Counting points as a video game

D. J. Bernstein
University of Illinois at Chicago

Want efficient computation of secure twist-secure genus-2 C **with very small coefficients** for fastest known Diffie–Hellman. Can't do that with CM.

This talk focuses on algorithms; does not report any computations. Need results today? Ask Gaudry.

But first an advertisement. . .

1985 H. Lange–Ruppert "Complete systems of addition laws on abelian varieties":

$A(\overline{k})$ has a complete system of addition laws, degree $\leq (3,3)$. Symmetry $\Rightarrow$ degree $\leq (2,2)$.

"The proof is nonconstructive. . . To determine explicitly a complete system of addition laws requires tedious computations already in the easiest case of an elliptic curve in Weierstrass normal form."

1985 Lange–Ruppert:
Explicit complete system
of 3 addition laws
for short Weierstrass curves.

Reduce formulas to 53 monomials
by introducing extra variables
$x_i y_j + x_j y_i$, $x_i y_j - x_j y_i$.
I won't copy the formulas here.

1987 Lange–Ruppert
"Addition laws on elliptic curves
in arbitrary characteristics":
Explicit complete system
of 3 addition laws
for long Weierstrass curves.

$$
\begin{aligned}
Y_3^{(2)} = {} & Y_1^2 Y_2^2 + a_1 X_2 Y_1^2 Y_2 + (a_1 a_2 - 3a_3)\, X_1 X_2^2 Y_1 \\
& + a_3 Y_1^2 Y_2 Z_2 - (a_2^2 - 3a_4)\, X_1^2 X_2^2 \\
& + (a_1 a_4 - a_2 a_3)(2X_1 Z_2 + X_2 Z_1)\, X_2 Y_1 \\
& + (a_1^2 a_4 - 2a_1 a_2 a_3 + 3a_3^2)\, X_1^2 X_2 Z_2 \\
& - (a_2 a_4 - 9a_6)\, X_1 X_2 (X_1 Z_2 + X_2 Z_1) \\
& + (3a_1 a_6 - a_3 a_4)(X_1 Z_2 + 2X_2 Z_1)\, Y_1 Z_2 \\
& + (3a_1^2 a_6 - 2a_1 a_3 a_4 + a_2 a_3^2 + 3a_2 a_6 - a_4^2)\, X_1 Z_2 (X_1 Z_2 + 2X_2 Z_1) \\
& - (3a_2 a_6 - a_4^2)(X_1 Z_2 + X_2 Z_1)(X_1 Z_2 - X_2 Z_1) \\
& + (a_1^3 a_6 - a_1^2 a_3 a_4 + a_1 a_2 a_3^2 - a_1 a_4^2 + 4a_1 a_2 a_6 - a_3^3 - 3a_3 a_6)\, Y_1 Z_1 Z_2^2 \\
& + (a_1^4 a_6 - a_1^3 a_3 a_4 + 5a_1^2 a_2 a_6 + a_1^2 a_2 a_3^2 - a_1 a_2 a_3 a_4 - a_1 a_3^3 - 3a_1 a_3 a_6 \\
& - a_1^2 a_4^2 + a_2^2 a_3^2 - a_2 a_4^2 + 4a_2^2 a_6 - a_3^2 a_4 - 3a_4 a_6)\, X_1 Z_1 Z_2^2 \\
& + (a_1^2 a_2 a_6 - a_1 a_2 a_3 a_4 + 3a_1 a_3 a_6 + a_2^2 a_3^2 - a_2 a_4^2 \\
& + 4a_2^2 a_6 - 2a_3^2 a_4 - 3a_4 a_6)\, X_2 Z_1^2 Z_2 \\
& + (a_1^3 a_3 a_6 - a_1^2 a_3^2 a_4 + a_1^2 a_4 a_6 + a_1 a_2 a_3^3 \\
& + 4a_1 a_2 a_3 a_6 - 2a_1 a_3 a_4^2 + a_2 a_3^2 a_4 \\
& + 4a_2 a_4 a_6 - a_3^4 - 6a_3^2 a_6 - a_4^3 - 9a_6^2)\, Z_1^2 Z_2^2,
\end{aligned}
$$

