Introduction to Pairings

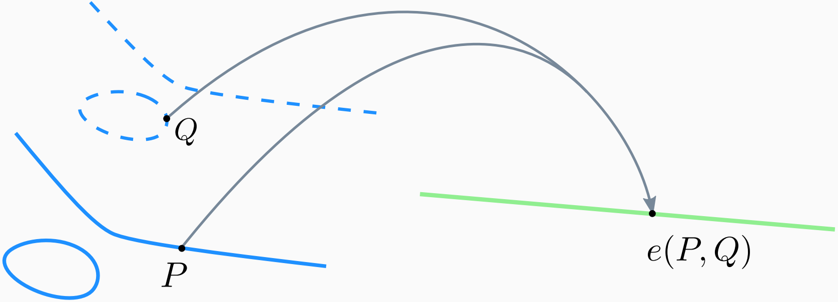
ECC "Summer" School

Diego F. Aranha

November 12, 2017

Institute of Computing – University of Campinas

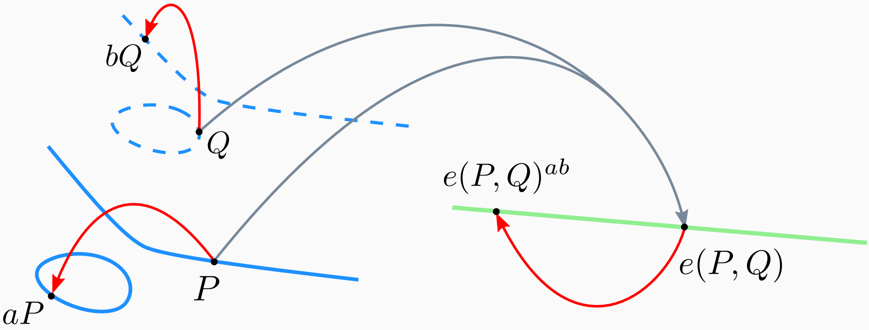
What is a pairing?



Why a bilinear pairing?

$$e(P+R,Q) = e(P,Q)\cdot e(R,Q) \text{ and } e(P,Q+S) = e(P,Q)\cdot e(P,S)$$

.



Introduction

Elliptic Curve Cryptography (ECC):

- Underlying problem **harder** than integer factoring (RSA)
- Same security level with **smaller** parameters
- Efficiency in storage (**short** keys) and execution time

Pairing-Based Cryptography (PBC):

- Initially **destructive**
- Allows for **innovative** protocols
- Makes curve-based cryptography more **flexible**

Introduction

Pairing-Based Cryptography (PBC) enables many elegant solutions to cryptographic problems:

- **Implicit certification schemes** (IBC, CLPKC, etc.)
- **Short signatures** (in group elements, BLS, BBS)
- **More efficient key agreements** (Joux's 3DH, NIKDS)
- **Low-depth homomorphic encryption** (BGN and variants)
- **Isogeny-based cryptography** (although not postquantum)

Pairing computation is the **most expensive** operation in PBC.

Net week: State-of-the art techniques to make it faster!

Elliptic curves

An **elliptic curve** is the set of solutions $(x, y) \in \mathbb{F}_{q^m} \times \mathbb{F}_{q^m}$ that satisfy the Weierstrass equation

$$E : y^2 = x^3 + ax + b$$

where $a, b \in \mathbb{F}_{q^m}$ with $\Delta \neq 0$, and a **point at infinity** ∞.

A degree d **twist** E' of E is a curve isomorphic to E over the algebraic closure of $\mathbb{F}_{q^m}$. The only *possible* degrees for elliptic curves are $d \in \{2, 3, 4, 6\}$.

Important: Very convenient mathematical setting where pairings can be constructed and evaluated efficiently.

Elliptic curves

Definitions

The order n of the curve is the number of points that satisfy the curve equation.

The **Hasse condition** states that $n = q^m + 1 - t$, $|t| \leq 2\sqrt{q^m}$.

The curve is **supersingular** when q divides t.

More definitions

The **order** of point P is the smallest integer r such that $rP = \infty$. We always have $r|n$.

The r-**torsion subgroup** $(E(\mathbb{F}_{q^m})[r])$ is the set of points P in which their order divides r.

