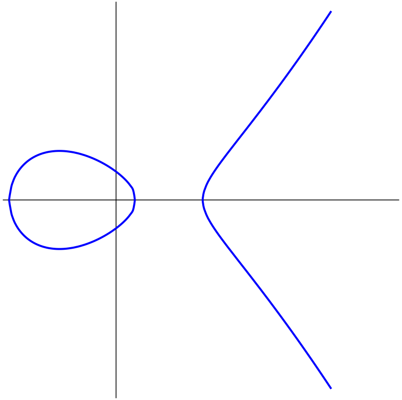
Cocks–Pinch curves with efficient ate pairing

Simon Masson
Joint work with A. Guillevic, E. Thomé

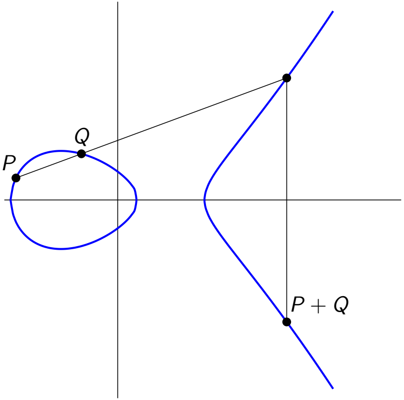
Thales – LORIA

June 21, 2019

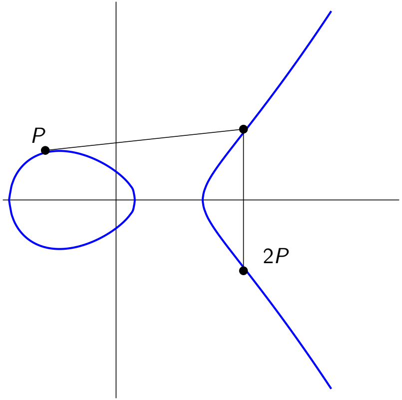
Elliptic curve. $y^2 = x^3 + Ax + B$



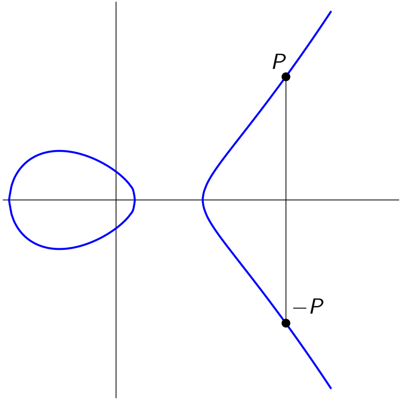
Elliptic curve. $y^2 = x^3 + Ax + B$



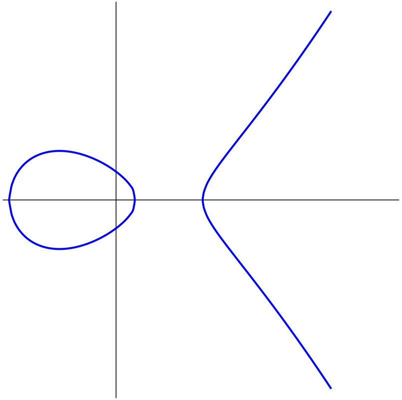
Elliptic curve. $y^2 = x^3 + Ax + B$



Elliptic curve. $y^2 = x^3 + Ax + B$



Elliptic curve. $y^2 = x^3 + Ax + B$

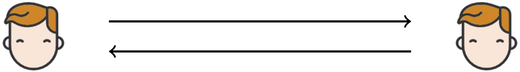


Points on an elliptic curve form a group (with group law $+$).

From $[s]P = \underbrace{P + \ldots + P}_{s \text{ times}}$ and P, it is difficult to recover s.

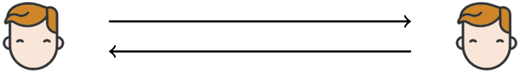
From $[s]P = \underbrace{P + \ldots + P}_{s \text{ times}}$ and P, it is difficult to recover s.

Diffie-Hellman key exchange.



From $[s]P = \underbrace{P + \ldots + P}_{s \text{ times}}$ and P, it is difficult to recover s.

Diffie-Hellman key exchange.

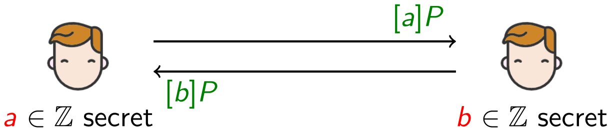


$a \in \mathbb{Z}$ secret

$b \in \mathbb{Z}$ secret

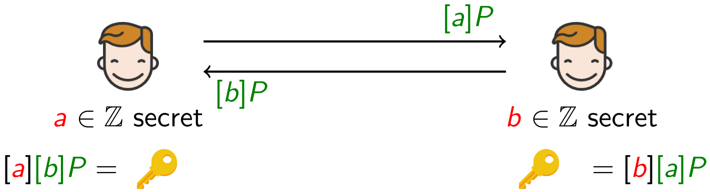
From $[s]P = \underbrace{P + \ldots + P}_{s \text{ times}}$ and P, it is difficult to recover s.

Diffie-Hellman key exchange.



From $[s]P = \underbrace{P + \ldots + P}_{s \text{ times}}$ and P, it is difficult to recover s.

Diffie-Hellman key exchange.



WhatsApp

Terms

Public Key Types

- `Identity Key Pair` – A long-term Curve25519 key pair, generated at install time.
- `Signed Pre Key` – A medium-term Curve25519 key pair, generated at install time, signed by the `Identity Key`, and rotated on a periodic timed basis.
- `One-Time Pre Keys` – A queue of Curve25519 key pairs for one time use, generated at install time, and replenished as needed.

Session Key Types

- `Root Key` – A 32-byte value that is used to create `Chain Keys`.
- `Chain Key` – A 32-byte value that is used to create `Message Keys`.
- `Message Key` – An 80-byte value that is used to encrypt message contents. 32 bytes are used for an AES-256 key, 32 bytes for a HMAC-SHA256 key, and 16 bytes for an IV.

Initiating Session Setup

To communicate with another WhatsApp user, a WhatsApp client first needs to establish an encrypted session. Once the session is established, clients do not need to rebuild a new session with each other until the existing session state is lost through an external event such as an app reinstall or device change.

To establish a session:

1. The initiating client ("initiator") requests the public `Identity Key`, public `Signed Pre Key`, and a single public `One-Time Pre Key` for the recipient.
2. The server returns the requested public key values. A `One-Time Pre Key` is only used once, so it is removed from server storage after being requested. If the recipient's latest batch of `One-Time Pre Keys` has been consumed and the recipient has not replenished them, no `One-Time Pre Key` will be returned.
3. The initiator saves the recipient's `Identity Key` as $I_{recipient}$, the `Signed Pre Key` as $S_{recipient}$, and the `One-Time Pre Key` as $O_{recipient}$.
4. The initiator generates an ephemeral Curve25519 key pair, $E_{initiator}$.
5. The initiator loads its own `Identity Key` as $I_{initiator}$.
6. The initiator calculates a master secret as $master\_secret = ECDH(I_{initiator}, S_{recipient}) \,||\, ECDH(E_{initiator}, I_{recipient}) \,||\, ECDH(E_{initiator}, S_{recipient}) \,||\, ECDH(E_{initiator}, O_{recipient})$. If there is no `One Time Pre Key`, the final ECDH is omitted.
7. The initiator uses HKDF to create a `Root Key` and `Chain Keys` from the `master_secret`.

Discrete logarithm problem (DLP).

Given $[s]P = \underbrace{P + \ldots + P}_{s \text{ times}}$ and P, it is hard to recover s if $\langle P \rangle$ is a large subgroup.

Discrete logarithm problem (DLP).
Given $[s]P = \underbrace{P + \ldots + P}_{s \text{ times}}$ and P, it is hard to recover s if $\langle P \rangle$ is a large subgroup.

Subgroup attack.

$\#E(\mathbb{F}_p) =$

$2^2 \times 5^5 \times 13 \times 37 \times 18575429 \times 505818037 \times 1089749937157876379177809361515176882436093600552189158080830008040550806174507 3$,
someone can choose a point on a small subgroup instead of the big one. Discrete logarithm problem is easy there !

Discrete logarithm problem (DLP).
Given $[s]P = \underbrace{P + \ldots + P}_{s \text{ times}}$ and P, it is hard to recover s if $\langle P \rangle$ is a large subgroup.

Subgroup attack.

$\#E(\mathbb{F}_p) =$

$2^2 \times 5^5 \times 13 \times 37 \times 18575429 \times 505818037 \times 10897499371578763791778093615151768824360936005521891580808300080405508061745073$,
someone can choose a point on a small subgroup instead of the big one. Discrete logarithm problem is easy there !

Counter the attack.

- Checking that P is of order 108...073:
  $[108...073]P = 0$, $[2^2 \times 5^5 \times 13 \times 37 \times 18575429 \times 505818037]P \neq 0$ and $P \neq 0$.
- Choosing a curve with $\#E(\mathbb{F}_p)$ with no small factor.

If $\gcd(r, p) = 1$ (think $r = 1089749937157876379177809361515176882436093600552189158080830008040550806174507 3$),

$$E(\overline{\mathbb{F}}_p)[r] \simeq \underbrace{\mathbb{Z}/r\mathbb{Z}}_{\mathbb{G}_1} \times \underbrace{\mathbb{Z}/r\mathbb{Z}}_{\mathbb{G}_2}$$

If $\gcd(r, p) = 1$ (think $r = 1089749937157876379177809361515176882436093600552189158080830008040550806174507 3$),

$$E(\overline{\mathbb{F}}_p)[r] \simeq \underbrace{\mathbb{Z}/r\mathbb{Z}}_{\mathbb{G}_1} \times \underbrace{\mathbb{Z}/r\mathbb{Z}}_{\mathbb{G}_2}$$

- $\mathbb{G}_1$: over $\mathbb{F}_p$, one part of the [r]-torsion ($r \mid E(\mathbb{F}_p)$)

If $\gcd(r, p) = 1$ (think $r = 1089749937157876379177809361515176882436093600552189158080830008040550806174507\!3$),

$$E(\overline{\mathbb{F}}_p)[r] \simeq \underbrace{\mathbb{Z}/r\mathbb{Z}}_{\mathbb{G}_1} \times \underbrace{\mathbb{Z}/r\mathbb{Z}}_{\mathbb{G}_2}$$

- $\mathbb{G}_1$: over $\mathbb{F}_p$, one part of the [r]-torsion $(r \mid E(\mathbb{F}_p))$
- $\mathbb{G}_2$: the full [r]-torsion is defined over an extension of $\mathbb{F}_p$.

