

Curve Arithmetic

Nigel Smart

nigel@cs.bris.ac.uk

January 27, 2009

Outline

Curve Arithmetic

Point Multiplication

Curve Arithmetic

Curve Arithmetic

Points can be added/doubled in a number of formats:

Affine:

- ▶ $P = (X, Y)$.

Projective (Standard):

- ▶ $P = (X, Y) = (x/z, y/z)$.

Projective (Jacobian):

- ▶ $P = (X, Y) = (x/z^2, y/z^3) = (x, y, z)$.

Chudnovsky:

- ▶ As Jacobian but store $P = (x, y, z, z^2, z^3)$.

Lopez-Dahab:

- ▶ $P = (X, Y) = (x/z, y/z^2) = (x, y, z)$

Mixed:

- ▶ A combination of any of the above.

Curve Arithmetic

Standard projective coordinates are rarely used.

We will concentrate on

- ▶ Affine,
- ▶ Jacobean Projective/Lopez-Dahab
- ▶ A mixture of the two.

Main problem with Affine is that we require field inversions.

- ▶ Field inversions are generally much slower than multiplications.

Projective coordinates allow us to trade a number of multiplications for an inversion.

We shall see later that its is point doubling which is most important.

Cost of Curve Arithmetic: Char p

In odd characteristic it is often best to use **Jacobian Projective** coordinates:

Operation	Affine	Projective	Mixed
Addition	$3M + 1I$	$16M$	$11M$
Doubling	$4M + 1I$	$10M$	n/a

In next few slides we give the formulae for computing

$$\blacktriangleright P_3 = P_1 + P_2$$

with

$$\blacktriangleright P_i = (X_i, Y_i, Z_i)$$

when the curve is given by

$$Y^2 = X^3 + AX + B.$$

Projective Addition Formulae: Char p

$$\begin{array}{rcl} \lambda_1 & = & X_1 Z_2^2 & 2M \\ \lambda_2 & = & X_2 Z_1^2 & 2M \\ \lambda_3 & = & \lambda_1 - \lambda_2 & \\ \lambda_4 & = & Y_1 Z_2^3 & 2M \\ \lambda_5 & = & Y_2 Z_1^3 & 2M \\ \lambda_6 & = & \lambda_4 - \lambda_5 & \\ \lambda_7 & = & \lambda_1 + \lambda_2 & \\ \lambda_8 & = & \lambda_4 + \lambda_5 & \\ Z_3 & = & Z_1 Z_2 \lambda_3 & 2M \\ X_3 & = & \lambda_6^2 - \lambda_7 \lambda_3^2 & 3M \\ \lambda_9 & = & \lambda_7 \lambda_3^2 - 2X_3 & \\ Y_3 & = & (\lambda_9 \lambda_6 - \lambda_8 \lambda_3^3)/2 & 3M \\ & & & \hline & & & 16M \end{array}$$

Projective Addition Formulae: Char p

In previous slide

$\lambda_3 = 0$ if and only if $P_1 = \pm P_2$

$\lambda_3 = \lambda_6 = 0$ if and only if $P_1 = P_2$

- ▶ In this case need to execute a point doubling (see later)

A mixed addition is when $Z_1 = 1$ in which case we simplify the formulae as in the next slide

Projective (Mixed) Addition Formulae: Char p

$$\begin{array}{rcl} \lambda_1 & = & X_1 Z_2^2 \qquad 2M \\ \lambda_3 & = & \lambda_1 - X_2 \\ \lambda_4 & = & Y_1 Z_2^3 \qquad 2M \\ \lambda_6 & = & \lambda_4 - Y_2 \\ \lambda_7 & = & \lambda_1 + X_2 \\ \lambda_8 & = & \lambda_4 + Y_2 \\ Z_3 & = & Z_2 \lambda_3 \qquad 1M \\ X_3 & = & \lambda_6^2 - \lambda_7 \lambda_3^2 \qquad 3M \\ \lambda_9 & = & \lambda_7 \lambda_3^2 - 2X_3 \\ Y_3 & = & (\lambda_9 \lambda_6 - \lambda_8 \lambda_3^3)/2 \qquad 3M \\ & & \hline & & 11M \end{array}$$

