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 $\mathbb{Z}/p$.

Here $c_6 = 189582862855666608$
$00040866854449392$
$64155046809686793$
$21075787234672564$

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 $n \in (\mathbb{Z}/\#\text{curve}(\mathbb{Z}/p))^*$.
Compressed Diffie-Hellman

Secret $K_{ab}$ ← Brian’s public key $K_b$

Alice’s secret $a$ ← Alice’s public key $K_a$

Brian’s secret $b$ → Secret $K_{ab}$
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 $K_1$.

cat A K1 | nistp224 > KA
cat A KB | nistp224 > KAB
Also a C-language library:

```c
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:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>595683</td>
<td>522639</td>
<td>Athlon</td>
</tr>
<tr>
<td>785900</td>
<td>668566</td>
<td>UltraSPARC</td>
</tr>
<tr>
<td>835530</td>
<td>734731</td>
<td>Pentium II</td>
</tr>
<tr>
<td>943244</td>
<td>827360</td>
<td>Pentium 4</td>
</tr>
<tr>
<td>1120824</td>
<td>985097</td>
<td>Pentium</td>
</tr>
<tr>
<td>1166080</td>
<td>1019027</td>
<td>RS64-III</td>
</tr>
</tbody>
</table>
Floating-point arithmetic

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

Round each real number $z$ to closest 53-bit fp number, $fp_{53} z$:

$|z - fp_{53} z| \leq 2^{e-1}$ if $|z| \leq 2^{e+53}$.

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 $\alpha_e = 3 \cdot 2^{e+51}$,

$r_1 = \text{fp}_{53}(\text{fp}_{53}(r + \alpha_e) - \alpha_e)$,

$r_0 = \text{fp}_{53}(r - r_1)$.

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 $\mathbb{Z}[t] \cap \overline{\mathbb{Z}}[2^{56/3} t] = \left\{ \sum_{i \geq 0} g_i t^i : g_i \in 2^{[56i/3]} \mathbb{Z} \right\}$.

$\mathbb{Z}[t] \rightarrow \mathbb{Z}/p$ 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 \( |r_i| \leq 0.51 \cdot 2^{\lceil 56(i+1)/3 \rceil} \).

\( r_0 \in 2^0 \mathbb{Z}, \ |r_0| \leq 0.51 \cdot 2^{19}. \)
\( r_1 \in 2^{19} \mathbb{Z}, \ |r_1| \leq 0.51 \cdot 2^{38}. \)
\( r_2 \in 2^{38} \mathbb{Z}, \ |r_2| \leq 0.51 \cdot 2^{56}. \)
\( r_3 \in 2^{56} \mathbb{Z}, \ |r_3| \leq 0.51 \cdot 2^{75}. \)
\( r_4 \in 2^{75} \mathbb{Z}, \ |r_4| \leq 0.51 \cdot 2^{94}. \)
\( r_5 \in 2^{94} \mathbb{Z}, \ |r_5| \leq 0.51 \cdot 2^{112}. \)

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

$$r_0s_0 \in 2^0 \mathbb{Z}, \ |r_0s_0| \leq 0.27 \cdot 2^{38}$$

so $r_0s_0 = \text{fp}_{53} r_0s_0$;

similarly $r_0s_1 + r_1s_0 = \text{fp}_{53}(\text{fp}_{53} r_0s_1 + \text{fp}_{53} r_1s_0)$;

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

\[(r^2)_1 = 2r_0 r_1,\]
\[(r^2)_2 = 2r_0 r_2 + r_1 r_1,\]
\[(r^2)_3 = 2(r_0 r_3 + r_1 r_2), \text{ etc.}\]

Precompute \(2r_0, \ldots, 2r_{10}.\)

11 doublings instead of 21.

Similarly compute and reduce \(r^2 - 8s, r(4s - u) - 8v^2, \text{ etc.}\)
Complete reduction mod $p$

Define $p_1 = 2^{-224} + 2^{-352} - 2^{-448}$.

If $x \in \mathbb{Z}$, $|x| < 2^{230}$, then $\lfloor x/p \rfloor = \lfloor p_1x + 2^{-225} \rfloor$, so $x \mod p = x - p \lfloor p_1x + 2^{-225} \rfloor$.

Can compute this using fp.
Elliptic-curve arithmetic

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

\((x, y, z) \in (\mathbb{Z}/p)^3\), with \(z \neq 0\) and with \(y^2 = x^3 - 3xz^4 + c_6z^6\), 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 \(z_1 \neq 0\):

\[
2\left(\frac{x_1}{z_1^2}, \frac{y_1}{z_1^3}\right) = \left(\frac{x_2}{z_2^2}, \frac{y_2}{z_2^3}\right)
\]

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
\[
\begin{align*}
\delta &= \text{reduce } s_1^2, \\
\gamma &= \text{reduce } r_1^2, \\
\beta &= \text{reduce } q_1 \gamma, \\
\alpha &= \text{reduce } 3(q_1 - \delta)(q_1 + \delta), \\
q_2 &= \text{reduce } (\alpha^2 - 8\beta), \\
s_2 &= \text{reduce } ((r_1 + s_1)^2 - \gamma - \delta), \\
r_2 &= \text{reduce } (\alpha(4\beta - q_2) - 8\gamma^2).
\end{align*}
\]

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 \( a = 2^{216}(a_0 + 120) + 2^{208}(a_1 - 136) + \cdots + (a_{27} - 136) \).

\texttt{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.