Extending Scalar Multiplication using Double Bases

Roberto Avanzi Vassil Dimitrov Christophe Doche Francesco Sica

26/06/2008

Talk Outline

Introduction to Elliptic Curves

Elliptic Curve Definition

$E/\mathbb{F}_q$ is given by an equation of a plane curve:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad \text{with } a_i \in \mathbb{F}_q$$

The set of solutions $(x, y) \in \mathbb{F}_q \times \mathbb{F}_q$ together with the point "at infinity" $\mathcal{O}$ is denoted by $E(\mathbb{F}_q)$

$q = 2^{\mathbf{p}}$	binary curve	$y^2 + xy = x^3 + a_2x^2 + a_6$
$q = 3^p$	ternary curve	$y^2 = x^3 + a_2x^2 + a_4x + a_6$
$q = p \geq 5$	prime curve	$y^2 = x^3 + a_4x + a_6$

Hasse's Bound

The number of $\mathbb{F}_q$-rational points on an elliptic curve $E/\mathbb{F}_q$ satisfies

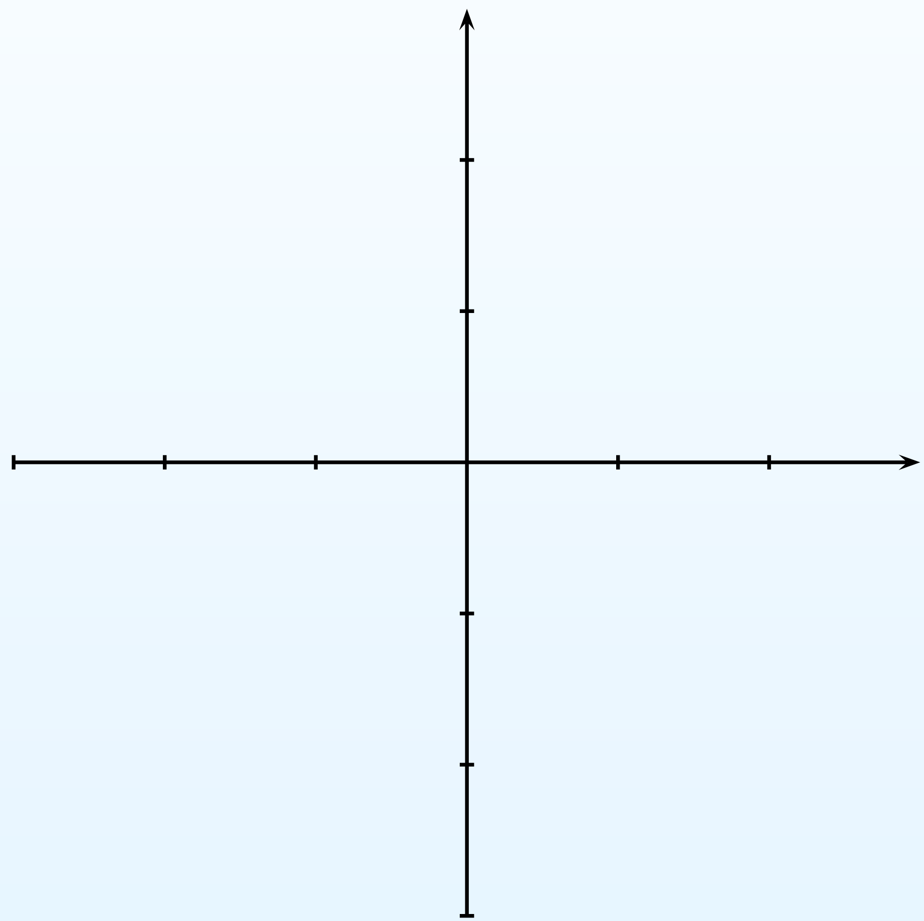
$$\#E(\mathbb{F}_q) = q + 1 - t$$

where

$$|t| \leq 2\sqrt{q}$$

