Public Key Cryptography – Performance Comparison and Benchmarking

Tanja Lange

Department of Mathematics

Technical University of Denmark

tanja@hyperelliptic.org

28.08.2006

What is the Fastest Public Key Cryptosystem?

Fastest Public Key system . . .

- for key agreement?
- for electronic signature?
- for encryption?
- for key generation?

Decision will depend on application and resources like

- low power embedded device,
- personal computer or laptop, or
- server handling millions of connections.

Even with complete specifications it is hard to decide from the theoretical description which is faster.

RSA

- $n = p \cdot q$, p, q primes.
- Choose random e, compute $d \equiv e^{-1} \bmod \varphi(n)$ (if possible); $\varphi(n) = (p-1)(q-1)$.
- Public key (e, n).
- Secret key d.

Example: signing of message m with hash function h (school book version!)

- Signer:
  Compute $h(m)$ and send $s \equiv h(m)^d \bmod n$ as signature.
- Verifier:
  Compute $h' \equiv s^e \bmod n$.
  Accept only if $h' = h(m)$.

RSA

- $n = p \cdot q$, p, q primes.
- Choose small e, compute $d \equiv e^{-1} \bmod \varphi(n)$ (if possible); $\varphi(n) = (p-1)(q-1)$.
- Public key (e, n).
- Secret key d.

Example: signing of message m with hash function h (school book version!)

- Signer:
  Compute $h(m)$ and send $s \equiv h(m)^d \bmod n$ as signature.
- Verifier:
  Compute $h' \equiv s^e \bmod n$.
  Accept only if $h' = h(m)$.

Costs of RSA signature

- Signer computes hash and computes 1 modular exponentiation.
- Verifier computes hash and computes 1 modular exponentiation.
- If RSA with small public exponent is used, verification gets cheaper.

Costs for encryption are similar.

DL in finite fields

- General system parameters:
  - p prime power, $\mathbb{F}_p$ finite field with p elements.
  - g generator of group of order q, with $q \mid p-1$.
- Choose random a, compute $h = g^a$.
- Public key h. (Note that public parameters are not included, they are assumed to be system-wide parameters.)
- Secret key a.

Schnorr signature on m

- Signer:
  - Choose k, compute $K = g^k$, compute $H = h(K, m)$.
  - Compute $S \equiv k - aH \bmod q$.
  - Signature is (H, S).
- Verifier:
  - Retrieve h.
  - Compute $K' = g^S \cdot h^H$.
  - Accept only if $H = h(K', m)$.
- Works since

$$K' = g^S \cdot h^H = g^{k-aH} \cdot g^{aH} = g^k = K$$

if the signature was computed correctly.

Costs of Schnorr signature

- Signer computes hash, 1 modular exponentiation and one multiplication modulo q (much smaller than modulus p).
- Verifier computes hash and 2 modular exponentiations (usually done as 1 multiexponentiation).

So on first sight this is more expensive than RSA – if $p \sim n$.

Elliptic curve

$$E : y^2 + \underbrace{(a_1x + a_3)}_{h(x)} y = \underbrace{x^3 + a_2x^2 + a_4x + a_6}_{f(x)}, \;\; h, f \in \mathbb{F}_q[x].$$

Group: $E(\mathbb{F}_q) = \{\, (x,y) \in \mathbb{F}_q^2 \,:\, y^2 + h(x)y = f(x) \,\} \cup \{\, P_\infty \,\}$

