Pairings implementation in the PARI computer algebra system

(explained by a mere programmer)

jerome.milan (at) lix.polytechnique.fr

Outline

1 Motivations and context

2 Pairings over elliptic curves

3 Pairing computation

4 Implementation in PARI

Outline

1 Motivations and context

2 Pairings over elliptic curves

3 Pairing computation

4 Implementation in PARI

Pairings at a glance

Let G_1 and G_2 be two groups written additively
Let G_3 be a group written multiplicatively

A pairing e is a application from $G_1 \times G_2$ to G_3

1. e is bilinear, *i.e.*
   1.1 $\forall A, X \in G_1, \forall Y \in G_2,\ e(A+X, Y) = e(A, Y) \cdot e(X, Y)$
   1.2 $\forall X \in G_1, \forall B, Y \in G_2,\ e(X, B+Y) = e(X, B) \cdot e(X, Y)$
2. e is non-degenerated, *i.e.*
   2.1 $\forall X \in G_1, \exists Y \in G_2 \mid e(X, Y) \neq 1$
   2.2 $\forall Y \in G_1, \exists X \in G_1 \mid e(X, Y) \neq 1$

Only interesting pairings in cryptography are defined over groups on Jacobians of abelian varieties

Pairings at a glance

This presentation → pairings on "standard" elliptic curves only

Consider $E(\mathbb{F}_q)$ and $r \mid \#E$

The embedding degree of E with respect to $r \equiv$ smallest k such that $r \mid q^k - 1$

Often, we will have

- $G_1 \subseteq E(\mathbb{F}_q)[r]$
- $G_2 \subset E(\mathbb{F}_{q^k})$
- $G_3 \subset \mathbb{F}^*_{q^k}$

A destructive application – the MOV reduction

The mandatory historical example!

Solve Elliptic Curve Discrete Log Problem

Given $P \in E(\mathbb{F}_q)$ of order r and $R \in \langle P \rangle$, find a such that $R = [a]P$

Overview

1. k such that $E[r] \subseteq E(F_{q^k})$ $(1 \leqslant k \leqslant 6$ for supersingular curves$)$
2. Pick $Q \in E[r]$
3. Compute $e_W(P, Q)$ and $e_W(R, Q)$
4. Since $e_W(R, Q) = e_W(P, Q)^a \in \mathbb{F}_{q^k}^* \rightarrow$ solve DLP in $\mathbb{F}_{q^k}^*$

A. Menezes, S. Vanstone, and T. Okamoto.
Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Trans. Inf. Theory, IT-39(5):1639-1646, 1993.

A plethora of constructive applications

- Identity-based cryptosystems
- Certificate-less public-key infrastructures
- Key agreement protocols
- Short signatures

$\vdots$

- Electronic cash!

$\vdots$

- And about a new application each week...

Outline

1 Motivations and context

2 Pairings over elliptic curves

3 Pairing computation

4 Implementation in PARI

The Weil pairing

Let $f_{n,P}$ be a function in $\mathbb{F}_{q^k}(E)$ with divisor

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

In practice, $f_{n,P}$ computed iteratively in $O(\log(n))$ steps

Definition – The Weil pairing

$$\begin{aligned} e_W : E[r] \times E[r] & \rightarrow \mu_r \\ (P, Q) & \mapsto (-1)^r \frac{f_{r,P}(Q)}{f_{r,Q}(P)} \end{aligned}$$

The Tate pairing

Definition – The unreduced Tate pairing

$$\begin{aligned} \hat{e}_T : E[r] \times E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k}) &\rightarrow \mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^r \\ (P, Q) &\mapsto f_{r,P}(Q) \end{aligned}$$

Defined up to a coset in $(\mathbb{F}^*_{q^k})^r$. To obtain unique representative, raise to the $(q^k - 1)/r$ power.

