Software implementation of pairings

Diego de Freitas Aranha

September 21, 2011

Department of Computer Science
University of Brasília

Joint work with
K. Karabina, P. Longa, C. Gebotys, J. López, D. Hankerson, A. Menezes, E. Knapp, F. Rodríguez-Henríquez, L. Fuentes-Castañeda, J.-L. Beuchat, J. Detrey, N. Estibals.

Introduction

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

- Identity-based encryption
- Short signatures
- Non-interactive authenticated key agreement

Pairing computation is the most expensive operation in PBC.

Important: Make it faster!

Objective

Explore new ways to accelerate serial and parallel implementations of cryptographic pairings:

- Maximize throughput
- Minimize latency

Applications: servers, real-time services.

Contributions

- Lazy reduction in extension fields
- Elimination of penalty for negative parameterizations
- Compressed cyclotomic squarings
- Parallelization of Miller's Algorithm
- Delayed squarings and new formulations
- Notes on high security levels and current state-of-the-art

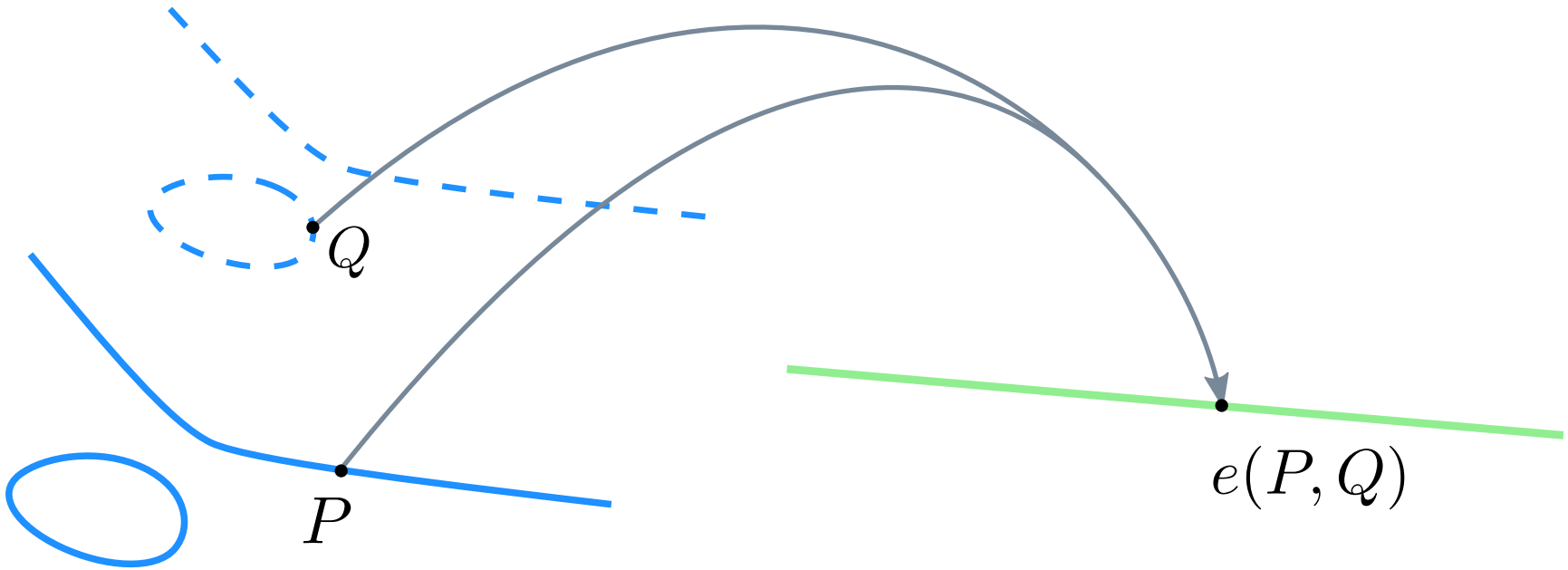
Bilinear pairings

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 n.

