



I. Criptografía avanzada

Santiago Tovar y Elia Seco

1. Cifrado simétrico
 - a. Cifrados de flujo vs de bloque
 - b. AES
 - i. ECB
 - ii. CBC
2. Cifrado asimétrico
 - a. RSA

OBJETIVO DE LA CLASE



CRIPTOGRAFÍA SIMÉTRICA

¿Qué es la criptografía simétrica?

Es un tipo de cifrado en el que se utiliza la misma clave tanto para cifrar como para descifrar un mensaje.

VENTAJAS

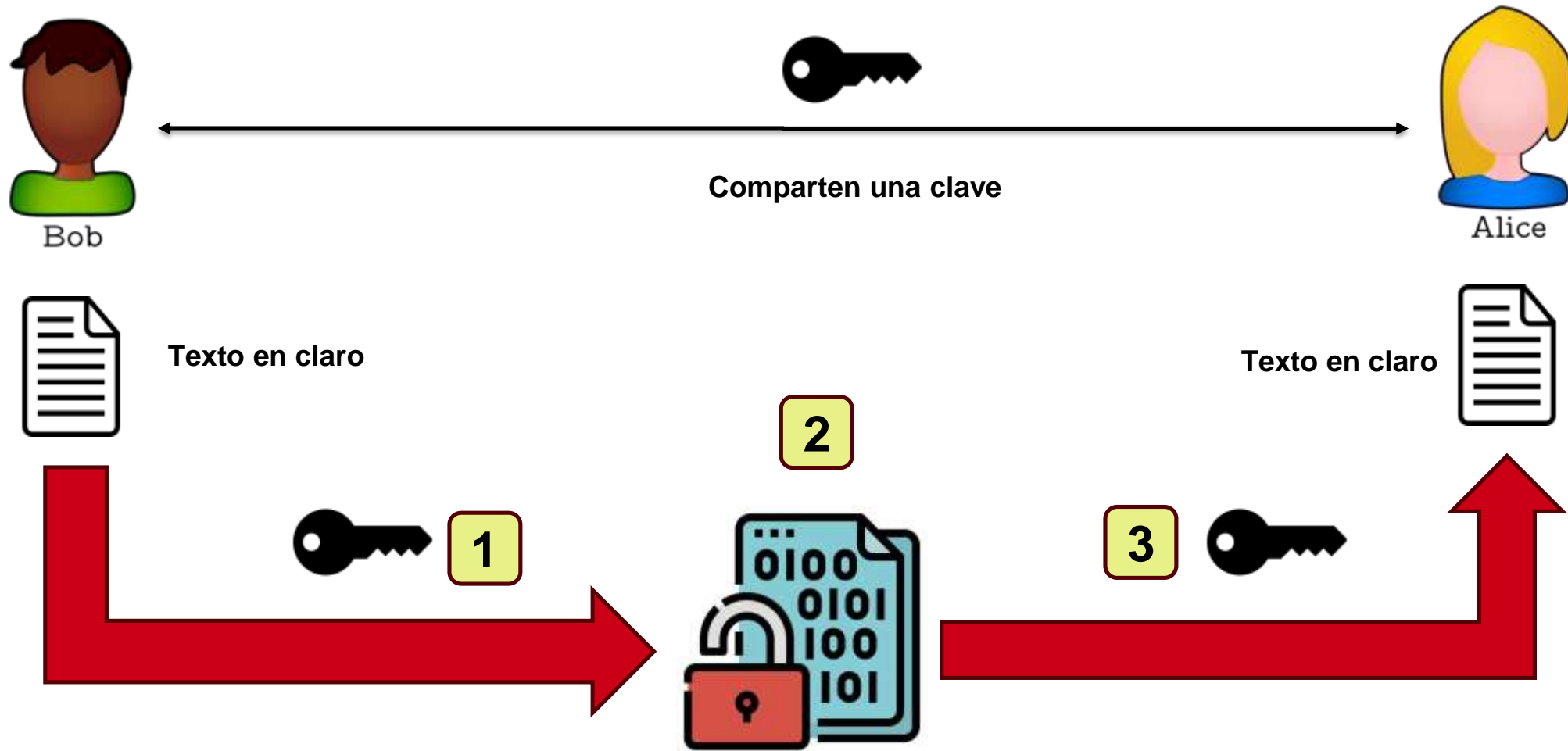
- Muy fácil de usar
- Muy útil
- Rápida y eficiente
- Segura

DESVENTAJAS

- ¿Cómo compartimos la clave?
- Demasiadas claves

CRIPTOGRAFÍA SIMÉTRICA

¿Cómo funciona?