Bilinear pairings

Let $\mathbb{G}_1 = \langle P \rangle$ and $\mathbb{G}_2 = \langle Q \rangle$ be additive groups and $\mathbb{G}_T$ be a multiplicative group such that $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T| =$ prime r.

Definition

An efficiently-computable map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an **admissible bilinear map** if the following properties are satisfied:

1. *Bilinearity:* given $(V, W) \in \mathbb{G}_1 \times \mathbb{G}_2$ and $(a, b) \in \mathbb{Z}_r^*$:

   $$e(aV, bW) = e(V, W)^{ab} = e(abV, W) = e(V, abW) = e(bV, aW).$$

2. *Non-degeneracy:* $e(P, Q) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_T}$ in $\mathbb{G}_T$.

Bilinear pairings

A general pairing

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

- $\mathbb{G}_1$ is typically a subgroup of $E(\mathbb{F}_q)$.
- $\mathbb{G}_2$ is typically a subgroup of $E(\mathbb{F}_{q^k})$.
- $\mathbb{G}_T$ is a multiplicative subgroup of $\mathbb{F}_{q^k}^*$.

Hence pairing-based cryptography involves arithmetic in $\mathbb{F}_{q^k}$.

Problem: In practice, we want small k for computable pairing!

Pairing-friendly curves

Definitions

The **embedding degree** of the curve is the smallest integer k such that $r|(q^k - 1)$.

In other words, it is the **smallest** extension of $\mathbb{F}_q$ in which we can **embed** the r-torsion group. For efficiency, we want the largest d such that $d|k$.

Random curves have $k \approx q$, but supersingular curves have $k \leq 6$ and there are **families** of ordinary curves with $k < 50$.

Pairing operations

A general pairing

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

Cryptographic schemes require multiple operations in pairing groups:

1. **Scalar multiplication**, **membership**, **compression** in $\mathbb{G}_1$ and $\mathbb{G}_2$.
2. **Exponentiation**, **membership**, **compression** in $\mathbb{G}_T$.
3. **Hashing** strings into groups $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$.
4. **Efficient maps** between $\mathbb{G}_1$ and $\mathbb{G}_2$.
5. Efficient **pairing computation**.

Problem: No concrete instantiation supports last three simultaneously!

Pairing types

If $\mathbb{G}_1 = \mathbb{G}_2$, the pairing is symmetric (or Type-1) and defined over a supersingular curve equipped with a **distortion map** $\psi : E(\mathbb{F}_q)[r] \rightarrow E(\mathbb{F}_{q^k})[r]$.

If $\mathbb{G}_1 \neq \mathbb{G}_2$, the pairing is asymmetric (or Type-3) and $\mathbb{G}_2$ is chosen as the group of points in the **twist** that is isomorphic to a subgroup of $E(\mathbb{F}_{q^k})[r]$. There is no **efficient** map $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$.

Important: Supersingular curves over small characteristic ($q = 2, 3$) are **broken** by quasi-polynomial algorithm by [Barbulescu et al. 2014]!

Security of pairings

A general pairing

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

Classical problems:

- **DLP**: Recover a from $\langle g, g^a \rangle$
- **CDHP**: Compute g^{ab} from $\langle g, g^a, g^b \rangle$

Underlying problems:

- **ECDLP**: Recover a from $\langle P, aP \rangle$
- **BCDHP**: Compute $e(P, Q)^{abc}$ from $\langle P, aP, bP, cP, Q, aQ, bQ, cQ \rangle$

Security of pairings

There are multiple security requirements to satisfy:

- The (EC)DLP problem must be hard in $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$.
- Parameters in $\mathbb{G}_1, \mathbb{G}_2$ should be large enough.
- Good balance can be found by choosing the **right** k.

The value $\rho = \frac{\log q}{\log r}$ describes how good the balance is ($\rho = 1$ is optimal) for a certain set of parameters.

Important: Plenty **research** into suitable curves for good values of k.

Applications

The first cryptographic application of pairings was attacking ECDLP!

The Menezes-Okamoto-Vanstone (MOV) attack

Given P and $Q = aP$ on curve E, find a:

1. Find point S of order n such that $e(P, Q) \neq 1_{\mathbb{G}_T}$.
2. Compute $e(P, S) = g$.
3. Compute $e(Q, S) = e(aP, S) = e(P, S)^a = g^a$.
4. Solve the DLP on $\langle g, g^a \rangle$ in $\mathbb{G}_T$.

