The Realm of the Pairings

Paulo S. L. M. Barreto

Universidade de São Paulo
USP

Prolegomena

- Thanks to the organizers of SAC 2013 for the invitation!
- Accompanying paper: joint work with D. Aranha, P. Longa and J. Ricardini.
- “I know I’m speaking in a marvelous accent without the slightest English” – Viktor Frankl, 1972.
- Hard task ahead: very first talk, to a very heterogeneous audience!

Cold start

- Tax payment authentication.
  - Government of São Paulo, Brazil.
  - $> 40 \times 10^6$ inhabitants, 1/3 of GDP.
- Old system (before 2001):
  - Mechanical, non-cryptographic authentication system (authenticating printer).
  - Manual verification, requiring a trusted user.
- Frauds!
  - Government admitted to 5% [sic] of tax payment evasion out of a $\$500 \times 10^6$ gross monthly tax revenue (for just one type of tax, namely, car licensing).
  - New system needed!

Requirements

- Automatic process, without manual intervention.
- Open specification, unencumbered by patents.
- Public-key scheme with security level roughly equivalent to RSA-1024.
- Authentication tag must be printable on two alphanumerical lines (320 bits).
- *Half of the available space is occupied by context information (user id, bank id, amount paid, date, etc).*
- *About* $2 - 4 \times 10^6$ *authentications a month must be handled on a single Pentium II 450 MHz PC.*
- Not for the faint of heart ☺

Assessment

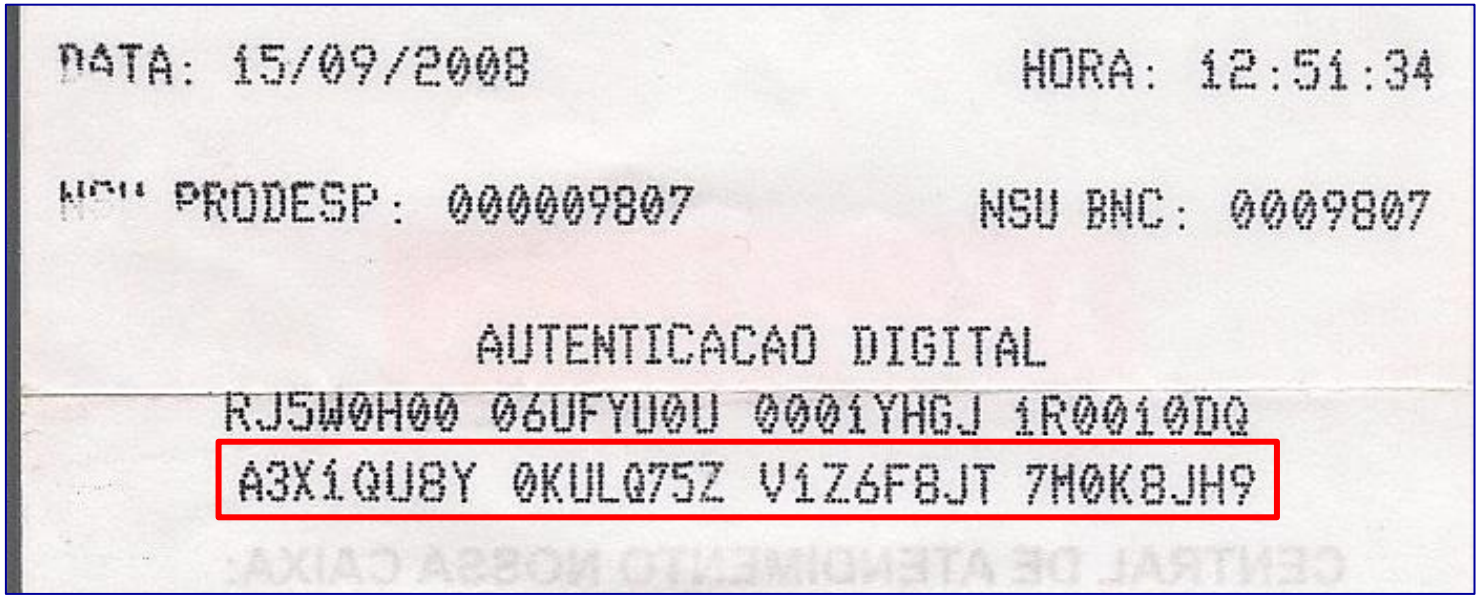
- ∴160-bit signatures: ECDSA would just not do.
- Available options at the time:
  - CFS
  - Quartz
  - OP/BLS (preprint)
- Would any of these be acceptable?