Definition – The reduced Tate pairing

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

The ate pairing

Let t be the trace of the Frobenius, $\#E(\mathbb{F}_q) = q + 1 - t$
Write $T = t - 1$

Definition – The reduced ate pairing

$$\begin{array}{rcl} e_a : E(\mathbb{F}_q)[r] \times E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k}) \cap \mathrm{Ker}(\pi_q - [q]) & \to & \mu_r \\ (P, Q) & \mapsto & f_{T,Q}(P)^{\frac{q^k-1}{r}} \end{array}$$

The twisted ate pairing

Suppose E admits a twist of order d

Write $e = k/\gcd(k,d)$

Definition – The reduced twisted ate pairing

$$\begin{aligned} e_{tw} : E(\mathbb{F}_q)[r] \times E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k}) \cap \mathrm{Ker}(\pi_q - [q]) &\rightarrow \mu_r \\ (P,Q) &\mapsto f_{T^e,P}(Q)^{\frac{q^k-1}{r}} \end{aligned}$$

Optimal pairings

Optimal pairing

Pairing computable with only $\log_2(r)/\varphi(k)$ iterations

The idea

Compute $f_{mr,Q}(P)$ with $mr = \sum_{i=0}^{l} \lambda_i q^i$ where the λ_i are small
and use Frobenius maps $f_{x,[q^i]Q} = f_{x,Q}^{q^i}$

F. Vercauteren. Optimal Pairings. IEEE Transactions on Information Theory, 56:455461, january 2010.

Florian Hess. Pairing Lattices. In Proceedings of the 2nd International Conference on Pairing-Based Cryptography, Pairing 08, pages 1838, 2008.

Optimal ate pairings

Let $mr = \sum_{i=0}^{l} \lambda_i q^i$ with $r \nmid m$

$$(P, Q) \mapsto \left(\prod_{i=0}^{l} f_{\lambda_i, Q}^{q^i}(P) \prod_{i=0}^{l-1} \frac{l_{[s_{i+1}]Q, [\lambda_i q^i]Q}(P)}{v_{[s_i]Q}(P)} \right)^{(q^k-1)/r}$$

with $s_i = \sum_{j=i}^{l} \lambda_j q^j$

defines a pairing

Optimal only if needs $\sim \log_2(r)/\varphi(k)$ iterations

Optimal ate pairings

Since $\Phi_k(p) \equiv 0 \pmod{r}$ consider only q^i with $i < \varphi(k)$

$$L_{ate} = \begin{pmatrix} r & 0 & 0 & \cdots & 0 \\ -q & 1 & 0 & \cdots & 0 \\ -q^2 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ -q^{\varphi(k)-1} & 0 & 0 & \cdots & 1 \end{pmatrix}$$

Find short vector $\Lambda = [\lambda_0, \lambda_1, \cdots, \lambda_l]$ using LLL

Example – Barreto-Naehrig curve, k=12

$$\begin{aligned} p(x) &= 36x^4 + 36x^3 + 24x^2 + 6x + 1 \\ r(x) &= 36x^4 + 36x^3 + 18x^2 + 6x + 1 \\ t(x) &= 6x^2 + 1 \end{aligned}$$

$\Lambda = [6x + 2, 1, -1, 1]$ gives the optimal pairing

$$f_{6x+2,Q} \cdot l_{[p^3]Q,[-p^2]Q} \cdot l_{[p^3-p^2]Q,[p]Q} \cdot l_{[p-p^2+p^3]Q,[6x+2]Q}$$

Optimal twisted ate pairings

Same as optimal ate but consider $mr = \sum_{i=0}^{l} \lambda_i T^{ei}$

Since $\Phi_k(q) \equiv 0 \mod r$ and $T \equiv q \mod r$ then $\Phi_{k/e}(T^e) \equiv 0 \mod r$
Consider only q^i with $i < \varphi(d)$.

$$L_{tw} = \begin{pmatrix} r & 0 \\ -T^e & 1 \end{pmatrix}$$

Compute short vector $[a, b]$ from LLL such that $a + bT^e \equiv 0 \pmod{r}$

Obtain the (unreduced) pairing

$$f_{a,P}(Q) \cdot f_{b,P}^{p^e}(Q) \cdot v_{[a]P}(Q)$$

Example – Barreto-Naehrig curves, k=12

$$[a, b] = [2x + 1, 6x^2 + 2x]$$

Yields the following unreduced pairing

$$f_{2x+1,P}(Q) \cdot f_{6x^2+2x,P}^{p^2}(Q)$$

Outline

1. Motivations and context
2. Pairings over elliptic curves
3. **Pairing computation**
4. Implementation in PARI

Computing a pairing

Most pairings require two steps

1. Computing $f_{x,P}(Q)$ or $f_{x,Q}(P)$ – The Miller part
2. Raising result to $(q^k - 1)/r$ – The final exponentiation

