Cryptanalysis Course
Part I

Tanja Lange
Technische Universiteit Eindhoven

28 Nov 2016

with some slides by
Daniel J. Bernstein

Main goal of this course:
We are the attackers.
We want to break ECC and RSA.

First need to understand ECC;
this is also needed for Dan's
high-speed crypto course.

Main motivation for ECC:
Avoid index-calculus attacks
that plague finite-field DL.
See, e.g., yesterday's talk by
P. T. H. Duong.

Diffie-Hellman key exchange

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

$$
\begin{tikzcd}
\text{Alice's secret key } a \arrow[d] \arrow[dd, bend right] & & \text{Bob's secret key } b \arrow[d] \arrow[dd, bend left] \\
\text{Alice's public key } aP \arrow[drr] & & \text{Bob's public key } b\,P \arrow[dll] \\
\{\text{Alice}, \text{Bob}\}\text{'s shared secret } ab\,P & = & \{\text{Bob}, \text{Alice}\}\text{'s shared secret } b\,aP
\end{tikzcd}
$$

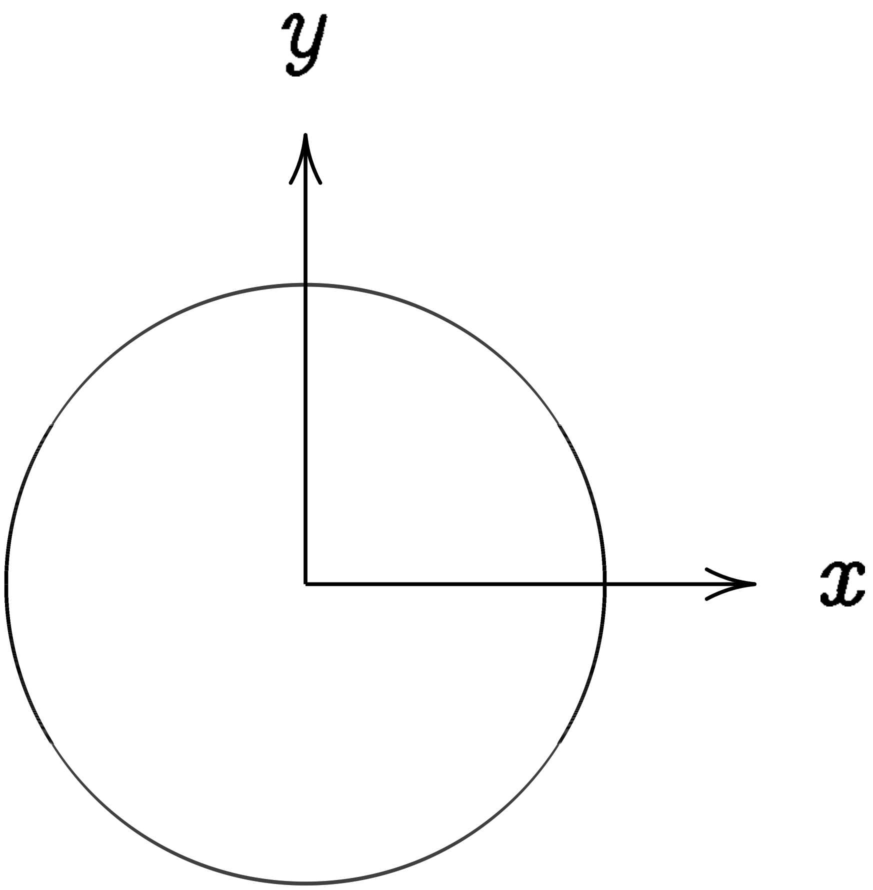
Diffie-Hellman key exchange

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

$$
\begin{tikzcd}
\begin{array}{c}\text{Alice's}\\ \text{secret key } a\end{array} \arrow[d] \arrow[dd, bend right] & & \begin{array}{c}\text{Bob's}\\ \text{secret key } b\end{array} \arrow[d] \arrow[dd, bend left] \\
\begin{array}{c}\text{Alice's}\\ \text{public key}\\ aP\end{array} \arrow[drr] & & \begin{array}{c}\text{Bob's}\\ \text{public key}\\ b\,P\end{array} \arrow[dll] \\
\begin{array}{c}\{\text{Alice}, \text{Bob}\}\text{'s}\\ \text{shared secret}\\ ab\,P\end{array} & = & \begin{array}{c}\{\text{Bob}, \text{Alice}\}\text{'s}\\ \text{shared secret}\\ b\,aP\end{array}
\end{tikzcd}
$$

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".

