Elliptic curves

Tanja Lange
Technische Universiteit Eindhoven

with some slides by
Daniel J. Bernstein

Diffie-Hellman key exchange

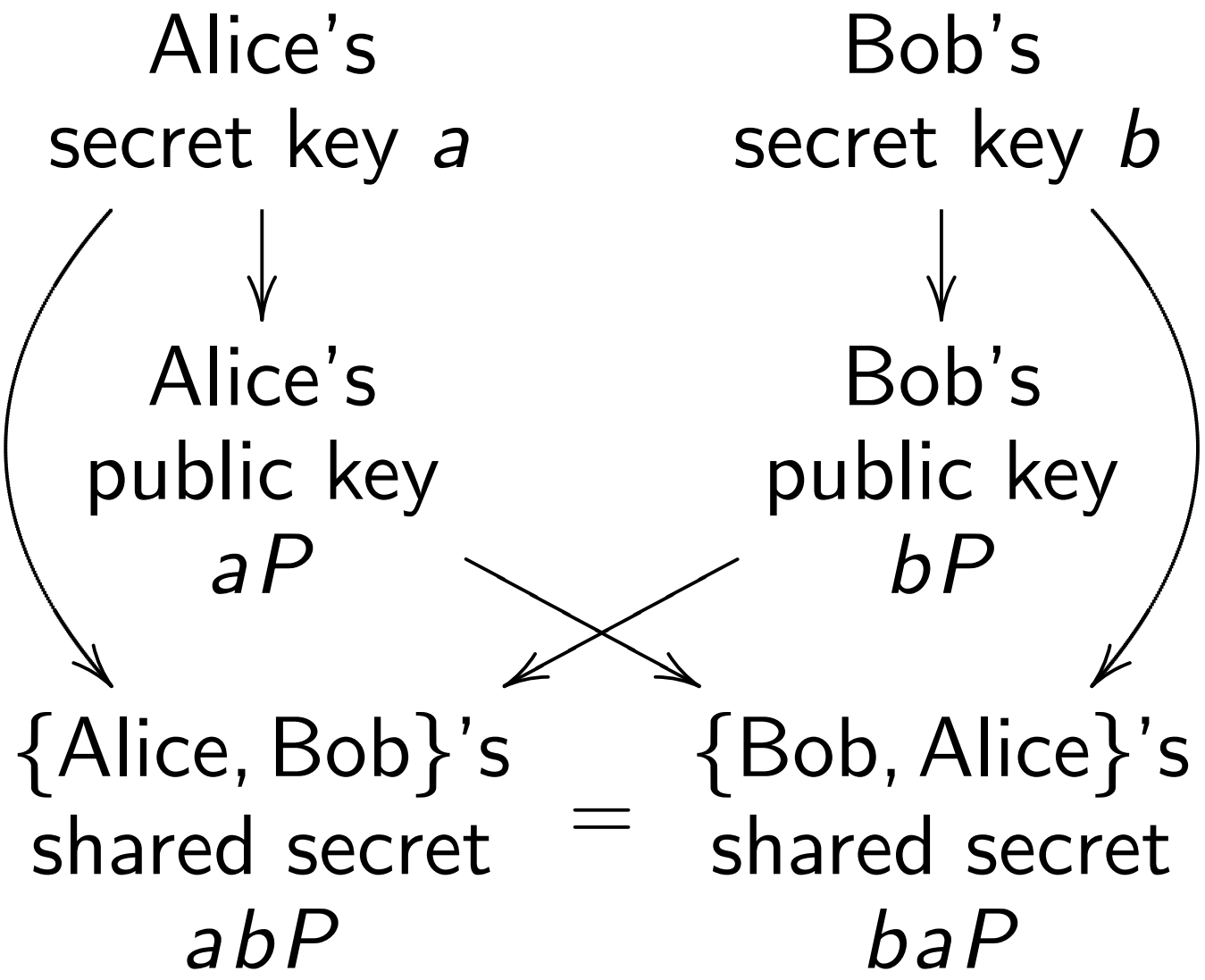
Pick some *generator*.

Diffie-Hellman key exchange

Pick some *generator*.

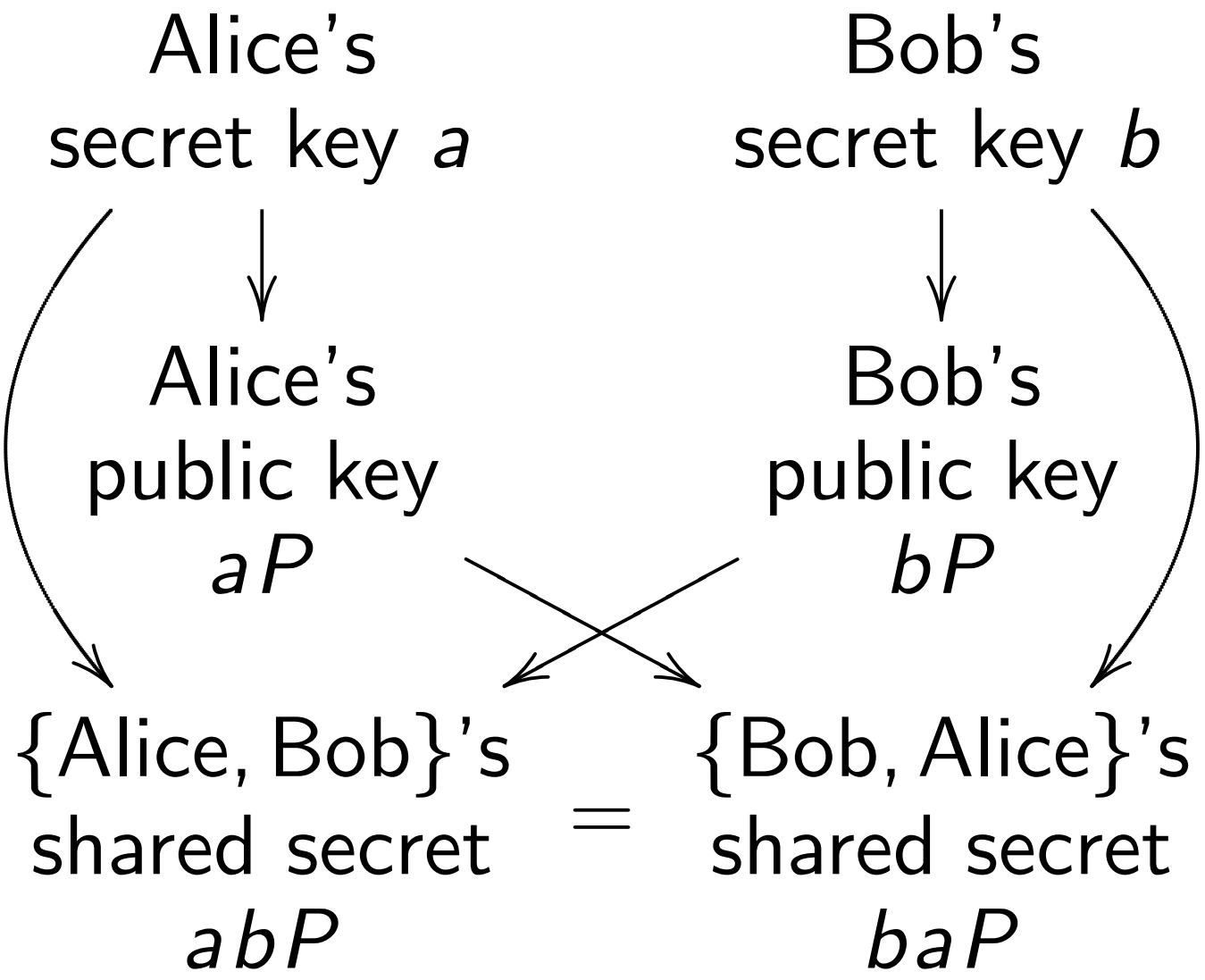
Diffie-Hellman key exchange

Pick some *generator* P,
i.e. some group element
(using additive notation here).



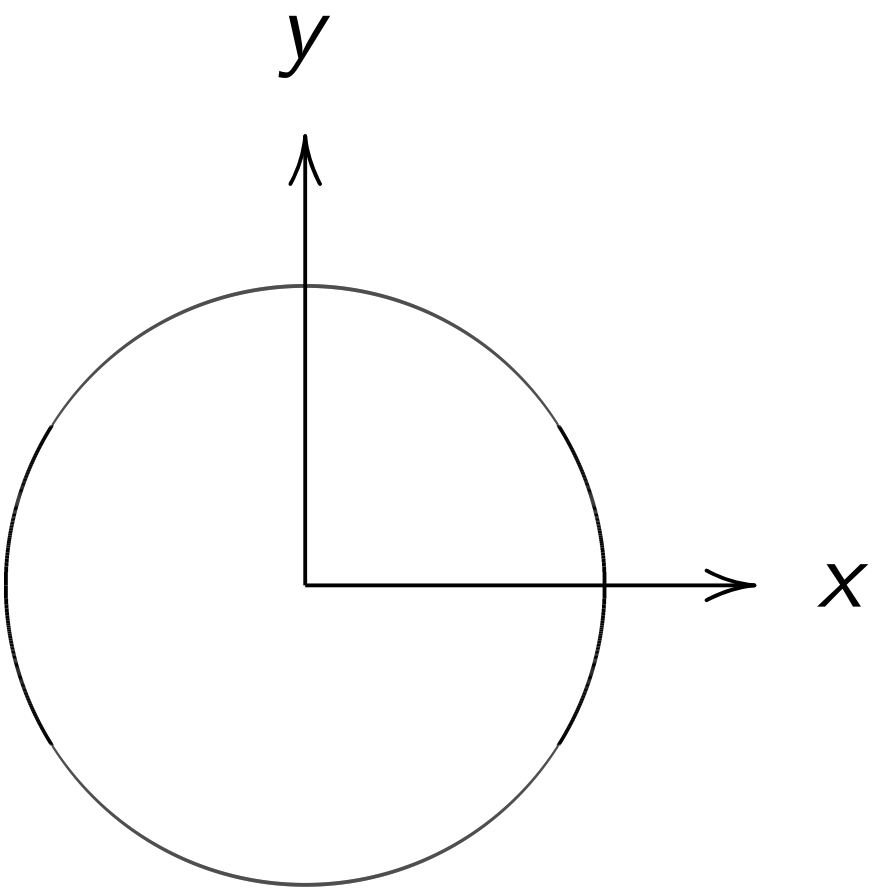
Diffie-Hellman key exchange

Pick some *generator* P,
i.e. some group element
(using additive notation here).



What does P look like &
how to compute $P + Q$?

The clock



This is the curve $x^2 + y^2 = 1$.

Warning:

This is *not* an elliptic curve.

"Elliptic curve" $\neq$ "ellipse."

Examples of points on this curve:

Examples of points on this curve:

$(0, 1) =$ “12:00”.

Examples of points on this curve:

$(0, 1) =$ “12:00”.

$(0, -1) =$ “6:00”.

Examples of points on this curve:

$(0, 1) =$ “12:00”.

$(0, -1) =$ “6:00”.

$(1, 0) =$ “3:00”.

Examples of points on this curve:

$(0, 1) =$ “12:00”.

$(0, -1) =$ “6:00”.

$(1, 0) =$ “3:00”.

$(-1, 0) =$ “9:00”.

Examples of points on this curve:

$(0, 1) =$ “12:00”.

$(0, -1) =$ “6:00”.

$(1, 0) =$ “3:00”.

$(-1, 0) =$ “9:00”.

$(\sqrt{3/4}, 1/2) =$

Examples of points on this curve:

$(0, 1) =$ “12:00”.

$(0, -1) =$ “6:00”.

$(1, 0) =$ “3:00”.

$(-1, 0) =$ “9:00”.

$(\sqrt{3/4}, 1/2) =$ “2:00”.

Examples of points on this curve:

$(0, 1) =$ “12:00”.

$(0, -1) =$ “6:00”.

$(1, 0) =$ “3:00”.

$(-1, 0) =$ “9:00”.

$(\sqrt{3/4}, 1/2) =$ “2:00”.

$(1/2, -\sqrt{3/4}) =$

Examples of points on this curve:

$(0, 1) =$ “12:00”.

$(0, -1) =$ “6:00”.

$(1, 0) =$ “3:00”.

$(-1, 0) =$ “9:00”.

$(\sqrt{3/4}, 1/2) =$ “2:00”.

$(1/2, -\sqrt{3/4}) =$ “5:00”.

$(-1/2, -\sqrt{3/4}) =$

Examples of points on this curve:

$(0, 1) =$ “12:00”.

$(0, -1) =$ “6:00”.

$(1, 0) =$ “3:00”.

$(-1, 0) =$ “9:00”.

$(\sqrt{3/4}, 1/2) =$ “2:00”.

$(1/2, -\sqrt{3/4}) =$ “5:00”.

$(-1/2, -\sqrt{3/4}) =$ “7:00”.

Examples of points on this curve:

$(0, 1) =$ “12:00”.

$(0, -1) =$ “6:00”.

$(1, 0) =$ “3:00”.

$(-1, 0) =$ “9:00”.

$(\sqrt{3/4}, 1/2) =$ “2:00”.

$(1/2, -\sqrt{3/4}) =$ “5:00”.

$(-1/2, -\sqrt{3/4}) =$ “7:00”.

$(\sqrt{1/2}, \sqrt{1/2}) =$ “1:30”.

$(3/5, 4/5)$. $(-3/5, 4/5)$.

Examples of points on this curve:

$(0, 1) =$ "12:00".

$(0, -1) =$ "6:00".

$(1, 0) =$ "3:00".

$(-1, 0) =$ "9:00".

$(\sqrt{3/4}, 1/2) =$ "2:00".

$(1/2, -\sqrt{3/4}) =$ "5:00".

$(-1/2, -\sqrt{3/4}) =$ "7:00".