Often $q = 2^r$ or $q = p$, prime. Isomorphic transformations lead to

$$y^2 = f(x) \qquad q \text{ odd},$$

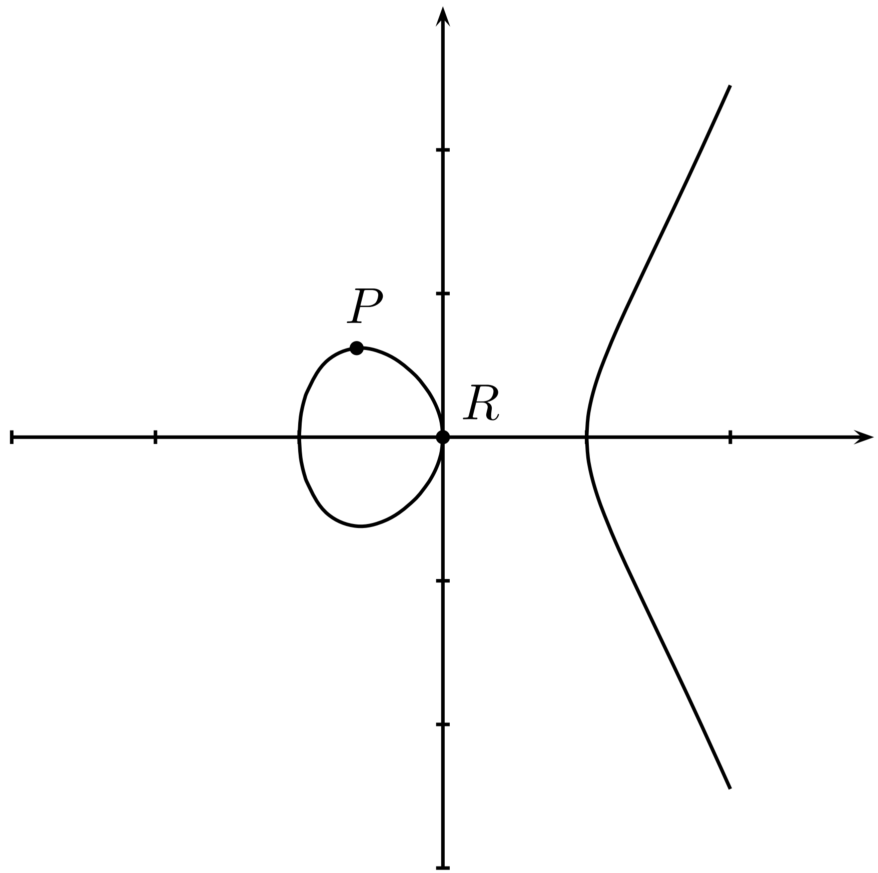
for

$$\begin{aligned} y^2 + xy &= x^3 + a_2x^2 + a_6 \\ y^2 + y &= x^3 + a_4x + a_6 \end{aligned} \qquad q = 2^r,$$