If $\gcd(r, p) = 1$ (think $r = 10897499371578763791778093615151768824360936005521891580808300080405508061745073$),

$$E(\overline{\mathbb{F}}_p)[r] \simeq \underbrace{\mathbb{Z}/r\mathbb{Z}}_{\mathbb{G}_1} \times \underbrace{\mathbb{Z}/r\mathbb{Z}}_{\mathbb{G}_2}$$

- $\mathbb{G}_1$: over $\mathbb{F}_p$, one part of the [r]-torsion $(r \mid E(\mathbb{F}_p))$
- $\mathbb{G}_2$: the full [r]-torsion is defined over an extension of $\mathbb{F}_p$.

Definition (embedding degree)

The embedding degree of E w.r.t. r (coprime to p) is the smallest integer k such that $E[r]$ is defined over $\mathbb{F}_{p^k}$.

Pairings on elliptic curves

Definition

A pairing on an elliptic curve E is a bilinear non-degenerate application $e : E \times E \longrightarrow \mathbb{F}_{p^k}^{\times}$

Pairings on elliptic curves

Definition

A pairing on an elliptic curve E is a bilinear non-degenerate application $e : E \times E \longrightarrow \mathbb{F}_{p^k}^{\times}$

For some particular $P, Q \in E[r]$ and $a, b \in \mathbb{Z}$,

$$e(aP, bQ)$$

Pairings on elliptic curves

Definition

A pairing on an elliptic curve E is a bilinear non-degenerate application $e : E \times E \longrightarrow \mathbb{F}_{p^k}^{\times}$

For some particular $P, Q \in E[r]$ and $a, b \in \mathbb{Z}$,

$$e(aP, bQ) = e(P, bQ)^a = e(P, Q)^{ab}$$

Pairings on elliptic curves

Definition

A pairing on an elliptic curve E is a bilinear non-degenerate application $e : E \times E \longrightarrow \mathbb{F}_{p^k}^{\times}$

For some particular $P, Q \in E[r]$ and $a, b \in \mathbb{Z}$,

$$e(aP, bQ) = e(P, bQ)^a = e(P, Q)^{ab}$$

elliptic curve

Pairings on elliptic curves

Definition

A pairing on an elliptic curve E is a bilinear non-degenerate application $e : E \times E \longrightarrow \mathbb{F}_{p^k}^{\times}$

For some particular $P, Q \in E[r]$ and $a, b \in \mathbb{Z}$,

$$e(aP, bQ) = e(P, bQ)^a = e(P, Q)^{ab}$$

pairing-friendly elliptic curve

Pairings on elliptic curves

Definition

A pairing on an elliptic curve E is a bilinear non-degenerate application $e : E \times E \longrightarrow \mathbb{F}_{p^k}^{\times}$

For some particular $P, Q \in E[r]$ and $a, b \in \mathbb{Z}$,

$$e(aP, bQ) = e(P, bQ)^a = e(P, Q)^{ab}$$

Secure pairing-friendly elliptic curve

Pairings on elliptic curves

Definition

A pairing on an elliptic curve E is a bilinear non-degenerate application $e : E \times E \longrightarrow \mathbb{F}_{p^k}^{\times}$

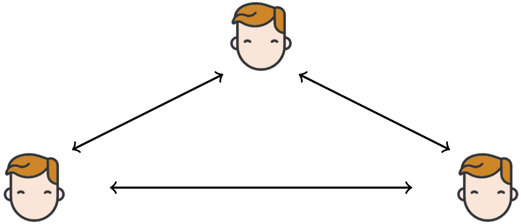
For some particular $P, Q \in E[r]$ and $a, b \in \mathbb{Z}$,

$$e(aP, bQ) = e(P, bQ)^a = e(P, Q)^{ab}$$

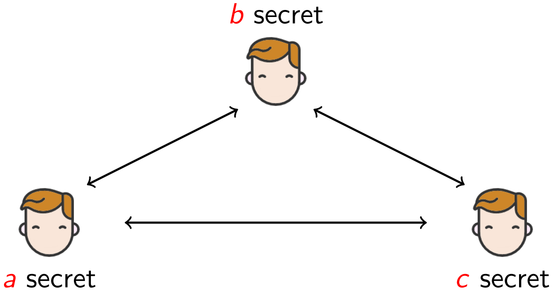
Secure pairing-friendly elliptic curve with an efficient pairing

Application 1. Tripartite one round key exchange. (Joux 2000)

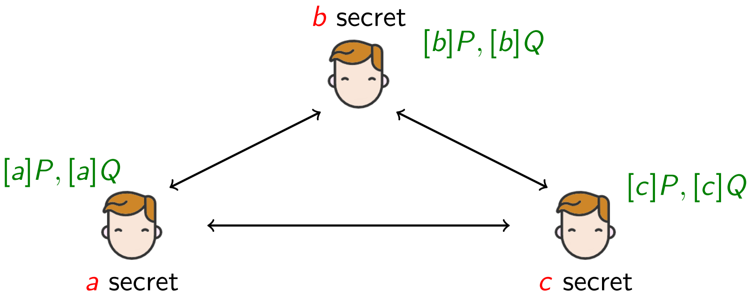
Application 1. Tripartite one round key exchange. (Joux 2000)



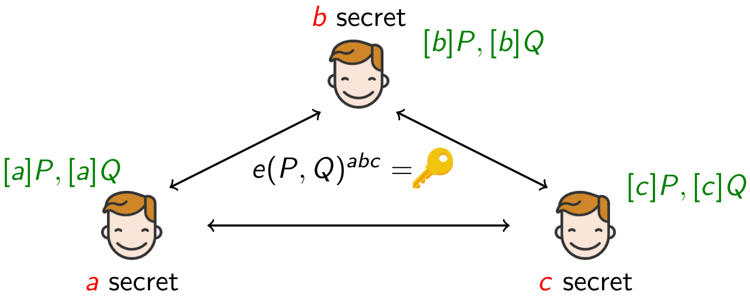
Application 1. Tripartite one round key exchange. (Joux 2000)



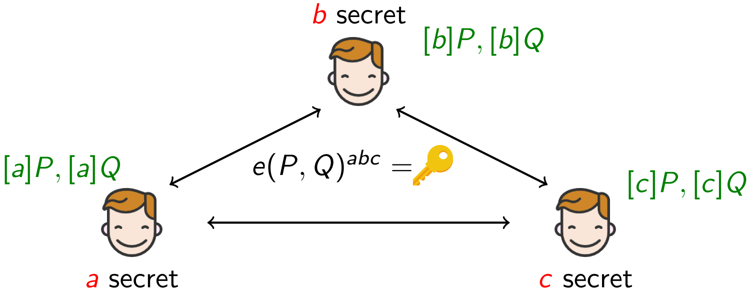
Application 1. Tripartite one round key exchange. (Joux 2000)



Application 1. Tripartite one round key exchange. (Joux 2000)



Application 1. Tripartite one round key exchange. (Joux 2000)



$$e([b]P, [c]Q)^{a} = e(P, Q)^{bca}$$

Application 2. BLS signature
$H : \{0,1\}^* \to \langle P \rangle$ is a hash function, with $P \in E(\mathbb{F}_p)$ of prime order r.

Application 2. BLS signature
$H : \{0,1\}^* \to \langle P \rangle$ is a hash function, with $P \in E(\mathbb{F}_p)$ of prime order r.
Secret key: $s_k \in \{2, \ldots, r-1\}$.
Public key: $P_k = [s_k]P$.

Application 2. BLS signature

$H : \{0,1\}^* \to \langle P \rangle$ is a hash function, with $P \in E(\mathbb{F}_p)$ of prime order r.

Secret key: $s_k \in \{2, \ldots, r-1\}$.

Public key: $P_k = [s_k]P$.

Signing a message $M \in \{0,1\}^*$: $\sigma = [s_k]H(M)$.

Verifying the signature: $e(P_k, H(M)) \stackrel{?}{=} e(P, \sigma)$.

Application 2. BLS signature

$H : \{0,1\}^* \to \langle P \rangle$ is a hash function, with $P \in E(\mathbb{F}_p)$ of prime order r.

Secret key: $s_k \in \{2, \ldots, r-1\}$.

Public key: $P_k = [s_k]P$.

Signing a message $M \in \{0,1\}^*$: $\sigma = [s_k]H(M)$.

Verifying the signature: $e(P_k, H(M)) \stackrel{?}{=} e(P, \sigma)$.

$$e(P_k, H(M)) = e([s_k]P, H(M)) = e(P, [s_k]H(M)) = e(P, \sigma)$$

Many other applications:

- Blind signature
- Identity-based encryption
- Post-quantum cryptography compressions (eprint 2017/1143)
- Short group signature (eprint 2018/1115)
- Verifiable delay functions (eprint 2019/166)
- etc.

Tate and ate pairing

1 Tate and ate pairing

2 Pairing-friendly curves for 128 bits of security

3 Timings and comparisons

The Tate and ate pairings are computed in two steps:

1. Evaluating a function at a point of the curve (Miller loop)
2. Exponentiating to the power $(p^k - 1)/r$ (final exponentiation).

The Tate and ate pairings are computed in two steps:

1. Evaluating a function at a point of the curve (Miller loop)
2. Exponentiating to the power $(p^k - 1)/r$ (final exponentiation).

Definition

For $P \in \mathbb{G}_1 = E(\mathbb{F}_p)[r]$, $Q \in \mathbb{G}_2 = E(\mathbb{F}_{p^k})[r]$,

$$\mathrm{Tate}(P, Q) := f_{r,P}(Q)^{(p^k-1)/r} \qquad \mathrm{ate}(P, Q) := f_{t-1,Q}(P)^{(p^k-1)/r}$$

Miller loop step.

Miller loop step.

Definition

The Miller loop computes the function $f_{s,Q}$ such that Q is a zero of order s, and $[s]Q$ is a pole of order 1, i.e

$$\mathrm{div}(f_{s,Q}) = s(Q) - ([s]Q) - (s-1)\mathcal{O}$$

Miller loop step.