$(\sqrt{1/2}, \sqrt{1/2}) =$ "1:30".

$(3/5, 4/5)$. $(-3/5, 4/5)$.

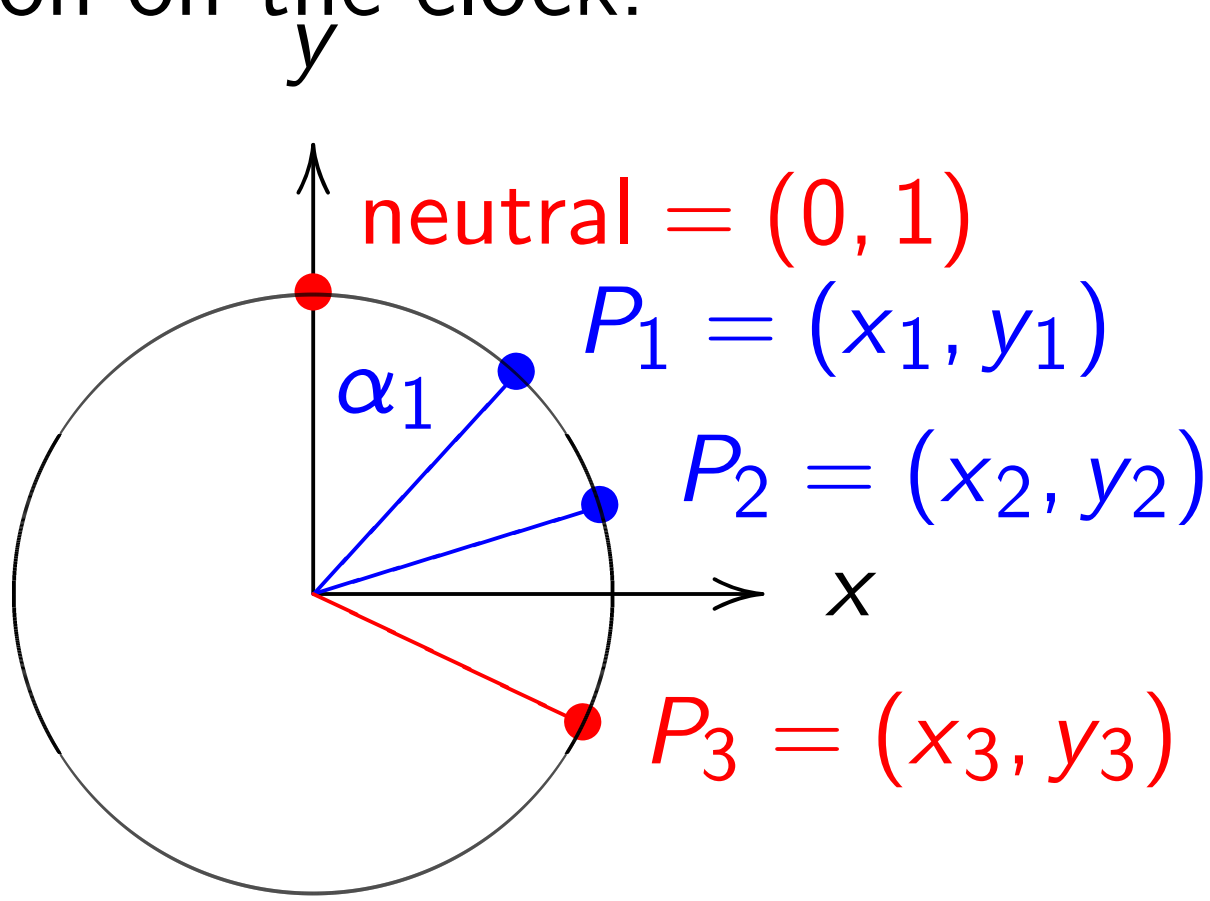
$(3/5, -4/5)$. $(-3/5, -4/5)$.

$(4/5, 3/5)$. $(-4/5, 3/5)$.

$(4/5, -3/5)$. $(-4/5, -3/5)$.

Many more.

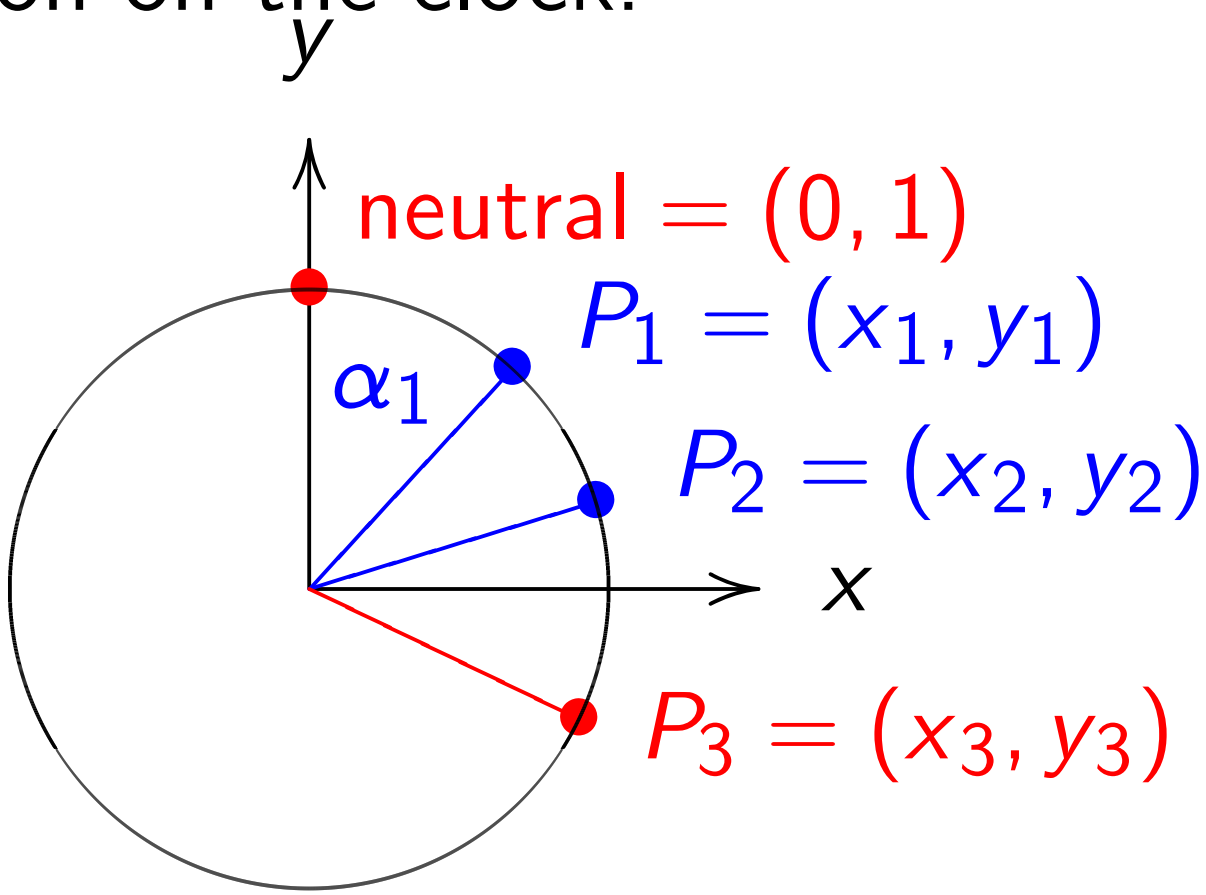
Addition on the clock:



$x^2 + y^2 = 1$, parametrized by

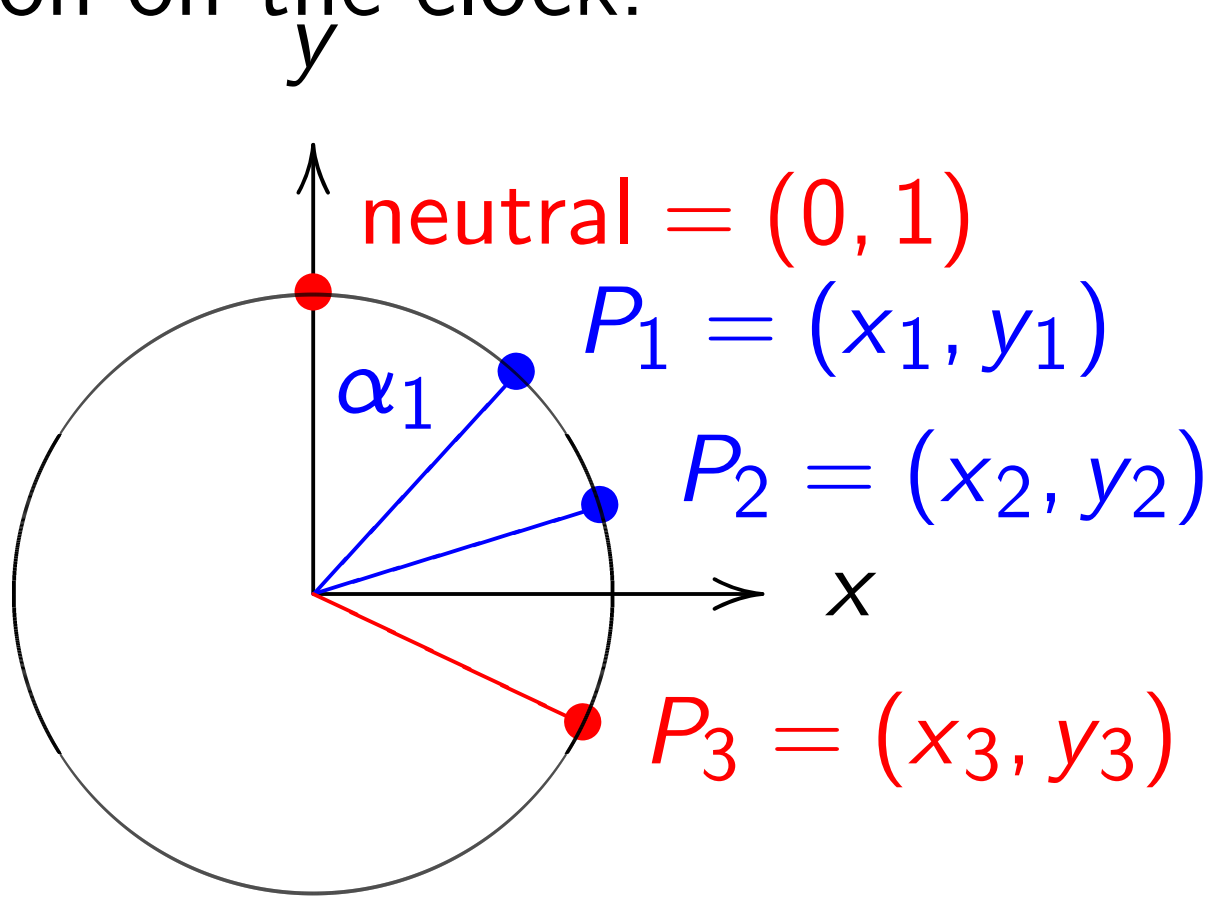
$x = \sin\alpha, \quad y = \cos\alpha.$

Addition on the clock:



$x^2 + y^2 = 1$, parametrized by $x = \sin\alpha, \quad y = \cos\alpha$. Recall $(\sin(\alpha_1 + \alpha_2), \cos(\alpha_1 + \alpha_2)) =$

Addition on the clock:

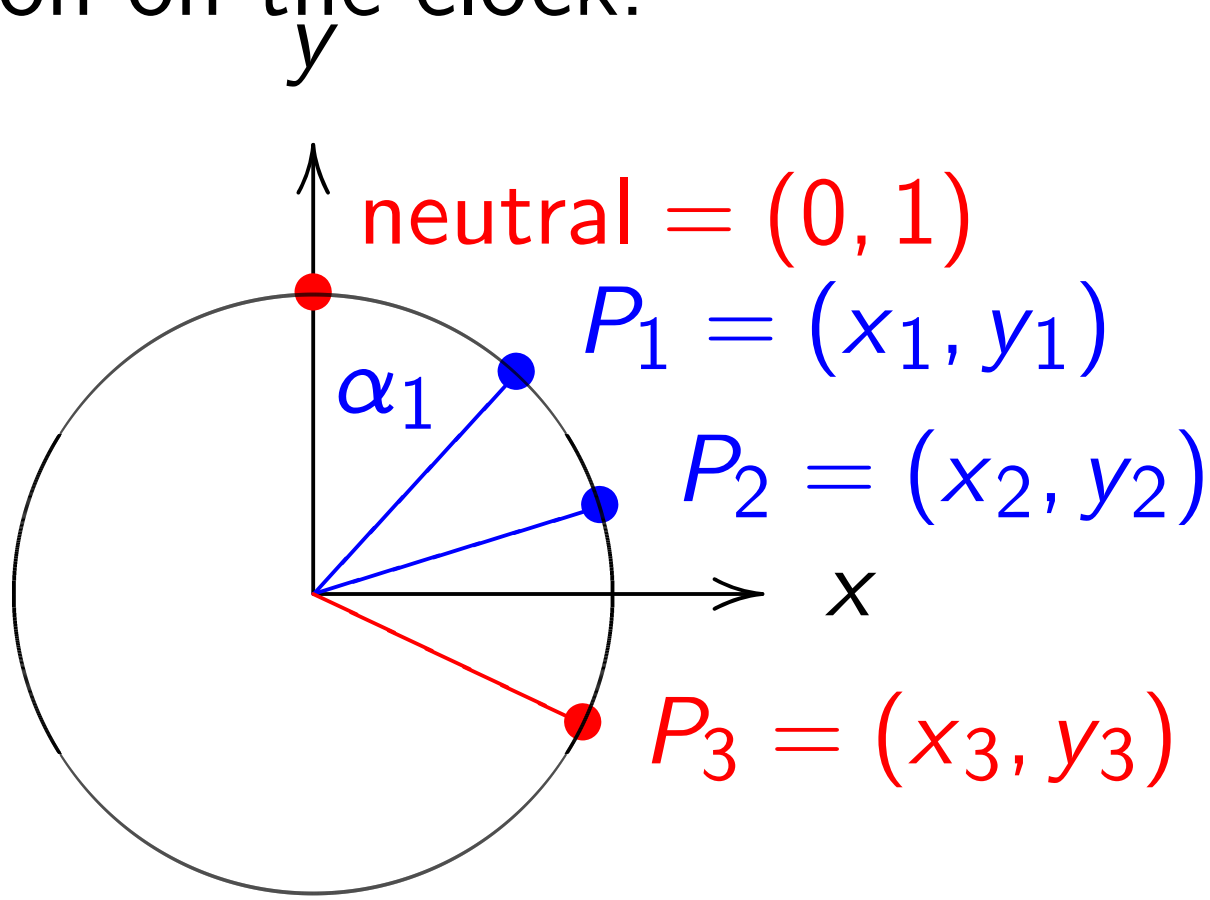


$x^2 + y^2 = 1$, parametrized by $x = \sin\alpha, \quad y = \cos\alpha$. Recall

$(\sin(\alpha_1 + \alpha_2), \cos(\alpha_1 + \alpha_2)) =$

$(\sin\alpha_1 \cos\alpha_2 + \cos\alpha_1 \sin\alpha_2,$

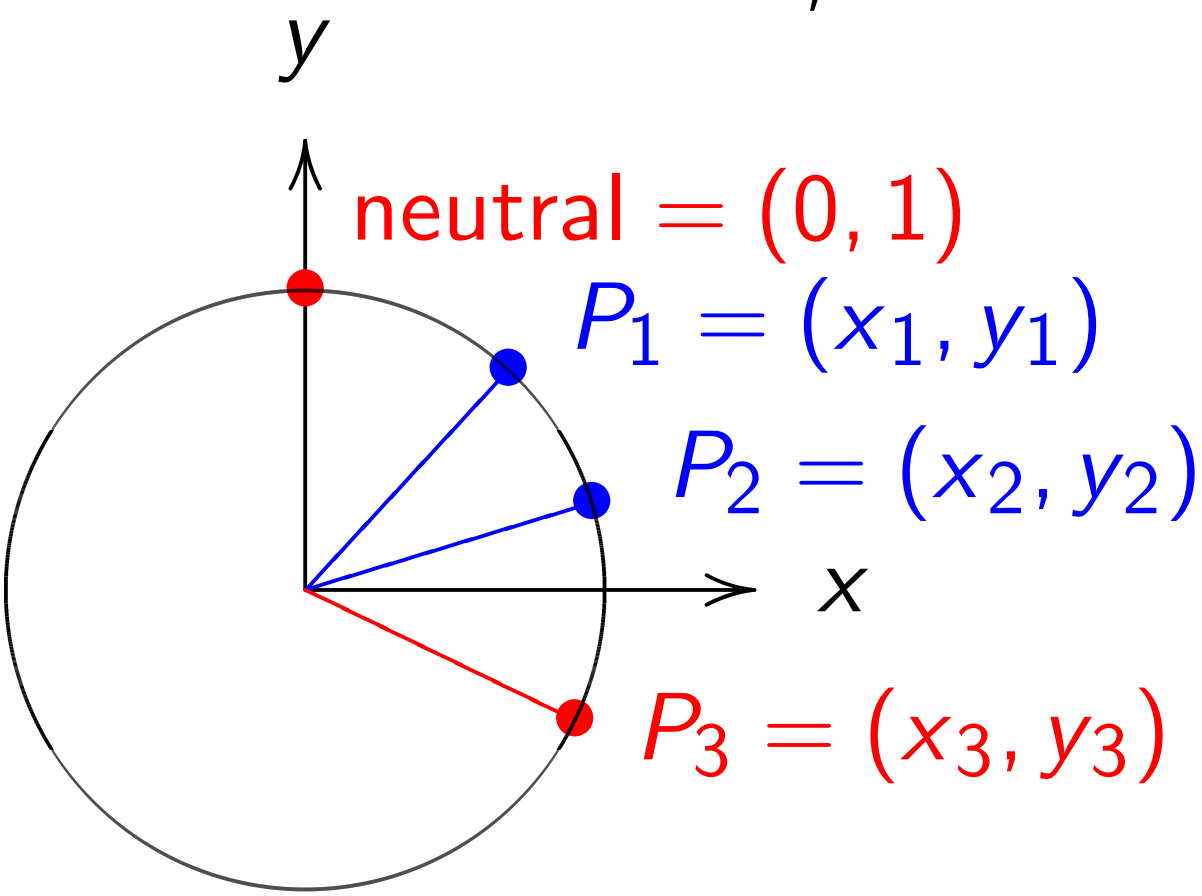
Addition on the clock:



$x^2 + y^2 = 1$, parametrized by $x = \sin\alpha, \quad y = \cos\alpha$. Recall

$$(\sin(\alpha_1 + \alpha_2), \cos(\alpha_1 + \alpha_2)) =$$
$$(\sin\alpha_1 \cos\alpha_2 + \cos\alpha_1 \sin\alpha_2,$$
$$\cos\alpha_1 \cos\alpha_2 - \sin\alpha_1 \sin\alpha_2).$$

Clock addition without sin, cos:



Use Cartesian coordinates for addition. Addition formula for the clock $x^2 + y^2 = 1$:

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

$= (x_1y_2 + y_1x_2, y_1y_2 - x_1x_2)$.

Note $(x_1, y_1) + (-x_1, y_1) = (0, 1)$.

$kP = \underbrace{P + P + \cdots + P}_{k \text{ copies}}$ for $k \geq 0$.

Examples of clock addition:

$$\text{“2:00”} + \text{“5:00”}$$
$$= (\sqrt{3/4}, 1/2) + (1/2, -\sqrt{3/4})$$
$$= (-1/2, -\sqrt{3/4}) = \text{“7:00”}.$$

$$\text{“5:00”} + \text{“9:00”}$$
$$= (1/2, -\sqrt{3/4}) + (-1, 0)$$
$$= (\sqrt{3/4}, 1/2) = \text{“2:00”}.$$

$$2\left(\frac{3}{5}, \frac{4}{5}\right) = \left(\frac{24}{25}, \frac{7}{25}\right).$$

Examples of clock addition:

$$\text{“2:00”} + \text{“5:00”}$$
$$= (\sqrt{3/4}, 1/2) + (1/2, -\sqrt{3/4})$$
$$= (-1/2, -\sqrt{3/4}) = \text{“7:00”}.$$

$$\text{“5:00”} + \text{“9:00”}$$
$$= (1/2, -\sqrt{3/4}) + (-1, 0)$$
$$= (\sqrt{3/4}, 1/2) = \text{“2:00”}.$$

$$2\left(\frac{3}{5}, \frac{4}{5}\right) = \left(\frac{24}{25}, \frac{7}{25}\right).$$

$$3\left(\frac{3}{5}, \frac{4}{5}\right) = \left(\frac{117}{125}, \frac{-44}{125}\right).$$

Examples of clock addition:

$$\text{“2:00”} + \text{“5:00”}$$
$$= (\sqrt{3/4}, 1/2) + (1/2, -\sqrt{3/4})$$
$$= (-1/2, -\sqrt{3/4}) = \text{“7:00”}.$$

$$\text{“5:00”} + \text{“9:00”}$$
$$= (1/2, -\sqrt{3/4}) + (-1, 0)$$
$$= (\sqrt{3/4}, 1/2) = \text{“2:00”}.$$

$$2\left(\frac{3}{5}, \frac{4}{5}\right) = \left(\frac{24}{25}, \frac{7}{25}\right).$$

$$3\left(\frac{3}{5}, \frac{4}{5}\right) = \left(\frac{117}{125}, \frac{-44}{125}\right).$$

$$4\left(\frac{3}{5}, \frac{4}{5}\right) = \left(\frac{336}{625}, \frac{-527}{625}\right).$$

Examples of clock addition:

$$\text{“2:00”} + \text{“5:00”}$$
$$= (\sqrt{3/4}, 1/2) + (1/2, -\sqrt{3/4})$$
$$= (-1/2, -\sqrt{3/4}) = \text{“7:00”}.$$

$$\text{“5:00”} + \text{“9:00”}$$
$$= (1/2, -\sqrt{3/4}) + (-1, 0)$$
$$= (\sqrt{3/4}, 1/2) = \text{“2:00”}.$$

$$2\left(\frac{3}{5}, \frac{4}{5}\right) = \left(\frac{24}{25}, \frac{7}{25}\right).$$

$$3\left(\frac{3}{5}, \frac{4}{5}\right) = \left(\frac{117}{125}, \frac{-44}{125}\right).$$

$$4\left(\frac{3}{5}, \frac{4}{5}\right) = \left(\frac{336}{625}, \frac{-527}{625}\right).$$

$$(x_1, y_1) + (0, 1) =$$

Examples of clock addition:

$$\text{“2:00”} + \text{“5:00”}$$
$$= (\sqrt{3/4}, 1/2) + (1/2, -\sqrt{3/4})$$
$$= (-1/2, -\sqrt{3/4}) = \text{“7:00”}.$$

$$\text{“5:00”} + \text{“9:00”}$$
$$= (1/2, -\sqrt{3/4}) + (-1, 0)$$
$$= (\sqrt{3/4}, 1/2) = \text{“2:00”}.$$

$$2\left(\frac{3}{5}, \frac{4}{5}\right) = \left(\frac{24}{25}, \frac{7}{25}\right).$$

$$3\left(\frac{3}{5}, \frac{4}{5}\right) = \left(\frac{117}{125}, \frac{-44}{125}\right).$$

$$4\left(\frac{3}{5}, \frac{4}{5}\right) = \left(\frac{336}{625}, \frac{-527}{625}\right).$$

$$(x_1, y_1) + (0, 1) = (x_1, y_1).$$

Examples of clock addition:

$$\text{“2:00”} + \text{“5:00”}$$
$$= (\sqrt{3/4}, 1/2) + (1/2, -\sqrt{3/4})$$
$$= (-1/2, -\sqrt{3/4}) = \text{“7:00”}.$$

$$\text{“5:00”} + \text{“9:00”}$$
$$= (1/2, -\sqrt{3/4}) + (-1, 0)$$
$$= (\sqrt{3/4}, 1/2) = \text{“2:00”}.$$

$$2\left(\frac{3}{5}, \frac{4}{5}\right) = \left(\frac{24}{25}, \frac{7}{25}\right).$$

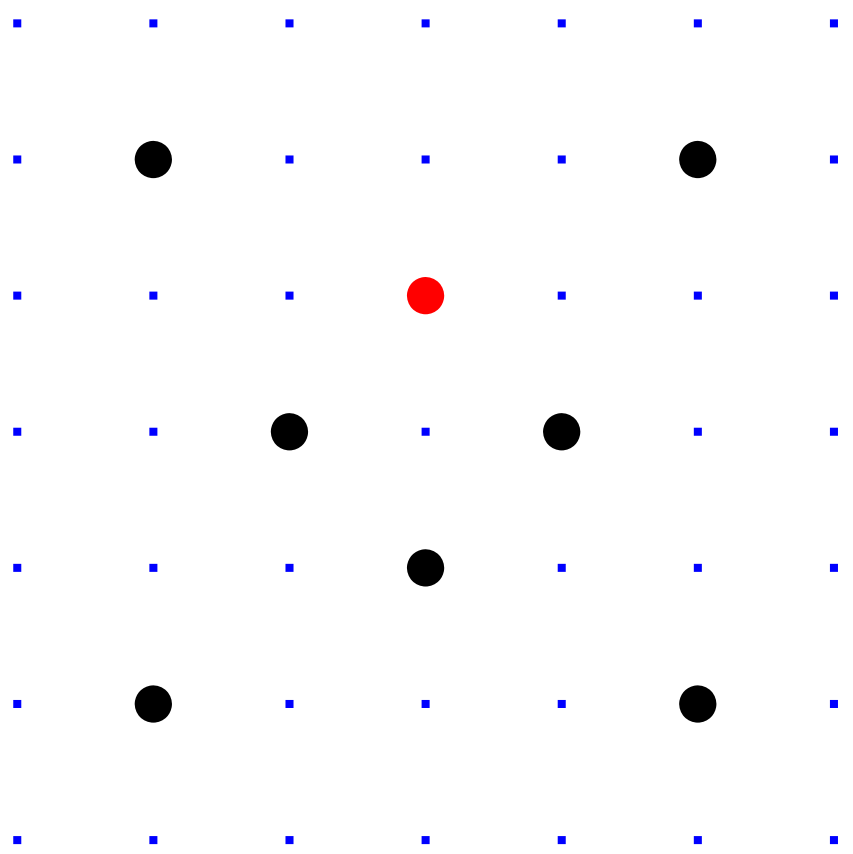
$$3\left(\frac{3}{5}, \frac{4}{5}\right) = \left(\frac{117}{125}, \frac{-44}{125}\right).$$

$$4\left(\frac{3}{5}, \frac{4}{5}\right) = \left(\frac{336}{625}, \frac{-527}{625}\right).$$

$$(x_1, y_1) + (0, 1) = (x_1, y_1).$$

$$-(x_1, y_1) = (-x_1, y_1).$$

Clocks over finite fields



$\mathrm{Clock}(\mathbf{F}_7) =$

$\{(x, y) \in \mathbf{F}_7 \times \mathbf{F}_7 : x^2 + y^2 = 1\}.$

Here $\mathbf{F}_7 = \{0, 1, 2, 3, 4, 5, 6\}$

$= \{0, 1, 2, 3, -3, -2, -1\}$

with $+, -, \times$ modulo 7.