Best general known algorithms for ECDLP run in $O(\sqrt{n})$, but there are **subexponential methods** such as *index calculus* for DLP in $\mathbb{G}_T$.

Note: this attacked **killed** the faster supersingular curves in the 90s.

Applications

Conventional paradigm (PKI):

- Three-party key agreement [Joux 2000]
- Short signatures [Boneh et al. 2001]

Alternate paradigms:

- Non-interactive identity-based AKE [Sakai et al. 2001]
- Identity-based encryption [Boneh et al., Sakai et al. 2001]

Applications

Joux's one-round Tripartite Diffie-Hellman [Joux 2000]:

- **Key generation**:
  1. Parties A, B, C generate short-lived secrets $a, b, c \in \mathbb{Z}_r^*$ respectively
  2. Parties A, B, C broadcast aG, bG, cG to the other parties

- **Key sharing**:
  1. A computes $K_A = e(bG, cG)^a$
  2. B computes $K_B = e(aG, cG)^b$
  3. C computes $K_C = e(aG, bG)^c$

Correctness: Shared key is $K = K_A = K_B = K_C = e(G, G)^{abc}$.

Applications

Boneh-Lynn-Schacham (BLS) short signatures in the conventional PKI paradigm [Boneh et al. 2001]:

- **Key generation**:
  1. Select a private key $x \in \mathbb{Z}_r^*$
  2. Compute the public key $V \leftarrow xP$
- **Signature**:
  1. Compute $H \leftarrow h(M) \in \mathbb{G}_1$
  2. Sign $S \leftarrow xH$
- **Verification**:
  1. Compute $H \leftarrow h(M)$
  2. Verify if $e(P, S) = e(V, H)$

Correctness: Works because $e(P, S) = e(P, xH) = e(xP, H) = e(V, H)$.

Applications

Identity-based encryption **facilitates** certification of public keys. If Alice wants to encrypt a message to Bob, she must be sure that an adversary did not **replace** his public key.

Conventional: Employ a *Certificate Authority* (CA) to compute a **signature** linking Bob and his public key. Alice can check the signature and learns Bob's public key.

However, certificates are **expensive** to manage (procedures, audits, revocation), thus Alice could use some trivially authentic information about Bob (e-mail address?).

Solution: Introduce authority to generate and distribute private keys.

Applications

Non-interactive identity-based AKE [Sakai et al. 2001]:

- **Initialization**:
  1. Central authority generates master key $s \in \mathbb{Z}_r^*$.
- **Key generation**:
  1. User with identity ID_i computes $P_i = H(ID_i)$
  2. Central authority generates private key $S_i = sP_i$
- **Key derivation**:
  1. Users A e B compute shared key $e(S_A, P_B) = e(S_B, P_A)$

Correctness: $e(S_A, P_B) = e(sP_A, P_B) = e(P_A, sP_B) = e(S_B, P_A)$.

Applications

Identity-based encryption [Boneh and Franklin 2001]:

- **Initialization**:
  1. Authority (PKG) generates master key $s \in \mathbb{Z}_r^*$ and computes its public key $P_{pub} = sP$
  2. Fix hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_1$ and $H_2 : \mathbb{G}_T \to \{0,1\}^m$.
- **Key generation**:
  1. User with identity ID_i computes public key $P_i = H_1(ID_i)$
  2. Central authority generates private key $S_i = sP_i$
- **Encryption**:
  1. To encrypt m, Bob selects random ℓ and computes $R = \ell P$ and $c = m \oplus H_2(e(P_A, P_{pub})^\ell)$.
  2. Bob sends (R, c) to Alice.
- **Decryption**:
  1. Alice uses her private key to compute $c \oplus H_2(e(S_A, R)) = c \oplus H_2(e(sP_A, \ell P)) = c \oplus H_2(e(P_A, P_{pub})^\ell) = m$.

Pairing computation

A general pairing

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

Many moving parts (parameters):

- What choice of curve?
- What is an appropriate embedding degree k?
- How to balance hardness of DLP among different groups?

