# A complete software implementation of NIST P-224

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

cr.yp.to/nistp224.html

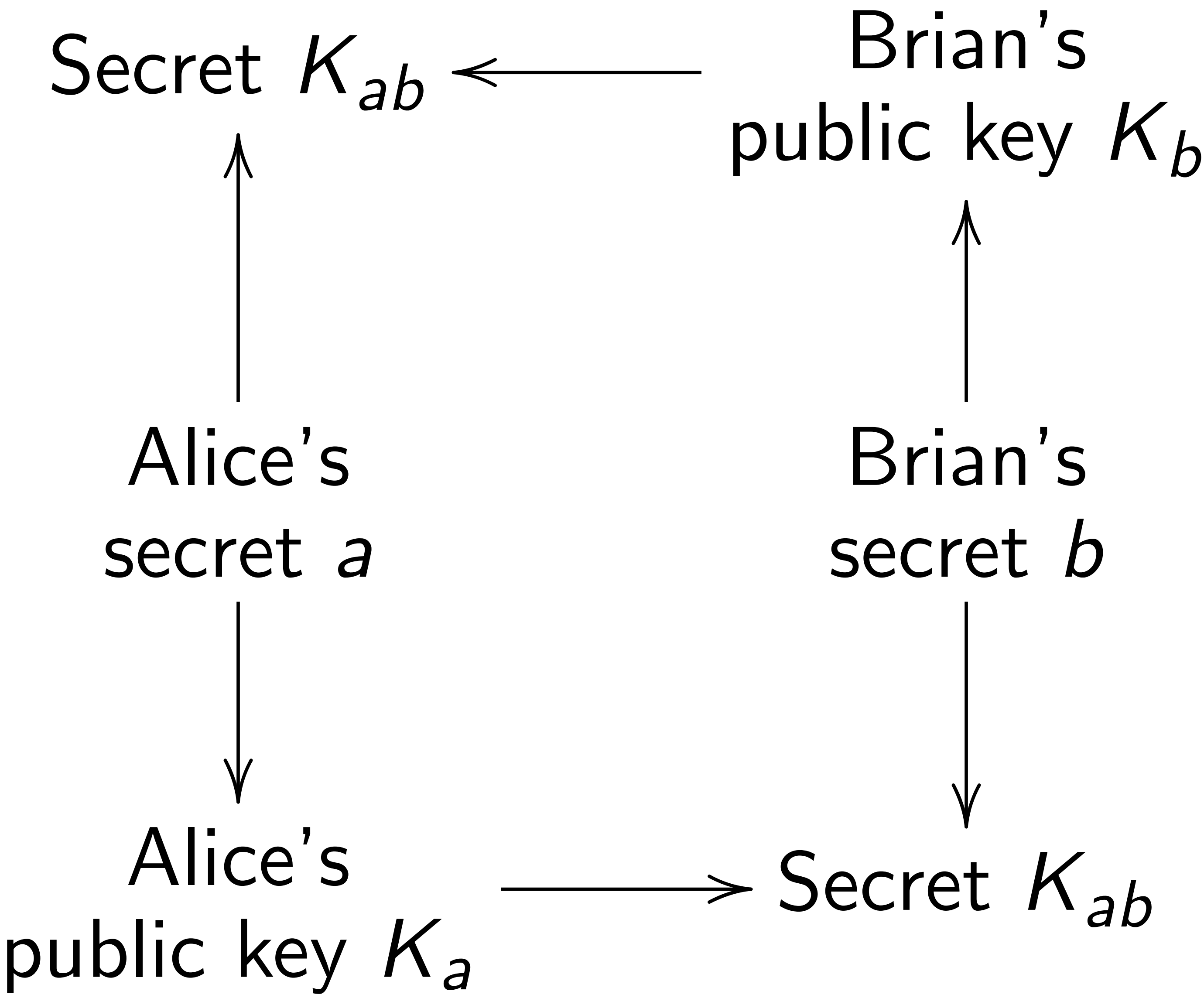
NIST P-224 is the elliptic curve $y^2 = x^3 - 3x + c_6$ over $\mathbf{Z}/p$.

Here $c_6 = 18958286285566608$
$000408668544493926$
$41550468096867932$
$107578723467256 4$

and $p = 2^{224} - 2^{96} + 1$.

Multiply $(10(2^{224}-1)/(2^8-1), \ldots)$ by $n$ on the curve to get $(K_n, \ldots)$, for $n \in (\mathbf{Z}/\#\text{curve}(\mathbf{Z}/p))^*$.

# Compressed Diffie-Hellman



# What `nistp224` does

`nistp224` is a new program
to compute $K_{ab}$ given $a, K_b$.

