



# I. Criptografía avanzada

---

Javier Sánchez, Alejandro Bermejo, Inés Martín, Carlos Alonso y Sergio Pérez

# Índice

1.XOR

2.RSA

3.AES - CBC

4.Primeros pasos Python

## XOR

- Cifrado que opera XOR en binario
- Utiliza una **clave secreta**.
- Como la **longitud** de la **clave** suele ser **menor al texto**, se repetirá **cíclicamente**.

### Propiedades



**1. Conmutativa:**  $A \text{ xor } B = B \text{ xor } A$

**2. Asociativa:**  $(A \text{ xor } B) \text{ xor } C = A \text{ xor } (B \text{ xor } C)$

**3. Autoinversa:**  $(A \text{ xor } B) \text{ xor } B = A$

<i>A</i>	<i>B</i>	XOR
0	0	0
0	1	1
1	0	1
1	1	0

## Reto 7 Atenea

XOR (4pts)  

**Dificultad:** ★☆☆☆☆

En criptografía, el cifrado XOR es, como su nombre indica, un algoritmo de cifrado basado en el operador binario XOR:

$A \text{ xor } 0 = A$   
 $A \text{ xor } A = 0$   
 $(B \text{ xor } A) \text{ xor } A = B$

Una cadena de texto puede ser cifrada aplicando el operador de bit XOR sobre cada uno de los caracteres utilizando una clave. Para descifrar la salida, sólo hay que volver a aplicar el operador XOR con la misma clave.

Usa la clave **encryptXOR** para descifrar el siguiente mensaje:

**UGFzc3dvcnQ6IHhvFzYMAcEfbIAgIA==**

Recuerda que la respuesta hay que ponerla en el formato correcto: flag{md5}

Referencias:  
[https://es.wikipedia.org/wiki/Cifrado\\_XOR](https://es.wikipedia.org/wiki/Cifrado_XOR)  
<http://xor.pw>  
<https://conv.darkbyte.ru>



<http://xor.pw/>  
<https://gchq.github.io/CyberChef/>



## EJEMPLO I

¡Ayúdanos a descifrar este texto! Conocemos la correspondencia de algunas cadenas:

"cifrado muy utilizado" = 0c 2d 03 3e 00 31 3d 6a 2e 36 2d 66 16 1b 04 1c 0c 0e 08 10 06

"propiedades importantes" = 3c 13 3a 22 23 26 27 35 22 06 1c 4d 19 08 04 06 06 1d 17 01 30 00  
3f

06 38 66 3b 20 3f 50 00 07 49 01 07 56 0c 2d 03 3e 00 31 3d 6a 2e 36 2d 66 16 1b 04 1c 0c 0e  
08 10 06 56 0a 2a 45 20 00 75 31 38 2a 33 20 29 04 1d 0c 16 0c 15 63 20 00 13 01 21 45 3c 13  
3a 22 23 26 27 35 22 06 1c 4d 19 08 04 06 06 1d 17 01 30 00 3f 6b 16 27 2b 2d 27 3b 66 17 06  
08 1e 00 07 49 01 07 56 0c 2d 03 3e 00 31 3d 6a 1b 0c 06 66 1a 4f 0e 1f 0b 1b 0a 11 1a 56 1f 25  
17 38 04 75 36 2f 2f 63 20 23 1b 1b 02 50 00 1a 49 17 05 17 1d 2b 45 3c 0e 31 20 ab 30 63 35  
36 0f 06 0e 11 17 54 05 15 1a 56 1f 36 0a 3c 08 30 36 2b 27 26 27 66 07 0a 01 50 3d 3b 3b 54  
19 17 1d 25 45 3f 00 36 33 38 63 2f 35 66 00 03 0c 06 00 58 49 54 0d 13 1c 27 0c 2a 13 34 20 26  
2c 63 2d 66 00 00 03 03 00 13 1c 1d 1b 56 03 25 45 2a 0d 34 35 40 16 11 1e 05 18 37 22 22 45  
11 1a 54 0f 17 0c 2d 09 31