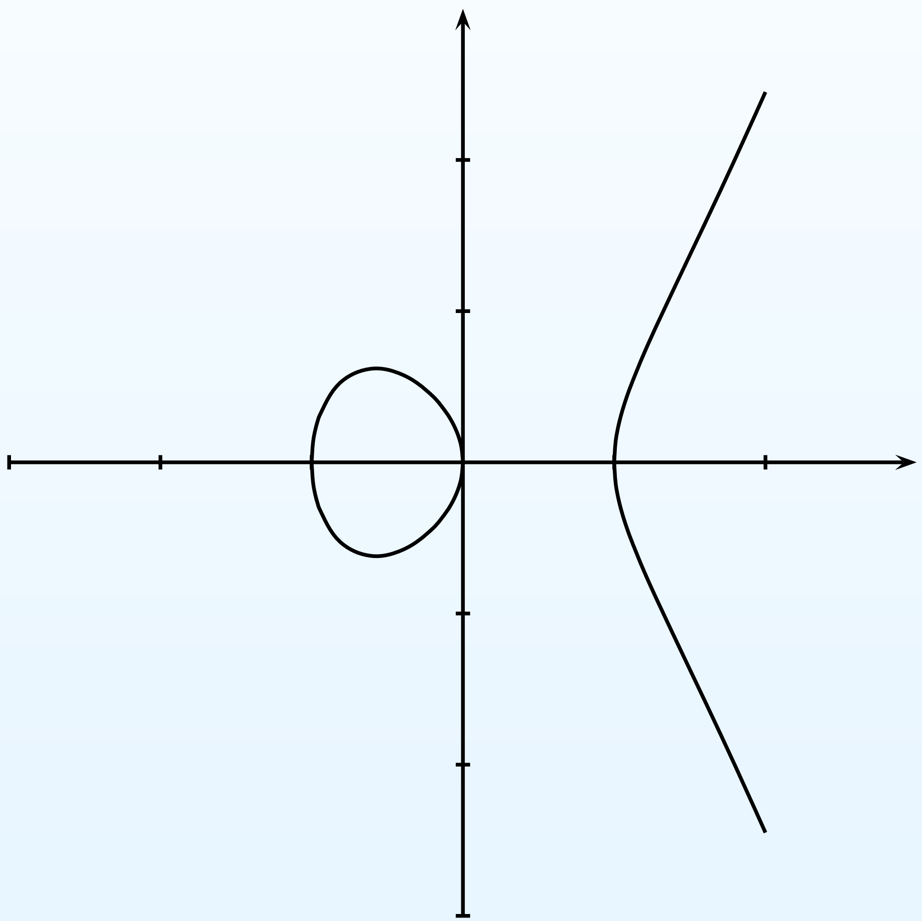
Group Law

$$y^2 = x^3 - x$$



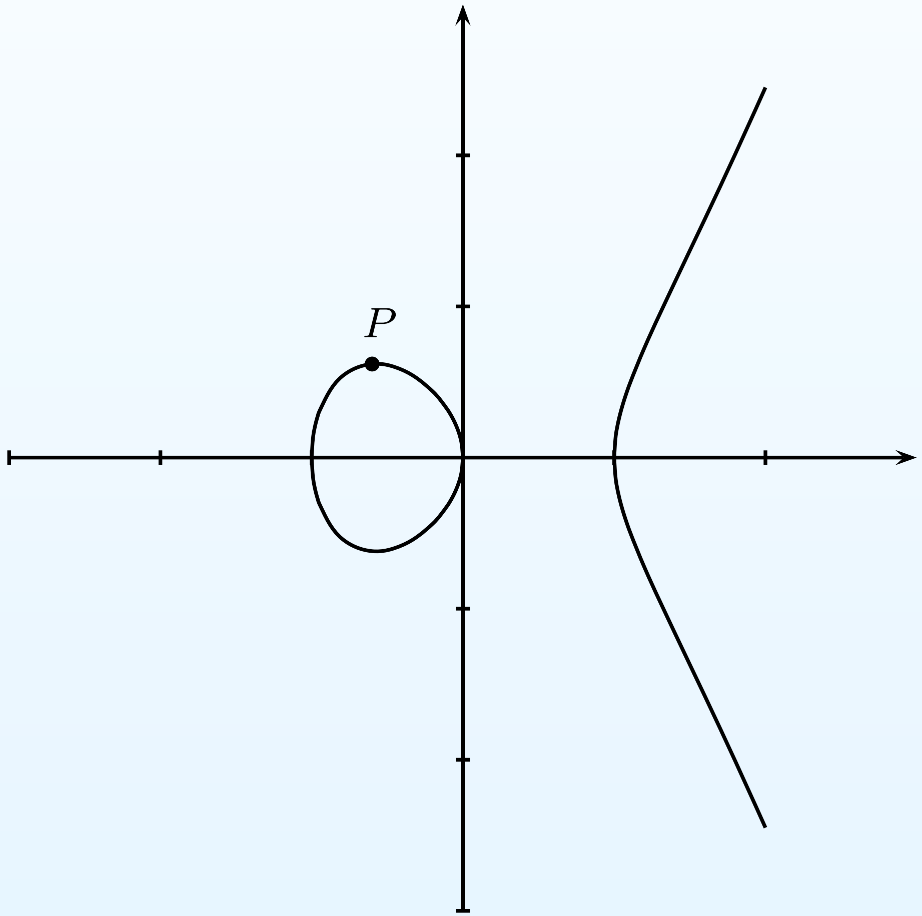
Group Law

$$y^2 = x^3 - x$$



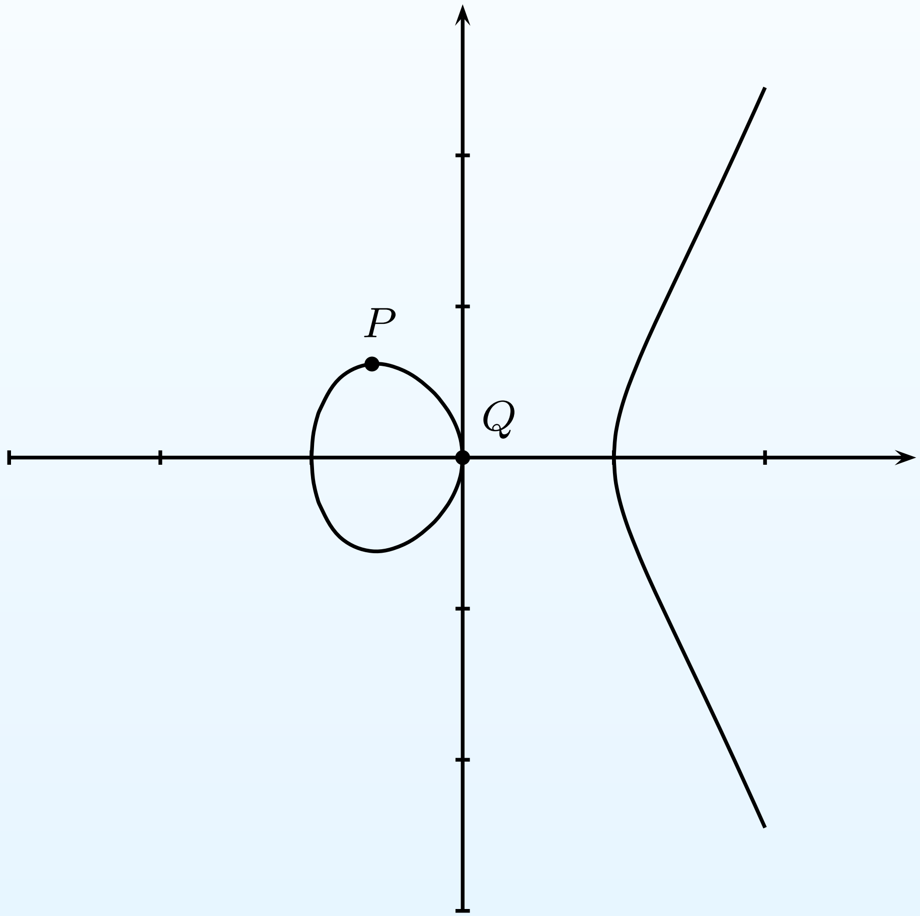
Group Law

$$y^2 = x^3 - x$$



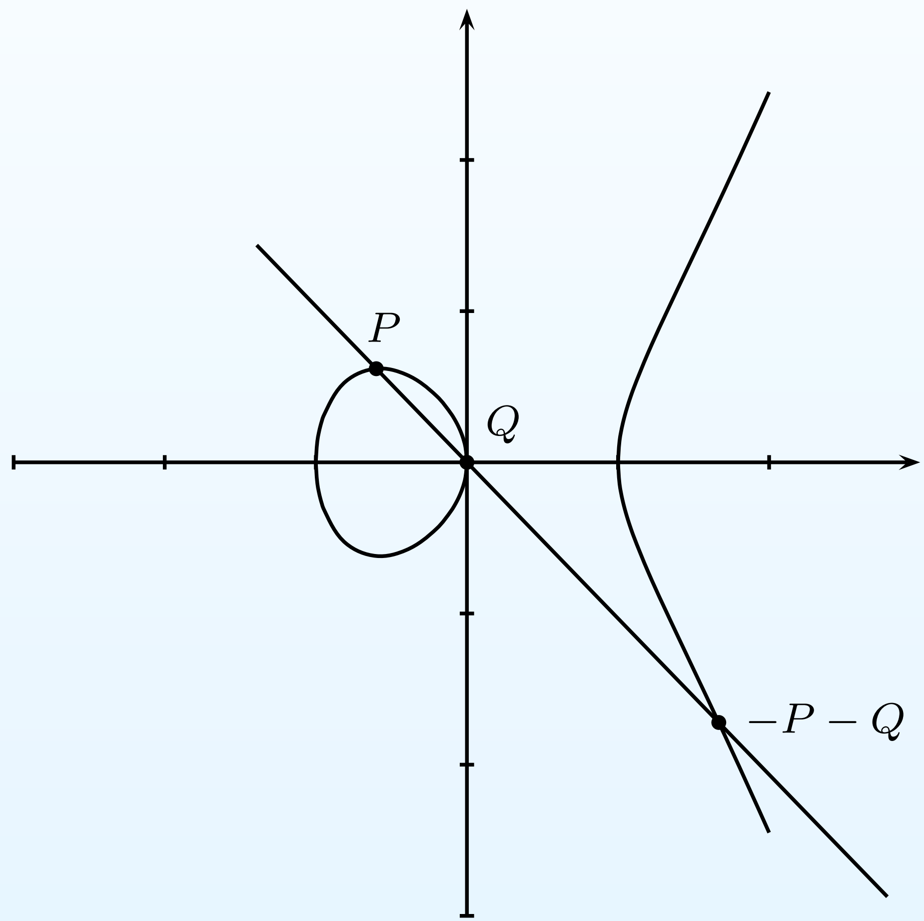
Group Law

$$y^2 = x^3 - x$$



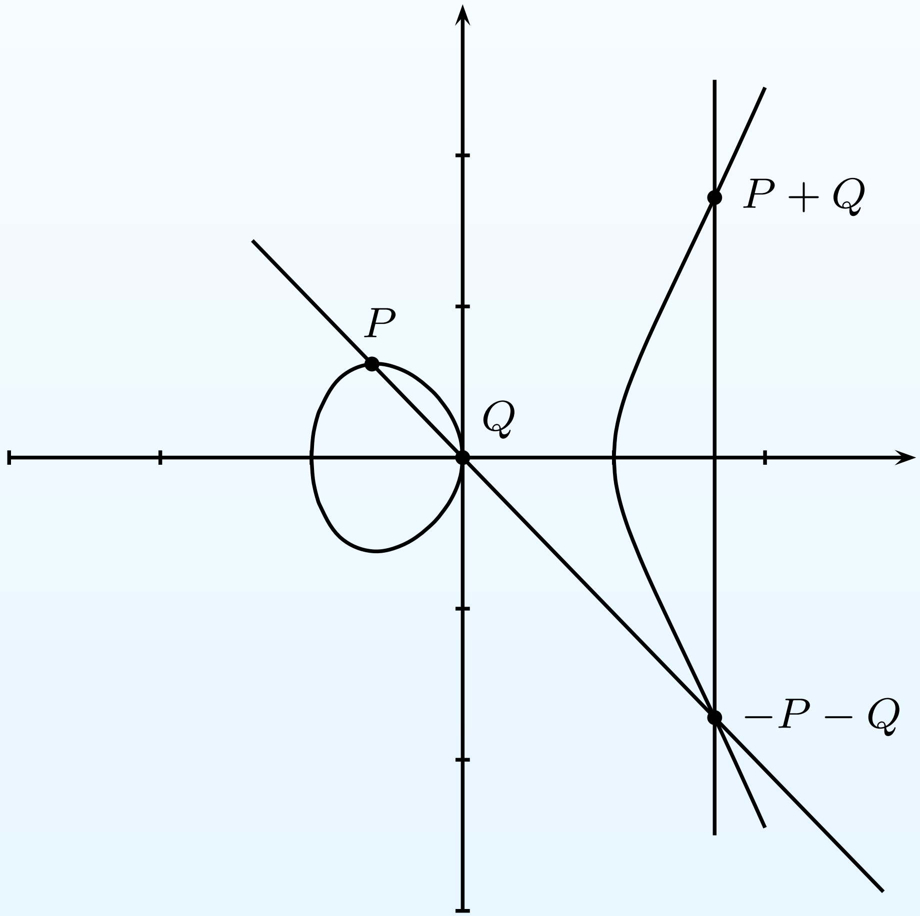
Group Law

$$y^2 = x^3 - x$$



Group Law

$$y^2 = x^3 - x$$



Koblitz Curves

$E_a/\mathbb{F}_{2^{\text{P}}}$ is a Koblitz curve if

$$y^2 + xy = x^3 + ax^2 + 1 \text{ with } a = 0, 1$$

Choice of a good extension is dictated by the existence of a large subgroup (of index less than 5 in $E_a(\mathbb{F}_{2^{\text{P}}})$) of prime order, generated, say, by a point P.

Supersingular Koblitz Curves in char 3

$E_b/\mathbb{F}_{3^p}$ is a supersingular ternary Koblitz curve if

$$y^2 = x^3 - x + b \text{ with } b = \pm 1$$