curve non-supersingula
curve supersingular

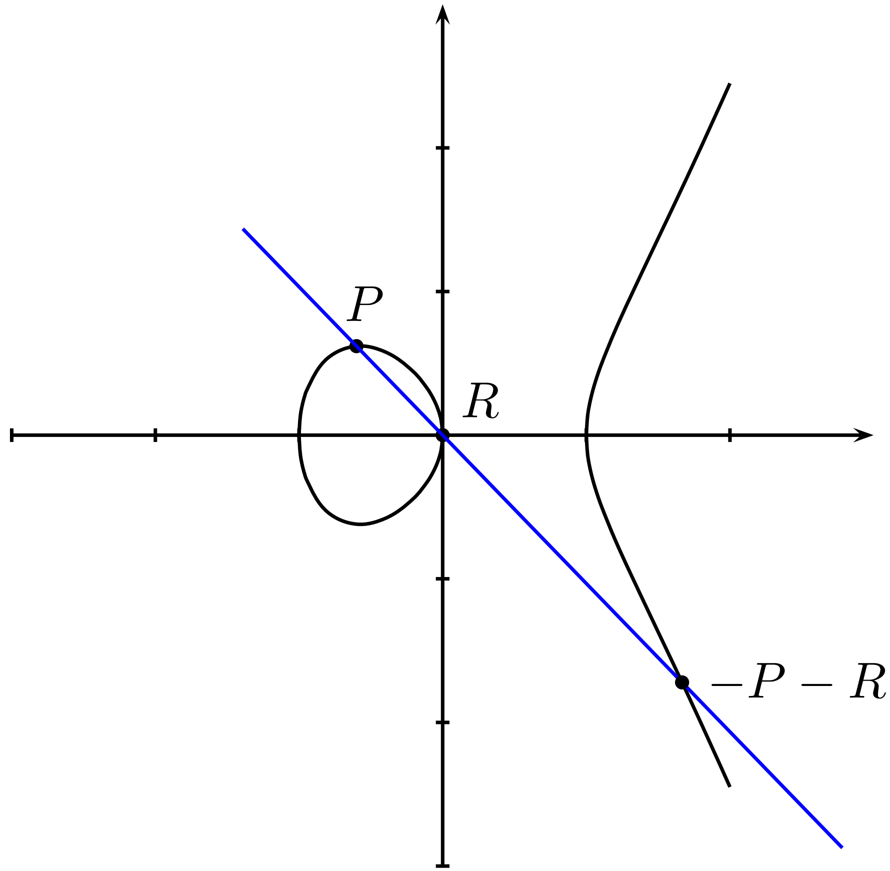
Group Law in $E(\mathbb{R}), h = 0$

$y^2 = x^3 - x$



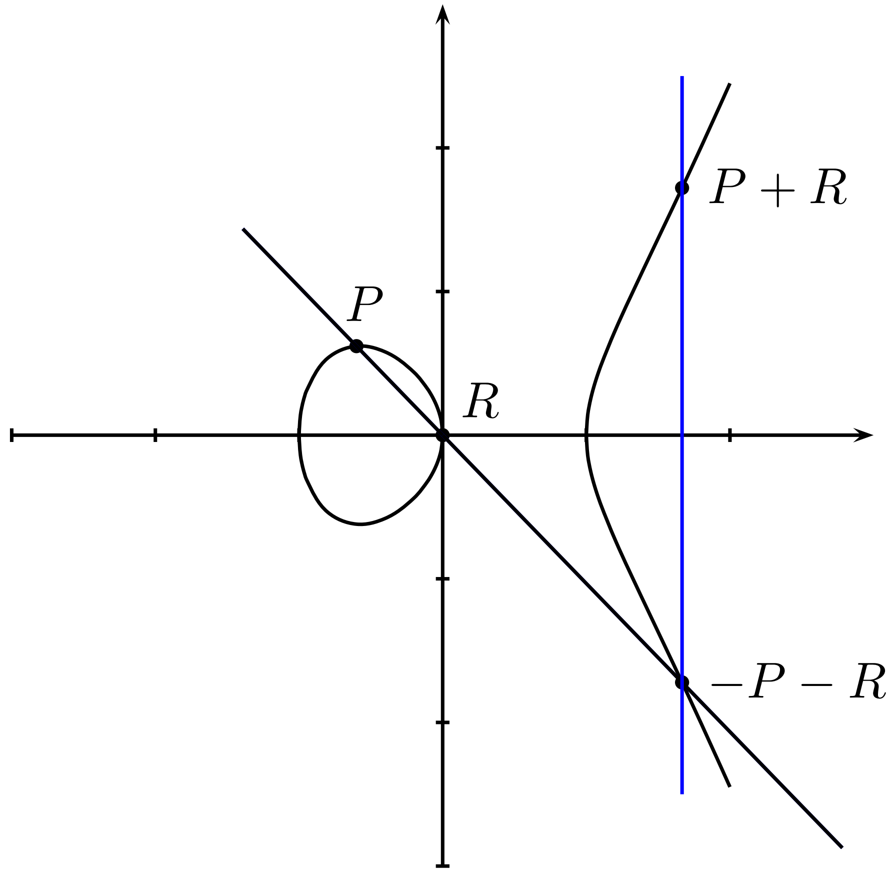
Group Law in $E(\mathbb{R}), h = 0$

$y^2 = x^3 - x$



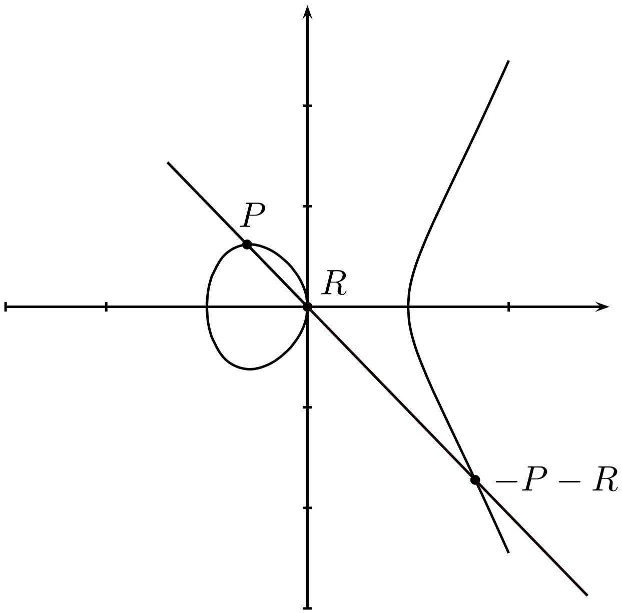
Group Law in $E(\mathbb{R}), h = 0$

$y^2 = x^3 - x$



Group Law (q odd)

$$E : y^2 = x^3 + a_4 x + a_6, \; a_i \in \mathbb{F}_q$$