Note: Hardness of $\mathbb{G}_T$ is given by $k \cdot |q|$.

Problem: How to build and compute map e?

Pairing computation

Definitions

A **divisor** is a formal sum of points and integer coefficients:

$$\mathcal{D} = \sum_{P \in E} d_P(P)$$

The **degree** of a divisor is the sum of integer coefficients:

$$deg(\mathcal{D}) = \sum_{P \in E} d_P$$

The **support** of a divisor is the set of points P with $d_P \neq 0$.

Pairing computation

The set of divisors forms an abelian group:

$$\sum_{P \in E} a_P(P) + \sum_{P \in E} b_P(P) = \sum_{P \in E} (a_P + b_P)(P)$$

Repeated addition of a divisor to itself is given by:

$$n\mathcal{D} = \sum_{P \in E} (nd_P)(P)$$

Pairing computation

Divisors are a mathematical device convenient to store **poles and zeroes** of rational functions and their **multiplicities**.

The divisor of a non-zero rational function $f : E(\mathbb{F}_{q^k}) \to \mathbb{F}_{q^k}$ is called **principal divisor** and defined as $div(f) = \sum_{P \in E} ord_P(P)$, where ord_P is the **multiplicity** of P.

If $\mathcal{D}$ is a principal divisor, then $deg(\mathcal{D}) = 0$ and $\sum\limits_{P \in E} d_P P = \infty$.

Two divisors $\mathcal{C}$ and $\mathcal{D}$ are **equivalent** ($\mathcal{C} \sim \mathcal{D}$) is their difference ($\mathcal{C} - \mathcal{D}$) is a principal divisor.

Pairing computation

When $div(f)$ and $\mathcal{D}$ have disjoint support:

$$f(\mathcal{D}) = \prod_{P \in E} f(P)^{d_P}$$

Let $P \in E(\mathbb{F}_{q^k})[r]$ and $\mathcal{D}$ a divisor equivalent to $(P) - (\infty)$.

Since $rP = \infty$ and $deg(\mathcal{D}) = 0$, the divisor $r\mathcal{D}$ is principal and there is a function $f_{r,P}$ such that $div(f_{r,P}) = r\mathcal{D} = r(P) - r(\infty)$.

Pairings are defined by the evaluation of $f_{r,P}$ on divisors.

Problem: How to construct and compute $f_{r,P}$?

Pairing computation

Let P, Q be r-torsion points. The pairing $e(P, Q)$ is defined by the evaluation of $f_{r,P}$ at a divisor related to Q.

[Miller 1986] constructed $f_{r,P}$ in stages combining **Miller functions** evaluated at divisors.

[Barreto et al. 2002] showed how to evaluate $f_{r,P}$ at Q using the final exponentiation employed by the Tate pairing.

Pairing computation

Let $g_{U,V}$ be the line equation through points $U, V \in E(\mathbb{F}_{q^k})$ and g_U the shorthand for $g_{U,-U}$.

For any integers a and b, we have:

1. $f_{a+b,P}(\mathcal{D}) = f_{a,P}(\mathcal{D}) \cdot f_{b,P}(\mathcal{D}) \cdot \dfrac{g_{aP,bP}(\mathcal{D})}{g_{(a+b)P}(\mathcal{D})}$
2. $f_{2a,P}(\mathcal{D}) = f_{a,P}(\mathcal{D})^2 \cdot \frac{g_{aP,aP}(\mathcal{D})}{g_{2aP}(\mathcal{D})}$
3. $f_{a+1,P}(\mathcal{D}) = f_{a,P}(\mathcal{D}) \cdot \frac{g_{(a)P,P}(\mathcal{D})}{g_{(a+1)P}(\mathcal{D})}$

Miller's algorithm

Algorithm 1 Miller's Algorithm [Miller 1986, Barreto et al. 2002].

Input: $r = \sum_{i=0}^{\log_2 r} r_i 2^i, P, Q$.

Output: $e_r(P, Q)$.

$T \leftarrow P$
$f \leftarrow 1$
for $i = \lfloor \log_2(r) \rfloor - 1$ **downto** 0 **do**
  $T \leftarrow 2T$
  $f \leftarrow f^2 \cdot \frac{g_{T,T}(\mathcal{D})}{g_{2T}(\mathcal{D})}$
  if $r_i = 1$ **then**
    $T \leftarrow T + P$
    $f \leftarrow f \cdot \frac{g_{T,P}(\mathcal{D})}{g_{T+P}(\mathcal{D})}$
  end if
end for
return f

Miller's algorithm

Let l be the line equation that passes through T and P in the addition $T+P$.

