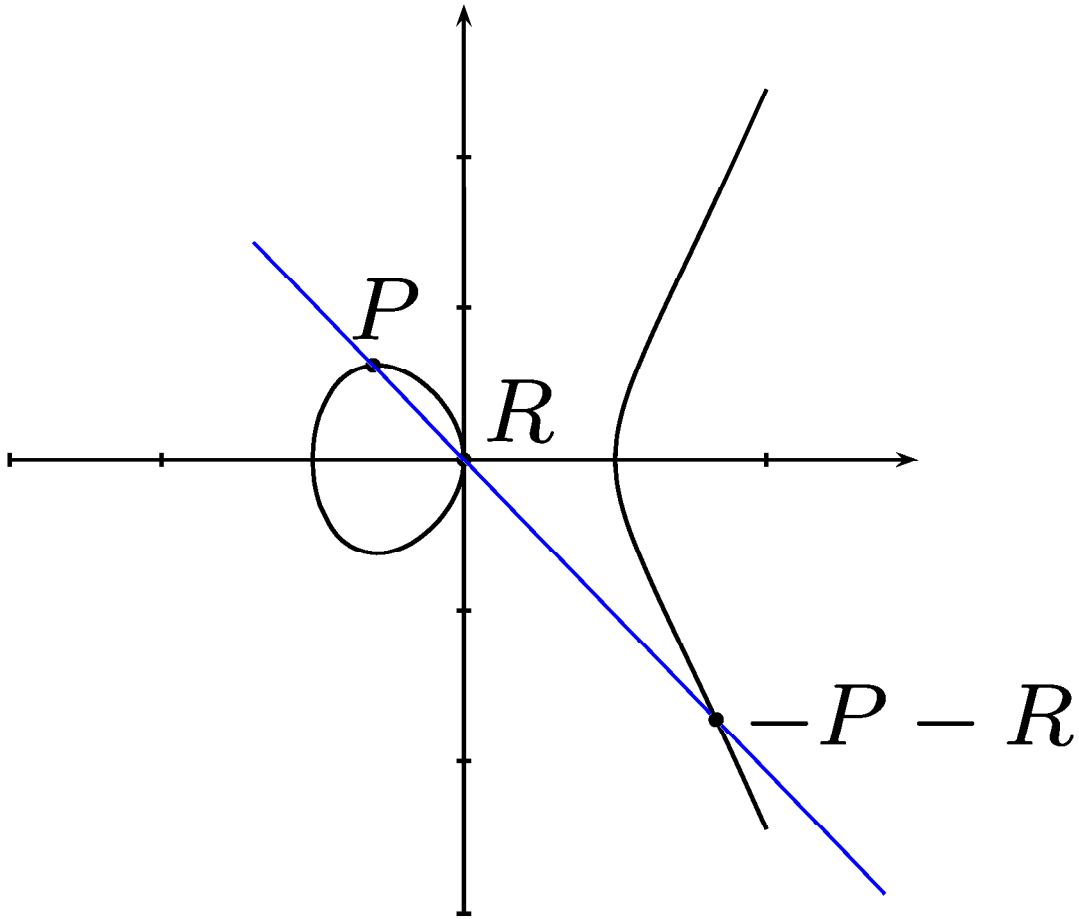
- Smaller key sizes due to exponential discrete logarithm problem on EC

  (160 bit EC vs. 1024 bit RSA)

