Twisted Hessian curves

cr.yp.to/papers.html#hessian

Daniel J. Bernstein
University of Illinois at Chicago &
Technische Universiteit Eindhoven

Joint work with:

Chitchanok Chuengsatiansup
Technische Universiteit Eindhoven

David Kohel
Aix-Marseille Université

Tanja Lange
Technische Universiteit Eindhoven

1986 Chudnovsky–Chudnovsky,
“Sequences of numbers
generated by addition
in formal groups
and new primality
and factorization tests”:

“The crucial problem becomes
the choice of the model
of an algebraic group variety,
where computations mod $p$
are the least time consuming.”

Most important computations:
ADD is $P, Q \mapsto P + Q$.
DBL is $P \mapsto 2P$. 
Hessian curves

Twisted Hessian curves

1986 Chudnovsky–Chudnovsky,
“Sequences of numbers
generated by addition
in formal groups
and new primality
and factorization tests”:

“The crucial problem becomes
the choice of the model
of an algebraic group variety,
where computations mod $p$
are the least time consuming.”

Most important computations:
ADD is $P, Q \mapsto P + Q$.
DBL is $P \mapsto 2P$.

“It is preferable to use
models of elliptic curves
lying in low-dimensional spaces,
for otherwise the number
of coordinates and operations is
increasing. This limits us
::: to
4 basic models of elliptic curves.

Short Weierstrass:
$y^2 = x^3 + ax + b$.

Jacobi intersection:
$s^2 + c^2 = 1$, as
$2^2 + d^2 = 1$.

Jacobi quartic:
$y^2 = x^4 + 2ax^2 + 1$.

Hessian:
$x^3 + y^3 + 1 = 3dxy$.
1986 Chudnovsky–Chudnovsky, “Sequences of numbers generated by addition in formal groups and new primality and factorization tests”:

“The crucial problem becomes the choice of the model of an algebraic group variety, where computations mod \(p\) are the least time consuming.”

Most important computations: ADD is \(P, Q \mapsto P + Q\). DBL is \(P \mapsto 2P\).

“It is preferable to use models of elliptic curves lying in low-dimensional spaces, for otherwise the number of coordinates and operations is increasing. This limits us to 4 basic models of elliptic curves:

- Short Weierstrass: \(y^2 = x^3 + ax + b\).
- Jacobi intersection: \(s^2 + c^2 = 1, as^2 + d^2 = 1\).
- Jacobi quartic: \(y^2 = x^4 + 2ax^2 + 1\).
- Hessian: \(x^3 + y^3 + 1 = 3dxy\).
1986 Chudnovsky–Chudnovsky, “Sequences of numbers generated by addition in formal groups and new primality and factorization tests”:

“The crucial problem becomes the choice of the model of an algebraic group variety, where computations mod $p$ are the least time consuming.”

Most important computations: ADD is $P, Q \mapsto P + Q$. DBL is $P \mapsto 2P$.

“It is preferable to use models of elliptic curves lying in low-dimensional spaces, for otherwise the number of coordinates and operations is increasing. This limits us to 4 basic models of elliptic curves:

- Short Weierstrass: $y^2 = x^3 + ax + b$.
- Jacobi intersection: $s^2 + c^2 = 1, as^2 + d^2 = 1$.
- Jacobi quartic: $y^2 = x^4 + 2ax^2 + 1$.
- Hessian: $x^3 + y^3 + 1 = 3dx$. 
1986 Chudnovsky–Chudnovsky, “Sequences of numbers generated by addition in formal groups and new primality and factorization tests”:

“The crucial problem becomes the choice of the model of an algebraic group variety, where computations mod $p$ are the least time consuming.”

Most important computations:
ADD is $P, Q \mapsto P + Q$.
DBL is $P \mapsto 2P$.

“It is preferable to use models of elliptic curves lying in low-dimensional spaces, for otherwise the number of coordinates and operations is increasing. This limits us . . . to 4 basic models of elliptic curves.”

Short Weierstrass:
$y^2 = x^3 + ax + b$.

Jacobi intersection:
$s^2 + c^2 = 1, \ as^2 + d^2 = 1$.

Jacobi quartic: $y^2 = x^4 + 2ax^2 + 1$.

Hessian: $x^3 + y^3 + 1 = 3dxy$. 
It is preferable to use models of elliptic curves lying in low-dimensional spaces, for otherwise the number of coordinates and operations is increasing. This limits us ... to 4 basic models of elliptic curves.”

Short Weierstrass:
\[ y^2 = x^3 + ax + b. \]

Jacobi intersection:
\[ s^2 + c^2 = 1, \quad as^2 + d^2 = 1. \]

Jacobi quartic: \[ y^2 = x^4 + 2ax^2 + 1. \]

Hessian: \[ x^3 + y^3 + 1 = 3dxy. \]
1986 Chudnovsky–Chudnovsky, "Sequences of numbers generated by addition in formal groups and new primality and factorization tests": "The crucial problem becomes the choice of the model of an algebraic group variety, where computations mod $p$ are the least time consuming."

Most important computations:

- **ADD**: $P; Q \mapsto P + Q$.
- **DBL**: $P \mapsto 2P$.

"It is preferable to use models of elliptic curves lying in low-dimensional spaces, for otherwise the number of coordinates and operations is increasing. This limits us ... to 4 basic models of elliptic curves."

- **Short Weierstrass**: $y^2 = x^3 + ax + b$.
- **Jacobi intersection**: $s^2 + c^2 = 1, as^2 + d^2 = 1$.
- **Jacobi quartic**: $y^2 = x^4 + 2ax^2 + 1$.
- **Hessian**: $x^3 + y^3 + 1 = 3dxy$.

Our experience shows that the expression of the law of addition on the cubic Hessian form (d) of an elliptic curve is by far the best and prettiest.

$X_3 = Y_1 X_2 \cdot Y_1 Z_2$
$Y_3 = X_1 Z_2 \cdot X_1 Y_2$
$Z_3 = Z_1 Y_2 \cdot Z_1 X_2$

12M for ADD, where M is the cost of multiplication in the field.

8.4M for DBL, assuming 0.8M for the cost of squaring in the field.


“It is preferable to use models of elliptic curves lying in low-dimensional spaces, for otherwise the number of coordinates and operations is increasing. This limits us... to 4 basic models of elliptic curves.”

Short Weierstrass:
\[ y^2 = x^3 + ax + b. \]

Jacobi intersection:
\[ s^2 + c^2 = 1, \quad as^2 + d^2 = 1. \]

Jacobi quartic: \[ y^2 = x^4 + 2ax^2 + 1. \]

Hessian: \[ x^3 + y^3 + 1 = 3dxy. \]

“Our experience shows that the expression of the law of addition on the cubic Hessian form (d) of an elliptic curve is by far the best and the prettiest.

\[ X_3 = Y_1X_2 \cdot Y_1Z_2 - Z_1Y_2 \cdot Z_3 = Z_1Y_2 \cdot Z_1X_2 - X_1Z_2. \]

12\(M\) for ADD, where \(M\) is the cost of multiplication in the field.

8.4\(M\) for DBL, assuming 0.8\(M\) for the cost of squaring in the field.
“It is preferable to use models of elliptic curves lying in low-dimensional spaces, for otherwise the number of coordinates and operations is increasing. This limits us ... to 4 basic models of elliptic curves.”

Short Weierstrass:
\[ y^2 = x^3 + ax + b. \]

Jacobi intersection:
\[ s^2 + c^2 = 1, \quad as^2 + d^2 = 1. \]

Jacobi quartic: \[ y^2 = x^4 + 2ax^2 + 1. \]

Hessian: \[ x^3 + y^3 + 1 = 3dxy. \]

“Our experience shows that the expression of the law of addition on the cubic Hessian form (d) of an elliptic curve is by far the best and the prettiest.”

\[ X_3 = Y_1X_2 \cdot Y_1Z_2 - Z_1Y_2 \cdot X_1Y_2, \]
\[ Y_3 = X_1Z_2 \cdot X_1Y_2 - Y_1X_2 \cdot Z_1X_2, \]
\[ Z_3 = Z_1Y_2 \cdot Z_1X_2 - X_1Z_2 \cdot Y_1Z_2. \]

12\text{M} for ADD, where \text{M} is the cost of multiplication in the field.

8.4\text{M} for DBL, assuming 0.8\text{M} for the cost of squaring in the field.
It is preferable to use models of elliptic curves lying in low-dimensional spaces, otherwise the number of coordinates and operations is increasing. This limits us to four basic models of elliptic curves.

**Short Weierstrass:**
\[ y^2 = x^3 + ax + b. \]

**Jacobi intersection:**
\[ s^2 + c^2 = 1, \quad as^2 + d^2 = 1. \]

**Jacobi quartic:**
\[ y^2 = x^4 + 2ax^2 + 1. \]

**Hessian:**
\[ x^3 + y^3 + 1 = 3dxy. \]

“Our experience shows that the expression of the law of addition on the cubic Hessian form (d) of an elliptic curve is by far the best and the prettiest.”

\[ X_3 = Y_1X_2 \cdot Y_1Z_2 - Z_1Y_2 \cdot X_1Y_2, \]
\[ Y_3 = X_1Z_2 \cdot X_1Y_2 - Y_1X_2 \cdot Z_1X_2, \]
\[ Z_3 = Z_1Y_2 \cdot Z_1X_2 - X_1Z_2 \cdot Y_1Z_2. \]

12\(M\) for ADD,
where \(M\) is the cost of multiplication in the field.

8.4\(M\) for DBL,
assuming 0.8\(M\) for the cost of squaring in the field.

1990s: ECC standards instead use short Weierstrass curves in Jacobian coordinates for “the fastest arithmetic.”

15.2\(M\) for ADD,
much slower than Hessian.

Why is this a good idea?
It is preferable to use models of elliptic curves lying in low-dimensional spaces, for otherwise the number of coordinates and operations is increasing. This limits us to the basic models of elliptic curves.

- Short Weierstrass: \( y^2 = x^3 + ax + b \).
- Jacobi intersection: \( s^2 + c^2 = 1 \), as \( 2s^2 + d^2 = 1 \).
- Jacobi quartic: \( y^2 = x^4 + 2ax^2 + 1 \).
- Hessian: \( x^3 + y^3 + 1 = 3dxy \).

