Speed records for cryptographic software: an update

D. J. Bernstein
University of Illinois at Chicago
NSF CCR–9983950

Elliptic-curve cryptography

Define $p = 2^{226} - 5$; p is prime.

Consider the elliptic curve
$y^2 = x^3 + 7530x^2 + x$ over $\mathbf{Z}/p$.

For $n \in \mathbf{Z}$: Multiply
$(53(2^{224} - 1)/(2^8 - 1), \ldots)$ by n
on the curve to get $(K_n, \ldots)$ or ∞.

Compressed Diffie-Hellman

Your secret key: $a \in 16\mathbf{Z}$
with $0 \le K_a < 2^{224}$.
Your public key: K_a.

My secret key: $b \in 16\mathbf{Z}$
with $0 \le K_b < 2^{224}$.
My public key: K_b.

Our shared secret: K_{ab}.

Given a, K_b, can compute K_{ab}
in $< 10^6$ cycles.

This is a very fast curve;
somewhat faster than NIST P-224.

Elliptic-curve DH is much faster
than other forms of DH
at this presumed security level.

Some computational tools

Curve shape $y^2 = x^3 + c_2 x^2 + x$
allows very fast compressed
curve multiplication.
(Montgomery 1987)

Small $c_2 = 7530$ saves some mults.

Curve multiplication is only
≈ 2000 multiplications in $\mathbf{Z}/p$
and one inversion in $\mathbf{Z}/p$.

Represent integers as sums of
floating-point numbers
at specified scales.

Use radix $R = 2^{28.25}$;
sparse $p = R^8 - 5$ allows
very fast reduction mod p.

Floatasm: new language
and generation/verification tools
for straight-line fp code.
Keep the multiplier busy!

Hashing with rare collisions

For 128-bit m_i's: Define
$h_r(m_0, m_1, \ldots, m_{\ell-1}) =$
$(r^{\ell+1} + m_0 r^{\ell} + \cdots + m_{\ell-1} r)$
$\bmod (2^{130} - 5)$.

$2^{130} - 5$ is prime $> 2^{128}$
to allow 128-bit m_i's.
Works well with radix 2^{26}.

Secret-key message authentication

I save $(k, r) = \text{SHA-256}(K_a, K_{ab})$.

You transmit nth message m
as $n, m, (F_k(n) + h_r(m)) \bmod 2^{128}$,
using strong secret-key cipher F.
(Easy to encrypt too.)

I reject n', m', s' if n' is old or if
$s' \neq (F_k(n') + h_r(m')) \bmod 2^{128}$.

Can compute $h_r(m)$ in $< 10^3$ cycles for typical lengths of m.

No precomputation needed, thanks to wide m_i's.

This is the fastest known method to handle a flood of forgeries while communicating with known users.

Equation verification

To check a ring equation
such as $s^2 = tn + fh$ where
s, t, n, f, h have thousands of bits:

Reduce s, t, n, f, h modulo
secret 115-bit prime ℓ.
Compute $s^2 - tn - fh \bmod \ell$.

Public-key signature verification

Signature of m under public key n is (r, h, f, s, t) where
r is a random 256-bit string,
h is a cryptographic hash of (r, m),
$f \in \{1, -1, 2, -2\}$, and
$s^2 = tn + fh$.

Check $s^2 = tn + fh$ in $< 10^4$ cycles for 3072-bit public keys.

This is the fastest known method at its presumed security level.

Hello, chip designers

Common chips have multiplier computing 128-bit product, then rounding to 64 bits.

Small additional cost:
Provide the whole product, and a 136-bit adder.
Roughly $4\times$ speedup in arithmetic.