$$
\begin{aligned}
Z_3^{(2)} = {} & 3X_1 X_2 (X_1 Y_2 + X_2 Y_1) + Y_1 Y_2 (Y_1 Z_2 + Y_2 Z_1) + 3a_1 X_1^2 X_2^2 \\
& + a_1 (2X_1 Y_2 + Y_1 X_2)\, Y_1 Z_2 + a_1^2 X_1 Z_2 (2X_2 Y_1 + X_1 Y_2) \\
& + a_2 X_1 X_2 (Y_1 Z_2 + Y_2 Z_1) \\
& + a_2 (X_1 Y_2 + X_2 Y_1)(X_1 Z_2 + X_2 Z_1) \\
& + a_1^3 X_1^2 X_2 Z_2 + a_1 a_2 X_1 X_2 (2X_1 Z_2 + X_2 Z_1) \\
& + 3a_3 X_1 X_2^2 Z_1 + a_3 Y_1 Z_2 (Y_1 Z_2 + 2Y_2 Z_1) \\
& + 2a_1 a_3 X_1 Z_2 (Y_1 Z_2 + Y_2 Z_1) \\
& + 2a_1 a_3 X_2 Y_1 Z_1 Z_2 + a_4 (X_1 Y_2 + X_2 Y_1)\, Z_1 Z_2 \\
& + a_4 (X_1 Z_2 + X_2 Z_1)(Y_1 Z_2 + Y_2 Z_1) \\
& + (a_1^2 a_3 + a_1 a_4)\, X_1 Z_2 (X_1 Z_2 + 2X_2 Z_1) + a_2 a_3 X_2 Z_1 (2X_1 Z_2 + X_2 Z_1) \\
& + a_3^2 Y_1 Z_1 Z_2^2 + (a_3^2 + 3a_6)(Y_1 Z_2 + Y_2 Z_1)\, Z_1 Z_2 \\
& + a_1 a_3^2 (2X_1 Z_2 + X_2 Z_1)\, Z_1 Z_2 + 3a_1 a_6 X_1 Z_1 Z_2^2 \\
& + a_3 a_4 (X_1 Z_2 + 2X_2 Z_1)\, Z_1 Z_2 + (a_3^3 + 3a_3 a_6)\, Z_1^2 Z_2^2.
\end{aligned}
$$

1995 Bosma–Lenstra:
Explicit complete system
of 2 addition laws
for long Weierstrass curves:
explicit polynomials
$X_3, Y_3, Z_3, X_3', Y_3', Z_3'$
$\in \mathbf{Z}[a_1, a_2, a_3, a_4, a_6,$
$X_1, Y_1, Z_1, X_2, Y_2, Y_2]$.

1995 Bosma–Lenstra:
Explicit complete system
of 2 addition laws
for long Weierstrass curves:
explicit polynomials
$X_3, Y_3, Z_3, X_3', Y_3', Z_3'$
$\in \mathbf{Z}[a_1, a_2, a_3, a_4, a_6,$
$X_1, Y_1, Z_1, X_2, Y_2, Y_2]$.

My previous slide in this talk:
Bosma–Lenstra Y_3', Z_3'.
Not human-comprehensible.

1995 Bosma–Lenstra:
Explicit complete system
of 2 addition laws
for long Weierstrass curves:
explicit polynomials
$X_3, Y_3, Z_3, X_3', Y_3', Z_3'$
$\in \mathbf{Z}[a_1, a_2, a_3, a_4, a_6,$
$X_1, Y_1, Z_1, X_2, Y_2, Y_2]$.

My previous slide in this talk:
Bosma–Lenstra Y_3', Z_3'.
Not human-comprehensible.
Actually, slide shows
$\text{Publish}(Y_3'), \text{Publish}(Z_3')$,
where Publish introduces typos.

What this means:

For all fields k,
all $\mathbf{P}^2$ Weierstrass curves
$E/k : Y^2 Z + a_1 XYZ + a_3 YZ^2 =$
$X^3 + a_2 X^2 Z + a_4 XZ^2 + a_6 Z^3$,
all $P_1 = (X_1 : Y_1 : Z_1) \in E(k)$,
all $P_2 = (X_2 : Y_2 : Z_2) \in E(k)$:

$(X_3 : Y_3 : Z_3)$
is $P_1 + P_2$ or $(0 : 0 : 0)$;
$(X_3' : Y_3' : Z_3')$
is $P_1 + P_2$ or $(0 : 0 : 0)$;
at most one of these is $(0 : 0 : 0)$.

2009.11 Bernstein–T. Lange,
eprint.iacr.org/2009/580:

For all fields k with $2 \neq 0$,
all $\mathbf{P}^1 \times \mathbf{P}^1$ Edwards curves E/k:
$X^2T^2 + Y^2Z^2 = Z^2T^2 + dX^2Y^2$,
all $P_1, P_2 \in E(k)$,
$P_1 = ((X_1 : Z_1), (Y_1 : T_1))$,
$P_2 = ((X_2 : Z_2), (Y_2 : T_2))$:

$(X_3 : Z_3)$ is $x(P_1 + P_2)$ or $(0 : 0)$;
$(X_3' : Z_3')$ is $x(P_1 + P_2)$ or $(0 : 0)$;
$(Y_3 : T_3)$ is $y(P_1 + P_2)$ or $(0 : 0)$;
$(Y_3' : T_3')$ is $y(P_1 + P_2)$ or $(0 : 0)$;
at most one of these is $(0 : 0)$.

$$X_3 = X_1Y_2Z_2T_1 + X_2Y_1Z_1T_2,$$
$$Z_3 = Z_1Z_2T_1T_2 + dX_1X_2Y_1Y_2,$$
$$Y_3 = Y_1Y_2Z_1Z_2 - X_1X_2T_1T_2,$$
$$T_3 = Z_1Z_2T_1T_2 - dX_1X_2Y_1Y_2,$$

$$X_3' = X_1Y_1Z_2T_2 + X_2Y_2Z_1T_1,$$
$$Z_3' = X_1X_2T_1T_2 + Y_1Y_2Z_1Z_2,$$
$$Y_3' = X_1Y_1Z_2T_2 - X_2Y_2Z_1T_1,$$
$$T_3' = X_1Y_2Z_2T_1 - X_2Y_1Z_1T_2.$$

Much, much, much simpler than Lange–Ruppert, Bosma–Lenstra.
Also much easier to prove.
Also useful for computations.