"Our experience shows that the expression of the law of addition on the cubic Hessian form (d) of an elliptic curve is by far the best and the prettiest."

\[
\begin{align*}
X_3 &= Y_1X_2 \cdot Y_1Z_2 - Z_1Y_2 \cdot X_1Y_2, \\
Y_3 &= X_1Z_2 \cdot X_1Y_2 - Y_1X_2 \cdot Z_1X_2, \\
Z_3 &= Z_1Y_2 \cdot Z_1X_2 - X_1Z_2 \cdot Y_1Z_2. \\
\end{align*}
\]

12\( M \) for ADD, where \( M \) is the cost of multiplication in the field.

8.4\( M \) for DBL, assuming 0.8\( M \) for the cost of squaring in the field.

1990s: ECC standards instead use short Weierstrass curves in Jacobian coordinates for "the fastest arithmetic." 15.2\( M \) for ADD, much slower than Hessian. Why is this a good idea?
It is preferable to use models of elliptic curves lying in low-dimensional spaces, for otherwise the number of coordinates and operations is increasing. This limits us to 4 basic models of elliptic curves.

- Short Weierstrass: \(y^2 = x^3 + ax + b\).
- Jacobi intersection: \(s^2 + c^2 = 1\), as \(2 + d^2 = 1\).
- Jacobi quartic: \(y^2 = x^4 + 2ax^2 + 1\).
- Hessian: \(x^3 + y^3 + 1 = 3dxy\).

Our experience shows that the expression of the law of addition on the cubic Hessian form (d) of an elliptic curve is by far the best and the prettiest.

\[
\begin{align*}
X_3 &= Y_1X_2 \cdot Y_1Z_2 - Z_1Y_2 \cdot X_1Y_2, \\
Y_3 &= X_1Z_2 \cdot X_1Y_2 - Y_1X_2 \cdot Z_1X_2, \\
Z_3 &= Z_1Y_2 \cdot Z_1X_2 - X_1Z_2 \cdot Y_1Z_2.
\end{align*}
\]

12\(\mathbf{M}\) for ADD, where \(\mathbf{M}\) is the cost of multiplication in the field.

1990s: ECC standards instead use short Weierstrass curves in Jacobian coordinates for “the fastest arithmetic”.

15.2\(\mathbf{M}\) for ADD, much slower than Hessian.

Why is this a good idea?
“Our experience shows that the expression of the law of addition on the cubic Hessian form (d) of an elliptic curve is by far the best and the prettiest.”

$$X_3 = Y_1 X_2 \cdot Y_1 Z_2 - Z_1 Y_2 \cdot X_1 Y_2,$$

$$Y_3 = X_1 Z_2 \cdot X_1 Y_2 - Y_1 X_2 \cdot Z_1 X_2,$$

$$Z_3 = Z_1 Y_2 \cdot Z_1 X_2 - X_1 Z_2 \cdot Y_1 Z_2.$$ 

12M for ADD, where M is the cost of multiplication in the field.

8.4M for DBL, assuming 0.8M for the cost of squaring in the field.

1990s: ECC standards instead use short Weierstrass curves in Jacobian coordinates for “the fastest arithmetic”.

15.2M for ADD, much slower than Hessian.

Why is this a good idea?
“Our experience shows that the expression of the law of addition on the cubic Hessian form (d) of an elliptic curve is by far the best and the prettiest.”

\[
\begin{align*}
X_3 &= Y_1 X_2 \cdot Y_1 Z_2 - Z_1 Y_2 \cdot X_1 Y_2, \\
Y_3 &= X_1 Z_2 \cdot X_1 Y_2 - Y_1 X_2 \cdot Z_1 X_2, \\
Z_3 &= Z_1 Y_2 \cdot Z_1 X_2 - X_1 Z_2 \cdot Y_1 Z_2.
\end{align*}
\]

12M for ADD, where M is the cost of multiplication in the field.

8.4M for DBL, assuming 0.8M for the cost of squaring in the field.

1990s: ECC standards instead use short Weierstrass curves in Jacobian coordinates for “the fastest arithmetic”.

15.2M for ADD, much slower than Hessian.

Why is this a good idea?
Answer: Only 7.2M for DBL with Chudnovsky–Chudnovsky formula.
“Our experience shows that the expression of the law of addition on the cubic Hessian form (d) of an elliptic curve is by far the best and the prettiest.”

\[
X_3 = Y_1 X_2 \cdot Y_1 Z_2 - Z_1 Y_2 \cdot X_1 Y_2, \\
Y_3 = X_1 Z_2 \cdot X_1 Y_2 - Y_1 X_2 \cdot Z_1 X_2, \\
Z_3 = Z_1 Y_2 \cdot Z_1 X_2 - X_1 Z_2 \cdot Y_1 Z_2.
\]

12\(M\) for ADD, where \(M\) is the cost of multiplication in the field.

8.4\(M\) for DBL, assuming 0.8\(M\) for the cost of squaring in the field.

1990s: ECC standards instead use short Weierstrass curves in Jacobian coordinates for “the fastest arithmetic”.

15.2\(M\) for ADD, much slower than Hessian.

Why is this a good idea?
Answer: Only 7.2\(M\) for DBL with Chudnovsky–Chudnovsky formula.

2001 Bernstein: 15\(M\), 7\(M\).
“Our experience shows that the expression of the law of addition on the cubic Hessian form (d) of an elliptic curve is by far the best and the prettiest.”

\[
\begin{align*}
X_3 &= Y_1X_2 \cdot Y_1Z_2 - Z_1Y_2 \cdot X_1Y_2, \\
Y_3 &= X_1Z_2 \cdot X_1Y_2 - Y_1X_2 \cdot Z_1X_2, \\
Z_3 &= Z_1Y_2 \cdot Z_1X_2 - X_1Z_2 \cdot Y_1Z_2.
\end{align*}
\]

12\(M\) for ADD, where \(M\) is the cost of multiplication in the field.

8.4\(M\) for DBL, assuming 0.8\(M\) for the cost of squaring in the field.

1990s: ECC standards instead use short Weierstrass curves in Jacobian coordinates for “the fastest arithmetic”.

15.2\(M\) for ADD, much slower than Hessian.

Why is this a good idea? Answer: Only 7.2\(M\) for DBL with Chudnovsky–Chudnovsky formula.

2001 Bernstein: 15\(M\), 7\(M\).

Compared to Hessian, Weierstrass saves 4\(M\) in typical DBL-DBL-DBL-DBL-DBL-ADD.
Our experience shows that the
expression of the law of addition
on the cubic Hessian form
d\footnote{12}{M for ADD,
where M is the cost
of multiplication in the field.}
of an elliptic curve is
the best and the prettiest.

\[ X^3 = Y_1 X_2 \cdot Y_2 - Z_1 Y_2 \cdot X_1 Y_2, \]
\[ Z_2 \cdot X_1 Y_2 - Y_1 X_2 \cdot Z_1 X_2, \]
\[ Y_2 \cdot Z_1 X_2 - X_1 Z_2 \cdot Y_1 Z_2. \]

\footnote{8}{4 M for DBL,\footnote{15}{2 M for ADD,
assuming 0 : 8 M for the cost
of squaring in the field.}}

1990s: ECC standards instead
use short Weierstrass curves
in Jacobian coordinates
for “the fastest arithmetic”.

15.2M for ADD,
much slower than Hessian.

Why is this a good idea?
Answer: Only 7.2M for DBL with
Chudnovsky–Chudnovsky formula.

2001 Bernstein: 15M, 7M.

Compared to Hessian,
Weierstrass saves 4M in typical
DBL-DBL-DBL-DBL-DBL-ADD.

Example:
\[ ((x_1 y_2 + y_1 x_2) = (1 - 30 x_1 x_2 y_1 y_2), \]
\[ (y_1 y_2 - x_1 x_2) = (1 + 30 x_1 x_2 y_1 y_2)). \]

2007 Bernstein–Lange: generalize,
analyze speed, completeness.

Neutral = (0; 1)
\footnote{2007}{Edwards: new curve shape.}
\footnote{2007}{Bernstein–Lange: generalize,
analyze speed, completeness.}
Our experience shows that the expression of the law of addition on the cubic Hessian form (d) of an elliptic curve is by far the best and the prettiest.

\[
X^3 = Y_1X_2 \cdot Y_2 - Z_1Y_2 \cdot X_1Y_2,
\]

\[
Y_3 = X_1Z_2 \cdot X_1 - Y_1X_2 \cdot Z_1X_2,
\]

\[
Z_3 = Z_1Y_2 \cdot Z_1 - X_1Z_2 \cdot Y_1Z_2.
\]

1990s: ECC standards instead use short Weierstrass curves in Jacobian coordinates for “the fastest arithmetic”.

15.2\(M\) for ADD, much slower than Hessian.

Why is this a good idea?
Answer: Only 7.2\(M\) for DBL with Chudnovsky–Chudnovsky formula.

2001 Bernstein: 15\(M\), 7\(M\).

Compared to Hessian, Weierstrass saves 4\(M\) in typical DBL-DBL-DBL-DBL-DBL-ADD.

Example: \(x^2 + y^2 = 1\). Sum of \((x_1, y_1)\) and \((x_2, y_2)\):

\[
\text{((}x_1y_2 + y_1x_2\text{)/}(1-x_1x_2)},
\]

\[
\text{(y_1y_2-x_1x_2)/}(1-x_1x_2).
\]

2007 Bernstein–Lange: generalize, analyze speed, completeness.
Our experience shows that the expression of the law of addition on the cubic Hessian form (d) of an elliptic curve is by far the best and the prettiest.

\[ X^3 = Y_1 X_2 Z_2 - Z_1 Y_2 X_1 Y_2; \]

\[ Y_3 = X_1 Z_2 X_1 Y_2 - Y_1 X_2 Z_1 X_2 Z_2; \]

\[ Z_3 = Z_1 Y_2 Z_1 X_2 - X_1 Z_2 Y_1 X_2 Z_2; \]

12M for ADD, where M is the cost of multiplication in the field.

8: 4M for DBL, assuming 0: 8M for the cost of squaring in the field.

1990s: ECC standards instead use short Weierstrass curves in Jacobian coordinates for “the fastest arithmetic”.

15.2M for ADD, much slower than Hessian.