Exception: the Weil pairing

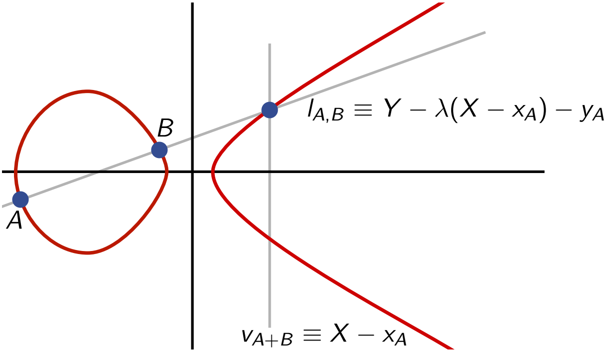
1. Computing $f_{x,P}(Q)$ (Miller light)
2. Computing $f_{x,Q}(P)$ (Full Miller)

Computing $f_{x,P}(Q)$ or $f_{x,Q}(P)$ – The Miller algorithm

Based on following relations:

$$f_{n+1,P} = f_{n,P} \cdot l_{P,[n]P} / v_{[n+1]P} \qquad f_{m+n,P} = f_{m,P} \cdot f_{n,P} \cdot l_{[n]P,[m]P} / v_{[m+n]P}$$

$$f_{-n,P} = 1/f_{n,P} \cdot v_{[n]P}$$



With $f_{1,P} = 1$ and $v_{\mathcal{O}} = 1$

Standard Miller algorithm with NAF

Data: $P \neq \mathcal{O}, Q$, two suitable points on an elliptic curve E over a field,
$x = \sum_{i=0}^{n} x_i 2^i$ with $x_i \in \{-1, 0, 1\}$ and $x_n \neq 0$

Result: $f_{x,P}(Q)$

$R \leftarrow P$, $f \leftarrow 1$, $g \leftarrow 1$

for $i \leftarrow n-1$ **downto** 0 **do**

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

$\quad R \leftarrow R + R$

$\quad g \leftarrow g^2 \cdot v_R(Q)$

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

$\quad\quad f \leftarrow f \cdot l_{R,P}(Q)$

$\quad\quad R \leftarrow R + P$

$\quad\quad g \leftarrow g \cdot v_R(Q)$

$\quad$ **if** $x_i = -1$ **then**

$\quad\quad f \leftarrow f \cdot l_{R,-P}(Q)$

$\quad\quad R \leftarrow R - P$

$\quad\quad g \leftarrow g \cdot v_R(Q) \cdot v_P(Q)$

return f/g

Standard Miller algorithm with NAF

Data: $P \neq \mathcal{O}, Q$, two suitable points on an elliptic curve E over a field,
$x = \sum_{i=0}^{n} x_i 2^i$ with $x_i \in \{-1, 0, 1\}$ and $x_n \neq 0$

Result: $f_{x,P}(Q)$

$R \leftarrow P$, $f \leftarrow 1$, $g \leftarrow 1$

for $i \leftarrow n-1$ **downto** 0 **do**

- $f \leftarrow f^2 \cdot l_{R,R}(Q)$
- $R \leftarrow R + R$
- ~~$g \leftarrow g^2 \cdot v_R(Q)$~~
- **if** $x_i = 1$ **then**
  - $f \leftarrow f \cdot l_{R,P}(Q)$
  - $R \leftarrow R + P$
  - ~~$g \leftarrow g \cdot v_R(Q)$~~
- **if** $x_i = -1$ **then**
  - $f \leftarrow f \cdot l_{R,-P}(Q)$
  - $R \leftarrow R - P$
  - ~~$g \leftarrow g \cdot v_R(Q) \cdot v_P(Q)$~~ Denominator elimination if k even

return f ~~$/g$~~

Boxall et al.'s Miller variant

A variant based on the relation

$$f_{m+n,P} = \frac{1}{f_{-m,P} \cdot f_{-n,P} \cdot l_{[-m]P,[-n]P}}$$

instead of the usual

$$f_{m+n,P} = f_{m,P} \cdot f_{n,P} \cdot l_{[n]P,[m]P} / v_{[m+n]P}$$

→ 3 terms involved instead of 4

Leads to a more complex algorithm

30 to 40% faster for odd k, not interesting for even k

J. Boxall, N. El Mrabet, F. Laguillaumie, and D-P. Le.
A Variant of Miller's Formula and Algorithm. LNCS volume 6487, 2010