Supersingular ternary Koblitz curves have found many applications to pairing-based cryptosystems (e.g. IBE).
Choice of a good extension is dictated by the existence of a large subgroup (of index less than 5 in $E_b(\mathbb{F}_{3^p})$) of prime order, generated, say, by a point P.

The Power of Frobenius

Frobenius map:

$$\begin{aligned} \tau : E_a(\mathbb{F}_{2^\mathbf{p}}) &\longrightarrow E_a(\mathbb{F}_{2^\mathbf{p}}) \\ (x, y) &\longmapsto (x^2, y^2) \end{aligned}$$

Frobenius is very cheap (time is $O(1)$ in normal bases and $O(\mathbf{p})$ using polynomial reduction).

Fast Triplication Formulas

On E_b computation of $3P$ is very fast (equivalent to 2 Frobeniuses)

$$\begin{aligned} \mathbf{3}: \; E_b(\mathbb{F}_{3^p}) &\longrightarrow E_b(\mathbb{F}_{3^p}) \\ (x, y) &\longmapsto (x^9 - b, -y^9) \end{aligned}$$

Duplication Formulas in char 3

$R = 2P$.

$$R = \left(\frac{1}{y_P^2} + x_P, -\frac{x_R + x_P^3 + x_P - 2b}{y_P} \right)$$

These operations are costly!

Scalar Multiplication Algorithms

Scalar multiplication nP

Costliest part in any EC-based crypto algorithm. Several methods. All input n and P and output nP.

Scalar multiplication nP

Costliest part in any EC-based crypto algorithm. Several methods. All input n and P and output nP.

- Double and add

Scalar multiplication nP

Costliest part in any EC-based crypto algorithm. Several methods. All input n and P and output nP.

- Double and add
- Double and add-subtract (with NAF)

Scalar multiplication nP

Costliest part in any EC-based crypto algorithm. Several methods. All input n and P and output nP.

- Double and add
- Double and add-subtract (with NAF)
- τ and add-subtract

Scalar multiplication nP

Costliest part in any EC-based crypto algorithm. Several methods. All input n and P and output nP.