Let v be the vertical line that passes through T and $-T$.

Recall that:

$$f(\mathcal{D}) = \prod_{P \in E} f(P)^{d_P}$$

We can replace:

1. $g_{T,P}(\mathcal{D}) = l_{T,P}((Q+R)-(R)) = \frac{l_{T,P}(Q+R)}{l_{T,P}(R)}$;
2. $g_T(\mathcal{D}) = v_T((Q+R)-(R)) = \frac{v_T(Q+R)}{v_T(R)}$.

Miller's algorithm

Algorithm 2 Miller's Algorithm [Miller, 1986].

Input: $r = \sum_{i=0}^{\log_2 r} r_i 2^i$, P, Q.

Output: $e_r(P, Q)$.

```
T ← P
f ← 1
for i = ⌊log_2(r)⌋ − 1 downto 0 do
    T ← 2T
    f ← f^2 · (l_{T,T}(Q+R) v_{2T}(R)) / (v_{2T}(Q+R) l_{T,T}(R))
    if r_i = 1 then
        T ← T + P
        f ← f · (l_{T,P}(Q+R) v_{T+P}(R)) / (v_{T+P}(Q+R) l_{T,P}(R))
    end if
end for
return f
```

$$T \leftarrow P$$
$$f \leftarrow 1$$
for $i = \lfloor \log_2(r) \rfloor - 1$ **downto** 0 **do**

$$T \leftarrow 2T$$
$$f \leftarrow f^2 \cdot \frac{l_{T,T}(Q+R)v_{2T}(R)}{v_{2T}(Q+R)l_{T,T}(R)}$$

if $r_i = 1$ **then**

$$T \leftarrow T + P$$
$$f \leftarrow f \cdot \frac{l_{T,P}(Q+R)v_{T+P}(R)}{v_{T+P}(Q+R)l_{T,P}(R)}$$

end if

end for

return f

Weil pairing

Let $\mathcal{P}, \mathcal{Q}$ divisors equivalent to $(P) - (\infty)$, $(Q) - (\infty)$, respectively. The Weil pairing is the map:

$$\begin{aligned} w_r \quad &: \quad E(\mathbb{F}_q)[r] \times E(\mathbb{F}_{q^k})[r] \to \mathbb{F}_{q^k}^* \\ w_r(P, Q) \quad &= \quad (-1)^r \cdot \frac{f_{r,P}(\mathcal{Q})}{f_{r,Q}(\mathcal{P})}. \end{aligned}$$

It turns out that we can evaluate the functions over **points** instead of divisors [Miller 1986].

Tate pairing

The **reduced** Tate pairing is the map:

$$\begin{aligned} e_r \quad &: \quad E(\mathbb{F}_q)[r] \times E(\mathbb{F}_{q^k})[r] \to \mathbb{F}_{q^k}^* \\ e_r(P, Q) \quad &= \quad f_{r,P}(\mathcal{D})^{(q^k-1)/r}. \end{aligned}$$

The final exponentiation by $(q^k - 1)/r$ allows [Barreto et al. 2002]:

- Choosing R with coordinates in a subfield to eliminate $l(R), v(R)$
- Choosing R as ∞ and evaluate f on Q instead of $\mathcal{D}$
- Using a distortion map to eliminate $v(Q)$
- Choosing k even and construct a quadratic extension such that the coordinates of Q are in a subfield to eliminate $v(Q)$

Miller's algorithm

Algorithm 3 Miller's Algorithm [Miller, 1986].