E.g. $2 \cdot 5 = 3$ and $3/2 = 5$ in $\mathbf{F}_7$.

```
>>> for x in range(7):
...   for y in range(7):
...     if (x*x+y*y) % 7 == 1:
...       print (x,y)
...
(0, 1)
(0, 6)
(1, 0)
(2, 2)
(2, 5)
(5, 2)
(5, 5)
(6, 0)
>>>
```

```
>>> class F7:
...   def __init__(self,x):
...     self.int = x % 7
...   def __str__(self):
...     return str(self.int)
...   __repr__ = __str__
...
>>> print F7(2)
2
>>> print F7(6)
6
>>> print F7(7)
0
>>> print F7(10)
3
```

```
>>> F7.__eq__ = lambda a,b: \
...    a.int == b.int
>>>
>>> print F7(7) == F7(0)
True
>>> print F7(10) == F7(3)
True
>>> print F7(-3) == F7(4)
True
>>> print F7(0) == F7(1)
False
>>> print F7(0) == F7(2)
False
>>> print F7(0) == F7(3)
False
```

```
>>> F7.__add__ = lambda a,b: \
...   F7(a.int + b.int)
>>> F7.__sub__ = lambda a,b: \
...   F7(a.int - b.int)
>>> F7.__mul__ = lambda a,b: \
...   F7(a.int * b.int)
>>>
>>> print F7(2) + F7(5)
0
>>> print F7(2) - F7(5)
4
>>> print F7(2) * F7(5)
3
>>>
```

Larger example: Clock($\mathbf{F}_{1000003}$).

```
p = 1000003

class Fp:

  ...


def clockadd(P1,P2):

  x1,y1 = P1

  x2,y2 = P2

  x3 = x1*y2+y1*x2

  y3 = y1*y2-x1*x2

  return x3,y3
```

```
>>> P = (Fp(1000),Fp(2))
>>> P2 = clockadd(P,P)
>>> print P2
(4000, 7)
>>> P3 = clockadd(P2,P)
>>> print P3
(15000, 26)
>>> P4 = clockadd(P3,P)
>>> P5 = clockadd(P4,P)
>>> P6 = clockadd(P5,P)
>>> print P6
(780000, 1351)
>>> print clockadd(P3,P3)
(780000, 1351)
>>>
```

```
>>> def scalarmult(n,P):
...     if n == 0: \
...     return (Fp(0),Fp(1))
...     if n == 1: return P
...     Q = scalarmult(n//2,P)
...     Q = clockadd(Q,Q)
...     if n % 2: Q = clockadd(P,Q)
...     return Q
...
>>> n = oursixdigitsecret
>>> scalarmult(n,P)
(947472, 736284)
>>>
```

Can you figure out our secret n?

Clock cryptography

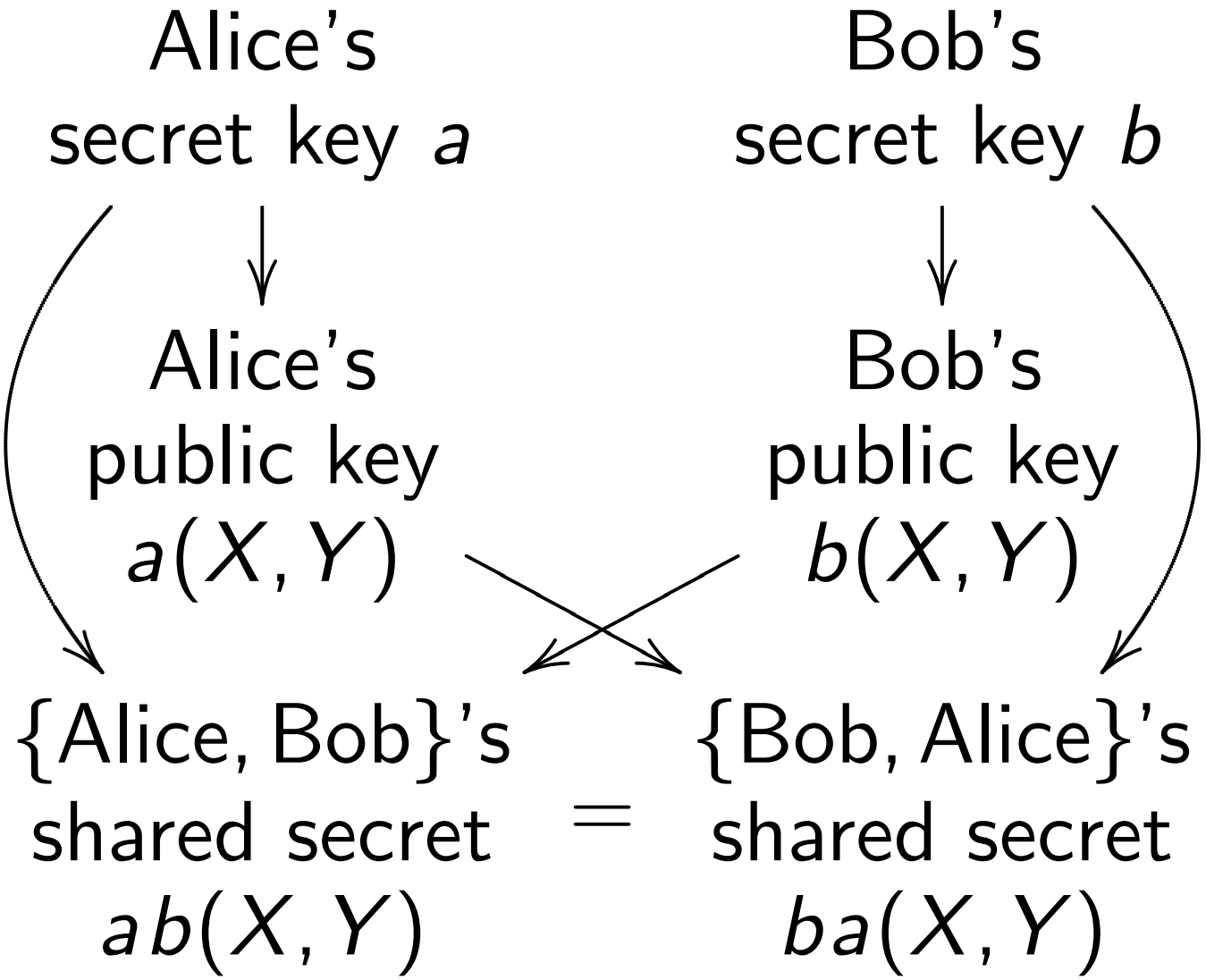
The "Clock Diffie–Hellman protocol":

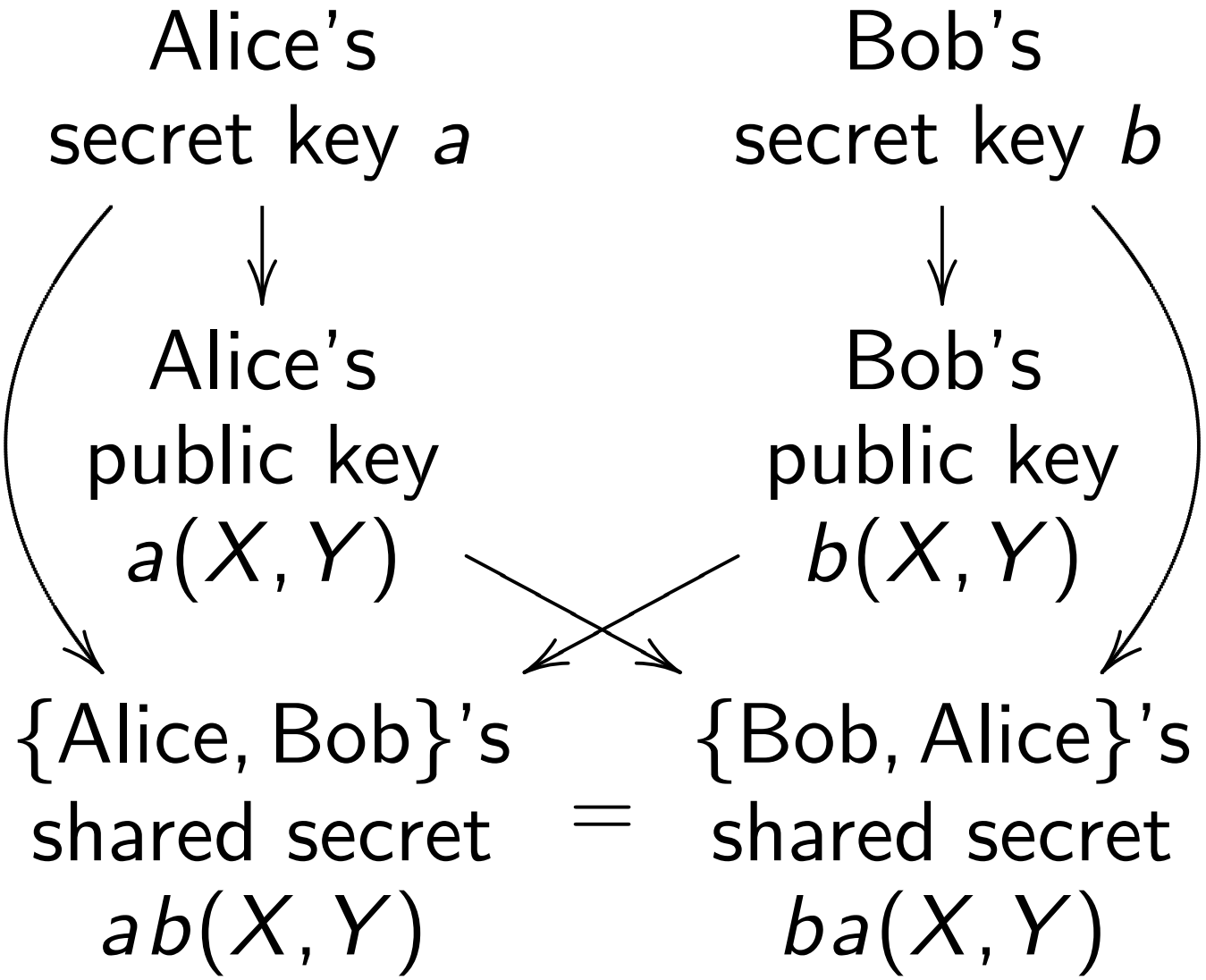
Standardize large prime p & **base point** $(x, y) \in \mathrm{Clock}(\mathbf{F}_p)$.

Alice chooses big secret a, computes her public key $a(x, y)$.

Bob chooses big secret b, computes his public key $b(x, y)$.

Alice computes $a(b(x, y))$.
Bob computes $b(a(x, y))$.
They use this shared secret to encrypt with AES-GCM etc.


Alice's
secret key a
Bob's
secret key b
Alice's
public key
$a(X, Y)$
Bob's
public key
$b(X, Y)$
{Alice, Bob}'s
shared secret
$ab(X, Y)$
=
{Bob, Alice}'s
shared secret
$ba(X, Y)$



Warning #1:

Many p are unsafe!

Warning #2:

Clocks aren't elliptic!

To match RSA-3072 security need $p \approx 2^{1536}$.

Warning #3:
Attacker sees more than
public keys $a(x, y)$ and $b(x, y)$.

Attacker sees how much *time*
Alice uses to compute $a(b(x, y))$.
Often attacker can see time for
each operation performed by
Alice, not just total time.
This reveals secret scalar a.

Break by timing attacks, e.g.,
2011 Brumley–Tuveri.

Warning #3:
Attacker sees more than
public keys $a(x, y)$ and $b(x, y)$.

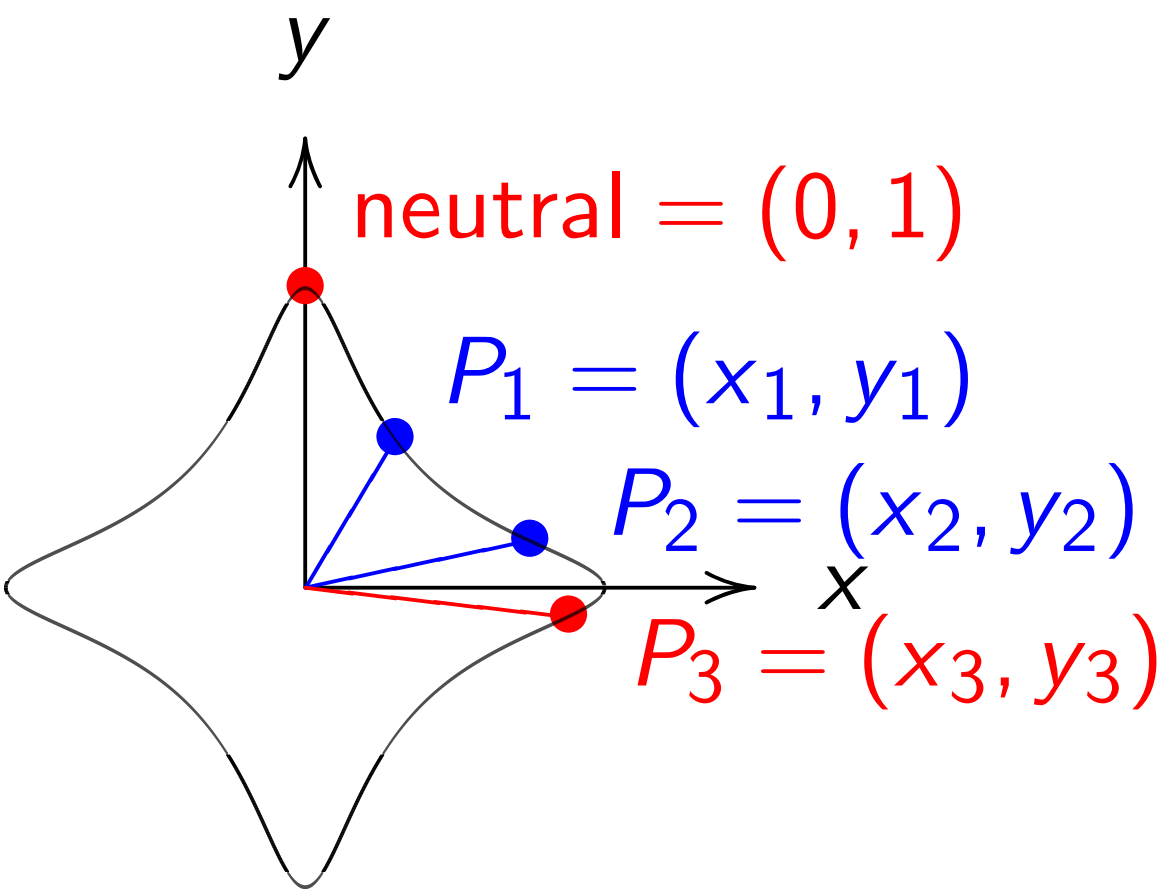
Attacker sees how much *time*
Alice uses to compute $a(b(x, y))$.
Often attacker can see time for
each operation performed by
Alice, not just total time.
This reveals secret scalar a.