Definition

The Miller loop computes the function $f_{s,Q}$ such that Q is a zero of order s, and $[s]Q$ is a pole of order 1, i.e

$$\mathrm{div}(f_{s,Q}) = s(Q) - ([s]Q) - (s-1)\mathcal{O}$$

Miller loop for Tate.
Compute $x = f_{r,P}(Q)$ with $P \in E(\mathbb{F}_p)[r]$ and $Q \in E(\mathbb{F}_{p^k})[r]$.

Miller loop step.

Definition

The Miller loop computes the function $f_{s,Q}$ such that Q is a zero of order s, and $[s]Q$ is a pole of order 1, i.e

$$\mathrm{div}(f_{s,Q}) = s(Q) - ([s]Q) - (s-1)\mathcal{O}$$

Miller loop for Tate.
Compute $x = f_{r,P}(Q)$ with $P \in E(\mathbb{F}_p)[r]$ and $Q \in E(\mathbb{F}_{p^k})[r]$.

Miller loop for ate.
For ate: compute $x = f_{t-1,Q}(P)$ with $P \in E(\mathbb{F}_p)[r]$ and $Q \in E(\mathbb{F}_{p^k})[r]$.

Algorithm: $\textsc{MillerLoop}(s, P, Q)$ – Compute $f_{s,Q}(P)$.

$f \leftarrow 1$
$S \leftarrow Q$
for b bit of s from second MSB to LSB **do**
  $f \leftarrow f^2 \cdot \ell_{S,S}(P)/v_{2S}(P)$
  $S \leftarrow [2]S$
  if $b = 1$ **then**
    $f \leftarrow f \cdot \ell_{S,Q}(P)/v_{S+Q}(P)$
    $S \leftarrow S + Q$
  end if
end for
return f such that $\mathrm{div}(f_{s,Q}) = s(Q) - ([s]Q) - (s-1)\mathcal{O}$

Example: $f_{5,Q}(P)$.

$$s = 5 = \overline{101}^2$$

Example: $f_{5,Q}(P)$.

$$s = 5 = \overline{\boxed{1}01}^{2}$$

$$f = 1$$

Example: $f_{5,Q}(P)$.

$$s = 5 = \overline{1\boxed{0}1}^2$$

$$f = 1$$

Example: $f_{5,Q}(P)$.

$$s = 5 = \overline{1\boxed{0}1}^{2}$$

$$f = 1^2$$

Example: $f_{5,Q}(P)$.

$$s = 5 = \overline{1\boxed{0}1}^2$$

$$f = 1^2 \cdot \ell_{Q,Q}(P)/v_{2Q}(P)$$

Example: $f_{5,Q}(P)$.

$$s = 5 = \overline{10\boxed{1}}^{2}$$

$$f = \left(1^2 \cdot \ell_{Q,Q}(P)/v_{2Q}(P)\right)^2$$

Example: $f_{5,Q}(P)$.

$$s = 5 = \overline{10\boxed{1}}^{2}$$

$$f = \left(1^2 \cdot \ell_{Q,Q}(P)/v_{2Q}(P)\right)^2 \cdot \ell_{2Q,2Q}(P)/v_{4Q}(P)$$

Example: $f_{5,Q}(P)$.

$$s = 5 = \overline{10\boxed{1}}^{2}$$

$$f = \left(1^2 \cdot \ell_{Q,Q}(P)/v_{2Q}(P)\right)^2 \cdot \ell_{2Q,2Q}(P)/v_{4Q}(P) \cdot \ell_{4Q,Q}(P)/v_{5Q}(P)$$

Example: $f_{5,Q}(P)$.

$$s = 5 = \overline{101}^2$$

$$f = \left(1^2 \cdot \ell_{Q,Q}(P)/v_{2Q}(P)\right)^2 \cdot \ell_{2Q,2Q}(P)/v_{4Q}(P) \cdot \ell_{4Q,Q}(P)/v_{5Q}(P)$$

Divisor:

$$4(Q) + 2(-2Q)$$

Example: $f_{5,Q}(P)$.

$$s = 5 = \overline{101}^2$$

$$f = \left(1^2 \cdot \ell_{Q,Q}(P)/v_{2Q}(P)\right)^2 \cdot \ell_{2Q,2Q}(P)/v_{4Q}(P) \cdot \ell_{4Q,Q}(P)/v_{5Q}(P)$$

Divisor:

$$4(Q) + 2(-2Q) + 2(2Q) + (-4Q)$$

Example: $f_{5,Q}(P)$.

$$s = 5 = \overline{101}^2$$

$$f = \left(1^2 \cdot \ell_{Q,Q}(P)/v_{2Q}(P)\right)^2 \cdot \ell_{2Q,2Q}(P)/v_{4Q}(P) \cdot \color{red}{\ell_{4Q,Q}(P)}/v_{5Q}(P)$$

Divisor:

$$4(Q) + 2(-2Q) + 2(2Q) + (-4Q) + \color{red}{(Q) + (4Q) + (-5Q)}$$

Example: $f_{5,Q}(P)$.

$$s = 5 = \overline{101}^2$$

$$f = \left(1^2 \cdot \ell_{Q,Q}(P)/v_{2Q}(P)\right)^2 \cdot \ell_{2Q,2Q}(P)/v_{4Q}(P) \cdot \ell_{4Q,Q}(P)/v_{5Q}(P)$$

Divisor:

$$4(Q) + 2(-2Q) + 2(2Q) + (-4Q) + (Q) + (4Q) + (-5Q)$$

$$-2(2Q) - 2(-2Q) - 2(\mathcal{O})$$

Example: $f_{5,Q}(P)$.

$$s = 5 = \overline{101}^2$$

$$f = \left(1^2 \cdot \ell_{Q,Q}(P)/v_{2Q}(P)\right)^2 \cdot \ell_{2Q,2Q}(P)/v_{4Q}(P) \cdot \ell_{4Q,Q}(P)/v_{5Q}(P)$$

Divisor:

$$4(Q) + 2(-2Q) + 2(2Q) + (-4Q) + (Q) + (4Q) + (-5Q)$$

$$-2(2Q) - 2(-2Q) - 2(\mathcal{O}) - (4Q) - (-4Q) - (\mathcal{O})$$

Example: $f_{5,Q}(P)$.

$$s = 5 = \overline{101}^2$$

$$f = \left(1^2 \cdot \ell_{Q,Q}(P)/v_{2Q}(P)\right)^2 \cdot \ell_{2Q,2Q}(P)/v_{4Q}(P) \cdot \ell_{4Q,Q}(P)/v_{5Q}(P)$$

Divisor:

$$4(Q) + 2(-2Q) + 2(2Q) + (-4Q) + (Q) + (4Q) + (-5Q)$$

$$-2(2Q) - 2(-2Q) - 2(\mathcal{O}) - (4Q) - (-4Q) - (\mathcal{O}) - (5Q) - (-5Q) - (\mathcal{O})$$

Example: $f_{5,Q}(P)$.

$$s = 5 = \overline{101}^2$$

$$f = \left(1^2 \cdot \ell_{Q,Q}(P)/v_{2Q}(P)\right)^2 \cdot \ell_{2Q,2Q}(P)/v_{4Q}(P) \cdot \ell_{4Q,Q}(P)/v_{5Q}(P)$$

Divisor:

$$4(Q) + 2(-2Q) + 2(2Q) + (-4Q) + (Q) + (4Q) + (-5Q)$$

$$-2(2Q) - 2(-2Q) - 2(\mathcal{O}) - (4Q) - (-4Q) - (\mathcal{O}) - (5Q) - (-5Q) - (\mathcal{O})$$

$$\mathrm{div}(f) = 5(Q) - (5Q) - 4(\mathcal{O})$$

Final exponentiation step.

Final exponentiation step.

$f_{r,P}(Q)$ and $f_{t-1,Q}(P)$ are r-th roots of unity.

We obtain a unique coset representative by elevating to the power $(p^k - 1)/r$.

Final exponentiation step.
$f_{r,P}(Q)$ and $f_{t-1,Q}(P)$ are r-th roots of unity.
We obtain a unique coset representative by elevating to the power $(p^k-1)/r$.

$$\frac{p^k-1}{r} = \frac{p^k-1}{\Phi_k(p)} \cdot \frac{\Phi_k(p)}{r}$$

Final exponentiation step.
$f_{r,P}(Q)$ and $f_{t-1,Q}(P)$ are r-th roots of unity.
We obtain a unique coset representative by elevating to the power $(p^k - 1)/r$.

$$\frac{p^k - 1}{r} = \frac{p^k - 1}{\Phi_k(p)} \cdot \frac{\Phi_k(p)}{r}$$

- First exponentiation: $\frac{p^k-1}{\Phi_k(p)}$.

Final exponentiation step.
$f_{r,P}(Q)$ and $f_{t-1,Q}(P)$ are r-th roots of unity.
We obtain a unique coset representative by elevating to the power $(p^k - 1)/r$.

$$\frac{p^k - 1}{r} = \frac{p^k - 1}{\Phi_k(p)} \cdot \frac{\Phi_k(p)}{r}$$

- First exponentiation: $\frac{p^k-1}{\Phi_k(p)}$.Polynomial in p with very small coefficients.
  Very efficent with Frobenius: if $\mathbb{F}_{p^k} = \mathbb{F}_p[x]/(x^k - \alpha)$,
  $a^p = \left(\sum_{i=0}^{k-1} a_i x^i\right)^p = \sum_{i=0}^{k-1} a_i x^{ip}$ and x^{ip} can be precomputed.

Final exponentiation step.

$f_{r,P}(Q)$ and $f_{t-1,Q}(P)$ are r-th roots of unity.

We obtain a unique coset representative by elevating to the power $(p^k-1)/r$.

$$\frac{p^k-1}{r} = \frac{p^k-1}{\Phi_k(p)} \cdot \frac{\Phi_k(p)}{r}$$

- First exponentiation: $\frac{p^k-1}{\Phi_k(p)}$.Polynomial in p with very small coefficients.
  Very efficent with Frobenius: if $\mathbb{F}_{p^k} = \mathbb{F}_p[x]/(x^k-\alpha)$,
  $a^p = \left(\sum_{i=0}^{k-1} a_i x^i\right)^p = \sum_{i=0}^{k-1} a_i x^{ip}$ and x^{ip} can be precomputed.
  A Frobenius costs $k-1$ multiplications over $\mathbb{F}_p$.

