Efficient pairing computation at the 192-bit and 256-bit security levels

Craig Costello

Technische Universiteit Eindhoven

October 30, 2012

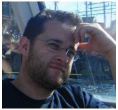
ECC2012 - Querétaro, Mexico

Contains joint work with. . . ...

Colin Boyd

Juanma Gonzalez Nieto

Kenneth Wong



Huseyin Hisil

Tanja Lange

Michael Naehrig

Kristin Lauter

Douglas Stebila

A brief history of pairing speeds...

- **1993**
  Menezes — **a few minutes**

  ⋮ ⋮

- **2002**
  Barreto-Kim-Lynn-Scott (BKLS) — **30-60ms**

  Galbraith-Harrison-Soldera (GHS) ⋮

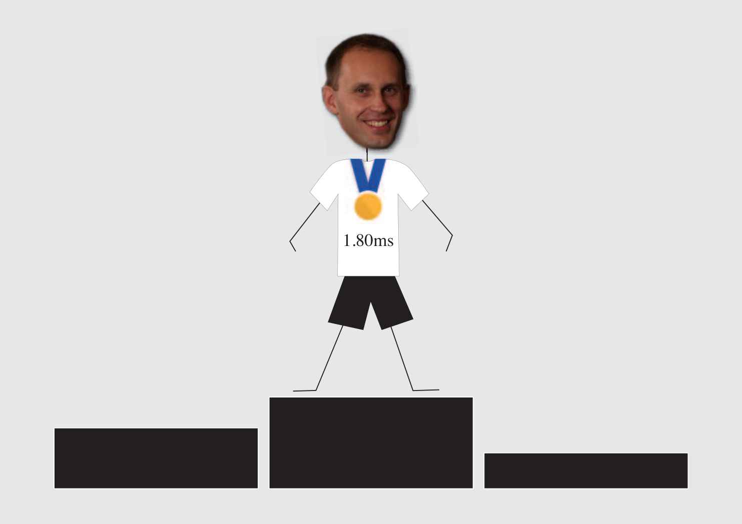
  ⋮ ⋮

- **2008**
  Hankerson-Menezes-Scott — **14.2ms**

  ⋮ ⋮

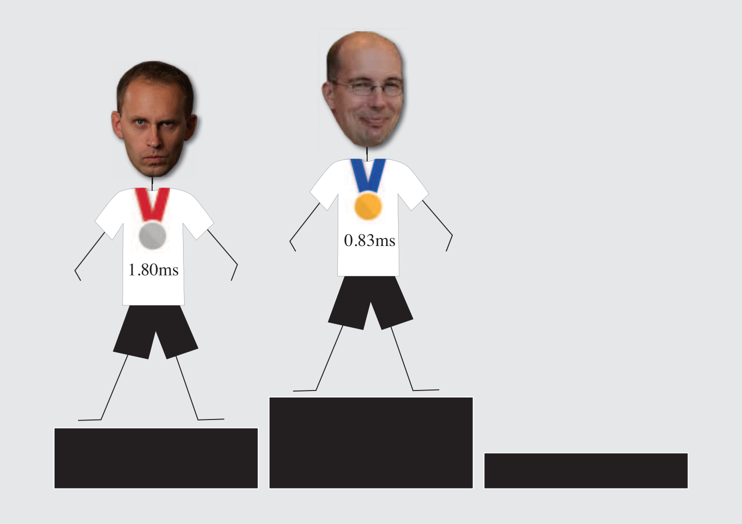
The 128-bit records...

6^{th} April 2010: Naehrig *et al.*: $4,300,000$ cycles



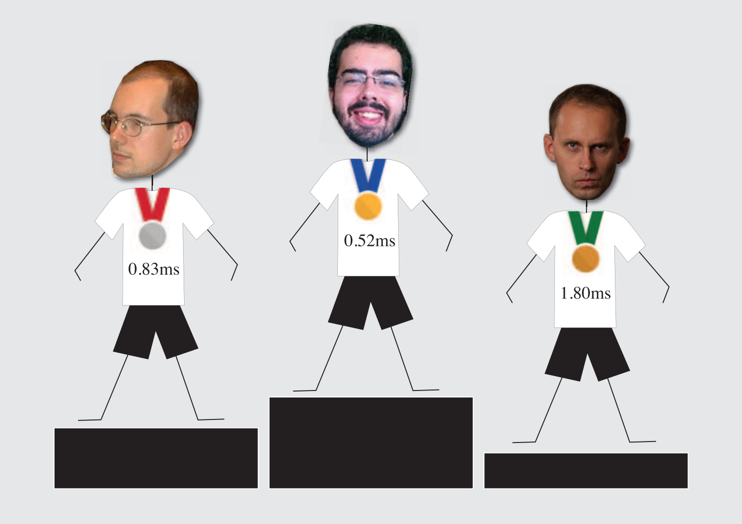
The 128-bit records...

17^{th} June 2010: Beuchat *et al.*: $2,600,000$ cycles



The 128-bit records...

13^{th} October 2010: Aranha *et al.*: $1,600,000$ cycles



Why high security?...

How safe are pairings at the 128-bit level? (www.keylength.com)

- **ECRYPT II** - 2011 safe until 2040
- **NIST (USA)** - 2011 safe until at least 2030
- **FNISA (France)** -2010 safe until at least 2020

Why high security?...

How safe are pairings at the 128-bit level? (www.keylength.com)

- **ECRYPT II** - 2011 — safe until 2040
- **NIST (USA)** - 2011 — safe until at least 2030
- **FNISA (France)** -2010 — safe until at least 2020

SOME REASONS ...