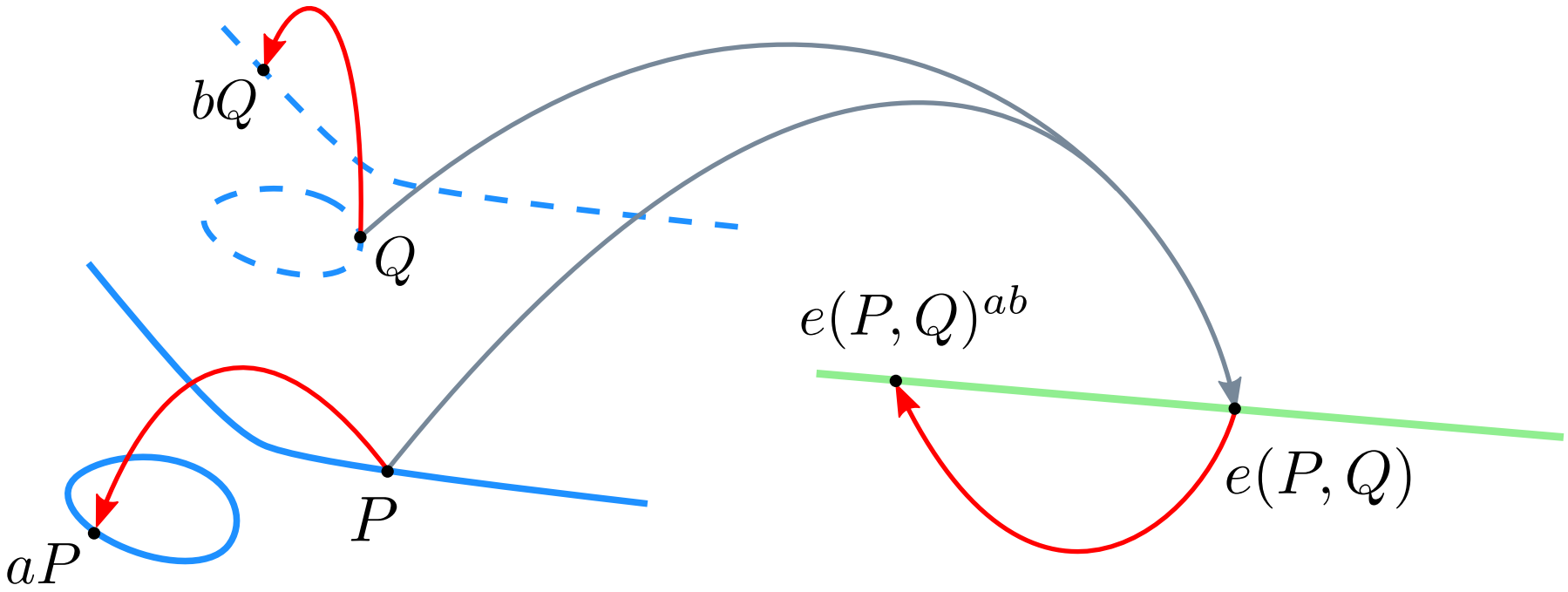
An efficiently-computable map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is an **admissible bilinear map** if the following properties are satisfied:

1. *Bilinearity:* given $(V,\ W) \in \mathbb{G}_1 \times \mathbb{G}_2$ and $(a,\ b) \in \mathbb{Z}_q^*$: $e(aV, bW) = e(V, W)^{ab} = e(abV, W) = e(V, abW)$.
2. *Non-degeneracy:* $e(P, Q) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_T}$ is the identity of the group $\mathbb{G}_T$.

Bilinear pairings



Bilinear pairings



If $\mathbb{G}_1 = \mathbb{G}_2$, the pairing is **symmetric**.

Barreto-Naehrig curves

Let u be an integer such that p and n below are prime:

- $p = 36u^4 + 36u^3 + 24u^2 + 6u + 1$
- $n = 36u^4 + 36u^3 + 18u^2 + 6u + 1$

Then $E : y^2 = x^3 + b, b \in \mathbb{F}_p$ is a curve of **order** n and **embedding degree** $k = 12$.

Example: $u = -(2^{62} + 2^{55} + 1), b = 2$ (implementation-friendly).

Pairing computation

The pairing $e_r(P, Q)$ 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.

Pairing computation

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

For any integers a and b, we have:

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

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

Algorithm 1 Miller's Algorithm.

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

1: $T \leftarrow P$
2: $f \leftarrow 1$
3: **for** $i = \lfloor \log_2(r) \rfloor - 1$ **downto** 0 **do**
4: $\quad f \leftarrow f^2 \cdot l_{T,T}(Q)$
5: $\quad T \leftarrow 2T$
6: $\quad$ **if** $r_i = 1$ **then**
7: $\quad\quad f \leftarrow f \cdot l_{T,P}(Q)$
8: $\quad\quad T \leftarrow T + P$
9: $\quad$ **end if**
10: **end for**
11: **return** $f^{(q^k-1)/n}$

Asymmetric pairing

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

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

The towering is:

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

Generalized lazy reduction

Intuitively, it is a trade-off between addition and modular reduction:

$$(a \cdot b) \bmod p + (c \cdot d) \bmod p = (a \cdot b + c \cdot d) \bmod p$$

Observation: Pairings use non-sparse primes for $\mathbb{F}_p$!

