Pairings on elliptic curves – parameter selection and efficient computation

Michael Naehrig

Microsoft Research
mnaehrig@microsoft.com

Workshop on Elliptic Curve Computation
Redmond, 19 October 2010

Pairings on elliptic curves

parameter selection and efficient computation

Three parts:

- Pairings and pairing-friendly curves,
- an optimal ate pairing on BN curves using the polynomial parametrization,
- affine coordinates for pairing computation at high security levels.

The embedding degree

Let E be an elliptic curve over $\mathbb{F}_q$ (of characteristic p) and

- $n = \#E(\mathbb{F}_q) = q + 1 - t, \quad |t| \leq 2\sqrt{q}$,
- $r \mid n$ a large prime divisor of n ($r \neq p$, $r \geq \sqrt{q}$).

The embedding degree of E with respect to r is the smallest positive integer k with

$$r \mid q^k - 1.$$

Then

- k is the order of q modulo r,
- r-th roots of unity $\mu_r \subseteq \mathbb{F}_{q^k}^*$,
- for $k > 1$, $E[r] \subseteq E(\mathbb{F}_{q^k})$.

The Tate pairing

The Tate-Lichtenbaum pairing

$$\begin{aligned} T_r : E(\mathbb{F}_{q^k})[r] \times E(\mathbb{F}_{q^k})/[r]E(\mathbb{F}_{q^k}) &\rightarrow \mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r, \\ (P, Q + [r]E(\mathbb{F}_{q^k})) &\mapsto f_{r,P}(\mathcal{D}_Q)(\mathbb{F}_{q^k}^*)^r \end{aligned}$$

is a non-degenerate, bilinear map, where

- $f_{r,P}$ is a function with divisor $(f_{r,P}) = r(P) - r(\mathcal{O})$,
- $\mathcal{D}_Q \sim (Q) - (\mathcal{O})$ has support disjoint from $\{\mathcal{O}, P\}$.

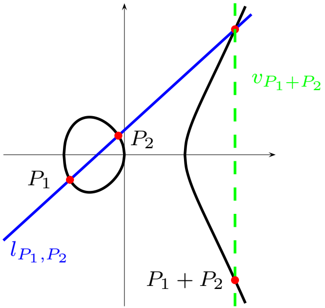
Assume $k > 1$, can use the reduced Tate pairing

$$\begin{aligned} t_r : E(\mathbb{F}_q)[r] \times E(\mathbb{F}_{q^k})[r] &\rightarrow \mu_r, \\ (P, Q) &\mapsto f_{r,P}(Q)^{\frac{q^k-1}{r}}. \end{aligned}$$

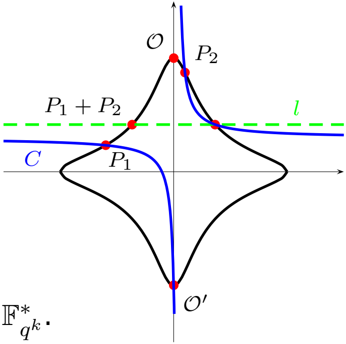
Computing Miller functions

To compute $f_{m,P}(Q)$, $m \in \mathbb{Z}$, with Miller's algorithm use

$$\begin{aligned} f_{2i,P}(Q) &= f_{i,P}(Q)^2 \frac{l_{[i]P,[i]P}(Q)}{v_{[2i]P}(Q)}, \\ f_{i\pm1,P}(Q) &= f_{i,P}(Q) \frac{l_{[i]P,\pm P}(Q)}{v_{[i\pm1]P}(Q)}. \end{aligned}$$



- square-&-multiply-like loop,
- evaluate at Q on the fly,
- update with fraction of line functions,
- on Edwards curves, use fraction of quadratic and line functions.



Computations are in $E(\mathbb{F}_q)$, $E(\mathbb{F}_{q^k})$ and $\mathbb{F}_{q^k}^*$.