Alice puts 28 random bytes into `A`,
28 newlines into `K1`.

```
cat A K1 | nistp224 > KA
cat A KB | nistp224 > KAB
```

Also a C-language library:

```
unsigned char a[28];
unsigned char kb[28];
unsigned char kab[28];
nistp224(kab,kb,a);
```

58612 bytes for library on PIII.

# Speed of version 0.76

Typical cycle counts, typical $a$'s:

| $x$ | $x, y$ | |
|---:|---:|---|
| 595683 | 522639 | Athlon |
| 785900 | 668566 | UltraSPARC |
| 835530 | 734731 | Pentium II |
| 943244 | 827360 | Pentium 4 |
| 1120824 | 985097 | Pentium |
| 1166080 | 1019027 | RS64-III |

# Floating-point arithmetic

A 53-**bit fp number**
is a real number $2^e f$
with $e, f \in \mathbf{Z}$ and $|f| < 2^{53}$.

Round each real number $z$ to
closest 53-bit fp number, $\mathrm{fp}_{53}\, z$:
$|z - \mathrm{fp}_{53}\, z| \le 2^{e-1}$ if $|z| \le 2^{e+53}$.
Round halves to even.

**Floating-point add**:
Given 53-bit fp numbers $r, s$,
compute $\text{fp}_{53}(r + s)$. (Or $-$.)

**Floating-point multiply**:
Given 53-bit fp numbers $r, s$,
compute $\text{fp}_{53}(rs)$.

**Fused multiply-accumulate**:
Given 53-bit fp numbers $r, s, t$,
compute $\text{fp}_{53}(rs + t)$.

In one cycle, UltraSPARC does one floating-point addition and one floating-point multiplication, subject to limits on $e$.
Results available after 3 cycles.

RS64-III does one addition or multiplication or fused mac.
At most 4 in a row.
Results available after 5 cycles.

# Carrying

If $r$ is a 53-bit fp number
and $|r| \leq 2^{e+51}$:

Define $\alpha_e = 3 \cdot 2^{e+51}$,
$r_1 = \mathrm{fp}_{53}(\mathrm{fp}_{53}(r + \alpha_e) - \alpha_e)$,
$r_0 = \mathrm{fp}_{53}(r - r_1)$.

Then $r_1 \in 2^e\mathbf{Z}$, $|r_0| \leq 2^{e-1}$,
and $r = r_0 + r_1$.
(Kahan 1965, et al.)

# Arithmetic mod $p$

Can build big-integer arithmetic using floating-point operations. (Veltkamp 1968; Dekker 1971)

`nistp224` uses $\mathbf{Z}[t] \cap \overline{\mathbf{Z}}[2^{56/3}t] = \left\{\sum_{i\geq 0} g_i t^i : g_i \in 2^{\lceil 56i/3\rceil}\mathbf{Z}\right\}$.

$\mathbf{Z}[t] \to \mathbf{Z}/p$ by $g \mapsto g(1)$.

Normally use small polynomials:
$r = r_0 + r_1 t + r_2 t^2 + \cdots + r_{11} t^{11}$
with $|r_i| \le 0.51 \cdot 2^{\lceil 56(i+1)/3 \rceil}$.

$r_0 \in 2^0\mathbf{Z}$, $|r_0| \le 0.51 \cdot 2^{19}$.
$r_1 \in 2^{19}\mathbf{Z}$, $|r_1| \le 0.51 \cdot 2^{38}$.
$r_2 \in 2^{38}\mathbf{Z}$, $|r_2| \le 0.51 \cdot 2^{56}$.
$r_3 \in 2^{56}\mathbf{Z}$, $|r_3| \le 0.51 \cdot 2^{75}$.
$r_4 \in 2^{75}\mathbf{Z}$, $|r_4| \le 0.51 \cdot 2^{94}$.
$r_5 \in 2^{94}\mathbf{Z}$, $|r_5| \le 0.51 \cdot 2^{112}$.
etc.

Use fp to compute $rs$
given small $r, s$:

$r_0 s_0 \in 2^0 \mathbf{Z}$, $|r_0 s_0| \le 0.27 \cdot 2^{38}$
so $r_0 s_0 = \mathrm{fp}_{53}\, r_0 s_0$;
similarly $r_0 s_1 + r_1 s_0 =$
$\mathrm{fp}_{53}(\mathrm{fp}_{53}\, r_0 s_1 + \mathrm{fp}_{53}\, r_1 s_0)$;
etc.

Could use Karatsuba.

Eliminate $t^{22}, t^{21}, \ldots, t^{16}$
and then $t^{15}, t^{14}, t^{13}$ using
$2^{411}t^{22} \equiv 2^{283}t^{15} - 2^{187}t^{10}$,
$2^{392}t^{21} \equiv 2^{264}t^{14} - 2^{168}t^{9}$,
etc.