Input: $r = \sum_{i=0}^{\log_2 r} r_i 2^i$, P, Q.
Output: $e_r(P, Q)$.

```
T ← P
f ← 1
for i = ⌊log₂(r)⌋ − 1 downto 0 do
    T ← 2T
    f ← f² · (l_{T,T}(Q+R) v_{2T}(R)) / (v_{2T}(Q+R) l_{T,T}(R))
    if r_i = 1 then
        T ← T + P
        f ← f · (l_{T,P}(Q+R) v_{T+P}(R)) / (v_{T+P}(Q+R) l_{T,P}(R))
    end if
end for
return f
```

$T \leftarrow P$

$f \leftarrow 1$

for $i = \lfloor \log_2(r) \rfloor - 1$ **downto** 0 **do**

$\quad T \leftarrow 2T$

$\quad f \leftarrow f^2 \cdot \frac{l_{T,T}(Q+R)v_{2T}(R)}{v_{2T}(Q+R)l_{T,T}(R)}$

$\quad$ **if** $r_i = 1$ **then**

$\quad\quad T \leftarrow T + P$

$\quad\quad f \leftarrow f \cdot \frac{l_{T,P}(Q+R)v_{T+P}(R)}{v_{T+P}(Q+R)l_{T,P}(R)}$

$\quad$ **end if**

end for

return f

Tate pairing

Algorithm 4 Tate pairing [Barreto et al. 2002].

Input: $r = \sum_{i=0}^{\log_2 r} r_i 2^i, P, Q$.
Output: $e_r(P, Q)$.

$T \leftarrow P$
$f \leftarrow 1$
$s \leftarrow r - 1$
for $i = \lfloor \log_2(s) \rfloor - 1$ **downto** 0 **do**
  $T \leftarrow 2T$
  $f \leftarrow f^2 \cdot l_{T,T}(Q)$
  if $r_i = 1, i \neq 0$ **then**
    $T \leftarrow T + P$
    $f \leftarrow f \cdot l_{T,P}(Q)$
  end if
end for
return $f^{(q^k - 1/r)}$

Pairing computation

Important: How can we optimize it?

The main optimization is to reduce the **length** of the loop keeping the **Hamming weight** of r small. There are several ways of doing this: Ate, Ate_i, R-ate, $\chi - ate$.

The **optimal pairing** construction reduces the loop iterations by a factor of $\phi(k)$.

We can observe that Miller's Algorithm employs:

- Extension field arithmetic
- Elliptic curve arithmetic
- Base field arithmetic.

Tate pairing

Algorithm 5 Tate pairing [Barreto et al. 2002].

Input: $r = \sum_{i=0}^{\log_2 r} r_i 2^i, P, Q$.

Output: $e_r(P, Q)$.

$T \leftarrow P$
$f \leftarrow 1$
$s \leftarrow r - 1$
for $i = \lfloor \log_2(s) \rfloor - 1$ **downto** 0 **do**
    $T \leftarrow 2T$
    $f \leftarrow f^2 \cdot l_{T,T}(Q)$
    if $s_i = 1$ **then**
        $T \leftarrow T + P$
        $f \leftarrow f \cdot l_{T,P}(Q)$
    end if
end for
return $f^{(q^k - 1/r)}$

Arithmetic levels

Protocols

$e(P, Q)$

$E(\mathbb{F}_q)$ $E(\mathbb{F}_{q^k})$

$\mathbb{F}_{q^k}$

$\mathbb{F}_q$

Low-level backend

Curve families

BN curves: $k = 12, \ \rho \approx 1$ $p(x) = 36x^4 + 36x^3 + 24x^2 + 6x + 1$ $r(x) = 36x^4 + 36x^3 + 18x^2 + 6x + 1, \quad t(x) = 6z^2 + 1$
BLS12 curves: $k = 12, \ \rho \approx 1.5$ $p(x) = (x-1)^2(x^4 - x^2 + 1)/3 + x,$ $r(x) = x^4 - x^2 + 1, \quad t(x) = x + 1$
KSS18 curves: $k = 18, \ \rho \approx 4/3$ $p(x) = (x^8 + 5x^7 + 7x^6 + 37x^5 + 188x^4 + 259x^3 + 343x^2 + 1763x + 2401)/21$ $r(x) = (x^6 + 37x^3 + 343)/343, \ t(x) = (x^4 + 16z + 7)/7$
BLS24 curves: $k = 24, \ \rho \approx 1.25$ $p(x) = (x-1)^2(x^8 - x^4 + 1)/3 + x,$ $r(x) = x^8 - x^4 + 1, \quad t(x) = x + 1$

Barreto-Naehrig curves

Let x be an integer such that $p(x)$ and $r(x)$ below are prime:

- $p(x) = 36x^4 + 36x^3 + 24x^2 + 6x + 1$
- $r(x) = 36x^4 + 36x^3 + 18x^2 + 6x + 1$

Then $y^2 = x^3 + b$, $b \in \mathbb{F}_p$ is a curve of **order** r and **embedding degree** $k = 12$ [Barreto and Naehrig 2012].