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

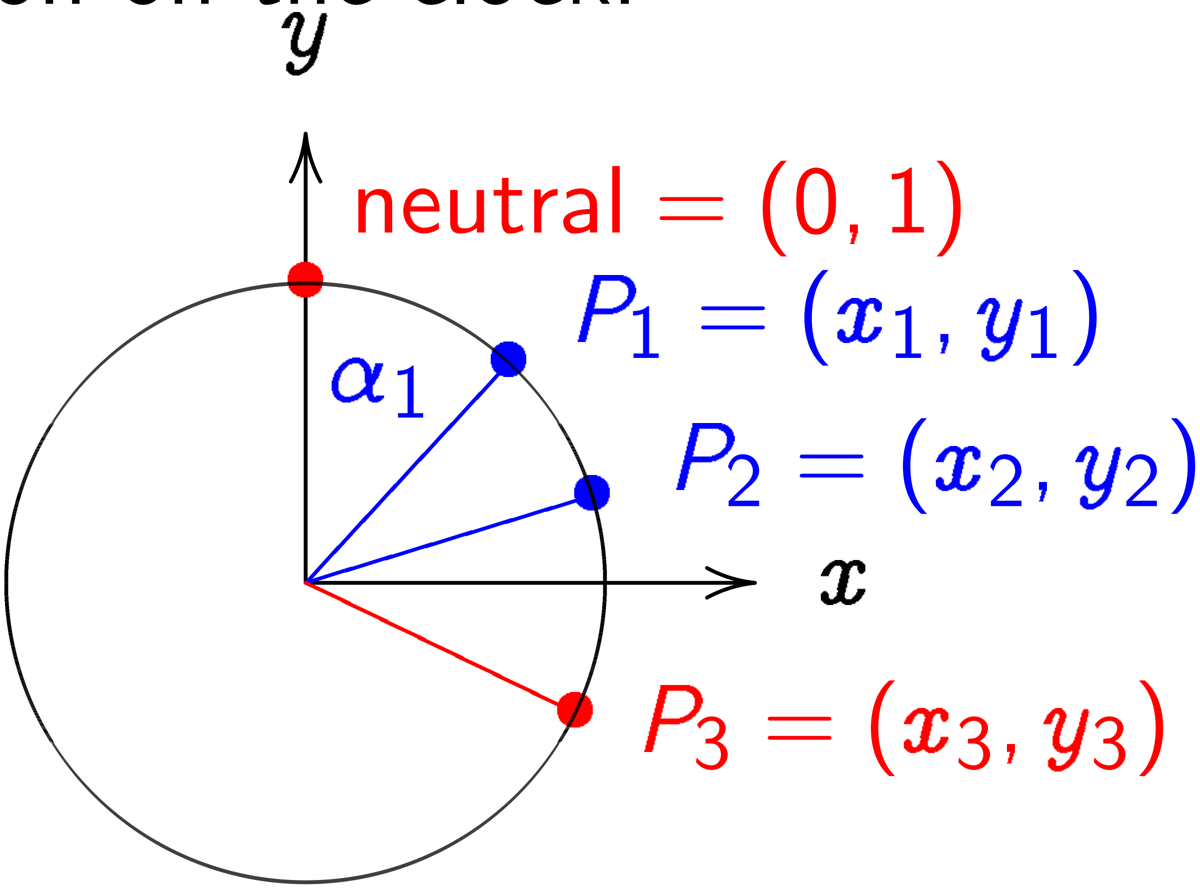
$(3/5, -4/5)$. $(-3/5, -4/5)$.

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

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

Many more.

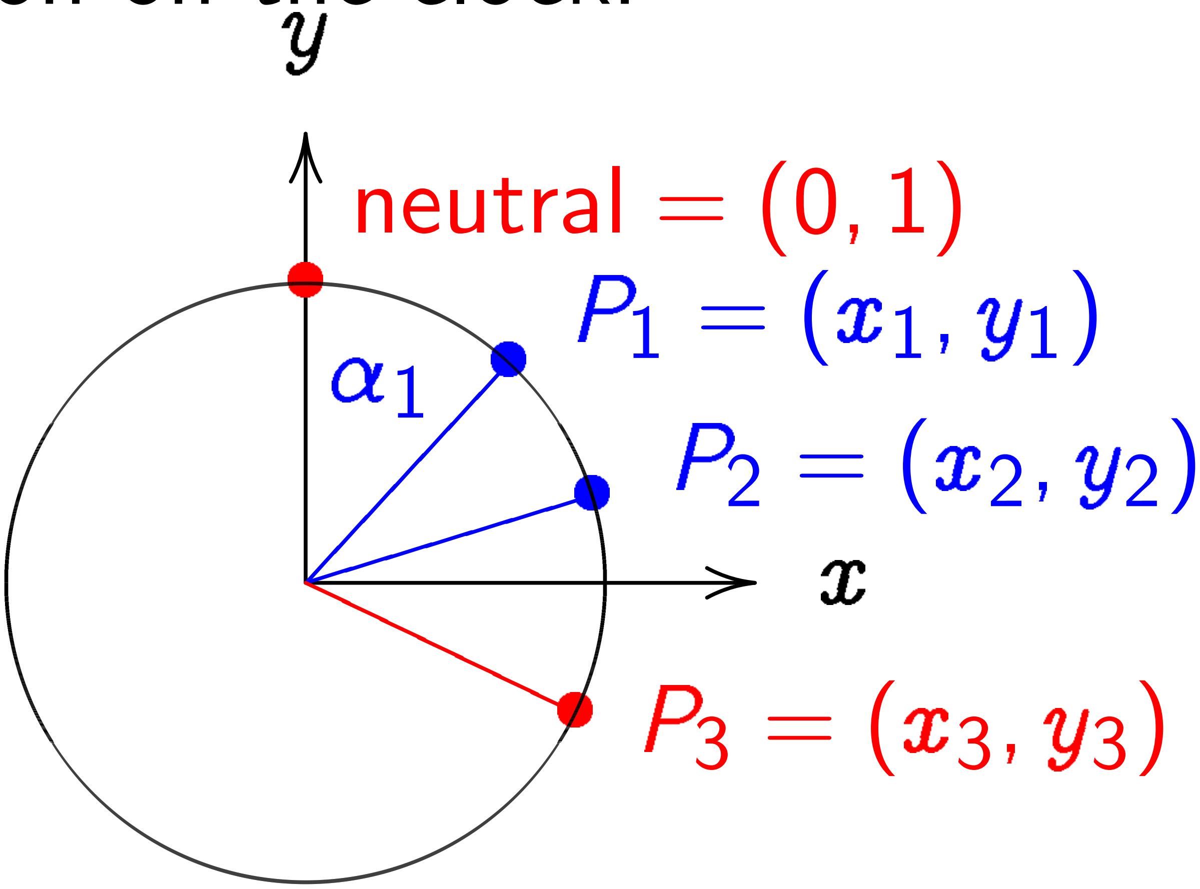
Addition on the clock:



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

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

Addition on the clock:

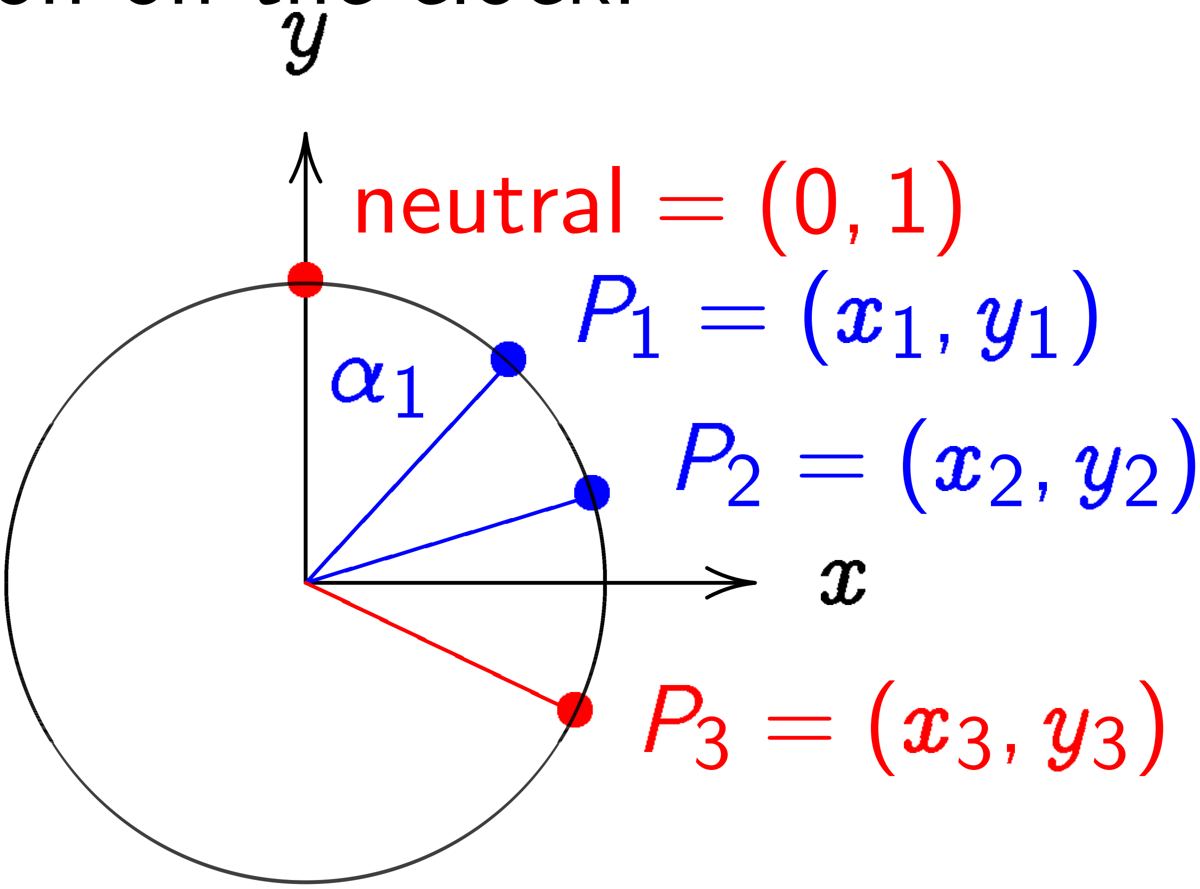


$x^2 + y^2 = 1$, parametrized by

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

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

Addition on the clock:



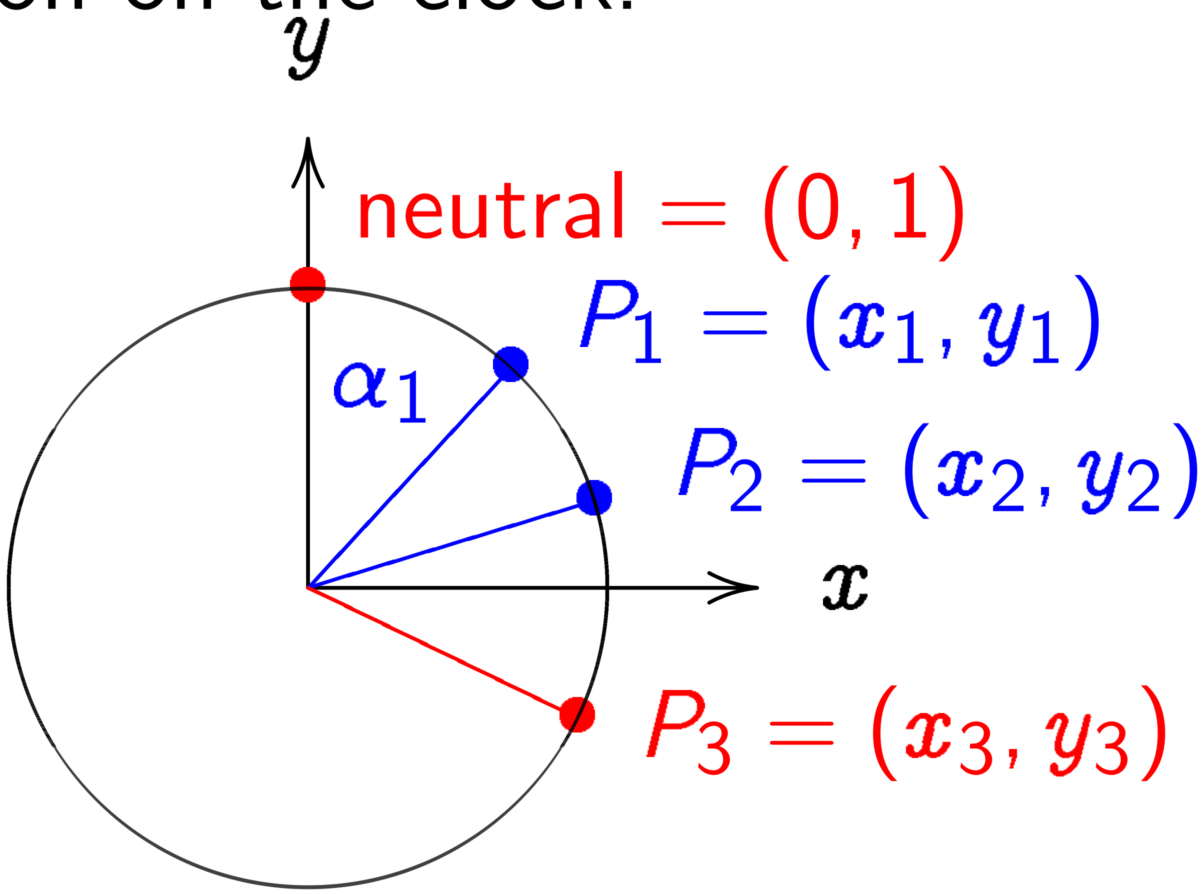
$x^2 + y^2 = 1$, parametrized by

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



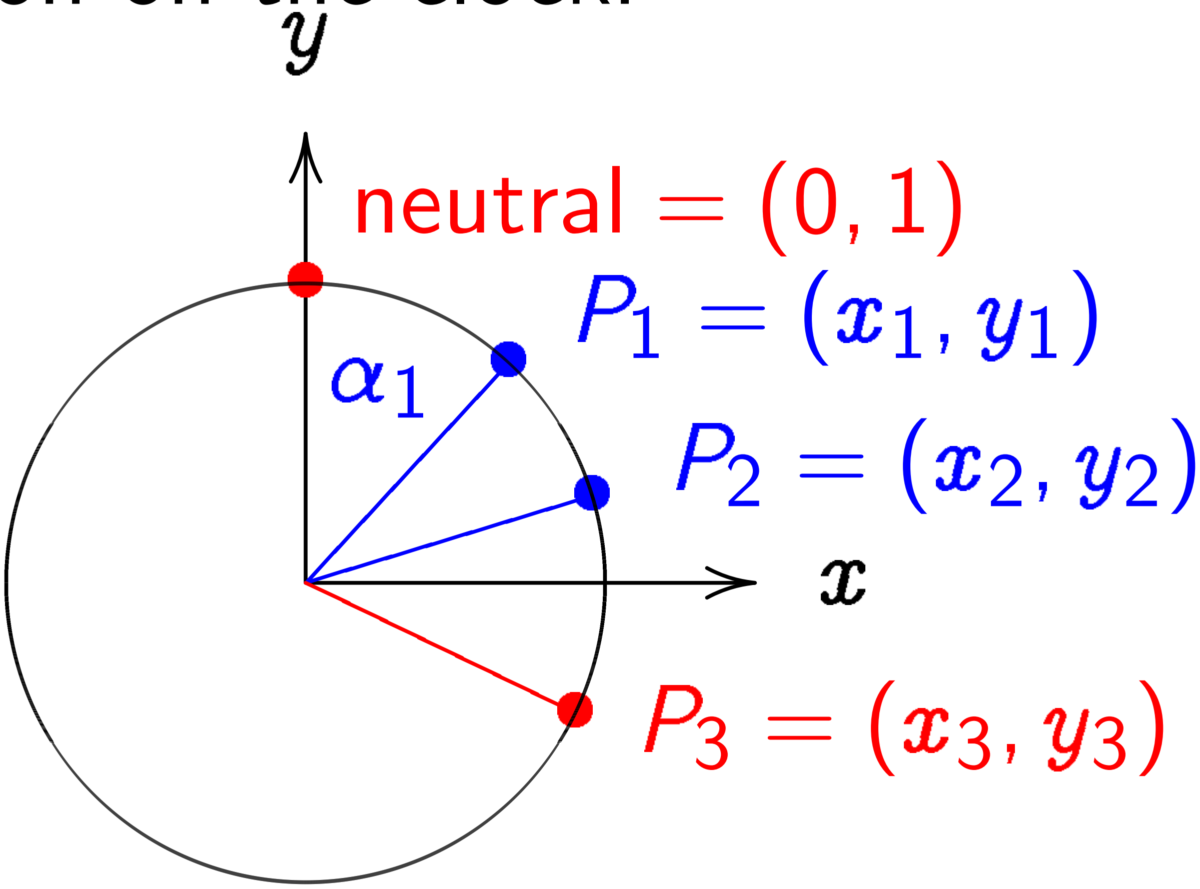
$x^2 + y^2 = 1$, parametrized by $x = \sin\alpha$, $\quad y = \cos\alpha$. Recall

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

Adding two points corresponds to adding the angles α_1 and α_2. Angles modulo 360° are a group, so points on clock are a group.

Neutral element: angle $\alpha = 0$; point $(0, 1)$; "12:00".
The point with $\alpha = 180°$
has order 2 and equals 6:00.
3:00 and 9:00 have order 4.
Inverse of point with α
is point with $-\alpha$
since $\alpha + (-\alpha) = 0$.
There are many more points
where angle α is not "nice."

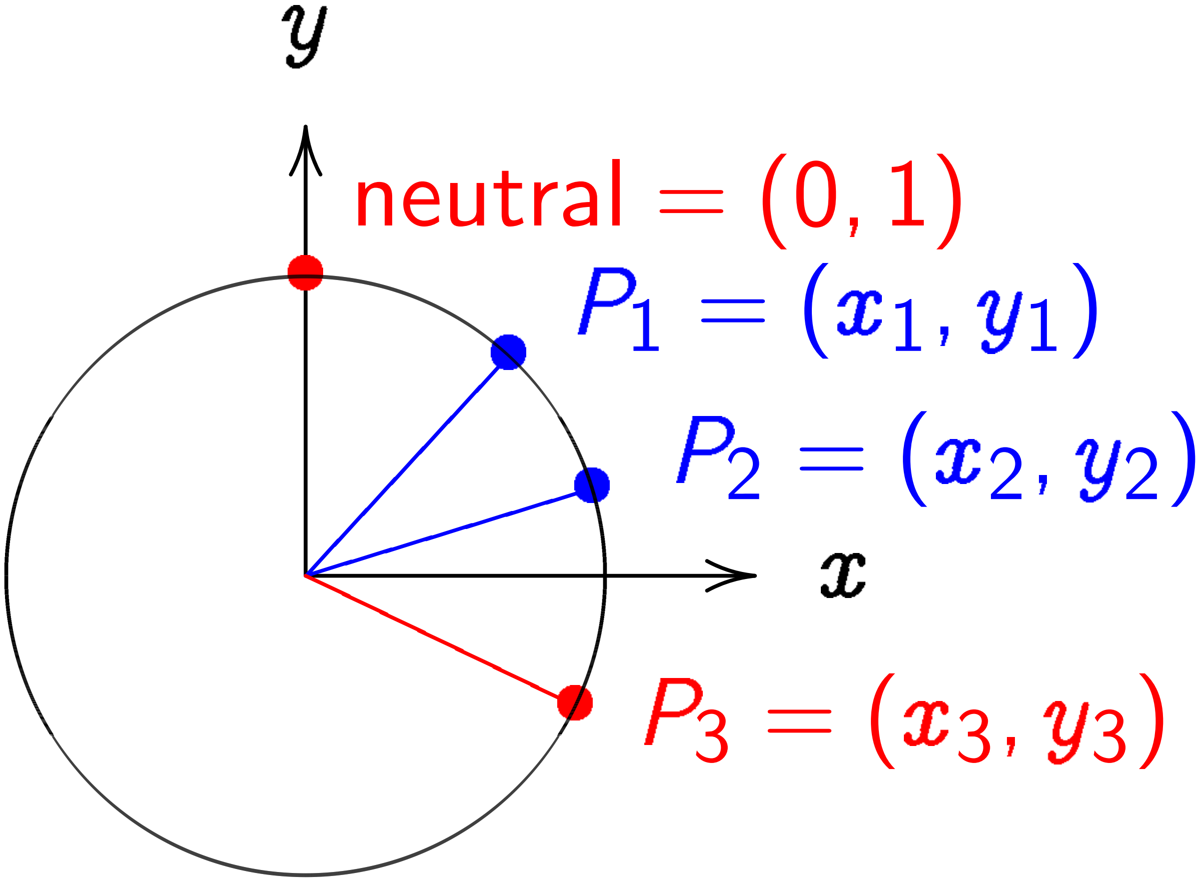
Addition on the clock:



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

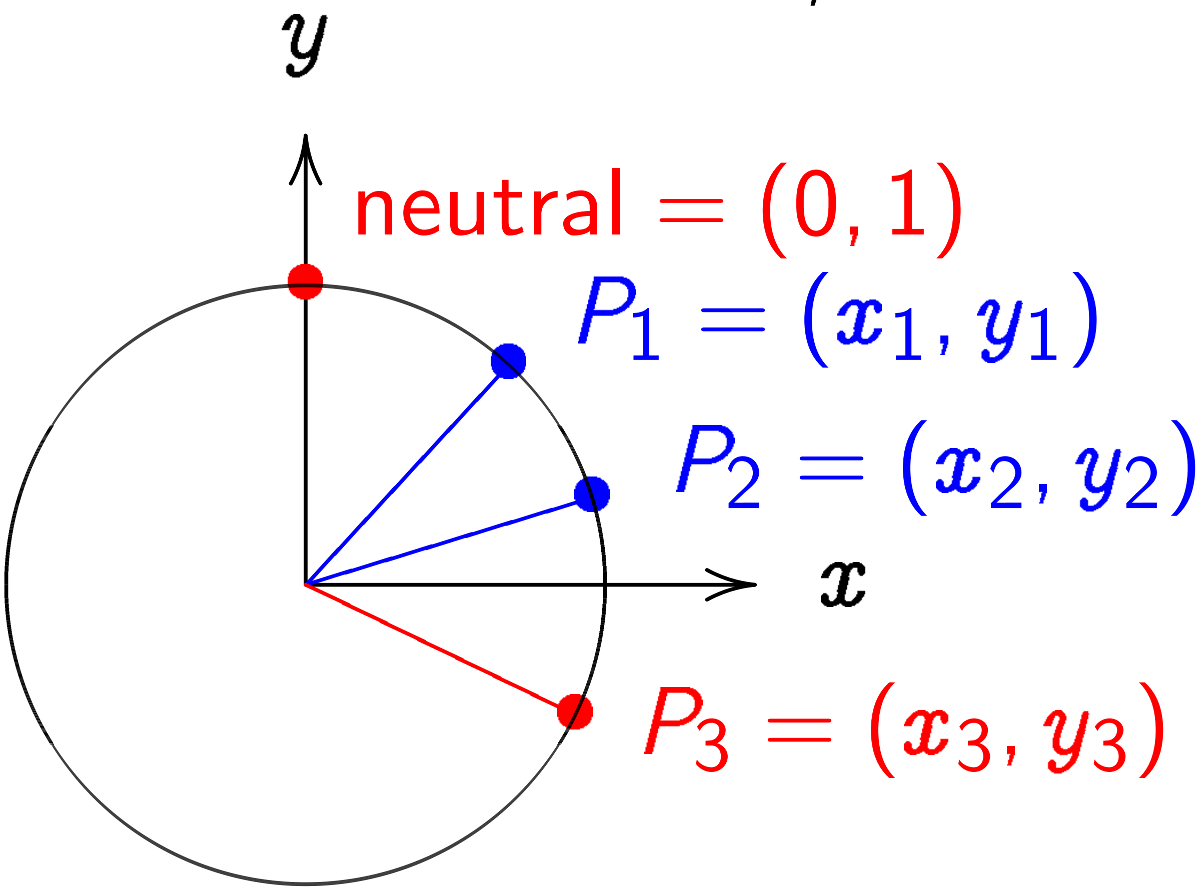
$(\sin(\alpha_1 + \alpha_2), \cos(\alpha_1 + \alpha_2)) =$
$(\sin\alpha_1 \cos\alpha_2 + \cos\alpha_1 \sin\alpha_2,$
$\cos\alpha_1 \cos\alpha_2 - \sin\alpha_1 \sin\alpha_2).$

Clock addition without sin, cos:



Use Cartesian coordinates for addition. Addition formula for the clock $x^2 + y^2 = 1$: sum $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$

Clock addition without sin, cos:



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

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

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

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:

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

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

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

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

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

Examples of clock addition:

"2:00" + "5:00"

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

"5:00" + "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:

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

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

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

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

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

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

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

Examples of clock addition:

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

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

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

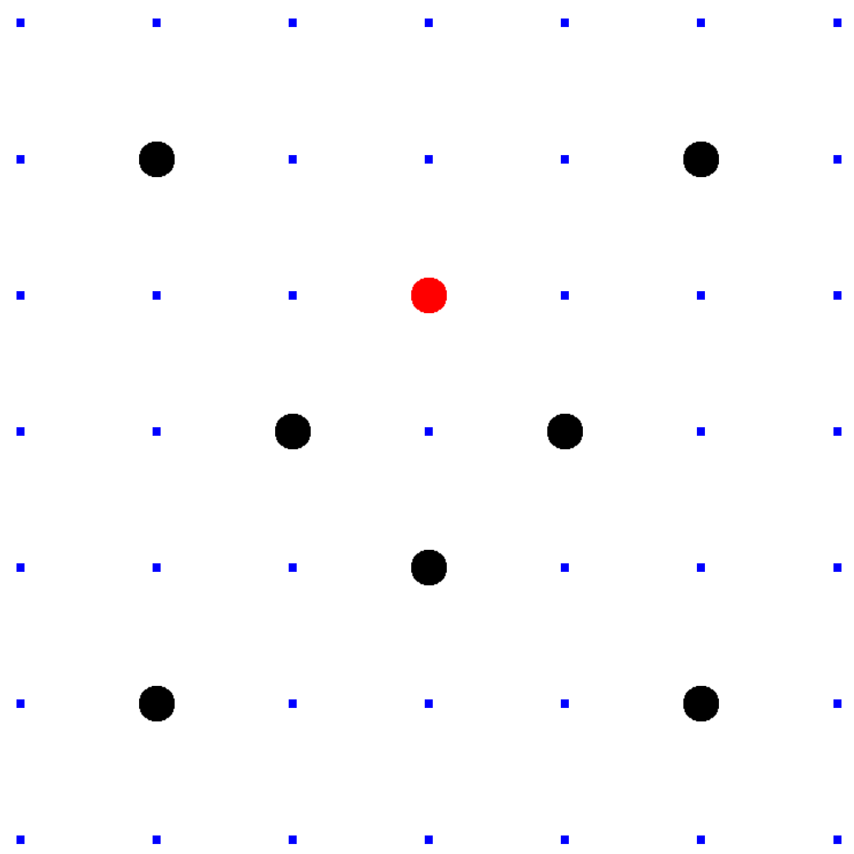
$$3\left(\frac{3}{5}, \frac{4}{5}\right) = \left(\frac{117}{125}, \frac{-44}{125}\right).$$

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

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

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

Clocks over finite fields



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

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.

$$
\begin{tikzcd}
\begin{array}{c}\text{Alice's}\\ \text{secret key } a\end{array} \arrow[d] \arrow[dd, bend right] & & \begin{array}{c}\text{Bob's}\\ \text{secret key } b\end{array} \arrow[d] \arrow[dd, bend left] \\
\begin{array}{c}\text{Alice's}\\ \text{public key}\\ a(X,Y)\end{array} \arrow[drr] & & \begin{array}{c}\text{Bob's}\\ \text{public key}\\ b\,(X,Y)\end{array} \arrow[dll] \\
\begin{array}{c}\{\text{Alice}, \text{Bob}\}\text{'s}\\ \text{shared secret}\\ ab\,(X,Y)\end{array} & = & \begin{array}{c}\{\text{Bob}, \text{Alice}\}\text{'s}\\ \text{shared secret}\\ b\,a(X,Y)\end{array}
\end{tikzcd}
$$

$$
\begin{tikzcd}
\begin{array}{c}\text{Alice's}\\ \text{secret key } a\end{array} \arrow[d] \arrow[dd, bend right] & & \begin{array}{c}\text{Bob's}\\ \text{secret key } b\end{array} \arrow[d] \arrow[dd, bend left] \\
\begin{array}{c}\text{Alice's}\\ \text{public key}\\ a(X,Y)\end{array} \arrow[drr] & & \begin{array}{c}\text{Bob's}\\ \text{public key}\\ b\,(X,Y)\end{array} \arrow[dll] \\
\begin{array}{c}\{\text{Alice, Bob}\}\text{'s}\\ \text{shared secret}\\ ab\,(X,Y)\end{array} & = & \begin{array}{c}\{\text{Bob, Alice}\}\text{'s}\\ \text{shared secret}\\ b\,a(X,Y)\end{array}
\end{tikzcd}
$$

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.

Exercise

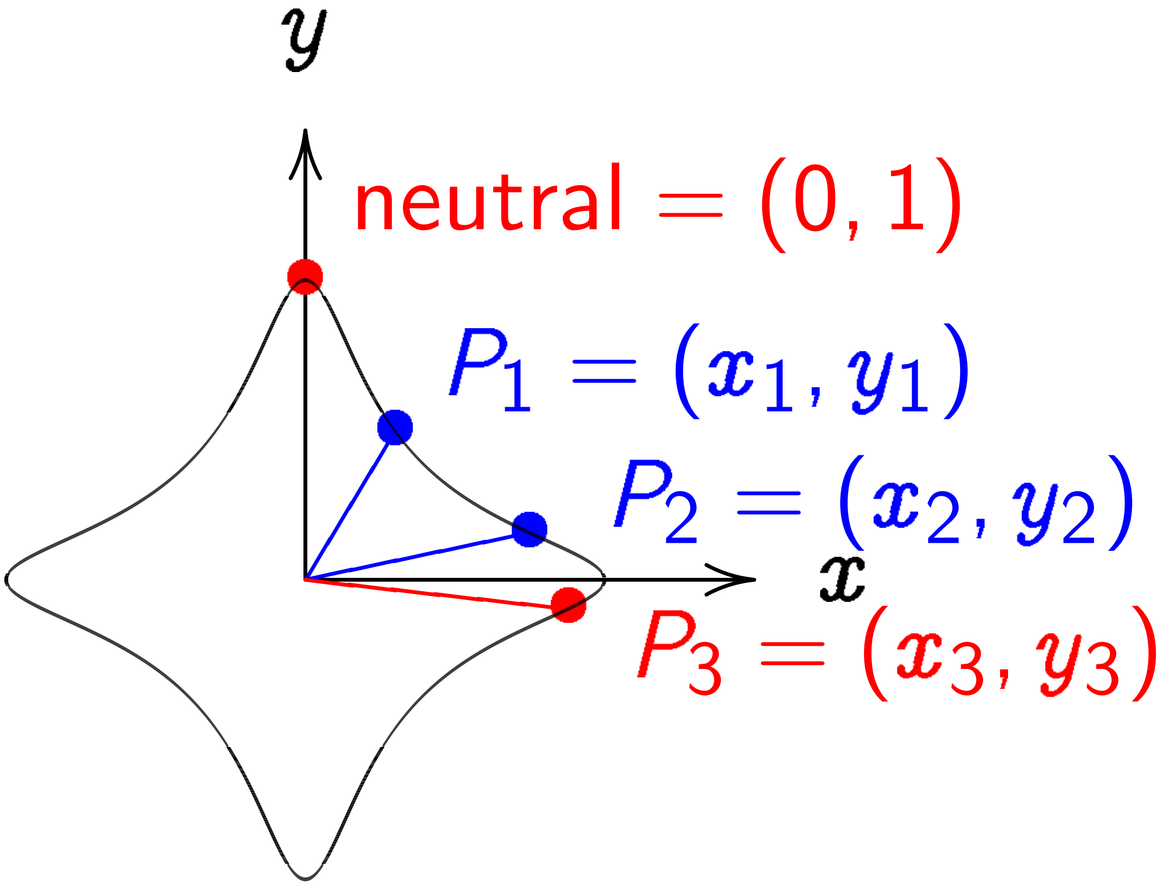
How many multiplications do you need to compute $(x_1 y_2 + y_1 x_2, y_1 y_2 - x_1 x_2)$?