Why is this a good idea?

Answer: Only 7.2M for DBL with Chudnovsky–Chudnovsky formula.

2001 Bernstein: 15M, 7M.

Compared to Hessian, Weierstrass saves 4M in typical DBL-DBL-DBL-DBL-DBL-ADD.


2007 Bernstein–Lange: generalize, analyze speed, completeness.

Example: \[ x^2 + y^2 = 1 - 30x_1 x_2 y_1 y_2. \]

Sum of \((x_1, y_1)\) and \((x_2, y_2)\) is

\[ ((x_1 y_2 + y_1 x_2)/(1-30x_1 x_2 y_1 y_2), (y_1 y_2 - x_1 x_2)/(1+30x_1 x_2 y_1 y_2))]. \]
1990s: ECC standards instead use short Weierstrass curves in Jacobian coordinates for “the fastest arithmetic”.

15.2\text{M} for ADD, much slower than Hessian.

Why is this a good idea?
Answer: Only 7.2\text{M} for DBL with Chudnovsky–Chudnovsky formula.

2001 Bernstein: 15\text{M}, 7\text{M}.

Compared to Hessian, Weierstrass saves 4\text{M} in typical DBL-DBL-DBL-DBL-DBL-ADD.

2007 Bernstein–Lange: generalize, analyze speed, completeness.

Example: $x^2 + y^2 = 1 - 30x^2y^2$. Sum of $(x_1, y_1)$ and $(x_2, y_2)$ is

\[
\left(\frac{x_1y_2+y_1x_2}{1-30x_1x_2y_1y_2}, \frac{y_1y_2-x_1x_2}{1+30x_1x_2y_1y_2}\right).
\]
ECC standards instead use short Weierstrass curves in Jacobian coordinates for “the fastest arithmetic”.

For ADD, much slower than Hessian.

15 M for ADD, only 7.2 M for DBL with Chudnovsky–Chudnovsky formula.

Bernstein: 15 M, 7 M.

Compared to Hessian, Weierstrass saves 4 M in typical DBL-DBL-DBL-DBL-DBL-ADD.

2007 Bernstein–Lange: generalize, analyze speed, completeness.

Example: $x^2 + y^2 = 1 - 30x^2y^2$.
Sum of $(x_1, y_1)$ and $(x_2, y_2)$ is

$((x_1y_2+y_1x_2)/(1-30x_1x_2y_1y_2), (y_1y_2-x_1x_2)/(1+30x_1x_2y_1y_2))$. 

2007 Bernstein–Lange: 10.8 M for ADD, 8 M for DBL.
1990s: ECC standards instead use short Weierstrass curves in Jacobian coordinates for “the fastest arithmetic”.

Hessian.

Why is this a good idea? 15 : \(2M\) for ADD, much slower than Hessian.

2001 Bernstein: 15 M, 7 M. Compared to Hessian, Weierstrass saves 4 M in typical DBL-DBL-DBL-DBL-DBL-ADD.


2007 Bernstein–Lange: generalize, analyze speed, completeness.

\[ \text{Example: } x^2 + y^2 = 1 - 30x^2y^2. \]

Sum of \((x_1, y_1)\) and \((x_2, y_2)\) is \(( (x_1y_2 + y_1x_2) / (1 - 30x_1x_2y_1y_2), (y_1y_2 - x_1x_2) / (1 + 30x_1x_2y_1y_2) )\).
1990s: ECC standards instead use short Weierstrass curves in Jacobian coordinates for "the fastest arithmetic."

15: \[ 2M \text{ for ADD, much slower than Hessian.} \]

Why is this a good idea?
Answer: Only 7: \[ 2M \text{ for DBL with Chudnovsky–Chudnovsky formula.} \]

2001 Bernstein: 15M, 7M. Compared to Hessian, Weierstrass saves 4M in typical DBL-DBL-DBL-DBL-DBL-ADD.

2007 Bernstein–Lange: generalize, analyze speed, completeness.

Example: \[ x^2 + y^2 = 1 - 30x^2y^2. \]
Sum of \((x_1, y_1)\) and \((x_2, y_2)\) is \((x_3, y_3)\) with
\[ P_1 = (x_1, y_1), \quad P_2 = (x_2, y_2), \quad P_3 = (x_3, y_3) \]

2007 Bernstein–Lange: 10.8M for ADD, 6.2M for DBL.
2007 Bernstein–Lange: generalize, analyze speed, completeness.

Example: \( x^2 + y^2 = 1 - 30x^2y^2 \).
Sum of \((x_1, y_1)\) and \((x_2, y_2)\) is 
\[
\left( \frac{x_1y_2 + y_1x_2}{1 - 30x_1x_2y_1y_2} \right),
\left( \frac{y_1y_2 - x_1x_2}{1 + 30x_1x_2y_1y_2} \right).
\]

2007 Bernstein–Lange: 10.8M for ADD, 6.2M for DBL.
2007 Bernstein–Lange: generalize, analyze speed, completeness.

Example: \( x^2 + y^2 = 1 - 30x^2y^2 \).
Sum of \((x_1, y_1)\) and \((x_2, y_2)\) is
\[
(\frac{x_1y_2 + y_1x_2}{1 - 30x_1x_2y_1y_2}, \frac{y_1y_2 - x_1x_2}{1 + 30x_1x_2y_1y_2}).
\]

2007 Bernstein–Lange: 10.8\( \text{M} \) for ADD, 6.2\( \text{M} \) for DBL.

2008 Hisil–Wong–Carter–Dawson: just 8\( \text{M} \) for ADD.
2007 Bernstein–Lange: generalize, analyze speed, completeness.

\[ y \]
\[ \uparrow \]
\[ \rightarrow \]
\[ \text{neutral} = (0, 1) \]
\[ P_1 = (x_1, y_1) \]
\[ P_2 = (x_2, y_2) \]
\[ P_3 = (x_3, y_3) \]

Example: \( x^2 + y^2 = 1 - 30x^2y^2 \).
Sum of \((x_1, y_1)\) and \((x_2, y_2)\) is
\[
\left( \frac{x_1y_2+y_1x_2}{1-30x_1x_2y_1y_2}, \frac{y_1y_2-x_1x_2}{1+30x_1x_2y_1y_2} \right).
\]

2007 Bernstein–Lange: 10.8M for ADD, 6.2M for DBL.
2008 Hisil–Wong–Carter–Dawson: just 8M for ADD.
Bernstein–Lange: generalize, analyze speed, completeness.

\[
\begin{align*}
\text{neutral} &= (0, 1) \\
P_1 &= (x_1, y_1) \\
P_2 &= (x_2, y_2) \\
P_3 &= (x_3, y_3)
\end{align*}
\]

\[y^2 = x^3\]

Example: \(x^2 + y^2 = 1 - 30x^2y^2\).

\((x_1, y_1)\) and \((x_2, y_2)\) is \((y_1x_2)/(1-30x_1x_2y_1y_2), x_1x_2)/(1+30x_1x_2y_1y_2)).

2007 Bernstein–Lange:
10.8M for ADD, 6.2M for DBL.

2008 Hisil–Wong–Carter–Dawson:
just 8M for ADD.
2007 Bernstein–Lange: generalize, analyze speed, completeness.

neutral = (0, 1)

$P_1 = (x_1, y_1)$

$P_2 = (x_2, y_2)$

$P_3 = (x_3, y_3)$

$y^2 = x^3 - 0.4x + 0.7$

Example:

$x^2 + y^2 = 1 - 30x^2y^2$.

And $(x_2, y_2)$ is

$(1 - 30x_1x_2y_1y_2)$,

$(1 + 30x_1x_2y_1y_2)$.

2007 Bernstein–Lange:

10.8\text{M} for ADD, 6.2\text{M} for DBL.

2008 Hisil–Wong–Carter–Dawson:

just 8\text{M} for ADD.
2007 Bernstein–Lange: generalize, analyze speed, completeness.

\[
y = x^2y^2.
\]

Example:
\[x^2 + y^2 = 1 - 30xxyy.
\]

\[P_1 = (x_1; y_1), \quad P_2 = (x_2; y_2), \quad P_3 = (x_3; y_3)
\]

\[(x_1y_2 + y_1x_2) = (1 - 30xxyy),
\]

\[(y_1y_2 - x_1x_2) = (1 + 30xxyy).
\]

2007 Bernstein–Lange: 10.8M for ADD, 6.2M for DBL.

2008 Hisil–Wong–Carter–Dawson: just 8M for ADD.

\[y^2 = x^3 - 0.4x + 0.7
\]
2007 Bernstein–Lange:
10.8M for ADD, 6.2M for DBL.

2008 Hisil–Wong–Carter–Dawson:
just 8M for ADD.

\[ y^2 = x^3 - 0.4x + 0.7 \]
2007 Bernstein–Lange:
for ADD, $6.2\text{M}$ for DBL.

2008 Hisil–Wong–Carter–Dawson:
just $8\text{M}$ for ADD.

$$y^2 = x^3 - 0.4x + 0.7$$
2007 Bernstein–Lange:
10 : 8 M for ADD, 6 : 2 M for DBL.

2008 Hisil–Wong–Carter–Dawson:
just 8 M for ADD.

$y^2 = x^3 - 0.4x + 0.7$

The Weierstrass-turtle: old, trusted, and slow. Warning: (picture) incomplete.
2007 Bernstein–Lange:
10 : 8 M for ADD, 6 : 2 M for DBL.