- Some governments, militaries are more paranoid than others
- Let history be our guide: we often underestimate ourselves (cf. Takuya's talk tomorrow)
- Much more fun to be had - interesting things happen beyond 128-bit pairings

Overview

1. Bilinearity and pairing-friendly curves
2. How to compute pairings
3. Optimisations
4. Pairings at the 256-bit security level (and beyond)
5. Pairings at the 192-bit security level
6. Work in progress

1. Bilinearity and pairing-friendly curves

Cryptographic pairings and bilinearity

- A cryptographic pairing on an elliptic curve $E/\mathbb{F}_q$ is a **bilinear** map

$$\begin{array}{lllll} e: & \mathbb{G}_1 & \times & \mathbb{G}_2 & \rightarrow & \mathbb{G}_T \\ e: & E[r] & \times & E[r] & \rightarrow & \mu_r \\ e: & P & \times & Q & \mapsto & e(P,Q) \end{array}$$

- **Bilinear** means

$$e(P+P',Q) = e(P,Q)\cdot e(P',Q),$$
$$e(P,Q+Q') = e(P,Q)\cdot e(P,Q'),$$

from which it follows that, for scalars $a, b \in \mathbb{Z}$, we have

$$e([a]P,[b]Q) = e(P,[b]Q)^a = e([a]P,Q)^b = e(P,Q)^{ab} = e([b]P,[a]Q).$$

We need more torsion!

- $\mathbb{G}_1 \in E[r]$ **and** $\mathbb{G}_2 \in E[r]$ **must be linearly independent**
- Cases of interest: we want r to be as close to $\#E(\mathbb{F}_q)$ as possible, so think $r \approx q$
- Only one order-r subgroup in $E(\mathbb{F}_q)$
- To find another linearly independent torsion subgroup, we must extend $\mathbb{F}_q$, but how far do we need to go?

The embedding degree k

- **Balasubramanian and Koblitz** told us exactly how far we need to extend $\mathbb{F}_q$ to find more torsion: namely, to $\mathbb{F}_{q^k}$ where:

The embedding degree k

The embedding degree is the smallest $k \in \mathbb{Z}^+$ such that $r \mid q^k - 1$.

- Once we find one more torsion point in $E(\mathbb{F}_{q^k})$, we find all r^2 points in $E(\overline{\mathbb{F}}_q)[r] \subseteq E(\mathbb{F}_{q^k})$
- $\mathbb{F}_{q^k}$ is also where we find μ_r
- So $\mathbb{F}_{q^k}$ is where the computations take place

$$\begin{array}{llllll} e: & E[r] & \times & E[r] & \to & \mu_r \\ e: & E(\mathbb{F}_{q^k}) & \times & E(\mathbb{F}_{q^k}) & \to & \mu_r \in \mathbb{F}_{q^k} \end{array}$$

- **We need k to be small, i.e. $k < 50$**

Pairing-friendly curves

Definition: E is a pairing-friendly curve if...

- k is small (less than 50)
- the prime r dividing $\#E$ has $r \geq \sqrt{q}$

- Hasse-Bound: group order can lie anywhere between $q + 1 - \lfloor 2\sqrt{q} \rfloor$ and $q + 1 + \lfloor 2\sqrt{q} \rfloor$

$$\begin{aligned} &115792089210356248762697446949407573529405578681527665431107311373540212604928 \\ q = &115792089210356248762697446949407573530086143415290314195533631308867097853951 \\ &115792089210356248762697446949407573530766708149052962959959951244193983102976 \end{aligned}$$

- ... then think of r and q as independent of each other (from half way down)
- k being small enough is extremely unlikely in general
- **Moral of the story: pairing-friendly curves are very rare!**

Polynomial parameterisations of pairing-friendly curves

- We need to find q, r, t such that there exists $E/\mathbb{F}_q$ with

$$r \mid \#E(\mathbb{F}_q) = q + 1 - t \qquad \textit{and} \qquad r \mid q^k - 1$$

  for some small k

- Miyaji-Nakabayashi-Takano (MNT): $r \mid q^k - 1$ implies $r \mid \Phi_k(q) \ldots$
- Barreto-Lynn-Scott (BLS): $r \mid \Phi_k(q)$ and $r \mid q + 1 - t$ together imply $r \mid \Phi_k(t - 1)$, so use this instead

General strategy for finding parameterised families

Fix k small, then search for $t(x)$ and $r(x)$ such that

$$r(x) \mid \Phi_k(t(x) - 1) \qquad \text{and} \qquad r(x) \mid q(x) + 1 - t(x)$$

- This strategy has been most successful in finding pairing-friendly curves ...

The Barreto-Naehrig (BN) family

- For $k = 12$, $\Phi_{12}(z) = z^4 - z^2 + 1$
- Setting $t(x) = 6x^2 + 1$, gives $\Phi_{12}(t(x) - 1) = r(x)r(-x)$ with
$$r(x) = 36x^4 + 36x^3 + 18x^2 + 6x + 1$$
- Set $\#E = r(x)$ and then $q(x) = r(x) - 1 + t(x)$, so
$$q(x) = 36x^4 + 36x^3 + 18x^2 + 6x + 1$$
- Search x's (appropriately sized) for $r(x)$ and $q(x)$ both prime
- Guaranteed curve always of form $E/\mathbb{F}_q : y^2 = x^3 + b$ (no CM needed)
- Guaranteed $k = 12$, so pairing computation takes place over $\mathbb{F}_{q^{12}}$

The ρ-value

- For example, a BN curve is found with $x = 448873741399$, where

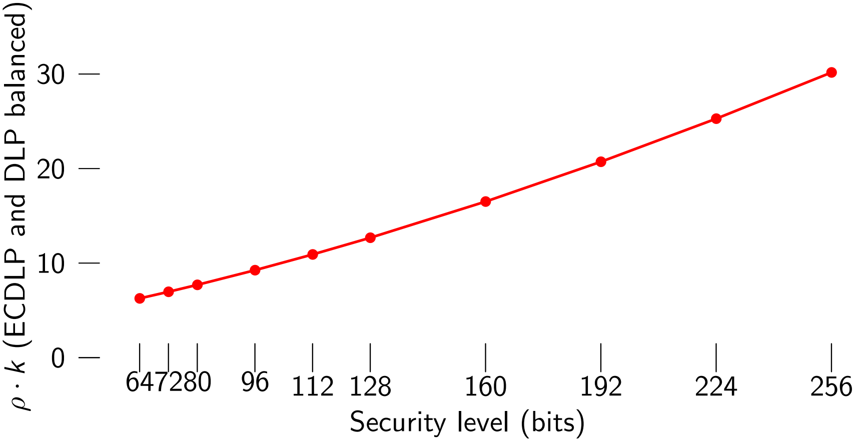
$$q(x) = 1461501624496790265145448589920785493717258890819$$
$$r(x) = 1461501624496790265145447380994971188499300027613$$