Assessment

- CFS
  - Very slow to generate (no more than $\sim 4 \times 10^4$ sigs/month on target platform).
- Quartz
  - Unknown security (now broken).
  - Covered by patents.
- OP/BLS
  - No patents.
  - Formal proof of security (under the gap Diffie-Hellman assumption).
  - Reported efficiency at the time scaled to $\sim 4 \times 10^5$ sigs/month on target platform.

Solution and results

- The pairing-based OP/BLS scheme was the only plausible choice, though performance needed a boost.



Solution and results

- All reqs satisfied:
  - CPU >80% idle after improvement by B., Kim, Lynn and Scott.
  - Room for business rule improvements.
- Government reported that frauds fell from 5% to 0% [sic], increasing tax revenue from $\$500 \times 10^6$ to... $\$1.5 \times 10^9$ [sic].
- Still in use today – no further modification was ever needed.

Bilinear maps

- Seminal use: cryptanalysis (MOV & FR attacks).
- Amazingly flexible tool for *constructing* cryptosystems with novel and useful features (Antoine Joux is one of the key researchers to blame ☺).
- Identity-based schemes:
  - Signatures (plain, blind, proxy, ring, undeniable, batch, …)
  - Encryption (plain, broadcast, keyword-search capable, …)
  - Signcryption
  - Key agreement (plain, authenticated, group, …)
  - Hierarchical cryptosystems
  - Threshold cryptosystems (secret sharing, signatures, …)
  - Chameleon hash and signatures
  - ...

Bilinear maps

- "Conventional" systems
  - Access control, identification and traitor tracing
  - Credentials (anonymous, hidden, self-blindable, …)
  - Key agreement and non-interactive key distribution
  - Encryption (strongly insulated, intrusion-resilient, …)
  - Signatures (short, group, aggregate, ring, verifiably encrypted, blind, partially blind, proxy, undeniable, limited-verifier, …)
  - Signcryption
  - Threshold cryptosystems (secret sharing, signatures)
  - Hierarchical and role-based cryptosystems
  - Chameleon hash and signatures
  - Certificateless and self-certified PKC
  - ...

Criticism

- "Pairings are too slow for practical consideration."
- To what extent is this (in)correct?
- But first, some theory (caveat: sloppy math ahead! ☺)

Bilinear maps: definition

- Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be groups of the same order n, the first two usually written additively and the third one written multiplicatively.

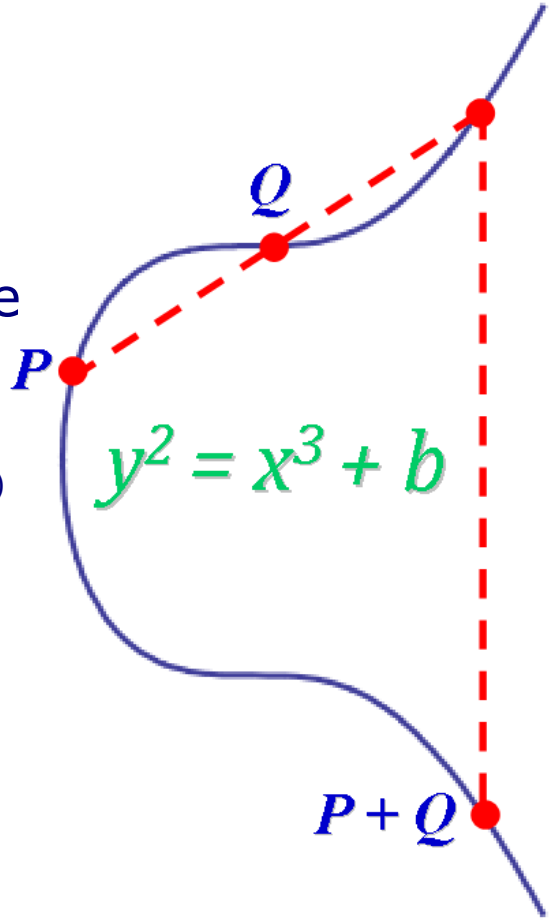
- A bilinear map (or *pairing*) is a function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ satisfying the conditions:
  - **Bilinearity**: $\forall P \in \mathbb{G}_1, Q \in \mathbb{G}_2, a \in \mathbb{Z}/n\mathbb{Z} : e(aP, Q) = e(P, aQ) = e(P, Q)^a$.
  - **Non-degeneracy**: $\forall P \in \mathbb{G}_1, \exists Q \in \mathbb{G}_2 : e(P, Q) \neq 1$.
  - **Efficiently computable**.

