High-speed cryptography

D. J. Bernstein
University of Illinois at Chicago

The UNIX command “`dig +dnssec -t any se @a.ns.se`” sends a 31-byte packet to IP address 192.36.144.107, which sends back a 3974-byte response.

The UNIX command "`dig +dnssec -t any se @a.ns.se`" sends a 31-byte packet to IP address 192.36.144.107, which sends back a 3974-byte response.

Can forge same 31-byte packet with return address 198.41.0.4. 192.36.144.107 sends 3974-byte packet to 198.41.0.4.

Can repeat trillions of times, flooding 198.41.0.4 with data.

"Distributed denial-of-service attack with 100× amplification."

“Project Titan,” starting 2007: VeriSign has been spending >$100000000 to upgrade the Internet’s `.com` DNS servers.

In a typical day in 2008, these servers together handled $5 \cdot 10^6$ clients sending $35 \cdot 10^9$ queries.

Beginning of 2009: $38 \cdot 10^9$.

“Project Titan,” starting 2007: VeriSign has been spending >$100000000 to upgrade the Internet’s `.com` DNS servers.

In a typical day in 2008, these servers together handled $5 \cdot 10^6$ clients sending $35 \cdot 10^9$ queries.

Beginning of 2009: $38 \cdot 10^9$.

What about non-typical days when servers were under attack?

VeriSign says: Will be prepared for flood of $4 \cdot 10^{12}$ packets/day totalling $2 \cdot 10^{15}$ bytes/day.

“DNSSEC,” starting 1993:
Cryptographic protection for DNS.
Millions of dollars of U.S. grants.

DNSSEC designers decided that busy servers can’t handle cryptographic computations.
$\Rightarrow$ DNSSEC skips encryption and *precomputes* all signatures.
$\Rightarrow$ Massive usability problems from signature storage, signature expiration, dynamic data, etc.

“DNSSEC,” starting 1993:
Cryptographic protection for DNS.
Millions of dollars of U.S. grants.

DNSSEC designers decided that busy servers can't handle cryptographic computations.
$\Rightarrow$ DNSSEC skips encryption and *precomputes* all signatures.
$\Rightarrow$ Massive usability problems from signature storage, signature expiration, dynamic data, etc.

16 years later: The Internet has $\approx$ 80000000 `*.com` names.

"DNSSEC," starting 1993:
Cryptographic protection for DNS.
Millions of dollars of U.S. grants.

DNSSEC designers decided that busy servers can't handle cryptographic computations.
⇒ DNSSEC skips encryption and *precomputes* all signatures.
⇒ Massive usability problems from signature storage, signature expiration, dynamic data, etc.

16 years later: The Internet has
≈ 80000000 `*.com` names.
≈ 300 have DNSSEC signatures.

If cryptography is too slow,
users turn it off.

If it *might be* too slow,
Internet designers turn it off
or screw it up—
reducing security,
compromising usability, etc.

If cryptography is too slow,
users turn it off.

If it *might be* too slow,
Internet designers turn it off
or screw it up—
reducing security,
compromising usability, etc.

My response:
1. Build strong cryptography
that's self-evidently fast enough
to protect every Internet packet.
2. Implement it in usable form.
3. Deploy it!

Performance measurement

European Union has funded NESSIE project (2000–2003), ECRYPT I network (2004–2008), ECRYPT II network (2008–2012).

Now 11 partners, 3 virtual labs. VAMPIRE is the "Virtual Application and Implementation Lab" led by Tanja Lange, Christof Paar.

One VAMPIRE project: eBACS, "ECRYPT Benchmarking of Cryptographic Systems."
http://bench.cr.yp.to

eBACS toolkit includes
346 software implementations of
48 public-key primitives,
26 stream ciphers, and
50 hash functions.
Thanks to all contributors!

eBACS has collected, published
measurements on 78 machines;
112 machine-ABI combinations.

Each implementation is
recompiled 1226 times
with various compiler options
to identify best working option
for implementation, machine.

Implementation

European Union has also funded "Computer-Aided Cryptography Engineering" project (2008–2010).

12 partners, 5 work packages. NaCl, "Networking and Cryptography library," is main task of CACE WP 2, led by D. J. Bernstein, Tanja Lange.

C NaCl news: first release! Thanks to Adam Langley@Google, Matthew Dempsky@Mochi Media.

http://nacl.cace-project.eu

Other libraries exist for networking and cryptography: e.g., OpenSSL library.

Compared to previous libraries, NaCl improves security; NaCl improves usability; and NaCl improves speed.

