Elliptic Curves - Elliptic Nodes

Dawid Karpiński

20.11.2024 r.

Challenge dotyczy wykorzystania ataku opartego na specyficznym przypadku podatności krzywej eliptycznej danej równaniem $y^2 = x^3 + ax + b$. Przy doborze złych parametrów a i b, wielomian posiada podwójny pierwiastek, co prowadzi do powstania **punktu osobliwego**. Ten fakt znacząco upraszcza problem logarytmu dyskretnego.

Podane parametry:

```
p = 4368590184733545720227961182704359358435747188309319510520316493183539079703

# Punkt generujący G
gx = 8742397231329873984594235438374590234800923467289367269837473862487362482
gy = 225987949353410341392975247044711665782695329311463646299187580326445253608

# Punkt Q
qx = 2582928974243465355371953056699793745022552378548418288211138499777818633265
qy = 2421683573446497972507172385881793260176370025964652384676141384239699096612
```

Krzywa eliptyczna jest zadana równaniem ogólnym:

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

Znając dwa punkty na krzywej (w tym przypadku G i Q), można wyznaczyć parametry a i b poprzez rozwiązanie układu dwóch równań:

$$\begin{cases} y_G^2 = x_G^3 + ax_G + b \\ y_Q^2 = x_Q^3 + ax_Q + b \end{cases}$$

Otrzymano więc:

$$a = \frac{y_Q^2 - y_G^2 + x_G^3 - x_Q^3}{x_Q - x_G} \mod p$$

$$b = y_G^2 - x_G^3 - a \cdot x_G \mod p$$

Po obliczeniu parametrów a i b okazało się, że krzywa jest **krzywą osobliwą**, nie spełniając tym samym warunku na krzywą eliptyczną: $4a^3 + 27b^2 = 0$

Aby rozwiązać ten problem, przeprowadzono następujące przekształcenia na podstawie opisu podobnego problemu na forum Crypto Stackexchange[1] .

```
# utworzenie pierścienia wielomianów nad ciałem skończonym
F.<x> = GF(p)[]
f = x ^ 3 + a * x + b

# znalezienie podwójnego pierwiastka
*, (s, *) = f.roots()
```

[1] https://crypto.stackexchange.com/a/61434

```
# przesunięcie wielomianu, aby punkt osobliwy był w (0,0)
f = f.subs(x=x + s)

# przesunięcie współrzędnych punktów
gx = (gx - s) % p
qx = (qx - s) % p
```

Krzywa przeszła z postaci ogólnej do postaci $y^2 = x^2 \cdot (x + c)$, co reprezentuje się jako węzeł eliptyczny (node).

Konsekwencją takiej postaci jest fakt, iż istnieje transformacja pozwalająca na mapowanie problemu dyskretnego logarytmu do multiplikatywnej grupy ciała skończonego F_c^*, co znacząco upraszcza rozwiązanie problemu DLP.

Zatem, funkcja mapująca to:

$$(x, y) \mapsto \frac{y + x\sqrt{c}}{y - x\sqrt{c}}$$

```
# znalezienie pierwiastka wielomianu
(c, *), ** = f.roots()

# obliczenie pierwiastka kwadratowego (mod p)
m = GF(p)(-c).square_root()

# mapowanie do formy multiplikatywnej
u = (gy + m * gx) / (gy - m * gx)
v = (qy + m * qx) / (qy - m * qx)
```

Po przekształceniach można zastosować zwykły algorytm logarytmu dyskretnego:

```
d = discrete_log(v, u)
```

Ostatecznie, na podstawie otrzymanej wartości d, można odszyfrować flagę:

```
from Crypto.Util.number import long_to_bytes
print(long_to_bytes(d).decode("utf-8"))  # == crypto{s1ngul4r_s1mplif1c4t1on}
```