IC-Práctica 1 (PARTE 2) Cifrado / Firmas

Ejercicio 11
Cifrado XOR con clave temporal autogenerada
● *Resolver el reto alojado en el puerto 11015 del sitio ic.catedras.linti.unlp.edu.ar*
Rta: IC{t0d0_3s_x0r34bl3_en_3sta_v1d4}

```
┌──(ic)─(kali㉿kali)-[~]
└─$ nc ic.catedras.linti.unlp.edu.ar 11015.

   ____   __
  / _/__ / /_______
 / // _ \/ __/ __/ _ \
_/ // // / /_/ /_/ // /
/___/_//_/\__/_/  \___/

  ______ __                                                  _     __                __
 / ____(_) /_  ____ _____ ___  ____ ___  ____ ____  ____(_)___/ /___  _____/ /
/ /   / / __ \/ __ `/ __ `__ \/ __ `__ \/ __ `/ _ \/ __/ / __  / __ \/ ___/ /
/ /___/ / /_/ / /_/ / / / / / / / / / / / /_/ /  __/ /_/ / /_/ / /_/ (__  )_/
\____/_/_.___/\__,_/_/ /_/ /_/_/ /_/ /_/\__, /\___/\__/_/\__,_/\____/____(_)
                                        /____/

Bienvenidos! Ejercicio XOR:
Debe mandar el texto plano de este hexstring, encriptada con 4 caracteres, como pista le da
mos que la primer palabra es:
reunir
30333e262b246b382b3c2a2523763f2d30383e3a23762f2d2e302226623524242b382a
Mmmm tardaste mucho amiguito
```

Ejecutó por 2da vez y noto que la palabra siempre cambia y los numeros tambien:

```
┌──(ic)─(kali㉿kali)-[~]
└─$ nc ic.catedras.linti.unlp.edu.ar 11015.

   ____   __
  / _/__ / /_______
 / // _ \/ __/ __/ _ \
_/ // // / /_/ /_/ // /
/___/_//_/\__/_/  \___/

  ______ __                                                  _     __                __
 / ____(_) /_  ____ _____ ___  ____ ___  ____ ____  ____(_)___/ /___  _____/ /
/ /   / / __ \/ __ `/ __ `__ \/ __ `__ \/ __ `/ _ \/ __/ / __  / __ \/ ___/ /
/ /___/ / /_/ / /_/ / / / / / / / / / / / /_/ /  __/ /_/ / /_/ / /_/ (__  )_/
\____/_/_.___/\__,_/_/ /_/ /_/_/ /_/ /_/\__, /\___/\__/_/\__,_/\____/____(_)
                                        /____/

Bienvenidos! Ejercicio XOR:
Debe mandar el texto plano de este hexstring, encriptada con 4 caracteres, como pista le da
mos que la primer palabra es:
astuto
181001040d0c5510090207100d0c5516180d1619164317100d06071818430518130218 10
Mmmm tardaste mucho amiguito
```

El servidor me da:
Un hexstring cifrado y la pista que la primera palabra del texto plano siempre cambia.

Script → tp1_ejer11Optimizado.py

```python
from pwn import remote
import re

def deducir_clave(palabra, hexstring):
```

```
    texto_cifrado = bytes.fromhex(hexstring)
    bytes_pista = palabra.encode()
    xor_segmento = [texto_cifrado[i] ^ bytes_pista[i] for i in
range(len(bytes_pista))]
    clave = xor_segmento[:4]
    return bytes(clave)

def descifrar_mensaje(hexstring, clave):
    texto_cifrado = bytes.fromhex(hexstring)
    clave_repetida = (clave * (len(texto_cifrado) // len(clave) +
1))[:len(texto_cifrado)]
    return bytes([c ^ k for c, k in zip(texto_cifrado, clave_repetida)])

def main():
    conn = remote("ic.catedras.linti.unlp.edu.ar", 11015)
    data = conn.recv(timeout=2).decode()

    match_palabra = re.search(r"la primer palabra es:.*?\n(\w+)", data,
re.IGNORECASE)
    match_hex = re.search(r"([0-9a-f]{8,})", data)

    if not match_palabra or not match_hex:
        print(" No se encontro la palabra pista o el hexstring.")
        return

    palabra = match_palabra.group(1)
    hexstring = match_hex.group(1)

    clave = deducir_clave(palabra, hexstring)
    mensaje = descifrar_mensaje(hexstring, clave)

    conn.sendline(mensaje)
    respuesta = conn.recv(timeout=2).decode(errors="replace")

    print(f"[ Respuesta del servidor]:\n{respuesta}")

    flag = re.search(r"IC\{.*?\}", respuesta)
    if flag:
        print(f"[FLAG ENCONTRADO] {flag.group(0)}")
    else:
        print(" No se encontro ningún flag con formato IC{...}")

if __name__ == "__main__":
    main()
```