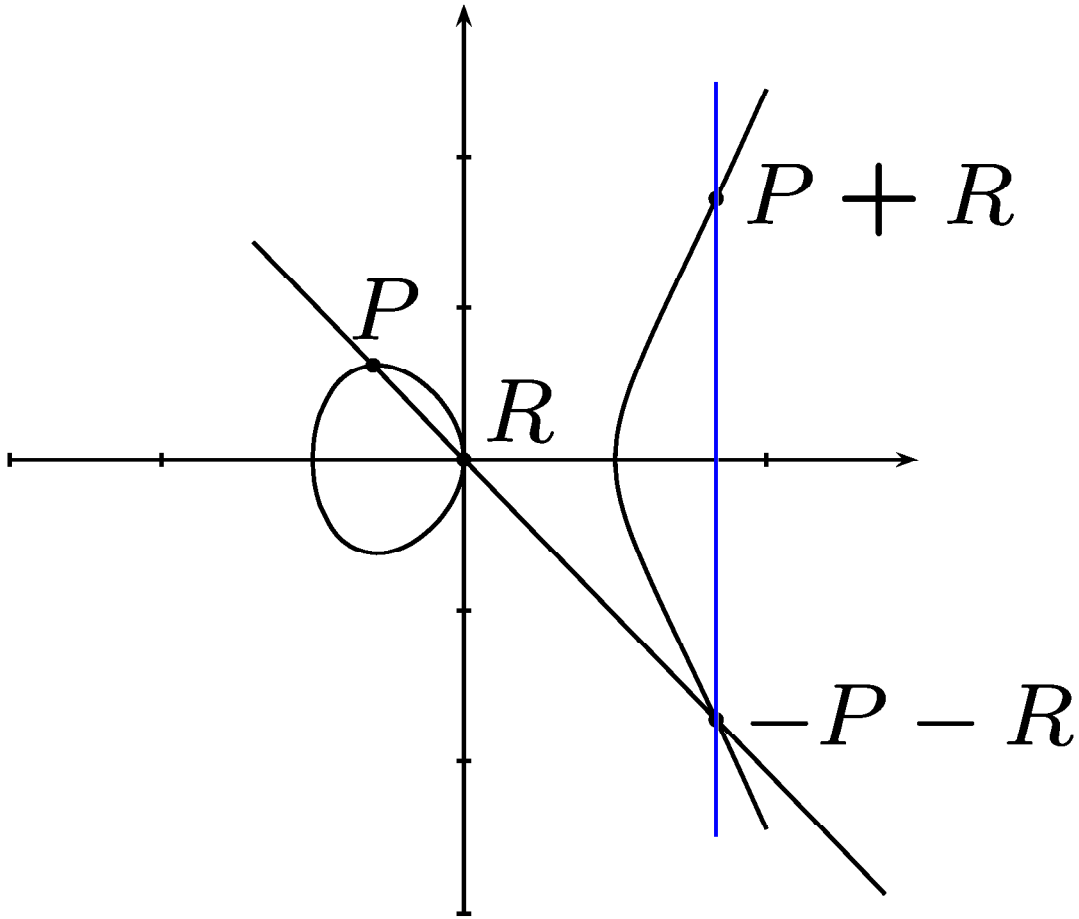
Group operation over $\mathbb{R}$



Group operation over $\mathbb{R}$



Group operation over $\mathbb{R}$



Explicit formulae

- Binary fields, $\mathbb{F}_{2^n}$

Addition	Doubling
$\lambda = \dfrac{y_1 + y_2}{x_1 + x_2}$	$\lambda = \dfrac{y_1}{x_1} + x_1$
$x' = \lambda^2 + \lambda + x_1 + x_2 + f_2$	$x' = \lambda^2 + \lambda + f_2$
$y' = (x_1 + x')\lambda + x' + y_1$	$y' = (x_1 + x')\lambda + x' + y_1$

1 inversion, 2 multiplications, 1 squaring

Hyperelliptic Curves

- Generalisation of Elliptic Curves

$$C: \quad y^2 + h(x)y = f(x) \in \mathbb{F}_q[x, y]$$

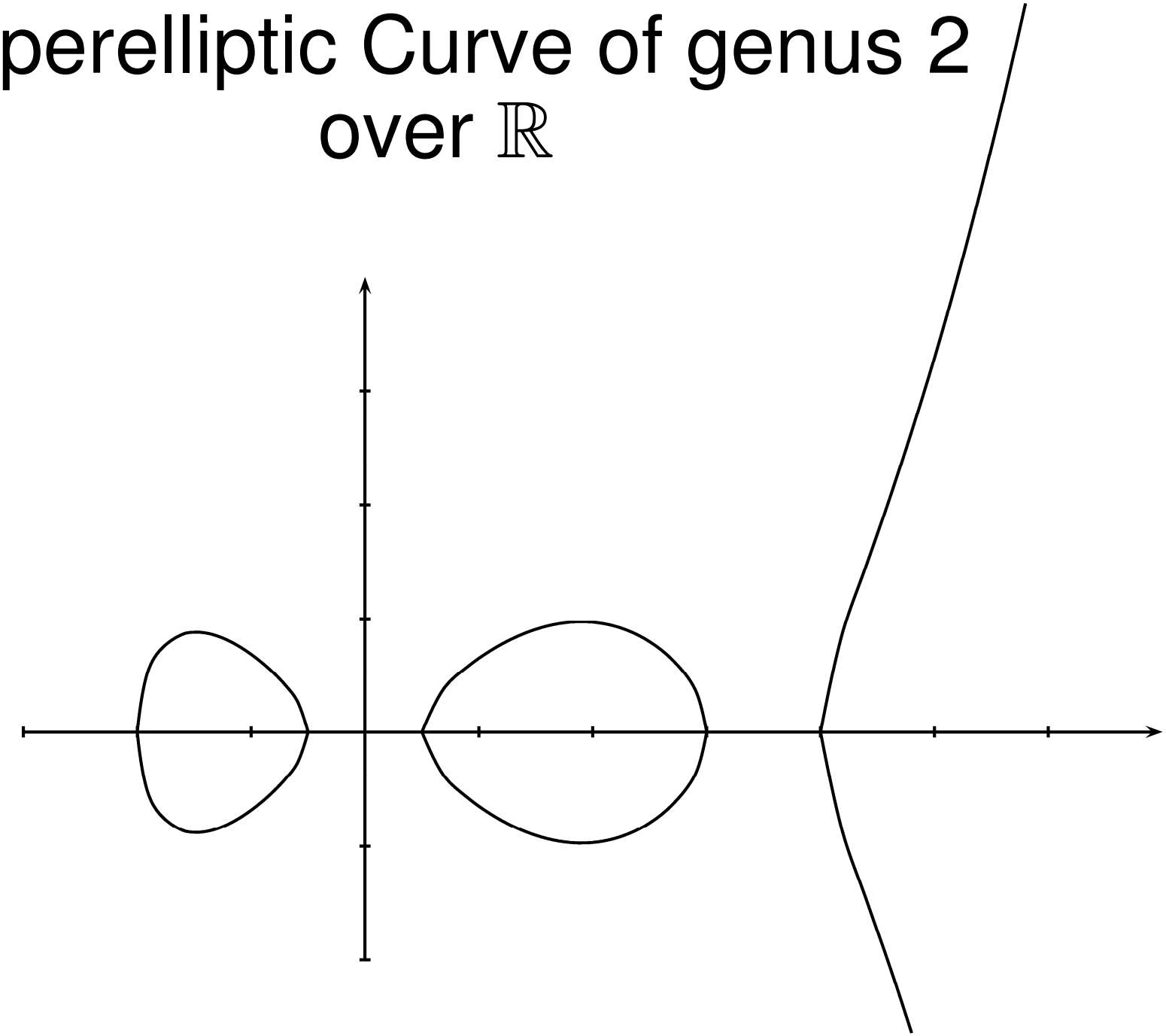
$$h(x), f(x) \in \mathbb{F}_q[x], \ f \ \text{monic}$$

$$\deg h \leq g, \ \deg f = 2g + 1$$

For which no *(x,y)* $\in$ *C* satisfies
both partial derivative equations

- *g* is called genus
- Elliptic Curves are HEC of genus 1

Hyperelliptic Curve of genus 2 over $\mathbb{R}$



Hyperelliptic Curves

- Points on the curve *C*

$$\mathcal{P} = \{(x, y) \in \overline{\mathbb{F}_q^2} : y^2 + h(x)y = f(x)\} \cup \{\infty\}$$

  do NOT form a group for genus $g > 1$

- Instead use Divisors:

  i.e. a finite formal sum of points with multiplicity

$$D = \sum_{i \in I} n_i P_i, \qquad P_i \in \mathcal{P}, \; n_i \in \mathbb{Z}$$