Important: BN curves **used to be** optimal at the 128-bit security level.

Optimal ate pairing

$$\begin{aligned} a_{opt} : \mathbb{G}_2 \times \mathbb{G}_1 &\rightarrow \mathbb{G}_T \\ (Q, P) &\rightarrow \left(f_{r,Q}(P) \cdot l_{rQ,\pi_p(Q)}(P) \cdot l_{rQ+\pi_p(Q),-\pi_p^2(Q)}(P)\right)^{(p^{12}-1)/n} \end{aligned}$$

with $r = 6x + 2, \mathbb{G}_1 = E(\mathbb{F}_p), \mathbb{G}_2 = E(\mathbb{F}_{p^2})[n]$.

Fix $x = -(2^{62} + 2^{55} + 1)$ and $b = 2$. Since $p \equiv 3 \pmod 4$, the towering can be:

- $\mathbb{F}_{p^2} = \mathbb{F}_p[i]/(i^2 - \beta)$, where $\beta = -1$
- $\mathbb{F}_{p^4} = \mathbb{F}_{p^2}[s]/(s^2 - \epsilon)$, where $\xi = 1 + i$
- $\mathbb{F}_{p^6} = \mathbb{F}_{p^2}[v]/(v^3 - \xi)$, where $\xi = 1 + i$
- $\mathbb{F}_{p^{12}} = \mathbb{F}_{p^4}[v]/(t^3 - s)$ or $\mathbb{F}_{p^6}[w]/(w^2 - v)$

Important: Choice of representation changes formulas (and costs)!

Software libraries

There are many different software implementations of pairings:

1. **RELIC**: UNICAMP, flexible and state-of-the-art.
2. **Ate-pairing:** CINVESTAV, used to be state-of-the-art.
3. **mcl:** new library at “new” 128-bit level by Shigeo Mitsunari.
4. **MIRACL**: special support for constrained platforms.
5. **Panda**: not as efficient, but constant-time.
6. **PBC**: on top of GMP, horribly outdated.

Questions?

Code, documentation and tests at the `pairings` branch of my private OpenSSL fork:

`https://github.com/dfaranha/openssl`

Recommended further reading: *Pairings for Beginners*, by Craig Costello, and the early papers by Mike Scott for the optimization techniques.

Questions?

D. F. Aranha

dfaranha@ic.unicamp.br

@dfaranha

References

- [1] Victor S. Miller: The Weil Pairing, and Its Efficient Calculation. J. Cryptology 17(4): 235-261 (2004)
- [2] Dan Boneh, Matthew K. Franklin: Identity-Based Encryption from the Weil Pairing. SIAM J. Comput. 32(3): 586-615 (2003)
- [3] Paulo S. L. M. Barreto, Hae Yong Kim, Ben Lynn, Michael Scott: Efficient Algorithms for Pairing-Based Cryptosystems. CRYPTO 2002: 354-368
- [4] Ryuichi Sakai, Masao Kasahara: ID based Cryptosystems with Pairing on Elliptic Curve. IACR Cryptology ePrint Archive 2003: 54 (2003)

References

- [5] Antoine Joux: A One Round Protocol for Tripartite Diffie-Hellman. J. Cryptology 17(4): 263-276 (2004)
- [6] Paulo S. L. M. Barreto, Michael Naehrig: Pairing-Friendly Elliptic Curves of Prime Order. Selected Areas in Cryptography 2005: 319-331
- [7] Diego F. Aranha, Koray Karabina, Patrick Longa, Catherine H. Gebotys, Julio López: Faster Explicit Formulas for Computing Pairings over Ordinary Curves. EUROCRYPT 2011: 48-68
- [8]Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, Emmanuel Thom?: A Heuristic Quasi-Polynomial Algorithm for Discrete Logarithm in Finite Fields of Small Characteristic. EUROCRYPT 2014: 1-16