Common group choices, Tate and ate pairing

Arguments usually restricted to groups

- $G_1 = E(\mathbb{F}_{q^k})[r] \cap \ker(\phi_q - [1]) = E(\mathbb{F}_q)[r]$,
- $G_2 = E(\mathbb{F}_{q^k})[r] \cap \ker(\phi_q - [q])$.

Get mainly two variants:

- reduced Tate pairing

$$t_r : G_1 \times G_2 \to G_3,\ (P, Q) \mapsto f_{r,P}(Q)^{\frac{q^k-1}{r}},$$

- ate pairing ($T = t - 1$, $\log(T) \lesssim \log(r)/2$)

$$a_T : G_2 \times G_1 \to G_3,\ (Q, P) \mapsto f_{T,Q}(P)^{\frac{q^k-1}{r}}.$$

Has more efficient variants: optimal ate pairings that are computed from some $f_{m,Q}(P)$ with $\log(m) \approx \log(r)/\varphi(k)$.

Using a twist to represent G_2

Let $p > 5$ and $E : y^2 = x^3 + ax + b$.
Here: A twist E' of E is a curve isomorphic to E over $\mathbb{F}_{q^k}$.

- A twist is given by

$$E' : y^2 = x^3 + (a/\omega^4)x + (b/\omega^6), \omega \in \mathbb{F}_{q^k}^*$$

  with isomorphism $\psi : E' \to E, (x', y') \mapsto (\omega^2 x', \omega^3 y')$.
- If E' is defined over $\mathbb{F}_{q^{k/d}}$ for $d \mid k$, and ψ is defined over $\mathbb{F}_{q^k}$ and no smaller field, d is called the degree of E'.
- Possible twist degrees: can have $d = 2$, $d = 4$ (for $b = 0$ only), $d = 3$ and $d = 6$ (both for $a = 0$ only).
- Let $d_0 = 6$ if $a = 0$, let $d_0 = 4$ if $b = 0$, and $d_0 = 2$ otherwise. Then there exists a unique twist E' of degree $d = \gcd(d_0, k)$ with $r \mid \#E'(\mathbb{F}_{q^{k/d}})$.

Using a twist to represent G_2

Let E' be the unique twist of degree d with $r \mid \#E'(\mathbb{F}_{q^{k/d}})$.

- Let $G_2' = E'(\mathbb{F}_{q^{k/d}})[r]$, then $\psi : G_2' \to G_2$ is a group isomorphism,
- if $\mathbb{F}_{q^k} = \mathbb{F}_{q^{k/d}}(\omega)$, ψ is very convenient,
- points in G_2 *almost* have coefficients in subfield $\mathbb{F}_{q^{k/d}}$.

$$\begin{tikzcd}
E'(\mathbb{F}_{q^k}) \arrow[r, "\psi"'] \arrow[d, dash] & E(\mathbb{F}_{q^k}) \arrow[d, dash] \\
E'(\mathbb{F}_{q^{k/d}}) & E(\mathbb{F}_{q^{k/d}}) \arrow[d, dash] \\
 & E(\mathbb{F}_q)
\end{tikzcd}$$

$$\begin{tikzcd}
G_1' & & G_2 \\
G_2' \arrow[urr, "\psi"] & & \\
 & & G_1 \arrow[uull, "\psi^{-1}"']
\end{tikzcd}$$

Minimal requirements for security

- k should be small, but DLPs must be hard enough.

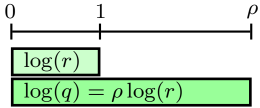
Security level (bits)	EC base point order r (bits)	Extension field size of q^k (bits)		ratio $\rho \cdot k$	
		NIST	ECRYPT	NIST	ECRYPT
80	160	1024	1248	6.4	7.8
112	224	2048	2432	9.1	10.9
128	256	3072	3248	12.0	12.7
192	384	7680	7936	20.0	20.7
256	512	15360	15424	30.0	30.1

NIST/ECRYPT II recommendations

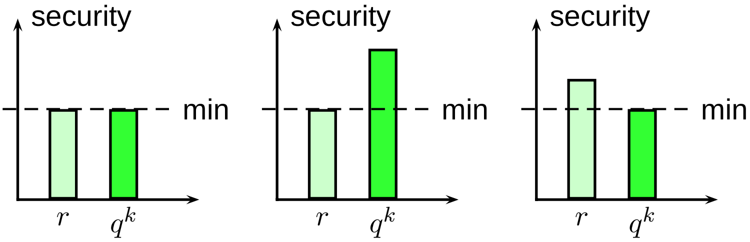
The ρ-value of E is defined as $\rho = \log(q)/\log(r)$.