Generalized lazy reduction

Intuitively, it is a trade-off between addition and modular reduction:

$$(a \cdot b) \bmod p + (c \cdot d) \bmod p = (a \cdot b + c \cdot d) \bmod p$$

Observation: Pairings use non-sparse primes for $\mathbb{F}_p$!

Previous state-of-the-art ($3M + 2R$ in $\mathbb{F}_{p^2}$):

$$a \cdot b = (a_0 b_0 + a_1 b_1 \beta) + [(a_0 + a_1)(b_0 + b_1) - a_0 b_0 - a_1 b_1]\, i,$$

For $k = 2^i 3^j$, total of $(3^i \cdot 6^j)M + (2 \cdot 3^{i-1} \cdot 6^j)R$.

Generalized lazy reduction

Idea: Suppose $\mathbb{F}_{p^2}$ is a higher extension and apply recursively!

Any component c of an element in $\mathbb{F}_{p^k}$ is ultimately computed as $c = \sum \pm a_i b_j \bmod p$, requiring a single reduction.

New state-of-the-art: total of $(3^i \cdot 6^j)M + kR$.

Generalized lazy reduction

Idea: Suppose $\mathbb{F}_{p^2}$ is a higher extension and apply recursively!

Any component c of an element in $\mathbb{F}_{p^k}$ is ultimately computed as $c = \sum \pm a_i b_j \bmod p$, requiring a single reduction.

New state-of-the-art: total of $(3^i \cdot 6^j)M + kR$.

Remark 1: Montgomery bounds should be maintained for intermediate results. Choose $|p|$ acoordingly.

Remark 2: Same idea applies to arithmetic in $E'(\mathbb{F}_{p^2})$.

Example: Multiplication in $\mathbb{F}_{p^{12}}$ goes from $54M + 36R$ to $54M + 12R$. In total, 40% of reductions are saved.

Removing the inversion penalty

Consider $(p^{12}-1)/n = (p^6-1)(p^2+1)(p^4-p^2+1)/n$.

The **hard part** is $(p^4-p^2+1)/n$ which requires 3 $|u|$-th powers.

If $u < 0$, from pairing definition:

$$a_{opt}(Q,P) = \left[f_{|r|,Q}(P)^{-1} \cdot h\right]^{\frac{p^{12}-1}{n}}.$$

By distributing the power $(p^{12}-1)/n$, we can compute instead:

$$a_{opt}(Q,P) = \left[f_{|r|,Q}(P)^{p^6} \cdot h\right]^{\frac{p^{12}-1}{n}}.$$

Revised pairing computation

Algorithm 2 Miller's Algorithm for general r, even k.

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

1: $T \leftarrow P$
2: $f \leftarrow 1$
3: **for** $i = \lfloor \log_2(r) \rfloor - 1$ **downto** 0 **do**
4: $\quad f \leftarrow f^2 \cdot l_{T,T}(Q)$
5: $\quad T \leftarrow 2T$
6: $\quad$ **if** $r_i = 1$ **then**
7: $\quad\quad f \leftarrow f \cdot l_{T,P}(Q)$
8: $\quad\quad T \leftarrow T + P$
9: $\quad$ **end if**
10: **end for**
11: **if** $u < 0$ **then** $T \leftarrow -T$, $f \leftarrow f^{q^{k/2}}$
12: **return** $f^{(q^k-1)/n}$

Compressed cyclotomic squarings

Consider $\mathbb{F}_{p^{12}} = \mathbb{F}_{p^4}[t]/(t^3 - s)$.

Let $g = \sum_{i=0}^{2} (g_{2i} + g_{2i+1}s)t^i \in \mathbb{G}_{\phi_6}(\mathbb{F}_{p^2})$ and
$g^2 = \sum_{i=0}^{2} (h_{2i} + h_{2i+1}s)t^i$ with $g_i, h_i \in \mathbb{F}_{p^2}$.

Given $C(g) = [g_2, g_3, g_4, g_5]$, it is efficient to compute
$C(g^2) = [h_2, h_3, h_4, h_5]$.

Important: Decompression map D requires one inversion in $\mathbb{F}_{p^2}$.

Compressed cyclotomic squarings

Recall that $|u| = 2^{62} + 2^{55} + 1$.

Idea: $g^{|u|}$ can now be computed in three steps:

1. Compute $\mathcal{C}(g^{2^i})$ for $1 \leq i \leq 62$ and store $\mathcal{C}(g^{2^{55}})$ and $\mathcal{C}(g^{2^{62}})$
2. Compute $\mathcal{D}(\mathcal{C}(g^{2^{55}})) = g^{2^{55}}$ and $\mathcal{D}(\mathcal{C}(g^{2^{62}})) = g^{2^{62}}$
3. Compute $g^{|u|} = g^{2^{62}} \cdot g^{2^{55}} \cdot g$

Remark: Montgomery's simultaneous inversion allows simultaneous decompression.