Geometrically, all elliptic curves.
(Handle $2 = 0$ separately.)

5. Explicit Formulae

From [5, Chapter III, 2.3] it follows that $f = m^*(X/Z)$ and $g = m^*(Y/Z)$ are given by

$$f = \lambda^2 + a_1\lambda - \frac{X_1Z_2 + X_2Z_1}{Z_1Z_2} - a_2, \qquad g = -(\lambda + a_1)f - \nu - a_3,$$

where

$$\lambda = \frac{Y_1Z_2 - Y_2Z_1}{X_1Z_2 - X_2Z_1} \qquad \text{and} \qquad \nu = -\frac{Y_1X_2 - Y_2X_1}{X_1Z_2 - X_2Z_1}.$$

Applying the automorphism of $E \times E$ mapping (P_1, P_2) to $(P_1, -P_2)$ we find that

$$s^*(X/Z) = \kappa^2 + a_1\kappa - \frac{X_1Z_2 + X_2Z_1}{Z_1Z_2} - a_2$$

and

$$s^*(Y/Z) = -(\kappa + a_1)\, s^*(X/Z) - \mu - a_3,$$

where

$$\kappa = \frac{Y_1Z_2 + Y_2Z_1 + a_1X_2Z_1 + a_3Z_1Z_2}{X_1Z_2 - X_2Z_1}$$

and

$$\mu = -\frac{Y_1X_2 + Y_2X_1 + a_1X_1X_2 + a_3X_1Z_2}{X_1Z_2 - X_2Z_1}.$$

The bijection of Theorem 2 maps $(0:0:1)$ to the addition law given by $X_3^{(1)} = fZ_0$, $Y_3^{(1)} = gZ_0$, $Z_3^{(1)} = Z_0$, which in explicit terms is found to be given by

$$\begin{aligned}
X_3^{(1)} = {} & (X_1Y_2 - X_2Y_1)(Y_1Z_2 + Y_2Z_1) + (X_1Z_2 - X_2Z_1)\, Y_1Y_2 \\
& + a_1X_1X_2(Y_1Z_2 - Y_2Z_1) + a_1(X_1Y_2 - X_2Y_1)(X_1Z_2 + X_2Z_1) \\
& - a_2X_1X_2(X_1Z_2 - X_2Z_1) + a_3(X_1Y_2 - X_2Y_1)\, Z_1Z_2 \\
& + a_3(X_1Z_2 - X_2Z_1)(Y_1Z_2 + Y_2Z_1) \\
& - a_4(X_1Z_2 + X_2Z_1)(X_1Z_2 - X_2Z_1) \\
& - 3a_6(X_1Z_2 - X_2Z_1)\, Z_1Z_2,
\end{aligned}$$

$$\begin{aligned}
Y_3^{(1)} = &-3X_1X_2(X_1Y_2 - X_2Y_1)\\
&- Y_1Y_2(Y_1Z_2 - Y_2Z_1) - 2a_1(X_1Z_2 - X_2Z_1)\,Y_1Y_2\\
&+ (a_1^2 + 3a_2)\,X_1X_2(Y_1Z_2 - Y_2Z_1)\\
&- (a_1^2 + a_2)(X_1Y_2 + X_2Y_1)(X_1Z_2 - X_2Z_1)\\
&+ (a_1a_2 - 3a_3)\,X_1X_2(X_1Z_2 - X_2Z_1)\\
&- (2a_1a_3 + a_4)(X_1Y_2 - X_2Y_1)\,Z_1Z_2\\
&+ a_4(X_1Z_2 + X_2Z_1)(Y_1Z_2 - Y_2Z_1)\\
&+ (a_1a_4 - a_2a_3)(X_1Z_2 + X_2Z_1)(X_1Z_2 - X_2Z_1)\\
&+ (a_3^2 + 3a_6)(Y_1Z_2 - Y_2Z_1)\,Z_1Z_2\\
&+ (3a_1a_6 - a_3a_4)(X_1Z_2 - X_2Z_1)\,Z_1Z_2,\\
Z_3^{(1)} = {}&3X_1X_2(X_1Z_2 - X_2Z_1) - (Y_1Z_2 + Y_2Z_1)(Y_1Z_2 - Y_2Z_1)\\
&+ a_1(X_1Y_2 - X_2Y_1)\,Z_1Z_2 - a_1(X_1Z_2 - X_2Z_1)(Y_1Z_2 + Y_2Z_1)\\
&+ a_2(X_1Z_2 + X_2Z_1)(X_1Z_2 - X_2Z_1) - a_3(Y_1Z_2 - Y_2Z_1)\,Z_1Z_2\\
&+ a_4(X_1Z_2 - X_2Z_1)\,Z_1Z_2.
\end{aligned}$$

The corresponding exceptional divisor is $3 \cdot \Delta$, so a pair of points P_1, P_2 on E is exceptional for this addition law if and only if $P_1 = P_2$.