- Double and add
- Double and add-subtract (with NAF)
- τ and add-subtract
- Triple and add-subtract

Scalar multiplication nP

Costliest part in any EC-based crypto algorithm. Several methods. All input n and P and output nP.

- Double and add
- Double and add-subtract (with NAF)
- τ and add-subtract
- Triple and add-subtract
- Double base algorithms

Scalar multiplication nP

Costliest part in any EC-based crypto algorithm. Several methods. All input n and P and output nP.

- Double and add
- Double and add-subtract (with NAF)
- τ and add-subtract
- Triple and add-subtract
- Double base algorithms

Note that all the single base algorithms have a *linear* $(> c \log n)$ cost in the number of elliptic curve operations whereas double base algorithms have a *sublinear* cost in $O(\log n / \log \log n)$.

Double and add

$n = \langle n_{\mathbf{p}-1} n_{\mathbf{p}-2} \dots n_0 \rangle_2$ with $n_i = 0, 1$

1. $\boldsymbol{Q} = \mathcal{O}$
2. for $i = \mathbf{p} - 1$ down to 0
   (a) $\boldsymbol{Q} = 2\boldsymbol{Q}$
   (b) if $n_i \neq 0$ then $\boldsymbol{Q} = \boldsymbol{Q} + P$
3. return $\boldsymbol{Q}$

Average Cost: $\mathbf{p}$ doublings and $\mathbf{p}/2$ additions

Double and add-subtract (NAF)

Idea: if $P = (x, y)$ then $-P = (x, -y)$

$n = \langle n_{\mathbf{p}-1} n_{\mathbf{p}-2} \dots n_0 \rangle_2$ with $n_i = 0, \pm 1$ and $n_i n_{i+1} = 0$

1. $\boldsymbol{Q} = \mathcal{O}$
2. for $i = \mathbf{p} - 1$ down to 0
   (a) $\boldsymbol{Q} = 2\boldsymbol{Q}$
   (b) if $n_i \neq 0$ then $\boldsymbol{Q} = \boldsymbol{Q} + n_i P$
3. return $\boldsymbol{Q}$

Average Cost: $\mathbf{p}$ doublings and $\mathbf{p}/3$ additions

τ and add-subtract (char 2)

$n = \langle n_{\mathbf{p}-1} n_{\mathbf{p}-2} \ldots n_0 \rangle_\tau$ with $n_i = 0, \pm 1$ and $n_i n_{i+1} = 0$

1. $\boldsymbol{Q} = \mathcal{O}$
2. for $i = \mathbf{p} - 1$ down to 0
   (a) $\boldsymbol{Q} = \tau(\boldsymbol{Q})$
   (b) if $n_i \neq 0$ then $\boldsymbol{Q} = \boldsymbol{Q} + n_i P$
3. return $\boldsymbol{Q}$

Average Cost: $\mathbf{p}$ Frobeniuses and $\mathbf{p}/3$ additions

Triple and add-subtract

$n = \langle n_{p-1} n_{p-2} \dots n_0 \rangle_3$ with $n_i = 0, \pm 1$

1. $\boldsymbol{Q} = \mathcal{O}$
2. for $i = p - 1$ down to 0
   (a) $\boldsymbol{Q} = 3\boldsymbol{Q}$
   (b) if $n_i \neq 0$ then $\boldsymbol{Q} = \boldsymbol{Q} + n_i P$
3. return $\boldsymbol{Q}$

Average Cost: p triplings and $2p/3$ additions. Note that $p = \mathbf{p} \log 2 / \log 3$

Double Base Expansions

It is a finite expansion of n into a double base $\{A, B\}$ of the form

$$n = \sum_{s,t} A^s B^t$$

We can reorder the exponents s, t in lexicographic order as

$$n = \sum_{i=1}^{\mathcal{I}} A^{s_i} \sum_{j=1}^{\mathcal{J}_i} B^{t_{i,j}} \;, \quad s_i > s_{i+1} \quad \text{and} \quad t_{i,j} > t_{i,j+1}$$

where the map $P \mapsto BP$ is fast (Frobenius or triplication on supersingular ternary curves)

Double Base Scalar Multiplication

1. $Q \leftarrow \mathcal{O}$
2. For $i = 1$ to $\mathcal{I} - 1$
3. $\quad R \leftarrow P$
4. $\quad$ For $j = 1$ to $\mathcal{J}_i$
5. $\quad\quad R \leftarrow B^{t_{i,j} - t_{i,j+1}} R + P$
6. $\quad\quad Q \leftarrow Q + R$
7. $\quad Q \leftarrow A^{s_i - s_{i+1}} Q$
8. $R \leftarrow P$
9. For $j = 1$ to $\mathcal{J}_{\mathcal{I}}$
10. $\quad R \leftarrow B^{t_{\mathcal{I},j} - t_{\mathcal{I},j+1}} R + P$
11. $\quad Q \leftarrow Q + R$
12. Return Q

Performance of Double Base Scalar Multiplication

If multiplication by B can be neglected, then total cost is bounded by

$$c \cdot \frac{\log n}{\log \log n}$$

elliptic curve operations if the number of addends is less than this bound and $\max s_i = s_1 = o\left(\frac{\log n}{\log \log n}\right)$.