Balanced security

Do not want to waste recources, so balance the security as much as possible.



- If ρ is too large, q is larger than necessary.
- If ρk is too large, q^k is larger than necessary.
- If ρk is too small, r is larger than necessary.



(a) good ρk (b) ρk too large (c) ρk too small

Pairing-friendly curves

Supersingular curves have small embedding degree ($k \leq 6$, large char $p > 3$: $k \leq 2$ only).

To find ordinary curves with small embedding degree:
Fix k and find primes r, p and an integer n with the following conditions:

- $n = p + 1 - t$, $|t| \leq 2\sqrt{p}$,
- $r \mid n$,
- $r \mid p^k - 1$,
- $t^2 - 4p = Dv^2 < 0$, $D, v \in \mathbb{Z}$, $D < 0$, $|D|$ small enough to compute the Hilbert class polynomial for $\mathbb{Q}(\sqrt{D})$.

Given such parameters, a corresponding elliptic curve over $\mathbb{F}_p$ can be constructed using the CM method.

Pairing-friendly curve construction methods

Freeman, Scott, Teske: A taxonomy of pairing-friendly elliptic curves

security	construction	curve	k	ρ	ρk	d	k/d
128	BN (Ex. 6.8)	$a = 0$	12	1.00	12	6	2
	Ex. 6.10	$b = 0$	8	1.50	12	4	2
	Freeman (5.3)	$a, b \neq 0$	10	1.00	10	2	5
	Constr. 6.7+	$a, b \neq 0$	12	1.75	21	2	6
192	KSS (Ex. 6.12)	$a = 0$	18	1.33	24	6	3
	KSS (Ex. 6.11)	$b = 0$	16	1.25	20	4	4
	Constr. 6.3+	$a, b \neq 0$	14	1.50	21	2	7
256	Constr. 6.6	$a = 0$	24	1.25	30	6	4
	Constr. 6.4	$b = 0$	28	1.33	37	4	7
	Constr. 6.24+	$a, b \neq 0$	26	1.17	30	2	13

BN curves

(Barreto-N., 2005)

If $u \in \mathbb{Z}$ such that

$$\begin{aligned} p = p(u) &= 36u^4 + 36u^3 + 24u^2 + 6u + 1, \\ n = n(u) &= 36u^4 + 36u^3 + 18u^2 + 6u + 1 \end{aligned}$$

are both prime, then there exists an ordinary elliptic curve

- with equation $E : y^2 = x^3 + b,\ b \in \mathbb{F}_p$,
- $r = n = \#E(\mathbb{F}_p)$ is prime, i. e. $\rho \approx 1$,
- the embedding degree is $k = 12$,
- $t(u)^2 - 4p(u) = -3(6u^2 + 4u + 1)^2$,
- there exists a twist $E' : y^2 = x^3 + b/\xi$ over $\mathbb{F}_{p^2}$ of degree 6 with $n \mid \#E'(\mathbb{F}_{p^2})$.

BN curves

(Barreto-N., 2005)

$$\begin{aligned} p \; = \; p(u) &= 36u^4 + 36u^3 + 24u^2 + 6u + 1, \\ n \; = \; n(u) &= 36u^4 + 36u^3 + 18u^2 + 6u + 1, \\ E : y^2 &= x^3 + b, \\ E' : y^2 &= x^3 + b/\xi \end{aligned}$$

Thus we can represent G_2 by $G_2' = E'(\mathbb{F}_{p^2})[n]$.

