Mathematics - Ellipse Curve Cryptography

Dawid Karpiński

06.11.2024 r.

Krzywa eliptyczna użyta do wygenerowania kluczy publicznych Alice i Boba, to tak naprawdę hiperbola o postaci:

$$x^2 - Dy^2 = 1$$

Równanie zatem można łatwo przekształcić do:

$$(x - \sqrt{D}y)(x + \sqrt{D}y) = 1$$

Następnie, wprowadzając podstawienia: $u = x - \sqrt{D}y$, $v = x + \sqrt{D}y$, otrzymano:

$$uv = 1.$$

Przekształcenie punktu (x, y) na krzywej do postaci (u, v) jest możliwe tylko gdy D jest liczbą, z której można obliczyć pierwiastek.

Postawiony w ten sposób problem logarytmu dyskretnego na hiperboli można rozwiązać w wystarczająco krótkim czasie obliczeniowym.

Parametry użyte w generowaniu kluczy publicznych wzięto ze skryptu załączonego do challengu:

```
p = 173754216895752892448109692432341061254596347285717132408796456167143559
D = 529

# punkt generator
G = Point(
    29394812077144852405795385333766317269085018265469771684226884125940148,
    94108086667844986046802106544375316173742538919949485639896613738390948,
)
```

Podano także wynik działania skryptu z właściwą flagą:

```
A = Point(
    x=155781055760279718382374741001148850818103179141959728567110540865590463,
    y=737947855613466778488107782339018328130726975043353069377993361265037l4,
)
B = Point(
    x=171226959585314864221294077932510094779925634276949970785138593200069419,
    y=543539718395166529385333354761155034368655459663564612927080423053176З0,
)

iv = bytes.fromhex("64bc75c8b38017e1397c46f85d4e332b")
encrypted_flag = bytes.fromhex(
    "13e4d200708b786d8f7c3bd2dc...3e3ff75f7fda9c30a92171bbbc5acbf"
)
```

Do rozwiązania challengu wykorzystano także gotowe funkcje do wykonywania operacji na punktach z krzywej eliptycznej:

```
def point_addition(P, Q):
    Rx = (P.x * Q.x + D * P.y * Q.y) % p
    Ry = (P.x * Q.y + P.y * Q.x) % p
    return Point(Rx, Ry)


def scalar_multiplication(P, n):
    Q = Point(1, 0)
    while n > 0:
        if n % 2 == 1:
            Q = point_addition(Q, P)
        P = point_addition(P, P)
        n = n // 2
    return Q
```

Głównym elementem rozwiązania jest obliczenie logarytmu dyskretnego, aby otrzymać klucz prywatny Alice:

```
n_a = discrete_log(p, v, u)
```

Następnie wystarczy wykorzystać własności krzywych eliptycznych, aby ostatecznie otrzymać wspólny sekret poprzez pomnożenie punktu B, n_a razy i wzięcie współrzędnej x z wyniku:

```
shared = scalar_multiplication(B, n_a).x
```

Na koniec wygenerowano klucz AES używając tych samych operacji ze skryptu podanego razem z challengem (szyfrowanie SHA-1) i otrzymano flagę:

```
key = sha1(str(shared).encode("ascii")).digest()[:16]
print(cipher.decrypt(encrypted_flag))
```

Zatem, w przeciwieństwie do prawdziwych krzywych eliptycznych, użyta hiperbola nie zapewnia wystarczająco silnego bezpieczeństwa kryptograficznego.