Boxall et al.'s Miller variant

$$
\begin{aligned}
f_7 &= \frac{1}{f_{-6} \cdot f_{-1} \cdot l_{-1,-6}} \\
f_{-6} &= \frac{1}{f_3 \cdot f_3 \cdot l_{3,3}} \\
f_3 &= \frac{1}{f_{-2} \cdot f_{-1} \cdot l_{-1,-2}} \\
f_{-2} &= \frac{1}{f_1 \cdot f_1 \cdot l_{1,1}}
\end{aligned}
$$

And since $f_1 = 1$

$$
f_7 = \frac{l_{3,3} \cdot l_{1,1}^2}{f_{-1}^2 \cdot l_{-1,-2}^2 \cdot l_{-1,-6}}
$$

No verticals explicitly evaluated (except f_{-1})

Boxall et al.'s Miller variant – Algorithm

Data: $P \neq \mathcal{O}$, Q, two suitable points on an elliptic curve E over a field,
$x = \sum_{i=0}^{n} x_i 2^i$ with $x_i \in \{0, 1\}$ and $x_n = 1$

Result: $f_{x,P}(Q)$

$R \leftarrow P$, $f \leftarrow 1$, $g \leftarrow 1$, $\delta \leftarrow 0$

if *$n + h$ is even* **then**
    $\delta \leftarrow 1$; $g \leftarrow f_{-1,P}(Q)$

for $i \leftarrow n - 1$ **downto** 0 **do**
    if $\delta = 0$ **then**
        $f \leftarrow f^2 \cdot l_{R,R}(Q)$; $g \leftarrow g^2$
        $R \leftarrow R + R$; $\delta \leftarrow 1$
        if $x_i = 1$ **then**
            $g \leftarrow g \cdot l_{-R,-P} \cdot f_{-1}$
            $R \leftarrow R + P$, $\delta \leftarrow 0$
    else
        $g \leftarrow g^2 \cdot l_{-R,-R}(Q)$; $f \leftarrow f^2$
        $R \leftarrow R + R$; $\delta \leftarrow 0$
        if $x_i = 1$ **then**
            $f \leftarrow f \cdot l_{R,P}$, $R \leftarrow R + P$, $\delta \leftarrow 1$

return f/g

Final exponentiation

Let i be the smallest integer greater than 1 dividing p

$$\begin{aligned}\frac{p^k-1}{r} &= (p^{k/i}-1)\cdot\frac{p^k-1}{(p^{k/i}-1).\Phi_k(p)}\cdot\frac{\Phi_k(p)}{r} \\ &= \mathrm{easy}_1\cdot\mathrm{easy}_2\cdot\mathrm{hard}\end{aligned}$$

k	easy_1	easy_2	Degree Φ_k
11	$p-1$	1	10
12	p^6-1	p^2+1	4
15	p^5-1	p^2+p+1	8
17	$p-1$	1	16
18	p^9-1	p^3+1	6
19	$p-1$	1	18
24	$p^{12}-1$	p^4+1	8
25	p^5-1	1	20
26	$p^{13}-1$	$p+1$	12
27	p^9-1	1	18

Generic multi-exponentiation

Compute $m = f_{x,P}(Q)^{\text{easy}_1 \cdot \text{easy}_2}$ using multiplications and Frobenius

Write $e \equiv \Phi_k(p)/r$ in base q and use multi-exponentiation techniques

$$e = \sum_{i=0}^{n} e_i q^i$$

Simplest algorithm known as interleaving method

1. Compute m^{2^j} for all $0 < 2^j < q$
2. Compute all $m_i = m^{e_i}$ from the m^{2^j}
3. Compute all $m_i^{p^i} = \varphi_i(m_i)$ using precomputed Frobenius powers

Can do better finding patterns in binary representation of the e_i

General case is NP-hard

Generic multi-exponentiation – Kato et al's method

Identify simple common patterns in binary representation of the e_i by arranging them in n_r rows and n_c columns

$$e = \sum_{i=0}^{n_r-1} \sum_{j=0}^{n_c-1} e_{ij} q^{n_c i+j}$$

Kato H, Nogami Y, Nekado K, and Morikawa Y.
Fast Exponentiation in Extension Field with Frobenius Mappings. Memoirs of the Faculty of Engineering of Okayama University, 42:3643, Jan. 2008.