## DADO UN PAR TEXTO EN CLARO/TEXTO CIFRADO

Aplicando la propiedad autoinversa del XOR podremos descifrar la clave

$$(A \text{ xor } B) \text{ xor } B = A$$

**Texto en claro:**  $A$

**Texto cifrado:**  $(A \text{ xor } K)$



**Clave:**  $\text{Texto} \text{ xor } \text{Cifrado}$

**Clave:**  $A \text{ xor } (A \text{ xor } K) = K$

## EJEMPLO II

**Esta vez nos dan directamente la flag, pero parece que está cifrada:**

Flag: 13 3e 2b 24 3d 3d 14 02 66 1f 04 47 34 09 11 0e 32 0d 41 0b 27 4c 02 0b 27 1a 04 47 28  
03 41 02 35 4c 0c 12 3f 4c 12 02 21 19 13 08 3b

**Formato de la flag: URJC{}**

## ¿Cómo lo resolvemos?

Conocemos el formato de la flag...

CIFRADO	FLAG
13	U
3e	R
2b	J
24	C
3d	{
3d	?
14	?
02	?
66	?
...	...

$$(A \text{ xor } B) \text{ xor } B = A$$

**Tenemos:**

Cifrado = Texto[i] XOR Clave[i]

Flag = Texto[i]

**Entonces:**

Cifrado[i] XOR Texto[i] = Clave[i]



# I. XOR

## ¿Cómo lo resolvemos?

CIFRADO		FLAG		CLAVE
13		U		k[0]
3e		R		k[1]
2b		J		k[2]
24		C		k[3]
3d	XOR	{	=	k[4]
3d		?		?
14		?		?
02		?		?
66		?		?
...		...		...

$$(A \text{ xor } B) \text{ xor } B = A$$

**Tenemos:**

Cifrado = Texto[i] XOR Clave[i]

Flag = Texto[i]

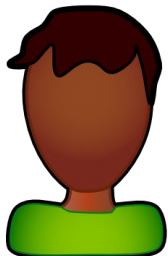
**Entonces:**

Cifrado[i] XOR Texto[i] = Clave[i]

## EJEMPLO III: Alice y Bob quieren intercambiar mensajes

¡Te mando la contraseña secreta cifrada con mi clave!

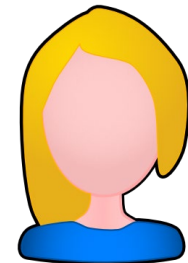
16 3e 2b 35 1e 06 11 0a 1e 0c 0e 00 43 63 08 0e 05 45 25 00 00 17 16 54 0d 50 63 0f 0d 17 13 36 45  
0b 13 06 11 41 40 36 09 41 05 00 73 06 02 1c 06 11 0d 54 3e



Bob

Te la mando de vuelta cifrada con mi clave

5a 5f 78 50 79 73 7f 6e 7f 47 6b 79 0f 02 5b 6b 62 30 4b 64 61 5c 73 2d 41 31 30 6a 6a 62 7d 52 24 40  
76 7f 5d 20 13 53 6e 34 6b 64 12 4d 67 65 4a 70 5e 31 59



Alice

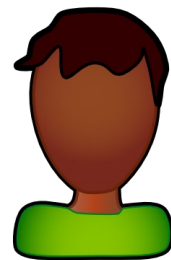
Genial, voy a volver a cifrar otra vez con mi clave

19 33 19 26 1c 20 1a 0d 0d 22 1f 18 3e 41 37 0a 14 55 18 01 02 2e 16 59 20 00 73 06 0b 14 18 01 41  
23 04 1a 29 41 22 10 02 55 1d 01 41 28 04 17 2f 04 3f 00 1a



# I. XOR

¿Qué está pasando?