First release prioritizes high-speed high-security cryptographic applications that can't survive without state-of-the-art cryptography; e.g., usable security for DNS.

The critical primitives

For every new client:
Use public-key cryptography
to share a secret key.

For every new packet:
Use secret-key cryptography
to authenticate packet and
(if desired) encrypt packet.

Main bottleneck can be
sharing secret keys or
encrypting packets or
authenticating packets.
Depends on data volume,
number of clients, etc.

Standard encryption method:
xor nth message with
$\text{AES}_k(n, 1), \text{AES}_k(n, 2), \ldots$
where k is the secret key.

Typical code: 20 cycles/byte.

2008 Bernstein–Schwabe:
10.6 cycles/byte on Core 2,
14.1 cycles/byte on P4, etc.;
very low per-packet overhead.

2009 Käsper–Schwabe:
7.8 cycles/byte on Core 2;
low per-packet overhead.

Mild slowdown for 256-bit key.

Improve speed by changing cipher.

2004.11: eSTREAM ("ECRYPT Stream Cipher Project") calls for submissions of stream ciphers. Receives 34 submissions from 97 cryptographers around the world.

2008.04: After two hundred papers and several conferences, eSTREAM selects portfolio of four fast software ciphers (and some small hardware ciphers).

Much faster ciphers than AES. e.g. 2.6 cycles/byte on Core 2 for my "Salsa20/12" cipher.

Combine with advances in packet-authentication speed.

What does this mean in practice?

A 2.5GHz Intel Core 2 Quad Q9300 CPU costs US$225. Complete computer: $400.

This CPU has 4 cores. Each core carries out $2.5 \cdot 10^9$ cycles/second.

CPU encrypts and authenticates 10^{11} typical-size packets/day; keeps up with Gbps network while leaving most cycles free for other work.

But what about public-key costs?

Need, e.g., 256-bit elliptic-curve single-scalar multiplication for every new client—i.e., every new public key.

What if there are billions of different public keys? Too many to cache them all?

What if an attacker sends flood of new public keys? (Hopefully not amplified!)

Will CPU be able to keep up?

Bernstein, ECC 2005, PKC 2006:

640838 Pentium M cycles for high-security Diffie–Hellman; specifically, Curve25519 ECDH. Also good speeds on other CPUs.

More than twice as fast as previous DH results at similar security level.

Curve25519 is the Montgomery curve $y^2 = x^3 + 486662x^2 + x$ modulo the prime $2^{255} - 19$. Tuned for speed, security, twist-security, et al.

Gaudry–Thomé, SPEED 2007, ECC 2007:

New mpFq library produces speed records on Core 2.

386000 cycles for Curve25519.

888000 cycles for binary (i.e., characteristic 2).

405000 cycles for genus 2.
Faster genus-2 curves exist, but so far nobody has computed a secure twist-secure example.

687000 cycles for binary genus 2.

Recall Project Titan:
VeriSign spending $>$\$100000000
to be prepared for flood
of $4 \cdot 10^{12}$ packets/day.

Worst case: Every packet
has a new public key.
$4 \cdot 10^{12}$ Curve25519's/day.

Recall Project Titan:
VeriSign spending $>$\$100000000
to be prepared for flood
of $4 \cdot 10^{12}$ packets/day.

Worst case: Every packet
has a new public key.
$4 \cdot 10^{12}$ Curve25519's/day.

Can handle these computations
using Gaudry–Thomé software
on < 2000 of the \$400 computers.

Expensive but should fit
easily into VeriSign's budget.
Can we further reduce costs?

Speed records now broken in three (combinable!) ways.

Dai, reported May 2009, building on 2009 Costigan–Schwabe: better use of CPU instructions.

Speed records now broken
in three (combinable!) ways.

Dai, reported May 2009, building on 2009 Costigan–Schwabe: better use of CPU instructions.

Galbraith–Lin–Scott, ECC 2008, Eurocrypt 2009:
Twist $E(\mathbf{F}_{p^2})$ for $E/\mathbf{F}_p$;
exploit a fast endomorphism.
Current implementation uses Edwards-form $E/\mathbf{F}_{2^{127}-1}$:
$x^2 + y^2 = 1 + 42x^2y^2$.

Speed records now broken in three (combinable!) ways.

Dai, reported May 2009, building on 2009 Costigan–Schwabe: better use of CPU instructions.

Galbraith–Lin–Scott, ECC 2008, Eurocrypt 2009:
Twist $E(\mathbf{F}_{p^2})$ for $E/\mathbf{F}_p$;
exploit a fast endomorphism.
Current implementation uses Edwards-form $E/\mathbf{F}_{2^{127}-1}$:
$x^2 + y^2 = 1 + 42x^2y^2$.