Generic multi-exponentiation – Kato et al's method

Example from Kato *et al.*'s paper

$e = \sum_{i=0}^{5} e_i q^i$	$e_1 = (1001)_2$	$e_0 = (1110)_2$
$n_c = 2$	$e_3 = (1101)_2$	$e_2 = (1110)_2$
$n_r = 3$	$e_5 = (1111)_2$	$e_4 = (0101)_2$

$$e = (e_5 q + e_4) q^4 + (e_3 q + e_2) q^2 + (e_1 q + e_0)$$

R_0	$=$	$\varphi_1($	m^8			m^1	$)$	m^8	m^4	m^2	
R_1	$=$	$\varphi_1($	m^8	m^4		m^1	$)$	m^8	m^4	m^2	
R_2	$=$	$\varphi_1($	m^8	m^4	m^2	m^1	$)$		m^4		m^1
			C_{111}	C_{110}	C_{100}	C_{111}		C_{011}	C_{111}	C_{011}	C_{100}

Generic multi-exponentiation – Kato et al's method

$$
\begin{array}{rcllllllllll}
R_0 & = & \varphi_1(& m^8 & & & m^1 &) & m^8 & m^4 & m^2 & \\
R_1 & = & \varphi_1(& m^8 & m^4 & & m^1 &) & m^8 & m^4 & m^2 & \\
R_2 & = & \varphi_1(& m^8 & m^4 & m^2 & m^1 &) & & m^4 & & m^1 \\
\hline
 & & & C_{111} & C_{110} & C_{100} & C_{111} & & C_{011} & C_{111} & C_{011} & C_{100}
\end{array}
$$

$$
\begin{array}{lll}
C_{000} = 1 & C_{001} = 1 & C_{010} = 1 \\
C_{011} = m^8 m^2 & C_{100} = \varphi(m^2) m^1 & C_{101} = 1 \\
C_{110} = \varphi(m^4) & C_{111} = \varphi(m^8 m^1) m^4 &
\end{array}
$$

$$
\begin{aligned}
R_0 &= C_{111} C_{011} \\
R_1 &= C_{111} C_{110} C_{011} \\
R_2 &= C_{111} C_{110} C_{100}
\end{aligned}
$$

$$m^e = \varphi^4(R_2)\varphi^2(R_1)R_0$$

Generic multi-exponentiation – Kato et al's method

Overview

1. Compute m^{2^j} for all $0 < 2^j < q$
2. Compute C_i for all $0 < i < 2^{n_r}$
3. Compute R_j for all $0 < j < n_r$
4. Compute m_e using precomputed Frobenius and R_j

Cost

$$\begin{aligned} ct(1 - 1/2^r) + r(2^r - 1)/2 + r - 1 &\quad \text{multiplications in } \mathbb{F}_{q^k} \\ (c-1)(2^r - 1) + r - 1 &\quad \text{applications of Frobenius maps} \end{aligned}$$

with $t = \lfloor \log_2(p-1) \rfloor$

Family-dependent exponentiation – Scott et al's method

Overview

1. Use polynomial representation of q and r to express e_i as polynomials
2. Find vectorial addition-chain for each coefficient in $e_i(x)$
3. Deduce sequence of multiplications squarings

M. Scott, N. Benger, M. Charlemagne, L. Dominguez Perez, and E. Kachisa. On the Final Exponentiation for Calculating Pairings on Ordinary Elliptic Curves. LNCS Volume 5671, pages 78-88, 2009.

Family-dependent exponentiation – Scott et al's method

Example – Barreto-Naehrig family, k=12

$$
\begin{aligned}
p(x) &= 36x^4 + 36x^3 + 24x^2 + 6x + 1 \\
r(x) &= 36x^4 + 36x^3 + 18x^2 + 6x + 1 \\
e(x) &= \left(p(x)^4 - p(x)^2 + 1\right)/r(x) \\
&= e_3(x)p3 + e_2(x)p2 + e_1(x)p + e_0(x) \\
e_3(x) &= 1 \\
e_2(x) &= 6x^2 + 1 \\
e_1(x) &= -36x^3 - 18x^2 - 12x + 1 \\
e_0(x) &= -36x^3 - 30x^2 - 18x - 2
\end{aligned}
$$