Multiplying the addition law just given by $s^*(Y/Z)$ we obtain the addition law corresponding to $(0:1:0)$. It reads as follows:

$$\begin{aligned}
X_3^{(2)} = {}&Y_1Y_2(X_1Y_2 + X_2Y_1) + a_1(2X_1Y_2 + X_2Y_1)\,X_2Y_1 + a_1^2X_1X_2^2Y_1\\
&- a_2X_1X_2(X_1Y_2 + X_2Y_1) - a_1a_2X_1^2X_2^2 + a_3X_2Y_1(Y_1Z_2 + 2Y_2Z_1)\\
&+ a_1a_3X_1X_2(Y_1Z_2 - Y_2Z_1) - a_1a_3(X_1Y_2 + X_2Y_1)(X_1Z_2 - X_2Z_1)\\
&- a_4X_1X_2(Y_1Z_2 + Y_2Z_1) - a_4(X_1Y_2 + X_2Y_1)(X_1Z_2 + X_2Z_1)\\
&- a_1^2a_3X_1^2X_2Z_2 - a_1a_4X_1X_2(2X_1Z_2 + X_2Z_1)\\
&- a_2a_3X_1X_2^2Z_1 - a_3^2X_1Z_2(2Y_2Z_1 + Y_1Z_2)\\
&- 3a_6(X_1Y_2 + X_2Y_1)\,Z_1Z_2\\
&- 3a_6(X_1Z_2 + X_2Z_1)(Y_1Z_2 + Y_2Z_1) - a_1a_3^2X_1Z_2(X_1Z_2 + 2X_2Z_1)\\
&- 3a_1a_6X_1Z_2(X_1Z_2 + 2X_2Z_1) + a_3a_4(X_1Z_2 - 2X_2Z_1)\,X_2Z_1\\
&- (a_1^2a_6 - a_1a_3a_4 + a_2a_3^2 + 4a_2a_6 - a_4^2)(Y_1Z_2 + Y_2Z_1)\,Z_1Z_2\\
&- (a_1^3a_6 - a_1^2a_3a_4 + a_1a_2a_3^2 + 4a_1a_2a_6 - a_1a_4^2)\,X_1Z_1Z_2^2\\
&- a_3^3(X_1Z_2 + X_2Z_1)\,Z_1Z_2 - 3a_3a_6(X_1Z_2 + 2X_2Z_1)\,Z_1Z_2\\
&- (a_1^2a_3a_6 - a_1a_3^2a_4 + a_2a_3^3 + 4a_2a_3a_6 - a_3a_4^2)\,Z_1^2Z_2^2,
\end{aligned}$$

$$
\begin{aligned}
Y_3^{(2)} = {} & Y_1^2 Y_2^2 + a_1 X_2 Y_1^2 Y_2 + (a_1 a_2 - 3a_3) X_1 X_2^2 Y_1 \\
& + a_3 Y_1^2 Y_2 Z_2 - (a_2^2 - 3a_4) X_1^2 X_2^2 \\
& + (a_1 a_4 - a_2 a_3)(2X_1 Z_2 + X_2 Z_1) X_2 Y_1 \\
& + (a_1^2 a_4 - 2a_1 a_2 a_3 + 3a_3^2) X_1^2 X_2 Z_2 \\
& - (a_2 a_4 - 9a_6) X_1 X_2 (X_1 Z_2 + X_2 Z_1) \\
& + (3a_1 a_6 - a_3 a_4)(X_1 Z_2 + 2X_2 Z_1) Y_1 Z_2 \\
& + (3a_1^2 a_6 - 2a_1 a_3 a_4 + a_2 a_3^2 + 3a_2 a_6 - a_4^2) X_1 Z_2 (X_1 Z_2 + 2X_2 Z_1) \\
& - (3a_2 a_6 - a_4^2)(X_1 Z_2 + X_2 Z_1)(X_1 Z_2 - X_2 Z_1) \\
& + (a_1^3 a_6 - a_1^2 a_3 a_4 + a_1 a_2 a_3^2 - a_1 a_4^2 + 4a_1 a_2 a_6 - a_3^3 - 3a_3 a_6) Y_1 Z_1 Z_2^2 \\
& + (a_1^4 a_6 - a_1^3 a_3 a_4 + 5a_1^2 a_2 a_6 + a_1^2 a_2 a_3^2 - a_1 a_2 a_3 a_4 - a_1 a_3^3 - 3a_1 a_3 a_6 \\
& - a_1^2 a_4^2 + a_2^2 a_3^2 - a_2 a_4^2 + 4a_2^2 a_6 - a_3^2 a_4 - 3a_4 a_6) X_1 Z_1 Z_2^2 \\
& + (a_1^2 a_2 a_6 - a_1 a_2 a_3 a_4 + 3a_1 a_3 a_6 + a_2^2 a_3^2 - a_2 a_4^2 \\
& + 4a_2^2 a_6 - 2a_3^2 a_4 - 3a_4 a_6) X_2 Z_1^2 Z_2 \\
& + (a_1^3 a_3 a_6 - a_1^2 a_3^2 a_4 + a_1^2 a_4 a_6 + a_1 a_2 a_3^3 \\
& + 4a_1 a_2 a_3 a_6 - 2a_1 a_3 a_4^2 + a_2 a_3^2 a_4 \\
& + 4a_2 a_4 a_6 - a_3^4 - 6a_3^2 a_6 - a_4^3 - 9a_6^2) Z_1^2 Z_2^2, \\
Z_3^{(2)} = {} & 3X_1 X_2 (X_1 Y_2 + X_2 Y_1) + Y_1 Y_2 (Y_1 Z_2 + Y_2 Z_1) + 3a_1 X_1^2 X_2^2 \\
& + a_1 (2X_1 Y_2 + Y_1 X_2) Y_1 Z_2 + a_1^2 X_1 Z_2 (2X_2 Y_1 + X_1 Y_2) \\
& + a_2 X_1 X_2 (Y_1 Z_2 + Y_2 Z_1) \\
& + a_2 (X_1 Y_2 + X_2 Y_1)(X_1 Z_2 + X_2 Z_1) \\
& + a_1^3 X_1^2 X_2 Z_2 + a_1 a_2 X_1 X_2 (2X_1 Z_2 + X_2 Z_1) \\
& + 3a_3 X_1 X_2^2 Z_1 + a_3 Y_1 Z_2 (Y_1 Z_2 + 2Y_2 Z_1) \\
& + 2a_1 a_3 X_1 Z_2 (Y_1 Z_2 + Y_2 Z_1) \\
& + 2a_1 a_3 X_2 Y_1 Z_1 Z_2 + a_4 (X_1 Y_2 + X_2 Y_1) Z_1 Z_2 \\
& + a_4 (X_1 Z_2 + X_2 Z_1)(Y_1 Z_2 + Y_2 Z_1) \\
& + (a_1^2 a_3 + a_1 a_4) X_1 Z_2 (X_1 Z_2 + 2X_2 Z_1) + a_2 a_3 X_2 Z_1 (2X_1 Z_2 + X_2 Z_1) \\
& + a_3^2 Y_1 Z_1 Z_2^2 + (a_3^2 + 3a_6)(Y_1 Z_2 + Y_2 Z_1) Z_1 Z_2 \\
& + a_1 a_3^2 (2X_1 Z_2 + X_2 Z_1) Z_1 Z_2 + 3a_1 a_6 X_1 Z_1 Z_2^2 \\
& + a_3 a_4 (X_1 Z_2 + 2X_2 Z_1) Z_1 Z_2 + (a_3^3 + 3a_3 a_6) Z_1^2 Z_2^2.
\end{aligned}
$$