Bernstein, Crypto 2009:
something completely different.

Edwards curves

Fix a non-binary field k.

Edwards addition law for curve $x^2 + y^2 = 1 + dx^2y^2$ with $d \in k - \{0, 1\}$:

$$x_3 = \frac{x_1y_2 + y_1x_2}{1 + dx_1x_2y_1y_2},$$

$$y_3 = \frac{y_1y_2 - x_1x_2}{1 - dx_1x_2y_1y_2}.$$

Every elliptic curve over k with a point of order 4 is birationally equivalent over k to an Edwards curve over k.

2007 Bernstein–Lange:

If d is not a square in k then

$$\{(x, y) \in k \times k : \\ x^2 + y^2 = 1 + dx^2y^2\}$$

is a commutative group under Edwards addition law.

The denominators

$1 + dx_1x_2y_1y_2$,

$1 - dx_1x_2y_1y_2$

are never zero.

No exceptional cases!

1995 Bosma–Lenstra theorem:
"The smallest cardinality of a complete system of addition laws on E equals two."

1995 Bosma–Lenstra theorem:
"The smallest cardinality of a complete system of addition laws on E equals two." . . . meaning:
Any addition formula
for a Weierstrass curve E
in projective coordinates
must have exceptional cases
in $E(\overline{k}) \times E(\overline{k})$, where
$\overline{k} =$ algebraic closure of k.

1995 Bosma–Lenstra theorem:
"The smallest cardinality of a complete system of addition laws on E equals two." . . . meaning:
Any addition formula for a Weierstrass curve E in projective coordinates must have exceptional cases in $E(\overline{k}) \times E(\overline{k})$, where $\overline{k}$ = algebraic closure of k.

Edwards addition formula has exceptional cases for $E(\overline{k})$ . . . but not for $E(k)$.
We do computations in $E(k)$.

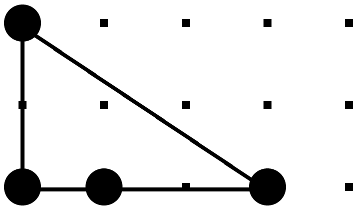
Completeness eases
implementations, avoids
simple side-channel attacks.

What about elliptic curves
without points of order 4?
What about elliptic curves
over binary fields?

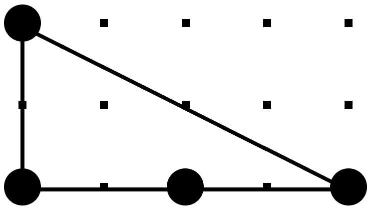
Continuing project (B.–L.):
For *every* elliptic curve E,
find complete addition law for E
with best possible speeds.

Maybe slower than Edwards
but maybe still useful.

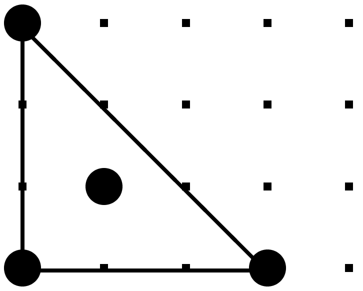
Some Newton polygons



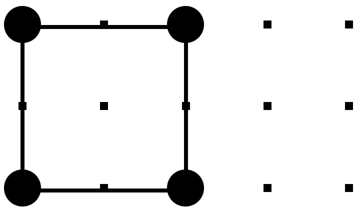
Short Weierstrass



Jacobi quartic



Hessian



Edwards

1893 Baker: genus is generically number of interior points.

2000 Poonen–Villegas classified polygons with 1 interior point.

How to generalize Edwards?

Design decision: want quadratic in x and in y.

Design decision: want $x \leftrightarrow y$ symmetry.

$$\begin{matrix} d_{20} & d_{21} & d_{22} \\ d_{10} & d_{11} & d_{21} \\ d_{00} & d_{10} & d_{20} \end{matrix}$$

Curve shape $d_{00} + d_{10}(x + y) + d_{11}xy + d_{20}(x^2 + y^2) + d_{21}xy(x + y) + d_{22}x^2y^2 = 0$.

Suppose that $d_{22} = 0$:

$$\begin{array}{ccc} d_{20} & d_{21} & \cdot \\ \\ d_{10} & d_{11} & d_{21} \\ \\ d_{00} & d_{10} & d_{20} \end{array}$$

Genus 1 $\Rightarrow$ $(1,1)$ is an interior point $\Rightarrow$ $d_{21} \neq 0$.