2008 Hisil–Wong–Carter–Dawson:
just 8 M for ADD.

\[ y^2 = x^3 - 0.4x + 0.7 \]

The Weierstrass-turtle: old, trusted and slow. Warning: (picture) incomplete!
$y^2 = x^3 - 0.4x + 0.7$

The Weierstrass-turtle: old, trusted and slow. Warning: (picture) incomplete!
The Weierstrass-turtle: old, trusted and slow. Warning: (picture) incomplete!
The WeierstrASS-turtle: old, trusted and slow. Warning: (picture) incomplete!

\[ x^2 + y^2 = 1 - 300 \]
The Weierstrass-turtle: old, trusted and slow. Warning: (picture) incomplete!

\[ x^2 + y^2 = 1 - 300x^2y^2 \]
The Weierstrass turtle: old, trusted and slow. Warning: (picture) incomplete!

\[ x^2 + y^2 = 1 - 300x^2y^2 \]
\[ x^2 + y^2 = 1 - 300x^2y^2 \]
$x^2 + y^2 = 1 - 300x^2y^2$

The Edwards starfish: new, fast and complete!
The Edwards starfish: new, fast and complete!
The Edwards starfish: new, fast and complete!
\[ x^2 + y^2 = 1 - 300x^2y^2 \]

The Edwards starfish: new, fast and complete!

\[ x^2 = y^4 \]
The Edwards starfish: new, fast and complete!
The Edwards starfish: new, fast and complete!

\[ x^2 = y^4 - 1.9y^2 + 1 \]
The Edwards starfish: new, fast and complete!

\[ x^2 = y^4 - 1.9y^2 + 1 \]
new, complete:

\[ x^2 = y^4 - 1.9y^2 + 1 \]
The Jacobi-quartic extended to XXYZZR giant squid.
The Jacobi-quartic squid: can be extended to XXYZZR giant squid.

\[ x^2 = y^4 - 1.9y^2 + 1 \]
The Jacobi-quartic squid: can be extended to XXYZZR giant squid.

\[ x^2 = y^4 - 1.9y^2 + 1 \]
The Jacobi-quartic squid: can be extended to $XXYZZR$ giant squid.
The Jacobi-quartic squid: can be extended to XXYZZR giant squid.
The Jacobi-quartic squid: can be extended to XXYZZR giant squid.

\[ x^3 - y^3 + 1 = 0.3xy \]
The Jacobi-quartic squid: can be extended to XXYZZR giant squid.

\[ x^3 - y^3 + 1 = 0.3xy \]
Kobi-quartic squid: can be related to R^2 squid.

\[ x^3 - y^3 + 1 = 0.3xy \]
x^3 - y^3 + 1 = 0.3xy
\[ x^3 - y^3 + 1 = 0.3xy \]
\[ x^3 - y^3 + 1 = 0.3xy \]

The Hessian-ray: uniform

but not strongly so
\[ 3x - y + 1 = 0.3xy \]

The Hessian-ray: uniform

but not strongly so
The Hessian-ray: uniform but not strongly so
The Hessian-ray: uniform
but not strongly so
The Hessian-ray: uniform but not strongly so
The Hessian-ray: uniform
but not strongly so
sian-ray: uniform

but

not strongly so
uniform

but

strangely so

1985
Mar

Zoom
Mar

Zoom
Mar

Zoom
Faster Hessian arithmetic
2007 Hisil–Carter–Dawson: 7 : 8 M for DBL.
Faster Hessian arithmetic
2007 Hisil–Carter–Dawson:
7.8M for DBL.
Faster Hessian arithmetic
2007 Hisil–Carter–Dawson: 7.8M for DBL.
Faster Hessian arithmetic

2007 Hisil–Carter–Dawson:
7.8M for DBL.
Faster Hessian arithmetic

2007 Hisil–Carter–Dawson: 7.8M for DBL.

2010 Hisil: 11M for ADD.
Faster Hessian arithmetic

2007 Hisil–Carter–Dawson: 7.8M for DBL.

2010 Hisil: 11M for ADD.

Hessian tied with Weierstrass for DBL-DBL-DBL-DBL-DBL-ADD.

Need to zoom in closer: analyze exact S/M, overhead for checking for special cases, extra DBL, extra ADD, etc.
Faster Hessian arithmetic

2007 Hisil–Carter–Dawson: $7.8M$ for DBL.

2010 Hisil: $11M$ for ADD.

Hessian tied with Weierstrass for DBL-DBL-DBL-DBL-DBL-ADD.

Need to zoom in closer: analyze exact $S/M$, overhead for checking for special cases, extra DBL, extra ADD, etc.

Or speed up Hessian more.
Faster Hessian arithmetic

2007 Hisil–Carter–Dawson: 7.8M for DBL.

2010 Hisil: 11M for ADD.

Hessian tied with Weierstrass for DBL-DBL-DBL-DBL-ADD.

Need to zoom in closer: analyze exact S/M, overhead for checking for special cases, extra DBL, extra ADD, etc.

Or speed up Hessian more.

New: 7.6M for DBL.
Faster Hessian arithmetic

2007 Hisil–Carter–Dawson: 7.8\text{M} for DBL.

2010 Hisil: 11\text{M} for ADD.

Hessian tied with Weierstrass for DBL-DBL-DBL-DBL-DBL-ADD.

Need to zoom in closer:
analyze exact \textbf{S/M}, overhead for checking for special cases, extra DBL, extra ADD, etc.

Or speed up Hessian more.

New: 7.6\text{M} for DBL.

New (announced July 2009):

Generalize to more curves:

\textit{twisted} Hessian curves

\( aX^3 + Y^3 + Z^3 = dXYZ \)

with \( a(27a - d^3) \neq 0 \).

2007 7.8\text{M} DBL idea fails, but

2010 11\text{M} ADD generalizes,

new 7.6\text{M} DBL generalizes.
Faster Hessian arithmetic

2007 Hisil–Carter–Dawson: 7.8M for DBL.

2010 Hisil: 11M for ADD.

Hessian tied with Weierstrass for DBL-DBL-DBL-DBL-DBL-ADD.

Need to zoom in closer:
analyze exact S/M, overhead for checking for special cases, extra DBL, extra ADD, etc.

Or speed up Hessian more.

New: 7.6M for DBL.

New (announced July 2009):
Generalize to more curves:
twisted Hessian curves
\[ aX^3 + Y^3 + Z^3 = dXYZ \]
with \( a(27a - d^3) \neq 0 \).

2007 7.8M DBL idea fails, but 2010 11M ADD generalizes, new 7.6M DBL generalizes.
Faster Hessian arithmetic

2007 Hisil–Carter–Dawson: 7.8M for DBL.

2010 Hisil: 11M for ADD.

Hessian tied with Weierstrass for DBL-DBL-DBL-DBL-DBL-ADD.

Need to zoom in closer:
analyze exact S/M, overhead for checking for special cases, extra DBL, extra ADD, etc.

Or speed up Hessian more.

New: 7.6M for DBL.

New (announced July 2009):
Generalize to more curves: twisted Hessian curves
\[ aX^3 + Y^3 + Z^3 = dXYZ \]
with \( a(27a - d^3) \neq 0 \).

2007 7.8M DBL idea fails, but
2010 11M ADD generalizes, new 7.6M DBL generalizes.
Faster Hessian arithmetic

2007 Hisil–Carter–Dawson:
7.8M for DBL.

2010 Hisil: 11M for ADD.

Hessian tied with Weierstrass for DBL-DBL-DBL-DBL-DBL-ADD.

Need to zoom in closer:
analyze exact $S/M$, overhead for checking for special cases, extra DBL, extra ADD, etc.

Or speed up Hessian more.

New: 7.6M for DBL.

New (announced July 2009):
Generalize to more curves:
**twisted Hessian curves**

$$aX^3 + Y^3 + Z^3 = dXYZ$$
with $a(27a - d^3) \neq 0$.

2007 7.8M DBL idea fails, but 2010 11M ADD generalizes, new 7.6M DBL generalizes.
Faster Hessian arithmetic

2007 Hisil–Carter–Dawson: 7.8M for DBL.

2010 Hisil: 11M for ADD.

Hessian tied with Weierstrass for DBL-DBL-DBL-DBL-DBL-ADD.

Need to zoom in closer: analyze exact S/M, overhead for checking for special cases, extra DBL, extra ADD, etc.

Or speed up Hessian more.

New: 7.6M for DBL.

New (announced July 2009):

Generalize to more curves: twisted Hessian curves

\[ aX^3 + Y^3 + Z^3 = dXYZ \]

with \( a(27a - d^3) \neq 0 \).

2007 7.8M DBL idea fails, but 2010 11M ADD generalizes, new 7.6M DBL generalizes.

Rotate addition law so that it also works for DBL; complete if \( a \) is not a cube.

Eliminates special-case overhead, helps stop side-channel attacks.
Hessian arithmetic

2007 Hisil–Carter–Dawson: 7 : 8 M for DBL.

2010 Hisil: 11 M for ADD.

Triplings (assuming $d \neq 0$)
TPL is $P \mapsto 3P$.

New (announced July 2009):
Generalize to more curves: twisted Hessian curves $aX^3 + Y^3 + Z^3 = dXYZ$ with $a(27a - d^3) \neq 0$.

2007 7.8 M DBL idea fails, but 2007 12 : 8 M for Hessian TPL.
Generalizes to twisted Hessian.

New: 7 : 6 M for DBL.

Hessian tied with Weierstrass for DBL-DBL-DBL-DBL-DBL-ADD.

Need to zoom in closer:
Exact $S/M$, overhead for checking for special cases, extra DBL, extra ADD, etc.

Or speed up Hessian more.

New: 7 : 6 M for DBL.

Rotate addition law so that it also works for DBL;
complete if $a$ is not a cube.

Eliminates special-case overhead, helps stop side-channel attacks.
New (announced July 2009):

Generalize to more curves:

**twisted Hessian curves**

\[ aX^3 + Y^3 + Z^3 = dXYZ \]

with \( a(27a - d^3) \neq 0 \).

2007 7.8M DBL idea fails, but 2010 11M ADD generalizes, new 7.6M DBL generalizes.

**Rotate** addition law

so that it also works for DBL;

**complete** if \( a \) is not a cube.

Eliminates special-case overhead, helps stop side-channel attacks.
New (announced July 2009):
Generalize to more curves:
**twisted Hessian curves**
\[ aX^3 + Y^3 + Z^3 = dXYZ \]
with \( a(27a - d^3) \neq 0 \).

2007 7.8M DBL idea fails, but
2010 11M ADD generalizes,
new 7.6M DBL generalizes.

**Rotate** addition law
so that it also works for DBL;
**complete** if \( a \) is not a cube.
Eliminates special-case overhead,
helps stop side-channel attacks.

Triplings (assuming \( d \neq 0 \))
TPL is \( P \mapsto 3P \).

2007 Hisil–Carter–Dawson:
12.8M for Hessian TPL.
Generalizes to twisted Hessian.
New (announced July 2009):

Generalize to more curves: twisted Hessian curves
\[ aX^3 + Y^3 + Z^3 = dXYZ \]
with \( a(27a - d^3) \neq 0 \).