Final exponentiation step.
$f_{r,P}(Q)$ and $f_{t-1,Q}(P)$ are r-th roots of unity.
We obtain a unique coset representative by elevating to the power $(p^k - 1)/r$.

$$\frac{p^k - 1}{r} = \frac{p^k - 1}{\Phi_k(p)} \cdot \frac{\Phi_k(p)}{r}$$

- First exponentiation: $\frac{p^k-1}{\Phi_k(p)}$.Polynomial in p with very small coefficients.
  Very efficent with Frobenius: if $\mathbb{F}_{p^k} = \mathbb{F}_p[x]/(x^k - \alpha)$,
  $a^p = \left(\sum_{i=0}^{k-1} a_i x^i\right)^p = \sum_{i=0}^{k-1} a_i x^{ip}$ and x^{ip} can be precomputed.
  A Frobenius costs $k-1$ multiplications over $\mathbb{F}_p$.
- Second exponentiation: more expensive. Possible optimizations.

Pairing-friendly curves for 128 bits of security

1 Tate and ate pairing

2 Pairing-friendly curves for 128 bits of security

3 Timings and comparisons

Parameters of pairing-friendly curves.

An elliptic curve E defined over $\mathbb{F}_p$, of trace t and discriminant D is pairing-friendly of embedding degree k if

- p, r are primes and t is relatively prime to p
- r divides $p+1-t$ and p^k-1 but does not divide p^i-1 for $1 \leq i < k$
- $4p - t^2 = Dy^2$ for a sufficiently small positive integer D and an integer y.

Parameters of pairing-friendly curves.
An elliptic curve E defined over $\mathbb{F}_p$, of trace t and discriminant D is pairing-friendly of embedding degree k if

- p, r are primes and t is relatively prime to p
- r divides $p+1-t$ and p^k-1 but does not divide p^i-1 for $1 \leq i < k$
- $4p - t^2 = Dy^2$ for a sufficiently small positive integer D and an integer y.

Equation of the curve. (Complex multiplication method).

Parameters of pairing-friendly curves.
An elliptic curve E defined over $\mathbb{F}_p$, of trace t and discriminant D is pairing-friendly of embedding degree k if

- p, r are primes and t is relatively prime to p
- r divides $p+1-t$ and p^k-1 but does not divide p^i-1 for $1 \leq i < k$
- $4p - t^2 = Dy^2$ for a sufficiently small positive integer D and an integer y.

Equation of the curve. (Complex multiplication method).

1. Compute the discriminant D of the curve : $t^2 - 4p = -Dy^2$ with D square-free.
   `sage: (t**2-4*p).squarefree_part()`

Parameters of pairing-friendly curves.
An elliptic curve E defined over $\mathbb{F}_p$, of trace t and discriminant D is pairing-friendly of embedding degree k if

- p, r are primes and t is relatively prime to p
- r divides $p+1-t$ and p^k-1 but does not divide p^i-1 for $1 \leq i < k$
- $4p - t^2 = Dy^2$ for a sufficiently small positive integer D and an integer y.

Equation of the curve. (Complex multiplication method).

1. Compute the discriminant D of the curve : $t^2 - 4p = -Dy^2$ with D square-free.
   sage: `(t**2-4*p).squarefree_part()`
2. Compute the Hilbert class polynomial $H_D(X)$ whose roots are the j-invariants of curves with discriminant D.
   sage: `hilbert_class_polynomial(D)`

Parameters of pairing-friendly curves.
An elliptic curve E defined over $\mathbb{F}_p$, of trace t and discriminant D is pairing-friendly of embedding degree k if

- p, r are primes and t is relatively prime to p
- r divides $p+1-t$ and p^k-1 but does not divide p^i-1 for $1 \leq i < k$
- $4p-t^2 = Dy^2$ for a sufficiently small positive integer D and an integer y.

Equation of the curve. (Complex multiplication method).

1. Compute the discriminant D of the curve : $t^2-4p = -Dy^2$ with D square-free.
   sage: `(t**2-4*p).squarefree_part()`
2. Compute the Hilbert class polynomial $H_D(X)$ whose roots are the j-invariants of curves with discriminant D.
   sage: `hilbert_class_polynomial(D)`
3. Compute a curve whose j-invariant is one of these roots.
   sage: `EllipticCurve_from_j(j0)`.

Generation of curves with given prime k, and square-free D.

Algorithm: $\textsc{Cocks-Pinch}(k, D)$ – Compute a pairing-friendly curve $E/\mathbb{F}_p$ of trace t with a subgroup of order r, such that $t^2 - Dy^2 = 4p$.

Set a prime r such that $k \mid r - 1$ and $\sqrt{-D} \in \mathbb{F}_r$
Set T such that $r \mid \Phi_k(T)$
$t \leftarrow T + 1$
$y \leftarrow (t - 2)/\sqrt{-D}$
Lift $t, y \in \mathbb{Z}$ such that $t^2 + Dy^2 \equiv 0 \mod 4$
$p \leftarrow (t^2 + Dy^2)/4$
if p is prime **then return** $[p, t, y, r]$ **else** Repeat with another r.

Generation of curves with given prime k, and square-free D.

Algorithm: $\textsc{Cocks-Pinch}(k, D)$ – Compute a pairing-friendly curve $E/\mathbb{F}_p$ of trace t with a subgroup of order r, such that $t^2 - Dy^2 = 4p$.

Set a prime r such that $k \mid r - 1$ and $\sqrt{-D} \in \mathbb{F}_r$
Set T such that $r \mid \Phi_k(T)$
$t \leftarrow T + 1$
$y \leftarrow (t - 2)/\sqrt{-D}$
Lift $t, y \in \mathbb{Z}$ such that $t^2 + Dy^2 \equiv 0 \mod 4$
$p \leftarrow (t^2 + Dy^2)/4$
if p is prime **then return** $[p, t, y, r]$ **else** Repeat with another r.

Large trace $t \implies$ the ate pairing is not very efficient 

Brezing-Weng families.

Brezing-Weng families.

1. Find $r(x) \in \mathbb{Z}[x]$ such that $K := \mathbb{Q}[x]/(r(x))$ is a number field containing $\sqrt{-D}$ and $\mathbb{Q}(\zeta_k)$ for a chosen primitive k-th root ζ_k.

Brezing-Weng families.

1. Find $r(x) \in \mathbb{Z}[x]$ such that $K := \mathbb{Q}[x]/(r(x))$ is a number field containing $\sqrt{-D}$ and $\mathbb{Q}(\zeta_k)$ for a chosen primitive k-th root ζ_k.
2. Let $t(x), y(x) \in \mathbb{Q}[x]$ mapping respectively to $\zeta_k + 1 \in K, (\zeta_k - 1)/\sqrt{-D} \in K$.

Brezing-Weng families.

1. Find $r(x) \in \mathbb{Z}[x]$ such that $K := \mathbb{Q}[x]/(r(x))$ is a number field containing $\sqrt{-D}$ and $\mathbb{Q}(\zeta_k)$ for a chosen primitive k-th root ζ_k.
2. Let $t(x), y(x) \in \mathbb{Q}[x]$ mapping respectively to $\zeta_k + 1 \in K, (\zeta_k - 1)/\sqrt{-D} \in K$.
3. Let $p(x) \in \mathbb{Q}[x]$ be given by $(t(x)^2 + Dy(x)^2)/4$.

Brezing-Weng families.

1. Find $r(x) \in \mathbb{Z}[x]$ such that $K := \mathbb{Q}[x]/(r(x))$ is a number field containing $\sqrt{-D}$ and $\mathbb{Q}(\zeta_k)$ for a chosen primitive k-th root ζ_k.
2. Let $t(x), y(x) \in \mathbb{Q}[x]$ mapping respectively to $\zeta_k + 1 \in K, (\zeta_k - 1)/\sqrt{-D} \in K$.
3. Let $p(x) \in \mathbb{Q}[x]$ be given by $(t(x)^2 + Dy(x)^2)/4$.

Choose $x_0 \in \mathbb{Z}$ such that $p = p(x_0), t = t(x_0)$ and $r = r(x_0)$ lead to a pairing friendly curve of embedding degree k of discriminant D.

Brezing-Weng families.

1. Find $r(x) \in \mathbb{Z}[x]$ such that $K := \mathbb{Q}[x]/(r(x))$ is a number field containing $\sqrt{-D}$ and $\mathbb{Q}(\zeta_k)$ for a chosen primitive k-th root ζ_k.
2. Let $t(x), y(x) \in \mathbb{Q}[x]$ mapping respectively to $\zeta_k + 1 \in K, (\zeta_k - 1)/\sqrt{-D} \in K$.
3. Let $p(x) \in \mathbb{Q}[x]$ be given by $(t(x)^2 + Dy(x)^2)/4$.

Choose $x_0 \in \mathbb{Z}$ such that $p = p(x_0), t = t(x_0)$ and $r = r(x_0)$ lead to a pairing friendly curve of embedding degree k of discriminant D.

Example (BN curves).

Brezing-Weng families.

1. Find $r(x) \in \mathbb{Z}[x]$ such that $K := \mathbb{Q}[x]/(r(x))$ is a number field containing $\sqrt{-D}$ and $\mathbb{Q}(\zeta_k)$ for a chosen primitive k-th root ζ_k.
2. Let $t(x), y(x) \in \mathbb{Q}[x]$ mapping respectively to $\zeta_k + 1 \in K, (\zeta_k - 1)/\sqrt{-D} \in K$.
3. Let $p(x) \in \mathbb{Q}[x]$ be given by $(t(x)^2 + Dy(x)^2)/4$.

Choose $x_0 \in \mathbb{Z}$ such that $p = p(x_0), t = t(x_0)$ and $r = r(x_0)$ lead to a pairing friendly curve of embedding degree k of discriminant D.

Example (BN curves). Barreto-Naehrig curves are elliptic curves of embedding degree $k = 12$ with

$$p = 36x_0^4 + 36x_0^3 + 24x_0^2 + 6x_0 + 1$$

$$r = 36x_0^4 + 36x_0^3 + 18x_0^2 + 6x_0 + 1$$

$$t = 6x_0^2 + 1$$

BN Miller loop.