Homogenize:

$d_{00}Z^3 + d_{10}(X+Y)Z^2 + d_{11}XYZ + d_{20}(X^2+Y^2)Z + d_{21}XY(X+Y) = 0$.

Points at ∞ are $(X : Y : 0)$ with $d_{21}XY(X+Y) = 0$: i.e., $(1:0:0)$, $(0:1:0)$, $(1:-1:0)$.

Study $(1:0:0)$ by setting $y = Y/X$, $z = Z/X$ in homogeneous curve equation:

$$d_{00}z^3 + d_{10}(1+y)z^2 + d_{11}yz + d_{20}(1+y^2)z + d_{21}y(1+y) = 0.$$

Nonzero coefficient of y so $(1:0:0)$ is nonsingular. Addition law cannot be complete (unless k is tiny).

So we require $d_{22} \neq 0$.

Points at ∞ are $(X : Y : 0)$ with $d_{22}X^2Y^2 = 0$: i.e., $(1 : 0 : 0)$, $(0 : 1 : 0)$.

Study $(1 : 0 : 0)$ again:
$d_{00}z^4 + d_{10}(1+y)z^3 + d_{11}yz^2 + d_{20}(1+y^2)z^2 + d_{21}y(1+y)z + d_{22}y^2 = 0$.

Coefficients of $1, y, z$ are 0 so $(1 : 0 : 0)$ is singular.

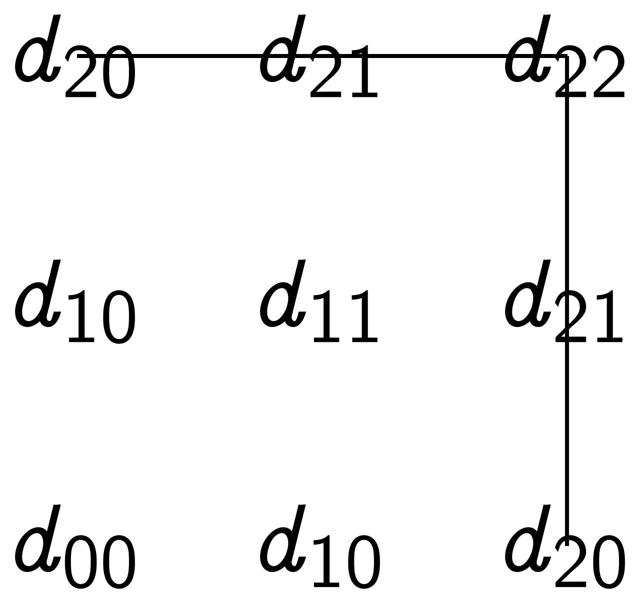
Put $y = uz$, divide by z^2 to blow up singularity:

$$d_{00}z^2 + d_{10}(1 + uz)z + d_{11}uz + d_{20}(1 + u^2z^2) + d_{21}u(1 + uz) + d_{22}u^2 = 0.$$

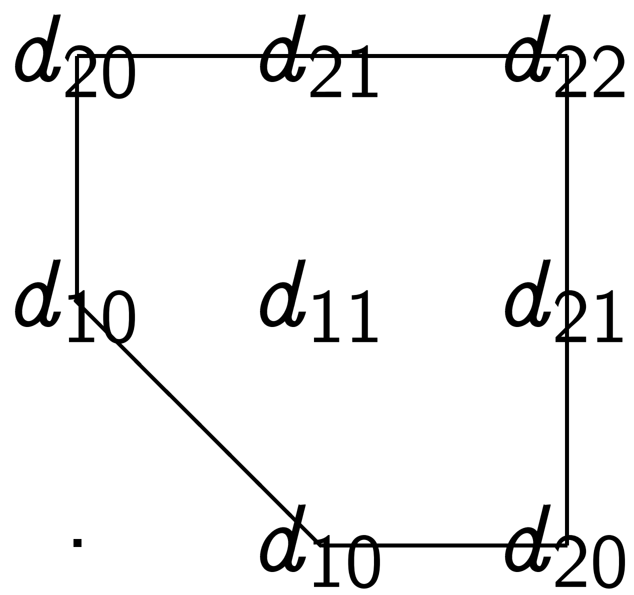
Substitute $z = 0$ to find points above singularity: $d_{20} + d_{21}u + d_{22}u^2 = 0.$

We require the quadratic $d_{20} + d_{21}u + d_{22}u^2$ to be irreducible in k.
Special case: complete Edwards, $1 - du^2$ irreducible in k.