2007 7.8M DBL idea fails, but 2010 11M ADD generalizes,
new 7.6M DBL generalizes.

**Rotate** addition law
so that it also works for DBL;
**complete** if \( a \) is not a cube.
Eliminates special-case overhead,
helps stop side-channel attacks.

Triplings (assuming \( d \neq 0 \))
TPL is \( P \mapsto 3P \).

2007 Hisil–Carter–Dawson:
12.8M for Hessian TPL.
Generalizes to twisted Hessian.
New (announced July 2009):

Generalize to more curves:

**twisted Hessian curves**

\[ aX^3 + Y^3 + Z^3 = dXYZ \]

with \( a(27a - d^3) \neq 0 \).

2007 7.8M DBL idea fails, but
2010 11M ADD generalizes,
new 7.6M DBL generalizes.

**Rotate** addition law
so that it also works for DBL;
**complete** if \( a \) is not a cube.
Eliminates special-case overhead,
helps stop side-channel attacks.

Triplings (assuming \( d \neq 0 \))

TPL is \( P \mapsto 3P \).

2007 Hisil–Carter–Dawson:
12.8M for Hessian TPL.
Generalizes to twisted Hessian.

2015 Kohel: 11.2M.
New (announced July 2009):

Generalize to more curves:
**twisted Hessian curves**

\[ aX^3 + Y^3 + Z^3 = dXYZ \]

with \( a(27a - d^3) \neq 0 \).

2007 7.8M DBL idea fails, but 2010 11M ADD generalizes, new 7.6M DBL generalizes.

**Rotate** addition law
so that it also works for DBL;
**complete** if \( a \) is not a cube.
Eliminates special-case overhead, helps stop side-channel attacks.

2007 Hisil–Carter–Dawson: 12.8M for Hessian TPL.
Generalizes to twisted Hessian.

2015 Kohel: 11.2M.

New: 10.8M assuming field with fast primitive \( \sqrt[3]{1} \);
e.g., \( F_q[\omega]/(\omega^2 + \omega + 1) \), or \( F_p \) with \( 7p = 2^{298} + 2^{149} + 1 \).

(More history in small char. See paper for details.)

Triplings (assuming \( d \neq 0 \))

TPL is \( P \mapsto 3P \).

2007 Hisil–Carter–Dawson: 12.8M for Hessian TPL.
New (announced July 2009):
Generalize to more curves:
Hessian curves
\[ aX^3 + Y^3 + Z^3 = dXYZ \]
with \( a (27a - d^3) \neq 0. \)

2007 Hisil–Carter–Dawson:
12.8M for Hessian TPL.

2015 Kohel: 11.2M.

New: 10.8M assuming
field with fast primitive \( \sqrt[3]{1}; \)
e.g., \( F_q[\omega] / (\omega^2 + \omega + 1), \) or
\( F_p \) with \( 7p = 2^{298} + 2^{149} + 1. \)

Compose these 3-isogenies:
\( (X^3 : Y^3 : Z^3) = 3(X : Y : Z). \)

Triplings (assuming \( d \neq 0) \)
TPL is \( P \mapsto 3P. \)

Generalizes to twisted Hessian.

If \( aX^3 + Y^3 + Z^3 = dXYZ \) then \( VW(V + dU + aW) = U^3 \)
where
\( U = -XYZ, \)
\( V = Y^3, \)
\( W = X^3. \)

If \( VW(V + dU + aW) = U^3 \) then \( aX^3 + Y^3 + Z^3 = dXYZ \)
where
\( Q = dU, \)
\( R = aW, \)
\( S = -(V + Q + R), \)
\( dX_3 = P^3, \)
\( Y_3 = RS, \)
\( Z_3 = RV. \)

(See paper for details.)
New (announced July 2009):
Generalize to more curves:
twisted Hessian curves
\[ aX^3 + Y^3 + Z^3 = dXYZ \]
with \( a (27a^2 - d^3) \neq 0 \).

2007 Hisil–Carter–Dawson:
12.8M for Hessian TPL.

Generalizes to twisted Hessian.

2015 Kohel: 11.2M.

New: 10.8M assuming field with fast primitive \( 3\sqrt{1} \);
e.g., \( \mathbb{F}_q[\omega]/(\omega^2 + \omega + 1) \), or \( \mathbb{F}_p \) with \( 7p = 2^{298} + 2^{149} + 1 \).

Triplings (assuming \( d \neq 0 \))

TPL is \( P \mapsto 3P \).

2007 Hisil–Carter–Dawson:
12.8M for Hessian TPL.

Generalizes to twisted Hessian.

2015 Kohel: 11.2M.

New: 10.8M assuming field with fast primitive \( 3\sqrt{1} \);
e.g., \( \mathbb{F}_q[\omega]/(\omega^2 + \omega + 1) \), or \( \mathbb{F}_p \) with \( 7p = 2^{298} + 2^{149} + 1 \).

Triplings (assuming \( d \neq 0 \))

TPL is \( P \mapsto 3P \).

2007 Hisil–Carter–Dawson:
12.8M for Hessian TPL.

Generalizes to twisted Hessian.

2015 Kohel: 11.2M.

New: 10.8M assuming field with fast primitive \( 3\sqrt{1} \);
e.g., \( \mathbb{F}_q[\omega]/(\omega^2 + \omega + 1) \), or \( \mathbb{F}_p \) with \( 7p = 2^{298} + 2^{149} + 1 \).

Compose these 3-isogenies:
\( (X_3 : Y_3 : Z_3) = 3( X : Y : Z ) \).
Triplings (assuming \(d \neq 0\))

TPL is \(P \mapsto 3P\).

2007 Hisil–Carter–Dawson:
12.8M for Hessian TPL.

Generalizes to twisted Hessian.

2015 Kohel: 11.2M.

New: 10.8M assuming
field with fast primitive \(\sqrt[3]{1}\);
e.g., \(\mathbb{F}_q[\omega]/(\omega^2 + \omega + 1)\), or
\(\mathbb{F}_p\) with \(7p = 2^{298} + 2^{149} + 1\).

(More history in small char.
See paper for details.)

If \(aX^3 + Y^3 + Z^3 = dXYZ\)
then \(VW(V + dU + aW) = U^3\),
where
\(U = -XYZ, V = Y^3, W = X^3\).

If \(VW(V + dU + aW) = U^3\)
then \(aX_3^3 + Y_3^3 + Z_3^3 = dX_3Y_3Z_3\),
where \(Q = dU, R = aW, S = -(V + Q + R),
\(dX_3 = R^3 + S^3 + V^3 - 3RSV,\)
\(Y_3 = RS^2 + SV^2 + VR^2 - 3RS\),
\(Z_3 = RV^2 + SR^2 + VS^2 - 3RS\).

Compose these 3-isogenies:
\((X_3 : Y_3 : Z_3) = 3(X : Y : Z)\).
Triplings (assuming $d \neq 0$)

TPL is $P \mapsto 3P$.

2007 Hisil–Carter–Dawson: $12.8\text{M}$ for Hessian TPL.

Generalizes to twisted Hessian.

2015 Kohel: $11.2\text{M}$.

New: $10.8\text{M}$ assuming field with fast primitive $\sqrt[3]{1}$; e.g., $F_q[\omega]/(\omega^2 + \omega + 1)$, or $F_p$ with $7p = 2^{298} + 2^{149} + 1$.

(More history in small char. See paper for details.)

If $aX^3 + Y^3 + Z^3 = dXYZ$, then $VW(V + dU + aW) = U^3$ where $U = -XYZ$, $V = Y^3$, $W = X^3$.

If $VW(V + dU + aW) = U^3$ then $aX_3^3 + Y_3^3 + Z_3^3 = dX_3Y_3Z_3$ where $Q = dU$, $R = aW$, $S = -(V + Q + R)$, $dX_3 = R^3 + S^3 + V^3 - 3RSV$, $Y_3 = RS^2 + SV^2 + VR^2 - 3RSV$, $Z_3 = RV^2 + SR^2 + VS^2 - 3RSV$.

Compose these 3-isogenies: $(X_3 : Y_3 : Z_3) = 3(X : Y : Z)$. 
Triplings (assuming $d \neq 0$)
$P \mapsto 3P$

Hisil–Carter–Dawson:
12 : 8 M for Hessian TPL.

Generalizes to twisted Hessian.

Kohel: 11 : 2 M.

New: 10 : 8 M assuming field with fast primitive $\sqrt[3]{1}$; e.g., $F_q[!] = (!]^2 + ! + 1)$, or $F_p$ with $7_p = 2^{298} + 2^{149} + 1$.

To quickly triple $(X : Y : Z)$:
Three cubings for $R; S; V$.
For three choices of constants $(\alpha, \beta, \gamma)$ compute $(\alpha R + \beta S + \gamma V) \cdot (\alpha S + \beta V + \gamma R) \cdot (\alpha V + \beta R + \gamma S) = \alpha \beta \gamma dX_3^3 + (\alpha \beta^2 + \gamma \beta + \gamma \alpha) Y_3^3 + (\beta \alpha^2 + \gamma \alpha + \gamma \beta) Z_3^3$.

Also use $a (R + S + V)^3 = d^3 RSV$.
Solve for $dX_3^3; Y_3^3; Z_3^3$.

Compose these 3-isogenies:
$(X_3 : Y_3 : Z_3) = 3(X : Y : Z)$.

If $aX^3 + Y^3 + Z^3 = dXYZ$
then $VW(V + dU + aW) = U^3$
where $U = -XYZ$, $V = Y^3$, $W = X^3$.

If $VW(V + dU + aW) = U^3$
then $aX_3^3 + Y_3^3 + Z_3^3 = dX_3 Y_3 Z_3$
where $Q = dU$, $R = aW$,
$S = -(V + Q + R)$,
$dX_3 = R^3 + S^3 + V^3 - 3RSV$,
$Y_3 = RS^2 + SV^2 + VR^2 - 3RSV$,
$Z_3 = RV^2 + SR^2 + VS^2 - 3RSV$.

See paper for details.
If \( d \neq 0 \)

TPL is \( P \mapsto 3P \).