Now compute m^x, m^{x^2}, m^{x^3} and $m^p, m^{p^2}, m^{p^3}, (m^x)^p, (m^x)^{p^2}, (m^x)^{p^3}, (m^{x^2})^{p^2}$

Family-dependent exponentiation – Scott et al's method

Example – Barreto-Naehrig family (continued)

m^e becomes

$$\begin{aligned} m^e &= [m^p \cdot m^{p^2} \cdot m^{p^3}] \cdot [1/m]^2 \cdot [(m^{x^2})p^2]^6 \cdot [1/(m^x)^p]^{12} \cdot [1/(m^x \cdot (m^{x^2})^p)]^{18} \\ &\quad \cdot [1/m^{x^2}]^{30} \cdot [1/(m^{x^3} \cdot (m^{x^3})^p)]^{36} \\ &= y_0 \cdot y_1^2 \cdot y_2^6 \cdot y_3^{12} \cdot y_4^{18} \cdot y_5^{30} \cdot y_6^{36} \end{aligned}$$

Compute addition-chain $[1, 2, 3, 6, 12, 18, 30, 36]$

Family-dependent exponentiation – Scott et al's method

Example – Barreto-Naehrig family (continued)

Compute vectorial addition-chain

$$
\begin{array}{ccccccc}
(1 & 0 & 0 & 0 & 0 & 0 & 0) \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
(0 & 0 & 0 & 0 & 0 & 0 & 1) \\
(2 & 0 & 0 & 0 & 0 & 0 & 0) \\
(2 & 0 & 1 & 0 & 0 & 0 & 0) \\
(2 & 1 & 1 & 0 & 0 & 0 & 0) \\
(0 & 1 & 0 & 1 & 0 & 0 & 0) \\
(2 & 2 & 1 & 1 & 0 & 0 & 0) \\
(2 & 1 & 1 & 0 & 1 & 0 & 0) \\
(4 & 4 & 2 & 2 & 0 & 0 & 0) \\
(6 & 5 & 3 & 2 & 1 & 0 & 0) \\
(12 & 10 & 6 & 4 & 2 & 0 & 0) \\
(12 & 10 & 6 & 4 & 2 & 1 & 0) \\
(12 & 10 & 6 & 4 & 2 & 0 & 1) \\
(24 & 20 & 12 & 8 & 4 & 2 & 0) \\
(36 & 30 & 18 & 12 & 6 & 2 & 1)
\end{array}
$$

Family-dependent exponentiation – Scott et al's method

Example – Barreto-Naehrig family (continued)

Deduce sequence of operations

$$
\begin{aligned}
T_0 &\leftarrow y_6^2 \\
T_0 &\leftarrow T_0 \cdot y_4 \\
T_0 &\leftarrow T_0 \cdot y_5 \\
T_1 &\leftarrow y_3 \cdot y_5 \\
T_1 &\leftarrow T_1 \cdot T_0 \\
T_0 &\leftarrow T_0 \cdot y_2 \\
T_1 &\leftarrow (T_1)^2 \\
T_1 &\leftarrow T_1 \cdot T_0 \\
T_1 &\leftarrow (T_1)^2 \\
T_0 &\leftarrow T_1 \cdot y_1 \\
T_1 &\leftarrow T_1 \cdot y_0 \\
T_0 &\leftarrow (T_0)^2 \\
Result &\leftarrow T_0 \cdot T_1
\end{aligned}
$$

Family-dependent exponentiation – Scott et al's method

Problem

What if rational coefficient in the $e_i(x)$?

$\rightarrow$ Compute a power of the pairing

Up to twice as fast as generic multi-exponentiation depending on families

Outline

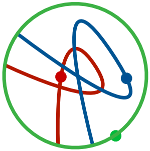
1 Motivations and context

2 Pairings over elliptic curves

3 Pairing computation

4 Implementation in PARI

A PARI module for pairing computation



- APIP – Another Pairing Implementation in PARI
- Dynamically loadable
- A general module
  No emphasis on special pairings / curves
- Licence: GPL if permission from CNRS
- Requires SCons build tool – `http://www.scons.org`

Features

- Pairings
  - Tate, Weil, ate and twisted ate
  - Optimal ate and optimal twisted ate for selected curve families
- Miller variants
  - Standard, NAF, Boxall *et al.*
- Coordinate systems
  - Affine, projective, jacobian
- Final exponentiation
  - Naive, interleaving, Kato *et al.*, Scott *et al.*
- Arithmetic
  - Custom reduction for $(x^k + a)$ and $(x^k + x + a)$ defining polynomials