Ejercicio 12
Revele el mensaje cifrado con RSA:
p:141168104496224770047142463070837492564875854409388187 7
q:102547776473911617023200175596292656948983894912123276 7
e:65537
C:244800329353906336350382253088680972646706962639783844335948234085022348400763256559770095538177770365047075

Rta: IC{sabiendo_P_y_Q_es_muy_facil}
p: primitivo grande
q: primitivo grande
e: exponente público
C: mensaje cifrado

Paso 1: calcular n y phi(n)
n= p*q
phi = (p-1) * (q-1)

Paso 2: .Calcular la clave privada d
Usamos el inverso multiplicativo de e módulo phi(n):

```
from Crypto.Util.number import inverse
d = inverse(e, phi)
```

Paso 3: Descifrar el mensaje → usamos la fórmula RSA

```
M = pow(C, d, n)
```

(esto te da el mensaje en forma de número decimal)

Paso 4: Convertir el número a texto

```
mensaje_bytes = M.to_bytes((M.bit_length() + 7) // 8, byteorder='big')
mensaje = mensaje_bytes.decode(errors='replace')
```

Script → tp1_ejer12.py

```
from Crypto.Util.number import inverse
p = 14116810449622477004714246307083749256487585440938818777
q = 10254777647391161702320017559629265694898389491212327677
e = 65537
C = 244800329353906336350382253088680972646706962639783844335948234085022348400763256559770095538177770365047075
n = p * q
phi = (p - 1) * (q - 1)
d = inverse(e, phi)
M = pow(C, d, n)
mensaje_bytes = M.to_bytes((M.bit_length() + 7) // 8, byteorder='big')
mensaje = mensaje_bytes.decode(errors='replace')
```

```
print("[+] Mensaje descifrado:")
print(mensaje)
```

Ejercicio 13

● *Resolver el reto alojado en el puerto 11012 del sitio ic.catedras.linti.unlp.edu.ar*

Rta: IC{rsa_is_eeeeeeasy}

Pistas: es RSA nos dieron los 2 primitivos grandes (p y q), el exponente público (e) y el mensaje cifrado (C) . **Tener en cuenta que estos datos cambian en cada conexión.**

Script → tp1_ejer13.py

```
from Crypto.Util.number import inverse
from pwn import remote
import re

def extraer_parametros(data):
```

```
    try:
        p = int(re.search(r"p\s*=\s*(\d+)", data).group(1))
        q = int(re.search(r"q\s*=\s*(\d+)", data).group(1))
        e = int(re.search(r"e\s*=\s*(\d+)", data).group(1))
        c = int(re.search(r"c\s*=\s*(\d+)", data).group(1))
        return p, q, e, c
    except AttributeError:
        print(" No se pudo encontrar uno de los parámetros. Texto
recibido:")
        print(repr(data))
        raise

def descifrar_rsa(p, q, e, c):
    n = p * q
    phi = (p - 1) * (q - 1)
    d = inverse(e, phi)
    m = pow(c, d, n)
    mensaje_bytes = m.to_bytes((m.bit_length() + 7) // 8, 'big')
    return mensaje_bytes

def main():
    conn = remote("ic.catedras.linti.unlp.edu.ar", 11012)
    data = conn.recv(timeout=2).decode(errors="replace")

    try:
        p, q, e, c = extraer_parametros(data)
    except Exception as ex:
        print("Error al extraer parámetros:", ex)
        conn.close()
        return

    mensaje_bytes = descifrar_rsa(p, q, e, c)
    mensaje = mensaje_bytes.decode(errors="replace").strip()

    conn.sendline(mensaje.encode())
    respuesta = conn.recv(timeout=2).decode(errors="replace")

    print("[DEBUG] Mensaje descifrado:", mensaje)
    print("[DEBUG] Respuesta del servidor:\n", respuesta)

    flag = re.search(r"IC\{.*?\}", respuesta)
    if flag:
        print("[FLAG ENCONTRADO]", flag.group(0))
    else:
        print("No se encontró ningún flag en la respuesta.")
```

```
    conn.close()

if __name__ == "__main__":
    main()
```

Ejercicio 14
Revele el mensaje cifrado con RSA, esta vez no tenemos P ni Q.
Pista: Hay que factorizar o encontrar un buen lugar donde lo hagan…
n: 145244918462453563575744908598820448749422224850949389929975 9
e: 65537
C: 1280743944712857143060627969938538851911171*950125979945026152*

Rta: ***IC{factordb_ftw}***
Me dan n (módulo), e (exponente público) y C (el mensaje cifrado). Esta vez no dan ni p ni q. O sea hay que factorizar.