Line $y = \lambda x + \mu$ has slope

$$\lambda = \frac{y_R - y_P}{x_R - x_P}.$$

Equating gives

$$(\lambda x + \mu)^2 = x^3 + a_4 x + a_6.$$

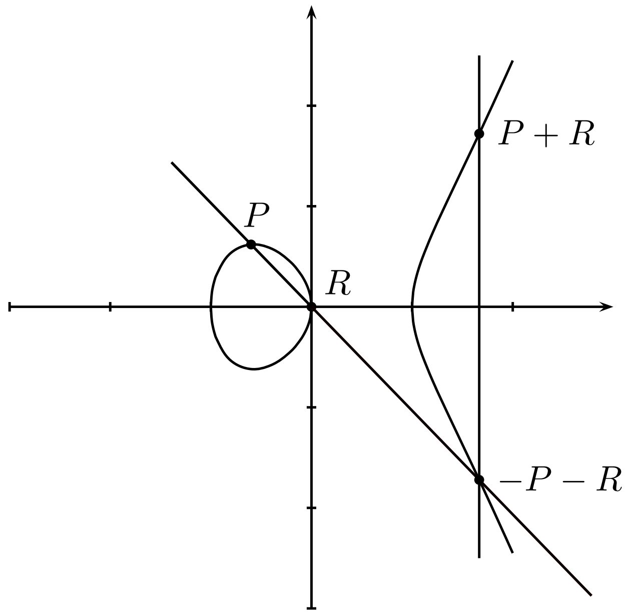
This equation has 3 solutions, the x-coordinates of P, R and $-P - R$, thus

$$(x - x_P)(x - x_R)(x - x_{-P-R}) = x^3 - \lambda^2 x^2 + (a_4 - 2\lambda\mu)x + a_6 -$$

$$x_{-P-R} = \lambda^2 - x_P - x_R$$

Group Law (q odd)