How many multiplications do you need to double a point, i.e. to compute $(x_1 y_1 + y_1 x_1, y_1 y_1 - x_1 x_1)$?

How can you optimize the computation if squarings are cheaper than multiplications? Assume $\mathbf{S} < \mathbf{M} < 2\mathbf{S}$.

Addition on an Edwards curve

Change the curve on which Alice and Bob work.

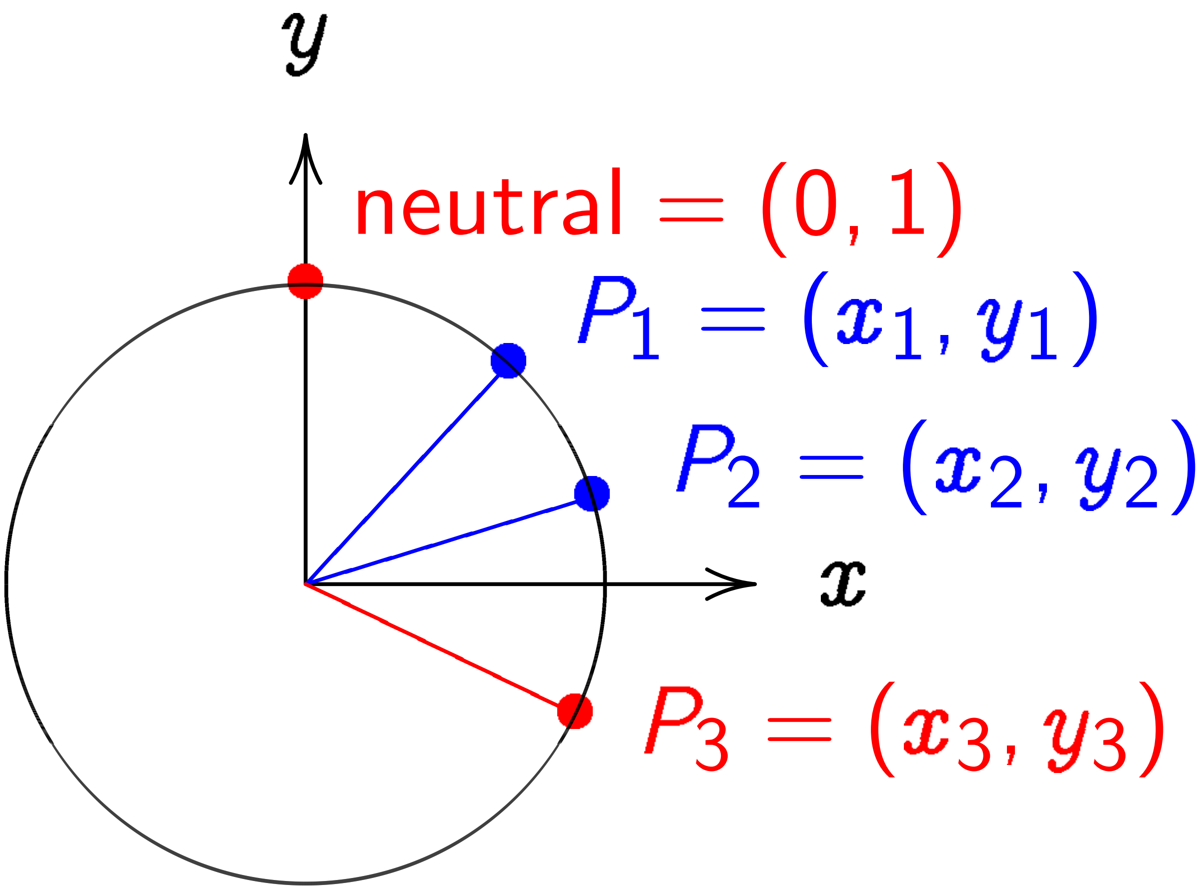


$x^2 + y^2 = 1 - 30x^2y^2$.

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

$(x_1 y_2 + y_1 x_2,$

$y_1 y_2 - x_1 x_2).$

“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

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

$((x_1 y_2 + y_1 x_2)/(1 - 30 x_1 x_2 y_1 y_2),$

$(y_1 y_2 - x_1 x_2)/(1 + 30 x_1 x_2 y_1 y_2))$

is a group law for the curve

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

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.