2007 Hisil–Carter–Dawson:
12 : 8 M for Hessian TPL.
Generalizes to twisted Hessian.
2015 Kohel: 11 : 2 M.
New: 10 : 8 M assuming field with fast primitive 3\( \sqrt{1} \); e.g., \( \mathbb{F}_{q^2} \left[ \omega \right] = (\omega^2 + \omega + 1), \) or \( \mathbb{F}_p \) with 7\( p = 2^{298} + 2^{149} + 1 \).
(More history in small char. See paper for details.)

To quickly triple \( (X : Y : Z) \):
Three cubings for \( R; S; V \).
For three choices of constants \((\alpha, \beta, \gamma)\) compute
\((\alpha R + \beta S + \gamma V) \cdot (\alpha S + \beta V + \gamma R) \cdot (\alpha V + \beta R + \gamma S) = \alpha \beta \gamma dX_3 \)
+ (\alpha \beta^2 + \beta \gamma^2 + \gamma \alpha^2)
+ (\beta \alpha^2 + \gamma \beta^2 + \alpha \gamma^2)
+ (\alpha + \beta + \gamma)^3 RSV .

Also use \( a(R + S + V)^3 = d^3 RSV \).

Solve for \( dX_3, Y_3, Z_3 \).

If \( aX^3 + Y^3 + Z^3 = dXYZ \)
then \( VW(V + dU + aW) = U^3 \)
where
\( U = -XYZ, V = Y^3, W = X^3 \).

If \( VW(V + dU + aW) = U^3 \)
then \( aX^3 + Y^3 + Z^3 = dX_3 Y_3 Z_3 \)
where \( Q = dU, R = aW, \)
\( S = -(V + Q + R), \)
\( dX_3 = R^3 + S^3 + V^3 - 3RSV, \)
\( Y_3 = RS^2 + SV^2 + VR^2 - 3RSV, \)
\( Z_3 = RV^2 + SR^2 + VS^2 - 3RSV. \)

Compose these 3-isogenies:
\( (X_3 : Y_3 : Z_3) = 3(X : Y : Z) \).
If \( aX^3 + Y^3 + Z^3 = dXYZ \) then \( VW(V + dU + aW) = U^3 \) where 
\[
U = -XYZ, \quad V = Y^3, \quad W = X^3.
\]
If \( VW(V + dU + aW) = U^3 \) then \( aX^3 + Y^3 + Z^3 = dX_3Y_3Z_3 \) where \( Q = dU, \quad R = aW, \quad S = -(V + Q + R), \quad dX_3 = R^3 + S^3 + V^3 - 3RSV, \quad Y_3 = RS^2 + SV^2 + VR^2 - 3RSV, \quad Z_3 = RV^2 + SR^2 + VS^2 - 3RSV. \)

Compose these 3-isogenies:
\[
\]

To quickly triple \((X : Y : Z)\):
Three cubings for \( R, S, V \).
For three choices of constants \((\alpha, \beta, \gamma)\) compute
\[
(\alpha R + \beta S + \gamma V) \cdot (\alpha S + \beta V + \gamma R) \cdot (\alpha V + \beta R + \gamma S) = \alpha \beta \gamma dX_3 \\
+ (\alpha \beta^2 + \beta \gamma^2 + \gamma \alpha^2) Y_3 \\
+ (\beta \alpha^2 + \gamma \beta^2 + \alpha \gamma^2) Z_3 \\
+ (\alpha + \beta + \gamma)^3 RSV.
\]
Also use \( a(R + S + V)^3 = d^3 RSV \).
Solve for \( dX_3, Y_3, Z_3 \).
If \( aX^3 + Y^3 + Z^3 = dXYZ \)
then \( VW(V + dU + aW) = U^3 \)
where
\( U = -XYZ \), \( V = Y^3 \), \( W = X^3 \).

If \( VW(V + dU + aW) = U^3 \)
then \( aX_3^3 + Y_3^3 + Z_3^3 = dX_3Y_3Z_3 \)
where \( Q = dU \), \( R = aW \),
\( S = -(V + Q + R) \),
\( dX_3 = R^3 + S^3 + V^3 - 3RSV \),
\( Y_3 = RS^2 + SV^2 + VR^2 - 3RSV \),
\( Z_3 = RV^2 + SR^2 + VS^2 - 3RSV \).

Compose these 3-isogenies:
\( (X_3 : Y_3 : Z_3) = 3(X : Y : Z) \).

To quickly triple \((X : Y : Z)\):
Three cubings for \( R, S, V \).

For three choices of constants
\((\alpha, \beta, \gamma)\) compute
\((\alpha R + \beta S + \gamma V) \cdot \\
(\alpha S + \beta V + \gamma R) \cdot \\
(\alpha V + \beta R + \gamma S) \cdot \\
= \alpha\beta\gamma dX_3 \\
+ (\alpha\beta^2 + \beta\gamma^2 + \gamma\alpha^2)Y_3 \\
+ (\beta\alpha^2 + \gamma\beta^2 + \alpha\gamma^2)Z_3 \\
+ (\alpha + \beta + \gamma)^3 RSV \).

Also use \( a(R + S + V)^3 = d^3 RSV \).
Solve for \(dX_3, Y_3, Z_3\).
\[-Y^3 + Z^3 = dXYZ\]
\[V(V + dU + aW) = U^3\]
\[(YZ, V = Y^3, W = X^3).\]
\[V + dU + aW) = U^3\]
\[X_3^3 + Y_3^3 + Z_3^3 = dX_3Y_3Z_3\]
\[d = dU, R = aW,\]
\[V + Q + R),\]
\[R^3 + S^3 + V^3 - 3RSV,\]
\[S^2 + SV^2 + VR^2 - 3RSV,\]
\[V^2 + SR^2 + VS^2 - 3RSV.\]

To quickly triple \((X : Y : Z):\)
Three cubings for \(R, S, V.\)
For three choices of constants \((\alpha, \beta, \gamma)\) compute
\((\alpha R + \beta S + \gamma V) \cdot\]
\((\alpha S + \beta V + \gamma R) \cdot\]
\((\alpha V + \beta R + \gamma S)\]
\[= \alpha \beta \gamma dX_3\]
\[+ (\alpha \beta^2 + \beta \gamma^2 + \gamma \alpha^2)Y_3\]
\[+ (\beta \alpha^2 + \gamma \beta^2 + \alpha \gamma^2)Z_3\]
\[+ (\alpha + \beta + \gamma)^3 RSV.\]
Also use \(a(R + S + V)^3 = d^3 RSV.\)
Solve for \(dX_3, Y_3, Z_3.\)
\[
\begin{align*}
X^3 + Y^3 + Z^3 &= dXYZ \\
VW(V + dU + aW) &= U^3
\end{align*}
\]
where
\[
U = -XYZ, 
V = Y^3, \quad W = X^3.
\]

To quickly triple \((X : Y : Z)\):

Three cubings for \(R, S, V\).

For three choices of constants \((\alpha, \beta, \gamma)\) compute
\[
\begin{align*}
(\alpha R + \beta S + \gamma V) \\
(\alpha S + \beta V + \gamma R) \\
(\alpha V + \beta R + \gamma S)
\end{align*}
\]
\[
= \alpha \beta \gamma dX_3 \\
+ \alpha \beta^2 \gamma^2 + \gamma \alpha \gamma^2 Y_3 \\
+ \beta \alpha^2 \gamma^2 + \alpha \gamma^2 Z_3 \\
+ (\alpha + \beta + \gamma)^3 RSV.
\]

Also use \(a(R + S + V)^3 = d^3 RSV\).

Solve for \(dX_3, Y_3, Z_3\).
To quickly triple \((X : Y : Z)\):

Three cubings for \(R, S, V\).

For three choices of constants \((\alpha, \beta, \gamma)\) compute

\[
(\alpha R + \beta S + \gamma V) \cdot (\alpha S + \beta V + \gamma R) \cdot (\alpha V + \beta R + \gamma S) = \alpha \beta \gamma dX_3 + (\alpha \beta^2 + \beta \gamma^2 + \gamma \alpha^2)Y_3 + (\beta \alpha^2 + \gamma \beta^2 + \alpha \gamma^2)Z_3 + (\alpha + \beta + \gamma)^3 RSV.
\]

Also use \(a(R + S + V)^3 = d^3 RSV\).

Solve for \(dX_3, Y_3, Z_3\).

2015 Kohel’s 11.2M (4 cubings + 4 mults) introduced this TPL idea with

\((\alpha, \beta, \gamma) = (1, 1, 1)\),

\((\alpha, \beta, \gamma) = (1, -1, 0)\),

\((\alpha, \beta, \gamma) = (1, 1, 0)\).
To quickly triple \((X : Y : Z)\):

Three cubings for \(R, S, V\).

For three choices of constants \((\alpha, \beta, \gamma)\) compute

\[
(\alpha R + \beta S + \gamma V) \cdot \\
(\alpha S + \beta V + \gamma R) \cdot \\
(\alpha V + \beta R + \gamma S)
\]

\[
= \alpha \beta \gamma dX
\]

\[
+ (\alpha^2 \beta + \beta \gamma^2 + \gamma \alpha^2) Y
\]

\[
+ (\beta \alpha^2 + \gamma \beta^2 + \alpha \gamma^2) Z
\]

\[
+ (\alpha + \beta + \gamma)^3 RSV.
\]

Also use \(a(R + S + V)^3 = d^3 RSV\).

Solve for \(dX, Y, Z\).

2015 Kohel’s 11.2M

(4 cubings + 4 mults)

introduced this TPL idea with

\[
(\alpha, \beta, \gamma) = (1, 1, 1),
\]

\[
(\alpha, \beta, \gamma) = (1, -1, 0),
\]

\[
(\alpha, \beta, \gamma) = (1, 1, 0).
\]
To quickly triple \((X : Y : Z)\):

Three cubings for \(R, S, V\).

For three choices of constants \((\alpha, \beta, \gamma)\) compute

\[
\begin{align*}
&\left(\alpha R + \beta S + \gamma V\right) \cdot \\
&\left(\alpha S + \beta V + \gamma R\right) \cdot \\
&\left(\alpha V + \beta R + \gamma S\right) \\
= &\, \alpha \beta \gamma dX_3 \\
+ &\left(\alpha \beta^2 + \beta \gamma^2 + \gamma \alpha^2\right) Y_3 \\
+ &\left(\beta \alpha^2 + \gamma \beta^2 + \alpha \gamma^2\right) Z_3 \\
+ &\left(\alpha + \beta + \gamma\right)^3 RSV.
\end{align*}
\]

Also use \(a(R + S + V)^3 = d^3 RSV\).