- Replace all points $R \in G_2$ by $R' \in G_2'$ via $R = \psi(R')$,
- curve arithmetic over $\mathbb{F}_{p^2}$ instead of $\mathbb{F}_{p^{12}}$,
- represent field extensions of $\mathbb{F}_{p^2}$ using ξ

$$\mathbb{F}_{p^{2j}} = \mathbb{F}_{p^2}[X]/(X^j - \xi), \quad j \in \{2, 3, 6\}.$$

An optimal ate pairing on BN curves

Input: $P \in G_1 = E(\mathbb{F}_p)$, $Q = \psi(Q')$, $Q' \in G_2' \subseteq E'(\mathbb{F}_{p^2})$,
$m = 6u + 2 = (1, m_{s-1}, \ldots, m_0)_{\mathrm{NAF}}$.

Output: $a_{\mathrm{opt}}(Q, P)$.

$R \leftarrow Q$, $f \leftarrow 1$
for $(i \leftarrow s-1;\ i \geq 0;\ i--)$ **do**
  $f \leftarrow f^2 \cdot l_{R,R}(P)$, $R \leftarrow [2]R$
  if $(m_i = \pm 1)$ **then**
    $f \leftarrow f \cdot l_{R,\pm Q}(P)$, $R \leftarrow R \pm Q$
  end if
end for
if $u < 0$ **then**
  $f \leftarrow 1/f$, $R \leftarrow -R$
end if
$Q_1 = \phi_p(Q)$, $Q_2 = \phi_{p^2}(Q)$
$f \leftarrow f \cdot l_{R,Q_1}(P)$, $R \leftarrow R + Q_1$
$f \leftarrow f \cdot l_{R,-Q_2}(P)$, $R \leftarrow R - Q_2$
$f \leftarrow f^{p^6-1}$
$f \leftarrow f^{p^2+1}$
$f \leftarrow f^{(p^4-p^2+1)/n}$
return f

An optimal ate pairing on BN curves

Input: $P \in G_1 = E(\mathbb{F}_p)$, $Q = \psi(Q')$, $Q' \in G_2' \subseteq E'(\mathbb{F}_{p^2})$,
$m = 6u + 2 = (1, m_{s-1}, \ldots, m_0)_{\mathrm{NAF}}$.

Output: $a_{\mathrm{opt}}(Q, P)$.

$R \leftarrow Q$, $f \leftarrow 1$
for $(i \leftarrow s-1;\ i \geq 0;\ i--)$ **do**
  $f \leftarrow f^2 \cdot l_{R,R}(P)$, $R \leftarrow [2]R$
  if $(m_i = \pm 1)$ **then**
    $f \leftarrow f \cdot l_{R,\pm Q}(P)$, $R \leftarrow R \pm Q$
  end if
end for
if $u < 0$ **then**
  $f \leftarrow 1/f$, $R \leftarrow -R$
end if
$Q_1 = \phi_p(Q)$, $Q_2 = \phi_{p^2}(Q)$
$f \leftarrow f \cdot l_{R,Q_1}(P)$, $R \leftarrow R + Q_1$
$f \leftarrow f \cdot l_{R,-Q_2}(P)$, $R \leftarrow R - Q_2$
$f \leftarrow f^{p^6-1}$
$f \leftarrow f^{p^2+1}$
$f \leftarrow f^{(p^4-p^2+1)/n}$
return f

The importance of suitable curve parameters

The best performance is obtained by choosing

- $6u + 2$ as sparse as possible,
- u sparse or with a good addition chain,
- $p \equiv 3 \pmod 4$, so $\mathbb{F}_{p^2} = \mathbb{F}_p(i)$, $i^2 = -1$,
- ξ as "small" as possible to make field extension arithmetic more efficient.

One should also consider non-pairing operations:

- elliptic curve scalar multiplication,
- square root and cube root computation.

Constrained devices might not even need to compute pairings in certain pairing-based protocols.

- In some scenarios, pairings on Edwards curves could be the best choice.