Compression for $\mathbb{G}_2$.

BN Miller loop.

Compression for $\mathbb{G}_2$.

When k is even, for $u \in \mathbb{F}_p$ non-square,

$$\begin{array}{rcl} y^2 = x^3 + Ax + B : E/\mathbb{F}_p & \xrightarrow{\sim} & {}^tE/\mathbb{F}_p : uy^2 = x^3 + Ax + B \\ (x, y) & \longmapsto & (x, \sqrt{u}y) \end{array}$$

BN Miller loop.

Compression for $\mathbb{G}_2$.

When k is even, for $u \in \mathbb{F}_p$ non-square,

$$\begin{array}{rcl} y^2 = x^3 + Ax + B : E/\mathbb{F}_p & \xrightarrow{\sim} & {}^tE/\mathbb{F}_p : uy^2 = x^3 + Ax + B \\ (x, y) & \longmapsto & (x, \sqrt{u}y) \end{array}$$

$$E(\mathbb{F}_{p^{12}})[r] \simeq {}^tE(\mathbb{F}_{p^6})[r]$$

BN Miller loop.

Compression for $\mathbb{G}_2$*.*

When k is even, for $u \in \mathbb{F}_p$ non-square,

$$\begin{array}{rcl} y^2 = x^3 + Ax + B : E/\mathbb{F}_p & \xrightarrow{\sim} & {}^tE/\mathbb{F}_p : uy^2 = x^3 + Ax + B \\ (x, y) & \longmapsto & (x, \sqrt{u}y) \end{array}$$

$$E(\mathbb{F}_{p^{12}})[r] \simeq {}^tE(\mathbb{F}_{p^6})[r]$$

More automorphisms for $j = 0$ and 1728 curves. Compression by a factor 4 or 6.

BN Miller loop.

Compression for $\mathbb{G}_2$.

When k is even, for $u \in \mathbb{F}_p$ non-square,

$$\begin{array}{rcl} y^2 = x^3 + Ax + B : E/\mathbb{F}_p & \xrightarrow{\sim} & {}^tE/\mathbb{F}_p : uy^2 = x^3 + Ax + B \\ (x, y) & \longmapsto & (x, \sqrt{u}y) \end{array}$$

$$E(\mathbb{F}_{p^{12}})[r] \simeq {}^tE(\mathbb{F}_{p^6})[r]$$

More automorphisms for $j = 0$ and 1728 curves. Compression by a factor 4 or 6.

$$f = \left(1^2 \cdot \ell_{Q,Q}(P)/v_{2Q}(P)\right)^2 \cdot \ell_{2Q,2Q}(P)/v_{4Q}(P) \cdot \ell_{4Q,Q}(P)/v_{5Q}(P)$$

BN Miller loop.

Compression for $\mathbb{G}_2$.

When k is even, for $u \in \mathbb{F}_p$ non-square,

$$\begin{array}{rcl} y^2 = x^3 + Ax + B : E/\mathbb{F}_p & \xrightarrow{\sim} & {}^tE/\mathbb{F}_p : uy^2 = x^3 + Ax + B \\ (x, y) & \longmapsto & (x, \sqrt{u}y) \end{array}$$

$$E(\mathbb{F}_{p^{12}})[r] \simeq {}^tE(\mathbb{F}_{p^6})[r]$$

More automorphisms for $j = 0$ and 1728 curves. Compression by a factor 4 or 6.

$$f = \left(1^2 \cdot \ell_{Q,Q}(P)/v_{2Q}(P)\right)^2 \cdot \ell_{2Q,2Q}(P)/v_{4Q}(P) \cdot \ell_{4Q,Q}(P)/v_{5Q}(P)$$

$$v_{2Q}(P)^{\frac{p^k-1}{r}} = v_{4Q}(P)^{\frac{p^k-1}{r}} = v_{5Q}(P)^{\frac{p^k-1}{r}} = 1$$

BN Miller loop.

Compression for $\mathbb{G}_2$*.*

When k is even, for $u \in \mathbb{F}_p$ non-square,

$$\begin{array}{rcl} y^2 = x^3 + Ax + B : E/\mathbb{F}_p & \xrightarrow{\sim} & {}^tE/\mathbb{F}_p : uy^2 = x^3 + Ax + B \\ (x, y) & \longmapsto & (x, \sqrt{u}y) \end{array}$$

$$E(\mathbb{F}_{p^{12}})[r] \simeq {}^tE(\mathbb{F}_{p^6})[r]$$

More automorphisms for $j = 0$ and 1728 curves. Compression by a factor 4 or 6.

$$f = \left(1^2 \cdot \ell_{Q,Q}(P)\right)^2 \cdot \ell_{2Q,2Q}(P) \cdot \ell_{4Q,Q}(P)$$

BN final exponentiation.

$$\frac{p^{12}-1}{r} = (p^6-1)(p^2+1)\frac{p^4-p^2+1}{r}$$

BN final exponentiation.

$$\frac{p^{12}-1}{r} = (p^6-1)(p^2+1)\frac{p^4-p^2+1}{r}$$

$y = (x^{p^6-1})^{p^2+1}$ is easy with Frobenius powers.

BN final exponentiation.

$$\frac{p^{12}-1}{r} = (p^6-1)(p^2+1)\frac{p^4-p^2+1}{r}$$

$y = (x^{p^6-1})^{p^2+1}$ is easy with Frobenius powers.

$\frac{p^4-p^2+1}{r}$ is specific because $p = p(x_0)$ and $r = r(x_0)$.

BN final exponentiation.

$$\frac{p^{12}-1}{r} = (p^6-1)(p^2+1)\frac{p^4-p^2+1}{r}$$

$y = (x^{p^6-1})^{p^2+1}$ is easy with Frobenius powers.

$\frac{p^4-p^2+1}{r}$ is specific because $p = p(x_0)$ and $r = r(x_0)$.

$y^{\frac{p(x_0)^4-p(x_0)^2+1}{r(x_0)}} = y^{p^3+\lambda_2(x_0)p^2+\lambda_1(x_0)p+\lambda_0(x_0)}$: few exponentiations by x_0.

BN final exponentiation.

$$\frac{p^{12}-1}{r} = (p^6-1)(p^2+1)\frac{p^4-p^2+1}{r}$$

$y = (x^{p^6-1})^{p^2+1}$ is easy with Frobenius powers.

$\frac{p^4-p^2+1}{r}$ is specific because $p = p(x_0)$ and $r = r(x_0)$.

$y^{\frac{p(x_0)^4-p(x_0)^2+1}{r(x_0)}} = y^{p^3+\lambda_2(x_0)p^2+\lambda_1(x_0)p+\lambda_0(x_0)}$: few exponentiations by x_0.

Efficient pairing. But how secure are these curves ?

Security of pairing curves.

$$e : E(\mathbb{F}_p) \times E(\mathbb{F}_{p^k}) \longrightarrow \mathbb{F}_{p^k}$$

Security of pairing curves.

$$e : E(\mathbb{F}_p) \times E(\mathbb{F}_{p^k}) \longrightarrow \mathbb{F}_{p^k}$$

- Security against DLP in elliptic curve: best attack in $\mathcal{O}(\sqrt{r})$.
  $\log_2(r) = 256$ for 128 bits of security.

Security of pairing curves.

$$e : E(\mathbb{F}_p) \times E(\mathbb{F}_{p^k}) \longrightarrow \mathbb{F}_{p^k}$$

- Security against DLP in elliptic curve: best attack in $\mathcal{O}(\sqrt{r})$.
  $\log_2(r) = 256$ for 128 bits of security.
- Security against DLP in $\mathbb{F}_{p^k}$: Number Field Sieve attacks in progress.
  special prime p $\Longrightarrow$ 1993: Special NFS attack
  $k > 1$ $\Longrightarrow$ 2015: Tower NFS attack
  composite k and special p $\Longrightarrow$ 2016: STNFS attack

Security of pairing curves.

$$e : E(\mathbb{F}_p) \times E(\mathbb{F}_{p^k}) \longrightarrow \mathbb{F}_{p^k}$$

- Security against DLP in elliptic curve: best attack in $\mathcal{O}(\sqrt{r})$.
  $\log_2(r) = 256$ for 128 bits of security.
- Security against DLP in $\mathbb{F}_{p^k}$: Number Field Sieve attacks in progress.
  special prime p $\Longrightarrow$ 1993: Special NFS attack
  $k > 1$ $\Longrightarrow$ 2015: Tower NFS attack
  composite k and special p $\Longrightarrow$ 2016: STNFS attack

BN curves are threatened by STNFS... 
Need a 5500 bits field $\mathbb{F}_{p^{12}}$ to get 128 bits of security.

Curves of embedding degree 1. [eprint 2016/403]

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{F}_p$$

Curves of embedding degree 1. [eprint 2016/403]

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{F}_p$$

No NFS variants on $\mathbb{F}_p$: $|\mathbb{F}_p| \approx 2^{3072}$ is small 

Curves of embedding degree 1. [eprint 2016/403]

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{F}_p$$

No NFS variants on $\mathbb{F}_p$: $|\mathbb{F}_p| \approx 2^{3072}$ is small

p is very large, only the Tate pairing on these curves: not efficient

Generation of curves with given prime k, square-free D and no structure on p.

Algorithm: $\textsc{Cocks-Pinch}(k, D)$ – Compute a pairing-friendly curve $E/\mathbb{F}_p$ of trace t with a subgroup of order r, such that $t^2 - Dy^2 = 4p$.

Set a prime r such that $k \mid r - 1$ and $\sqrt{-D} \in \mathbb{F}_r$
Set T such that $r \mid \Phi_k(T)$
$t \leftarrow T + 1$
$y \leftarrow (t - 2)/\sqrt{-D}$
Lift $t, y \in \mathbb{Z}$ such that $t^2 + Dy^2 \equiv 0 \mod 4$
$p \leftarrow (t^2 + Dy^2)/4$
if p is prime **then return** $[p, t, y, r]$ **else** Repeat with another r.

Generation of curves with given prime k, square-free D and no structure on p.

Algorithm: $\textsc{Cocks-Pinch}(k, D)$ – Compute a pairing-friendly curve $E/\mathbb{F}_p$ of trace t with a subgroup of order r, such that $t^2 - Dy^2 = 4p$.

