Fortgeschrittene Techniken der Kryptographie

Jürgen Fuß

Episode 9: Elliptische Kurven in der Kryptographie

Elliptische Kurven über $\mathbb{Z}_p$

Elliptische Kurven über $\mathbb{Z}_p$

Die Formeln für elliptische Kurven kann man auch verwenden, wenn man die Koordinaten der Punkte als Restklassen modulo $p \in \mathbb{P}$ betrachtet, in den Formeln kommen nur die Grundrechenoperationen $+$, $-$, $\cdot$ und $/$ vor, die ja auch für solche Restklassen funktionieren.

Satz

Es sei $p \in \mathbb{P}$ und $p > 3$. Weiterhin seien $a, b \in \mathbb{Z}_p$ so, dass $4a^3 + 27b^2 \neq 0$ und $\mathcal{E}: \; y^2 = x^3 + ax + b \pmod{p}$ eine elliptische Kurve über $\mathbb{Z}_p$ und $P = (x_1, y_1)$ und $Q = (x_2, y_2)$ zwei Punkte auf $\mathcal{E}$. Dann lassen sich die Koordinaten (x_3, y_3) von $R := P + Q$ nach den selben Formeln wie zuvor berechnen.

$y^2 = x^3 + ax + b$

(x, y) $(x, -y)$

pMögl. pMögl.

$p^2 + 1$ max.

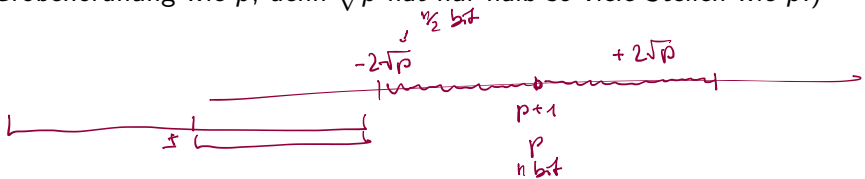
1 min.

Satz von Hasse

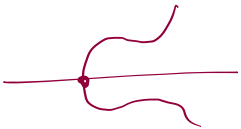
Es sei n die Anzahl der Punkte auf einer elliptischen Kurve modulo p. Dann ist

$$(p+1) - 2\sqrt{p} \leq n \leq (p+1) + 2\sqrt{p}.$$

(D.h. die Anzahl der Punkte auf der elliptischen Kurve ist von der selben Größenordnung wie p, denn $\sqrt{p}$ hat nur halb so viele Stellen wie p.)



Beispiel (1)



Untersuchen wir die elliptische Kurve

$$y^2 = x^3 + 4x + 4 \pmod{29}.$$

Dazu bestimmen wir für jedes $x \in \mathbb{Z}_{29}$ den Wert der rechten Seite der Kurvengleichung.

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
	4	9	20	14	26	4	12	27	26	15	0	16	11	20	20

x	15	16	17	18	19	20	21	22	23	24	25	26	27	28
	17	17	26	21	8	22	11	10	25	4	11	23	17	28

Beispiel (2)

Untersuchen wir die elliptische Kurve

$$y^2 = x^3 + 4x + 4 \pmod{29}.$$

Nun berechnen wir für alle $y \in \mathbb{Z}_{29}$ den Wert der linken Seite der Kurvengleichung.

y	0	1	2	3	4	**5**	6	7	8	9	10	11	12	13	14
	0	1	4	9	16	**25**	7	20	6	23	13	5	28	24	22

y	28	27	26	25	**24**	23	22	21	20	19	18	17	16	15
	1	4	9	16	**25**	7	20	6	23	13	5	28	24	22

Beispiel (3)

Damit lässt sich nun einfach erkennen, dass die folgenden Punkte Elemente der elliptischen Kurve sind.

$$(0,2),(0,27),(1,3),(1,26),(2,7),(2,22),(5,2),(5,27),(10,0),$$
$$(11,4),(11,25),(13,7),(13,22),(14,7),(14,22),(20,14),(20,15),$$
$$(23,5),(\mathbf{23},\mathbf{24}),(24,2),(24,27),(26,9),(26,20),(28,12),(28,17),\infty$$

Die Ordnung der Gruppe ist 26. Dies ist innerhalb der Hasse-Grenzen $(29+1)\pm 2\sqrt{29}$, die hier eine Ordnung zwischen 20 und 40 ergeben.