Implementation-friendly BN curves

joint work with P. Barreto, G. Pereira, M. Simplicío

Theorem

Given a BN curve $E : y^2 = x^3 + b$ *with* $b = N(\xi)$ *for* $\xi \in \mathbb{F}_{p^2}$*, then the sextic twist* $E' : y^2 = x^3 + b/\xi$ *satisfies* $\#E(\mathbb{F}_p) \mid \#E'(\mathbb{F}_{p^2})$*.*

Suggestions for choosing BN curves:

- Choose low-weight u s.t.
- $6u + 2$ has low weight, and s.t.
- $p \equiv 3 \pmod 4$, i.e. $\mathbb{F}_{p^2} = \mathbb{F}_p(i)$, $i^2 = -1$,
- choose "small" $\xi = c^2 + id^3$, s.t. $b = c^4 + d^6$ is small,
- get obvious simple generator $P = (-d^2, c^2)$ of $E(\mathbb{F}_p)$,
- and point $P' = (-id, c) \in E'(\mathbb{F}_{p^2})$, that (almost) always gives a generator $Q' = [h]P'$ of $E'(\mathbb{F}_{p^2})[n]$, where $\#E'(\mathbb{F}_{p^2}) = hn$.

Implementation-friendly BN curves

Example curve:

$$u = -(2^{62} + 2^{55} + 1),\ c = 1,\ d = 1$$

Then

- $p \equiv 3 \pmod 4$,
- p has 254 bits,
- $6u + 2$ has NAF-weight 5,
- $E : y^2 = x^3 + 2$, $P = (-1, 1)$,
- $\xi = 1 + i$,
- $E' : y^2 = x^3 + (1 - i)$, $Q' = [h](-i, 1)$.

`http://eprint.iacr.org/2010/429`

Modular multiplication

Using the polynomial representation

- The pairing algorithm can be improved in all parts by improving arithmetic in $\mathbb{F}_p$.
- Can the polynomial shape

$$p = 36u^4 + 36u^3 + 24u^2 + 6u + 1$$

  be used to speed up multiplication modulo p?
- Fan, Vercauteren, Verbauwhede (CHES 2009) demonstrate this for hardware with $u = 2^l + s$, s small.
- What about software?

Using the polynomial representation

joint work with P. Schwabe and R. Niederhagen,
inspired by Dan Bernstein's Curve25519 paper

- Consider the ring $R = \mathbb{Z}[x] \cap \overline{\mathbb{Z}}[\sqrt{6}ux]$ and the element

$$\begin{aligned} P &= 36u^4x^4 + 36u^3x^3 + 24u^2x^2 + 6ux + 1 \\ &= (\sqrt{6}ux)^4 + \sqrt{6}(\sqrt{6}ux)^3 + 4(\sqrt{6}ux)^2 + \sqrt{6}(\sqrt{6}ux) + 1. \end{aligned}$$

  Then $P(1) = p$.

- Represent $f \in \mathbb{F}_p$ as a polynomial $F \in R$

$$\begin{aligned} F &= f_0 + f_1 \cdot \sqrt{6}(\sqrt{6}ux) + f_2 \cdot (\sqrt{6}ux)^2 + f_3 \cdot \sqrt{6}(\sqrt{6}ux)^3 \\ &= f_0 + f_1 \cdot (6u)x + f_2 \cdot (6u^2)x^2 + f_3 \cdot (36u^3)x^3 \end{aligned}$$

  such that $F(1) = f$.

- f corresponds to coefficient vector $[f_0, f_1, f_2, f_3],\ f_i \in \mathbb{Z}$.

Polynomial multiplication and degree reduction

- Polynomial multiplication of f and g gives polynomial with 7 coefficients.

$$\begin{aligned} f \cdot g &= h_0 + h_1 \cdot (6u)x + h_2 \cdot (6u^2)x^2 + h_3 \cdot (36u^3)x^3 \\ &+ h_4 \cdot (36u^4)x^4 + h_5 \cdot (216u^5)x^5 + h_6 \cdot (216u^6)x^6 \end{aligned}$$

- Reduce modulo P using $(36u^4)x^4 = -(36u^3)x^3 - 4(6u^2)x^2 - (6u)x - 1$.