Break by timing attacks, e.g.,
2011 Brumley–Tuveri.

Fix: **constant-time** code,
performing same operations
no matter what scalar is.

Addition on an Edwards curve

Change the curve on which Alice and Bob work.

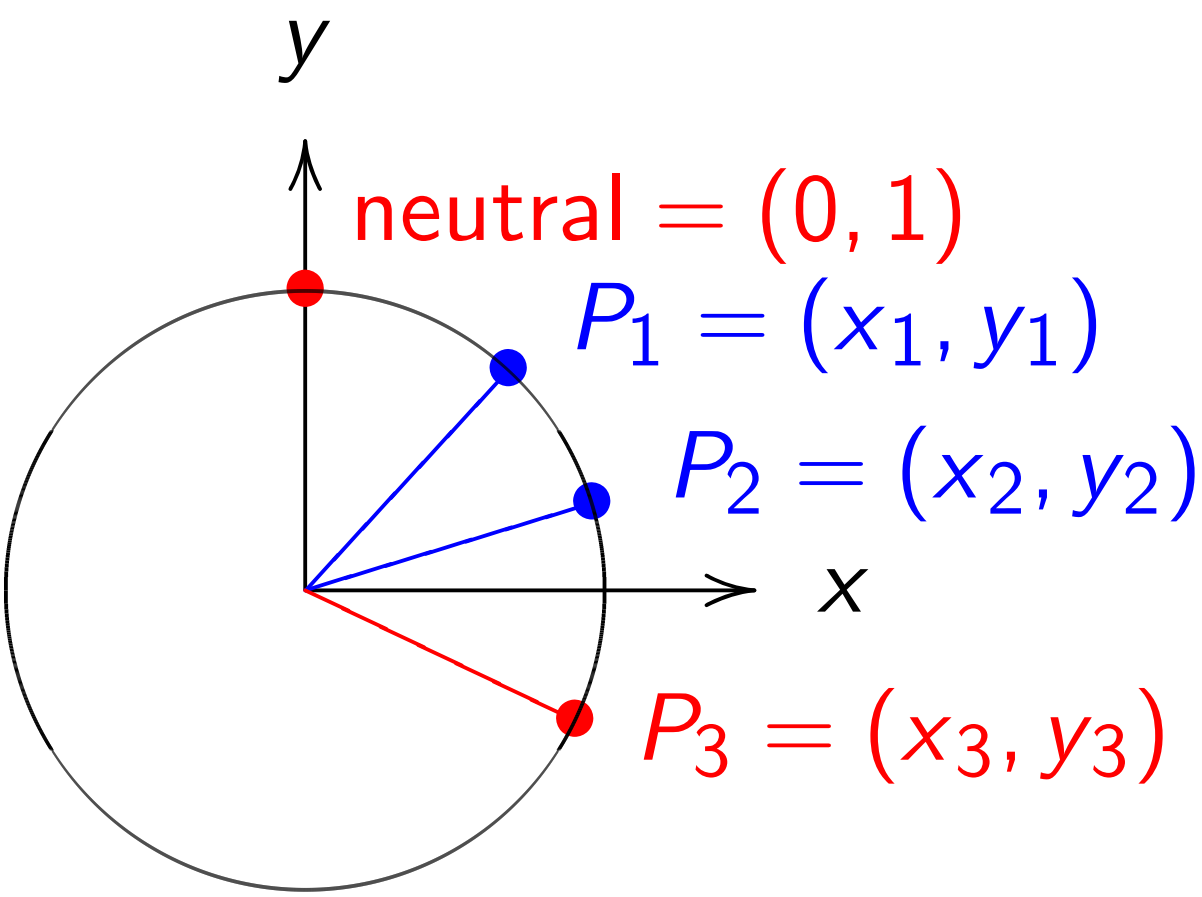


$x^2 + y^2 = 1 - 30x^2y^2$.

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

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

The clock again, for comparison:



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

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

$(x_1y_2 + y_1x_2,$

$y_1y_2 - x_1x_2)$.

"Hey, there were divisions
in the Edwards addition law!
What if the denominators are 0?"
Answer: They aren't!
If $x_i = 0$ or $y_i = 0$ then
$1 \pm 30x_1x_2y_1y_2 = 1 \neq 0$.
If $x^2 + y^2 = 1 - 30x^2y^2$
then $30x^2y^2 < 1$
so $\sqrt{30}\ |xy| < 1$.

"Hey, there were divisions
in the Edwards addition law!
What if the denominators are 0?"
Answer: They aren't!
If $x_i = 0$ or $y_i = 0$ then
$1 \pm 30x_1x_2y_1y_2 = 1 \neq 0$.
If $x^2 + y^2 = 1 - 30x^2y^2$
then $30x^2y^2 < 1$
so $\sqrt{30}\,|xy| < 1$.

If $x_1^2 + y_1^2 = 1 - 30x_1^2y_1^2$
and $x_2^2 + y_2^2 = 1 - 30x_2^2y_2^2$
then $\sqrt{30}\,|x_1y_1| < 1$
and $\sqrt{30}\,|x_2y_2| < 1$

"Hey, there were divisions
in the Edwards addition law!
What if the denominators are 0?"
Answer: They aren't!
If $x_i = 0$ or $y_i = 0$ then
$1 \pm 30x_1x_2y_1y_2 = 1 \neq 0$.
If $x^2 + y^2 = 1 - 30x^2y^2$
then $30x^2y^2 < 1$
so $\sqrt{30}\,|xy| < 1$.

If $x_1^2 + y_1^2 = 1 - 30x_1^2y_1^2$
and $x_2^2 + y_2^2 = 1 - 30x_2^2y_2^2$
then $\sqrt{30}\,|x_1y_1| < 1$
and $\sqrt{30}\,|x_2y_2| < 1$
so $30\,|x_1y_1x_2y_2| < 1$
so $1 \pm 30x_1x_2y_1y_2 > 0$.

The Edwards addition law
$(x_1, y_1) + (x_2, y_2) =$
$((x_1y_2+y_1x_2)/(1-30x_1x_2y_1y_2),$
$(y_1y_2-x_1x_2)/(1+30x_1x_2y_1y_2))$
is a group law for the curve
$x^2 + y^2 = 1 - 30x^2y^2$.

Some calculation required:

addition result is on curve;

addition law is associative.

Other parts of proof are easy:

addition law is commutative;

$(0, 1)$ is neutral element;

$(x_1, y_1) + (-x_1, y_1) = (0, 1)$.

Edwards curves mod p

Choose an odd prime p.

Choose a *non-square* $d \in \mathbf{F}_p$.

$\{(x, y) \in \mathbf{F}_p \times \mathbf{F}_p :$

$$x^2 + y^2 = 1 + dx^2y^2\}$$

is a "complete Edwards curve".

Roughly $p + 1$ pairs (x, y).

```
def edwardsadd(P1,P2):
  x1,y1 = P1
  x2,y2 = P2
  x3 = (x1*y2+y1*x2)/ \
  (1+d*x1*x2*y1*y2)
  y3 = (y1*y2-x1*x2)/ \
  (1-d*x1*x2*y1*y2)
  return x3,y3
```

Denominators are never 0.

But need different proof;

"$x^2 + y^2 > 0$" doesn't work.

Denominators are never 0.

But need different proof;

"$x^2 + y^2 > 0$" doesn't work.

Answer: Can prove that

the denominators are never 0.

Addition law is **complete**.

Denominators are never 0.

But need different proof;
"$x^2 + y^2 > 0$" doesn't work.

Answer: Can prove that
the denominators are never 0.
Addition law is **complete**.

This proof relies on
choosing *non-square* d.

Denominators are never 0.
But need different proof;
"$x^2 + y^2 > 0$" doesn't work.

Answer: Can prove that
the denominators are never 0.
Addition law is **complete**.

This proof relies on
choosing *non-square* d.

If we instead choose square d:
curve is still elliptic, and
addition *seems to work*,
but there are failure cases,
often exploitable by attackers.
Safe code is more complicated.

Elliptic-curve cryptography

Standardize prime p,
safe non-square d,
base point (x, y) on elliptic curve.

Alice knows her secret key a
and Bob's public key $b(x, y)$.
Alice computes (and caches)
shared secret $ab(x, y)$.

Alice uses shared secret to encrypt
and authenticate packet for Bob.

Packet overhead (high security):
32 bytes for Alice's public key,
24 bytes for nonce,
16 bytes for authenticator.

Bob receives packet,
sees Alice's public key $a(x, y)$.
Bob computes (and caches)
shared secret $ab(x, y)$.

Bob uses shared secret to
verify authenticator and decrypt
packet.

Alice and Bob
reuse the same shared secret to
encrypt, authenticate, verify, and
decrypt all subsequent packets.

All of this is so fast that
we can afford to
encrypt all packets.

A safe example

Choose $p = 2^{255} - 19$.

Choose $d = 121665/121666$;

this is non-square in $\mathbf{F}_p$.

$x^2 + y^2 = 1 + dx^2y^2$

is a safe curve for ECC.

A safe example

Choose $p = 2^{255} - 19$.
Choose $d = 121665/121666$;
this is non-square in $\mathbf{F}_p$.

$x^2 + y^2 = 1 + dx^2y^2$
is a safe curve for ECC.

$-x^2 + y^2 = 1 - dx^2y^2$
is another safe curve
using the same p and d.

A safe example

Choose $p = 2^{255} - 19$.
Choose $d = 121665/121666$;
this is non-square in $\mathbf{F}_p$.

$x^2 + y^2 = 1 + dx^2y^2$
is a safe curve for ECC.

$-x^2 + y^2 = 1 - dx^2y^2$
is another safe curve
using the same p and d.

Actually, the second curve
is the first curve in disguise:
replace x in first curve
by $\sqrt{-1} \cdot x$, using $\sqrt{-1} \in \mathbf{F}_p$.

Eliminating divisions

Typical computation:
$P \mapsto nP$.

Decompose into additions:
$P, Q \mapsto P + Q$.

Addition $(x_1, y_1) + (x_2, y_2) =$
$((x_1y_2 + y_1x_2)/(1 + dx_1x_2y_1y_2),$
$(y_1y_2 - x_1x_2)/(1 - dx_1x_2y_1y_2))$
uses expensive divisions.

Better: postpone divisions
and work with fractions.
Represent (x, y) as $(X : Y : Z)$
with $x = X/Z$ and $y = Y/Z$
for $Z \neq 0$.

Addition now has to handle fractions as input:

$$\left(\frac{X_1}{Z_1}, \frac{Y_1}{Z_1}\right) + \left(\frac{X_2}{Z_2}, \frac{Y_2}{Z_2}\right) =$$

$$\left(\frac{\frac{X_1}{Z_1}\frac{Y_2}{Z_2} + \frac{Y_1}{Z_1}\frac{X_2}{Z_2}}{1 + d\frac{X_1}{Z_1}\frac{X_2}{Z_2}\frac{Y_1}{Z_1}\frac{Y_2}{Z_2}}, \right.$$

$$\left. \frac{\frac{Y_1}{Z_1}\frac{Y_2}{Z_2} - \frac{X_1}{Z_1}\frac{X_2}{Z_2}}{1 - d\frac{X_1}{Z_1}\frac{X_2}{Z_2}\frac{Y_1}{Z_1}\frac{Y_2}{Z_2}} \right) =$$

Addition now has to handle fractions as input:

$$\left(\frac{X_1}{Z_1}, \frac{Y_1}{Z_1}\right) + \left(\frac{X_2}{Z_2}, \frac{Y_2}{Z_2}\right) =$$

$$\left(\frac{\frac{X_1}{Z_1}\frac{Y_2}{Z_2} + \frac{Y_1}{Z_1}\frac{X_2}{Z_2}}{1 + d\frac{X_1}{Z_1}\frac{X_2}{Z_2}\frac{Y_1}{Z_1}\frac{Y_2}{Z_2}}, \right.$$

$$\left.\frac{\frac{Y_1}{Z_1}\frac{Y_2}{Z_2} - \frac{X_1}{Z_1}\frac{X_2}{Z_2}}{1 - d\frac{X_1}{Z_1}\frac{X_2}{Z_2}\frac{Y_1}{Z_1}\frac{Y_2}{Z_2}}\right) =$$

$$\left(\frac{Z_1 Z_2 (X_1 Y_2 + Y_1 X_2)}{Z_1^2 Z_2^2 + d X_1 X_2 Y_1 Y_2}, \right.$$

$$\left.\frac{Z_1 Z_2 (Y_1 Y_2 - X_1 X_2)}{Z_1^2 Z_2^2 - d X_1 X_2 Y_1 Y_2}\right)$$

i.e. $\left(\frac{X_1}{Z_1}, \frac{Y_1}{Z_1}\right) + \left(\frac{X_2}{Z_2}, \frac{Y_2}{Z_2}\right)$

$= \left(\frac{X_3}{Z_3}, \frac{Y_3}{Z_3}\right)$

i.e. $\left(\frac{X_1}{Z_1}, \frac{Y_1}{Z_1}\right) + \left(\frac{X_2}{Z_2}, \frac{Y_2}{Z_2}\right)$

$= \left(\frac{X_3}{Z_3}, \frac{Y_3}{Z_3}\right)$

where

$F = Z_1^2 Z_2^2 - dX_1X_2Y_1Y_2,$

$G = Z_1^2 Z_2^2 + dX_1X_2Y_1Y_2,$

$X_3 = Z_1Z_2(X_1Y_2 + Y_1X_2)F,$

$Y_3 = Z_1Z_2(Y_1Y_2 - X_1X_2)G,$

$Z_3 = FG.$

Input to addition algorithm:

$X_1, Y_1, Z_1, X_2, Y_2, Z_2.$

Output from addition algorithm:

$X_3, Y_3, Z_3.$ No divisions needed!