- The situation here is ideal because $\log_2(q) \approx \log_2(r)$
- We know this happens for large x because $q(x)$ and $r(x)$ have the same degree
- The ρ-value tells us the ratio between the sizes of q and r

$$\rho = \frac{\log(\deg(q(x)))}{\log(\deg(r(x)))}$$

- So $\rho \cdot k$ gives us the ratio between $\mathbb{F}_{q^k}$ (where DLP lies) and r (where ECDLP lies)

Balancing ECDLP and DLP security

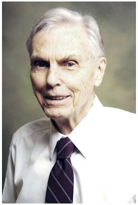


- So to target a particular security level, we consider families whose $\rho \cdot k$ values are close to optimal

2. How to compute pairings

The Weil and Tate pairings of $P, Q \in E[r]$

André Weil



John Tate

- Define the divisors $D_P \sim (P) - (\mathcal{O})$ and $D_Q \sim (Q) - (\mathcal{O})$
- Let $f_{r,P}$ be the (unique up to constant) function with divisor

$$(f_{r,P}) = r(P) - r(\mathcal{O})$$

Weil pairing (in crypto): $e(P, Q) = \frac{f_{r,P}(Q)}{f_{r,Q}(P)}$

Tate pairing (in crypto): $e(P, Q) = f_{r,P}(Q)^{(q^k - 1)/r}$

The function $f_{r,P}(Q)$ is huuuuuge!

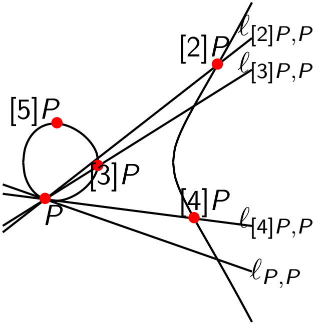
The size of $f_{r,P}(Q)$ for 128-bit security

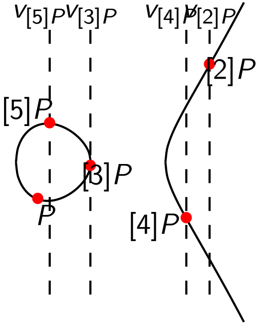
- The pairing function $f_{r,P}(Q)$ is of degree r, where
  $r = 16798108731015832284940804142231733909759579603404752749028378864165570215949$
- The coefficients in $f_{r,P}(Q)$ depend on P's coordinates, so are all of the size
  $P_x = 15283023184232661393336451140837190640382743162584629974443682653991135323854$
- This huge function is impossible to store with all the computing power in the world. Somehow we need to evaluate it at D_Q, where Q's x coordinate is

$$
\begin{aligned}
Q_x = {} & ((15550921060303536733405227206218421303411153835059642979852113370177068459559 \cdot u + \\
& 36006906447969872904421351370312852062497895145888276790029208075555440045456) \cdot v^2 + \\
& (5475264847170057761513968927972623766794030526092071182289628553939256498415 \cdot u + \\
& 16045231392378269041781500461472571507692250280489500368315808811462293278705) \cdot v + \\
& (13578969743206791049626159973437892548805434308942546900125761281664803554809 \cdot u + \\
& 8414705805435201691796063348962631501393112240468038251361145485591996962517)) \cdot w + \\
& (2095760324718272519234982374519336043146898698412090865684809945855004557738 \cdot u + \\
& 10991749562144480578133596744105999544930359103290000221828602811069330922292) \cdot v^2 + \\
& (563526440913857199739302175501170867491400605855901007410492904987821568516 \cdot u + \\
& 12175465566401923735806619064706225201231722038674162959277121785143969709483) \cdot v + \\
& 5977392629488041467394421854470109162392545860735885669496575455742917555185 \cdot u + \\
& 16414735455238441715243107544357668247548687753217062857281803216595664241398
\end{aligned}
$$

- Even bigger for higher security levels!

A naive (pre-Miller) pairing





$$
\begin{aligned}
\ell_{P,P}/v_{[2]P}: &\quad (P)+(P)-([2]P)-(\mathcal{O})\\
\ell_{[2]P,P}/v_{[3]P}: &\quad (P)+([2]P)-([3]P)-(\mathcal{O})\\
\ell_{[3]P,P}/v_{[4]P}: &\quad (P)+([3]P)-([4]P)-(\mathcal{O})\\
\vdots\quad &\quad \vdots\quad\ \vdots\qquad\quad \vdots\qquad\quad \vdots\\
\ell_{[r-2]P,P}/v_{[r-1]P} &\quad (P)+([r-2]P)-([r-1]P)-(\mathcal{O})\\
\ell_{[r-1]P,P}/v_{[r]P} &\quad (P)+([r-1]P)-([r]P)-(\mathcal{O})
\end{aligned}
$$

$$\left(\prod \ell/v'\mathrm{s}\right) = r(P) - r(\mathcal{O})$$

Miller observed . . .

- At any intermediate stage of the "naive" algorithm, we have a function $f_{m,P}$ with divisor

$$(f_{m,P}) = m(P) - ([m]P) - (m-1)(\mathcal{O})$$

- Squaring the function doubles the number of zeros and poles

$$\begin{tikzcd}
f_{m,P} \arrow[r, "\cdot\frac{\ell_{[m]P,P}}{v_{[m+1]P}}"] \arrow[rrrrr, bend right, "f_{m,P}^2\cdot\frac{\ell_{[m]P,[m]P}}{v_{[2m]P}}"'] & f_{m+1,P} \arrow[r, "\cdot\frac{\ell_{[m+1]P,P}}{v_{[m+2]P}}"] & \cdots & \cdots \arrow[r, "\cdot\frac{\ell_{[2m-2]P,P}}{v_{[2m-1]P}}"] & f_{2m-1,P} \arrow[r, "\cdot\frac{\ell_{[2m-1]P,P}}{v_{[2m]P}}"] & f_{2m,P}
\end{tikzcd}$$

- **We can get from $f_{m,P}$ to $f_{2m,P}$ in one step**

Miller's algorithm for $f_{r,P}(D_Q)$

$r = (r_{l-1}, \ldots, r_1, r_0)_2$ and initialize: $R = P$, $f = 1$

for $i = l - 2$ to 0 do

a. i. Compute ℓ/v in the doubling of R

ii. $R \leftarrow [2]R$ //(DBL)

