Post-quantum cryptanalysis

D. J. Bernstein
University of Illinois at Chicago

Cryptographic speed

What is the fastest
public-key encryption system?
Or public-key signature system?

Cryptographic speed

What is the fastest
public-key encryption system?
Or public-key signature system?

RSA-1024 is quite fast.

Cryptographic speed

What is the fastest public-key encryption system? Or public-key signature system?

RSA-1024 is quite fast. RSA-512 is faster.

Cryptographic speed

What is the fastest
public-key encryption system?
Or public-key signature system?

RSA-1024 is quite fast.
RSA-512 is faster.
RSA-256 is even faster.

Cryptographic speed

What is the fastest
public-key encryption system?
Or public-key signature system?

RSA-1024 is quite fast.
RSA-512 is faster.
RSA-256 is even faster.
This question is stupid.

Cryptographic speed

What is the fastest public-key encryption system with security level $\geq 2^b$?

Cryptographic speed

What is the fastest public-key encryption system with security level $\geq 2^b$?

(Plausible-sounding definition: breaking costs $\geq 2^b$.)

Cryptographic speed

What is the fastest public-key encryption system with security level $\ge 2^b$?

(Plausible-sounding definition: breaking with probability 1 costs $\ge 2^b$.)

Cryptographic speed

What is the fastest public-key encryption system with security level $\geq 2^b$?

(Plausible-sounding definition: for each $\epsilon > 0$, breaking with probability $\geq \epsilon$ costs $\geq 2^b \epsilon$.)

Cryptographic speed

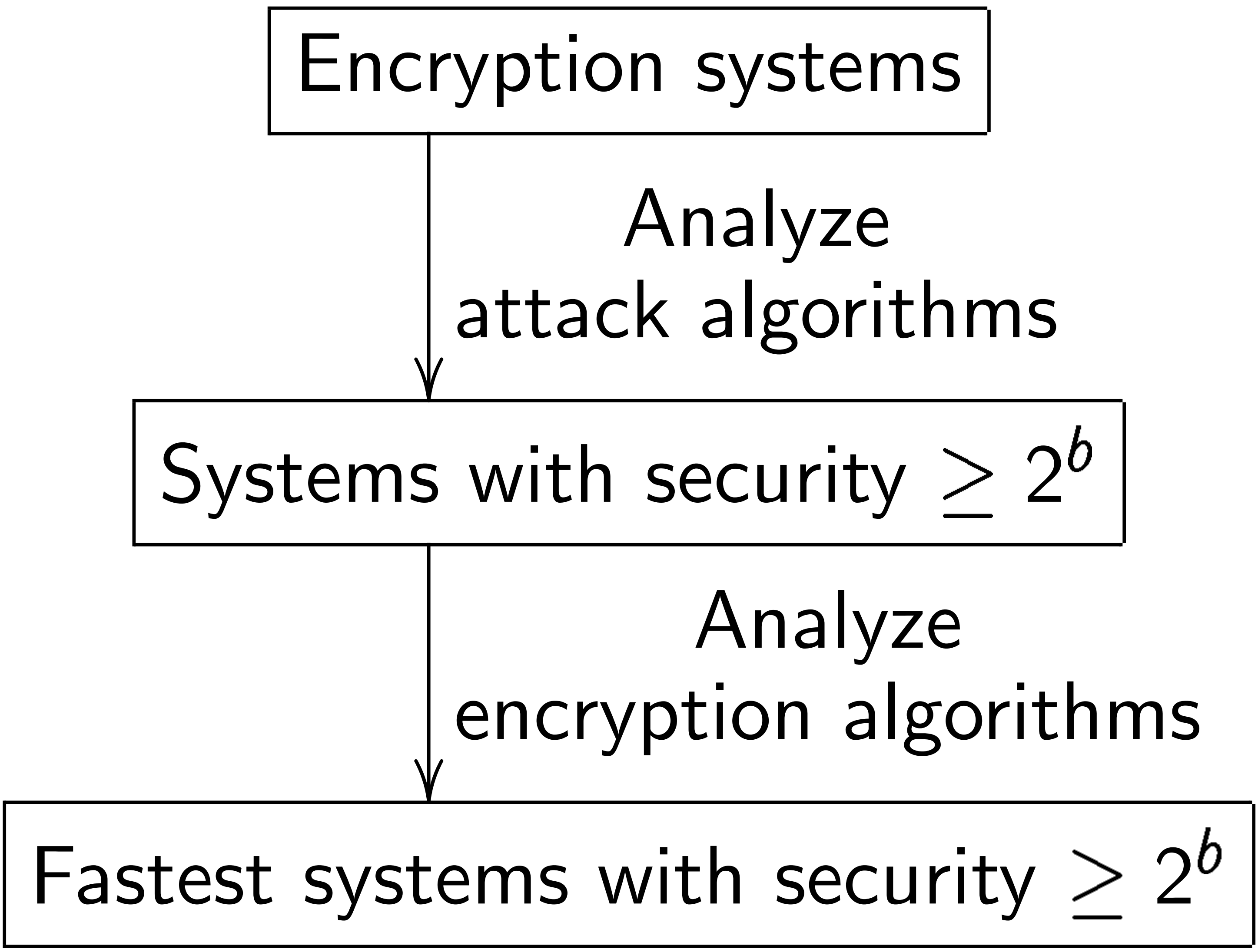
What is the fastest public-key encryption system with security level $\geq 2^b$?