$(rs)_{10} + 2^{-128}(rs)_{17} - 2^{-224}(rs)_{22}$
is a 53-bit fp number.

Carry from $t^8$ to $t^9$ to
$t^{10}$ to $t^{11}$ to $t^{12}$.
Eliminate $t^{12}$.
Carry from $t^0$ to $t^1$ to
$t^2$ to $t^3$ to $t^4$ to $t^5$ to
$t^6$ to $t^7$ to $t^8$ to $t^9$.

Can reduce latency by
doing a few more carries.

# Faster squaring

$(r^2)_1 = 2r_0r_1$,
$(r^2)_2 = 2r_0r_2 + r_1r_1$,
$(r^2)_3 = 2(r_0r_3 + r_1r_2)$, etc.

Precompute $2r_0, \ldots, 2r_{10}$.
11 doublings instead of 21.

Similarly compute and reduce
$r^2 - 8s$, $r(4s - u) - 8v^2$, etc.

# Complete reduction mod $p$

Define $p_1 = 2^{-224} + 2^{-352} - 2^{-448}$.
If $x \in \mathbf{Z}$, $|x| < 2^{230}$,
then $\lfloor x/p \rfloor = \left\lfloor p_1 x + 2^{-225} \right\rfloor$, so
$x \bmod p = x - p \left\lfloor p_1 x + 2^{-225} \right\rfloor$.

Can compute this using fp.

# Elliptic-curve arithmetic

Use Jacobian coordinates.
(Miller 1985, et al.)

$(x, y, z) \in (\mathbf{Z}/p)^3$, with $z \neq 0$
and with $y^2 = x^3 - 3xz^4 + c_6 z^6$,
represents $(x/z^2, y/z^3)$ on curve.

Use small polynomials $q, r, s$
to represent $x, y, z$.

# Elliptic-curve doubling

Given $(x_1, y_1, z_1)$ with $z_1 \neq 0$:

$2(x_1/z_1^2, y_1/z_1^3) = (x_2/z_2^2, y_2/z_2^3)$
where $\delta = z_1^2$, $\gamma = y_1^2$, $\beta = x_1\gamma$,
$\alpha = 3(x_1 - \delta)(x_1 + \delta)$,
$x_2 = \alpha^2 - 8\beta$, $z_2 = 2y_1z_1$,
$y_2 = \alpha(4\beta - x_2) - 8\gamma^2$.

4 squares, 4 mults, 8 reduces.

`nistp224` computes

$\delta = \text{reduce}\, s_1^2$,

$\gamma = \text{reduce}\, r_1^2$,

$\beta = \text{reduce}\, q_1\gamma$,

$\alpha = \text{reduce}\, 3(q_1 - \delta)(q_1 + \delta)$,

$q_2 = \text{reduce}(\alpha^2 - 8\beta)$,

$s_2 = \text{reduce}((r_1 + s_1)^2 - \gamma - \delta)$,

$r_2 = \text{reduce}(\alpha(4\beta - q_2) - 8\gamma^2)$.

5 squares, 3 mults, 7 reduces.

# Elliptic-curve addition

Given $(x_1, y_1, z_1)$ and $(x_2, y_2, z_2)$ with $z_1 \neq 0$, $z_2 \neq 0$, and $(x_1/z_1^2, y_1/z_1^3) \neq (x_2/z_2^2, y_2/z_2^3)$: Use 4 squares and 12 mults to obtain sum $(x_3, y_3, z_3)$.

Again eliminate one reduction. Could again trade mult for square.

Some of the intermediate results are $z_1^2$, $z_1^3$, $z_2^2$, $z_2^3$.

When reusing $(x_1, y_1, z_1)$, also reuse $z_1^2, z_1^3$.

(Chudnovsky, Chudnovsky 1987; Cohen, Miyaji, Ono 1998)

# Elliptic-curve multiplication

$a_0, \ldots, a_{27} \in \{0, 1, \ldots, 255\}$.
Define $a = 2^{216}(a_0 + 120) + 2^{208}(a_1 - 136) + \cdots + (a_{27} - 136)$.

`nistp224` uses simplest base-16 chain for $a$, coeffs $\{-8, -7, \ldots, 7\}$.
225 doubles, $\leq 59$ adds.
Could eliminate a few adds.
Could exploit initial $z = 1$.

# Plans: better scheduling

Worst-case $a$, using $x, y$:
385372 floating-point mults,
523578 floating-point adds.

678099 UltraSPARC cycles.

Rearrange operations
to reduce gap.

# Plans: better computers

Many things to try.

MMX, SSE, SSE2, etc.
Simultaneous integer/fp.

Suggestions for chip designers:
`FCARRY` $r_1, r_0, k$ carries
multiple of $2^k$ from $r_0$ to $r_1$.
`FMCARRY` multiplies and carries.