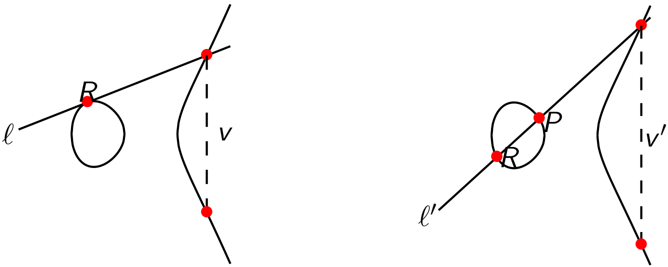
iii. $f \leftarrow f^2 \cdot \ell/v(D_Q)$

b. if $r_i = 1$ then

i. Compute ℓ'/v' in the addition of $R + P$

ii. $R \leftarrow R + P$ //(ADD)

iii. $f \leftarrow f \cdot \ell'/v'(D_Q)$



3. Optimisations

Miller 2.0: BKLS-GHS algorithm for the Tate pairing

$r = (r_{l-1}, \ldots, r_1, r_0)_2$ and initialize: $R = P$, $f = 1$
for $i = l - 2$ to 0 do

1. //(Miller loop)
   a. i. Compute ℓ/v in the doubling of R
      ii. $R \leftarrow [2]R$
      iii. $f \leftarrow f^2 \cdot \ell/v(Q)$
   b. if $r_i = 1$ then
      i. Compute ℓ'/v' in the addition of $R + P$
      ii. $R \leftarrow R + P$
      iii. $f \leftarrow f \cdot \ell'/v'(Q)$
2. $f \leftarrow f^{(q^k-1)/r}$ (final exponentiation)

 *et al.* &  *et al.*: **Never mind D_Q, use Q!**

Miller 2.0: BKLS-GHS algorithm for the Tate pairing

$r = (r_{l-1}, \ldots, r_1, r_0)_2$ and initialize: $R = P$, $f = 1$

for $i = l - 2$ to 0 do

1. //(Miller loop)
   a. i. Compute ℓ in the doubling of R
      ii. $R \leftarrow [2]R$
      iii. $f \leftarrow f^2 \cdot \ell(Q)$
   b. if $r_i = 1$ then
      i. Compute ℓ' in the addition of $R + P$
      ii. $R \leftarrow R + P$
      iii. $f \leftarrow f \cdot \ell'(Q)$
2. $f \leftarrow f^{(q^k-1)/r}$ (final exponentiation)

 *et al.* &  *et al.*: **Can do without $v(Q)$!**

Miller 2.0: BKLS-GHS algorithm for the Tate pairing

$r = (r_{l-1}, \ldots, r_1, r_0)_2$ and initialize: $R = P$, $f = 1$

for $i = l - 2$ to 0 do

1. //(Miller loop)
   a. i. Compute ℓ in the (projective) doubling of R
      ii. $R \leftarrow [2]R$
      iii. $f \leftarrow f^2 \cdot \ell(Q)$
   b. if $r_i = 1$ then
      i. Compute ℓ' in the (projective) addition of $R + P$
      ii. $R \leftarrow R + P$
      iii. $f \leftarrow f \cdot \ell'(Q)$
2. $f \leftarrow f^{(q^k-1)/r}$ (final exponentiation)

 *et al.*,  *et al.*: **Avoid inversions altogether!**

Miller 2.0: BKLS-GHS algorithm for the Tate pairing

$r = (r_{l-1}, \ldots, r_1, r_0)_2$ and initialize: $R = P$, $f = 1$
for $i = l - 2$ to 0 do

1. //(Miller loop)
   a. i. Compute ℓ in the projective doubling of R
      ii. $R \leftarrow [2]R$
      iii. $f \leftarrow f^2 \cdot \ell(Q)$
   b. if $r_i = 1$ then
      i. Compute ℓ' in the projective addition of $R + P$
      ii. $R \leftarrow R + P$
      iii. $f \leftarrow f \cdot \ell'(Q)$
2. $f \leftarrow f^{(q^k - 1)/r}$ (final exponentiation)

r=411878056433041014834992998411348078 81

r=[1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1,
0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0,
1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1]

Miller 2.0: BKLS-GHS algorithm for the Tate pairing

$r = (r_{l-1}, \ldots, r_1, r_0)_2$ and initialize: $R = P$, $f = 1$

for $i = l - 2$ to 0 do

1. //(Miller loop)
   a.
      i. Compute ℓ in the projective doubling of R
      ii. $R \leftarrow [2]R$
      iii. $f \leftarrow f^2 \cdot \ell(Q)$

   b. if $r_i = 1$ (unlikely) then
      i. Compute ℓ'' in the projective addition of $R + P$
      ii. $R \leftarrow R + P$
      iii. $f \leftarrow f \cdot \ell'(Q)$
2. $f \leftarrow f^{(q^k-1)/r}$ (final exponentiation)

r=170141183460469231731687303715884105793 (low hamming-weight)

r=[1, 0, 0, 0, 0, 0,1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

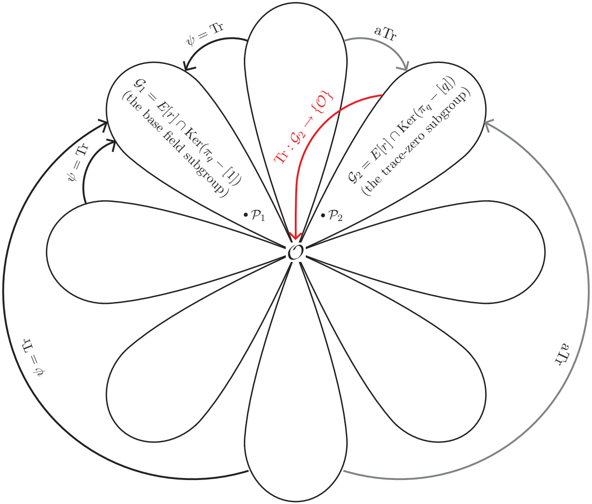
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]

Torsion subgroups, twisted curves, and Type 3 pairings

- Recall that once we extend up to $\mathbb{F}_{q^k}$, we collect all r^2 points in the r-torsion

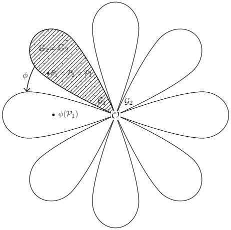
- They must form $(r+1)$ cyclic subgroups of order r ...