Projective Doubling Addition Formulae: Char p

A point doubling is performed via

$$\begin{array}{rcl} \lambda_1 & = & 3X_1^2 + AZ_1^4 & 4M \\ Z_3 & = & 2Y_1Z_1 & 1M \\ \lambda_2 & = & 4X_1Y_1^2 & 2M \\ X_3 & = & \lambda_1^2 - 2\lambda_2 & 1M \\ \lambda_3 & = & 8Y_1^4 & 1M \\ Y_3 & = & \lambda_1(\lambda_2 - X_3) - \lambda_3 & 1M \\ & & & \hline & & & 10M \end{array}$$

Projective Doubling Addition Formulae: Char p

Recall we said usually $A = -3$ then

$$\begin{array}{rcl} \lambda_1 & = & 3X_1^2 + AZ_1^4 = 3(X_1 + Z_1^2)(X_1 - Z_1^2) & 2M \\ Z_3 & = & 2Y_1Z_1 & 1M \\ \lambda_2 & = & 4X_1Y_1^2 & 2M \\ X_3 & = & \lambda_1^2 - 2\lambda_2 & 1M \\ \lambda_3 & = & 8Y_1^4 & 1M \\ Y_3 & = & \lambda_1(\lambda_2 - X_3) - \lambda_3 & 1M \\ & & & \hline & & & 8M \end{array}$$

Cost of Curve Arithmetic: Char 2

In even characteristic it is often best to use [Lopez-Dahab](#) coordinates:

Operation	Affine	Lopez-Daheb	Mixed
Addition	$2M + 1l$	13 M	8 M
Doubling	$2M + 1l$	4 M	n/a

Note squaring is free in even characteristic.

In next few slides we give the formulae for computing

$$\blacktriangleright P_3 = P_1 + P_2$$

with

$$\blacktriangleright P_i = (X_i, Y_i, Z_i)$$

when the curve is given by

$$Y^2 + XY = X^3 + AX^2 + B.$$

Projective Addition Formulae: Char 2

λ_1	$=$	$X_1 Z_2$	$1M$
λ_2	$=$	$X_2 Z_1$	$1M$
λ_3	$=$	$\lambda_1 + \lambda_2$	
λ_4	$=$	λ_1^2	free
λ_5	$=$	λ_2^2	free
λ_6	$=$	$\lambda_4 + \lambda_5$	
λ_7	$=$	$Y_1 Z_2^2$	$1M$
λ_8	$=$	$Y_2 Z_1^2$	$1M$
λ_9	$=$	$\lambda_7 + \lambda_8$	
λ_{10}	$=$	$\lambda_3 \lambda_9$	$1M$
Z_3	$=$	$Z_1 Z_2 \lambda_6$	$2M$
X_3	$=$	$\lambda_1(\lambda_8 + \lambda_5) + \lambda_2(\lambda_7 + \lambda_4)$	$2M$
Y_3	$=$	$(\lambda_1 \lambda_{10} + \lambda_7 \lambda_6) \lambda_6 + (\lambda_{10} + Z_3) X_3$	$4M$
			<hr/> $13M$

Projective (Mixed) Addition Formulae: Char 2

$$\begin{array}{rcl} \lambda_1 & = & Z_2 Y_1 + Y_2 & 1M \\ \lambda_2 & = & Z_2 X_1 + X_2 & 1M \\ \lambda_3 & = & Z_2 \lambda_2 & 1M \\ Z_3 & = & \lambda_3^2 & \text{free} \\ \lambda_5 & = & Z_3 X_1 & 1M \\ \lambda_6 & = & X_1 + Y_1 & \\ X_3 & = & \lambda_1^2 + \lambda_3(\lambda_1 + \lambda_2^2 + A\lambda_3) & 2M \\ Y_3 & = & (\lambda_5 + X_3)(\lambda_3 \lambda_1 + Z_3) + \lambda_6 Z_3^2 & \frac{3M}{9M} \end{array}$$

If $A = 1$ then this reduces to $8M$.

Projective Doubling Addition Formulae: Char 2

A point doubling is performed via

$$\begin{array}{rcl} \lambda_1 & = & X_1^2 \qquad \text{free} \\ \lambda_2 & = & \lambda_1 + Y_1 \\ \lambda_3 & = & X_1 Z_1 \qquad 1M \\ Z_3 & = & \lambda_3^2 \qquad \text{free} \\ \lambda_5 & = & \lambda_2 Z_3 \qquad 1M \\ X_3 & = & \lambda_2^2 + \lambda_3 + AZ_3 \qquad 1M \\ Y_3 & = & (Z_3 + \lambda_5)X_3 + \lambda_1^2 Z_3 \qquad 2M \\ & & \hline & & 5M \end{array}$$

Which reduces to $4M$ if we choose $A = 1$.