CRIPTOGRAFÍA SIMÉTRICA: FLUJO VS BLOQUE

FLUJO

- Cifrado bit a bit
- $\text{XOR}(\text{bit}, \text{key_bit})$
- Más ligero
- Más difícil de implementar de forma segura
- No reutilizar claves

BLOQUE

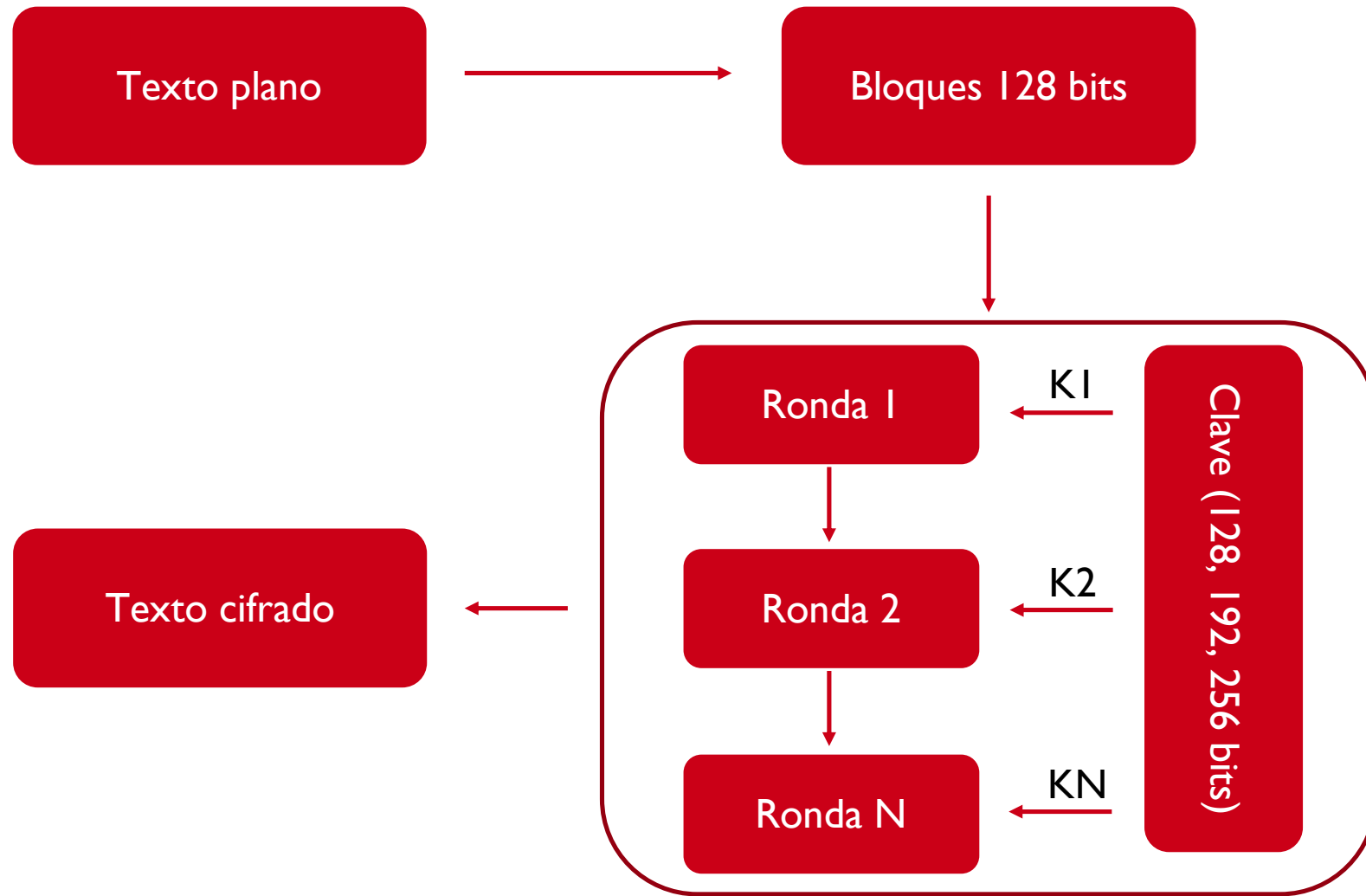
- Cifrado por bloques
- AES, DES
- Más pesado
- Más fácil de implementar de forma segura
- Se pueden reutilizar claves (¡EN ALGUNOS MODOS!)