How can we achieve this bound?

Decomposition Algorithms

Greedy Binary

To find $n = \langle n_{\mathbf{p}-1} n_{\mathbf{p}-2} \ldots n_0 \rangle_2 = \sum_{i=0}^{\mathbf{p}-1} n_i 2^i$ do

1. $n_i \leftarrow 0$, $N \leftarrow n$, $i \leftarrow \mathbf{p} - 1$
2. while $N > 0$
   (a) find largest power $2^k \leq N$ with $k \leq i$
   (b) $n_k \leftarrow 1$
   (c) $N \leftarrow N - 2^k$
   (d) $i \leftarrow k - 1$
3. return $\langle n_{\mathbf{p}-1} n_{\mathbf{p}-2} \ldots n_0 \rangle_2$

Algebraic Binary

In this case we find bits from least significant to most significant (right to left).

1. $n_i \leftarrow 0$, $N \leftarrow n$, $i \leftarrow 0$
2. while $N > 0$
   (a) If 2 does not divide N
      i. $n_i \leftarrow 1$
      ii. $N \leftarrow (N - 1)$
   (b) $N \leftarrow N/2$
   (c) $i \leftarrow i + 1$
3. return $\langle n_{\mathbf{p}-1} n_{\mathbf{p}-2} \dots n_0 \rangle_2$

Greedy NAF

Greedy algorithms quickly become cumbersome.

1. $n_i \leftarrow 0, N \leftarrow n, i \leftarrow \mathbf{p} - 1, \sigma \leftarrow 1$
2. while $N > 0$
   (a) find $k \leq i + 1$ with $|N - 2^k| \leq 2^{k-2}$
   (b) $n_k \leftarrow \sigma$
   (c) $\sigma \leftarrow \mathrm{sign}(N - 2^k)$
   (d) $N \leftarrow |N - 2^k|$
   (e) $i \leftarrow k - 2$
3. return $\langle n_{\mathbf{p}-1} n_{\mathbf{p}-2} \ldots n_0 \rangle_2$

Algebraic NAF

1. $n_i \leftarrow 0$, $N \leftarrow n$, $i \leftarrow 0$
2. while $N > 0$
   (a) If 2 does not divide N
      i. $n_i \leftarrow \pm 1$ where $\pm 1 \equiv N \pmod 4$
      ii. $N \leftarrow (N \mp 1)$ (opposite sign from above)
   (b) $N \leftarrow N/2$
   (c) $i \leftarrow i + 1$
3. return $\langle n_{\mathbf{p}-1} n_{\mathbf{p}-2} \ldots n_0 \rangle_2$

Algebraic τ-NAF

1. $n_i \leftarrow 0$, $\zeta \leftarrow n$, $i \leftarrow 0$
2. while $N > 0$
   (a) If τ does not divide ζ
      i. $n_i \leftarrow \pm 1$ where $\pm 1 \equiv \zeta \pmod{\tau^2}$
      ii. $\zeta \leftarrow (\zeta \mp 1)$ (opposite sign from above)
   (b) $\zeta \leftarrow \zeta / \tau$
   (c) $i \leftarrow i + 1$
3. return $\langle n_{\mathbf{p}-1} n_{\mathbf{p}-2} \ldots n_0 \rangle_\tau$

Double Base Recodings

At first:

- recoding of n given by a greedy algorithm

Double Base Recodings

At first:

- recoding of n given by a greedy algorithm
- no theoretical analysis (decomposition by trial and error)

Double Base Recodings

At first:

- recoding of n given by a greedy algorithm
- no theoretical analysis (decomposition by trial and error)
- makes use of the continued fraction expansion of $\log 3/\log 2$ and diophantine approximation

Double Base Recodings

At first:

- recoding of n given by a greedy algorithm
- no theoretical analysis (decomposition by trial and error)
- makes use of the continued fraction expansion of $\log 3/\log 2$ and diophantine approximation
- produces poor bound on the Hamming weight of the expansion of n more precisely

Double Base Recodings

At first:

- recoding of n given by a greedy algorithm
- no theoretical analysis (decomposition by trial and error)
- makes use of the continued fraction expansion of $\log 3/\log 2$ and diophantine approximation
- produces poor bound on the Hamming weight of the expansion of n more precisely
- constant c is not optimal (3 for $\{2,3\}$-base, 12 for $\{3,\tau\}$-base) when $c = 1$ is conjectured

Double Base Recodings

At first:

- recoding of n given by a greedy algorithm
- no theoretical analysis (decomposition by trial and error)
- makes use of the continued fraction expansion of $\log 3/\log 2$ and diophantine approximation
- produces poor bound on the Hamming weight of the expansion of n more precisely
- constant c is not optimal (3 for $\{2, 3\}$-base, 12 for $\{3, \tau\}$-base) when $c = 1$ is conjectured
- fails in the very interesting case of a double complex base, like $\{\bar{\tau}, \tau\}$ on Koblitz curves

Greedy Double Base Recodings

- **Unsigned binary**: use the fact that given n, there exists a power of 2, say N, such that $n/2 < N \leq n$ (optimal result). Then use inductive argument replacing n by $n - N$ to get all the bits of n.