Beispiel (4)

- Für den Punkt $P = (10, 0)$ ist $2 \cdot P = P + P = \infty$.
- Die Ordnung von P ist 2.
- Punkte auf dieser elliptischen Kurve haben die Ordnungen 1, 2, 13 oder 26, denn dies sind die einzigen Teiler der Gruppenordnung.
- Der Punkt $Q = (2, 7)$ hat die Ordnung 26. Um dies zu verifizieren, muss ausgeschlossen werden, dass $13 \cdot Q = \infty$ und dass $2 \cdot Q = \infty$.
  Dazu berechnet man $2 \cdot Q = (5, 2) \neq \infty$ und
  $13 \cdot Q = 8Q + 4Q + Q = (11, 25) + (23, 5) + (2, 7) = (10, 0) \neq \infty$.
- Somit ist Q erzeugendes Element der Gruppe.
- Der diskrete Logarithmus von $(10, 0)$ zur Basis Q ist 13.

si.py

```
> import si
> from si import EC, Point
> e = EC( 4, 4, 29 )
> si.hasse_bounds( 29 )
(20, 40)
> e.order()
26
> e.list_of_points()
[Point( EC( 4, 4, 29 ), ( 0, 2 ) ),
 Point( EC( 4, 4, 29 ), ( 0, 27 ) ),
 [...]
 Point( EC( 4, 4, 29 ), ( 28, 17 ) ),
 Point( EC( 4, 4, 29 ), None )]
```

$7 \cdot P = 1P + 2P + 4P$

$7 = 2^0 + 2^1 + 2^2$

$1 + 2 + 4$

$2P$

$2(2P) = 4P$

$7P = 8P - 1P$

3 x Verd.

si.py

```
> p = Point( e, (10,0) )
> p.double()
Point( EC( 4, 4, 29 ), None )
> 2*p
Point( EC( 4, 4, 29 ), None )
> q = Point( e, (2,7) )
> q.inverse()
Point( EC( 4, 4, 29 ), ( 2, 22 ) )
> -q
Point( EC( 4, 4, 29 ), ( 2, 22 ) )
> q.double()
Point( EC( 4, 4, 29 ), ( 5, 2 ) )
```

si.py

```
> q.double( verbose=2 )
'       ----------------------------
'       doubling ( 2, 7 ):
'            k = (3*2**2 + 4) / (2*7)
'              = 16 / 14
'              = 16 * 27
'              = 26
'           x3 = 26**2 - 2*2
'              = 5
'           y3 = -7 + 26*(2-5)
'              = -7 + 26*26
'              = 2
'       2 * ( 2, 7 ) = ( 5, 2 )
'       ============================
Point( EC( 4, 4, 29 ), ( 5, 2 ) )
```

si.py

```
> 13*q
Point( EC( 4, 4, 29 ), ( 10, 0 ) )
> q.mult( 13, verbose=1 )
computing 13 * ( 2, 7 )
adding ... doubling ... doubling ...
adding ... doubling ... adding ...
Point( EC( 4, 4, 29 ), ( 10, 0 ) )
> p+q
Point( EC( 4, 4, 29 ), ( 1, 3 ) )
```

```
> p.add( q, verbose=2 )

-----------------------------------
adding ( 10, 0 ) and ( 2, 7 ):
  k = (0-7) / (10-2)
    = 22 / 8
    = 22 * 11
    = 10
 x3 = 10**2 - 10 - 2
    = 1
 y3 = -0 + 10*(10-1)
    = -0  + 10*9
    = 3
( 10, 0 ) + ( 2, 7 ) = ( 1, 3 )
===================================
Point( EC( 4, 4, 29 ), ( 1, 3 ) )
```

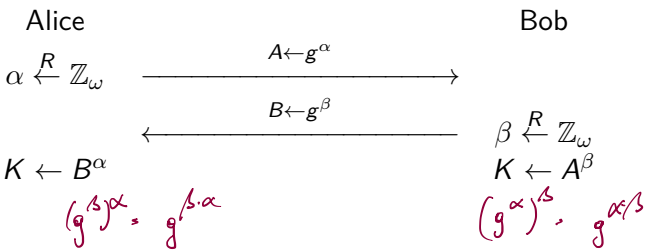
Kryptografische Verfahren mit elliptischen Kurven

Rückblick: Diffie-Hellman mit Gruppen

Diffie-Hellman-Schlüsselaustausch in einer Gruppe $\mathbb{G}$

Setup: Alice und Bob einigen sich auf eine Gruppe $\mathbb{G}$ und auf ein Element $g \in \mathbb{G}$ mit der Ordnung ω. Diese Parameter (Domain Parameter) sind öffentlich.