Divisors

$$D = \sum_{i \in I} n_i P_i$$

- Degree of *D* is $\deg D = \sum_{i \in I} n_i$
- $Div^0{}_C$ is the group of degree zero divisors

$$Div_C^0 = \{D \ : \ \deg D = 0\}$$

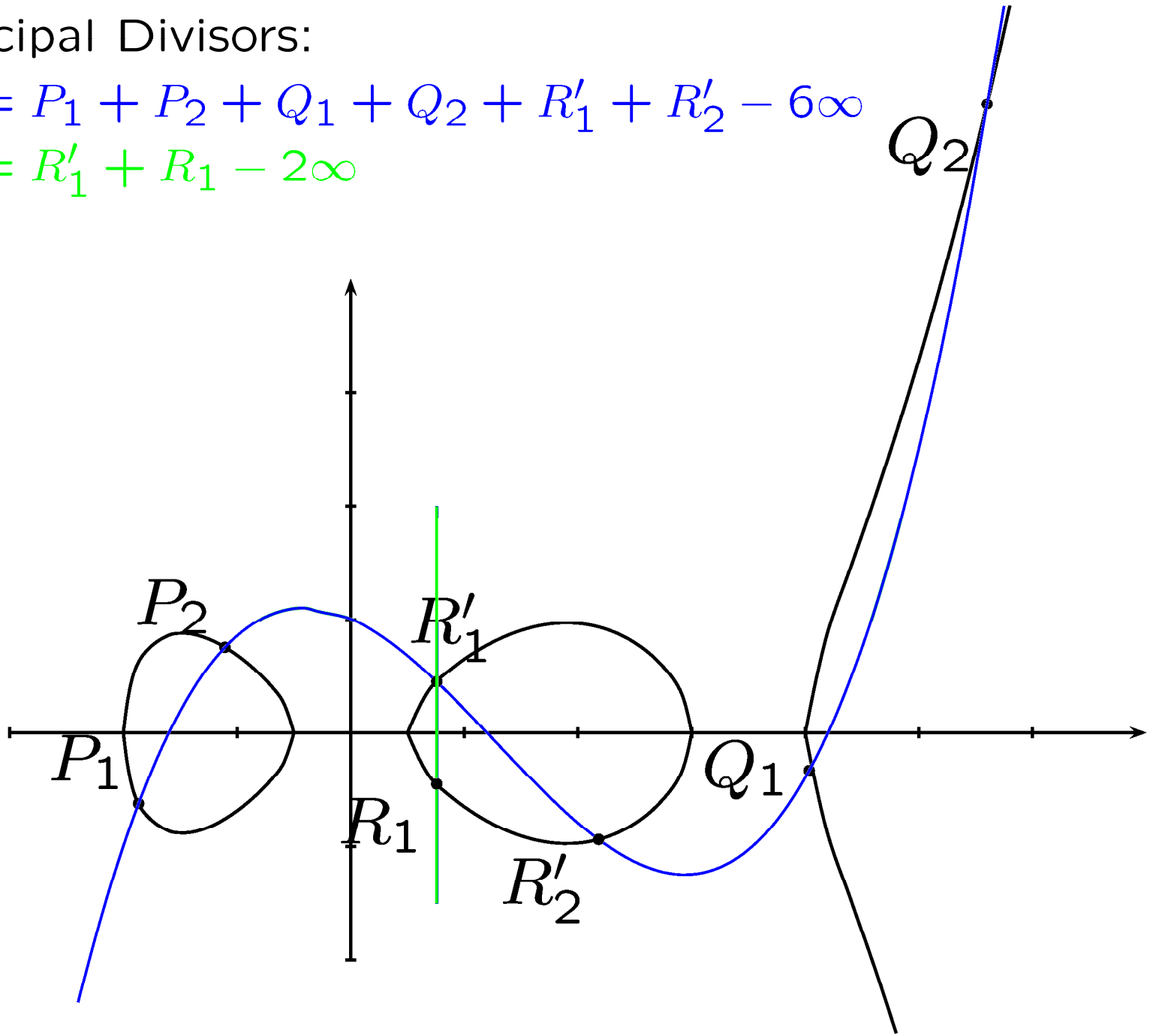
- *Princ* is the group of principal divisors
  - Divisors associated with functions

    Sum of intersection points of a function and curve
  - Subgroup of $Div^0{}_C$

Principal Divisors:

$D_1 = P_1 + P_2 + Q_1 + Q_2 + R_1' + R_2' - 6\infty$

$D_2 = R_1' + R_1 - 2\infty$



Hyperelliptic curves

- Divisors defined over $\mathbb{F}_q$:

  $$Div_C^0(\mathbb{F}_q) = \{D \in Div_C^0 \,:\, \sigma(D) = D\}$$

  σ Frobenius

- Cryptographic group:

  Degree zero divisors modulo principal divisors

  $$Pic_C^0(\mathbb{F}_q) = Div_C^0(\mathbb{F}_q)/Princ_C$$

  Group order is about $|\mathbb{F}_q|^g$

Hyperelliptic curves

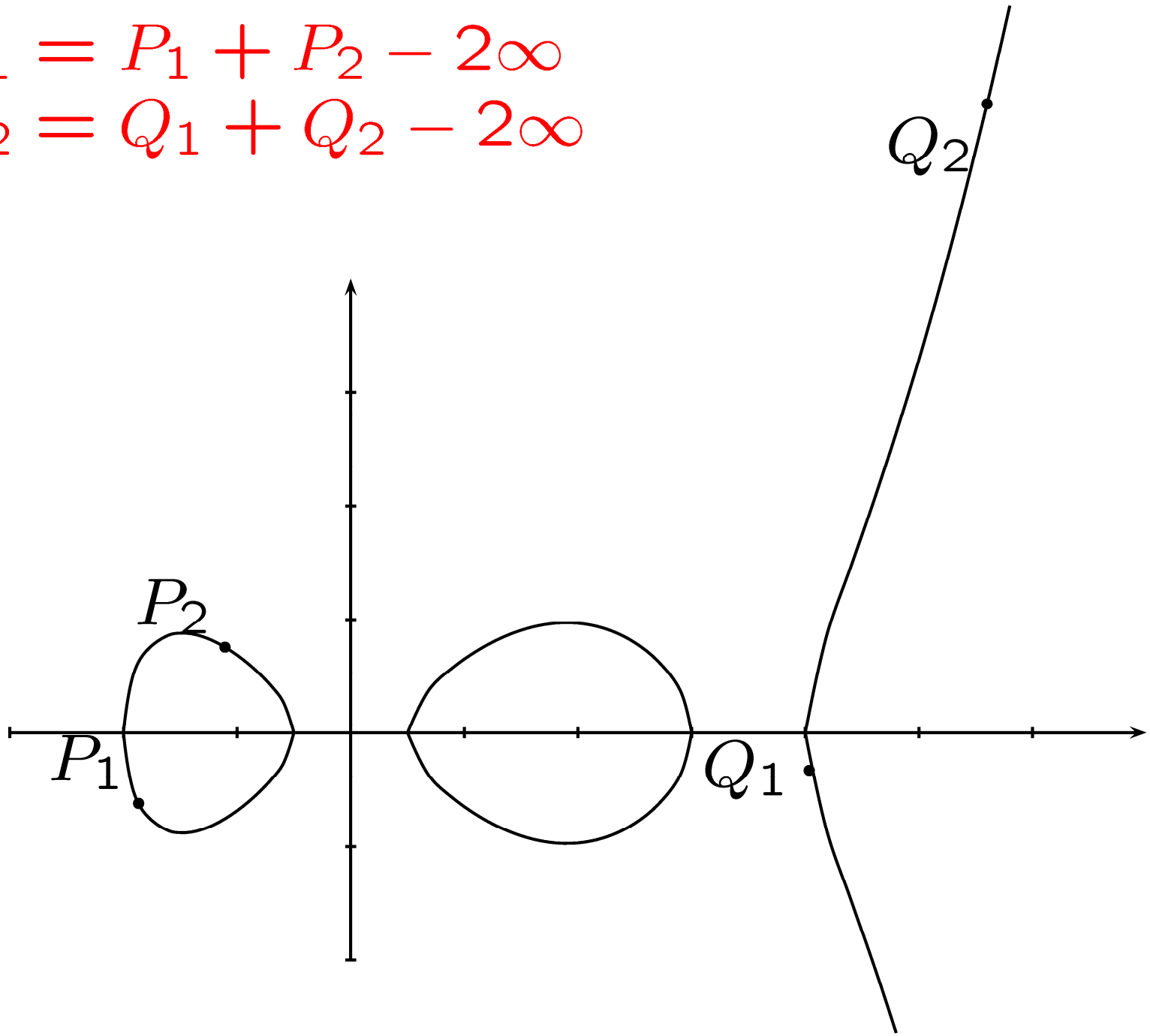
- Semi-reduced divisor

$$D = \sum_{i=1}^{r} P_i - r\infty$$
$$P_i \neq -P_j, \quad i \neq j$$

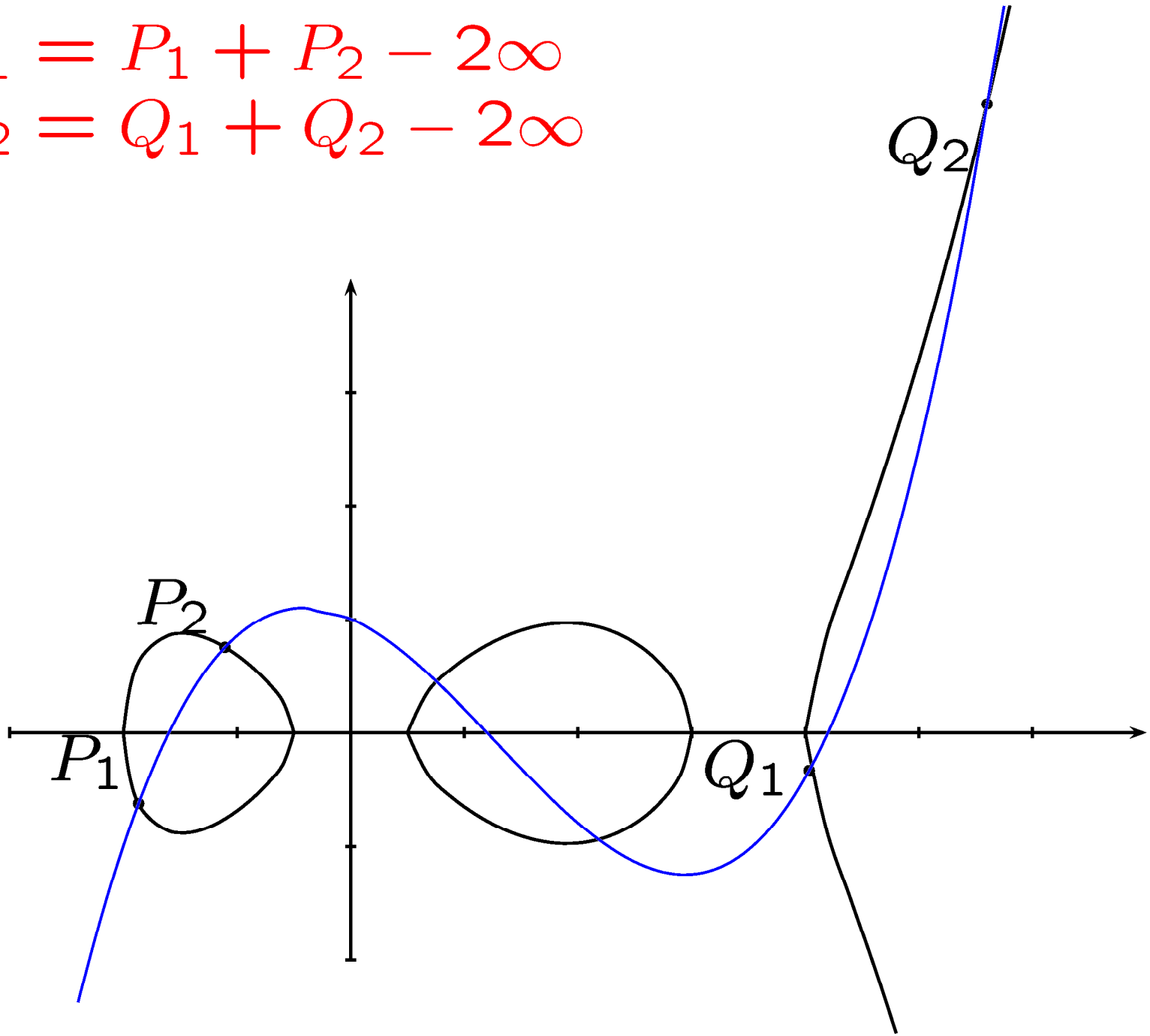
- Divisor class has unique representative