Defining $\mathbb{G}_1$ and $\mathbb{G}_2$ in the torsion $E[r] \cong \mathbb{Z}_r \times \mathbb{Z}_r$

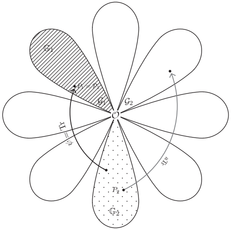


- **Galbraith-Paterson-Smart (Shacham)**: 4 types of pairings depending on our placement of $\mathbb{G}_2$

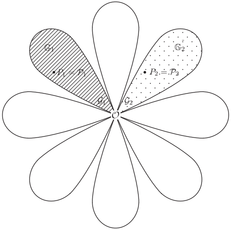
Pairing types



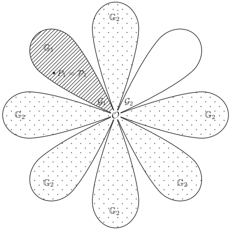
Type 1: supersingular only $k \le 6$



Type 2: can't keep hashing



Type 3: no $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$



Type 4: $\mathbb{G}_2$ not cyclic

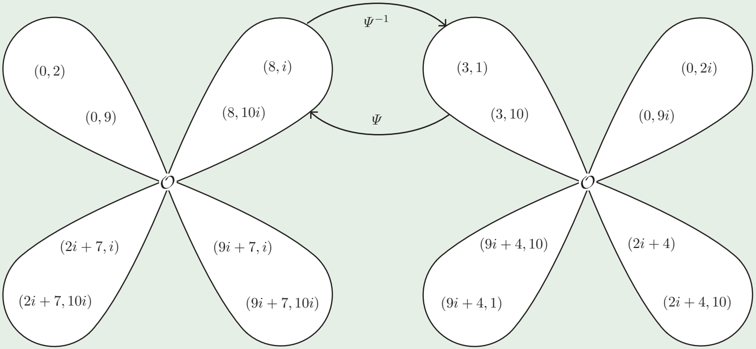
The twisted curve

- There is an efficiently computable isomorphism from the trace-zero subgroup $\mathbb{G}_2 \in E[r]$ to the "base-field" subgroup of its twist $E'/\mathbb{F}_{q^{k/d}}$

The twisted curve

- There is an efficiently computable isomorphism from the trace-zero subgroup $\mathbb{G}_2 \in E[r]$ to the "base-field" subgroup of its twist $E'/\mathbb{F}_{q^{k/d}}$

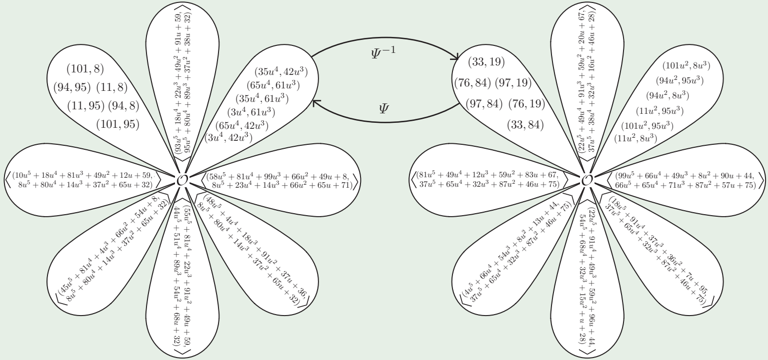
e.g. $E/\mathbb{F}_{11} : y^2 = x^3 + 4$, $\#E(\mathbb{F}_{11}) = 12$, so $r = 3$ with $k = 2$ and $d = 2$, i.e. $E'/\mathbb{F}_{11} : y^2 = x^3 - 4$



The twisted curve

- A twist of degree d means E' is defined over $\mathbb{F}_{q^{k/d}}$.
- For elliptic curves we can have $d \in \{2, 3, 4, 6\}$

e.g. $E/\mathbb{F}_{103} : y^2 = x^3 + 72,\ \#E(\mathbb{F}_{103}) = 84,\ r = 7,\ k = 6,\ d = 6$



- **Warning**: can work with $Q' = \Psi^{-1}(Q) \in E'(\mathbb{F}_{q^{k/d}})$, but must move back at function evaluation time so pairing is in $\mathbb{F}_{q^k}$

High-degree twists: $d = 3, 4, 6$

- $d = 2$ quadratic twists always available (when $2 \mid k$), but higher twists need special curves
  - $d = 3, 6$ need $y^2 = x^3 + b$ (i.e. $j(E) = 0$ or $D = -3$)
  - $d = 4$ need $y^2 = x^3 + ax$ (i.e. $j(E) = 1728$ or $D = -1$)
- **Fortunately all of the best parameterised families give curves of the shape we want** (e.g. $k = 12$ BN had $y^2 = x^3 + b$)
- We also prefer $k = 2^i \cdot 3^j$ because ...

Towered extension field arithmetic

Koblitz-Menezes '05

- For $k = 2^i 3^j$, build extension field as a sequence of quadratic and cubic subextensions (preferably binomials)
  - Karatsuba-like tricks make arithmetic much faster
  - easier to implement and twisted subfields constructed inherently
- e.g. a $k = 12$ tower

$$\mathbb{F}_q \xrightarrow{\beta^2-\alpha} \mathbb{F}_{q^2} \xrightarrow{\gamma^3-\beta} \mathbb{F}_{q^6} \xrightarrow{\delta^2-\gamma} \mathbb{F}_{q^{12}}.$$

- Instead of $\mathbb{F}_{q^{12}}$ multiplications costing 144 $\mathbb{F}_q$ multiplications, they cost $3 \cdot 3 \cdot 6 = 54$ $\mathbb{F}_q$ multiplications
- **Finding a nice tower is not always possible**