(Plausible-sounding definition: for each $\epsilon > 2^{-b/2}$, breaking with probability $\geq \epsilon$ costs $\geq 2^b \epsilon$.)

Cryptographic speed

What is the fastest public-key encryption system with security level $\geq 2^b$?

How to evaluate candidates:



Two pre-quantum examples

RSA (with small exponent, reasonable padding, etc.):

Factoring n costs $2^{(\lg n)^{1/3+o(1)}}$ by the number-field sieve. Conjecture: this is the optimal attack against RSA.

Key size: Can take $\lg n \in b^{3+o(1)}$ ensuring $2^{(\lg n)^{1/3+o(1)}} \ge 2^b$.

Encryption: Fast exp costs $(\lg n)^{1+o(1)}$ bit operations.

Summary: RSA costs $b^{3+o(1)}$.

ECC (with strong curve/$\mathbf{F}_q$, reasonable padding, etc.):

ECDL costs $2^{(1/2+o(1))\lg q}$ by Pollard's rho method.
Conjecture: this is the optimal attack against ECC.

Can take $\lg q \in (2+o(1))b$.

Encryption: Fast scalar mult costs $(\lg q)^{2+o(1)} = b^{2+o(1)}$.

Summary: ECC costs $b^{2+o(1)}$.
Asymptotically faster than RSA: i.e., more security for same cost.
Bonus: also $b^{2+o(1)}$ *decryption*.

These analyses are quite crude.

To really understand costs
need much more precise
analysis and optimization
of attack algorithms
and encryption algorithms.

e.g. **R**-algebraic complexity
of size-n DFT over **C**,
when n is a power of 2:
$n^{1+o(1)}$: Gauss FFT.

These analyses are quite crude.

To really understand costs need much more precise analysis and optimization of attack algorithms and encryption algorithms.

e.g. **R**-algebraic complexity of size-n DFT over **C**, when n is a power of 2:
$n^{1+o(1)}$: Gauss FFT.
$O(n \lg n)$: Gauss FFT.

These analyses are quite crude.

To really understand costs need much more precise analysis and optimization of attack algorithms and encryption algorithms.

e.g. **R**-algebraic complexity of size-n DFT over **C**, when n is a power of 2:
$n^{1+o(1)}$: Gauss FFT.
$O(n \lg n)$: Gauss FFT.
$(5 + o(1))n \lg n$: Gauss FFT.

These analyses are quite crude.

To really understand costs need much more precise analysis and optimization of attack algorithms and encryption algorithms.

e.g. **R**-algebraic complexity of size-n DFT over **C**, when n is a power of 2:
$n^{1+o(1)}$: Gauss FFT.
$O(n \lg n)$: Gauss FFT.
$(5 + o(1))n \lg n$: Gauss FFT.
$(4 + o(1))n \lg n$: split-radix FFT.

These analyses are quite crude.

To really understand costs need much more precise analysis and optimization of attack algorithms and encryption algorithms.

e.g. **R**-algebraic complexity of size-n DFT over **C**, when n is a power of 2:
$n^{1+o(1)}$: Gauss FFT.
$O(n \lg n)$: Gauss FFT.
$(5 + o(1))n \lg n$: Gauss FFT.
$(4 + o(1))n \lg n$: split-radix FFT.
$(34/9 + o(1))n \lg n$: tangent FFT.

Cryptanalysis is slowly moving to a realistic model of computation.

A circuit is a 2-dimensional mesh of small parallel gates. Have fast communication *between neighboring gates*. Try to optimize time T as function of area A. See, e.g., classic area-time theorem from 1981 Brent–Kung.