Save multiplications by eliminating common subexpressions:

$A = Z_1 \cdot Z_2$; $B = A^2$;
$C = X_1 \cdot X_2$;
$D = Y_1 \cdot Y_2$;
$E = d \cdot C \cdot D$;
$F = B - E$; $G = B + E$;
$X_3 = A \cdot F \cdot (X_1 \cdot Y_2 + Y_1 \cdot X_2)$;
$Y_3 = A \cdot G \cdot (D - C)$;
$Z_3 = F \cdot G$.

Cost: $11\mathbf{M} + 1\mathbf{S} + 1\mathbf{D}$.
Can do better: $10\mathbf{M} + 1\mathbf{S} + 1\mathbf{D}$.

Faster doubling

$(x_1, y_1) + (x_1, y_1) =$
$((x_1y_1+y_1x_1)/(1+dx_1x_1y_1y_1),$
$(y_1y_1-x_1x_1)/(1-dx_1x_1y_1y_1)) =$
$((2x_1y_1)/(1 + dx_1^2y_1^2),$
$(y_1^2-x_1^2)/(1 - dx_1^2y_1^2)).$

$x_1^2 + y_1^2 = 1 + dx_1^2y_1^2$ so
$(x_1, y_1) + (x_1, y_1) =$
$((2x_1y_1)/(x_1^2 + y_1^2),$
$(y_1^2-x_1^2)/(2 - x_1^2 - y_1^2)).$

Again eliminate divisions
using $\mathbf{P}^2$: only $3\mathbf{M} + 4\mathbf{S}$.
Much faster than addition.
Useful: many doublings in ECC.

More addition strategies

Dual addition formula:

$(x_1, y_1) + (x_2, y_2) =$

$((x_1y_1 + x_2y_2)/(x_1x_2 + y_1y_2),$

$(x_1y_1 - x_2y_2)/(x_1y_2 - x_2y_1)).$

Low degree, no need for d.

Warning: fails for doubling!

Is this really "addition"?

Most EC formulas have failures.

More addition strategies

Dual addition formula:

$(x_1, y_1) + (x_2, y_2) =$

$((x_1y_1 + x_2y_2)/(x_1x_2 + y_1y_2),$

$(x_1y_1 - x_2y_2)/(x_1y_2 - x_2y_1)).$

Low degree, no need for d.

Warning: fails for doubling!

Is this really "addition"?

Most EC formulas have failures.

More coordinate systems:

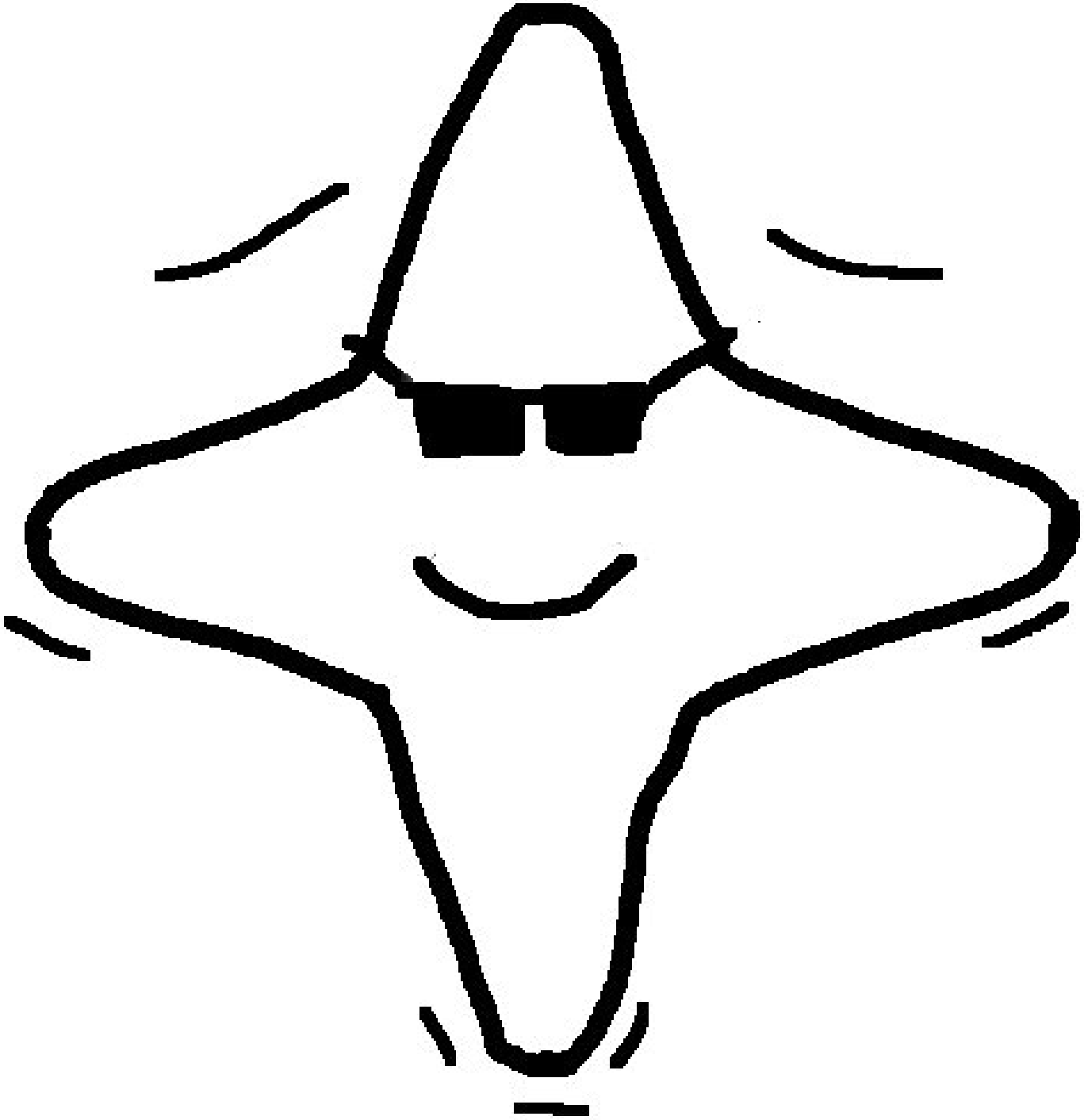
E.g. extended: $x = X/Z$,

$y = Y/Z$, $xy = T/Z$.

See "Explicit Formulas Database"

hyperelliptic.org/EFD

Edwards curves are cool



Birational equivalence

Starting from point (x, y)
on $x^2 + y^2 = 1 + dx^2y^2$:

Define $A = 2(1 + d)/(1 - d)$,
$B = 4/(1 - d)$;
$u = (1 + y)/(1 - y)$,
$v = u/x = (1 + y)/(x(1 - y))$.
(Skip a few exceptional points.)

$Bv^2 = u^3 + Au^2 + u$.

Maps Edwards to Montgomery.
Compatible with point addition!

Easily invert this map:
$x = u/v$, $y = (u - 1)/(u + 1)$.

Montgomery curves with the "Montgomery ladder".

```
def scalarmult(n,x1):
  x2,z2,x3,z3 = 1,0,x1,1
  for i in reversed(range(maxnbits)):
    bit = 1 & (n >> i)
    x2,x3 = cswap(x2,x3,bit)
    z2,z3 = cswap(z2,z3,bit)
    x3,z3 = ((x2*x3-z2*z3)^2,
             x1*(x2*z3-z2*x3)^2)
    x2,z2 = ((x2^2-z2^2)^2,
            4*x2*z2*(x2^2+A*x2*z2+z2^2)
    x2,x3 = cswap(x2,x3,bit)
    z2,z3 = cswap(z2,z3,bit)
  return x2*z2^(p-2)
```

More elliptic curves

Edwards curves are elliptic.
Easiest way to understand
elliptic curves is Edwards.

Geometrically, all elliptic curves
are Edwards curves.

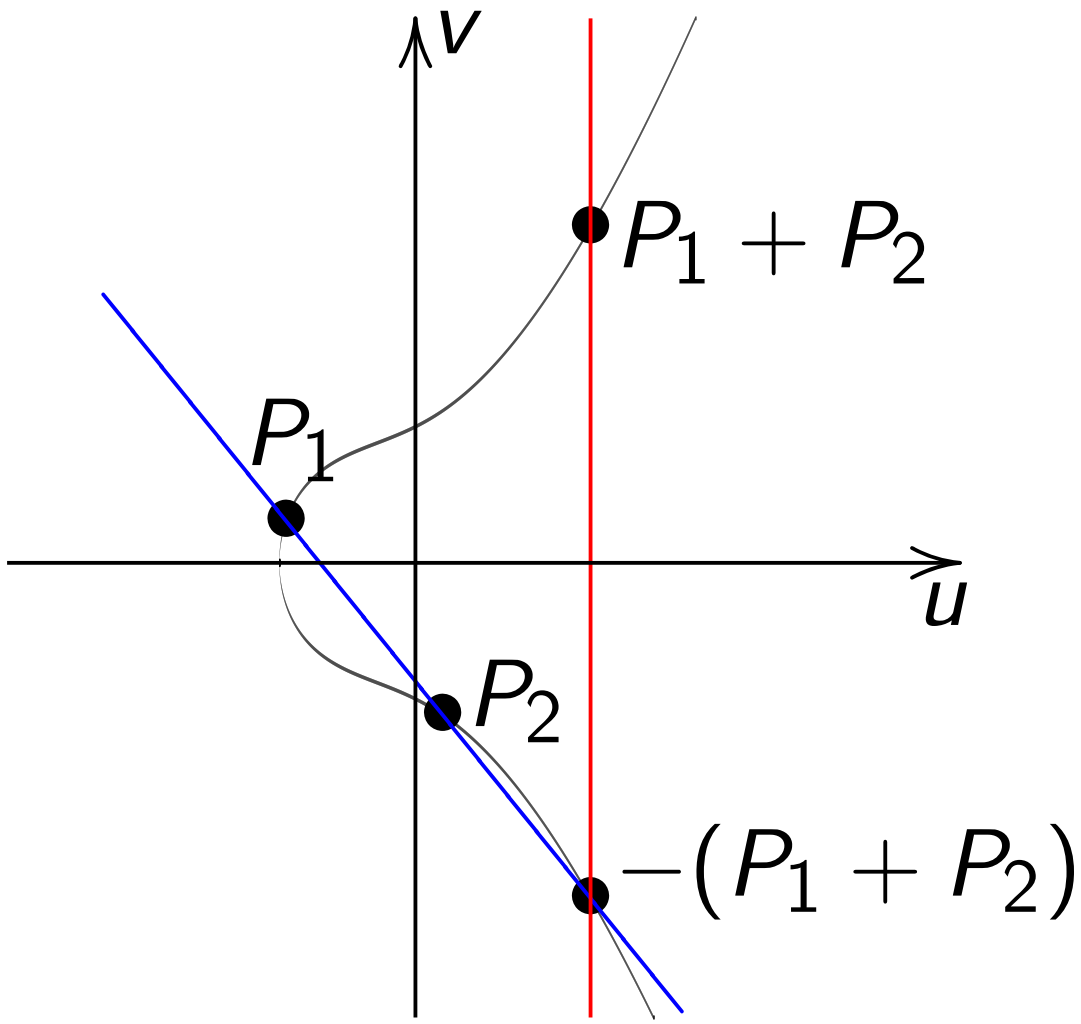
Algebraically,
more elliptic curves exist
(not always point of order 4).

Every odd-char curve can be
expressed as Weierstrass curve
$v^2 = u^3 + a_2u^2 + a_4u + a_6$.

Warning: "Weierstrass" has
different meaning in char 2.

