**🎯 Mục tiêu:**

Tìm được flag bằng cách **thu hồi lại a** từ phương trình:

hint=gamod  p\text{hint} = g^a \mod phint=gamodp

Rồi sử dụng a để tạo key AES và giải mã ciphertext.

**📄 Thông tin thu được:**

Từ main.py và cycles.txt:

* g=3g = 3g=3
* p=một soˆˊ nguyeˆn toˆˊ lớnp = \text{một số nguyên tố lớn}p=một soˆˊ nguyeˆn toˆˊ lớn
* ciphertext=48 bytes AES ECB\text{ciphertext} = \text{48 bytes AES ECB}ciphertext=48 bytes AES ECB
* hint = 1

**🔍 Phân tích kỹ thuật**

**🔐 Mã hóa AES**

key = long\_to\_bytes(a)[:16]

cipher = AES.new(key, AES.MODE\_ECB)

ct = cipher.encrypt(pad(flag, AES.block\_size))

→ AES sử dụng key từ 16 byte đầu của a.

**❗ Dòng rất quan trọng:**

hint = pow(g, a, p)

Giá trị hint đã được in là 1.

**🧠 Vấn đề toán học cần giải (Discrete Log):**

Bạn cần giải phương trình:

3a≡1mod  p3^a \equiv 1 \mod p3a≡1modp

**🧩 Khi nào ga≡1mod  pg^a \equiv 1 \mod pga≡1modp?**

Khi a≡0mod  ordp(g)a \equiv 0 \mod \text{ord}\_p(g)a≡0modordp​(g), tức là:

a=k⋅ordp(g)a = k \cdot \text{ord}\_p(g)a=k⋅ordp​(g)

Vì g=3g = 3g=3, ta cần tìm **bậc (order)** của 3 modulo p.

Nhưng dễ hơn: Nếu ga≡1mod  pg^a \equiv 1 \mod pga≡1modp, thì a≡0mod  (p−1)a \equiv 0 \mod (p-1)a≡0mod(p−1) (do Fermat's Little Theorem), với điều kiện ggg là nguyên thủy hoặc có bậc cao.

Vì vậy, nếu hint = 1, có nghĩa:

3a≡1mod  p⇒a=k⋅(p−1)3^a \equiv 1 \mod p \Rightarrow a = k \cdot (p-1)3a≡1modp⇒a=k⋅(p−1)

Trước tiên là thử bruteforce:

File bruteforceK.py

**📂 Điều tra thêm**

**Xem lại main.py, biến flag được mã hóa như sau:**

python

CopyEdit

cipher.encrypt(pad(flag, AES.block\_size))

Không có encryption nào khác → plaintext là flag.

**🔎 1. Phân tích bài toán**

Bạn được cung cấp:

* Một số nguyên tố lớn p
* Một cơ sở g = 3
* hint = g^a mod p = 1
* AES ciphertext (48 bytes, mã hóa ECB)
* Mã hóa AES với key lấy từ 16 byte đầu tiên của a (sau khi convert sang bytes)

**Mục tiêu**: Tìm flag, tức là **giải mã ciphertext AES**, mà **không biết a**.

**🧠 2. Ý tưởng then chốt**

Từ hint = 1 ⇒ ta suy ra:

ga≡1mod  pg^a \equiv 1 \mod pga≡1modp

Vì ppp là số nguyên tố, theo định lý Fermat:

gp−1≡1mod  pg^{p-1} \equiv 1 \mod pgp−1≡1modp

⇒ Nếu a=k⋅(p−1)a = k \cdot (p - 1)a=k⋅(p−1) thì:

ga≡1mod  pg^a \equiv 1 \mod pga≡1modp

**=> Đây là điểm yếu:** a bị chọn trúng bội của (p - 1), nên g^a ≡ 1 mod p.

**🛠️ 3. Quy trình giải**

**✅ Bước 1: Nhận ra cấu trúc a = k \* (p - 1)**

* Vì gamod  p=1g^a \mod p = 1gamodp=1
* ⇒ Brute-force k∈[1,N]k \in [1, N]k∈[1,N] sao cho a = k(p-1)

**✅ Bước 2: Tạo AES key**

* Lấy a → bytes → key = first 16 bytes
* Giải mã AES với ECB mode

**✅ Bước 3: Kiểm tra kết quả**

* Nếu plaintext giải được và hợp lý (có thể decode UTF-8), đó là flag.

**🔐 4. Lỗ hổng chính**

* **Lỗi trong lựa chọn a**: chọn a sao cho g^a mod p = 1, tức là a thuộc bội số của (p - 1) — làm cho **logarithm trở nên vô dụng**.
* Với hint = 1, không cần dùng Discrete Log hoặc thuật toán mạnh như Baby-step Giant-step hay Pollard Rho.

**✅ 5. Kết luận**

Đây là bài toán khai thác lỗi trong thiết lập exponent a. Việc g^a mod p = 1 dẫn đến brute-force khả thi trên a = k(p-1), làm mất đi tính một chiều (one-wayness) của phép lũy thừa mô.