Bob



K1

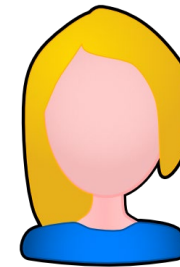
$$M1 = \text{Texto XOR } K1$$



$$M2 = M1 \text{ XOR } K2$$



$$M3 = M2 \text{ XOR } K1$$



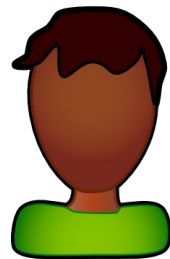
Alice



K2

# I. XOR

¿Qué está pasando?



Bob



K1

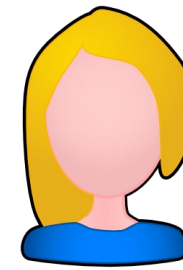
$$M1 = \text{Texto XOR } K1$$



$$M2 = (\text{Texto XOR } K1) \text{ XOR } K2$$



$$M3 = ((\text{Texto XOR } K1) \text{ XOR } K2) \text{ XOR } K1$$



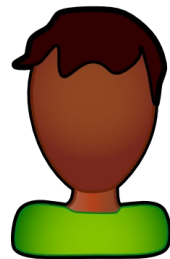
Alice



K2

# I. XOR

¿Qué está pasando?



Bob



K1

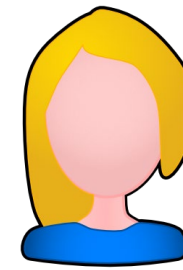
$$M1 = \text{Texto XOR } K1$$



$$M2 = (\text{Texto XOR } K1) \text{ XOR } K2$$



$$M3 = ((\text{Texto XOR } K1) \text{ XOR } K2) \text{ XOR } K1$$



Alice



K2

## ¿Cómo descifrarlo?

Si hacemos **M2 XOR M3** conseguiremos K1, y con la clave ya podremos descifrar el primer mensaje (M1)



