RSA - Endless Emails

Dawid Karpiński

19.12.2024 r.

Challenge opiera się na tzw. *Hastad's Broadcast Attack*. Zaszyfrowana wiadomość została wysłana do wielu odbiorców za pomocą kluczy RSA. Każdy odbiorca posiada inny moduł N, ale ten sam wykładnik publiczny $e = 3$, który jest mały.

Zadaniem jest odzyskanie oryginalnej wiadomości, wiedząc, że liczba odbiorców $k \geq e$.

Załóżmy, że Bob chce wysłać wiadomość M do k odbiorców $P_1, P_2, \ldots, P_k$. Każdy odbiorca posiada swój klucz RSA (N_i, e), gdzie N_i to moduł RSA. Bob szyfruje wiadomość dla każdego odbiorcy w następujący sposób:

$$C_i = M^e \mod N_i.$$

Jeśli atakujący przechwyci co najmniej $k \geq e$ takich szyfrogramów oraz zakładamy, że $\gcd(N_i, N_j) = 1$ (dla $i \neq j$), to może zastosować chińskie twierdzenie o resztach (CRT), aby obliczyć wartość $C_0 \in \mathbb{Z}_{N_1 N_2 \ldots N_k}$, która spełnia[1]:

$$C_0 = M^e \mod N_1 N_2 \ldots N_k.$$

Ponieważ wiadomość M jest mniejsza od każdego z N_i, to wiadomo, że $M^e < N_1 N_2 \ldots N_k$. W rezultacie $C_0 = M^e$ w zbiorze liczb całkowitych. Aby odzyskać M, wystarczy obliczyć pierwiastek e-tego stopnia z C_0.

Aby to zrobić, na początek stworzono listę par (N_i, C_i), podanych w zadaniu.

```
pairs = [(192873..., 723984...), ...]
```

Następnie użyto `itertools.combinations`, aby wygenerować wszystkie możliwe kombinacje $e = 3$ par.

```
import itertools

e = 3
combs = list(itertools.combinations(pairs, e))
```

Funkcja `solve_congruence` z biblioteki `sympy` rozwiązuje układ kongruencji za pomocą CRT.

```
from sympy.ntheory.modular import solve_congruence

for comb in combs:
    result = solve_congruence(*comb)
    m3, n123 = result  # m3 = M^3, n123 = N1*N2*N3
```

Funkcja `gmpy2.iroot` oblicza pierwiastek 3-ciego stopnia z $m3$, dzięki czemu odzyskano wiadomość M.

```
import gmpy2
from Crypto.Util.number import long_to_bytes

m = gmpy2.iroot(m3, e)
m = long_to_bytes(m[0]) #

try:
```

[1] Boneh, D. (1999). Twenty Years of Attacks on the RSA Cryptosystem (https://www.ams.org/notices/199902/boneh.pdf)

```
    print(m.decode("utf-8"))
except:
    continue
```

Ostatecznie otrzymujemy odszyfrowaną wiadomość:

> yes
>
> ---
>
> Johan Hastad Professor in Computer Science in the Theoretical Computer Science Group at the School of Computer Science and Communication at KTH Royal Institute of Technology in Stockholm, Sweden.
>
> crypto{1f_y0u_d0nt_p4d_y0u_4r3_Vuln3rabl3}