OP/BLS signatures

- Setup: $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T,\ H : \{0,1\}^* \to \mathbb{G}_1$.
- Key pair: $(s \xleftarrow{\$} \mathbb{Z}/n\mathbb{Z},\ V \leftarrow sQ \in \mathbb{G}_2)$.
- Signature: $\Sigma \leftarrow sH(m) \in \mathbb{G}_1$.
- Verification: accept $(m, \Sigma) \Leftrightarrow$ $e(\Sigma, Q) = e(H(m), V)$.
- Explanation: $e(\Sigma, Q) = e(sH(m), Q) =$ $e(H(m), Q)^s = e(H(m), sQ) = e(H(m), V)$.

Elliptic curves and pairings

- Pairings of interest are certain rational functions on elliptic curves.
- An elliptic curve is a smooth projective algebraic curve of genus 1 with at least one marked point (∞).
  - Projective equation: points $[X : Y : Z]$ with $Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$ (*)
  - Affine part equation: points (x, y) with $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$, together with an extra point at infinity, which corresponds to $Z = 0$ in the projective form.
- Group law defined for the points of a curve (chord-and-tangent method).



(*) actually other kinds of projective coordinates are usually adopted

Projective and affine coordinates

- $E : Y^2Z = X^3 + aXZ^2 + bZ^3$
- $P = [X_P : Y_P : Z_P]$
- $Q = [X_Q : Y_Q : Z_Q]$
- $R = P + Q = [X_R : Y_R : Z_R]$

- $\mu \leftarrow X_Q Z_P - X_P Z_Q$
- $\lambda \leftarrow Y_Q Z_P - Y_P Z_Q$
- $X_R \leftarrow \lambda^2 \mu Z_P Z_Q - (X_P Z_Q + X_Q Z_P)\mu^3$
- $Y_R \leftarrow -\lambda^3 Z_P Z_Q + \lambda \mu^2 X_Q Z_P - \mu^3 Y_P Z_Q$
- $Z_R \leftarrow Z_P Z_Q \mu^3$

- $E : y^2 = x^3 + ax + b$
- $P = (x_P, y_P)$
- $Q = (x_Q, y_Q)$
- $R = P + Q = (x_R, y_R)$

- $\lambda \leftarrow (y_Q - y_P)(x_Q - x_P)^{-1}$
- $x_R \leftarrow \lambda^2 - (x_P + x_Q)$
- $y_R \leftarrow -\lambda^3 + \lambda x_Q - y_P$

Look more complicated, but involve no inversion, and have lots of common factors

Multiplication by scalar

- **Input**: $P \in E;\ r = (r_t, r_{t-1}, \ldots, r_0)_2 : r_t = 1$
- **Output**: rP

1. $A \leftarrow P$
2. **for** $j \leftarrow t-1$ **downto** 0 **do**
3. $\quad A \leftarrow 2A$
4. $\quad$ **if** $r_j = 1$ **then**
5. $\quad\quad A \leftarrow A + P$
6. $\quad$ **end if**
7. **end for**
8. **return** A

- called double-and-add method, for obvious reasons

Rational maps

- A rational map f over $\mathbb{K}$ is a function of form $f(z) = cg(z)/h(z)$, where g and h are monic polynomials over $\mathbb{K}$ and $c \in \mathbb{K}$ is a constant.
- Both $g(z)$ and $h(z)$ split over $\overline{\mathbb{K}}$:

$$g(z) = \prod_{j=1}^{s} (z - a_j)^{m_j}, \qquad h(z) = \prod_{k=1}^{t} (z - b_k)^{n_k}$$