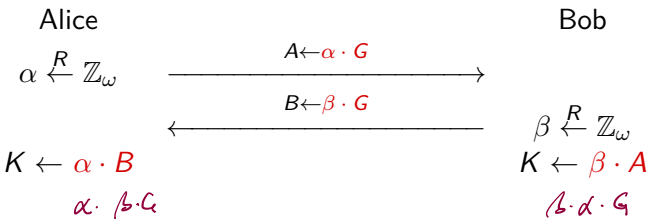
Key Agreement:



Ephemeral ECDH

Setup: Alice und Bob einigen sich auf eine Primzahl p, eine elliptische Kurve $\mathcal{E} : y^2 = x^3 + ax + b$ modulo p und einen **Punkt** G mit primer Ordnung ω auf $\mathcal{E}$. Als Domain Parameter werden $\mathcal{E}$, G, p und ω veröffentlicht.

Key Agreement:



Rückblick: DSA – Domain-Parameter und Schlüsselerzeugung

Setup: Als Hashfunktion wird eine Hashfunktion H aus der SHA-x-Familie verwendet. Eine L Bit große Primzahl p wird vereinbart, so dass $p-1$ einen N Bit großen Primfaktor ω besitzt, weiterhin ein Element g der Ordnung ω in der Gruppe $\mathbb{Z}_p^*$.

Schlüsselerzeugung: Alice wählt zufällig eine Zahl $\alpha \in \mathbb{Z}_\omega$. Sie berechnet

$$A := g^\alpha \bmod p$$

und veröffentlicht ihren Public Key A. Den Private Key α hält sie geheim.

ECDSA (FIPS 186)

$y^2 = x^3 + ax + b \pmod p$

Setup: Hier werden als Domain Parameter eine zumindest 256 Bit lange Primzahl p und eine **elliptische Kurve** $\mathcal{E}$ modulo p vereinbart, deren Ordnung einen großen Primfaktor ω besitzt, weiterhin ein Element G der Ordnung ω auf $\mathcal{E}$. Als Hashfunktion wird eine Hashfunktion H aus der SHA-x-Familie verwendet.

Schlüsselerzeugung: Alice wählt zufällig eine Zahl $\alpha \in \mathbb{Z}_\omega$. Sie berechnet

$$A := \alpha \cdot G$$

und veröffentlicht ihren Public Key A. Den Private Key α hält sie geheim.

Rückblick: DSA – Signieren und verifizieren

Signieren: Um zu signieren, wählt Alice zufällig eine Zahl $k \in \mathbb{Z}_\omega$. Dann berechnet sie die Signatur (r, s) der Nachricht m als

$$r := (g^k \bmod p) \bmod \omega,$$
$$s := k^{-1}(H(m) + \alpha r) \bmod \omega.$$

Verifizieren: Will Bob die Signatur überprüfen, so führt er die folgenden Schritte durch:

1. Er prüft: Ist $1 \leq r < \omega$ und $1 \leq s < \omega$?
2. Er berechnet $x := s^{-1} \cdot H(m) \bmod \omega$ und $y := s^{-1} \cdot r \bmod \omega$.
3. Er prüft: Ist $r = (g^x \cdot A^y \bmod p) \bmod \omega$?

$y^2 = x^3 + ax + b \bmod p$

ECDSA (FIPS 186-5)

Signieren: Um zu signieren, wählt Alice zufällig eine Zahl $k \in \mathbb{Z}_\omega$. Dann berechnet sie die Signatur (r, s) der Nachricht m als

$$r := (k \cdot G)_{x\text{-Koord}} \bmod \omega,$$
$$s := k^{-1}(H(m) + \alpha r) \bmod \omega.$$

Verifizieren: Will Bob die Signatur überprüfen, so führt er die folgenden Schritte durch:

1. Er prüft: Ist $1 \leq r < \omega$ und $1 \leq s < \omega$?
2. Er berechnet $x := s^{-1} \cdot H(m) \bmod \omega$ und $y := s^{-1} \cdot r \bmod \omega$.
3. Er prüft: Ist $r = (x \cdot G + y \cdot A)_{x\text{-Koord}} \bmod \omega$?