Bob

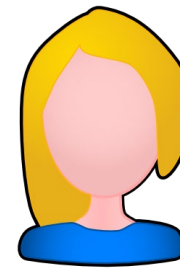


K1

$$M1 = \text{Texto XOR } K1$$

$$M2 = (\text{Texto XOR } K1) \text{ XOR } K2$$

$$M3 = \text{Texto XOR } K2$$



Alice

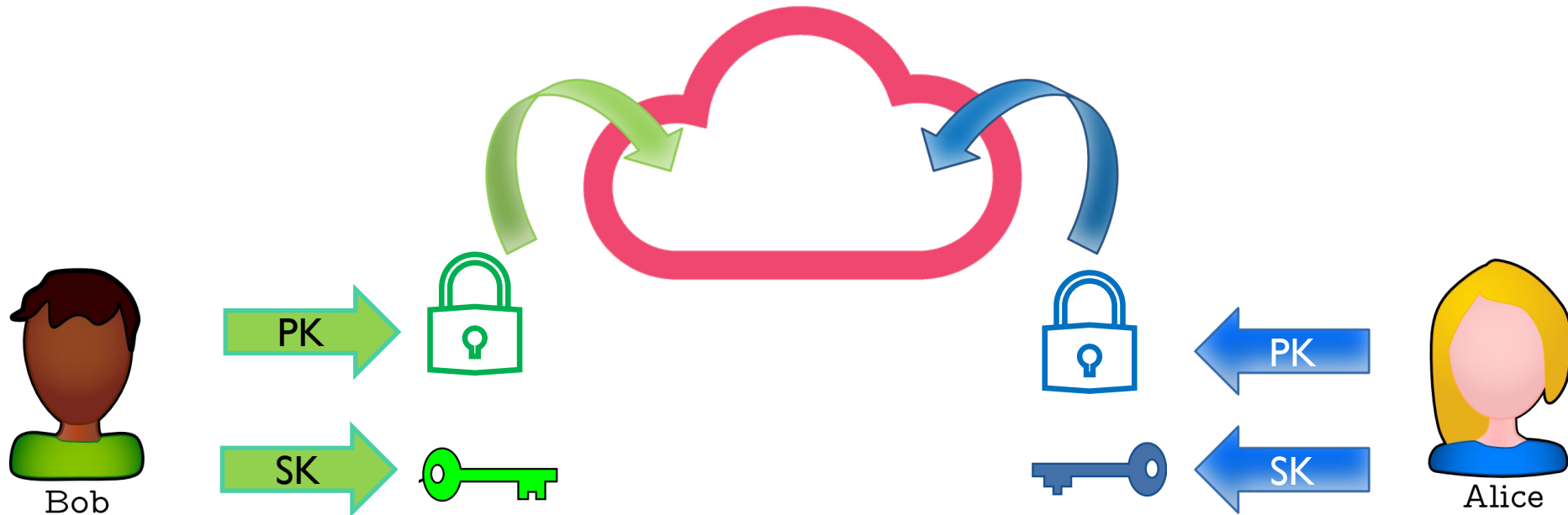


K2

## ¿Qué es la criptografía asimétrica?

Es un tipo de cifrado que utiliza una clave pública para cifrar y otra privada para descifrar.

