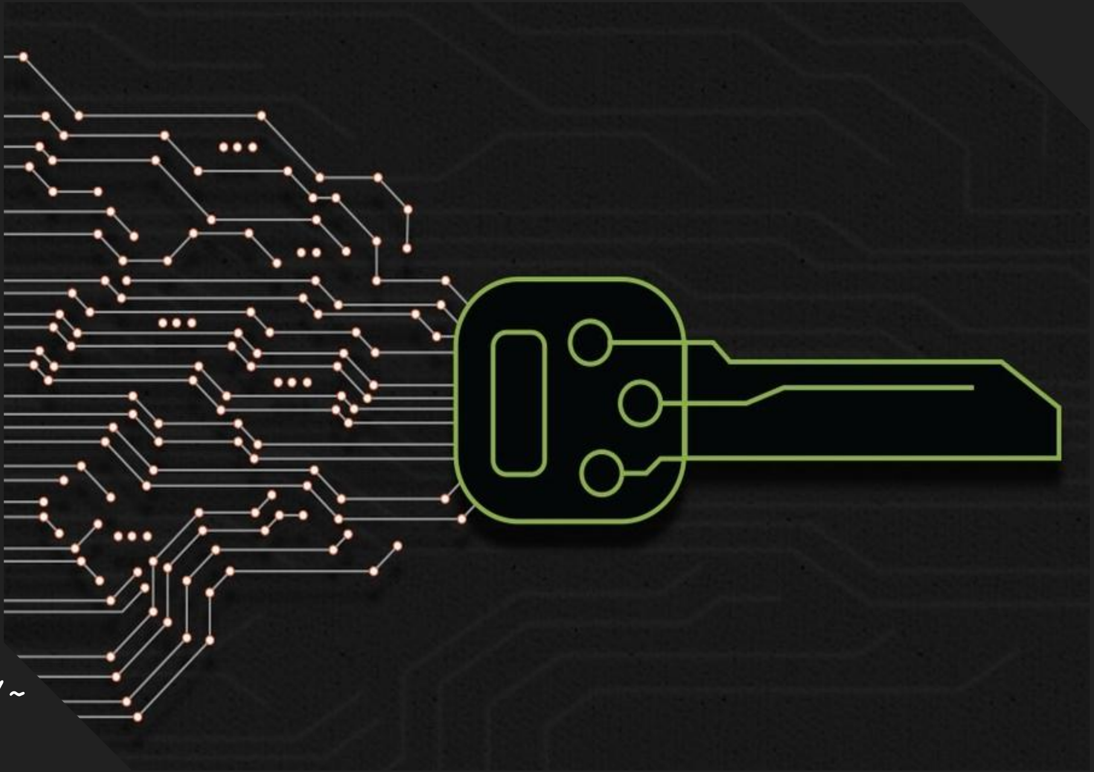
Day 2 : Cryptography



~ [CCSC] CIT Cyber Security Cell ~

OUSSAMA RAHALI

OMAR AOUAJ

cat README.md

Presentation outline

Warning !

As we head through this meeting, we're gonna have some challenges for you to answer.. If you were able to solve one, please write DONE in the chat without writing the solution.

Don't spoil solutions on your friends :) !

Challenges

Challenge name : Back in Time

>> *Find me.*

7wc0Fmcn52bD9VehN1XuF2QfV2Vfd3bO9Vev9WW

Challenges

Challenge name : Scanner

[+] my name is 64>>1

NB2HI4DTHIXS62LCMIXGG3ZPK5THARDDJBRQ

RSA

Notions

RSA, first described in 1977, is the most famous public-key cryptosystem.
It has two main use-cases:

- Public key encryption enables a user, Alice, to distribute a public key and others can use that public key to encrypt messages to her. Alice can then use her private key to decrypt the messages.
- Digital signatures enable Alice to use her private key to "sign" a message. Anyone can use Alice's public key to verify that the signature was created with her corresponding private key,.

Components

n: decimal number (used in the modulo, must be big for more security)
e: exponent (must be >= 3)
m: plain text
c: cipher text

Before we begin

Since the plaintext is a string (mostly), we must convert it to an integer so that it can be compatible with the arithmetic operations of the encryption, the ciphertext will also be an integer.

How ??

we'll go from base 256 to base 10

Before we begin

Given the string “hello”, we’ll convert it from base 256 to base 10 as the following:

`ord(‘o’)*(256**0)+ord(‘l’)*(256**1)+...`

ord is a python function which gives the ascii number of a character.

Yet!! For a very long string, it’s agony. So there’s a function in python called `bytes_to_long` in the library `Crypto.Util.number` which does the same thing.