History of these addition laws:

1761 Euler, 1866 Gauss:
Beautiful addition law
for $x^2 + y^2 = 1 - x^2y^2$,
the "lemniscatic elliptic curve."

$(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$ with

$$x_3 = \frac{x_1y_2 + x_2y_1}{1 - x_1x_2y_1y_2},$$

$$y_3 = \frac{y_1y_2 - x_1x_2}{1 + x_1x_2y_1y_2}.$$

1986 Chudnovsky–Chudnovsky
factorization-speed study begins
with $\mathbf{G}_\mathrm{a}$, $\mathbf{G}_\mathrm{m}$, $\mathbf{T}_2$, lemniscate;
but focuses on curve *families*.

2007 Edwards:
Obtain all elliptic curves over $\overline{\mathbf{Q}}$ by generalizing to curve $x^2 + y^2 = 1 + dx^2y^2$.

$(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$ with

$$x_3 = \frac{x_1y_2 + x_2y_1}{1 + dx_1x_2y_1y_2},$$

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

Edwards actually used $d = c^4$.
Scaling: $x^2 + y^2 = c^2(1 + x^2y^2)$.
But $x^2 + y^2 = 1 + dx^2y^2$ lowers j degree; includes lemniscate; simplifies degeneration to clock.

Embed E into $\mathbf{P}^1 \times \mathbf{P}^1$,
as recommended by Edwards.

$$(\infty, \frac{\pm 1}{\sqrt{d}}), (\frac{\pm 1}{\sqrt{d}}, \infty) \in E(k(\sqrt{d})).$$

Edwards commented that
the addition law works for

$$(x_1, y_1) + (\frac{1}{\sqrt{d}}, \infty) = (\frac{1}{y_1\sqrt{d}}, \frac{-1}{x_1\sqrt{d}}).$$

Can easily use this to obtain
a dual addition law:

$$x_3 = \frac{x_1 y_1 + x_2 y_2}{x_1 x_2 + y_1 y_2},$$

$$y_3 = \frac{x_1 y_1 - x_2 y_2}{x_1 y_2 - x_2 y_1}.$$

Here's how: $(x_1, y_1) + (x_2, y_2)$

$$= (x_1, y_1) + \left(\frac{1}{\sqrt{d}}, \infty\right) \\ \qquad + (x_2, y_2) - \left(\frac{1}{\sqrt{d}}, \infty\right)$$