Warning: Naive student model— `a=x[i]` costs 1, like `a=b+c` —gives wildly unrealistic algorithm-scalability conclusions.

"Maybe there's a better attack breaking your 'secure' systems. Maybe security costs far more!"

This is a familiar risk.
This is why the community puts tremendous effort into cryptanalysis: analyzing and optimizing attack algorithms.

Results of cryptanalysis:
Some systems are killed.
Some systems need larger keys but still have competitive cost.
Some systems inspire confidence.

Post-quantum cryptography

Assume that attacker
has a large quantum computer,
making qubit operations
as cheap as bit operations.

(Yes, that's too extreme.
Tweak for more plausibility:
maybe $2^b/b^3$ qubit operations
are similar to 2^b bit operations.)

Consequence of this assumption:
Attacker has old algorithm arsenal
(ECM, ISD, LLL, XL, F4, F5, . . .)
plus Grover and Shor.

Conventional wisdom:
Factoring n costs $(\lg n)^{2+o(1)}$
by Shor (in naive model),
so RSA is dead.
Similarly DSA and ECDSA.

Conventional wisdom:
Factoring n costs $(\lg n)^{2+o(1)}$
by Shor (in naive model),
so RSA is dead.
Similarly DSA and ECDSA.

More careful RSA evaluation:

Can take $\lg n \in 2^{(1/2+o(1))b}$
ensuring $(\lg n)^{2+o(1)} \geq 2^b$.
Can reduce RSA encryption,
decryption, key generation
to $2^{(1/2+o(1))b}$ bit ops,
far below attacker's cost.

Conventional wisdom:
Factoring n costs $(\lg n)^{2+o(1)}$
by Shor (in naive model),
so RSA is dead.
Similarly DSA and ECDSA.

More careful RSA evaluation:

Can take $\lg n \in 2^{(1/2+o(1))b}$
ensuring $(\lg n)^{2+o(1)} \geq 2^b$.
Can reduce RSA encryption,
decryption, key generation
to $2^{(1/2+o(1))b}$ bit ops,
far below attacker's cost.

. . . but other systems are better!
Here are some leading candidates.

Hash-based signatures.
Example: 1979 Merkle hash trees.

Code-based encryption.
Example: 1978 McEliece hidden Goppa codes.

Lattice-based encryption.
Example: 1998 "NTRU."

Multivariate-quadratic-equations signatures.
Example: 1996 Patarin "HFE^{v-}" public-key signature system.

Secret-key cryptography.
Example: 1998 Daemen–Rijmen "Rijndael" cipher, aka "AES."

A hash-based signature system

Standardize a 256-bit hash function H.

Signer's public key: 512 strings $y_1[0], y_1[1], \ldots, y_{256}[0], y_{256}[1]$, each 256 bits.
Total: 131072 bits.

Signature of a message m: 256-bit strings $r, x_1, \ldots, x_{256}$ such that the bits $(h_1, \ldots, h_{256})$ of $H(r, m)$ satisfy
$y_1[h_1] = H(x_1)$, ...,
$y_{256}[h_{256}] = H(x_{256})$.

Signer's secret key:
512 independent uniform
random 256-bit strings
$x_1[0], x_1[1], \ldots, x_{256}[0], x_{256}[1]$.

Signer computes
$y_1[0], y_1[1], \ldots, y_{256}[0], y_{256}[1]$
as $H(x_1[0]), H(x_1[1]), \ldots,$
$H(x_{256}[0]), H(x_{256}[1])$.

To sign m:
generate uniform random r;
$H(r, m) = (h_1, \ldots, h_{256})$;
reveal $(r, x_1[h_1], \ldots, x_{256}[h_{256}])$;
discard remaining x values;
refuse to sign more messages.

This is the “Lamport–Diffie one-time signature system.”

How to sign
more than one message?

Easy answer: “Chaining.”
Signer expands m to include a newly generated public key that will sign next message.

More advanced answers (Merkle et al.)
scale logarithmically with the number of messages signed.

Grover finds $x_1[0]$ from $y_1[0]$ using $\approx 2^{128}$ qubit ops.

Maybe H has some structure allowing faster inversion . . . but most functions don't seem to have such structures.

"SHA-3 competition":
2008: 191 cryptographers submitted 64 proposals for H.
Ongoing: Extensive public review.
2011 status: 5 finalists.
2012: SHA-3 is standardized.

Chaum–van Heijst–Pfitzmann, 1991: $H(a, b) = 4^a 9^b \bmod p$.