Before we begin

Given the string “hello”, we’ll convert it from base 256 to base 10 as the following:

`ord(‘o’)*(256**0)+ord(‘l’)*(256**1)+...`

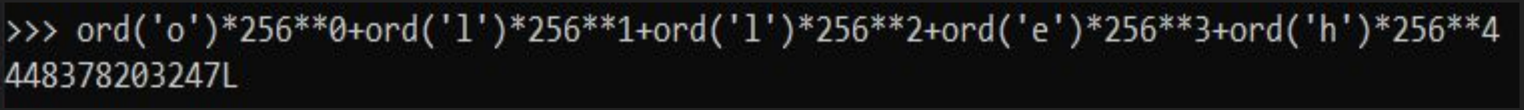
ord is a python function which gives the ascii number of a character.

Yet!! For a very long string, it’s agony. So there’s a function in python called bytes_to_long in the library Crypto.Util.number which does the same thing. It’s inverse is long_to_bytes.

Encrypting

Suppose m="hello"

1- convert m to base 10



```
>>> ord('o')*256**0+ord('l')*256**1+ord('l')*256**2+ord('e')*256**3+ord('h')*256**4
448378203247L
```

m is now 448378203247

2- encrypting:

Given n and e (= public key):

c = m**e [n] (m to the power of e modulo n)

c is the cipher text and it's a decimal number

Decrypting

Having c,e and the prime decomposition of n (p and q with n=p*q), equivalent of private key.

1- calculate phi=(p-1)*(q-1)

2- compute the modular inverse of phi

    d with d*e=1[phi]

3- calculate m:

    m = c**d [n]

4- convert m from base 10 to base 256:

    using Crypto.Util.number.long_to_bytes

Illegal Decrypting

Without knowing the prime decomposition of n, you can exploit a weak RSA encryption and find p and q by many attacks covered by RsaCTFTool

You can also be lucky and find the prime decomposition of n in factordb.

And also it depends on the situation, maybe you can have a relation between p and q, and therefore solving a second degree equation… It always depends.

Implement w/ Python

- https://lmgtfy.app/?q=RSA

Hands on lab

Challenge : R-SAYYYY

>>

```
n=88256459553622414063962598765941602942
6239230804614613279163
e = 65537
cipher=348226401953293105681528097015649
9240126029191801966285141813
```

HASHing

Hash is any function of encryption which can’t be reversible.. There are many hashes as SHA1, SHA2, SHA256, SHA512, MD5, PKZIP, BCRYPT,...

HASHing

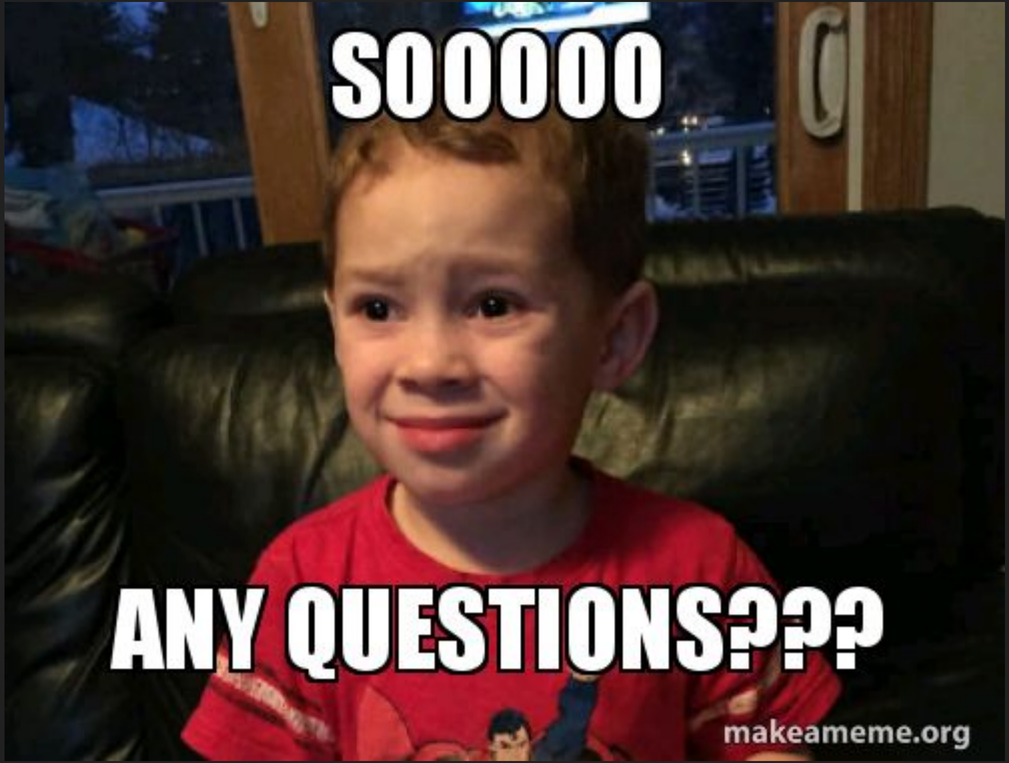
create a md5 hash :

```
echo -n "badboy" | md5sum
```

Crack the md5 hash :

https://crackstation.net/

shutdown



ls -al .Contact_us

OUSSAMA RAHALI

OMAR AOUAJ

Facebook : /oussama.rahali.925

Facebook : /omar.aouaj.77