Rational maps

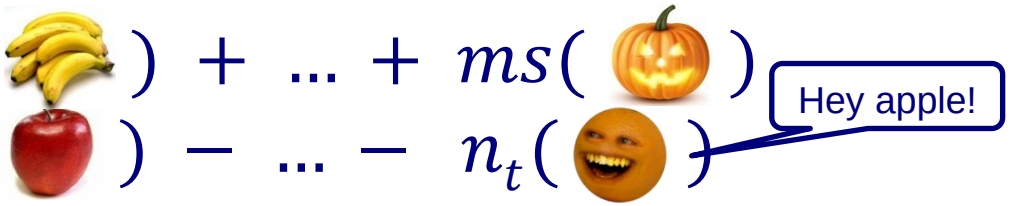
- Assume that $\gcd(g,h) = 1$, i.e. $a_j \neq b_k$. The zeroes of f are the a_j with multiplicities m_j, and the poles of f are the b_k with multiplicities $-n_k$. The multiplicity of f at ∞ is $\deg(h) - \deg(g) = -\left(\sum_j m_j - \sum_k n_k\right)$.

- All one needs to know to define f up to the constant c are its zeroes and poles with their respective multiplicities.

Divisors

- The divisor of a rational map f is a tabular device to represent it:

$$(f) = m_1(\ ) + \ldots + ms(\ ) - n_1(\ ) - \ldots - n_t(\ ) - (\Sigma_j m_j - \Sigma_k n_k)(\infty).$$



- The degree of a divisor is the sum of all multiplicities. Therefore $\deg((f)) = 0$.
- Properties: $(c) = 0$, $(fg) = (f) + (g)$, $(f/g) = (f) - (g)$.

Divisors

- Divisors of functions defined on the points of an elliptic curve over $\mathbb{F}_q$ are rational functions of the point coordinates over the algebraic closure $\overline{\mathbb{F}_q}$.
- General divisors are unrestricted tabular associations: $\mathfrak{D} = \sum_P m_P(P)$.
- Not all possible divisors correspond to function. In particular, if $\deg(\mathfrak{D}) \neq 0$ then $\mathfrak{D}$ does not correspond to a function.

Divisors

- Divisors constitute an Abelian group under pointwise coefficient addition: $\sum_P m_P(P) + \sum_P n_P(P) = \sum_P (m_P + n_P)(P)$.

- A divisor may be huge – there are infinite choices of zeroes and poles. It is thus advantageous to define equivalence classes so as to keep the representation small.

Divisors

- Two divisors $\mathfrak{D}_1$ and $\mathfrak{D}_2$ are equivalent iff their difference is the divisor of a function, i.e. $\mathfrak{D}_1 \sim \mathfrak{D}_2 \Leftrightarrow \mathfrak{D}_1 - \mathfrak{D}_2 = (f)$ for some f.

- The Cantor-Koblitz algorithm reduces any divisor to a uniquely defined equivalent divisor of the form $\sum_P m_P(P) - (\sum_P m_P)(\infty)$ where $\sum_P m_P \leq g$ where g is the curve genus.

- Reduced divisors over elliptic curves are of the form $(P) - (\infty)$ for some P.

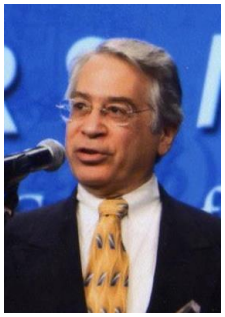
Miller functions

- A Miller function is any function $f_{i,P}$ such that $(f_{i,P}) = i(P) - (iP) - (i-1)(\infty)$.
  - Notice that $(f_{n,P}) = n(P) - n(\infty)$.
  - Also, $(f_{0,P}) = (f_{1,P}) = 0$, i.e. $f_{0,P}$ and $f_{1,P}$ are constant.
- The line $\ell_{U,V}$ through points U and V has divisor $(\ell_{U,V}) = (U) + (V) + (-U - V) - 3(\infty)$.
- The vertical line v_P through a point P has divisor $(v_P) = (P) + (-P) - 2(\infty)$.
- Miller functions satisfy a recursive relation $f_{i+j,P} = c\, f_{i,P}\, f_{j,P}\, \ell_{iP,jP}\, v_{(i+j)P}$.