Example: Computing a $|u|$-th power is now 30% faster.

Implementation results

Table: Operation counts for different implementations of the Optimal Ate pairing at the 128-bit security level.

Work	**Phase**	**Operations in $\mathbb{F}_p$**
Beuchat *et al.* 2010	ML	$6992M + 5040R$
	FE	$4647M + 4244R$
	ML+FE	$11639M + 9284R$
Aranha *et al.* 2011	ML	$6504M + 2736R$
	FE	$3648M + 1926R$
	ML+FE	$10152M + 4662R$

[Pereira *et al.* 2011] has a slightly faster operation count, but which produces a slower implementation in the target platform.

Implementation results

Table: Timings in cycles for the asymmetric setting on 64-bit processors.

	Beuchat *et al.* 2010			
Operation	Phenom II	Core i7	Opteron	Core 2 Duo
Mult in $\mathbb{F}_{p^2}$	440	435	443	590
Squaring in $\mathbb{F}_{p^2}$	353	342	355	479
Miller Loop	1,338,000	1,330,000	1,360,000	1,781,000
Final Exp.	1,020,000	1,000,000	1,040,000	1,370,000
Pairing	2,358,000	2,330,000	2,400,000	3,151,000
	Aranha et al. 2011			
Operation	Phenom II	Core i5	Opteron	Core 2 Duo
Mult in $\mathbb{F}_{p^2}$	368	412	390	560
Squaring in $\mathbb{F}_{p^2}$	288	328	295	451
Miller Loop	898,000	978,000	988,000	1,275,000
Final Exp.	664,000	710,000	722,000	919,000
Pairing	1,562,000	1,688,000	1,710,000	2,194,000
Improvement	34%	28%	29%	30%

Important: Latency of around 0.5 milisec in a 3GHz Phenom II X4.

Property of Miller functions

$$f_{a\cdot b,P}(\mathcal{D}) = f_{b,P}(\mathcal{D})^{a} \cdot f_{a,bP}(\mathcal{D})$$

Property of Miller functions

$$f_{a\cdot b,P}(\mathcal{D}) = f_{b,P}(\mathcal{D})^{a} \cdot f_{a,bP}(\mathcal{D})$$

We can write $r = 2^w r_1 + r_0$ and compute $f_{r,P}(\mathcal{D})$:

$$\begin{aligned} f_{r,P}(\mathcal{D}) &= f_{2^w r_1 + r_0,P}(\mathcal{D}) \\ &= f_{r_1,P}(\mathcal{D})^{2^w} \cdot f_{2^w,r_1 P}(\mathcal{D}) \cdot f_{r_0,P}(\mathcal{D}) \cdot \frac{l_{(2^w r_1)P,r_0 P}(\mathcal{D})}{v_{rP}(\mathcal{D})}. \end{aligned}$$

Property of Miller functions

$$f_{a\cdot b,P}(\mathcal{D}) = f_{b,P}(\mathcal{D})^a \cdot f_{a,bP}(\mathcal{D})$$

We can write $r = 2^w r_1 + r_0$ and compute $f_{r,P}(\mathcal{D})$:

$$\begin{aligned} f_{r,P}(\mathcal{D}) &= f_{2^w r_1 + r_0,P}(\mathcal{D}) \\ &= f_{r_1,P}(\mathcal{D})^{2^w} \cdot f_{2^w,r_1 P}(\mathcal{D}) \cdot f_{r_0,P}(\mathcal{D}) \cdot \frac{l_{(2^w r_1)P,r_0 P}(\mathcal{D})}{v_{rP}(\mathcal{D})}. \end{aligned}$$

If r has low Hamming weight, w can be chosen so that r_0 is **small**.

For many processors, we can:

- Apply the formula recursively
- Write r as $r = 2^{w_i} r_i + \cdots + 2^{w_2} r_2 + 2^{w_1} r_1 + r_0$

If P is fixed (private key), $r_i P$ can also be precomputed.

Load balancing

Problem: We must determine an optimal partition w_i.

Let $c_1(1)$ be the cost of a serial loop and $c_\pi(i)$ be the cost of a parallel loop for processor $1 \leq i \leq \pi$.

Load balancing

Problem: We must determine an optimal partition w_i.

Let $c_1(1)$ be the cost of a serial loop and $c_\pi(i)$ be the cost of a parallel loop for processor $1 \leq i \leq \pi$.