Here's how: $(x_1, y_1) + (x_2, y_2)$

$$= (x_1, y_1) + \left(\frac{1}{\sqrt{d}}, \infty\right)$$
$$+ (x_2, y_2) - \left(\frac{1}{\sqrt{d}}, \infty\right)$$
$$= \left(\frac{1}{y_1\sqrt{d}}, \frac{-1}{x_1\sqrt{d}}\right) + (x_2, y_2) - \left(\frac{1}{\sqrt{d}}, \infty\right)$$

Here's how: $(x_1, y_1) + (x_2, y_2)$

$$= (x_1, y_1) + (\frac{1}{\sqrt{d}}, \infty)$$

$$+ (x_2, y_2) - (\frac{1}{\sqrt{d}}, \infty)$$

$$= (\frac{1}{y_1\sqrt{d}}, \frac{-1}{x_1\sqrt{d}}) + (x_2, y_2) - (\frac{1}{\sqrt{d}}, \infty)$$

$$= \left(\frac{\frac{y_2}{y_1\sqrt{d}} - \frac{x_2}{x_1\sqrt{d}}}{1 - \frac{dx_2y_2}{dx_1y_1}}, \frac{\frac{-y_2}{x_1\sqrt{d}} - \frac{x_2}{y_1\sqrt{d}}}{1 + \frac{dx_2y_2}{dx_1y_1}} \right)$$

$$- (\frac{1}{\sqrt{d}}, \infty)$$

Here's how: $(x_1, y_1) + (x_2, y_2)$

$$= (x_1, y_1) + \left(\frac{1}{\sqrt{d}}, \infty\right)$$
$$\quad + (x_2, y_2) - \left(\frac{1}{\sqrt{d}}, \infty\right)$$
$$= \left(\frac{1}{y_1\sqrt{d}}, \frac{-1}{x_1\sqrt{d}}\right) + (x_2, y_2) - \left(\frac{1}{\sqrt{d}}, \infty\right)$$
$$= \left(\frac{\frac{y_2}{y_1\sqrt{d}} - \frac{x_2}{x_1\sqrt{d}}}{1 - \frac{dx_2y_2}{dx_1y_1}}, \frac{\frac{-y_2}{x_1\sqrt{d}} - \frac{x_2}{y_1\sqrt{d}}}{1 + \frac{dx_2y_2}{dx_1y_1}}\right)$$
$$\quad - \left(\frac{1}{\sqrt{d}}, \infty\right)$$

$$= \left(\frac{\frac{x_1y_2 - x_2y_1}{\sqrt{d}}}{x_1y_1 - x_2y_2}, \frac{\frac{-y_1y_2 - x_1x_2}{\sqrt{d}}}{x_1y_1 + x_2y_2}\right)$$
$$\quad - \left(\frac{1}{\sqrt{d}}, \infty\right)$$

Here's how: $(x_1, y_1) + (x_2, y_2)$

$$= (x_1, y_1) + (\tfrac{1}{\sqrt{d}}, \infty)$$
$$\quad + (x_2, y_2) - (\tfrac{1}{\sqrt{d}}, \infty)$$
$$= (\tfrac{1}{y_1\sqrt{d}}, \tfrac{-1}{x_1\sqrt{d}}) + (x_2, y_2) - (\tfrac{1}{\sqrt{d}}, \infty)$$
$$= \left(\frac{\frac{y_2}{y_1\sqrt{d}} - \frac{x_2}{x_1\sqrt{d}}}{1 - \frac{dx_2y_2}{dx_1y_1}}, \frac{\frac{-y_2}{x_1\sqrt{d}} - \frac{x_2}{y_1\sqrt{d}}}{1 + \frac{dx_2y_2}{dx_1y_1}} \right)$$
$$\quad - (\tfrac{1}{\sqrt{d}}, \infty)$$

$$= \left(\frac{\frac{x_1y_2 - x_2y_1}{\sqrt{d}}}{x_1y_1 - x_2y_2}, \frac{\frac{-y_1y_2 - x_1x_2}{\sqrt{d}}}{x_1y_1 + x_2y_2} \right)$$
$$\quad - (\tfrac{1}{\sqrt{d}}, \infty)$$

$$= (\tfrac{x_1y_1 + x_2y_2}{x_1x_2 + y_1y_2}, \tfrac{x_1y_1 - x_2y_2}{x_1y_2 - x_2y_1}).$$

2007 Bernstein–Lange:
Edwards addition law gives speed records for ECM, ECC, etc.

2008 Hisil–Wong–Carter–Dawson:
First publication of dual addition law; new speed records.
(Completely different derivation.)

2009.11 Bernstein–Lange:
Addition law and dual form a complete system.

Elementary, computational proof, giving elementary, computational *definition* of the group $E(k)$ using these formulas.

1987 Lenstra "Elliptic curves and number-theoretic algorithms":

Use Lange–Ruppert complete system of addition laws to give computational definition of the Weierstrass group $E(R)$ for more general rings R.

Define $\mathbf{P}^2(R) = \{(X : Y : Z) : X, Y, Z \in R;\ XR + YR + ZR = R\}$ where $(X : Y : Z)$ is the module $\{(\lambda X, \lambda Y, \lambda Z) : \lambda \in R\}$.

Define $E(R) = \{(X : Y : Z) \in \mathbf{P}^2(R) : Y^2 Z = X^3 + a_4 X Z^2 + a_6 Z^3\}$.