RSA o el Gamal



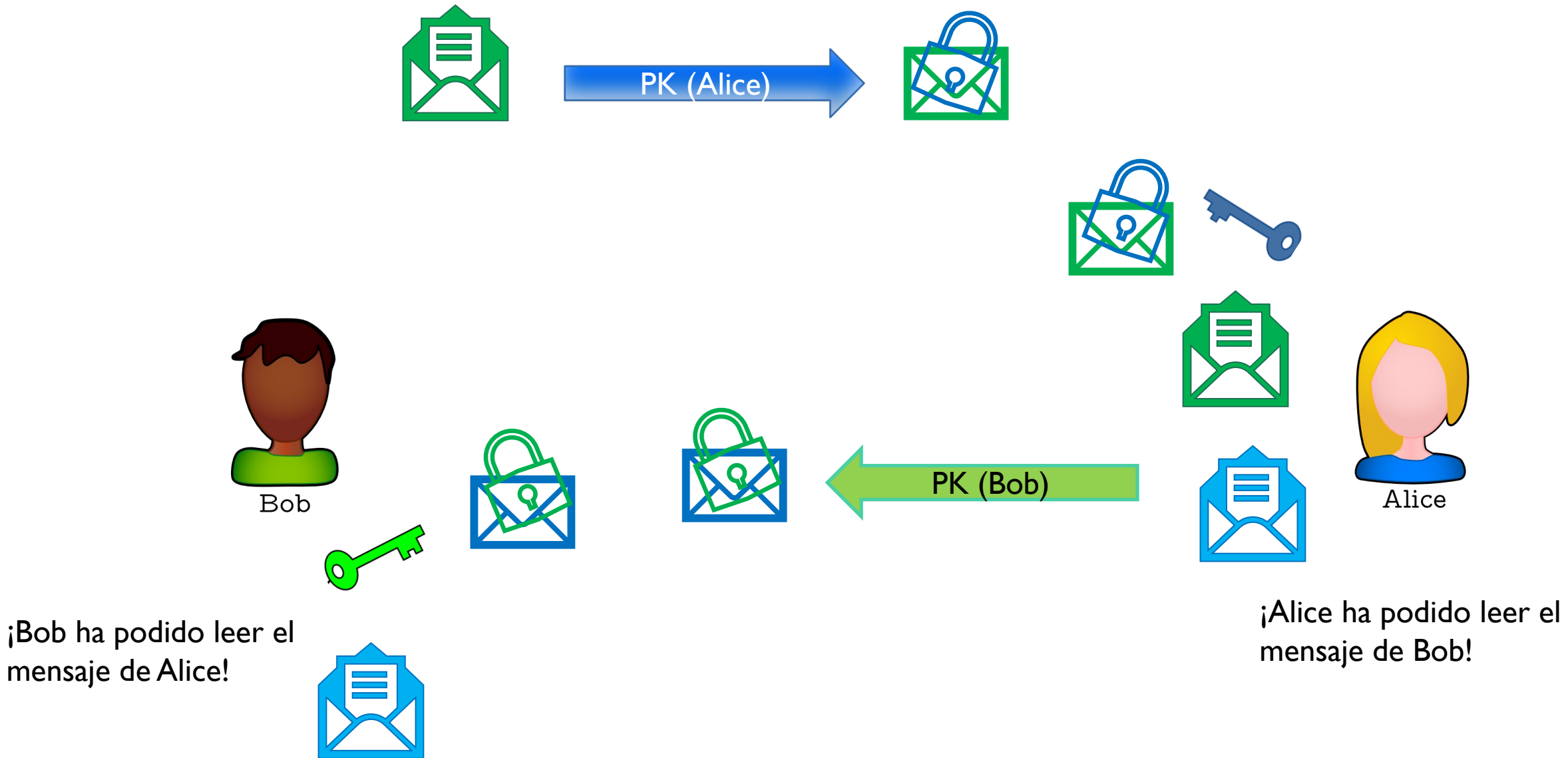
# Criptografía - Asimétrica

**Bob quiere mandar un mensaje seguro a Alice  
pero no han