We can count the operations executed by each processor and solve the system $c_\pi(1) = c_\pi(i)$ to obtain w_i. The **speedup** is:

$$s(\pi) = \frac{c_1(1)+exp}{c_\pi(1)+par+exp},$$

where *par* is the cost of parallelization and *exp* is the cost of the final exponentiation.

Symmetric pairing

A **pairing-friendly supersingular binary elliptic curve** is the set of solutions $(x, y) \in \mathbb{F}_{2^m} \times \mathbb{F}_{2^m}$ satisfying the equation

$$y^2 + y = x^3 + x + \mathbf{b},$$

where $\mathbf{b} \in \{0, 1\}$, and a **point at infinity** ∞.

Symmetric pairing

Choosing $T = 2^m - N$ and a prime n dividing N,
[Barreto *et al.* 2004] defined the reduced η_T pairing:

$$\begin{aligned} \eta_T &: E(\mathbb{F}_{2^m})[n] \times E(\mathbb{F}_{2^m})[n] \to \mathbb{F}^*_{2^{4m}} \\ \eta_T(P,Q) &= f_{T',P'}(\psi(Q))^{\frac{2^{4m}-1}{N}}, \end{aligned}$$

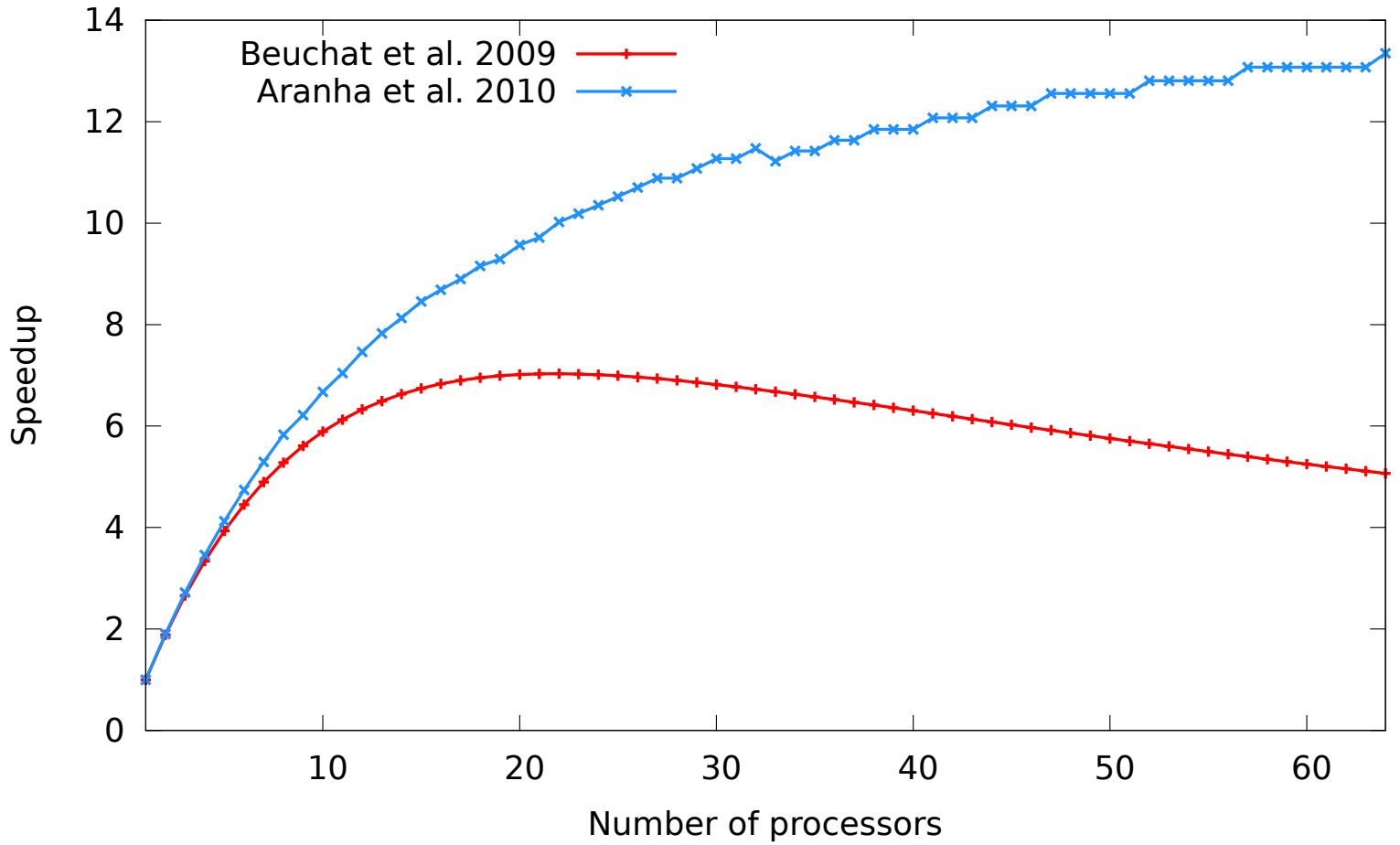
where $T' = \pm T$ and $P' = \pm P$.