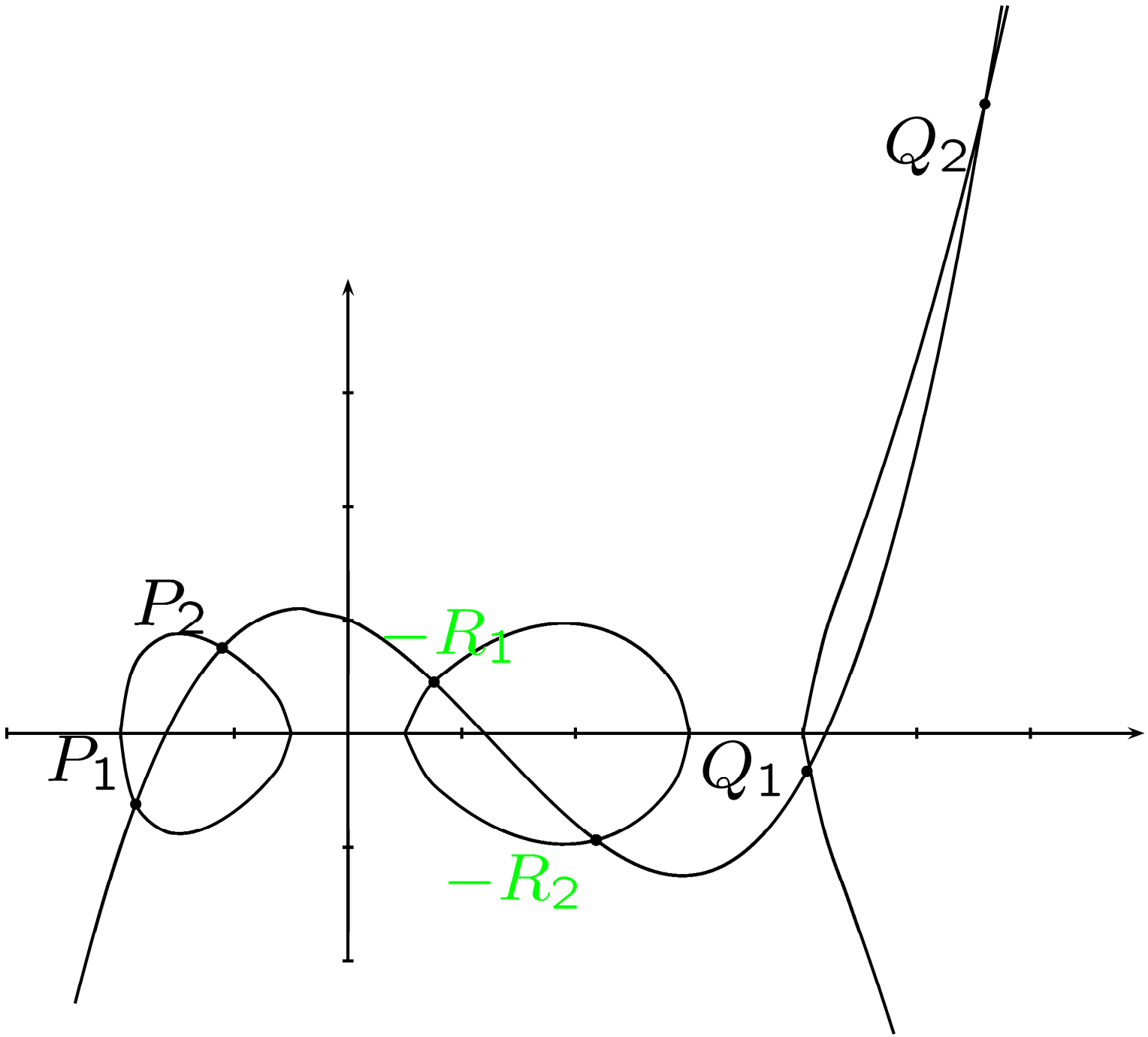
$$D = \sum_{i=1}^{r} P_i - r\infty, \; r \leq g$$
$$P_i \neq -P_j, \quad i \neq j$$