$$\begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \end{bmatrix} \to \begin{bmatrix} h_0 \\ h_1 \\ h_2 - h_6 \\ h_3 - h_6 \\ h_4 - 4h_6 \\ h_5 - h_6 \\ 0 \end{bmatrix} \to \cdots \begin{bmatrix} h_0 - h_4 + 6h_5 - 2h_6 \\ h_1 - h_4 + 5h_5 - h_6 \\ h_2 - 4h_4 + 18h_5 - 3h_6 \\ h_3 - h_4 + 2h_5 + h_6 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Four coefficients are not enough

- 256-bit numbers in 4 coefficients: Each coefficient 64 bits, small multiples in the reduction are larger than 128 bits.
- Easy to realize in hardware, not in software, for software we need more coefficients.
- Idea: Consider $u = v^3$, use 12 coefficients $f_0, \dots, f_{11}$

$$\begin{aligned} f = & f_0 + 6vf_1x + 6v^2f_2x^2 + 6v^3f_3x^3 + 6v^4f_4x^4 \\ & + 6v^5f_5x^5 + 6v^6f_6x^6 + 36v^7f_7x^7 + 36v^8f_8x^8 \\ & + 36v^9f_9x^9 + 36v_{10}f_{10}x^{10} + 36v^{11}f_{11}x^{11}. \end{aligned}$$

  v has about 21 bits, product coefficients have about 42 bits.
- Double-precision floats have 53-bit mantissa.
- Use double-precision floats, still some space to add up coefficients and compute small multiples.

Reducing coefficients

- At some point the coefficients will *overflow* (become larger than 53 bits)
- Need to do coefficient reduction (carry)
- Carry from f_0 to f_1

  $c \leftarrow \text{round}(f_0/6v)$
  $f_0 \leftarrow f_0 - c \cdot 6v$
  $f_1 \leftarrow f_1 + c$
- Carry from f_1 to f_2

  $c \leftarrow \text{round}(f_1/v)$
  $f_1 \leftarrow f_1 - c \cdot v$
  $f_2 \leftarrow f_2 + c$
- $f_0 \in [-3v, 3v], f_1 \in [-v/2, v/2]$
- Carry from f_{11} goes to f_0, f_3, f_6, and f_9

Implementation on a Core 2 processor

- Use fast vector instructions `mulpd` and `addpd`, 2 multiplications/ 2 additions in one instruction, 1 `mulpd` and 1 `addpd` (and one `mov`) per cycle.
- Problem: $\mathbb{F}_p$ arithmetic requires a lot of shuffeling, combining etc., Solution: Implement arithmetic in $\mathbb{F}_{p^2}$.
- Use schoolbook multiplication in $\mathbb{F}_{p^2}$: 4 mults. in $\mathbb{F}_p$, squaring in $\mathbb{F}_{p^2}$: 2 multiplications in $\mathbb{F}_p$.
- Perform 2 $\mathbb{F}_p$ multiplications in parallel using vector instructions.
- Only two $\mathbb{F}_p$ polynomial reductions and two coefficient reductions per multiplication in $\mathbb{F}_{p^2}$ (also SIMD).
- To decide where to do a reduction, detect overflows, perform arithmetic on values and in parallel on worst-case values.

Performance results

- On an Intel Core 2 Quad Q6600 (65 nm): 4,134,643 cycles
- Comparison: Fastest published pairing benchmark (on one core) before: 10,000,000 cycles on a Core 2 by Hankerson, Menezes, Scott, 2008,
  Unpublished: 7,850,000 cyc on Core 2 T5500 (Scott 2010).
- New paper by Beuchat, González Díaz, Mitsunari, Okamoto, Rodríguez-Henríquez, and Teruya (Pairing 2010) claims: 2,330,000 cycles on a Core i7, 2,950,000 cycles on a Core 2 with Visual Studio 2008.

Cycle counts on a Core 2 Q6600 with gcc-4.3.3

	dclxvi	[BGM+10]
multiplication in $\mathbb{F}_{p^2}$	~ 585	~ 588
squaring in $\mathbb{F}_{p^2}$	~ 359	~ 487
optimal ate pairing	$\sim 4,135,000$	$\sim 3,269,000$

Why is our software slower?