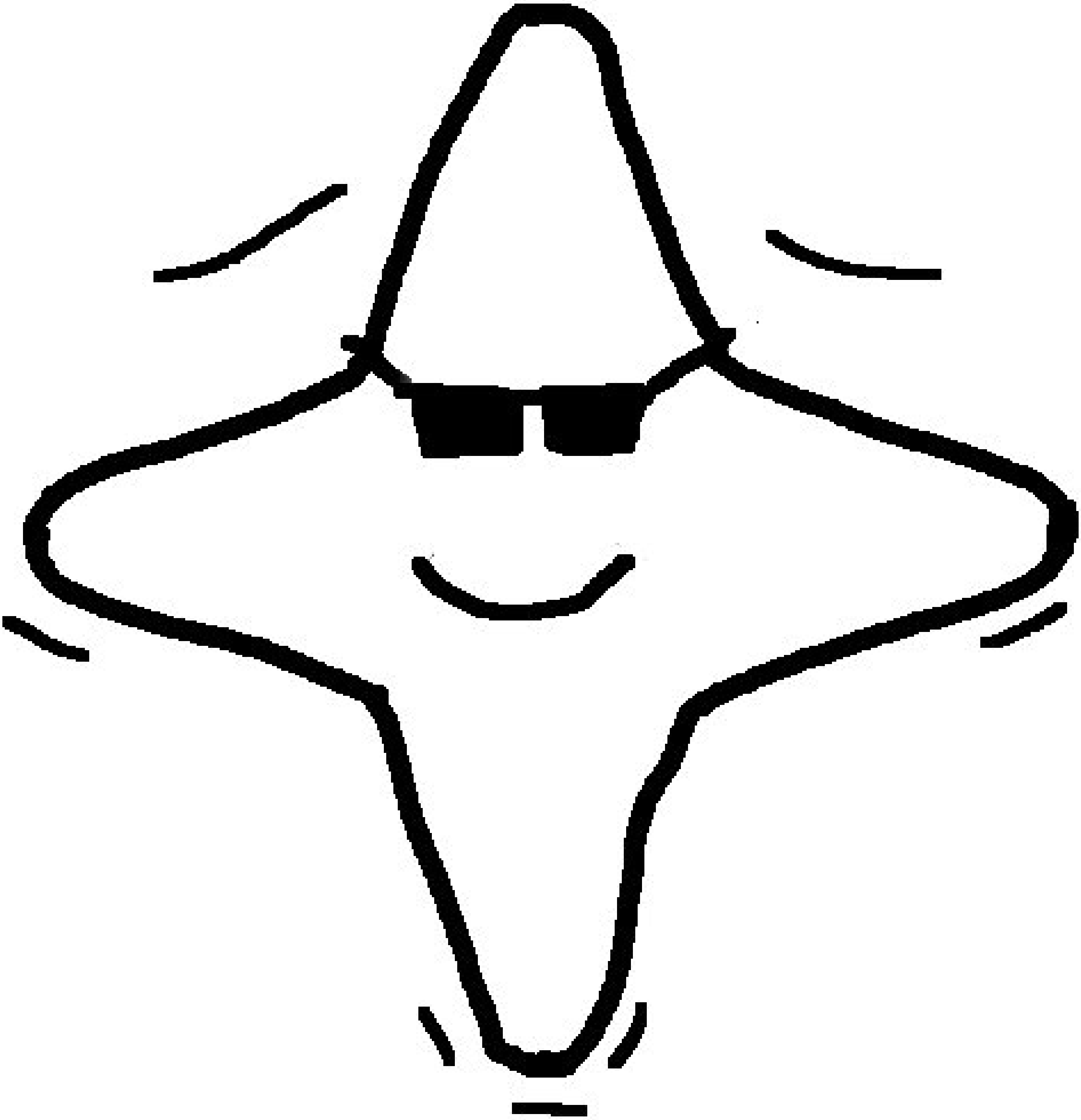
Denominators are never 0.
But need different proof;
"$x^2 + y^2 > 0$" doesn't work.

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.

Edwards curves are cool



ECDSA

Users can sign messages using Edwards curves.

Take a point P on an Edwards curve modulo a prime $p > 2$.

ECDSA signer needs to know the *order of* P.

There are only finitely many other points; about p in total.
Adding P to itself will eventually reach $(0, 1)$; let ℓ be the smallest integer > 0 with $\ell P = (0, 1)$.
This ℓ is the order of P.

The signature scheme has as system parameters a curve E; a base point P; and a hash function h with output length at least $\lfloor \log_2 \ell \rfloor + 1$.
Alice's secret key is an integer a and her public key is $P_A = aP$.

To sign message m,
Alice computes $h(m)$;
picks random k;
computes $R = kP = (x_1, y_1)$;
puts $r \equiv y_1 \bmod \ell$; computes
$s \equiv k^{-1}(h(m) + r \cdot a) \bmod \ell$.

The signature on m is (r, s).

Anybody can verify signature given m and (r, s):
Compute $w_1 \equiv s^{-1}h(m) \bmod \ell$ and $w_2 \equiv s^{-1} \cdot r \bmod \ell$.
Check whether the y-coordinate of $w_1P + w_2P_A$ equals r modulo ℓ and if so, accept signature.

Alice's signatures are valid:

$$
\begin{aligned}
w_1P + w_2P_A &= \\
&(s^{-1}h(m))P + (s^{-1} \cdot r)P_A = \\
&(s^{-1}(h(m) + ra))P = kP
\end{aligned}
$$

and so the y-coordinate of this expression equals r,
the y-coordinate of kP.

Attacker's view on signatures

Anybody can produce an $R = kP$.
Alice's private key is only used in
$s \equiv k^{-1}(h(m) + r \cdot a) \bmod \ell$.

Can fake signatures if one can
break the DLP, i.e., if one can
compute a from P_A.
Most of this course deals with
methods for breaking DLPs.
Sometimes attacks are easier. . .

If k is known for some $m, (r, s)$ then $a \equiv (sk - h(m))/r \bmod \ell$.

If two signatures $m_1, (r, s_1)$ and $m_2, (r, s_2)$ have the same value for r: assume $k_1 = k_2$; observe $s_1 - s_2 = k_1^{-1}(h(m_1) + ra - (h(m_2) + ra))$; compute $k = (s_1 - s_2)/(h(m_1) - h(m_2))$. Continue as above.

If bits of many k's are known (biased PRNG) can attack $s \equiv k^{-1}(h(m) + r \cdot a) \bmod \ell$ as hidden number problem using lattice basis reduction.

Malicious signer

Alice can set up her public key so that two messages of her choice share the same signature,
i.e., she can claim to have signed m_1 or m_2 at will:
$R = (x_1, y_1)$ and $-R = (-x_1, y_1)$ have the same y-coordinate.
Thus, (r, s) fits $R = kP$,
$s \equiv k^{-1}(h(m_1) + ra) \bmod \ell$ and
$-R = (-k)P$,
$s \equiv -k^{-1}(h(m_2) + ra) \bmod \ell$ if
$a \equiv -(h(m_1)+h(m_2))/2r \bmod \ell$.

Malicious signer

Alice can set up her public key so that two messages of her choice share the same signature,
i.e., she can claim to have signed m_1 or m_2 at will:
$R = (x_1, y_1)$ and $-R = (-x_1, y_1)$ have the same y-coordinate.
Thus, (r, s) fits $R = kP$,
$s \equiv k^{-1}(h(m_1) + ra) \bmod \ell$ and
$-R = (-k)P$,
$s \equiv -k^{-1}(h(m_2) + ra) \bmod \ell$ if
$a \equiv -(h(m_1)+h(m_2))/2r \bmod \ell$.

(Easy tweak: include bit of x_1.)