Greedy Double Base Recodings

- **Unsigned binary**: use the fact that given n, there exists a power of 2, say N, such that $n/2 < N \leq n$ (optimal result). Then use inductive argument replacing n by $n - N$ to get all the bits of n.
- **Unsigned $\{2, 3\}$ number**: given n, there exists $N = 2^u 3^v$ with

$$n\left(1 - \frac{1}{\sqrt{\log n}}\right) < N \leq n$$

Use inductive argument to get binumber expansion of length $k = O\left(\frac{\log n}{\log \log n}\right)$

Double Base Algebraic Recoding

1. $N \leftarrow n$
2. While $N > 0$
3. Until $3 \mid N$, $N \leftarrow N/3$
4. Find $0 \leq j \leq 3^{u-1}2$ with $N \equiv 2^j \pmod{3^u}$
5. $N \leftarrow (N - 2^j)/3^u$
6. Return “dibits”

Analysis of Double Base Algebraic Recoding

At each loop, Step 5 divides N at least by 3^u. Therefore at most $\dfrac{\log n}{u \log 3}$ loops are needed. So a total of at most

Analysis of Double Base Algebraic Recoding

At each loop, Step 5 divides N at least by 3^u. Therefore at most $\dfrac{\log n}{u \log 3}$ loops are needed. So a total of at most

$$(1+\epsilon)\frac{\log n}{\log\log n}$$

curve operations with $u = (1+\epsilon)^{-1}\dfrac{\log\log n}{\log 3}$, as $n \to \infty$.

Analysis of Double Base Algebraic Recoding

At each loop, Step 5 divides N at least by 3^u. Therefore at most $\dfrac{\log n}{u \log 3}$ loops are needed. So a total of at most

$$(1+\epsilon)\frac{\log n}{\log \log n}$$

curve operations with $u = (1+\epsilon)^{-1}\dfrac{\log\log n}{\log 3}$, as $n \to \infty$.

Note that highest power of slow endomorphism (here $P \mapsto 2P$) is small: $s_1 = O\left(\log^{1-\epsilon} n\right)$

Complex Double Bases

1. $N \leftarrow n \pmod{\frac{\tau^{\mathbf{p}} - 1}{\tau - 1}}$
2. While $|N| \geq 2^{2^{u-3}}$ do
3. Until $\tau \mid N$, $N \leftarrow N/\tau$
4. Find $0 \leq j \leq 2^{u-2}$ and $e = 0, 1$ with $N \equiv (-1)^e \bar{\tau}^j \pmod{\tau^u}$
5. $N \leftarrow (N - (-1)^e \bar{\tau}^j)/\tau^u$
6. Produce the τ-NAF expansion of N
7. Return “dibits”

Analysis of Complex Double Base Recoding

At each loop, Step 5 divides N at least by τ^u. At the end we must find the τ-NAF expansion of an integer in $\mathbb{Z}[\tau]$ of norm less than $2^{2^{u-2}}$. Its expected Hamming weight is therefore around $2^{u-2}/3$.

Analysis of Complex Double Base Recoding

At each loop, Step 5 divides N at least by τ^u. At the end we must find the τ-NAF expansion of an integer in $\mathbb{Z}[\tau]$ of norm less than $2^{2^{u-2}}$. Its expected Hamming weight is therefore around $2^{u-2}/3$. Hence we expect a total of

$$\frac{\log n}{u \log 2} + \frac{2^{u-2}}{3} \text{ additions and } 2^{u-2} \text{ applications of } \bar{\tau}$$

Analysis of Complex Double Base Recoding

At each loop, Step 5 divides N at least by τ^u. At the end we must find the τ-NAF expansion of an integer in $\mathbb{Z}[\tau]$ of norm less than $2^{2^{u-2}}$. Its expected Hamming weight is therefore around $2^{u-2}/3$. Hence we expect a total of

$$\frac{\log n}{u \log 2} + \frac{2^{u-2}}{3} \text{ additions and } 2^{u-2} \text{ applications of } \bar{\tau}$$

So a total of at most

$$(1+\epsilon)\frac{\log n}{\log \log n}$$

curve operations again with $u = (1+\epsilon)^{-1}\dfrac{\log \log n}{\log 2}$, as $n \to \infty$.

Note that highest power of slow endomorphism (here $P \mapsto \bar{\tau}P$) is small: $s_1 = O\left(\log^{1-\epsilon} n\right)$

Replacing $\bar{\tau}$ with $1/2$

Taking advantage of the fact that in char 2 halving is 50% faster than $\bar{\tau} = 1 - \tau$.

Replacing $\bar{\tau}$ with $1/2$

Taking advantage of the fact that in char 2 halving is 50% faster than $\bar{\tau} = 1 - \tau$.

Suppose we want to compute ζP, where $\zeta \in \mathbb{Z}[\tau]$. We let $\zeta' = 2^{2^{u-2}}\zeta \pmod{\dfrac{\tau^{\mathbf{p}} - 1}{\tau - 1}}$.