Miller functions

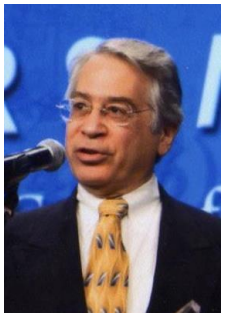
- $$\begin{aligned}\left(f_{i+j,P}\right) &= (i+j)(P) - \left((i+j)P\right) - (i+j-1)(\infty) \\ &= i(P) - (iP) - (i-1)(\infty) \\ &\quad + j(P) - (jP) - (j-1)(\infty) \\ &\quad + (iP) + (jP) + (-(i+j)P) - 3(\infty) \\ &\quad - \left((i+j)P\right) - (-(i+j)P) + (1+1)(\infty) \\ &= \left(f_{i,P}\right) + \left(f_{j,P}\right) + \left(\ell_{iP,jP}\right) - \left(v_{(i+j)P}\right) \\ &= \left(f_{i,P}\, f_{j,P}\, \ell_{iP,jP}\, v_{(i+j)P}\right).\end{aligned}$$

- $\therefore f_{i+j,P} = c\, f_{i,P}\, f_{j,P}\, \ell_{iP,jP}\, v_{(i+j)P}$ for some c.



’s algorithm

- Miller’s algorithm computes $f_{n,P}(Q)$ up to c concomitantly with the double-and-add scalar multiplication nP.
- The trick:
  - $f_{i+1,P} = f_{i,P}\, f_{1,P}\, \ell_{iP,P}/v_{(i+1)P}\ (= f_{i,P}\, \ell_{iP,P}/v_{(i+1)P})$.
  - $f_{2i,P} = f_{i,P}^2\, \ell_{iP,iP}/v_{(2i)P}$.
- Functions $\ell_{iP,P}$, $\ell_{iP,iP}$, $v_{(i+1)P}$, and $v_{(2i)P}$ appear naturally during the computation of nP.
  - Denominators might be evaluated on a point where they vanish: use $n(P+R) - n(R) \sim n(P) - n(\infty)$ for random R.



's algorithm

- **Input**: $P \in \mathbb{G}_1,\ Q \in \mathbb{G}_2,\ n = (n_t, n_{t-1}, \ldots, n_0)_2 : n_t = 1$
- **Output**: $f_{n,P}(Q)$ up to a constant

1. $f \leftarrow 1, R \xleftarrow{\$} E, A \leftarrow P$
2. **for** $j \leftarrow t-1$ **downto** 0 **do**
3. $\quad f \leftarrow f^2\, \ell_{A,A}(Q+R)\, v_{2A}(R) \,/\, \ell_{A,A}(R)\, v_{2A}(Q+R), A \leftarrow 2A$
4. $\quad$ **if** $n_j = 1$ **then**
5. $\quad\quad f \leftarrow f\, \ell_{A,P}(Q+R)\, v_{A+P}(R) \,/\, \ell_{A,P}(R)\, v_{A+P}(Q+R), A \leftarrow A+P$
6. $\quad$ **end if**
7. **end for**
8. **return** f

The Weil pairing

- The Weil pairing of order n at points P and Q is $w(P,Q) := f_{n,P}(Q)/f_{n,Q}(P)$. Note that the constant factor c of f is irrelevant – it does not affect the pairing value.

- Miller's algorithm computes $f_{n,P}(Q)$ (and of course $f_{n,Q}(P)$ as well), and hence the Weil pairing.

- Exercise: show that this function is indeed a bilinear map.

The Tate pairing

- The Tate pairing at points *P* and *Q* is defined as $\tau(P,Q) := f_{n,P}(Q)^z$ where $z := (q^k - 1)/n$.

- The $(q^k - 1)/n$ exponent contains all factors $q^i - 1$ where $i \mid k$. This eliminates c by Fermat's Little Theorem.

- One invocation of Miller's algorithm is traded for one exponentiation (usually faster).

Improvements

- NSA's Jerry Solinas claimed (verbally) to have been the first person to actually implement Miller's algorithm.
- ~15 minutes on a PC (around 1990, so possibly an Intel 80386 or similar).
- Boneh et al. reported ~2.9 s on a 1GHz Pentium III for the (then estimated as) 2^{80} security level.
- Long way to go...

Improvements

- Anecdote: more than one researcher claim to have reviewed more than one paper on speeding up the Weil pairing during the 1990's.
- All rejected!
- "What's the point? You only get a few bits in the MOV attack, and it only applies to a few curves nobody really uses."

Improvements

- Tate-like pairings: any factor or denominator in a subfield vanishes due to the final exponentiation.
- Swap the arguments: computing $f_{n,Q}(P)$ instead of $f_{n,Q}(P)$ may lead to shorter loop length.
- Optimal pairings: shortest possible loop length $\ell \sim \phi(k)$ for embedding degree k.
- Final exponentiation may become the bottleneck!
- [R.I.P.] η_T pairing: was the fastest known pairing, but is now defunct (Antoine Joux has more to say about this ☺).

