Elliptic-curve cryptography II

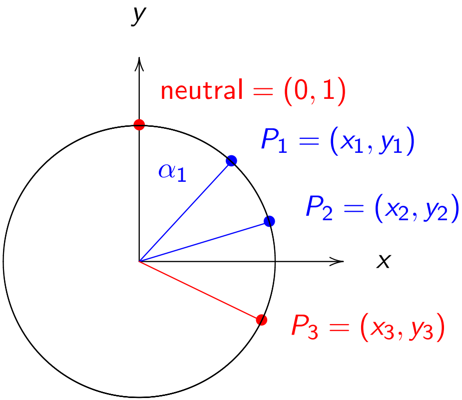
Clocks over finite fields

Tanja Lange

Eindhoven University of Technology

2MMC10 – Cryptology

Addition on the clock



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

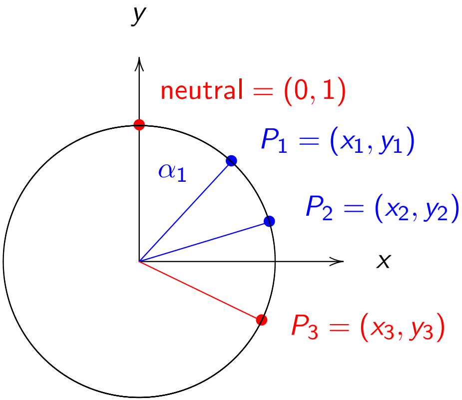
$$(x_1, y_1) + (0, 1) = (x_1, y_1).$$
$$(x_1, y_1) + (-x_1, y_1) = (0, 1).$$

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

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

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

Addition on the clock



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

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

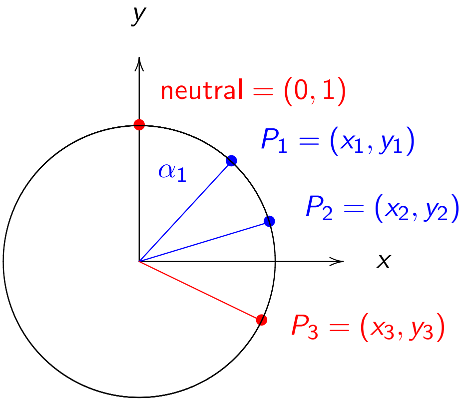
$$2\left(\frac{3}{5}, \frac{4}{5}\right) = \left(\frac{24}{25}, \frac{7}{25}\right).$$

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

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

Problem: The coordinates show a clear growth!
$625 = 5^4$ leaks 4 and pattern continues.

Addition on the clock



$$(x_1, y_1)+(x_2, y_2) = (x_3, y_3)$$
$$= (x_1y_2 + y_1x_2, y_1y_2 - x_1x_2).$$

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

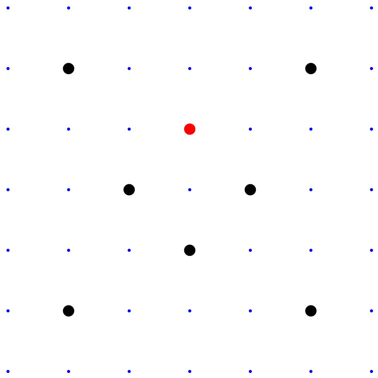
$$2\left(\frac{3}{5}, \frac{4}{5}\right) = \left(\frac{24}{25}, \frac{7}{25}\right).$$

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

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

Problem: The coordinates show a clear growth!
$625 = 5^4$ leaks 4 and pattern continues.
Solution: Consider points on clock modulo a prime.

Clocks over finite fields

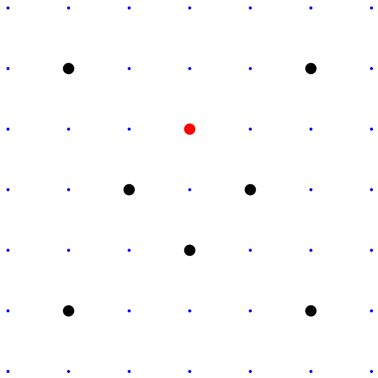


$\mathrm{Clock}(\mathbf{F}_7) =$
$\{(x, y) \in \mathbf{F}_7 \times \mathbf{F}_7 : x^2 + y^2 = 1\}$.