In particular $d_{20} \neq 0$:



Design decision: Explore a deviation from Edwards. Require $d_{00} = 0$, $d_{10} \neq 0$.



Now $(0, 0)$ is on curve.

Design decision:
$(0, 0)$ is neutral element.
Then $-(x, y) = (y, x)$.

By scaling x, y
and scaling curve equation
can limit $d_{10}, d_{11}, d_{20}, d_{21}, d_{22}$
to three degrees of freedom.

Can choose other neutral
elements, as in Edwards.
Warning: bad choice can produce
surprisingly expensive negation.

B.–L.–Rezaeian Farashahi,
CHES 2008:
complete addition law for
"binary Edwards curves"
$d_1(x+y)+d_2(x^2+y^2) =$
$(x+x^2)(y+y^2)$.
Covers all ordinary elliptic curves
over $\mathbf{F}_{2^n}$ for $n \geq 3$.
Also surprisingly fast!

B.–L.–Rezaeian Farashahi,
CHES 2008:
complete addition law for
"binary Edwards curves"
$d_1(x+y)+d_2(x^2+y^2) =$
$(x+x^2)(y+y^2)$.
Covers all ordinary elliptic curves
over $\mathbf{F}_{2^n}$ for $n \geq 3$.
Also surprisingly fast!

B.–L., posted April 2009:
complete addition law for
another specialization
covering all the NIST curves
over *non-binary* fields.

Consider, e.g., the curve

$$x^2 + y^2 = x + y + txy + dx^2y^2$$

with $d = -1$ and

$$t = \begin{array}{r} 78751018041117252545420999954 \\ 76717646453854506081463020284 \\ 13956511758592017999 \end{array}$$

over $\mathbf{F}_p$ where $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$.

Note: d is non-square in $\mathbf{F}_p$.

Birationally equivalent to standard "NIST P-256" curve $v^2 = u^3 - 3u + a_6$ where

$$a_6 = \begin{array}{r} 41058363725152142129326129780 \\ 04726840911444101599372555483. \\ 5256314039467401291 \end{array}$$

An addition law for
$x^2 + y^2 = x + y + txy + dx^2y^2$,
complete if d is not a square:

$$x_3 = \frac{\begin{array}{l} x_1 + x_2 + (t-2)x_1x_2 + \\ (x_1 - y_1)(x_2 - y_2) + \\ dx_1^2(x_2y_1 + x_2y_2 - y_1y_2) \end{array}}{\begin{array}{l} 1 - 2dx_1x_2y_2 - \\ dx_1^2(x_2 + y_2 + (t-2)x_2y_2) \end{array}};$$

$$y_3 = \frac{\begin{array}{l} y_1 + y_2 + (t-2)y_1y_2 + \\ (y_1 - x_1)(y_2 - x_2) + \\ dy_1^2(y_2x_1 + y_2x_2 - x_1x_2) \end{array}}{\begin{array}{l} 1 - 2dy_1y_2x_2 - \\ dy_1^2(y_2 + x_2 + (t-2)y_2x_2) \end{array}}.$$

Note on computing addition laws:
An easy Magma script uses Riemann–Roch to find addition law given a curve shape.

Are those laws nice? No!
Find lower-degree laws by Monagan–Pearce algorithm, ISSAC 2006; or by evaluation at random points on random curves.

Are those laws complete? No!
But always seems easy to find complete addition laws among low-degree laws where denominator constant term $\neq 0$.

Batch binary Edwards

(2009 Bernstein)

Can multiply 251-bit polynomials over $\mathbf{F}_2$ using a straight-line sequence of 33200 bit operations (ANDs and XORs).

Much better than schoolbook 125501 bit operations.

Most of the improvement is standard Karatsuba, but also have some new multiplication speedups.

Put $d = t^{57} + t^{54} + t^{44} + 1$, $k = \mathbf{F}_2[t]/(t^{251} + t^7 + t^4 + t^2 + 1)$.

Can compute $n, P \mapsto nP$ on the binary Edwards curve $d(x+x^2+y+y^2) = (x+x^2)(y+y^2)$ over k using a straight-line sequence of 45076017 bit operations.

This curve meets the usual paranoid security criteria.

Operation count benefits from small number of mults and from completeness.

Handle 128 inputs in parallel using 128-bit vector operations: “bitsliced ECC.”

Bottleneck: at most three operations per Core 2 cycle, so > 117385 cycles per scalar multiplication.

Current prototype code actually uses 346317 cycles.

Right now I’m working on a new instruction scheduler. Also expect smaller bit-operation counts for Koblitz curves, genus 2, etc.