Pairings galore

- Weil pairing: $w(P,Q) := f_{n,P}(Q)/f_{n,Q}(P)$.
- Tate pairing: $\tau(P,Q) := f_{n,P}(Q)^z$ where $z := (q^k - 1)/n$.
- Eta pairing (or twisted Ate pairing when defined over an ordinary curve): $\eta(P,Q) := f_{\lambda,P}(Q)^z$ where $\lambda^d \equiv 1 \pmod{n}$.
- Ate pairing: $a(P,Q) := f_{t-1,Q}(P)^z$, where t is the trace of the Frobenius endomorphism.
- Optimized Ate and twisted Ate pairings: $a_c(P,Q) := f_{(t-1)^c \bmod n,Q}(P)^z$, $\eta_c(P,Q) := f_{\lambda^c \bmod n,P}(Q)^z$, for some $0 < c < k$.
- Optimal Ate pairing: $a_{\text{opt}}(P,Q) := f_{\ell,Q}(P)^z$ for a certain ℓ such that $\lg \ell \approx (\lg n)/\varphi(k)$.
- Eil pairing: $w_s(P,Q) := -\omega f_{s,P}(Q)/f_{s,Q}(P)$ where $s := q \bmod n$ and $\omega^k \equiv 1$.
- ...

Pairing-friendly curves

- Not all elliptic curves are suitable for pairing applications.
- On the one hand, the embedding degree k must be small enough to make $\mathbb{F}_{q^k}$ arithmetic tractable (but it is usually enormous).
- On the other hand, k must not be proportionally too small: the value of $k \lg q$ must ensure that the discrete logarithm problem in $\mathbb{F}_{q^k}^*$, where subexponential algorithms exist, remains intractable.

Pairing-friendly curves

- Supersingular curves allow for the limited range $k \in \{2,3,4,6\}$ (more recently, only $k = 2$). Hyperelliptic supersingular curves do not effectively improve. MNT curves are ordinary but restrict $k \in \{3,4,6\}$.
- Several methods to construct curves containing a subgroup with arbitrary k are known, but the resulting group size is relatively small: $\rho := \lg q / \lg n \sim 2$.

Pairing-friendly curves

- Algebraic constructions allow for a better relation between n and q for certain values of k:
  - BN curves attain $\rho = 1$ for $k = 12$; ideal around security level around 2^{128}.
  - KSS and BLS12 curves provide the best tradeoff for security level 2^{192}.
  - BLS24 curves are the most suitable family known when addressing the 2^{256} security level.
- Holistic trend: choose pairing-friendly curves that improve all operations needed by cryptosystems (nor just pairing computation).

BN curves in a nutshell

- $E/\mathbb{F}_p : y^2 = x^3 + b$ (a Bachet curve)
- $\#E = n = p + 1 - t$ where, for some $u \in \mathbb{Z}$:
  - $p = p(u) = 36u^4 + 36u^3 + 24u^2 + 6u + 1$
  - $n = n(u) = 36u^4 + 36u^3 + 18u^2 + 6u + 1$
  - $t = t(u) = 6u^2 + 1$
- Abundant and easy to find.
- Embedding degree 12 (ideal at security level 2^{128} but good between legacy 2^{80} and long-term 2^{192}).
- Very friendly holistic subfamilies.

Optimal Ate pairing on general BN curves

- **Input**: $P \in \mathbb{G}_1,\ Q \in \mathbb{G}_2,\ \ell = |6u+2| = \sum_{i=0}^{\lg \ell} \ell 2^i$
- **Output**: $a_{opt}(Q,P)$