Projective Formulae Summary

Odd Characteristic

Operation	Affine	Projective	Mixed
Addition	$3M + 1l$	16M	11M
Doubling	$4M + 1l$	10M	n/a

Even Characteristic

Operation	Affine	Lopez-Daheb	Mixed
Addition	$2M + 1l$	13 M	8 M
Doubling	$2M + 1l$	4 M	n/a

Doubling is **very** fast in even characteristic.

- ▶ This can make up for the slow software field arithmetic in a final implementation.

Point Multiplication

Point Multiplication

The basic cryptographic operation is to compute

$$Q = [d]P$$

for some integer d and point P .

The binary method:

- ▶ $Q = 0$.
- ▶ For $j = \log_2(d) - 1$ to 0
 - ▶ $Q = [2]Q$.
 - ▶ If $d_j = 1$ then $Q = Q + P$.

This requires

- ▶ Exactly $\log_2(d)$ point doublings.
- ▶ About $\log_2(d)/2$ general point additions.
 - ▶ If P affine and Q projective used mixed addition

We can reduce the number of general point additions.

Point Multiplication

m-ary method

The binary method uses a fixed window of size 2.

The *m*-ary method uses a fixed window of size $m = 2^r$.

- ▶ Taking *m* bits at a time.

Requires precomputation of

$$2^{r-1} - 1$$

general point additions.

Then requires

- ▶ $\log_2(d)$ doublings.
- ▶ About $\log_2(d)/r$ general point additions.

Point Multiplication

Sliding Window Method

This method slides the window of length m across runs of zero's in the binary expansion.

Requires precomputation of

$$2^{r-1} - 1$$

general point additions.

Then requires

- ▶ $\log_2(d)$ doublings.
- ▶ About $\log_2(d)/(r + 1)$ general point additions.

Point Multiplication

Signed Window Methods

For elliptic curves negation comes for free

$$-P = (x, -y) \text{ or } (x, y + x).$$

Hence we could use a signed binary representation.

$$7 = 2^2 + 2 + 1 \text{ or } 7 = 2^3 - 1.$$

This allows us to reduce the number of point additions even further

- ▶ In both the m -ary and the sliding window algorithms.

Signed Sliding Window Method

The following precomputation is done once for each point P ;

Precomputation

- ▶ $P_1 = P, P_2 = [2]P.$
- ▶ For $i = 1$ to $2^{r-2} - 1$
 - ▶ $P_{2i+1} = P_{2i-1} + P_{2i}.$
- ▶ $Q = P_{d_{l-1}}.$

These are computed in affine coordinates

Next we encode the number d Set

$$d = \sum_{i=0}^{l-1} d_i 2^{e_i}$$

with $e_{i+1} - e_i \geq r$ and

$$d_i \in \{\pm 1, \pm 3, \dots, \pm 2^{r-1} - 1\}.$$

Signed Sliding Window Method

Main Loop

- ▶ For $i = l - 2$ to 0
 - ▶ $Q = [2^{e_{i+1} - e_i}]Q$.
 - ▶ If $l_i > 0$ then $Q = Q + P_{d_i}$.
 - ▶ Else $Q = Q - P_{-d_i}$.
- ▶ $Q = [2^{e_0}]Q$.

The Q is held in projective coordinates

- ▶ Since P_i are affine can use mixed addition
- ▶ Can use efficient projective doubling formulae

Point Multiplication

The signed sliding window method generally comes out to be the fastest.

Remember

In all cases CPU time is dominated by the time to compute

$$\log_2(d)$$

point doublings.

Lesson

Optimise the doubling operation at all times.

- ▶ This is often ignored.

Point Multiplication

Notice that doubling in char 2 requires less multiplications than in char p .

This can often lead (depending on the processor or the implementation) that the relative advantage of char p field multiplication can be cancelled out by the doubling operation.

This will become even more pronounced when the new instruction set extensions to Intel and other chips come out which will provide native carry-free multipliers.

- ▶ i.e. char 2 field multiplications will run as fast as char p field multiplications
- ▶ Indeed possibly faster.