Set a prime r such that $k \mid r - 1$ and $\sqrt{-D} \in \mathbb{F}_r$
Set T such that $r \mid \Phi_k(T)$
$t \leftarrow T + 1$
$y \leftarrow (t - 2)/\sqrt{-D}$
Lift $t, y \in \mathbb{Z}$ such that $t^2 + Dy^2 \equiv 0 \mod 4$
$p \leftarrow (t^2 + Dy^2)/4$
if p is prime **then return** $[p, t, y, r]$ **else** Repeat with another r.

Large trace $t \implies$ the ate pairing is not very efficient 

Generation of curves with given prime k, square-free D and no structure on p.

Algorithm: $\textsc{Cocks-Pinch}(k, D)$ – Compute a pairing-friendly curve $E/\mathbb{F}_p$ of trace t with a subgroup of order r, such that $t^2 - Dy^2 = 4p$.

Set a small T
Set a prime r such that $k \mid r-1$, $\sqrt{-D} \in \mathbb{F}_r$ and $r \mid \Phi_k(T)$
$t \leftarrow T + 1$
$y \leftarrow (t-2)/\sqrt{-D}$
Lift $t, y \in \mathbb{Z}$ such that $t^2 + Dy^2 \equiv 0 \mod 4$
$p \leftarrow (t^2 + Dy^2)/4$
if p is prime **then return** $[p, t, y, r]$ **else** Repeat with another r.

Large trace $t \Longrightarrow$ the ate pairing is not very efficient

Fix: first fix a small T and then choose r. $t = T + 1$ is small

Generation of curves with given prime k, square-free D and no structure on p.

Algorithm: $\textsc{Cocks-Pinch}(k, D)$ – Compute a pairing-friendly curve $E/\mathbb{F}_p$ of trace t with a subgroup of order r, such that $t^2 - Dy^2 = 4p$.

Set a small T
Set a prime r such that $k \mid r-1$, $\sqrt{-D} \in \mathbb{F}_r$ and $r \mid \Phi_k(T)$
$t \leftarrow T+1$
$y \leftarrow (t-2)/\sqrt{-D}$
Lift $t, y \in \mathbb{Z}$ such that $t^2 + Dy^2 \equiv 0 \mod 4$
$p \leftarrow (t^2 + Dy^2)/4$
if p is prime and $p = 1 \bmod k$ **then return** $[p, t, y, r]$ **else** Repeat with another r.

Large trace $t \implies$ the ate pairing is not very efficient

Fix: first fix a small T and then choose r. $t = T+1$ is small $\mathbb{F}_{p^k} = \mathbb{F}_p[u]/(u^k - \alpha)$

Parameter choices for 128 bits of security.

- $k = 5$:

Parameter choices for 128 bits of security.

- $k = 5$: $D \simeq 10^{10}$,

Parameter choices for 128 bits of security.

- $k = 5$: $D \simeq 10^{10}$, $\Phi_k(T) = r \implies \log_2(T) = 256/\varphi(k)$ (sparse)

Parameter choices for 128 bits of security.

- $k = 5$: $D \simeq 10^{10}$, $\Phi_k(T) = r \Longrightarrow \log_2(T) = 256/\varphi(k)$ (sparse)
  NFS: $|\mathbb{F}_{p^5}| \approx 2^{3318} \Longrightarrow \log_2(p) = 664$

Parameter choices for 128 bits of security.

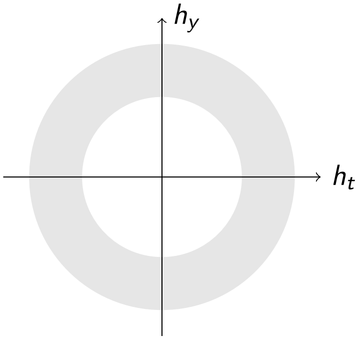
- $k = 5$: $D \simeq 10^{10}$, $\Phi_k(T) = r \Longrightarrow \log_2(T) = 256/\varphi(k)$ (sparse)
  NFS: $|\mathbb{F}_{p^5}| \approx 2^{3318} \Longrightarrow \log_2(p) = 664$

$$2^{663} \leq \frac{1}{4}\left((t + h_t \cdot r)^2 + D(y + h_y \cdot r)^2\right) < 2^{664}$$

Parameter choices for 128 bits of security.

- $k = 5$: $D \simeq 10^{10}$, $\Phi_k(T) = r \Longrightarrow \log_2(T) = 256/\varphi(k)$ (sparse)
  NFS: $|\mathbb{F}_{p^5}| \approx 2^{3318} \Longrightarrow \log_2(p) = 664$

$$2^{663} \leq \frac{1}{4}\left((t + h_t \cdot r)^2 + D(y + h_y \cdot r)^2\right) < 2^{664}$$

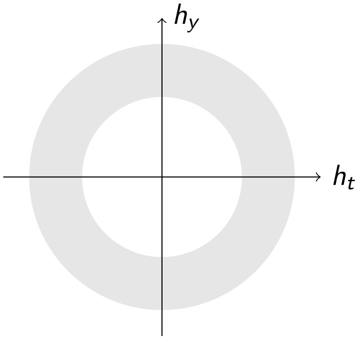


Parameter choices for 128 bits of security.

- $k = 5$: $D \simeq 10^{10}$, $\Phi_k(T) = r \Longrightarrow \log_2(T) = 256/\varphi(k)$ (sparse)
  NFS: $|\mathbb{F}_{p^5}| \approx 2^{3318} \Longrightarrow \log_2(p) = 664$

$$2^{663} \leq \frac{1}{4}\left((t + h_t \cdot r)^2 + D(y + h_y \cdot r)^2\right) < 2^{664}$$

Choose $\log_2(h_y) = 61$ so $\log_2(p) = 664$.



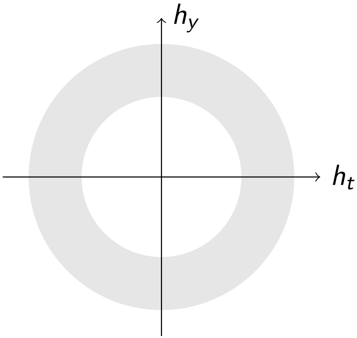
Parameter choices for 128 bits of security.

- $k = 5$: $D \simeq 10^{10}$, $\Phi_k(T) = r \Longrightarrow \log_2(T) = 256/\varphi(k)$ (sparse)
  NFS: $|\mathbb{F}_{p^5}| \approx 2^{3318} \Longrightarrow \log_2(p) = 664$

$$2^{663} \leq \frac{1}{4}\left((t + h_t \cdot r)^2 + D(y + h_y \cdot r)^2\right) < 2^{664}$$

Choose $\log_2(h_y) = 61$ so $\log_2(p) = 664$.

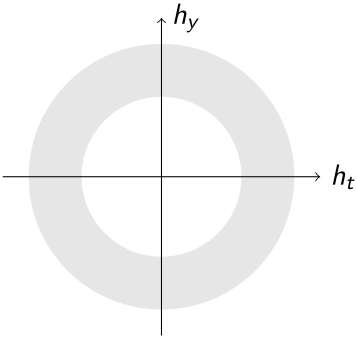
Large discriminant, small finite field $\mathbb{F}_{p^k}$



Parameter choices for 128 bits of security.

- $k = 5$: $D \simeq 10^{10}$, $\Phi_k(T) = r \Longrightarrow \log_2(T) = 256/\varphi(k)$ (sparse)
  NFS: $|\mathbb{F}_{p^5}| \approx 2^{3318} \Longrightarrow \log_2(p) = 664$

$$2^{663} \leq \frac{1}{4}\left((t + h_t \cdot r)^2 + D(y + h_y \cdot r)^2\right) < 2^{664}$$



Choose $\log_2(h_y) = 61$ so $\log_2(p) = 664$.

Large discriminant, small finite field $\mathbb{F}_{p^k}$

Large p, no compression for $\mathbb{G}_2$

Parameter choices for 128 bits of security.

- $k = 7$:

Parameter choices for 128 bits of security.

- $k = 7$: $\log_2(T) = 256/\varphi(7) = 43$

Parameter choices for 128 bits of security.

- $k = 7$: $\log_2(T) = 256/\varphi(7) = 43$
  NFS: $|\mathbb{F}_{p^7}| \approx 2^{3584} \Longrightarrow \log_2(p) = 512$

Parameter choices for 128 bits of security.

- $k = 7$: $\log_2(T) = 256/\varphi(7) = 43$
  NFS: $|\mathbb{F}_{p^7}| \approx 2^{3584} \Longrightarrow \log_2(p) = 512$
  Small D because $\log_2(y^2) \approx 512$.

Parameter choices for 128 bits of security.

- $k = 7$: $\log_2(T) = 256/\varphi(7) = 43$
  NFS: $|\mathbb{F}_{p^7}| \approx 2^{3584} \implies \log_2(p) = 512$
  Small D because $\log_2(y^2) \approx 512$.

  512-bit p, small finite field $\mathbb{F}_{p^k}$, small Miller length

Parameter choices for 128 bits of security.

- $k = 7$: $\log_2(T) = 256/\varphi(7) = 43$
  NFS: $|\mathbb{F}_{p^7}| \approx 2^{3584} \Longrightarrow \log_2(p) = 512$
  Small D because $\log_2(y^2) \approx 512$.

  512-bit p, small finite field $\mathbb{F}_{p^k}$, small Miller length

  No compression for $\mathbb{G}_2$, not efficient

Parameter choices for 128 bits of security.

- $k = 7$: $\log_2(T) = 256/\varphi(7) = 43$
  NFS: $|\mathbb{F}_{p^7}| \approx 2^{3584} \implies \log_2(p) = 512$
  Small D because $\log_2(y^2) \approx 512$.

  512-bit p, small finite field $\mathbb{F}_{p^k}$, small Miller length 

  No compression for $\mathbb{G}_2$, not efficient 
- $k = 6$:

Parameter choices for 128 bits of security.

- $k = 7$: $\log_2(T) = 256/\varphi(7) = 43$
  NFS: $|\mathbb{F}_{p^7}| \approx 2^{3584} \implies \log_2(p) = 512$
  Small D because $\log_2(y^2) \approx 512$.

  512-bit p, small finite field $\mathbb{F}_{p^k}$, small Miller length

  No compression for $\mathbb{G}_2$, not efficient
- $k = 6$: $D = 3$,

Parameter choices for 128 bits of security.