1. $d \leftarrow g_{Q,Q}(P),\ T \leftarrow 2Q,\ e \leftarrow 1$
2. **if** $\ell_{\lfloor \lg \ell \rfloor - 1} = 1$ **then** $e \leftarrow g_{T,Q}(P),\ T \leftarrow T+Q$
3. $f \leftarrow d \cdot e$
4. **for** $i = \lfloor \lg \ell \rfloor - 2$ **downto** 0 **do**
5. $\quad f \leftarrow f^2 \cdot g_{T,T}(P),\ T \leftarrow 2T$
6. $\quad$ **if** $\ell_i = 1$ **then** $f \leftarrow f \cdot g_{T,Q}(P),\ T \leftarrow T+Q$
7. **end for**
8. $Q_1 \leftarrow \phi_p(Q),\ Q_2 \leftarrow \phi_p^2(Q)$
9. **if** $u < 0$ **then** $T \leftarrow -T,\ f \leftarrow f^{p^6}$
10. $d \leftarrow g_{T,Q_1}(P),\ T \leftarrow T+Q_1,\ e \leftarrow g_{T,-Q_2}(P),\ T \leftarrow T-Q_2, f \leftarrow f \cdot (d \cdot e)$
11. $f \leftarrow f^{(p^6-1)(p^2+1)(p^4-p^2+1)/n}$
12. **return** f

Complicated but efficient!

Still doing elliptic arithmetic…

Affine pairings

- Projective coordinates avoid field inversions, trading them for multiplications and additions.
- Affine coordinates need inversions, hence they are bad for both elliptic operations and pairing computations.
- … or are they?

Benchmarks

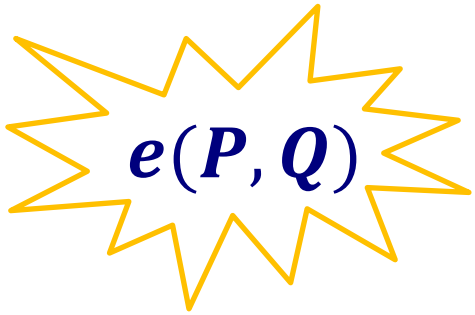
Who	Mcyc	Processor	Coord
Hankerson, Scott, Menezes	10.0	Core 2	projective
Naehrig, Niederhagen, Schwabe	4.40	Core 2	projective
Beuchat et al. 2010	2.95	Core 2	projective
Beuchat et al. 2010	2.90	Nehalem	projective
Aranha et al. 2011	2.20	Core 2	projective
Aranha et al. 2011	2.00	Nehalem	projective
Aranha et al. 2011	1.56	Phenom II	projective
Zavattoni et al. 2013	1.51	Sandy Bridge	projective
Mitsunari 2013	1.33	Haswell	projective
Mitsunari 2013	1.17	Haswell+`mulx`	projective
[new]	1.42	Sandy Bridge	projective
[new]	1.21	Haswell	projective
[new]	1.18	Haswell+`mulx`	projective
Acar, Lauter,Naehrig, Shumow	15.6	Core 2	affine
[new]	2.43	Nehalem	affine

Benchmarks

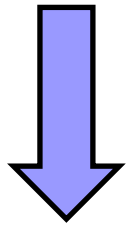
- "Pairings are too slow for practical consideration."
- Sandy Bridge, Wei Dai's Crypto++ 5.6.2 for RSA and RELIC for pairings:

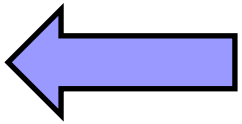
Operation	Timings (Mcyc)
RSA 3072 signing	25.56
Affine pairing	1.94
Projective pairing	1.43
RSA 3072 verification	0.29

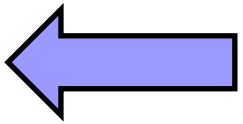
Benchmarks



- "Wait! How about storage, how about embedded processors?"
- On a MICAz, the RELIC library computes a pairing at the 2^{80} security level in ~8s.
- State-of-the-art RSA 1024 takes ~10s.
- … both quickly taking all available 4 KiB RAM, hence long way to go... ☺







Future directions and challenges

- At ECC 2004 I pointed out a number of challenges in pairing-based crypto.
- Many of those have been solved (e.g. BN curves!).
- It seems only fit to make some new ones now!

Future directions

- What is the speed limit for pairings?
  - Clearly, this depends on the platform.
  - Perhaps the real question is: what platforms are closest to ideal?
- Higher security
  - Better parameters than BLS curves?
- Very constrained platforms (e.g. SIM cards).
  - Internet of Things, including embedded processors and WSNs.

Challenges

- Implement projective (and affine) pairings taking (substantially) less than 10^6 cycles.
- Find a family of pairing-friendly curves of prime order à la BN, but $k/g \sim 30$ (or prove that none exists).
- Implement pairings convincingly on a very constrained platform (not a coprocessor).

QUESTIONS?