acordado ninguna clave secreta  
previamente**

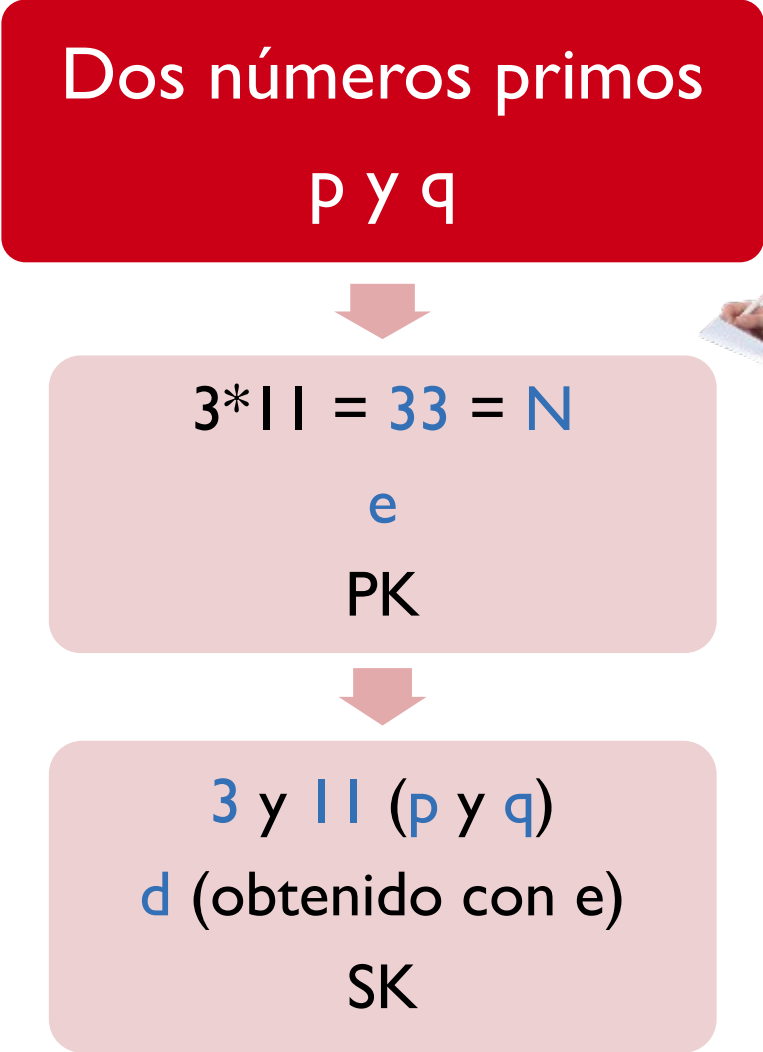
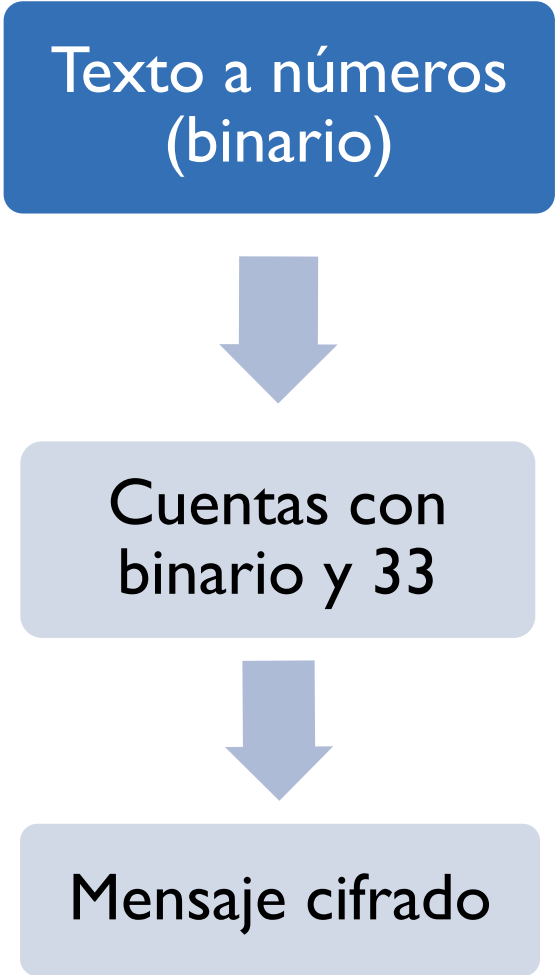
**¿Cómo lo hacen?**