Simple, beautiful, structured. Allows “provable security”: e.g., H collisions imply computing a discrete logarithm, when p is chosen sensibly.

Chaum–van Heijst–Pfitzmann, 1991: $H(a, b) = 4^a 9^b \bmod p$.

Simple, beautiful, structured. Allows “provable security”: e.g., H collisions imply computing a discrete logarithm, when p is chosen sensibly.

But very bad cryptography. Horrible security for its speed. Far worse security record than “unstructured” H designs.

Chaum–van Heijst–Pfitzmann, 1991: $H(a, b) = 4^a 9^b \bmod p$.

Simple, beautiful, structured. Allows "provable security": e.g., H collisions imply computing a discrete logarithm, when p is chosen sensibly.

But very bad cryptography. Horrible security for its speed. Far worse security record than "unstructured" H designs.

Some newer efforts to sacrifice security for provability: VSH; 2007 Moore–Russell–Vazirani.

An MQ signature system

Signer's public key:
polynomials $P_1, \ldots, P_{300}$
$\in \mathbf{F}_2[w_1, \ldots, w_{600}]$.

Extra requirements
on each of these polynomials:
degree ≤ 2, no squares;
i.e., linear combination of
$1, w_1, \ldots, w_{600}$,
$w_1 w_2, w_1 w_3, \ldots, w_{599} w_{600}$.

Overall 54090300 bits.

Signature of m:
a 300-bit string r and values $w_1, \ldots, w_{600} \in \mathbf{F}_2$ such that $H(r, m) = (P_1(w_1, \ldots, w_{600}), \ldots, P_{300}(w_1, \ldots, w_{600}))$.

Only 900 bits!

Verifying a signature uses one evaluation of H and millions of bit operations to evaluate $P_1, \ldots, P_{300}$.

Main challenge for attacker:
find bits $w_1, \ldots, w_{600}$
producing specified outputs

$$(P_1(w_1, \ldots, w_{600}), \ldots, P_{300}(w_1, \ldots, w_{600})).$$

Random guess: on average,
only 2^{-300} chance of success.

"XL" etc.: fewer operations,
but still not a threat.

Signer generates public key with secret “HFE^{v-}” structure.

Standardize a degree-450 irreducible polynomial $\varphi \in \mathbf{F}_2[t]$. Define $L = \mathbf{F}_2[t]/\varphi$.

Critical step in signing: finding roots of a secret polynomial in $L[x]$ of degree at most 300.

Secret polynomial is chosen with all nonzero exponents of the form $2^i + 2^j$ or 2^i. (So degree ≤ 288.)

If $x_0, x_1, \ldots, x_{449} \in \mathbf{F}_2$ and
$x = x_0 + x_1 t + \cdots + x_{449} t^{449}$ then
$x^2 = x_0 + x_1 t^2 + \cdots + x_{449} t^{898}$,
$x^4 = x_0 + x_1 t^4 + \cdots + x_{449} t^{1796}$,
etc.

In general, $x^{2^i+2^j}$
is a quadratic polynomial
in the variables $x_0, \ldots, x_{449}$.

Signer's secret key:

invertible 600×600 matrix S;

300×450 matrix T of rank 300;

$Q \in L[x, v_1, v_2, \ldots, v_{150}]$.

Each term in Q
has one of the forms

$\ell x^{2^i+2^j}$ with $\ell \in L$, $2^i < 2^j$,
$2^i + 2^j \leq 300$;

$\ell x^{2^i} v_j$ with $\ell \in L$, $2^i \leq 300$;

$\ell v_i v_j$;

ℓx^{2^i};

ℓv_j;

ℓ.

To compute public key:

Compute $S(w_1, \ldots, w_{600}) = (x_0, \ldots, x_{449}, v_1, \ldots, v_{150})$.

In $L[w_1, \ldots, w_{600}]$
compute $x = \sum x_i t^i$
and $y = Q(x, v_1, v_2, \ldots, v_{150})$
modulo $w_1^2 - w_1, \ldots, w_{600}^2 - w_{600}$.

Write $y = y_0 + \cdots + y_{449} t^{449}$
with $y_i \in \mathbf{F}_2[w_1, \ldots, w_{600}]$.

Compute $(P_1, \ldots, P_{300}) = T(y_0, y_1, \ldots, y_{449})$.

Sign by working backwards.

Given values $(P_1, \ldots, P_{300})$, invert T to obtain values $(y_0, \ldots, y_{449})$. 2^{150} choices; randomize.

Choose $(v_1, \ldots, v_{150})$ randomly. Substitute into $Q(x, v_1, \ldots, v_{150})$ to obtain $Q(x) \in L[x]$.