Solve for \(dX_3, Y_3, Z_3\).

2015 Kohel’s 11.2\textbf{M}

(4 cubings + 4 mults)
introduced this TPL idea with

\[
\begin{align*}
&\left(\alpha, \beta, \gamma\right) = (1, 1, 1), \\
&\left(\alpha, \beta, \gamma\right) = (1, -1, 0), \\
&\left(\alpha, \beta, \gamma\right) = (1, 1, 0).
\end{align*}
\]

New 10.8\textbf{M} (6 cubings)

makes faster choices

assuming fast primitive \(\omega = \sqrt[3]{1}\):

\[
\begin{align*}
&\left(\alpha, \beta, \gamma\right) = (1, 1, 1), \\
&\left(\alpha, \beta, \gamma\right) = (1, \omega, \omega^2), \\
&\left(\alpha, \beta, \gamma\right) = (1, \omega^2, \omega).
\end{align*}
\]
To quickly triple \((X : Y : Z)\):

Three cubings for \(R, S, V\).

For three choices of constants \((\alpha, \beta, \gamma)\) compute

\[
(\alpha R + \beta S + \gamma V) \cdot (\alpha S + \beta V + \gamma R) \cdot (\alpha V + \beta R + \gamma S) = \alpha \beta \gamma \cdot d X^3 \end{equation}

\[
= (\alpha^2 \beta + \beta^2 \gamma + \gamma^2 \alpha) Y_3 = (\alpha \beta^2 + \beta \gamma^2 + \gamma \alpha^2) Z_3 = (\alpha + \gamma \beta + \beta \gamma) R S V.
\]

Also use

\[
a(R + S + V)^3 = d^3 R S V.
\]

For \(d X_3, Y_3, Z_3\).

2015 Kohel's 11.2M

(4 cubings + 4 mults)

introduced this TPL idea with

\((\alpha, \beta, \gamma) = (1, 1, 1),\)

\((\alpha, \beta, \gamma) = (1, -1, 0),\)

\((\alpha, \beta, \gamma) = (1, 1, 0).\)

New 10.8M (6 cubings)

makes faster choices

assuming fast primitive \(\omega = \sqrt[3]{1}:

\((\alpha, \beta, \gamma) = (1, 1, 1),\)

\((\alpha, \beta, \gamma) = (1, \omega, \omega^2),\)

\((\alpha, \beta, \gamma) = (1, \omega^2, \omega).\)

Are triplings useful?

2005 Dimitrov–Imbert–Mishra

"double-base chains": e.g.,
compute 314159\(P\) as

\[
2^{15}3^2 P + 2^{11}3^2 P + 2^83^2 P - 2^43^1 P - 2^03^0 P
\]

after precomputing 3\(P\); 5\(P\); 7\(P\).

3TPL, 13DBL, 6ADD.

2006 Doche–Imbert

generalized double-base chains:

e.g., compute 314159\(P\) as

\[
2^{12}3^33^1 P - 2^{7}3^35^1 P - 2^43^17^1 P - 2^03^00^1 P
\]

after precomputing 3\(P\); 5\(P\); 7\(P\).

3TPL, 13DBL, 6ADD.
To quickly triple \((X : Y : Z)\):

Three cubings for \(R, S, V\).

For three choices of constants \((\alpha, \beta, \gamma)\) compute

\[
(\alpha R + \beta S + \gamma V)^3 \times (\alpha S + \beta V + \gamma R)^3 \times (\alpha V + \beta R + \gamma S)^3 = \alpha \beta \gamma d^3 X^3 + (\alpha \beta^2 + \beta \alpha^2 + \gamma^2) Y^3 + (\alpha^2 \beta + \beta^2 \alpha + \gamma \beta) Z^3.
\]

2015 Kohel’s 11.2\(\mathbf{M}\)

(4 cubings + 4 mults)

introduced this TPL idea with

\((\alpha, \beta, \gamma) = (1, 1, 1),\)
\((\alpha, \beta, \gamma) = (1, -1, 0),\)
\((\alpha, \beta, \gamma) = (1, 1, 0).\)

New 10.8\(\mathbf{M}\) (6 cubings)
makes faster choices

assuming fast primitive \(\omega = \sqrt[3]{1}:

\((\alpha, \beta, \gamma) = (1, 1, 1),\)
\((\alpha, \beta, \gamma) = (1, \omega, \omega^2),\)
\((\alpha, \beta, \gamma) = (1, \omega^2, \omega).\)

Are triplings useful?

2005 Dimitrov–Imbert–Mishra

“double-base chains”: e.g.,
compute \(314159 \cdot P\) as

\[2^{15} 3^2 P + 2^{11} 3^2 P + 2^4 3^1 P - 2^0 3^0 P\]

2TPL, 15DBL, 4ADD.

2006 Doche–Imbert

generalized double-base chains:
e.g., compute \(314159 \cdot P\) as

\[2^{12} 3^3 3^3 P - 2^7 3^3 5^3 P\]

after precomputing \(3^0 P; 5^0 P; 7^0 P\).

3TPL, 13DBL, 6ADD.
To quickly triple $(X:Y:Z)$:
Three cubings for $R;S;V$.
For three choices of constants $(\alpha;\beta;\gamma)$ compute

$$(\alpha R + \beta S + \gamma V) \cdot (\alpha S + \beta V + \gamma R) \cdot (\alpha V + \beta R + \gamma S) = \alpha \beta \gamma d^3X^3 + (\alpha \beta^2 + \beta \gamma^2 + \gamma \alpha^2)Y^3 + (\beta \alpha^2 + \gamma \beta^2 + \alpha \gamma^2)Z^3 + (\alpha + \beta + \gamma)^3 RSV.$$ 

Also use $a (R + S + V)^3 = d^3 RSV$.
Solve for $dX^3;Y^3;Z^3$.

Are triplings useful?

2015 Kohel’s 11.2M
(4 cubings + 4 mults)
introduced this TPL idea with
$(\alpha, \beta, \gamma) = (1, 1, 1),$
$(\alpha, \beta, \gamma) = (1, -1, 0),$
$(\alpha, \beta, \gamma) = (1, 1, 0).$

New 10.8M (6 cubings)
makes faster choices
assuming fast primitive $\omega = \sqrt[3]{1}$:

$(\alpha, \beta, \gamma) = (1, 1, 1),$
$(\alpha, \beta, \gamma) = (1, \omega, \omega^2),$
$(\alpha, \beta, \gamma) = (1, \omega^2, \omega).$

2005 Dimitrov–Imbert–Mishra
“double-base chains”: e.g.,
compute $314159P$ as

$2^{15}3^2P + 2^{11}3^2P + 2^83^1P$
$+ 2^43^1P - 2^03^0P.$
2TPL, 15DBL, 4ADD.

2006 Doche–Imbert
generalized double-base chains:
 e.g., compute $314159P$ as

$2^{12}3^33P - 2^73^35P - 2^43^17P -$
$+ 2^03^0P,$
after precomputing $3P,5P,7P.$
3TPL, 13DBL, 6ADD.
2015 Kohel’s 11.2M
(4 cubings + 4 mults)
introduced this TPL idea with
\((\alpha, \beta, \gamma) = (1, 1, 1),\)
\((\alpha, \beta, \gamma) = (1, -1, 0),\)
\((\alpha, \beta, \gamma) = (1, 1, 0).\)

New 10.8M (6 cubings)
makes faster choices
assuming fast primitive \(\omega = \sqrt[3]{1}:\)
\((\alpha, \beta, \gamma) = (1, 1, 1),\)
\((\alpha, \beta, \gamma) = (1, \omega, \omega^2),\)
\((\alpha, \beta, \gamma) = (1, \omega^2, \omega).\)

2005 Dimitrov–Imbert–Mishra
“double-base chains”: e.g.,
compute \(314159P\) as
\(2^{15}3^2P + 2^{11}3^2P + 2^83^1P\)
\(+ 2^43^1P - 2^03^0P.\)
2TPL, 15DBL, 4ADD.

2006 Doche–Imbert
generalized double-base chains:
e.g., compute \(314159P\) as
\(2^{12}3^33P - 2^73^35P - 2^43^17P - 2^03^0P\)
after precomputing \(3P, 5P, 7P.\)
3TPL, 13DBL, 6ADD.
Kohel’s 11.2M (4 cubings + 4 mults) introduced this TPL idea with
\( (\chi;\gamma;\nu) = (1, 1, 1), \)
\( (\chi;\gamma;\nu) = (1, -1, 0), \)
\( (\chi;\gamma;\nu) = (1, 1, 0). \)

8M (6 cubings) makes faster choices assuming fast primitive \( \omega = \sqrt[3]{1}: \)
\( (\chi;\gamma;\nu) = (1, 1, 1), \)
\( (\chi;\gamma;\nu) = (1, \omega, \omega^2), \)
\( (\chi;\gamma;\nu) = (1, \omega^2, \omega). \)

Are triplings useful?

2005 Dimitrov–Imbert–Mishra “double-base chains”: e.g.,
compute \( 314159P \) as
\[ 2^{15}3^2P + 2^{11}3^2P + 2^83^1P + 2^43^1P - 2^03^0P. \]
2TPL, 15DBL, 4ADD.

2006 Doche–Imbert generalized double-base chains:
e.g., compute \( 314159P \) as
\[ 2^{12}3^3P - 2^73^5P - 2^43^17P - 2^03^0P \]
after precomputing \( 3P, 5P, 7P. \)
3TPL, 13DBL, 6ADD.

Not good for constant time.
Good for signature verification, factorization, math, etc.
Also need time to compute chain.
Good for scalars used many times.
2015 Kohel’s TPL idea with
\( (\Delta; \beta; \gamma) = (1; 1; 1), \)
\( (\Delta; \beta; \gamma) = (1; -1; 0), \)
\( (\Delta; \beta; \gamma) = (1; 1; 0). \)

New TPL makes faster choices assuming fast primitive \( \omega = 3 \sqrt{1}: \)
\( (\Delta; \beta; \gamma) = (1; 1; 1), \)
\( (\Delta; \beta; \gamma) = (1; \omega; \omega^2), \)
\( (\Delta; \beta; \gamma) = (1; \omega^2; \omega). \)

Are triplings useful?