$$E: y^2 = x^3 + a_4 x + a_6,\ a_i \in \mathbb{F}_q$$



Point P is on line, thus

$y_P = \lambda x_P + \mu$, i.e.
$\mu = y_P - \lambda x_P$,

and

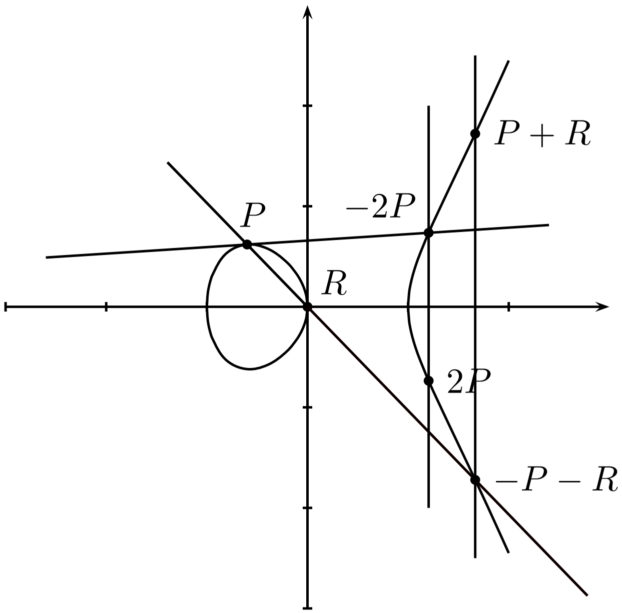
$$\begin{aligned} y_{-P-R} &= \lambda x_{-P-R} + \mu \\ &= \lambda x_{-P-R} + y_P - \lambda x_P \\ &= \lambda(x_{-P-R} - x_P) + y_P \end{aligned}$$

Point $P+R$ has the same x-coordinate but negative y-coordinate:

$$x_{P+R} = \lambda^2 - x_P - x_R, \quad y_{P+R} = \lambda(x_P - x_{-P-R}) - y_P$$

Group Law (q odd)

$$E : y^2 = x^3 + a_4 x + a_6,\ a_i \in \mathbb{F}_q$$



In general, for $(x_P, y_P) \neq (x_R, -y_R)$:

$$\begin{aligned}(x_P, y_P) + (x_R, y_R) &= \\ &= (x_{P+R}, y_{P+R}) = \\ &= (\lambda^2 - x_P - x_R, \lambda(x_P - x_{P+R}) - y_P),\end{aligned}$$

where

$$\lambda = \begin{cases} (y_R - y_P)/(x_R - x_P) & \text{if } x_P \neq x_R, \\ (3x_P^2 + a_4)/(2y_P) & \text{else.} \end{cases}$$

$\Rightarrow$ Addition and Doubling need
1 I, 2M, 1S and 1 I, 2M, 2S, respectively

Systems based on ECC

Use the group of points instead of finite field in previous signature scheme.

- General system parameters:
  - $\mathbb{F}_q$ finite field with q elements.
  - E elliptic curve over $\mathbb{F}_q$, group order n.
  - P generator of group of order ℓ with $\ell \mid n$.
- Choose random a, compute $Q = [a]P$.
- Public key Q. (Note that public parameters are not included, they are assumed to be system-wide parameters.)
- Secret key a.

ECDSA signature on m

- Signer:
  - Choose k, compute $K = [k]P$.
  - Compute $s \equiv (k^{-1}(h(m) - ah(K))) \bmod \ell$
  - Signature is (K, s).
- Verifier:
  - Retrieve Q.
  - Compute $R_1 = [h(K)]Q \oplus [s]K$.
  - Compute $R_2 = [h(m)]P$.
  - Accept only if $R_1 = R_2$.
- Works since

$$\begin{aligned} R_1 &= [h(K)]Q \oplus [s]K = [ah(K)]P \oplus [ks]P \\ &= [ah(K) + h(m) - ah(K)]P = [h(m)]P = R_2. \end{aligned}$$

Costs of ECDSA

- Signer computes hash, 1 scalar multiplication and one multiplication modulo ℓ.
- Verifier computes hash , 1 scalar multiplication and 1 multi-exponentiation.

Warning: this is not the most efficient version, one multi-exponentiation is sufficient.

- So the number and type of operations is similar to Schnorr signature,
- however, each group operation on the elliptic curve is much more complicated than in finite fields (actually composed of several finite field operations).
- BUT finite fields do NOT have the same size.

Fair comparison

Systems should offer same level of security!

- RSA is broken if n can be factored. There are subexponential algorithms for factoring.
- Schnorr's signature scheme is broken if a can be obtained from $h = g^a$. There are subexponential algorithms to solve the DLP in finite fields.
- ECDSA is broken if a can be obtained from $Q = [a]P$. We are not aware of any subexponential algorithm for solving the DLP on elliptic curves. Best known attacks on carefully chosen curves need $O(\sqrt{\ell})$ operations, so the DLP has exponential security.
- Hyperelliptic curves of small genus behave like elliptic curves.