Example

```
  [...]

pairing = apip_alloc_pairing(E1, p, f1e, f2e, r, family);

apip_compute_frobenius_powers(pairing);
apip_set_family_param(pairing, z);
apip_set_miller(pairing, "naf");
apip_set_coord(pairing, "affine");
apip_set_denom_elim(pairing, 1);
apip_set_frob_trace(pairing, t);
apip_set_twist_degree(pairing, 6);
apip_set_do_reduce(pairing, 1);
apip_set_do_naive_exp(pairing, 0);

t1   = apip_tate(pairing, P, QT);
w1   = apip_weil(pairing, P, QW);
a1   = apip_ate(pairing, P, QA);
o1   = apip_opti_ate(pairing, P, QA);
tw1  = apip_twisted(pairing, P, QA);
otw1 = apip_opti_twisted(pairing, P, QA);
```

A potential problem for integration in PARI

- Huge structure allocated on the heap

```
struct pairing_data_struct {
  GEN curve;
  GEN charac;
    [...]
  GEN (*miller_func_f1_f2) (struct pairing_data_struct*, GEN, GEN, GEN);
  GEN (*miller_func_f2_f1) (struct pairing_data_struct*, GEN, GEN, GEN);
  GEN (*opti_ate_func)     (struct pairing_data_struct*, GEN, GEN);
  GEN (*opti_twisted_func) (struct pairing_data_struct*, GEN, GEN);
  GEN (*final_exp_func)    (struct pairing_data_struct*, GEN);
    [...]
};
```

- Need for explicit memory management

```
apip_alloc_pairing(...)
apip_free_pairing(...)
```

Other shortcomings

- Large characteristic only

  Curves over $\mathbb{F}_p$ and $\mathbb{F}_{p^k}$ only

- Standard elliptic curves only

  No Edward curves

- Mostly standard finite field arithmetic from PARI

  No finite fields towers

  No special arithmetic depending on embedding degree

- Input restricted to suit cryptographic applications

- Could projective and jacobian coordinates be improved?

Benchmarks

Benchmarks on an early 2008 Macbook Pro laptop

- Intel Core 2 Duo @ 2.5 GHz and 2 GB RAM
- OS X 10.6
- GCC 4.2
- GMP 5.0
- PARI SVN version 12717 (December 2010)

Warning – Not for speed records

APIP is a general module – as such it is not competitive with respect to extremely specialized implementations found in the literature

Bit length recommendations

Benchmarks for the AES security levels

Security	$\log_2 r$	$\log_2 q^k$	Target $k\rho$
128	256	3248	12.7
192	384	7936	20.7
256	512	15424	30.1

Table: Security level according to the ECRYPT II recommendations.

Curves from "the taxonomy"

D. Freeman, M. Scott, and E. Teske.
A Taxonomy of Pairing-Friendly Elliptic Curves. Journal of Cryptology, 23:224280, April 2010.

Selected curves for benchmarks

Security	k	ρ	$k\rho$	Target $k\rho$	Curve	Construction
128	12	1	12	12.7	F_2, F_3	6.8
	11	6/5	13.2	12.7	G	6.6
192	19	10/9	21.1	20.7	H	6.6
	18	4/3	24	20.7	I	6.12
	17	9/8	19.1	20.7	–	6.6
	17	19/16	20.2	20.7	–	6.2
	15	3/2	22.5	20.7	P	6.6
	15	3/2	22.5	20.7	R	Duan *et al.*
	12	1	12	20.7	F_4	6.8
256	24	5/4	30	30.1	L_1, L_2	6.6
	27	10/9	30	30.1	M	6.6
	26	7/6	30.34	30.1	N	6.6
	25	13/10	32.5	30.1	O	6.6
	12	1	12	30.1	F_5	6.8

Table: Selected curves for each security level. Unless stated, construction in last column refers to “the taxonomy”.