Paso 1: Factorizar → uso https://factordb.com/ pego n y si ya fue factorizado, me devuelve p y q

Result:		
status (?)	digits	number
FF	61 (show)	1452449184...59<61> = 1153324775179431312178120797679<31> · 125935834890789310817539157152 1<31>

p = 1153324775179431312178120797679
q= 125935834890789310817539157152 1
y hago n = 1153224775179413121712807916 79 × 125953853408789310189351152111

Paso 2: Calculas phi(n) y d

```
from Crypto.Util.number import inverse

p = 1153324775179431312178120797679
q = 125935834890789310817539157152 1
e = 65537
n = p * q
phi = (p - 1) * (q - 1)
d = inverse(e, phi)
```

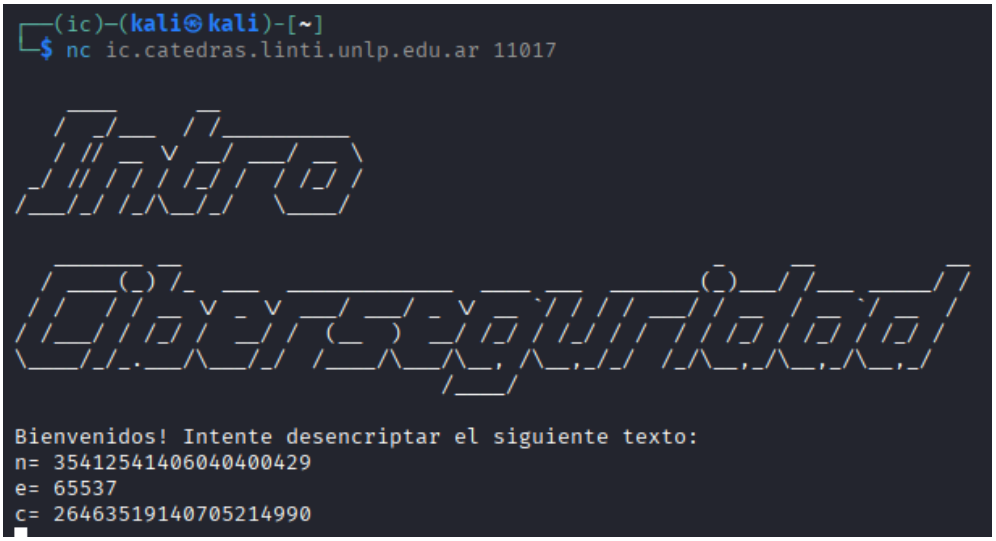
Paso 3: descifrar el mensaje

```
c = 1280743944712857143060627969938538851911171950125979945026152
m = pow(c, d, n)
mensaje_bytes = m.to_bytes((m.bit_length() + 7) // 8, 'big')
mensaje = mensaje_bytes.decode(errors="replace")
print("[+] Mensaje descifrado:", mensaje)
```

Script → tp1_ejer14.py

```python
from Crypto.Util.number import inverse

# Paso 1: valores dados
p = 11533247751794313121781207976 79
q = 12593583489078931081753915715 21
e = 65537
c = 12807439447128571430606279699385388519111719501259799450261 52

# Paso 2: calcular n, phi y d
n = p * q
phi = (p - 1) * (q - 1)
d = inverse(e, phi)

# Paso 3: descifrar el mensaje
m = pow(c, d, n)
mensaje_bytes = m.to_bytes((m.bit_length() + 7) // 8, 'big')
mensaje = mensaje_bytes.decode(errors="replace")

print("[+] Mensaje descifrado:")
print(mensaje)
```

Ejercicio 15
Desencriptar RSA sin p ni q
● *Resolver el reto alojado en el puerto 11017 del sitio ic.catedras.linti.unlp.edu.ar*



Pista:el servidor genera un nuevo reto RSA distinto de n, e y c.en cada conexión

hay algo en tp1_ejer15.py pero no anda porque n no tiene factores suficientemente cercanos como para que el método de Fermat lo pueda factorizar dentro del límite de iteraciones que definimos (max_iter=100000).
Probar talvez con otro metodo de factorizacion, probar mas tarde hora no tengo mas ganas.

Ejercicio 16
Diffie-Hellman
● *Resolver el reto alojado en el puerto 11018 del sitio ic.catedras.linti.unlp.edu.ar*

Ejercicio 17
Utilice la herramienta steghide para encontrar el mensaje oculto en la imagen.

Ejercicio 18
Encuentre el mensaje oculto en el archivo de audio.

Ejercicio 19
Utilice el diccionario indicado en el desafío publicado en la plataforma CTFd para crackear un archivo encriptado utilizando PGP con criptografía simétrica. Ver challenge ej20 - symmetric pgp

Ejercicio 20
Encriptar asimétrico: Encriptar con PGP el archivo "encriptar.txt" con la clave pública dada, realizar el submit del archivo encriptado en formato armor.