$$D_1 = P_1 + P_2 - 2\infty$$
$$D_2 = Q_1 + Q_2 - 2\infty$$

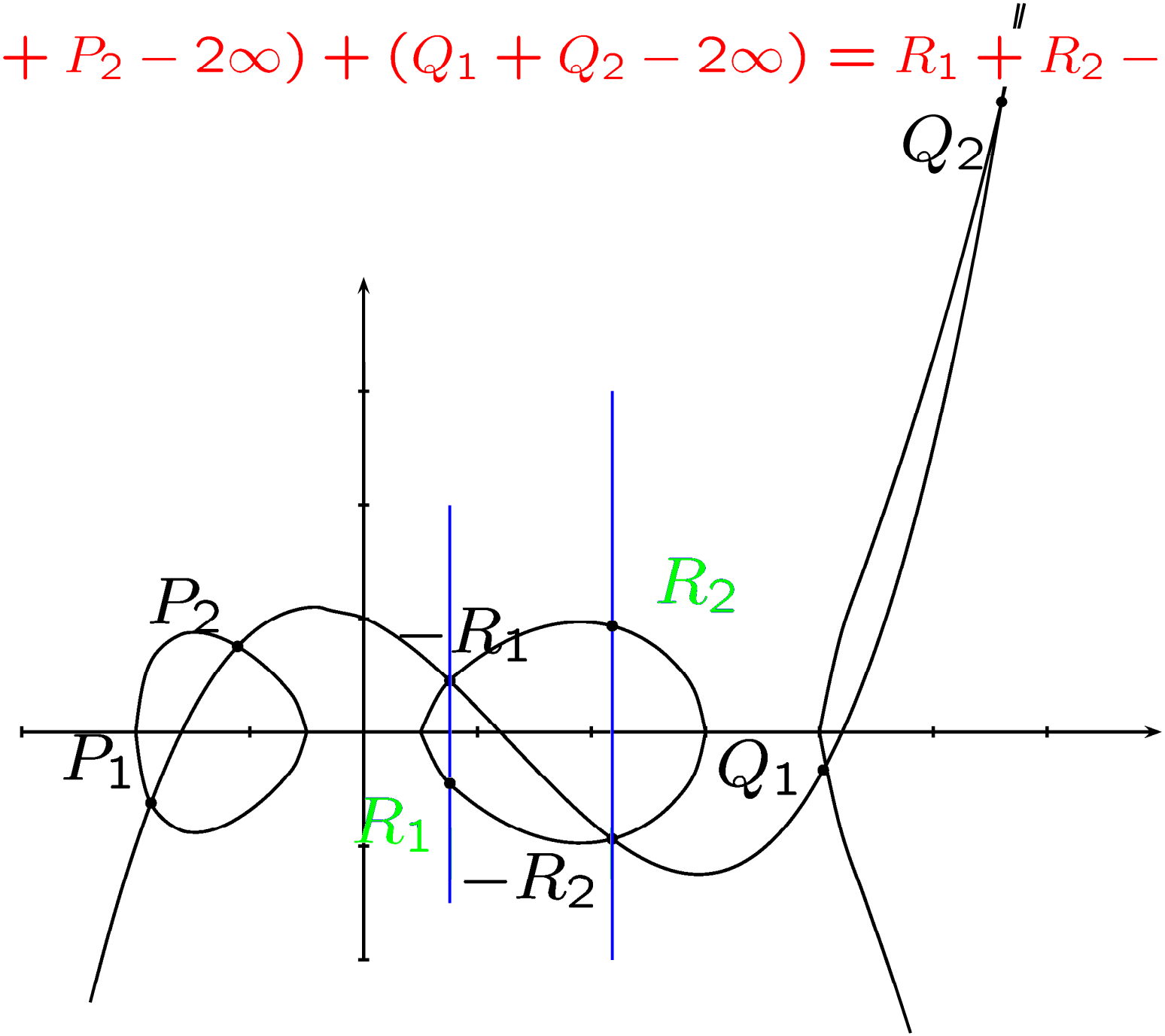


$$D_1 = P_1 + P_2 - 2\infty$$
$$D_2 = Q_1 + Q_2 - 2\infty$$




Q_2
P_2
$-R_1$
P_1
Q_1
$-R_2$

$$(P_1 + P_2 - 2\infty) + (Q_1 + Q_2 - 2\infty) = R_1 + R_2 - 2\infty$$



Efficient Arithmetic

- Representation of the Divisor class:
  Mumford representation

$$[u, v], \qquad u, v \in \mathbb{F}_q[x]$$

  – u is monic
  – deg v < deg $u \leq g$=2
  – $u \mid v^2 + vh - f$

Connection

$D = \sum n_i P_i - r\infty$
with $P_i = (x_i, y_i)$

then $u = \prod (x - x_i)^{n_i}$
$v(x_i) = y_i$ with multiplicity

Efficient Arithmetic

- Cantor's algorithm

Input $D_1 = [u_1, v_1],\ D_2 = [u_2, v_2]$

Step 1. Composition

Step 2. Reduction

Cantor: 1. Composition

Output $D = [u, v]$ semi-reduced with $D = D_1 + D_2$

1. compute $d_1 = \gcd(u_1, u_2) = e_1 u_1 + e_2 u_2$
2. compute $d = \gcd(d_1, v_1 + v_2 + h) = c_1 d_1 + c_2(v_1 + v_2 + h)$;
3. let $s_1 = c_1 e_1,\ s_2 = c_1 e_2,\ s_3 = c_2$;
4. $u = \frac{u_1 u_2}{d^2}$;
   $v = \frac{s_1 u_1 v_2 + s_2 u_2 v_1 + s_3(v_1 v_2 + f)}{d} \bmod u$.

Cantor: 2. Reduction

Output $D' = [u', v']$ reduced with $D' \equiv D$

1. let $u' = \frac{f - vh - v^2}{u}$, $v' = (-h - v) \bmod u'$;
2. if $\deg u' > g$ put $u = u', v = v'$;
   goto step 1;
3. make u' monic.

Genus 2 Explicit Formulae

- Focus on binary fields $\mathbb{F}_{2^n}$
- Explicit formulae avoid unnecessary calculations
- Addition more complex than Elliptic Curves:
  - 1 inversion, 22 multiplications, 3 squarings
  - EC: 1 inversion, 2 multiplications, 1 squaring
- Same security, half field size
  - 80 bit vs. EC 160 bit

G2 addition & doubling

- Explicit formulae for addition and doubling by Tanja Lange
- Most common case for doubling:
  - deg $u=2$
  - res$(h, u) \neq 0$
- Doubling general: 1 inv, 22 mul, 5 sqr
- Our improvements using $h_0=0$
  - At worst 1 inv, 17 mul, 5 sqr
  - At best when $h_2=0$: 1 inv, 5 mul, 6 sqr

Explicit formulae

- Break down of steps in Cantor's algorithm:

$$
\begin{aligned}
\tilde{v} &\equiv h \mod u \\
r &= \mathrm{res}(\tilde{v}, u) \\
inv' &\equiv r\tilde{v}^{-1} \mod u \equiv \tilde{v}_1 x + \tilde{v}_0 + u_1 \tilde{v}_1 \mod u \\
k &= (f + hv + v^2)/u = k' + u(x + f_4), \text{ with } k' \equiv k \mod u \\
s' &\equiv k' \; inv' \mod u \\
s'' &= s' \text{ made monic} \\
l'' &= s''u \\
u' &= s''^2 + (kr^2/{s'_1}^2 + hs''r/s'_1)/u \\
v' &\equiv h + (l''s'_1/r + v) \mod u'
\end{aligned}
$$