[BGM+10] uses Montgomery arithmetic in $\mathbb{F}_p$ and fast 64×64-bit integer multiplier.

Three reasons why we are slower

1. Restricted choice of $u = v^3$: need more operations in $\mathbb{F}_{p^2}$.
2. Additional coefficient reductions take quite a bit of time.
3. Multiplication is not (much) faster.

Why is our multiplication not faster?

- Always need to perform even number of $\mathbb{F}_p$ multiplications, have to use schoolbook instead of Karatsuba in $\mathbb{F}_{p^2}$, 4 instead of 3 multiplications in $\mathbb{F}_p$.
- Using vector instructions still requires quite some shuffeling, overhead: 60 cycles per $\mathbb{F}_{p^2}$ multiplication.

But still...

- Fastest (current) implementation based on double-precision floating-point arithmetic,
- exploits special p,
- on Intel (and AMD) processors: integer-based approach (with Montgomery arithmetic) is faster
- But: several architectures have much faster double-precision floating-point than integer arithmetic.

Paper: `http://cryptojedi.org/users/peter/#dclxvi`
Software: `http://cryptojedi.org/crypto/#dclxvi`
(public domain)

Affine coordinates for pairings?

joint work with K. Lauter, P. Montgomery

- Choose coordinate system for elliptic curve point operations and line function computation,
- projective coordinates avoid inversions by doing more of the other operations.

Galbraith (2005): *"One can use projective coordinates for the operations in $E(\mathbb{F}_q)$. The performance analysis depends on the relative costs of inversion to multiplication in $\mathbb{F}_q$.... and experiments show that affine coordinates are faster."*

- Finite field inversion in prime field very expensive,
- for plain ECC over $\mathbb{F}_p$: projective always better,
- current speed records for pairings: projective formulas.

Extension field inversions

Quadratic extension:

- $\mathbb{F}_{q^2} = \mathbb{F}_q(\alpha)$ with $\alpha^2 = \omega \in \mathbb{F}_q^*$,
- $$\frac{1}{b_0 + b_1\alpha} = \frac{b_0 - b_1\alpha}{b_0^2 - b_1^2\omega} = \frac{b_0}{b_0^2 - b_1^2\omega} - \frac{b_1}{b_0^2 - b_1^2\omega}\alpha,$$
- $b_0^2 - b_1^2\omega = N(b_0 + b_1\alpha) \in \mathbb{F}_q$,
- compute inversion in $\mathbb{F}_{q^2}$ by inversion in $\mathbb{F}_q$ and some other operations

$$\mathbf{I}_{q^2} \leq \mathbf{I}_q + 2\mathbf{M}_q + 2\mathbf{S}_q + \mathbf{M}_{(\omega)} + \mathbf{sub}_q + \mathbf{neg}_q.$$

- Assume $\mathbf{M}_{q^2} \geq 3\mathbf{M}_q$ and get

$$\mathbf{R}_{q^2} = \mathbf{I}_{q^2}/\mathbf{M}_{q^2} \leq (\mathbf{I}_q/3\mathbf{M}_q) + 2 = \mathbf{R}_q/3 + 2.$$

Extension field inversions

Degree-ℓ extension:

- generalization of Itoh-Tsujii inversion,
- standard way to compute inverses in optimal extension fields,
- assume $\mathbb{F}_{q^\ell} = \mathbb{F}_q(\alpha)$ with $\alpha^\ell = \omega \in \mathbb{F}_q^*$
- with $v = (q^\ell - 1)/(q-1) = q^{\ell-1} + \cdots + q + 1$, compute

$$\beta^{-1} = \beta^{v-1} \cdot \beta^{-v},$$

- for $\beta \in \mathbb{F}_{q^\ell}$, $\beta^v = N(\beta) \in \mathbb{F}_q$.

$$\mathbf{R}_{q^\ell} \leq \mathbf{R}_q / M(\ell) + C(\ell)$$

ℓ	2	3	4	5	6	7
$1/M(\ell)$	$1/3$	$1/6$	$1/9$	$1/13$	$1/17$	$1/22$
$C(\ell)$	3.33	4.17	5.33	5.08	6.24	6.05

Simultaneous inversions

Montgomery's n-th trick...