- $k = 7$: $\log_2(T) = 256/\varphi(7) = 43$
  NFS: $|\mathbb{F}_{p^7}| \approx 2^{3584} \Longrightarrow \log_2(p) = 512$
  Small D because $\log_2(y^2) \approx 512$.

  512-bit p, small finite field $\mathbb{F}_{p^k}$, small Miller length 

  No compression for $\mathbb{G}_2$, not efficient 
- $k = 6$: $D = 3$, $\log_2(T) = 128$

Parameter choices for 128 bits of security.

- $k = 7$: $\log_2(T) = 256/\varphi(7) = 43$
  NFS: $|\mathbb{F}_{p^7}| \approx 2^{3584} \Longrightarrow \log_2(p) = 512$
  Small D because $\log_2(y^2) \approx 512$.

  512-bit p, small finite field $\mathbb{F}_{p^k}$, small Miller length 

  No compression for $\mathbb{G}_2$, not efficient 
- $k = 6$: $D = 3$, $\log_2(T) = 128$
  (T)NFS: $|\mathbb{F}_{p^6}| \approx 2^{4032} \Longrightarrow \log_2(p) = 672$

Parameter choices for 128 bits of security.

- $k = 7$: $\log_2(T) = 256/\varphi(7) = 43$
  NFS: $|\mathbb{F}_{p^7}| \approx 2^{3584} \Longrightarrow \log_2(p) = 512$
  Small D because $\log_2(y^2) \approx 512$.

  512-bit p, small finite field $\mathbb{F}_{p^k}$, small Miller length 

  No compression for $\mathbb{G}_2$, not efficient 

- $k = 6$: $D = 3$, $\log_2(T) = 128$
  (T)NFS: $|\mathbb{F}_{p^6}| \approx 2^{4032} \Longrightarrow \log_2(p) = 672$

  Factor 6 compression for $\mathbb{G}_2$: $Q \in {}^{t6}E(\mathbb{F}_p)$ 

Parameter choices for 128 bits of security.

- $k = 7$: $\log_2(T) = 256/\varphi(7) = 43$
  NFS: $|\mathbb{F}_{p^7}| \approx 2^{3584} \Longrightarrow \log_2(p) = 512$
  Small D because $\log_2(y^2) \approx 512$.

  512-bit p, small finite field $\mathbb{F}_{p^k}$, small Miller length

  No compression for $\mathbb{G}_2$, not efficient

- $k = 6$: $D = 3$, $\log_2(T) = 128$
  (T)NFS: $|\mathbb{F}_{p^6}| \approx 2^{4032} \Longrightarrow \log_2(p) = 672$

  Factor 6 compression for $\mathbb{G}_2$: $Q \in {}^{t6}E(\mathbb{F}_p)$

  Large p, large Miller length

Parameter choices for 128 bits of security.

- $k = 8$:

Parameter choices for 128 bits of security.

- $k = 8$: $D = 4$,

Parameter choices for 128 bits of security.

- $k = 8$: $D = 4$, $\log_2(T) = 64$

Parameter choices for 128 bits of security.

- $k = 8$: $D = 4$, $\log_2(T) = 64$
  $\sqrt{-D} = \sqrt{-4} = 2\sqrt{-1}$ and T is a 8-th root of unity so $2T^2 = \sqrt{-D} \bmod r$.

Parameter choices for 128 bits of security.

- $k = 8$: $D = 4$, $\log_2(T) = 64$
  $\sqrt{-D} = \sqrt{-4} = 2\sqrt{-1}$ and T is a 8-th root of unity so $2T^2 = \sqrt{-D}$ mod r.

$$y = (T-2)/\sqrt{-D} = (T-2)/(2T^2) = -(T-2)\cdot T^2/2$$

After lifting in $\mathbb{Z}$, p is a polynomial in T, h_t and h_y. If $h_t, h_y \in \{0, 1, -1\}$, p is a univariate polynomial in T and **S**TNFS can exploit this property !

Parameter choices for 128 bits of security.

- $k = 8$: $D = 4$, $\log_2(T) = 64$
  $\sqrt{-D} = \sqrt{-4} = 2\sqrt{-1}$ and T is a 8-th root of unity so $2T^2 = \sqrt{-D} \bmod r$.

  $$y = (T-2)/\sqrt{-D} = (T-2)/(2T^2) = -(T-2)\cdot T^2/2$$

  After lifting in $\mathbb{Z}$, p is a polynomial in T, h_t and h_y. If $h_t, h_y \in \{0, 1, -1\}$, p is a univariate polynomial in T and **S**TNFS can exploit this property !
  We lift $y \leftarrow y + h_y \cdot r$ with $\log_2(h_y) = 16$ so that SNFS cannot exploit it.

Parameter choices for 128 bits of security.

- $k = 8$: $D = 4$, $\log_2(T) = 64$
  $\sqrt{-D} = \sqrt{-4} = 2\sqrt{-1}$ and T is a 8-th root of unity so $2T^2 = \sqrt{-D} \bmod r$.

  $$y = (T-2)/\sqrt{-D} = (T-2)/(2T^2) = -(T-2) \cdot T^2/2$$

  After lifting in $\mathbb{Z}$, p is a polynomial in T, h_t and h_y. If $h_t, h_y \in \{0, 1, -1\}$, p is a univariate polynomial in T and **S**TNFS can exploit this property !
  We lift $y \leftarrow y + h_y \cdot r$ with $\log_2(h_y) = 16$ so that SNFS cannot exploit it.
  (T)NFS: $|\mathbb{F}_{p^8}| \approx 2^{4349} \implies \log_2(p) = 544$

Parameter choices for 128 bits of security.

- $k = 8$: $D = 4$, $\log_2(T) = 64$
  $\sqrt{-D} = \sqrt{-4} = 2\sqrt{-1}$ and T is a 8-th root of unity so $2T^2 = \sqrt{-D} \bmod r$.

$$y = (T-2)/\sqrt{-D} = (T-2)/(2T^2) = -(T-2)\cdot T^2/2$$

After lifting in $\mathbb{Z}$, p is a polynomial in T, h_t and h_y. If $h_t, h_y \in \{0, 1, -1\}$, p is a univariate polynomial in T and **S**TNFS can exploit this property !
We lift $y \leftarrow y + h_y \cdot r$ with $\log_2(h_y) = 16$ so that SNFS cannot exploit it.
(T)NFS: $|\mathbb{F}_{p^8}| \approx 2^{4349} \Longrightarrow \log_2(p) = 544$
Factor 4 compression: $Q \in {}^{t4}E(\mathbb{F}_{p^2})$

Example of generation for $k = 8$.

Code is available at `https://gitlab.inria.fr/smasson/cocks-pinch-variant`.

Example of generation for $k = 8$.

Code is available at https://gitlab.inria.fr/smasson/cocks-pinch-variant.

$D = 4$ (automorphism of degree 4)

Example of generation for $k = 8$.

Code is available at https://gitlab.inria.fr/smasson/cocks-pinch-variant.

$D = 4$ (automorphism of degree 4)

$\log_2(T) = 64$ with small Hamming weight

Example of generation for $k = 8$.

Code is available at https://gitlab.inria.fr/smasson/cocks-pinch-variant.

$D = 4$ (automorphism of degree 4)

$\log_2(T) = 64$ with small Hamming weight

Lift t and y with 16-bit h_t and h_y, and restrict on $\log_2(p) = 544$

Example of generation for $k = 8$.

Code is available at https://gitlab.inria.fr/smasson/cocks-pinch-variant.

$D = 4$ (automorphism of degree 4)

$\log_2(T) = 64$ with small Hamming weight

Lift t and y with 16-bit h_t and h_y, and restrict on $\log_2(p) = 544$

Accept small cofactors for $E(\mathbb{F}_p)$, $E(\mathbb{F}_{p^8})$

Check subgroup-security and twist-subgroup-security.

Example of generation for $k = 8$.

Code is available at https://gitlab.inria.fr/smasson/cocks-pinch-variant.

$D = 4$ (automorphism of degree 4)

$\log_2(T) = 64$ with small Hamming weight

Lift t and y with 16-bit h_t and h_y, and restrict on $\log_2(p) = 544$

Accept small cofactors for $E(\mathbb{F}_p)$, $E(\mathbb{F}_{p^8})$

Check subgroup-security and twist-subgroup-security.

```
CocksPinchVariantResult(
    k=8,D=4,T=0xffffffffeff7c200,i=5,ht=5,hy=-0xd700,
    allowed_cofactor=420,allowed_size_cofactor=10,
    max_B1=600
    )
```

Example of generation for $k = 8$.

Code is available at https://gitlab.inria.fr/smasson/cocks-pinch-variant.

$D = 4$ (automorphism of degree 4)

$\log_2(T) = 64$ with small Hamming weight

Lift t and y with 16-bit h_t and h_y, and restrict on $\log_2(p) = 544$

Accept small cofactors for $E(\mathbb{F}_p)$, $E(\mathbb{F}_{p^8})$

Check subgroup-security and twist-subgroup-security.

```
CocksPinchVariantResult(
    k=8,D=4,T=0xffffffffeff7c200,i=5,ht=5,hy=-0xd700,
    allowed_cofactor=420,allowed_size_cofactor=10,
    max_B1=600
    )
```

Subgroup- and twist-subgroup- secure curves found for $k = 5, 6, 7$ and 8 !

Timings and comparisons

1. Tate and ate pairing
2. Pairing-friendly curves for 128 bits of security
3. Timings and comparisons

RELIC.[D. Aranha] available on github.com

RELIC.[D. Aranha] available on github.com
Efficient library for cryptography, state of the art for pairing computation.

RELIC.[D. Aranha] available on github.com
Efficient library for cryptography, state of the art for pairing computation.
Implementation for BN and BLS curves

RELIC.[D. Aranha] available on github.com
Efficient library for cryptography, state of the art for pairing computation.
Implementation for BN and BLS curves
Fast $\mathbb{F}_p$ arithmetic in assembly instructions for some given p.

RELIC.[D. Aranha] available on github.com
Efficient library for cryptography, state of the art for pairing computation.
Implementation for BN and BLS curves
Fast $\mathbb{F}_p$ arithmetic in assembly instructions for some given p.

How to compare curves.