# Criptografía - Asimétrica



# RSA – (Rivest – Shamir –Adleman)



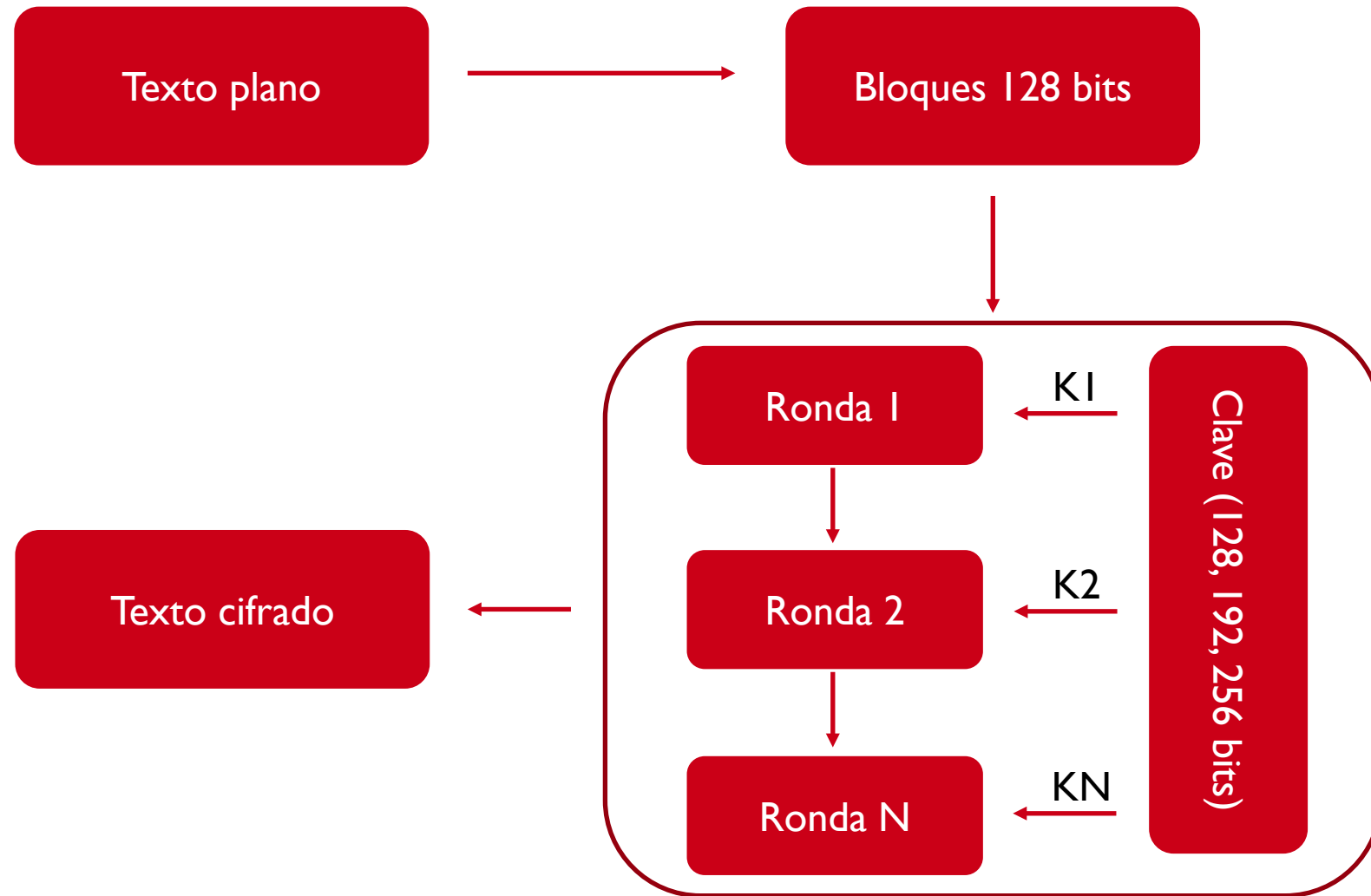
# RsaCtfTool

**Instalación en Kali Linux (OVA de VirtualBox):**

```
git clone https://github.com/Ganapati/RsaCtfTool.git
sudo apt-get install libgmp3-dev libmpc-dev
cd RsaCtfTool
pip3 install -r "requirements.txt"
python3 RsaCtfTool.py
```

[RsaCtfTool: https://github.com/Ganapati/RsaCtfTool](https://github.com/Ganapati/RsaCtfTool)

# AES – (Advanced Encryption Standard)



# AES – (Advanced Encryption Standard)

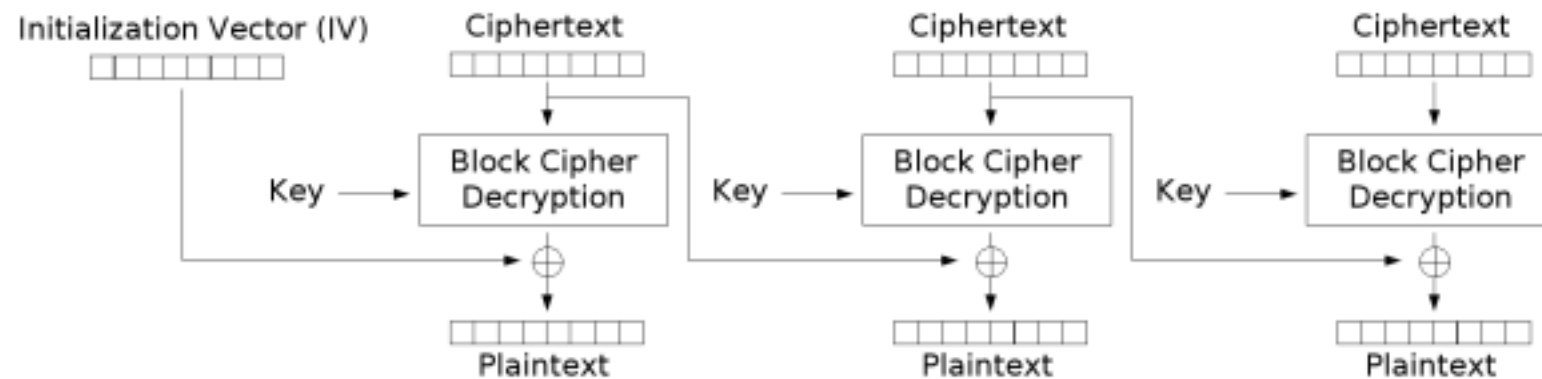
- En cada ronda, se calcula una **nueva clave** a partir de la **original**
- El **número de rondas** depende de la **longitud de la clave**