2005 Dimitrov–Imbert–Mishra
“double-base chains”: e.g.,
compute 314159\( P \) as
\[ 2^{15}3^2P + 2^{11}3^2P + 2^83^1P \]
\[ + 2^43^1P - 2^03^0P. \]
2TPL, 15DBL, 4ADD.

2006 Doche–Imbert
generalized double-base chains:
e.g., compute 314159\( P \) as
\[ 2^{12}3^3P - 2^73^35P - 2^43^17P - 2^03^0P \]
after precomputing 3\( P \), 5\( P \), 7\( P \).
3TPL, 13DBL, 6ADD.

Not good for constant time.
Good for signature verification, factorization, math, etc.
Also need time to compute chain.
Good for scalars used many times.
Are triplings useful?

2005 Dimitrov–Imbert–Mishra
“double-base chains”: e.g.,
compute $314159P$ as
$2^{15}3^2P + 2^{11}3^2P + 2^83^1P$
$+ 2^43^1P - 2^03^0P$.
2TPL, 15DBL, 4ADD.

2006 Doche–Imbert
generalized double-base chains:
e.g., compute $314159P$ as
$2^{12}3^35P - 2^73^35P - 2^43^17P - 2^03^0P$
after precomputing $3P, 5P, 7P$.
3TPL, 13DBL, 6ADD.

Not good for constant time.
Good for signature verification, factorization, math, etc.

Also need time to compute chain.
Good for scalars used many times.
Are triplings useful?

2005 Dimitrov–Imbert–Mishra
“double-base chains”: e.g., compute $314159P$ as
$2^{15}3^2P + 2^{11}3^2P + 2^83^1P$
$+ 2^43^1P − 2^03^0P$.
2TPL, 15DBL, 4ADD.

2006 Doche–Imbert
generalized double-base chains:
e.g., compute $314159P$ as
$2^{12}3^33P − 2^73^35P − 2^43^17P − 2^03^0P$
after precomputing $3P, 5P, 7P$.
3TPL, 13DBL, 6ADD.

Not good for constant time.
Good for signature verification, factorization, math, etc.
Also need time to compute chain.
Good for scalars used many times.
Are triplings useful?

2005 Dimitrov–Imbert–Mishra
“double-base chains”: e.g.,
compute $314159P$ as
$2^{15}3^2P + 2^{11}3^2P + 2^83^1P$
$+ 2^43^1P − 2^03^0P$.
2TPL, 15DBL, 4ADD.

2006 Doche–Imbert
generalized double-base chains:
e.g., compute $314159P$ as
$2^{12}3^33P − 2^73^5P − 2^43^17P − 2^03^0P$
after precomputing $3P, 5P, 7P$.
3TPL, 13DBL, 6ADD.

Not good for constant time.
Good for signature verification,
factorization, math, etc.

Also need time to compute chain.
Good for scalars used many times.

Analysis+optimization from 2007
Bernstein–Birkner–Lange–Peters:
Double-base chains speed up
Weierstrass curves slightly:
$9.29\text{M}/\text{bit}$ for 256-bit scalars.

More savings for, e.g., Hessian:
$9.65\text{M}/\text{bit}$. Still not competitive.
Are triplings useful?

2005 Dimitrov–Imbert–Mishra
"double-base chains": e.g.,
compute $314159P$ as
$2^{15}3^2P + 2^83^1P$
$2^43^1P - 2^03^0P$.
2TPL, 15DBL, 4ADD.

2006 Doche–Imbert
generalized double-base chains:
compute $314159P$ as
$2^{12}3^3P - 2^73^5P - 2^43^17P - 2^03^0P$.
After precomputing $3P, 5P, 7P$.
3TPL, 13DBL, 6ADD.

Good for signature verification, factorization, math, etc.
Also need time to compute chain.
Good for scalars used many times.

Analysis+optimization from 2007
Bernstein–Birkner–Lange–Peters:
Double-base chains speed up
Weierstrass curves slightly:
$9.29M_{/bit}$ for 256-bit scalars.
More savings for, e.g., Hessian:
$9.65M_{/bit}$. Still not competitive.

Revisit conclusions using latest Hessian formulas,
latest double-base techniques.
Are triplings useful?
2005 Dimitrov–Imbert–Mishra
“double-base chains”: e.g.,
compute $314159P$ as
$2^{15}3^2P + 2^{11}3^2P + 2^{0}3^0P$
2TPL, 15DBL, 4ADD.

2006 Doche–Imbert
generalized double-base chains:
e.g., compute $314159P$ as
$2^{12}3^3P − 2^{0}3^0P$
3TPL, 13DBL, 6ADD.

Not good for constant time.
Good for signature verification,
factorization, math, etc.
Also need time to compute chain.
Good for scalars used many times.

Analysis+optimization from 2007
Bernstein–Birkner–Lange–Peters:
Double-base chains speed up
Weierstrass curves slightly:
$9.29M$/bit for 256-bit scalars.
More savings for, e.g., Hessian:

Revisit conclusions
using latest Hessian formulas,
latest double-base techniques.
Are triplings useful?

2005 Dimitrov–Imbert–Mishra

"double-base chains": e.g., compute 314159 \( P \) as
\[
2^{15} 3^2 P + 2^{11} 3^2 P + 2^8 3^1 P - 2^{10} 3^0 P.
\]

2TPL, 15DBL, 4ADD.

2006 Doche–Imbert

generalized double-base chains:
\[
\text{e.g., compute } 314159 P \text{ as } 2^{12} 3^3 P - 2^{7} 3^3 5 P - 2^4 3^1 7 P - 2^{0} 3^0 P.
\]

3TPL, 13DBL, 6ADD.

Not good for constant time.
Good for signature verification, factorization, math, etc.

Also need time to compute chain.
Good for scalars used many times.

Analysis+optimization from 2007
Bernstein–Birkner–Lange–Peters:

Double-base chains speed up
Weierstrass curves slightly:
9.29M/bit for 256-bit scalars.

More savings for, e.g., Hessian:

Revisit conclusions
using latest Hessian formulas,
latest double-base techniques.
Not good for constant time.
Good for signature verification, factorization, math, etc.

Also need time to compute chain.
Good for scalars used many times.

Analysis+optimization from 2007
Bernstein–Birkner–Lange–Peters:

Double-base chains speed up Weierstrass curves slightly:
9.29\text{M}/bit for 256-bit scalars.

More savings for, e.g., Hessian:
9.65\text{M}/bit. Still not competitive.

Revisit conclusions using latest Hessian formulas, latest double-base techniques.
Not good for constant time. Good for signature verification, factorization, math, etc.

Also need time to compute chain. Good for scalars used many times.

Analysis + optimization from 2007 Bernstein–Birkner–Lange–Peters:

Double-base chains speed up Weierstrass curves slightly:
9.29M/bit for 256-bit scalars.

More savings for, e.g., Hessian:

Revisit conclusions using latest Hessian formulas, latest double-base techniques.

New: 8.77M/bit for 256 bits.
Not good for constant time. Good for signature verification, factorization, math, etc.

Also need time to compute chain. Good for scalars used many times.

Analysis+optimization from 2007 Bernstein–Birkner–Lange–Peters:

Double-base chains speed up Weierstrass curves slightly:

9.29\text{M}/bit for 256-bit scalars.

More savings for, e.g., Hessian:

9.65\text{M}/bit. Still not competitive.

Revisit conclusions using latest Hessian formulas, latest double-base techniques.

New: 8.77\text{M}/bit for 256 bits.

Comparison to Weierstrass for 1-bit, 2-bit, . . . , 64-bit scalars:

Uses 2008 Doche–Habsieger “tree search” and some new improvements: e.g., account for costs of ADD, DBL, TPL.
Not good for constant time.

Good for signature verification, factorization, math, etc.

Also need time to compute chain.

Also good for scalars used many times.

Analysis + optimization from 2007 Bernstein–Birkner–Lange–Peters:

Double-base chains speed up Weierstrass curves slightly:

\[ 9 : 29 \text{ M}/\text{bit for 256-bit scalars.} \]

More savings for, e.g., Hessian:

\[ 9 : 65 \text{ M}/\text{bit. Still not competitive.} \]

Revisit conclusions using latest Hessian formulas, latest double-base techniques.

New: 8.77 \text{M}/\text{bit for 256 bits.}

Comparison to Weierstrass for 1-bit, 2-bit, \ldots, 64-bit scalars:

Summary:

Twisted Hessian curves solidly beat Weierstrass.

Chuengsatiansup talk tomorrow:
even better double-base chains from shortest paths in DAG—and also new Edwards speeds!
Revisit conclusions using latest Hessian formulas, latest double-base techniques.

New: 8.77M/bit for 256 bits.

Comparison to Weierstrass for 1-bit, 2-bit, …, 64-bit scalars:

Summary:
Twisted Hessian curves solidly beat Weierstrass.

Chuengsatiansup talk tomorrow: even better double-base chains from shortest paths in DAG—and also new Edwards speeds!
Revisit conclusions using latest Hessian formulas, latest double-base techniques.

New: $8.77 \text{M}$/bit for 256 bits.

Comparison to Weierstrass for 1-bit, 2-bit, ..., 64-bit scalars:

Uses 2008 Doche–Habsieger “tree search” and some new improvements: e.g., account for costs of ADD, DBL, TPL.

Summary:
Twisted Hessian curves solidly beat Weierstrass.

Chuangsatsiansup talk tomorrow: even better double-base chains from shortest paths in DAG—and also new Edwards speeds!
Revisit conclusions using latest Hessian formulas, latest double-base techniques.

New: $8.77 \text{M}/\text{bit}$ for $256$ bits.

Comparison to Weierstrass for 1-bit, 2-bit, $\ldots$, 64-bit scalars:

![Graph](image)

Summary:
Twisted Hessian curves solidly beat Weierstrass.

Uses 2008 Doche–Habsieger “tree search” and some new improvements: e.g., account for costs of ADD, DBL, TPL.

Chuengsatiansup talk tomorrow: even better double-base chains from shortest paths in DAG—and also new Edwards speeds!