- Idea: To invert a and b, compute ab, then $(ab)^{-1}$ and

$$a^{-1} = b \cdot (ab)^{-1}, \quad b^{-1} = a \cdot (ab)^{-1},$$

  replace $2\mathbf{I}$ by $1\mathbf{I} + 3\mathbf{M}$.

- In general for s inversions at once: compute $c_i = a_1 \cdot \cdots \cdot a_i$ for $2 \leq i \leq s$, then c_s^{-1} and

$$c_{s-1}^{-1} = c_s^{-1} a_s, \quad a_{s-1}^{-1} = c_{s-2} c_{s-1}^{-1}, \quad \ldots$$

  replace $s\mathbf{I}$ by $1\mathbf{I} + 3(s-1)\mathbf{M}$.

- Average $\mathbf{I}/\mathbf{M}$ is

$$(s\mathbf{I})/(s\mathbf{M}) = \mathbf{I}/(s\mathbf{M}) + 3(s-1)/s \leq \mathbf{R}/s + 3.$$

Affine coordinates for pairings

Affine coordinates can be better than projective

- if the used implementation has small $\mathbf{I}/\mathbf{M}$,
- for ate pairings on curves with larger embedding degree, i.e. at high security levels (the ate pairing needs arithmetic in $E'(\mathbb{F}_{q^{k/d}})$, $\mathbf{I}/\mathbf{M}$ gets smaller in larger extensions),
- when high-degree twists are not being used (s.t. k/d is large),
- for computing several pairings (or products of several pairings) at once on independent point pairs.

Pairings based on Microsoft's bignum

optimal ate pairing on a 256-bit BN curve

Use MS bignum for

- base field arithmetic ($\mathbb{F}_p$) with Montgomery multiplication,
- 256-bit integers are split into 4 pieces of 64 bits,
- extension fields based on MS bignum field extensions, with inversions based on norm trick.

MS bignum + pairings

- is a C implementation (w/ little bit of assembly for mod mul on AMD64),
- not restricted to specific security level, curves, or processors,
- works under 32-bit and 64-bit Windows.

Pairings based on Microsoft's bignum

field arithmetic performance

Fields over 256-bit BN prime field with

- $p \equiv 3 \pmod 4$, i.e. $\mathbb{F}_{p^2} = \mathbb{F}_p(i)$, $i^2 = -1$.

Timings on a 3.16 GHz Intel Core 2 Duo E8500, 64-bit Windows 7

	$\mathbf{M}$		$\mathbf{S}$		$\mathbf{I}$		$\mathbf{I/M}$
	cyc	μs	cyc	μs	cyc	μs	
$\mathbb{F}_p$	414	0.13	414	0.13	9469	2.98	22.87
$\mathbb{F}_{p^2}$	2122	0.67	1328	0.42	11426	3.65	5.38
$\mathbb{F}_{p^6}$	18544	5.81	12929	4.05	40201	12.66	2.17
$\mathbb{F}_{p^{12}}$	60967	19.17	43081	13.57	103659	32.88	1.70

Pairings based on Microsoft's bignum

Pairings on a 256-bit BN curve with

- sparse parameter u (HW 7), sparse $6u + 2$ (HW 8).

Timings on a 3.16 GHz Intel Core 2 Duo E8500,
64-bit Windows 7

operation	CPU cycles	time
Miller loop	7,572,000	2.36 ms
optimal ate pairing	14,838,000	4.64 ms
20 opt. ate at once	14,443,000	4.53 ms
product of 20 opt. ate	4,833,000	1.52 ms
EC scalar mult in G_1	2,071,000	0.64 ms
EC scalar mult in G_2	8,761,000	2.74 ms

`http://eprint.iacr.org/2010/363`