Straight to ate (sorry η_T): Miller 3.0



Hess - Smart - Vercauteren

- In $\mathbb{G}_2$ (the trace-zero subgroup), we have $\pi(Q) = [q]Q$
- Frobenius acts non-trivially and stays within $\mathbb{G}_2$
- Use this to define a Tate-like pairing, but with a shorter loop

$$a_T(Q, P) = f_{T,Q}(P)^{(q^k-1)/r}$$

- Note Q and P have switched roles, so most of the work in Miller's algorithm is done in the extension field $\mathbb{F}_{q^k}$
- But we use the twist to pull computation down to $\mathbb{F}_{q^{k/d}}$
- Trade-off very favourable when $T \ll r$ and $d = 4, 6$

Vercauteren's optimal ate pairing: Miller 3.1

- **Vercauteren:** ate pairing a_T is just a special case of a more general pairing

  $$a_{\lambda_i}(Q, P) = f_{\lambda_i, Q}(P)^{(q^k-1)/r}$$

  where $\lambda_i = q^i \bmod r$.
- We want smallest λ_i (loop length) possible
- Can do even better: find linear combination $\sum_{i=0}^{l-1} c_i \lambda_i \equiv 0 \bmod r$, where c_i are all short, then

  $$(Q, P) \mapsto \prod_{i=0}^{l} f_{c_i, Q}(P) \cdot \prod_{i=0}^{l} \ell_i$$

  defines a bilinear pairing, where the ℓ_i are all simple "one-off" line functions
- Vercauteren proves: $\max\{c_i\} \leq r^{1/\varphi(k)}$
- **Optimal pairing**: loop length at most $\log_2 r/\varphi(k) + \epsilon$
- Parameterised families make it easy to satisfy this bound (one $c_i(x)$)

Miller 3.1: optimal ate pairing

$m = (m_{l-1}, \ldots, m_1, m_0)_2$ and initialize: $R' = Q' = \Psi^{-1}(Q)$,

$f = 1$

for $i = l - 2$ to 0 do

1. //(Miller loop)
   a. i. Compute ℓ in the projective doubling of R'
      ii. $R' \leftarrow [2]R'$
      iii. $f \leftarrow f^2 \cdot \ell(P)$ (untwist Q')
   b. if $m_i = 1$ then
      i. Compute ℓ'' in the projective addition of $R + P$
      ii. $R' \leftarrow R' + Q'$
      iii. $f \leftarrow f \cdot \ell'(P)$ (untwist Q')
2. $f \leftarrow f^{(q^k-1)/r}$ (final exponentiation)

BN $k = 12$ curves have $\varphi(k) = 4$

Instead of looping to $r(x) = 36x^4 + 36x^3 + 18x^2 + 6x + 1$, we loop to $c(x) = 6x + 2$, e.g. our loop changes from
$r = 1461501624496790265145447380994971188499300027613$
to $m = 448873741399$

Fast explicit formulas

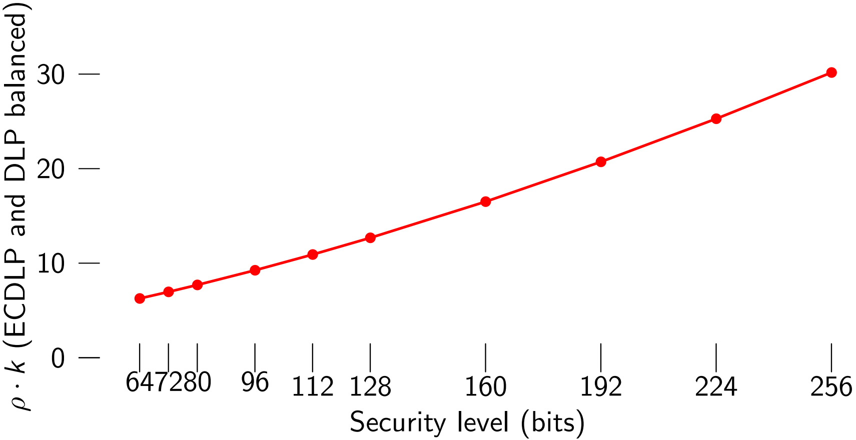
- C-Lange-Naehrig PKC2010: presented fastest explicit formulas for doing the group operation and line computation
- All practical scenarios covered

Curve Curve order Twist deg.	**Record**	DBL ADD mADD	**Prev. Record**	DBL ADD mADD
$y^2 = x^3 + ax$ - $d = 2, 4$	C-Lange-Naehrig'10 $\mathcal{W}_{(1,2)}$	$2\mathbf{m} + 8\mathbf{s}$ $12\mathbf{m} + 7\mathbf{s}$ $9\mathbf{m} + 5\mathbf{s}$	Ionica-Joux Arene *et al.* $\mathcal{J}$	$1\mathbf{m} + 11\mathbf{s}$ $10\mathbf{m} + 6\mathbf{s}$ $7\mathbf{m} + 6\mathbf{s}$
$y^2 = x^3 + c^2$ $3 \mid \#E$ $d = 2, 6$	C-Hisil-Boyd-Gonzalez Nieto-Wong'09 $\mathcal{P}$	$3\mathbf{m} + 5\mathbf{s}$ $14\mathbf{m} + 2\mathbf{s}$ $10\mathbf{m} + 2\mathbf{s}$	Arene *et al.* $\mathcal{P}$	$3\mathbf{m} + 8\mathbf{s}$ $10\mathbf{m} + 6\mathbf{s}$ $7\mathbf{m} + 6\mathbf{s}$
$y^2 = x^3 + b$ $3 \nmid \#E$ $d = 2, 6$	C-Lange-Naehrig'10 $\mathcal{P}$	$2\mathbf{m} + 7\mathbf{s}$ $14\mathbf{m} + 2\mathbf{s}$ $10\mathbf{m} + 2\mathbf{s}$	Arene *et al.* $\mathcal{J}$	$3\mathbf{m} + 8\mathbf{s}$ $10\mathbf{m} + 6\mathbf{s}$ $7\mathbf{m} + 6\mathbf{s}$
$y^2 = x^3 + b$ - $d = 3$	C-Lange-Naehrig'10 $\mathcal{P}$	$6\mathbf{m} + 7\mathbf{s}$ $16\mathbf{m} + 3\mathbf{s}$ $13\mathbf{m} + 3\mathbf{s}$	El Mrabet *et al.* $\mathcal{P}$	$8\mathbf{m} + 9\mathbf{s}$ *ADD/mADD* *not reported*