Implications

- Asymptotic behavior does not capture constants. ECRYPT's `www.ecrypt.eu.org` report on key-sizes states security of RSA as

$$s(n) = \left(\frac{64}{9}\right)^{1/3} \log_2(e)(n \ln 2)^{1/3}(\ln(n \ln 2))^{2/3} - 14.$$

- Sizes of n, p for RSA and Schnorr signature scheme grow much faster than group size ℓ in ECDSA.
- Often mentioned current recommendations are RSA or finite fields with 1024 bit modulus; ECC in fields of 160 bits.
- Often only discrete steps stated and contradicting answers.
  Nice compilation `www.keylength.com`.

Comparison seems possible

- For current security level (and thus also for future ones) ECDSA is faster than RSA or DSA in general.
- RSA with small public key has fast verification. Security is unclear.
- Implementations in soft- and hardware confirm this.
- Benchmarks are done (at least on one machine at a time), results usually point in the same direction and confirm above statement.
- Have theoretical comparison and real world measures (Pentium cycles, Athlon cycles, etc.)
- However, often implementor prefers his own system – are his results significant for other systems?

Other systems

- There are many more systems that are much harder to put into comparison:
  - SFLASH is an HFE based signature system.
  - Merkle-tree signatures are based on hash functions.
  - Coding based systems are around almost since the beginning of public key cryptography and still unbroken.
  - NTRU a lattice based encryption seems secure, NTRUsign is controversial.
- These systems are interesting in general.
- Additional advantage: they seem to resist quantum computing attacks (while RSA and DL would be broken completely).

eBATS

- eBATS: ECRYPT Benchmarking of Asymmetric Systems `www.ecrypt.eu.org/ebats`
- benchmark real world measures (Pentium cycles, Athlon cycles, etc.)
- for generating keys, signing, verifying, encrypting, decrypting;
- measure key bytes, signed-message bytes, ciphertext bytes, etc.
- of any submitted BAT (Benchmarkable Asymmetric Tool), i.e. public key system for signing, encrypting or key sharing.
- Benchmarking tool is called BATMAN (Benchmarking of Asymmetric Tools on Multiple Architectures, Non-Interactively).

Advantages

- BAT is submitted by person supporting this particular system.
- Only systems that find at least one interested person are considered.
- Independent benchmarking on a variety of machines.
- Unifying API so that code can run anywhere.
- Wrapper to make fixed length encryption/signature handle arbitrary length ones.
- OpenSSL, GMP and NTL are provided.