Addition on Weierstrass curve

$$v^2 = u^3 + u^2 + u + 1$$

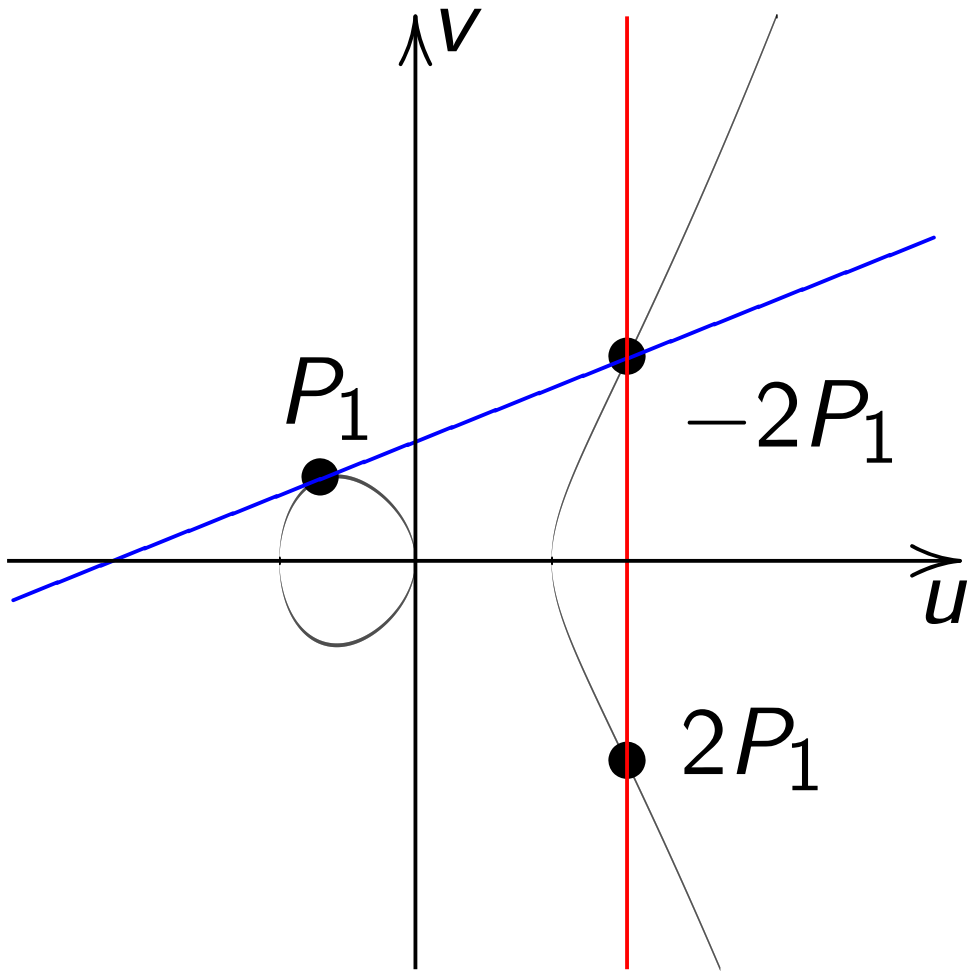


Slope $\lambda = (v_2 - v_1)/(u_2 - u_1)$.

Note that $u_1 \neq u_2$.

Doubling on Weierstrass curve

$v^2 = u^3 - u$



Slope $\lambda = (3u_1^2 - 1)/(2v_1)$.

In most cases
$(u_1, v_1) + (u_2, v_2) =$
(u_3, v_3) where $(u_3, v_3) =$
$(\lambda^2 - u_1 - u_2, \lambda(u_1 - u_3) - v_1)$.

$u_1 \neq u_2$, "addition" (alert!):
$\lambda = (v_2 - v_1)/(u_2 - u_1)$.
Total cost $1\mathbf{I} + 2\mathbf{M} + 1\mathbf{S}$.

$(u_1, v_1) = (u_2, v_2)$ and $v_1 \neq 0$,
"doubling" (alert!):
$\lambda = (3u_1^2 + 2a_2u_1 + a_4)/(2v_1)$.
Total cost $1\mathbf{I} + 2\mathbf{M} + 2\mathbf{S}$.

Also handle some exceptions:
$(u_1, v_1) = (u_2, -v_2)$; ∞ as input.
Messy to implement and test.

Some history

There are many perspectives on elliptic-curve computations.

1984 (published 1987) Lenstra: ECM, the elliptic-curve method of factoring integers.

1984 (published 1985) Miller, and independently
1984 (published 1987) Koblitz: Elliptic-curve cryptography.

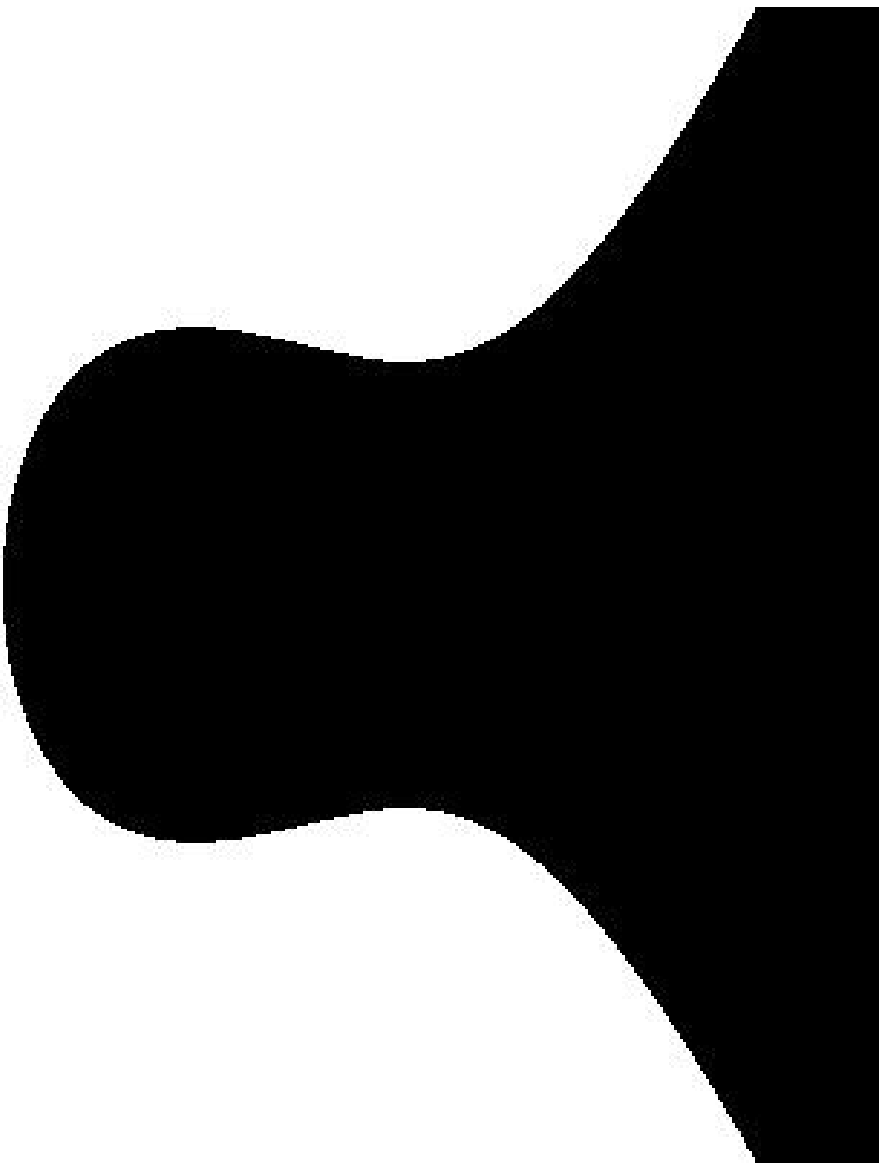
Bosma, Goldwasser–Kilian, Chudnovsky–Chudnovsky, Atkin: elliptic-curve primality proving.

The Edwards perspective is new!

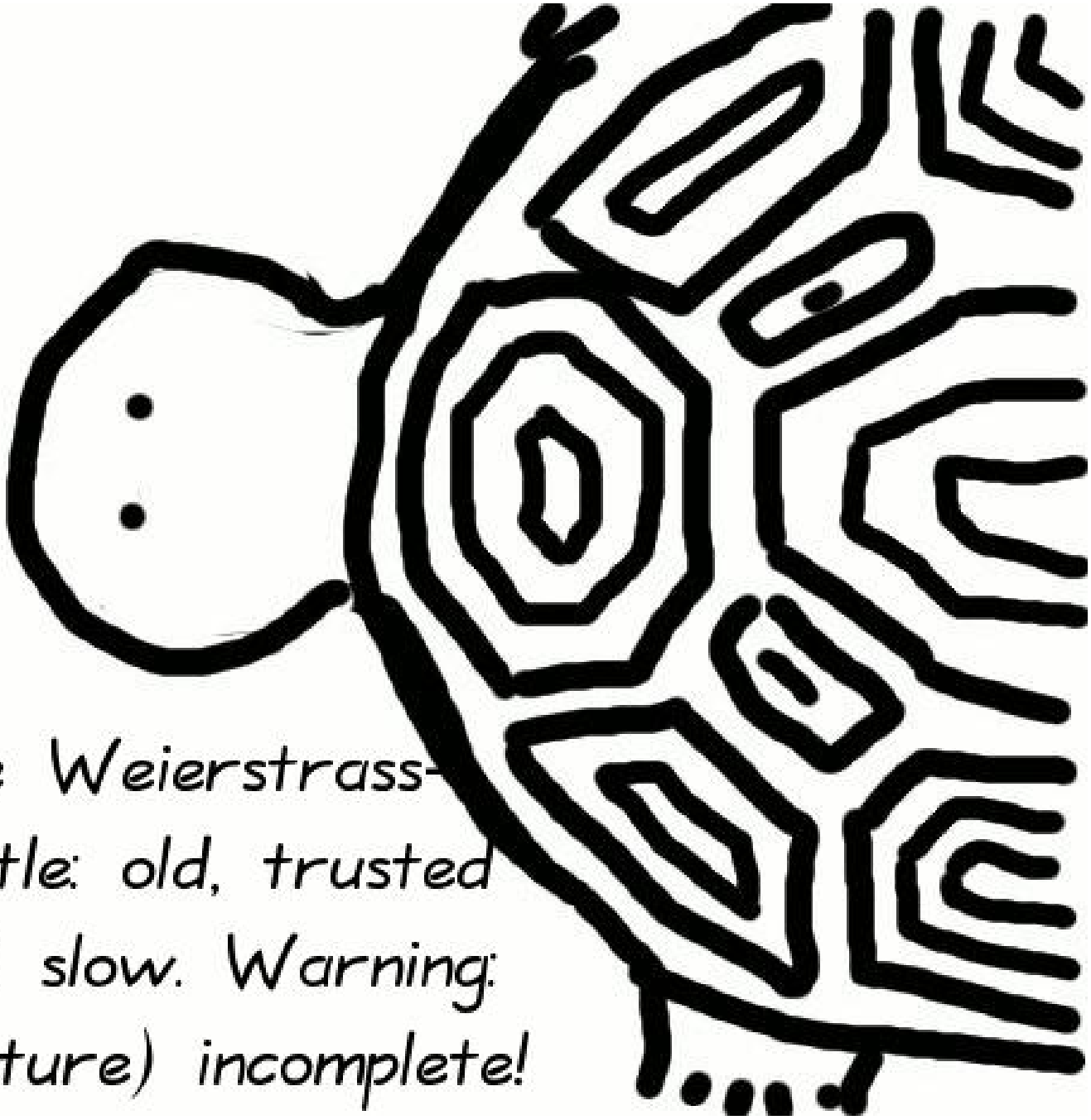
1761 Euler, 1866 Gauss
introduced an addition law
for $x^2 + y^2 = 1 - x^2y^2$,
the "lemniscatic elliptic curve."

2007 Edwards generalized to
many curves $x^2+y^2 = 1+c^4x^2y^2$.
Theorem: have now obtained
all elliptic curves over $\overline{\mathbf{Q}}$.

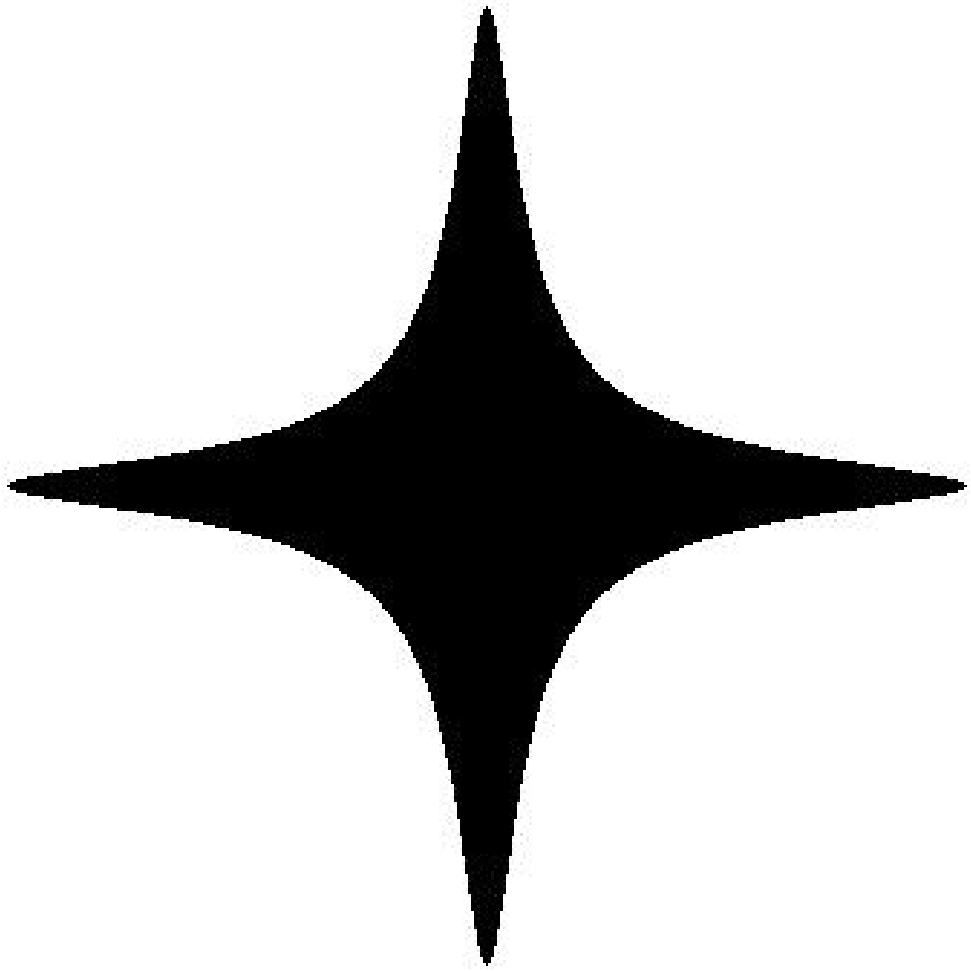
2007 Bernstein–Lange:
Edwards addition law is complete
for $x^2 + y^2 = 1 + dx^2y^2$ if $d \neq$ ■;
and gives new ECC speed records.