Replacing $\bar{\tau}$ with $1/2$

Taking advantage of the fact that in char 2 halving is 50% faster than $\bar{\tau} = 1 - \tau$.
Suppose we want to compute ζP, where $\zeta \in \mathbb{Z}[\tau]$. We let $\zeta' = 2^{2^{u-2}}\zeta \pmod{\frac{\tau^{\mathbf{p}}-1}{\tau-1}}$. We get

$$\zeta P = \sum_{i=0}^{k-1} (-1)^{e'_i} \frac{\bar{\tau}^{s'_i}}{2^{2^{u-2}}} \tau^{t'_i} P = \sum_{i=0}^{k-1} (-1)^{e'_i} \frac{\tau^{t'_i - s'_i}}{2^{2^{u-2} - s'_i}} P$$

$$= \sum_{i=0}^{k-1} (-1)^{e'_i} \frac{\tau^{\epsilon_i \mathbf{p} + t'_i - s'_i}}{2^{2^{u-2} - s'_i}} P$$

where $\epsilon_i = 1$ if $t'_i < s'_i$ and 0 else. We thus get a DB expansion in base $\{1/2, \tau\}$.

Memory Requirements

This recoding and scalar multiplication needs only $O\big((\log\log n)^2\big)$ bits.
Indeed, we only need to store the table giving, for each $N \pmod{3^u}$, say, the value $0 \le j \le 3^{u-1}2$ such that $N \equiv 2^j \pmod{3^u}$.

Performance Comparison

We compare our new algorithm (in an improved version on Koblitz curves in char 2) to existing ones without and with precomputation.

Field size $\mathbf{p}$	τ-NAF	w-τ-NAF	w	DBNS $(\frac{1}{2}, \tau)$	u	%/τ-NAF	%/w-τ-NAF
163	54.33	34.16	5	31.09	5	42.78%	8.99%
233	77.66	45.83	5	41.38	6	46.72%	9.71%
283	94.33	54.16	5	48.80	6	48.27%	9.90%
409	136.33	73.42	6	66.89	6	50.94%	8.90%
571	190.33	102.37	6	88.04	7	53.74%	14.00%

Comparison of scalar multiplication algorithms on Koblitz curves

Further results

Lower Bounds on a Double Base Expansion

Any unsigned $\{2,3\}$-expansion or any $\{2,3\}$-expansion (resp. $\{3,\tau\}$ or $\{\bar{\tau},\tau\}$-expansion) where the exponents of the base elements are bounded above by $C\log n$ (i.e. found by means of a greedy algorithm) must have length

$$k \geq \frac{\log n}{\log\log n} + o\left(\frac{\log n}{\log\log n}\right)$$

In particular, one cannot reasonably hope to go below this order of complexity and $c \geq 1$.

Explanation

For simplicity assume bases are $\{2,3\}$. Let $M = \lceil C \log n \rceil$. There are M^2 different numbers $2^s 3^t$ with $s, t \leq M$. Therefore the number of positive integers less than 2^M which can be represented by at most k $\{2,3\}$-numbers of this form is upper bounded by

$$\sum_{i=1}^{k} 2^i \binom{M^2}{i} \leq k\, 2^k \binom{M^2}{k} = \frac{\Gamma(M^2+1)}{\Gamma(k+1)\Gamma(M^2-k)}\, k\, 2^k$$

Substituting $k = c \log n / \log \log n$ and using Stirling's formula, we find that the right-hand side is $o(2^M)$ whenever $c < 1$, therefore the number of n's that can be represented in this way is $o(2^M)$ which is a negligible fraction of 2^M.

Double Base Chains

They are double base expansions

$$n = \sum_i A^{s_i} B^{t_i}$$

where $s_i \geq s_{i+1}$ and $t_i \geq t_{i+1}$.
Advantage: Scalar multiplication with a single loop, can be applied to all elliptic curves.
Unfortunately, our algorithm does not seem to give double base chains.

Recent work by others

Miri-Longa, Longa-Gebotys:

- NAF Double base chains using an algebraic method (compared to the greedy algorithm of Dimitrov-Imbert-Mishra)

Recent work by others

Miri-Longa, Longa-Gebotys:

- NAF Double base chains using an algebraic method (compared to the greedy algorithm of Dimitrov-Imbert-Mishra)
- Length better than usual NAF, but not sublinear

Recent work by others

Miri-Longa, Longa-Gebotys:

- NAF Double base chains using an algebraic method (compared to the greedy algorithm of Dimitrov-Imbert-Mishra)
- Length better than usual NAF, but not sublinear
- in large bases or w-NAF, need to precompute and store points

Conclusion

- Double bases provide a new generation of scalar multiplication implementations on curves with very fast endomorphisms (Koblitz curves).
- Idea of double bases: use fast multiplication “ad nauseam”. Still cheap!
- First algebraic (right-to-left) algorithm to write a double base expansion of a scalar n
- Works also when both bases are complex
- Much faster than previous algorithms, proven optimal length of $\log n / \log \log n$
- Extend this theoretical analysis to double base chains: could be adapted to all elliptic curves (however sublinearity seems hard to achieve)