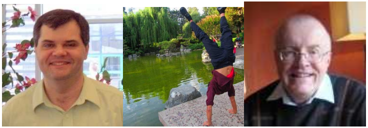
- Note: improvements/adjusements have since been made in various scenarios - Aranha *et al.* tweaked our formulas in their record-breaking paper

4. Pairings at the 256-bit security level (and beyond)

256-bit security wants $\rho \cdot k = 30$



The BLS family with $k = 24$



Barreto Lynn Scott

- e.g. Barreto-Lynn-Scott (among many other contributions) gave curves with $k = 24$:

$$q(x) = (x-1)^2(x^8 - x^4 + 1)/3 + x;$$
$$n(x) = (x-1)^2(x^8 - x^4 + 1)/3;$$
$$r(x) = x^8 - x^4 + 1; \quad t(x) = x + 1$$

- when $q = q(x)$, $r = r(x)$ are prime, guaranteed a curve $E/\mathbb{F}_q : y^2 = x^3 + b$ with $r \mid n = \#E$.
- Notice $\rho = 1.25$, so $\rho \cdot k = 30$ (nice!)
- Note that $\deg(t(x)) = \deg(r(x))/\varphi(k)$, so ate pairing is already optimal - loop length is x (nice)

Example: searching for BLS curves

$$q(x) = (x-1)^2(x^8 - x^4 + 1)/3 + x; \qquad r(x) = x^8 - x^4 + 1.$$

- Kick-start with $x = 2^{64} = 18446744073709551616$ (targeting 256-bit security): $x \equiv 1 \bmod 3$, $x \leftarrow x + 3$
- soon enough $x = 18446744073709563373$
  $q =$
  152081353920740809892727066524584946339781036330218959281772305234001103872205207355200355585054305961029358887567446121016058918174051639618221302567689792185243234190430804646778679690996022 1
- soon after $x = 18446744073709568134$
  $q =$
  1520813539207412024060742043441878459074161652061485145425476810606768714457121717514068260675858946726622675208621738650395266513452695828995492519266950330867144614888025492087559518474496777
- **moral: thousands/millions/billions... of possible curves to choose from... some of them are much better than others!**

Attractive subfamilies of BLS curves for high-security

C-Lauter-Naehrig'11

Instead of $x \equiv 1 \bmod 3$

x mod 72	$p(x)$ mod 72	$n(x)$ mod 72	efficient tower	curve E	correct twist E'
7	19	12	✓	$y^2 = x^3 + 1$	$y^2 = x^3 \pm 1/v$
16	19	3	✓	$y^2 = x^3 + 4$	$y^2 = x^3 \pm 4v$
31	43	12	✓	$y^2 = x^3 + 1$	$y^2 = x^3 \pm v$
64	19	27	✓	$y^2 = x^3 - 2$	$y^2 = x^3 \pm 2/v$

Can always tower with any of . . .

$$
\begin{tikzcd}
 & & & \mathbb{F}_{q^8} \arrow[r, "z^3+w"] & \mathbb{F}_{q^{24}}; \\
\mathbb{F}_q \arrow[r, "u^2+1"] & \mathbb{F}_{q^2} \arrow[r, "v^2+u+1"] & \mathbb{F}_{q^4} \arrow[ur, "w^2+v"] \arrow[rr, "z^6+v"] \arrow[dr, "w^3+v"'] & & \mathbb{F}_{q^{24}}; \\
 & & & \mathbb{F}_{q^{12}} \arrow[r, "z^2+w"'] & \mathbb{F}_{q^{24}};
\end{tikzcd}
$$

Twist type: M vs. D

- For quartic and sextic twists, there are actually two possibilities for the twist - only one has $r \mid \#E(\mathbb{F}_{q^{k/d}})$ - this is the one we want
- e.g. For $E/\mathbb{F}_q : y^2 = x^3 + b$, sextic twist is one of $y^2 = x^3 + b \cdot i$ (type M) or $y^2 = x^3 + b/i$ (type D)
- Scott'09: either type-M or type-D will have a worse "untwisting" isomorphism than the other, so reject those instances

Remedy: C-Lange-Naehrig'10

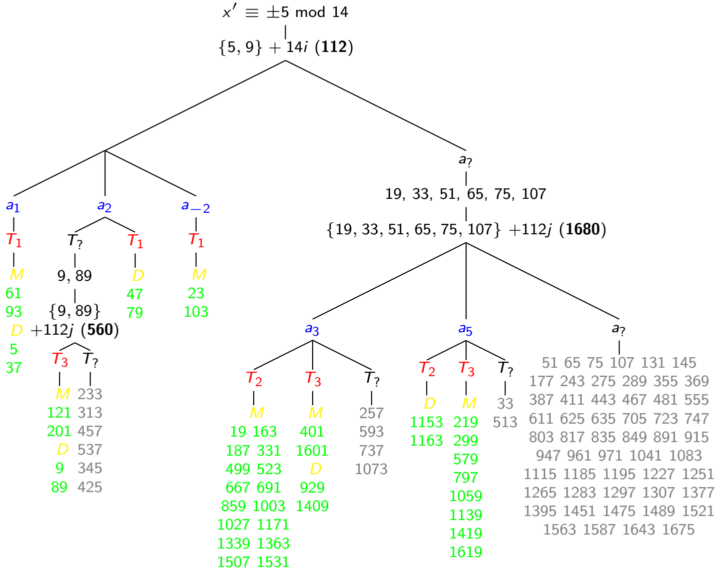
If (optimal) ate $a_m(Q, P) = a_m(\Psi(Q'), P)$ is bilinear, then so is $a_m(Q', P')$, i.e. can compute pairing entirely on E or E'

- If twist is Type-M, then $a_m(Q', P')$ (twisting) is best
- If twist is Type-D, then $a_m(Q, P)$ (untwisting) is best