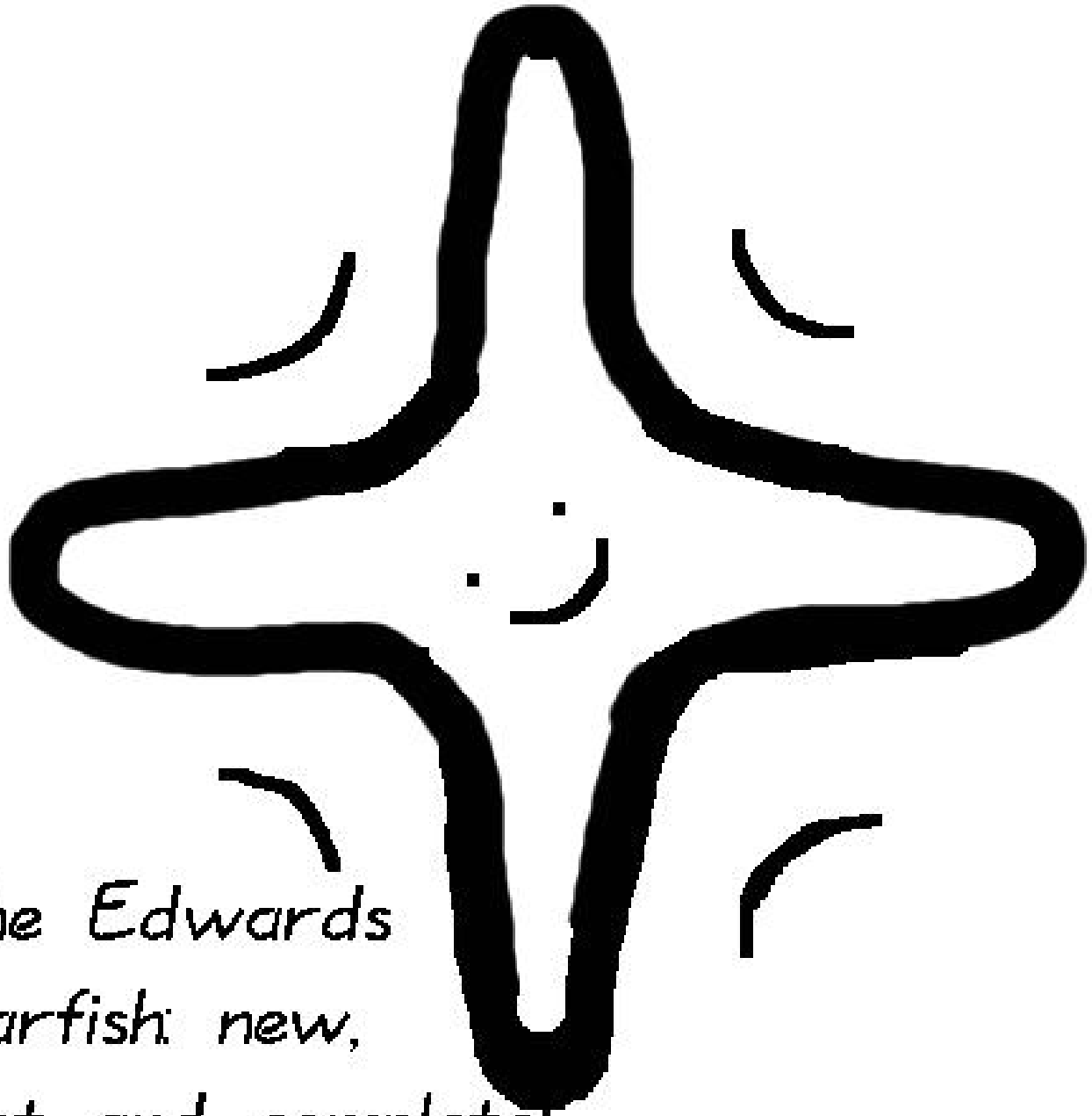
$$y^2 = x^3 - 0.4x + 0.7$$



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



$$x^2 + y^2 = 1 - 300x^2y^2$$



The Edwards starfish: new, fast and complete!

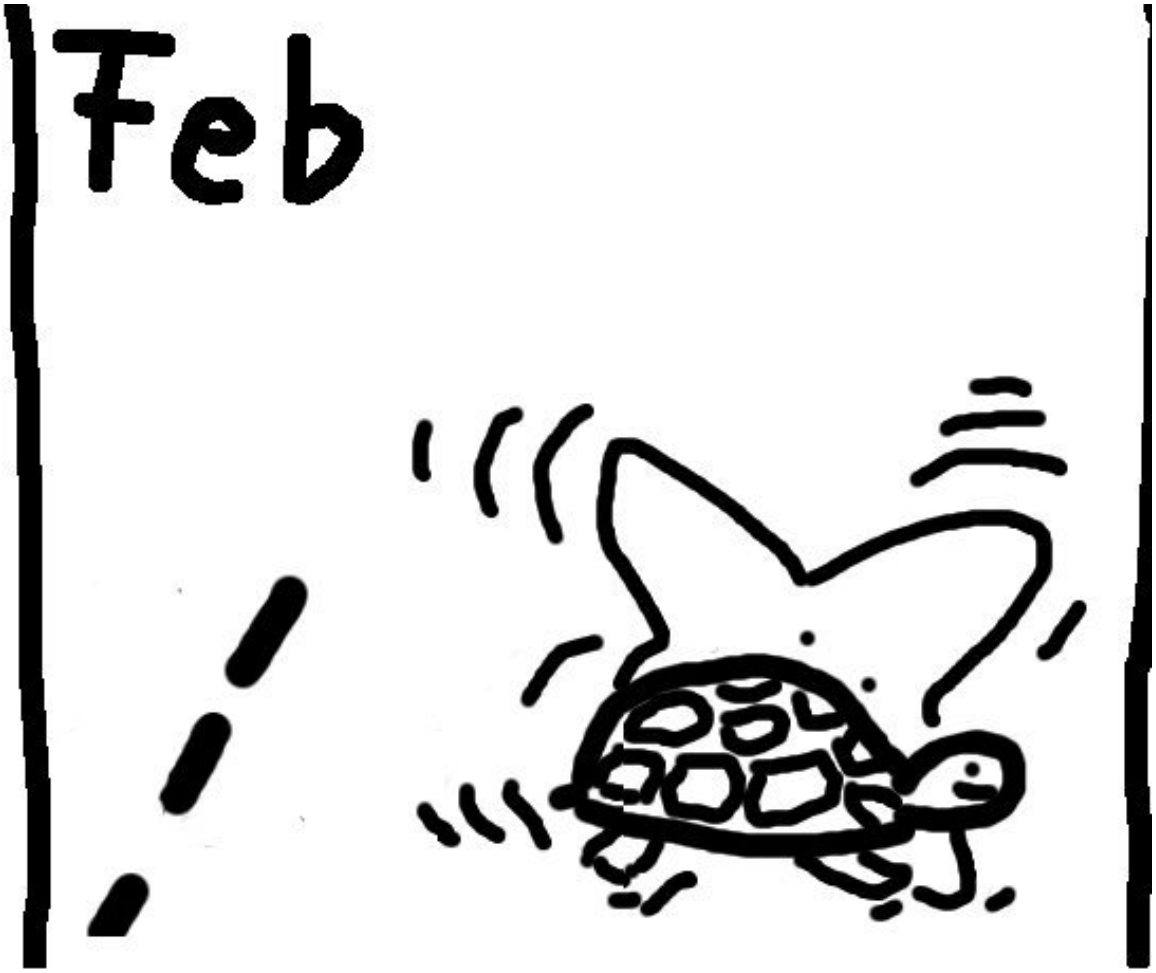
Start!

1985

Weierstrass sets off. Edwards
left behind sleeping

2007-Jan

Weierstrass has made some progress -
finally Edwards wakes up.



Exciting progress: Edwards about to overtake!!

Mar

And the winner is: Edwards!

Curve selection

How to defend yourself against an attacker armed with a mathematician:

1999 ANSI X9.62.
2000 IEEE P1363.
2000 Certicom SEC 2.
2000 NIST FIPS 186-2.
2001 ANSI X9.63.
2005 Brainpool.
2005 NSA Suite B.
2010 Certicom SEC 2 v2.
2010 OSCCA SM2.
2011 ANSSI FRP256V1.

Pick any of these standards.

What all of them achieve:
No known attack will compute
user's secret key from public key.
("Elliptic-curve discrete-log
problem for these curves.")

Example of common criterion:
Standard base point (x, y)
has huge prime "order" ℓ,
i.e., exactly ℓ different multiples.

Criteria are computer verifiable.
See our evaluation site for scripts:
safecurves.cr.yp.to

You do everything right.

You pick the Brainpool curve
`brainpoolP256t1`:
huge prime p,
$y^2 = x^3 - 3x +$ somehugenumber,
standard base point.

This curve isn't compatible
with Edwards or Montgomery.
So you check and test every case
in the Weierstrass formulas.

You make it all constant-time.
It's horrendously slow,
but it's secure.

Actually, it's not.

The attacker sent you (x', y')

$$x' = \begin{matrix}\texttt{1025b35abab9150d86770f6bda12f8ec} \\ \texttt{1e86bec6c6bac120535e4134fea87831}\end{matrix}$$

$$y' = \begin{matrix}\texttt{12ace5eeae9a5b0bca8ed1c0f9540d05} \\ \texttt{d123d55f68100099b65a99ac358e3a75}\end{matrix}.$$

You computed $a(x', y')$ using Weierstrass formulas.

You encrypted using AES-GCM with a hash$(a(x', y'))$ as a key.

Actually, it's not.

The attacker sent you (x', y')

$$x' = \texttt{1025b35abab9150d86770f6bda12f8ec}\\\texttt{1e86bec6c6bac120535e4134fea87831}$$

$$y' = \texttt{12ace5eeae9a5b0bca8ed1c0f9540d05}\\\texttt{d123d55f68100099b65a99ac358e3a75}.$$

You computed $a(x', y')$ using Weierstrass formulas.

You encrypted using AES-GCM with a hash$(a(x', y'))$ as a key.

What you never noticed:

(x', y') isn't his key $b(x, y)$;

it isn't even a point on `brainpoolP256t1`;

it's a point on $y^2 = x^3 - 3x + 5$ of order only 4999.

Your formulas worked for
$y^2 = x^3 - 3x + 5$
because they work for any
$y^2 = x^3 - 3x + a_6$:

Addition on Weierstrass curves
$y^2 = x^3 + a_4x + a_6$:
for $x_1 \neq x_2$, $(x_1, y_1) + (x_2, y_2) =$
(x_3, y_3) with $x_3 = \lambda^2 - x_1 - x_2$,
$y_3 = \lambda(x_1 - x_3) - y_1$,
$\lambda = (y_2 - y_1)/(x_2 - x_1)$;
for $y_1 \neq 0$, $(x_1, y_1) + (x_1, y_1) =$
(x_3, y_3) with $x_3 = \lambda^2 - x_1 - x_2$,
$y_3 = \lambda(x_1 - x_3) - y_1$,
$\lambda = (3x_1^2 + a_4)/2y_1$;
$(x_1, y_1) + (x_1, -y_1) = \infty$;
$(x_1, y_1) + \infty = (x_1, y_1)$;
$\infty + (x_2, y_2) = (x_2, y_2)$;
$\infty + \infty = \infty$.
Messy to implement and test.

} No a_6

Why this matters:
(x', y') has order 4999.
The attacker tries all 4999
possibilities, compares to the
AES-GCM output,
learns your secret a mod 4999.

Why this matters:
(x', y') has order 4999.
The attacker tries all 4999
possibilities, compares to the
AES-GCM output,
learns your secret a mod 4999.

Attacker then tries again with

$x' =$ `9bc001a0d2d5c43863aadb0f881df3bb`
`af3a5ea81eedd2385e6525521aa8b1e2`
$y' =$ `0d124e9e94dcede52aa0e3bcac1852cf`
`ed28eb86039c0d8e0cfaa4ae703eac07`,

a point of order 19559
on $y^2 = x^3 - 3x + 211$;
learns your secret a mod 19559.

Etc. Uses "Chinese remainder
theorem" to compute a.

Traditional response to this security failure:
Blame the implementor!

"You should have checked that the incoming (x', y') was on right curve and had right order."

Traditional response to this security failure:
Blame the implementor!

"You should have checked that the incoming (x', y') was on right curve and had right order."

But it's much better to *design the system without traps*.

Never send full (x, y).
Design protocols to compress one coordinate to 1 or 0 bits!
Drastically limits possibilities for attacker to choose points.

Always multiply by cofactor.

If curve has $c \cdot \ell$ points
and base point P has order ℓ
then c is called the cofactor
and $c \cdot \ell$ is called the curve order.

Design protocols to multiply by c.

Always use twist-secure curves.

Montgomery formulas use only A,
but modifying B gives only *two*
different curves. Require both to
have almost-prime order.

These choices are robust
against every common DH
implementation error.

Sage scripts to verify criteria
for ECDLP security
and ECC security:

safecurves.cr.yp.to

Analysis of manipulability of
various curve-generation methods:

safecurves.cr.yp.to/bada55.html

Many computer-verified addition
formulas:

hyperelliptic.org/EFD/

Python scripts for this talk:

ecchacks.cr.yp.to

Crypto library

nacl.cr.yp.to