The function f is a **Miller function** and ψ is the **distortion map** $\psi(x,y) = (x^2 + s, y + sx + t)$.

Implementation results

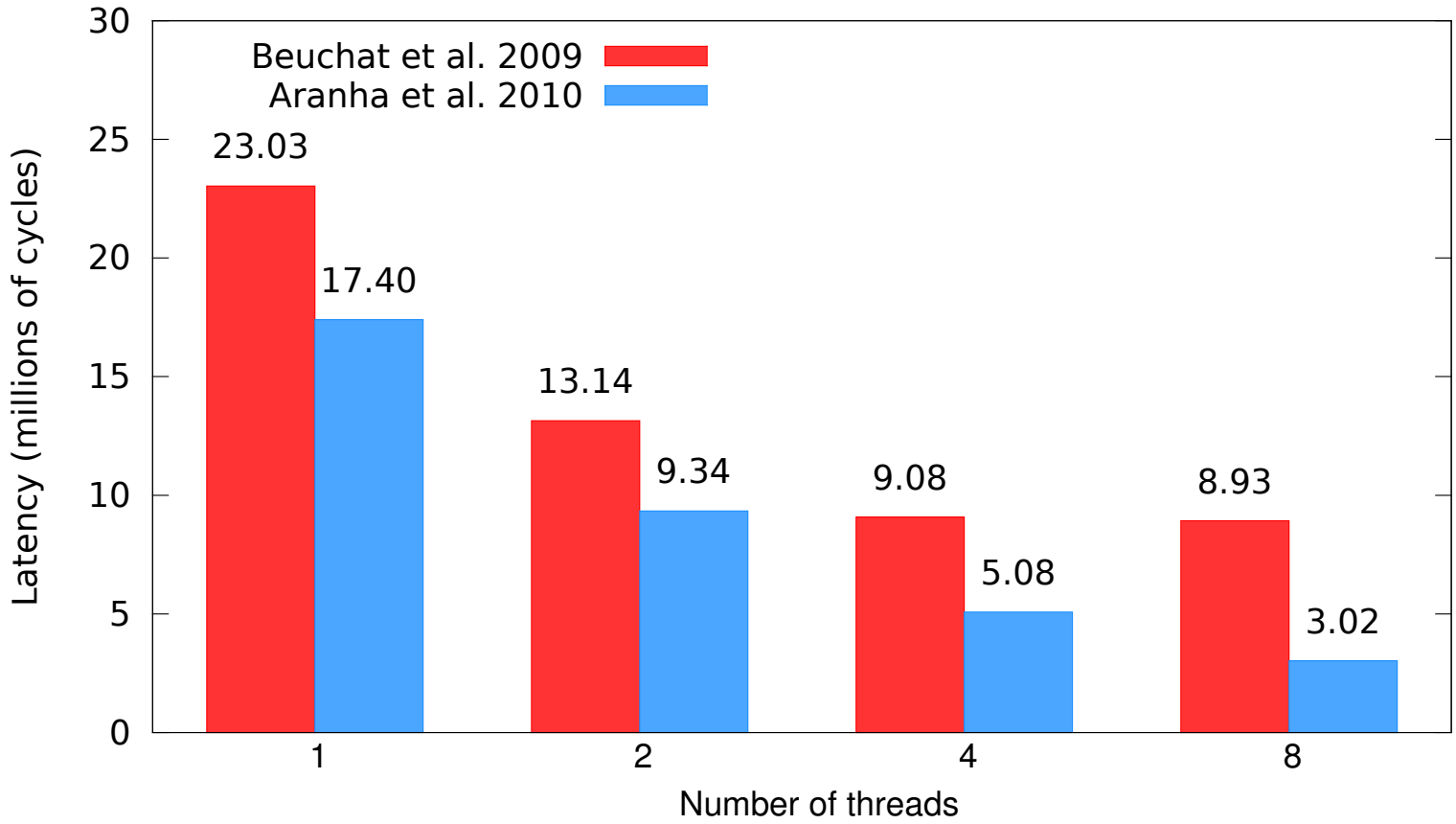
For the asymmetric setting, estimated speedup of only 10%.