To define (and compute) sum
$(X_1 : Y_1 : Z_1) + (X_2 : Y_2 : Z_2)$:

Consider (and compute)
Lange–Ruppert $(X_3 : Y_3 : Z_3)$,
$(X_3' : Y_3' : Z_3')$, $(X_3'' : Y_3'' : Z_3'')$.

Add these R-modules:

$$\begin{aligned}\{ \quad & (\lambda X_3, \lambda Y_3, \lambda Z_3) \\ & + (\lambda' X_3', \lambda' Y_3', \lambda' Z_3') \\ & + (\lambda'' X_3'', \lambda'' Y_3'', \lambda'' Z_3'') : \\ & \qquad\qquad \lambda, \lambda', \lambda'' \in R\}.\end{aligned}$$

Allow any ring R
having trivial class group.
Then this sum of modules can be
expressed as $(X : Y : Z)$.

Counting points: Schoof etc.

Input: prime ℓ; abelian variety $A/\mathbf{F}_q$, usually Jac(genus-g curve).

Write down generic point $P \in A$ with $\ell P = 0$.

Specifically: express $\ell P = 0$ as system of equations on coordinates of P; extend $\mathbf{F}_q$ to ring $R = \mathbf{F}_q[\text{coords}]/\text{equations}$; note that $\ell P = 0$ in $A(R)$.

Genus 1: $\#R \approx q^{\ell^2}$.

Genus 2: $\#R \approx q^{\ell^4}$. Much larger computations.

True often enough to be useful:

Genus 1: Unique linear equation
$\varphi^2(P) - s_1\varphi(P) + qP = 0$ for
qth-power $\varphi : A(R) \to A(R)$
with $s_1 \in \{0, 1, \ldots, \ell - 1\}$.
Then $1 - s_1 + q - \#A(\mathbf{F}_q) \in \ell\mathbf{Z}$.

Genus 2: Unique linear equation
$$\varphi^4(P) - s_1\varphi^3(P) + s_2\varphi^2(P) - qs_1\varphi(P) + q^2P = 0$$
with $s_1, s_2 \in \{0, 1, \ldots, \ell - 1\}$.
Then $1 - s_1 + s_2 - qs_1 + q^2 - \#A(\mathbf{F}_q) \in \ell\mathbf{Z}$.

Try many ℓ; deduce $\#A(\mathbf{F}_q)$.
Silly name: "ℓ-adic method."

Which coords to choose for $A = \text{Jac}(C)$ when C has genus 2?

2000 Gaudry–Harley,
2004 Gaudry–Schost,
2009 Gaudry–Schost:

Use Mumford coordinates for A, and write $P = P_1 - P_2$ with $P_i = (x_i, y_i) \in C \to A$.

$$R = \mathbf{F}_q[x_1, y_1, x_2, y_2]/($$
$$(x_1, y_1) \in C;$$
$$(x_2, y_2) \in C;$$
$$\ell(x_1, y_1) = \ell(x_2, y_2)$$
$$).$$

$\ell(x_1, y_1) = \ell(x_2, y_2)$
gives two equations in x_1, x_2
of degree $\ell^{2+o(1)}$. Eliminate x_2,
obtaining equation in x_1.

Elimination time $(\ell^6 \lg q)^{1+o(1)}$
using fast-arithmetic techniques.

Equation in x_1: degree $\ell^{4+o(1)}$.
Computing $\varphi(P)$ etc.:
time $(\ell^4 (\lg q)^2)^{1+o(1)}$.

Total time $(\lg q)^{8+o(1)}$
to handle all $\ell \le (\lg q)^{1+o(1)}$.

2004 Gaudry–Schost: symmetrize;
constant-factor speedup.

2000 Gaudry–Harley et al.
don't actually use $A(R)$.
They map R to a field,
allegedly saving time.

2000 Gaudry–Harley et al. don't actually use $A(R)$. They map R to a field, allegedly saving time.

But factorization is slow!

Latest factorization algorithm, 2008 Kedlaya–Umans, takes time $(\lg q)^{7+o(1)}$ to factor the x_1 equation. Sum over ℓ: $(\lg q)^{8+o(1)}$.

Closer analysis of $o(1)$ shows that factorization still loses time here, except for "free" factors.

Can save time in genus 1
by building a smaller R
that defines a φ-stable
subgroup of ℓ-torsion.
(1991 Elkies; 1992 Atkin)

Fastest such techniques reported
for genus 2: time $\ell^{12+o(1)}$.

Use for $\ell \le (\lg q)^{2/3+o(1)}$.
Asymptotic speedup $1 + o(1)$.

Also "kangaroos" / "cockroaches":
Asymptotic speedup $1 + o(1)$.

Also $\#A \bmod 2^2$ etc.:
Asymptotic speedup $1 + o(1)$.

Video games

Millions of people buy PCs
to "play video games":
i.e., to participate in
applied physics simulations,
often highly networked.

Society adjusts ultracomputer
to improve these simulations.

Algorithm designers
obtain much better results
by paying attention to the
ultracomputer architecture!
Most important fact:
$\#\text{ALUs} \in \Theta(\#\text{bits of RAM})$.