Use Montgomery's trick, Karatsuba, …

G2 doubling, general

Doubling, $\deg u = 2$			
Input	$[u, v], u = x^2 + u_1 x + u_0, v = v_1 x + v_0$		
Output	$[u', v'] = 2[u, v]$		
Step	Expression	odd	even
1	compute $\tilde{v} \equiv (h + 2v) \bmod u = \tilde{v}_1 x + \tilde{v}_0$: $\tilde{v}_1 = h_1 + 2v_1 - h_2 u_1$, $\tilde{v}_0 = h_0 + 2v_0 - h_2 u_0$;		
2	compute resultant $r = \mathrm{res}(\tilde{v}, u)$: $w_0 = v_1^2$, $w_1 = u_1^2$, $w_2 = \tilde{v}_1^2$, $w_3 = u_1 \tilde{v}_1$, $r = u_0 w_2 + \tilde{v}_0(\tilde{v}_0 - w_3)$;	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 $k' = (f - hv - v^2)/u \bmod u = k'_1 x + k'_0$: $w_3 = f_3 + w_1$, $w_4 = 2u_0$, $k'_1 = 2(w_1 - f_4 u_1) + w_3 - w_4 - h_2 v_1$; $k'_0 = u_1(2w_4 - w_3 + f_4 u_1 + h_2 v_1) + f_2 - w_0 - 2f_4 u_0 - h_1 v_1 - h_2 v_0$;	1M	2M (see below)
5	compute $s' = k' inv' \bmod u$: $w_0 = k'_0 inv'_0$, $w_1 = k'_1 inv'_1$; $s'_1 = (inv'_0 + inv'_1)(k'_0 + k'_1) - w_0 - w_1(1 + u_1)$, $s'_0 = w_0 - u_0 w_1$; If $s_1 = 0$ see below	5M	5M
6	compute $s'' = x + s_0/s_1$ and s_1: $w_1 = 1/(rs'_1)(= 1/r^2 s_1)$, $w_2 = rw_1(= 1/s'_1)$, $w_3 = s'^2_1 w_1(= s_1)$; $w_4 = rw_2(= 1/s_1)$, $w_5 = w_4^2$, $s''_0 = s'_0 w_2$;	I, 2S, 5M	I, 2S, 5M
7	compute $l' = s''u = x^3 + l'_2 x^2 + l'_1 x + l'_0$: $l'_2 = u_1 + s''_0$, $l'_1 = u_1 s''_0 + u_0$, $l'_0 = u_0 s''_0$;	2M	2M
8	compute $u' = s^2 + (h + 2v)s/u + (v^2 + hv - f)/u^2$: $u'_0 = s''^2_0 + w_4(h_2(s''_0 - u_1) + 2v_1 + h_1) + w_5(2u_1 - f_4)$; $u'_1 = 2s''_0 + h_2 w_4 - w_5$;	S, 2M	S, M
9	compute $v' \equiv -h - (l + v) \bmod u' = v'_1 x + v'_0$: $w_1 = l'_2 - u'_1$, $w_2 = u'_1 w_1 + u'_0 - l'_1$, $v'_1 = w_2 w_3 - v_1 - h_1 + h_2 u'_1$; $w_2 = u'_0 w_1 - l'_0$, $v'_0 = w_2 w_3 - v_0 - h_0 + h_2 u'_0$;	4M	4M
total		I, 5S, 22 M	I, 5S, 22 M
Special case $s = s_0$			
$6'$	compute s and precomputations: $w_1 = 1/r$, $s_0 = s'_0 w_1$, $w_2 = u_0 s_0 + v_0 + h_0$;	I,2M	I,2M
$7'$	compute $u' = (f - hv - v^2)/u^2 - (h + 2v)s/u - s^2$: $u'_0 = f_4 - s_0^2 - s_0 h_2 - 2u_1$;	S	S
$8'$	compute $v' \equiv -h - (su + v) \bmod u'$: $w_1 = s_0(u_1 - u'_0) - h_2^2 u'_0 + v_1 + h_1$, $v'_0 = u'_0 w_1 - w_2$;	2M	2M
total		I, 3S, 13M	I, 3S, 14M

G2 doubling, deg h=1

- Case $C: \; y^2 + h_1 xy = x^5 + f_3 x^3 + f_2 x^2 + f_0$
- Curve transformation
  - $h_0 = f_4 = f_1 = 0$
  - $(1/h_1)$ 'small'
- Formulas depend on h_1, h_1^2, h_1^{-1}, f_3, f_2, f_0
  - Case $h_1=1$: 6 sqr, 5 mul, 1 inv
  - Case $1/h_1$ 'small': 5 sqr, 7 mul, 1 inv
  - Case h_1 arbitrary: 5 sqr, 9 mul, 1 inv

G2 doubling, deg *h*=1

Doubling $\deg h = 1$, $\deg u = 2$				
Input	$[u, v], u = x^2 + u_1 x + u_0, v = v_1 x + v_0$; h_1^2, h_1^{-1}			
Output	$[u', v'] = 2[u, v]$			
Step	Expression	$h_1 = 1$	h_1^{-1} small	h_1 arbitrary
1	compute rs_1: $z_0 = u_0^2$, $z_1 = u_1^2$, $z_2 = v_0^2$, $k_1' = z_1 + f_3$; $w_0 = f_0 + z_2 (= rs_1'/h_1^3)$; If $w_0 = 0$ see below	3S	3S	3S
2	compute $1/s_1$ and s_0'': $w_1 = (1/w_0) z_0 (= h_1/s_1)$; $s_0'' = k_1' w_1 + u_1$, $z_3 = s_0''^2$;	I, S, 2M	I, S, 2M	I, S, 2M
3	compute u': $w_2 = h_1^2 w_1$, $u_1' = w_2 w_1$; $u_0' = z_3 + w_2$;	S	2M	2M
4	compute v': $w_3 = w_2 + k_1'$, $w_4 = s_0'' + u_1$; $v_1' = h_1^{-1}(w_3 w_4 + w_2 u_1' + f_2 + v_1^2)$; $v_0' = h_1^{-1}(w_3 u_0' + f_1 + z_0)$;	S, 3M	S, 3M	S, 5M
total		I, 6S, 5M	I, 5S, 7M	I, 5S, 9M
Special case $s = s_0$				
$2'$	compute s and precomputations: $s_0 = (1/h_1) k_1'$, $w_1 = u_0 s_0 + v_0$;	1M	1M	2M
$3'$	compute u': $u_0' = s_0^2$;	S	S	S
$4'$	compute v': $w_2 = s_0(u_1 + u_0') + v_1 + h_1$; $v_0' = u_0' w_2 + w_1$;	2M	2M	2M
total		4S, 3M	4S, 3M	4S, 4M