Particularly friendly members of family trees

- For BLS $k = 24$, we used $x = 7, 16, 31, 64 \bmod 72$ instead of $x \equiv 1 \bmod 3 \ldots$
- But what happened to the other congruencies?
- *"Particularly friendly members of family trees": eprint 2012/072* - wrote a script that exhausts the subcongruencies in each family until all the best options are found
- For all the most popular families with $k = 8, 12, 16, 18, 24, 27, 32, 36, 48$, constructs a *family tree*. . .
- Tree branches depending on
  1. Best tower
  2. Curve constant
  3. Twist type

e.g. KSS $k = 16$ family tree: $y^2 = x^3 + ax$



Picking fruits in the trees: KSS $k = 16$

rating	equiv. class for x' ($x' = x/5$)	tower	a	twist type	$\mathbb{G}_1$ gen. $[h](\cdot,\cdot)$	$\mathbb{G}_2'$ gen. $[h'](\cdot,\cdot)$	%
$\star\star\star\star\star$	$61, 93 \bmod 112$	T_1	1	M	-	$\left(v-1, \sqrt{(v-1)^3 + v(v-1)}\right)$	12.2
	$5, 37 \bmod 112$	T_1	1	D	-	$\left(-v, \sqrt{-v^3-1}\right)$	12.7
	$47, 79 \bmod 112$	T_1	2	D	-	$\left(2/v, \sqrt{\frac{8}{v^3}+\frac{4}{v^2}}\right)$	12.1
	$23, 103 \bmod 112$	T_1	-2	M	$(1, \sqrt{-1})$	-	13.1
$\star\star\star\star$	$\{19, \dots, 1531\}_{16} \bmod 1680$	T_2	3	M	$(1, 2)$	$\left(3/v, \sqrt{\frac{27}{v^3}+\frac{9}{v^2}}\right)$	7.9
$\star\star\star$	$1153, 1633 \bmod 1680$	T_2	5	D	$\left(2, 2\sqrt{3}\right)$	-	0.9

Favourite picks from the $k = 16$ KSS tree.

$\mathbb{F}_p \xrightarrow{\mathbb{F}_p[u]/(u^2+u_i)} \mathbb{F}_{p^2} \xrightarrow{\mathbb{F}_{p^2}[v]/(u^8-v_i)} \mathbb{F}_{p^{16}}$			
T_i	T_1	T_2	T_3
(u_i, v_i)	$(2, u)$	$(3, u)$	$(5, u)$

Efficient towering options in the $k = 16$ KSS tree.

Particularly friendly members of family trees

- Gives a way to streamline your search and pre-order your pairing properties

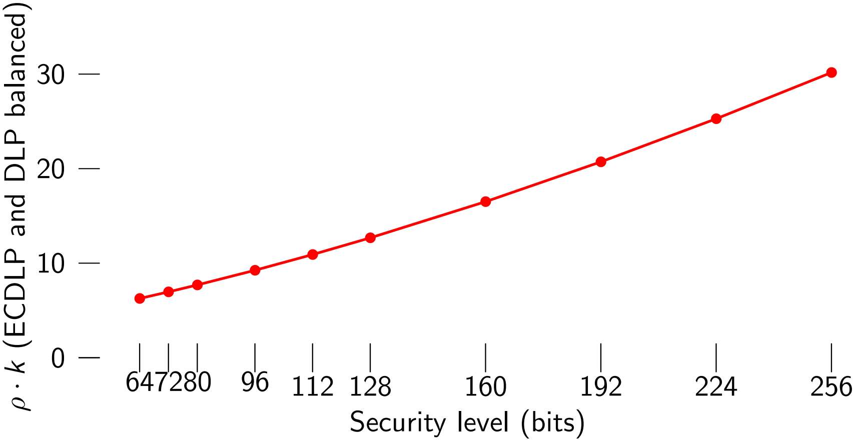
- More useful at higher security levels where
  - Search space grows
  - Primality testing slows down

- Details how to search, but also provides many low-hamming weight curves that save you having to

- **However, paper needs a re-write**

5. Pairings at the 192-bit security level

192-bit security wants $\rho \cdot k \approx 20$



The best family for 192-bit security

- BN curves with $k = 12$ fall short (i.e. ground field is too big)
- KSS curves with $k = 16$ have $\rho = 1.25$ so $\rho \cdot k = 20$, but only have a quartic twist down to $\mathbb{F}_{p^4}$
- KSS curves with $k = 18$ have $\rho = 1.33$ so $\rho \cdot k = 24$, but they have a sextic twist down to $\mathbb{F}_{p^3}$
- Final exponentiations are similar
- **So which family reigns supreme?**

BLS $k = 12$ for 192-bit security

- **Pairing 2012: Aranha, Fuentes-Castañeda, Knapp, Menezes and Rodríguez-Henríquez**: None of the above!!!
- Surprising result: BLS $k = 12$ curves have $\rho = 1.5$, and were overlooked by myself, Mike Scott, and others...
- 14,000,000 cycles for 192-bit pairing (compare to 1,600,000 at 128-bit level)
- Another cool result from Aranha *et al.* ($\times 2$): **the Weil pairing is back!**
- At higher levels of security, the final exponentiation swamps the computation, so the (shortened) Weil pairing can outperform optimal ate (in parallelizable environments)
- Simpler polynomials yield nicer final exponentiations (KSS are more complicated)

6. Work in progress

Whoops!

- Very common scenario: in the pairing $e(P, Q)$, one of the arguments is fixed as a long term secret key (or constant public param, etc)
- We can exploit this and perform precomputations
- C-Stebila'10 - merging iterations gives speedups for optimal ate pairings
- Scott'11: "would give a small but useful speedup"
- **But: I majorly stuffed up at LatinCrypt'10**: used affine, but recently realised projective would be much better

	128-bit optimal pairing $k = 12$ BN curve $\mathbb{F}_q = 254$ bits		256-bit optimal pairing $k = 24$ BLS curve $\mathbb{F}_q = 639$ bits	
precomp method	Miller loop cost	$\approx$ storage required (bits)	Miller loop cost	$\approx$ storage required (bits)
none	6469 $\mathbf{m}_1$	-	19069 $\mathbf{m}_1$	-
Scott '05	5017 $\mathbf{m}_1$	70,000	14794 $\mathbf{m}_1$	340,000
quadrupling	4446 $\mathbf{m}_1$	75,000	12898 $\mathbf{m}_1$	368,000
octupling	4053 $\mathbf{m}_1$	100,000	11673 $\mathbf{m}_1$	510,000

Updated projections

Thanks for your attention