For the symmetric setting:



Implementation results

Figure: Timings in the symmetric setting taken on an Intel Core 2 45nm.



Implementation results

New parallelization:

- No significant storage costs and almost-linear scalability
- Latency improvement of 28%, 44% and 66% in 2, 4, 8 processors

Limitations in the asymmetric setting:

- Serial final exponentiation
- Expensive point doublings
- Expensive extension field squarings

Delayed squaring

Idea: Delay the squarings until we reach the cyclotomic subgroup!

Recall the parallelization ($M = \frac{q^k-1}{r}$):

$$f_{r,P}(\mathcal{D})^M = \left(f_{r_1,P}(\mathcal{D})^M\right)^{2^w} \cdot f_{2^w,r_1P}(\mathcal{D})^M \cdot \left(f_{r_0,P}(\mathcal{D}) \cdot \frac{l_{(2^w r_1)P,r_0P}(\mathcal{D})}{v_{rP}(\mathcal{D})}\right)^M.$$

Remark: Delayed squarings increase speedup to 18-20%.

Parallel pairing derivations

Hess' instantiation (α-Weil)

$$\alpha(P,Q) = \left(\frac{f_{2u+1,P}(Q)}{f_{2u+1,Q}(P)} \left(\frac{f_{u,(6u+2)P}(Q) f^{u}_{6u+2,P}(Q)}{f_{u,(6u+2)Q}(P) f^{u}_{6u+2,Q}(P)} \right)^{p^2} \right)^{(p^6-1)(p^2+1)}$$

Critical path: $\left(\left(f^{u}_{u,(6u+2)Q}(P) \right)^{p^2} \right)^{(p^6-1)(p^2+1)}$

Parallel pairing derivations

New instantiation (β-Weil)

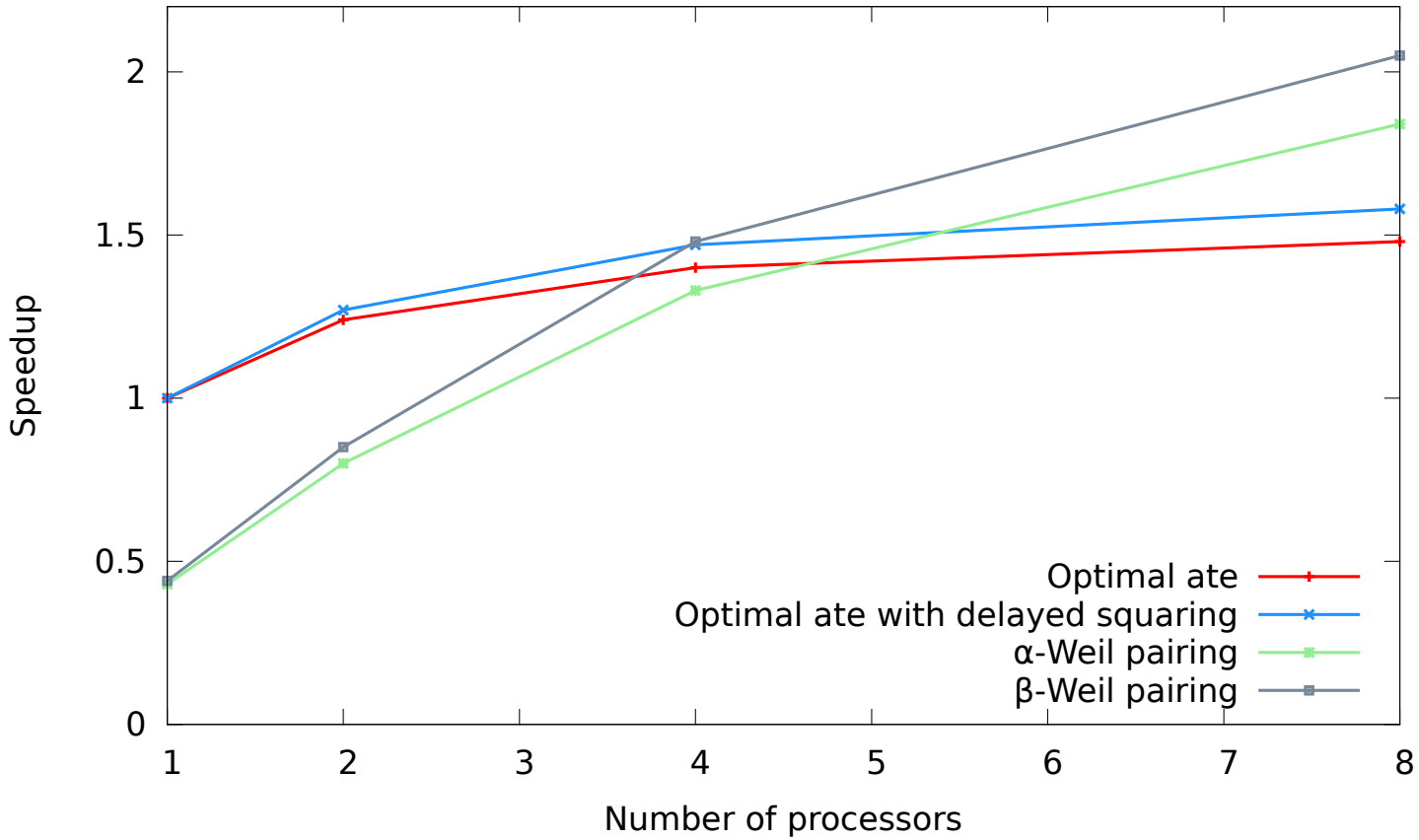
$$\beta(P,Q) = \left(\left(\frac{f_{p,h,P}(Q)}{f_{p,h,Q}(P)} \right)^{p} \frac{f_{p,h,pP}(Q)}{f_{p,h,Q}(pP)} \right)^{(p^6-1)(p^2+1)}$$