Relative cost of arithmetic operations

128 and 192 bit security level

Curve	F_2	F_3	G	H	I	P	R	F_4
π/M_2	0.19	0.21	0.63	0.95	0.16	0.15	0.15	0.17
I_1/M_1	15.2	10.5	11.0	12.9	11.7	13.2	13.3	11.8
I_2/M_2	8.6	8.8	7.9	8.7	8.8	8.1	8.1	8.1

256 bit security level

Curve	L_1	L_2	M	N	O	F_5
π/M_2	0.14	0.14	0.15	0.16	1.2	0.18
I_1/M_1	13.0	13.0	13.2	11.9	11.9	10.4
I_2/M_2	9.1	9.2	10.1	10.1	9.5	8.1

Miller part timings – 128 and 192 bit security level

Curve	Tate	Ate	Opti ate	Twisted	Opti twisted	Weil
F_2	14.6	17.5	8.9	12.8	7.3	158.4
F_3	15.9	18.6	9.6	14.7	8.7	168.7
G	38.8	103.9	19.8	–	–	208.5
	29.2	103.1	19.5	–	–	205.0
H	123.7	319.0	35.2	–	–	703.9
	90.8	338.7	39.9	–	–	699.1
I	80.7	147.3	34.7	152.0	35.8	905.0
P	133.0	242.9	41.2	–	71.0	740.8
	93.6	241.5	41.1	–	52.9	689.1
R	133.6	41.5	–	80.7	68.3	741.7
	94.9	42.5	–	59.3	52.7	702.7
F_4	95.0	105.7	53.6	90.5	52.4	961.4

Table: Timings of the Miller part in milliseconds. When applicable, timings obtained using the Boxall *et al.* variant are shown on a second line.

Miller part timings – 256 bit security level

Curve	Tate	Ate	Opti ate	Twisted	Opti twisted	Weil
L_1	184.4	58.1	–	88.6	–	2164.3
L_2	184.0	55.6	–	84.6	–	2160.2
M	371.0	510.9	53.3	1795.1	181.0	2274.2
	267.1	533.4	52.3	1307.3	132.5	2199.8
N	194.7	613.8	89.8	–	–	2375.6
O	419.8	1345.4	129.7	–	–	2517.3
	308.8	1406.7	125.2	–	–	2519.1
F_5	420.5	449.0	224.8	400.5	235.3	4213.3

Table: Timings of the Miller part in milliseconds. When applicable, timings obtained using the Boxall *et al.* variant are shown on a second line.

Final exponentiation timings

Curve	Full Naive	Hard Naive	Kato *et al.*	Scott *et al.*
F_2	57.6	17.2	7.6	4.2
F_3	70.3	20.4	8.5	5.4
G	80.2	77.8	24.4	20.5
H	463.6	460.0	110.0	83.2
I	680.9	212.6	83.2	48.6
P	486.9	253.7	105.0	50.0
R	486.7	254.4	90.7	47.6
F_4	383.1	104.1	36.9	25.5
L_1	2030.6	636.9	202.1	96.8
L_2	2032.7	638.7	202.0	96.6
M	2131.6	1403.6	313.3	131.3
N	2268.4	1015.6	267.5	172.5
O	2615.7	2137.7	471.5	321.6
F_5	1826.7	470.6	165.0	117.0

Table: Final exponentiation timings in milliseconds.

Full pairing timings – 128 and 192 bit security level

Curve	Pairing	Unreduced	Final exp	Reduced
F_2	opti twisted	7.3	4.2	11.5
F_3	opti twisted	8.7	5.4	14.1
G	opti ate	19.8	20.5	40.3
	opti ate	19.5		40.0
H	opti ate	35.2	83.2	118.4
	opti ate	39.9		123.1
I	opti ate	34.7	48.6	83.3
P	opti ate	41.2	50.0	91.2
	opti ate	41.1		91.1
R	ate	41.5	47.6	89.1
	ate	42.5		90.1
F_4	opti twisted	52.4	25.5	77.9

Table: Timings of fastest reduced pairings implemented, in milliseconds. When applicable, timings using the Boxall *et al.* variant are shown on a second line.

Full pairing timings – 256 bit security level

Curve	Pairing	Unreduced	Final exp	Reduced
L_1	ate	58.1	96.8	154.9
L_2	ate	55.6	96.6	152.2
M	opti ate	53.3	131.3	184.6
	opti ate	52.3		183.6
N	opti ate	89.8	172.5	262.3
O	opti ate	129.7	321.6	451.3
	opti ate	125.2		446.8
F_5	opti ate	224.8	117.0	341.8

Table: Timings of fastest reduced pairings implemented, in milliseconds. When applicable, timings using the Boxall *et al.* variant are shown on a second line.