LONGITUD DE LA CLAVE (bits)	RONDAS
128	10
192	12
256	14

- Lo que ocurre en cada una de esas rondas, queda a vuestra curiosidad

# AES – CBC (*Cipher Block Chaining*)

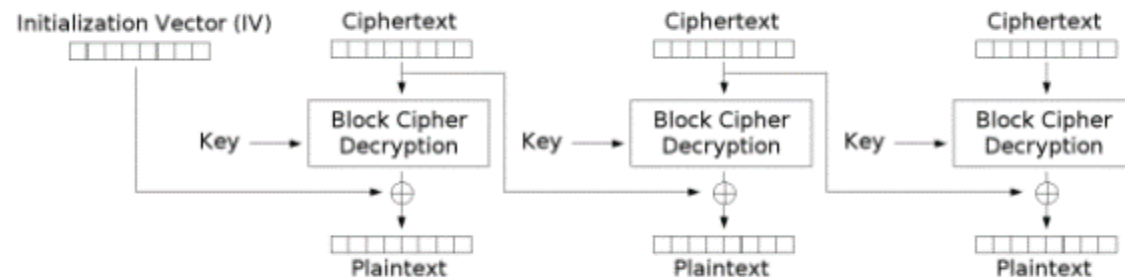
- Lo que acabamos de ver es el **modo básico** de AES (**ECB**)
- Es más interesante el **modo CBC** (típico en **CTF**)



Cipher Block Chaining (CBC) mode decryption

# AES – CBC (*Cipher Block Chaining*)

- Se añade el **vector de inicialización (IV)**
- Sirve para hacer **cada mensaje único**
- **A cada bloque de texto se le aplica una operación XOR con el bloque previo ya cifrado.**
  - **Antes de ser cifrado**
- **Cada bloque depende del anterior**



Cipher Block Chaining (CBC) mode decryption

# ¿AES CtfTool?

- Lamentablemente no, **nos toca trabajar a mano**
- O utilizar Cyberchef si sabemos el **modo de cifrado** empleado, la **clave** utilizada y el **IV**
  - Esta situación **no es común**, aunque a veces hay retos “secuenciales” en los que resolviendo otra parte del reto obtienes los datos de cifrado (suele ser **MISC**)
- Lo habitual es aplicar **ataques conocidos sobre AES** o atacar una **mala implementación** del mismo



# Primeros pasos en python

**Python es una herramienta fundamental para automatizar tareas en retos CTF**

**No podemos explicar Python desde 0, pero para los que tengáis curiosidad dejamos [en la web un script](#) que hace muchas de las codificaciones y cifrados que hemos visto**



# Para seguir aprendiendo...

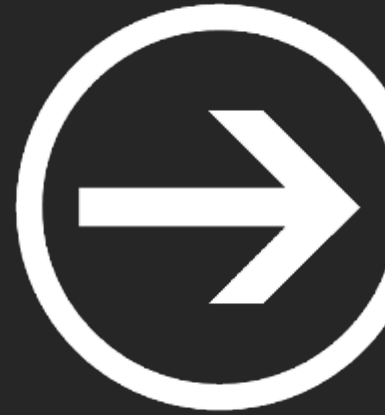
## Recursos de consulta y práctica

- CryptoHack. Retos de criptografía:

<https://cryptohack.org/challenges/>

- CrypTool. RSA paso a paso:

<https://www.cryptool.org/en/cto/rsa-step-by-step.html>



# I. Criptografía avanzada

---

Javier Sánchez, Alejandro Bermejo, Inés Martín, Carlos Alonso y Sergio Pérez