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 $\mathbf{F}_7$.

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 $\mathbf{F}_7$.

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

works also for coordinates in $\mathbf{F}_7$.

Clock cryptography

The "Clock Diffie–Hellman protocol":

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

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

They use $a(b(x, y)) = b(a(x, y))$ as shared secret.

Bigger example

Let $p = 1000003$. Let $P = (1000, 2)$

.

Bigger example

Let $p = 1000003$. Let $P = (1000, 2)$

.

Check $x^2 + y^2 = 1000^2 + 2^2 = 1000004 \equiv 1 \bmod 1000003$.

Bigger example

Let $p = 1000003$. Let $P = (1000, 2)$

.

Check $x^2 + y^2 = 1000^2 + 2^2 = 1000004 \equiv 1 \bmod 1000003$.

$2P = (4000, 7)$
$3P = (15000, 26)$
$6P = 3P + 3P = (780000, 1351)$

Alice picked some secret a, computed $aP = (947472, 736284)$.

Bigger example

Let $p = 1000003$. Let $P = (1000, 2)$

.

Check $x^2 + y^2 = 1000^2 + 2^2 = 1000004 \equiv 1 \bmod 1000003$.

$2P = (4000, 7)$
$3P = (15000, 26)$
$6P = 3P + 3P = (780000, 1351)$

Alice picked some secret a, computed $aP = (947472, 736284)$.

What is Alice's secret?

Double-and-add method

How to compute aP?

```
a = 44444 # our super secret scalar. No, not that one.
l = a.nbits()
A = a.bits()
R = P
for i in range(l-2,-1,-1):
  R = 2 R
  if A[i] == 1:
    R = R + P
print(R)
```

Double-and-add method

How to compute aP?

```
a = 44444 # our super secret scalar. No, not that one.
l = a.nbits()
A = a.bits()
R = P
for i in range(l-2,-1,-1):
  R = 2 R
  if A[i] == 1:
    R = R + P
print(R)
```

This is basically Horner's rule. E.g. $a = 11 = 2^3 + 2 + 1 = (1011)_2$.

$i = 2$: bit is 0, $R = 2P$.

Double-and-add method

How to compute aP?

```
a = 44444 # our super secret scalar. No, not that one.
l = a.nbits()
A = a.bits()
R = P
for i in range(l-2,-1,-1):
  R = 2 R
  if A[i] == 1:
    R = R + P
print(R)
```

This is basically Horner's rule. E.g. $a = 11 = 2^3 + 2 + 1 = (1011)_2$.

$i = 2$: bit is 0, $R = 2P$.
$i = 1$: bit is 1; $R = 4P$; $R = 4P + P = 5P$.

Double-and-add method

How to compute aP?

```
a = 44444 # our super secret scalar. No, not that one.
l = a.nbits()
A = a.bits()
R = P
for i in range(l-2,-1,-1):
  R = 2 R
  if A[i] == 1:
    R = R + P
print(R)
```

This is basically Horner's rule. E.g. $a = 11 = 2^3 + 2 + 1 = (1011)_2$.

$i = 2$: bit is 0, $R = 2P$.
$i = 1$: bit is 1; $R = 4P$; $R = 4P + P = 5P$.
$i = 0$: bit is 1; $R = 10P$; $R = 10P + P = 11P$.

Warnings

Warning #1:
Many p are unsafe!

Warnings

Warning #1:
Many p are unsafe!

Warning #2:
Clocks aren't elliptic!
We will see what problems this brings later.
Next sessions: elliptic curves.

Warnings

Warning #1:
Many p are unsafe!

Warning #2:
Clocks aren't elliptic!
We will see what problems this brings later.
Next sessions: elliptic curves.

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.

Do not use double-and-add as stated on previous page!
Calls to `if` leak bit pattern of a.