RELIC.[D. Aranha] available on github.com
Efficient library for cryptography, state of the art for pairing computation.
Implementation for BN and BLS curves
Fast $\mathbb{F}_p$ arithmetic in assembly instructions for some given p.

How to compare curves.

1. Bench $\mathbb{F}_p$ arithmetic for different sizes of prime p

RELIC.[D. Aranha] available on github.com
Efficient library for cryptography, state of the art for pairing computation.
Implementation for BN and BLS curves
Fast $\mathbb{F}_p$ arithmetic in assembly instructions for some given p.

How to compare curves.

1. Bench $\mathbb{F}_p$ arithmetic for different sizes of prime p
2. Count the number of $\mathbb{F}_p$ multiplications for a pairing computation on each curve

RELIC.[D. Aranha] available on github.com

Efficient library for cryptography, state of the art for pairing computation.

Implementation for BN and BLS curves

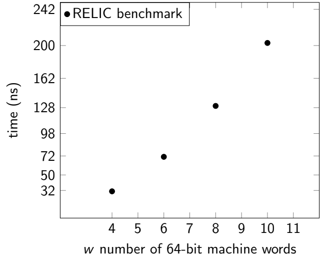
Fast $\mathbb{F}_p$ arithmetic in assembly instructions for some given p.

How to compare curves.

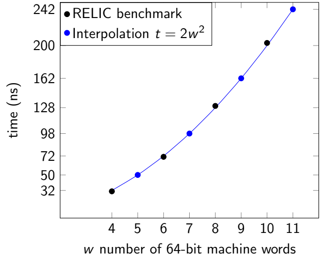
1. Bench $\mathbb{F}_p$ arithmetic for different sizes of prime p
2. Count the number of $\mathbb{F}_p$ multiplications for a pairing computation on each curve
3. Compare the estimated costs between the different curves.

Curve	64-bit words for p	$\mathbb{F}_p$ mult. timing
BN-462	□□□□□□□□	
BLS12-461	□□□□□□□□	
$k = 5$	□□□□□□□□□□□	
$k = 6$	□□□□□□□□□□□	
$k = 7$	□□□□□□□□	
$k = 8$	□□□□□□□□□	
$k = 1$	□ $\times$ 48	

Curve	64-bit words for p	$\mathbb{F}_p$ mult. timing
BN-462	□□□□□□□□	120ns
BLS12-461	□□□□□□□□	120ns
$k = 5$	□□□□□□□□□□□	
$k = 6$	□□□□□□□□□□□	
$k = 7$	□□□□□□□□	120ns
$k = 8$	□□□□□□□□□	
$k = 1$	□ $\times$ 48	

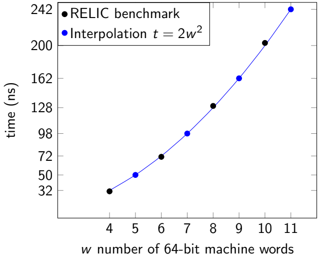


Curve	64-bit words for p	$\mathbb{F}_p$ mult. timing
BN-462	□□□□□□□□	120ns
BLS12-461	□□□□□□□□	120ns
$k = 5$	□□□□□□□□□□□	230ns*
$k = 6$	□□□□□□□□□□□	230ns*
$k = 7$	□□□□□□□□	120ns
$k = 8$	□□□□□□□□□	154ns*
$k = 1$	$\square \times 48$	



* Interpolation from the graph

Curve	**64-bit words for p**	**$\mathbb{F}_p$ mult. timing**
BN-462	□□□□□□□□	120ns
BLS12-461	□□□□□□□□	120ns
$k = 5$	□□□□□□□□□□□	230ns*
$k = 6$	□□□□□□□□□□□	230ns*
$k = 7$	□□□□□□□□	120ns
$k = 8$	□□□□□□□□□	154ns*
$k = 1$	□ $\times$ 48	4882ns**



* Interpolation from the graph

**Benchmark with GMP.

Pairing computation on BLS and BN curves.

- Automorphism of degree 6: $\mathbb{G}_2 \simeq {}^{t6}E(\mathbb{F}_{p^2})$
- Miller length: 117-bit for BN, 77-bit for BLS
- Efficient final exponentiation using $p = p(x_0)$ and $r = r(x_0)$
- Cyclotomic squarings

Pairing computation on BLS and BN curves.

- Automorphism of degree 6: $\mathbb{G}_2 \simeq {}^{t6}E(\mathbb{F}_{p^2})$
- Miller length: 117-bit for BN, 77-bit for BLS
- Efficient final exponentiation using $p = p(x_0)$ and $r = r(x_0)$
- Cyclotomic squarings

Curve	$\mathbb{F}_p$ mult. count	Estimated time
BN	17871	2.2ms
BLS	13878	1.6ms

Pairing computation on BLS and BN curves.

- Automorphism of degree 6: $\mathbb{G}_2 \simeq {}^{t6}E(\mathbb{F}_{p^2})$
- Miller length: 117-bit for BN, 77-bit for BLS
- Efficient final exponentiation using $p = p(x_0)$ and $r = r(x_0)$
- Cyclotomic squarings

Curve	$\mathbb{F}_p$ mult. count	Estimated time
BN	17871	2.2ms
BLS	13878	1.6ms

Benchmarks with RELIC: $\approx$ 10% of error 

Pairing computation on $k = 5$ and $k = 7$ curves.

- No compression: very large $\mathbb{G}_2 \simeq E(\mathbb{F}_{p^k})$
- Miller length: $\log_2(T) = 64$ or 43.
- Expensive final exponentiation (no structure on p). See gitlab.inria.fr.

Pairing computation on $k = 5$ and $k = 7$ curves.

- No compression: very large $\mathbb{G}_2 \simeq E(\mathbb{F}_{p^k})$
- Miller length: $\log_2(T) = 64$ or 43.
- Expensive final exponentiation (no structure on p). See gitlab.inria.fr.

Curve	$\mathbb{F}_p$ mult. count	Estimated time
$k = 5$	24373	5.6ms
$k = 7$	31793	3.8ms

Pairing computation on $k = 5$ and $k = 7$ curves.

- No compression: very large $\mathbb{G}_2 \simeq E(\mathbb{F}_{p^k})$
- Miller length: $\log_2(T) = 64$ or 43.
- Expensive final exponentiation (no structure on p). See gitlab.inria.fr.

Curve	$\mathbb{F}_p$ mult. count	Estimated time
$k = 5$	24373	5.6ms
$k = 7$	31793	3.8ms

Pairing estimations: very expensive 

Pairing computation on $k = 6$ curves.

- Automorphism of degree 6: $\mathbb{G}_2$ defined over $\mathbb{F}_p$
- Large Miller length: $\log_2(T) = 128$
- Final exponentiation faster than $k = 5$ and 7 (structure on p, cyclotomic squaring). See gitlab.inria.fr.

Pairing computation on $k = 6$ curves.

- Automorphism of degree 6: $\mathbb{G}_2$ defined over $\mathbb{F}_p$
- Large Miller length: $\log_2(T) = 128$
- Final exponentiation faster than $k = 5$ and 7 (structure on p, cyclotomic squaring). See gitlab.inria.fr.

Pairing computation on $k = 8$ curves.

- Automorphism of degree 4: $\mathbb{G}_2$ defined over $\mathbb{F}_{p^2}$
- Structure on p: $p = \frac{1}{4}((t + h_t \cdot r)^2 + 4(y + h_y \cdot r)^2) = p(T, h_t, h_y)$
  Cyclotomic squaring
  Fast final exponentiation. See gitlab.inria.fr.

Pairing computation on $k = 6$ curves.

- Automorphism of degree 6: $\mathbb{G}_2$ defined over $\mathbb{F}_p$
- Large Miller length: $\log_2(T) = 128$
- Final exponentiation faster than $k = 5$ and 7 (structure on p, cyclotomic squaring). See gitlab.inria.fr.

Pairing computation on $k = 8$ curves.

- Automorphism of degree 4: $\mathbb{G}_2$ defined over $\mathbb{F}_{p^2}$
- Structure on p: $p = \frac{1}{4}((t + h_t \cdot r)^2 + 4(y + h_y \cdot r)^2) = p(T, h_t, h_y)$
  Cyclotomic squaring
  Fast final exponentiation. See gitlab.inria.fr.

Curve	$\mathbb{F}_p$ mult. count	Estimated time
$k = 6$	8472	2.0ms
$k = 8$	11636	1.8ms

Pairing computation on $k = 6$ curves.

- Automorphism of degree 6: $\mathbb{G}_2$ defined over $\mathbb{F}_p$
- Large Miller length: $\log_2(T) = 128$
- Final exponentiation faster than $k = 5$ and 7 (structure on p, cyclotomic squaring). See gitlab.inria.fr.

Pairing computation on $k = 8$ curves.

- Automorphism of degree 4: $\mathbb{G}_2$ defined over $\mathbb{F}_{p^2}$
- Structure on p: $p = \frac{1}{4}((t + h_t \cdot r)^2 + 4(y + h_y \cdot r)^2) = p(T, h_t, h_y)$
  Cyclotomic squaring
  Fast final exponentiation. See gitlab.inria.fr.

Curve	$\mathbb{F}_p$ mult. count	Estimated time
$k = 6$	8472	2.0ms
$k = 8$	11636	1.8ms

Pairing estimations: competitive 

Curve	**Miller loop** time estimation	**Exponentiation** time estimation	time estimation
$k = 5$	3.3ms	2.3ms	5.6ms
$k = 6$	1.1ms	0.9ms	2.0ms
$k = 7$	2.2ms	1.6ms	3.8ms
$k = 8$	0.7ms	1.1ms	1.8ms
BN	1.5ms	0.7ms	2.2ms
BLS12	0.9ms	0.7ms	1.6ms
$k = 1$	22.7ms	20.0ms	42.7ms

Curve	**Miller loop** time estimation	**Exponentiation** time estimation	time estimation
$k = 5$	3.3ms	2.3ms	5.6ms
$k = 6$	1.1ms	0.9ms	2.0ms
$k = 7$	2.2ms	1.6ms	3.8ms
$k = 8$	0.7ms	1.1ms	1.8ms
BN	1.5ms	0.7ms	2.2ms
BLS12	0.9ms	0.7ms	1.6ms
$k = 1$	22.7ms	20.0ms	42.7ms

Thank you for your attention.