Critical path: $pP, (f_{p,h,Q}(pP))^{(p^6-1)(p^2+1)}$

Optimization:

$$pP = 2u(p^2-2)P + p^2P - P = 2u(\phi(P) - 2P) + \phi(P) - P.$$

Implementation results



Best results until now:

- Optimal *ate* pairing reaches speedup of 1.45 with 4 processors
- β-Weil pairing reaches speedup of 1.86 with 8 processors

Curve choice at higher security levels

Important: Pairing security is defined by the hardness of the DLP in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$.

- Barreto-Naehrig curves are optimal at the 128-bit level
- Security usually scaled by increasing embedding degree
- Kachisa-Scott-Schaefer curves with $k = 18$ have been pointed as the best family known for the 192-bit level
- What about other families?

Curve choice at higher security levels

Table: Operation counts for the Optimal Ate pairing at the 192-bit security level. M is the cost of multiplying two 512-bit integers in a 64-bit machine.

Family	**Phase**	**Operations in $\mathbb{F}_p$**
BLS ($k = 24, \lvert p\rvert = 478$)	ML	$14990M$
	FE	$25785M$
	ML+FE	$40775M$
BN ($k = 12, \lvert p\rvert = 638$)	ML	$26084M$
	FE	$11284M$
	ML+FE	$37368M$
KSS ($k = 18, \lvert p\rvert = 512$)	ML	$13817M$
	FE	$23022M$
	ML+FE	$36839M$
BW ($k = 12, \lvert p\rvert = 638$)	ML	$16823M$
	FE	$12647M$
	ML+FE	$29470M$

Table: Timings in 10^3 cycles on an Intel Core i7 Sandy Bridge 32nm at the 128-bit security level using the fastest multipliers available.

	Number of threads			
Asymmetric pairing	**1**	**2**	**4**	**8**
Optimal ate	1562	1287	1137	1107
Improved optimal ate	–	1260	1080	1056
α-Weil	–	–	1272	936
β-Weil	–	–	1104	840
Symmetric pairing	**1**	**2**	**4**	**8**
Genus-1 η_T	6455	3370	1794	1034
Genus-2 Optimal η – general	8265	–	–	–
Genus-2 Optimal η – degenerate	2358	–	–	–

Conclusions and future

New techniques for implementing pairings:

- Speed records for pairing computation in software (hardware)
- Dependency on architectural features
- Scalable parallelization
- New pairing derivations

Emphasis on implementation of protocols:

- Pairing type and optimizations differ greatly
- Higher security levels should be more interesting

RELIC cryptographic library:
`http://code.google.com/p/relic-toolkit/`

Thank you for your attention!
Any questions?

References

- D. F. Aranha, J. López, D. Hankerson. *High-speed parallel software implementation of η_T pairing*. CT-RSA 2010, 89–105.
- D. F. Aranha, J.-L. Beuchat, J. Detrey, N. Estibals. *Optimal Eta Pairing on Supersingular Genus-2 Binary Hyperelliptic Curves*. Cryptology ePrint Archive, Report 2010/559.
- D. F. Aranha, K. Karabina, P. Longa, C. Gebotys, J. López. *Faster Explicit Formulas for Computing Pairings over Ordinary Curves*. EUROCRYPT 2011, 48–68.
- D. F. Aranha, E. Knapp, A. Menezes, F. Rodríguez-Henríquez. *Parallelizing the Weil and Tate Pairings*. IMA-CC 2011, To appear.