My university has just spent $<$\$20000 on a cluster for me.

8 CPUs; 15 GTX 295 cards.
Each GTX 295 has 2 “GPUs.”

My university has just spent $<$\$20000 on a cluster for me.

8 CPUs; 15 GTX 295 cards.
Each GTX 295 has 2 “GPUs.”
Each GPU has 30 cores.

My university has just spent <$20000 on a cluster for me.

8 CPUs; 15 GTX 295 cards.
Each GTX 295 has 2 “GPUs.”
Each GPU has 30 cores.
Each core has 8 ALUs
and <100KB of fast RAM.
Each ALU performs a 32-bit operation every cycle @1.242GHz.

My university has just spent
<$20000 on a cluster for me.

8 CPUs; 15 GTX 295 cards.
Each GTX 295 has 2 “GPUs.”
Each GPU has 30 cores.
Each core has 8 ALUs
and <100KB of fast RAM.
Each ALU performs a 32-bit
operation every cycle @1.242GHz.

I also have accounts
on several “TeraGrid” clusters.
Right now I’m using 448 GPUs;
13440 cores; 107520 32-bit ALUs.

GPU cores can communicate through slow “global” RAM. $\leq$3 bits per ALU per cycle.

GPU cores can communicate through slow "global" RAM. $\leq$3 bits per ALU per cycle.

Cluster nodes can communicate through a slow network. $\leq$0.003 bits per ALU per cycle.

GPU cores can communicate through slow “global” RAM. $\leq$3 bits per ALU per cycle.

Cluster nodes can communicate through a slow network. $\leq$0.003 bits per ALU per cycle.

Algorithm-analysis students are taught to count algorithm “operations.” RAM access: 1 operation.

Resulting algorithms are poorly optimized for the real world. Gap grows with cluster size.

Much better model
developed 30 years ago:
Computation is carried out
on a 2-dimensional circuit.
Measure circuit area, time.

e.g. 1981 Brent–Kung:
multiply n-bit integers
in time $n^{0.5+o(1)}$
using circuit area $n^{1+o(1)}$.

Scalability in this model
is fairly close to scalability
of real-world computations.

Many other "buildable" models.

Time to sort n small integers
on machine of size $n^{1+o(1)}$:

$n^{2.0+o(1)}$: 1-tape Turing machine.
$n^{1.5+o(1)}$: 2-dimensional RAM.
$n^{1.0+o(1)}$: pipelined RAM.
$n^{0.5+o(1)}$: 2-dimensional circuit.

Why does anyone say that
sorting time is $n^{1+o(1)}$?
Why choose third machine?
Silly! Once n is large enough,
fourth machine is better.

Let's see what this means for genus-2 point-counting.

Machine cost: $(\lg q)^{5+o(1)}$.
$(\lg q)^{5+o(1)}$ ALUs.

Let's see what this means
for genus-2 point-counting.

Machine cost: $(\lg q)^{5+o(1)}$.
$(\lg q)^{5+o(1)}$ ALUs.

Multiplying two univariate
polynomials of degree $(\lg q)^{2+o(1)}$:
$(\lg q)^{3+o(1)}$ ALUs;
time $(\lg q)^{1.5+o(1)}$.

Let's see what this means
for genus-2 point-counting.

Machine cost: $(\lg q)^{5+o(1)}$.
$(\lg q)^{5+o(1)}$ ALUs.

Multiplying two univariate
polynomials of degree $(\lg q)^{2+o(1)}$:
$(\lg q)^{3+o(1)}$ ALUs;
time $(\lg q)^{1.5+o(1)}$.

$(\lg q)^{4+o(1)}$ resultants:
$(\lg q)^{5+o(1)}$ ALUs;
time $(\lg q)^{3.5+o(1)}$.

Multiplying mod x_1 equation:
$(\lg q)^{5+o(1)}$ ALUs;
time $(\lg q)^{2.5+o(1)}$.

Multiplying mod x_1 equation:
$(\lg q)^{5+o(1)}$ ALUs;
time $(\lg q)^{2.5+o(1)}$.

Computing $\varphi(P)$ etc.:
$(\lg q)^{5+o(1)}$ ALUs;
time $(\lg q)^{3.5+o(1)}$.

Total time $(\lg q)^{4.5+o(1)}$.

Multiplying mod x_1 equation:
$(\lg q)^{5+o(1)}$ ALUs;
time $(\lg q)^{2.5+o(1)}$.

Computing $\varphi(P)$ etc.:
$(\lg q)^{5+o(1)}$ ALUs;
time $(\lg q)^{3.5+o(1)}$.

Total time $(\lg q)^{4.5+o(1)}$.

In oversimplified RAM model,
$\lg q$ exponent was dominated
solely by the resultants.
No longer true here.

Most important computation:
big multiplication on GPUs,
and then on network of GPUs.

First steps: 2009 Emeliyanenko,
“Efficient multiplication of
polynomials on graphics
hardware.” Uses algorithm ideas
developed for FFT on tape,
π computation on disk, etc.

Recent tool development: 2010
Bernstein–Chen–Cheng–Lange–
Niederhagen–Schwabe–Yang,
“Usable assembly language for
GPUs: a success story.”