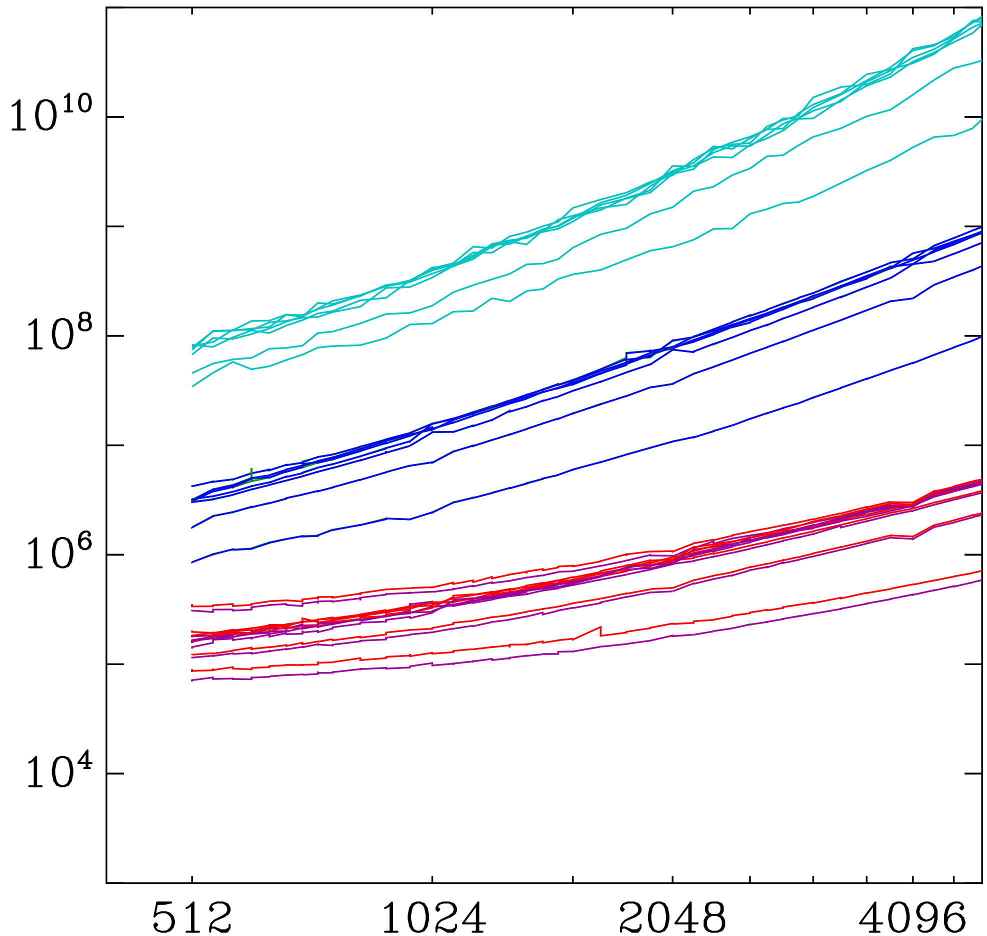
Disadvantages

- Only submitted systems are considered – might miss some systems.
- Result depends on programming abilities of submitter – might be slower than optimal.
- Wrapper might be slower than designated encryption/signature of arbitrary length messages.
- Provided software packages (OpenSSL, GMP, NTL) might not be optimal for small field sizes.

eBATS approach

- Some BATs are provided to guarantee presence of RSA, DL in finite fields.
- BATMAN comes with example BATs.
- Use of OpenSSL, GMP and NTL is optional. A BAT can come with full code for modular reduction etc.
- BATMAN tries all conceivable compiler options, also for included software.
- Source code is put online. Improvements are possible over the full duration of the competition.
- We accept multiple BATs for the same cryptographic primitive (`ronald` is a slow RSA BAT).
- Wrapper is optional. Implementation of full API is very welcome.

Example measurements with `ronald`



Just try to beat `ronald` and submit your BAT!

Example measured on a Pentium 4 f12:

	sflashv2-1	ronald-3 2048
key-gen cycles	462090336	2467681772
secret-key bytes	2823	2048
public-key bytes	19266	256
sign cycles	1908060	63607084
sign 29 bytes	66	256
sign 709 bytes	746	752
verify cycles	667684	575108

Results show which systems are faster.

Example measured on a Pentium 4 f12:

cycles	implementation
29646848	claus-1 (using OpenSSL)
21324260	claus++-1 (using NTL)
13919316	claus++-1 (using GMP)

Results show which implementations are faster.

Note to implementers: GMP is very fast!

claus++-1 measured on different machines:

cycles	CPU
28981828	Intel Pentium 1 52c
27069568	Motorola PowerPC G4
13919316	Intel Pentium 4 f12
11306413	Sun UltraSPARC IV
9892179	AMD Athlon 622
3273274	AMD Athlon 64 X2 fb1
3082045	DEC Alpha 21264 EV6

Results show which computers are faster.

Want to advertise your system/implementation?

- Take a few minutes to turn your software into a BAT (Benchmarkable Asymmetric Tool) and submit it to eBATS.
- Measurements are continuing.
- Major reports in December 2006, July 2007.
- Intermediate announcements on web pages.

www.ecrypt.eu.org/ebats

Submit your BAT


NOW!