Solve $Q(x) = y$ for $x \in L$.
If several roots, randomize.
If no roots, start over.

Invert S to obtain signature.

This is an "HFE$^{\text{v}-}$" example.

"HFE": "Hidden Field Equation" $Q(x) = y$.

"−": publish only 300 equations instead of 450.

"v": "vinegar" variables $v_1, \ldots, v_{150}$.

State-of-the-art attack breaks a simplified system with 0 vinegar variables, 1 term in Q.

Can build MQ systems in many other ways.

A code-based encryption system

Receiver's public key:
1800×3600 bit matrix K.

Messages suitable for encryption:
3600-bit strings of "weight 150";
i.e., 3600-bit strings
with exactly 150 nonzero bits.

Encryption of m
is 1800-bit string Km.

Attacker, by linear algebra, can easily work backwards from Km to *some* v such that $Kv = Km$.

Huge number of choices of v. Finding weight-150 choice ("syndrome-decoding K") seems extremely difficult for most choices of K.

Basic information-set decoding:
Choose set of 1800 columns
on which K is invertible.
Work backwards to v
supported in those 1800 columns.
Hope that $v = m$, i.e., that m is
supported in those 1800 columns.

2009 Bernstein:
Trivially apply Grover here.
iterations drops to square root.
But some ISD improvements
now become counterproductive.

New guess: “Some” includes
2011 May–Meurer–Thomae.

Receiver secretly generates a random Goppa code Γ and a random permutation P. Computes public key K as random parity-check matrix for permuted Goppa code ΓP.

Detecting this structure seems even more difficult than syndrome-decoding random K.

Knowing Γ and P allows receiver to decode 150 errors.

My current reading of 2011 Dinh–Moore–Russell:

Using Shor for $\Gamma, \Gamma P \mapsto P$ is very slow (for most Γ) thanks to group structure.

These cryptosystems thus "resist the natural analog of Shor's quantum attack."

This gives "the first rigorous results on the security of the McEliece-type cryptosystems in the face of quantum adversaries, strengthening their candidacy for post-quantum cryptography."

I find this quite puzzling.

1. I don't see how $\Gamma, \Gamma P \mapsto P$ relates to attacking McEliece. The attacker isn't given Γ.

I find this quite puzzling.

1. I don't see how $\Gamma, \Gamma P \mapsto P$ relates to attacking McEliece. The attacker isn't given Γ.

2. Broken variants of McEliece have the same group structure. Are they strong candidates too?

I find this quite puzzling.

1. I don't see how $\Gamma, \Gamma P \mapsto P$ relates to attacking McEliece. The attacker isn't given Γ.

2. Broken variants of McEliece have the same group structure. Are they strong candidates too?

3. The $\Gamma, \Gamma P \mapsto P$ problem is not hard. For almost all Γ, 1999 Sendrier computes $\Gamma, \Gamma P \mapsto P$ in polynomial time.

I find this quite puzzling.

1. I don't see how $\Gamma, \Gamma P \mapsto P$ relates to attacking McEliece. The attacker isn't given Γ.

2. Broken variants of McEliece have the same group structure. Are they strong candidates too?

3. The $\Gamma, \Gamma P \mapsto P$ problem is not hard. For almost all Γ, 1999 Sendrier computes $\Gamma, \Gamma P \mapsto P$ in polynomial time.

There are many interesting non-quantum algorithms.

How to make progress

1. Learn the target landscape.

2. Learn the existing attacks.
Add them into your toolbox.

3. Look for faster attacks.
e.g. FXL/"hybrid GB" has an outer search; apply Grover!

4. Analyze algorithms precisely.
Otherwise you miss most algorithm speedups.

Daniel J. Bernstein
Johannes Buchmann
Erik Dahmen
Editors

Post-Quantum Cryptography

Springer

Bernstein: "Introduction to post-quantum cryptography."

Hallgren, Vollmer: "Quantum computing."

Buchmann, Dahmen, Szydlo: "Hash-based digital signature schemes."

Overbeck, Sendrier: "Code-based cryptography."

Micciancio, Regev: "Lattice-based cryptography."

Ding, Yang: "Multivariate public key cryptography."

Latest updates:

pqcrypto.org:
introduction and bibliography.

PQCrypto conference series:
PQCrypto 2006 in Leuven.
PQCrypto 2008 in Cincinnati.
PQCrypto 2010 in Darmstadt.
PQCrypto 2011 soon in Taipei.
Hotel deadline: 30 September.