G2 doubling, deg h=2

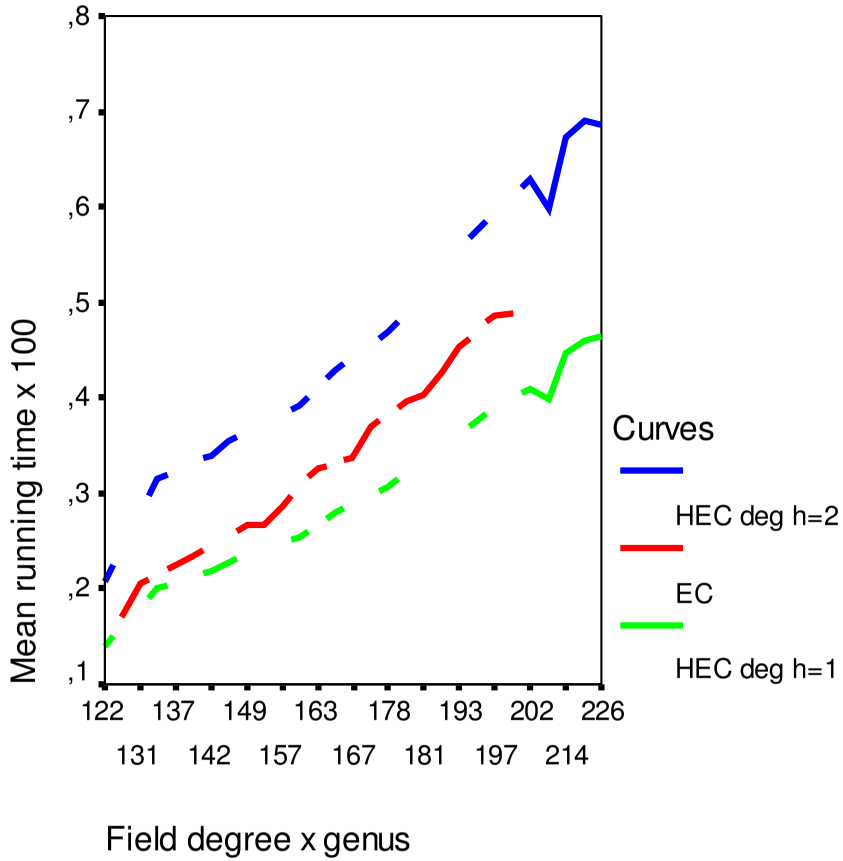
- Case deg h=2
- Curve transformation
  - $h_2 = 1$, $f_3 = f_2 = 0$
  - $h_0 = 0$ only if $h_1=0$ or $Tr(h_0/h_1^2)=0$
- Formulas depend on h_2, h_1, h_1^2, f_4
  - Case h_1 'small': 1 inv, 12 mul, 6 sqr
  - Case h_1 arbitrary: 1 inv, 17 mul, 5 sqr
  - If f_4 'small' then 2 mul cheap or for free

G2 doubling, deg h=2

Doubling, $\deg h = 2$, $h_0 = 0$, $\deg u = 2$			
Input	$[u, v], u = x^2 + u_1 x + u_0, v = v_1 x + v_0, h_1^2$		
Output	$[u', v'] = 2[u, v]$		
Step	Expression	h_1 small	h_1 arbitrary
1	compute k_1' and precomputations: $z_0 = u_0^2$, $z_1 = u_1^2$, $w_0 = v_1(h_1 + v_1)$; $k_1' = z_1 + v_1$, $z_2 = h_1 u_1$, $z_3 = f_4 u_1$;	(M,) 3S	(M,) 2S, 2M
2	compute resultant $r = \mathrm{res}(\tilde{v}, u)$: $\tilde{r} = u_0 + h_1^2 + z_2 = (r/u_0)$;		
3	compute s_1' and almost s_0': $w_2 = u_1(k_1' + z_3) + w_0$, $w_3 = v_0 + h_1 k_1'$; $s_1' = f_1 + z_0 + h_1 w_2$; $m_0 = w_2 + w_3 (= (s_0' - u_1 s_1')/u_0)$; If $s_1' = 0$ see below	M	3M
4	compute $s'' = x + s_0/s_1$ and s_1: $w_2 = 1/(s_1')(= 1/r s_1)$, $w_3 = u_0 w_2$; $w_4 = \tilde{r} w_3 (= 1/s_1)$, $w_5 = w_4^2$, $s_0'' = u_1 + m_0 w_3$;	I, S, 3M	I, S, 3M
5	compute $u' = s^2 + (h + 2v)s/u + (v^2 + hv - f)/u^2$: $z_4 = f_4 w_4$, $u_1' = w_4 + w_5$; $u_0' = s_0''^2 + w_4(s_0'' + h_1 + u_1 + z_4)$;	(M,) S, M	(M,) S, M
6	compute $v' \equiv -h - (l + v) \bmod u' = v_1' x + v_0'$: $z_5 = w_2(m_0^2 + k_1'(s_1' + h_1 m_0))$, $z_6 = s_0'' + h_1 + z_4 + z_5$; $v_0' = v_0 + z_2 + z_1 + w_4(u_0' + z_3) + s_0'' z_6$; $v_1' = v_1 + w_4(u_1' + s_0'' + f_4 + u_1) + z_5$;	S, 5M	S, 6M
total		(2M,) I, 6S, 10 M	(2M,) I, 5S, 15M
Special case $s = s_0$			
$3'$	compute s and precomputations: $w_1 = 1/\tilde{r}$, $s_0 = m_0 w_1$, $w_2 = u_0 s_0 + v_0 + h_0$;	I, 2M	I, 2M
$4'$	compute $u' = (f - hv - v^2)/u^2 - (h + 2v)s/u - s^2$: $u_0' = s_0^2 + s_0$;	S	S
$5'$	compute $v' \equiv -h - (su + v) \bmod u'$: $w_1 = s_0(u_1 + u_0') + u_0' + v_1 + h_1$, $v_0' = u_0' w_1 + w_2$;	2M	2M
total		(M,) I, 4S, 5M	(M,) I, 3S, 9M

Timings

- m-Fold timings using a sliding window method of size 3 (precomputes ±D, ±3D)
- Based on NTL library
- Timed on a AMD Athlon XP2500+
- Curves defined over F_2



The end