Criptografia de clave publica

Valentina S. Vispo

Octubre 2023

Índice

1 Desafio: Ataque de Broadcast sobre RSA

1.1 Enunciado

El desafio consiste en descifrar el mensaje secreto mediante un ataque de broadcast.

Se realiza una peticion al servidor en el cual se obtiene un mensaje con la siguiente estructura:

```
{
    "ciphertext": str,  # Codeado en b64
    "publicKey": {
        "n": int,       # Cambia en cada request
        "e": int        # Siempre es 3
    }
}
```

1.2 Analisis

El texto claro es un texto $ASCII$. Tanto el texto claro como el texto cifrado son cadenas de bytes. Para las conversiones desde y hacia números enteros se utiliza la convención *big endian*.

El valor M expresado más arriba resulta de convertir la secuencia de bytes que representa el texto claro en un entero, C es un entero, y el campo *ciphertext* que aparece en la respuesta del servidor es el resultado de codificar en base64 la secuencia de bytes que corresponde a C.

Cada resultado de multiplicar el valor anterior por una constante a, sumarle una constante b y calcular el resto de dividirlo por una constante m.

$$X_i = (aX_{i-1} + b) \mod m \text{ con } 1 \leq i$$

1.3 Solucion

Notas a tener en cuenta

1. Python no posee un desplazamiento a derecha sin signo.
2. Tuve problemas para realizar la raiz cubica en Python. La libreria math desde la version *3.11* en adelante tiene la funcion *cbrt* pero me tiraba el siguiente error: AttributeError: module ‘math‘ has no attribute ‘cbrt‘

Pasos

1. Obtener los textos cifrados con sus respectivas claves publicas. El texto cifrado esta codificado en base64

2. Luego obtengo el n de la clave publica. El e no me interesa obtenerlo de cada mensaje porque siempre es 3 (parte del enunciado)

$$n_1 = n_2 * n_3$$

$$n_2 = n_1 * n_3$$

$$n_3 = n_1 * n_3$$

$$N_i = \frac{N}{n_i}$$

$$N = n_1 * n_2 * n_3 \text{ — Producto de los valores } n \text{ de la clave publica}$$
$$x = ()() + () + ()) \mod N$$

Al obtener $x = m^3$

$$C = M^e \mod n \rightarrow C = M^3 \mod n$$

C: texto cifrado — bytes codeado en b64

M: texto claro — bytes

$e = 3$ — El desafio especifica este valor

1.3.1 Como correr la solución

```
python3 challenges/8_RSA_Broadcast.py
```

1.4 Evitar el problema

Si se cumplen las propiedades necesarias para la seguridad se cumple lo siguiente:

- Es posible hallar e, d y n tal que $M^{de} \equiv M \mod n \forall M < n$
- M^d y C^e son faciles de calcular $\forall M < n$
- Imposible calcular e dados n y d conocidos

Entonces, lo que se debe evitar es darle al usuario el e.

1.5 Herramientas

Como mencione en las notas, tuve problemas para la raiz cubica, y utilice esta funcion modificada para el caso particular:

```
def find_invpow(x,n):
"""Finds the integer component of the n'th root of x,
an integer such that y ** n <= x < (y + 1) ** n.
"""
high = 1
while high ** n <= x:
    high *= 2
low = high/2
while low < high:
    mid = (low + high) // 2
    if low < mid and mid**n < x:
        low = mid
    elif high > mid and mid**n > x:
        high = mid
    else:
        return mid
return mid + 1
```

```
def get_term_i(cipher_text: int, N_i: int, n_i: int) -> int:
"""
cipher_text * N_i (N_i^{-1} mod n_i)
```

```
    """
    return cipher_text * N_i * (inverse_mod(N_i, n_i))
```

1.6 Referencias

1. RSA-Kisbye
2. A Practical Attack on Broadcast RC4
3. https://stackoverflow.com/questions/356090/how-to-compute-the-nth-root-of-a-very-big-