AES – (Advanced Encryption Standard)

¿Qué es?

- Algoritmo de cifrado simétrico de bloque
- Se utiliza como estándar global de encriptación
- Se utiliza en aplicaciones como WhatsApp o BitLocker
- 128, 192 y 256 bits
- Evolución de DES --> más lento y más inseguro (clave corta)

¿Cómo funciona AES?



¿Cómo funciona AES?

- 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

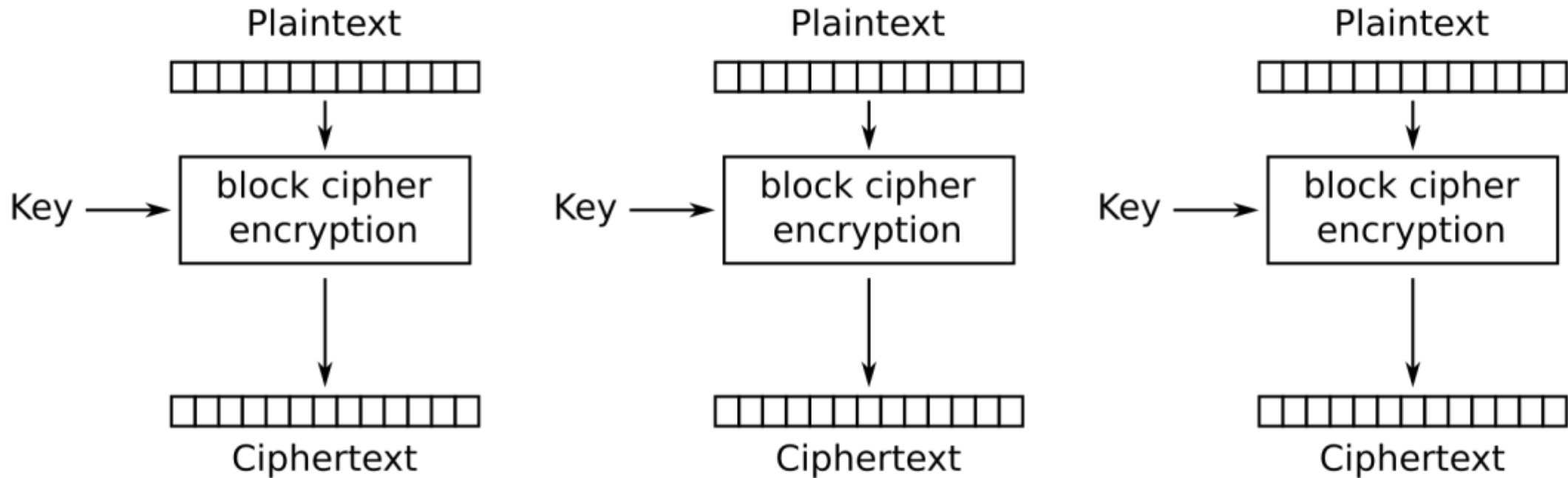
- Lo que ocurre en cada una de esas rondas, queda a vuestra curiosidad

Crypto en python

DISCLAIMER: Bullshit code

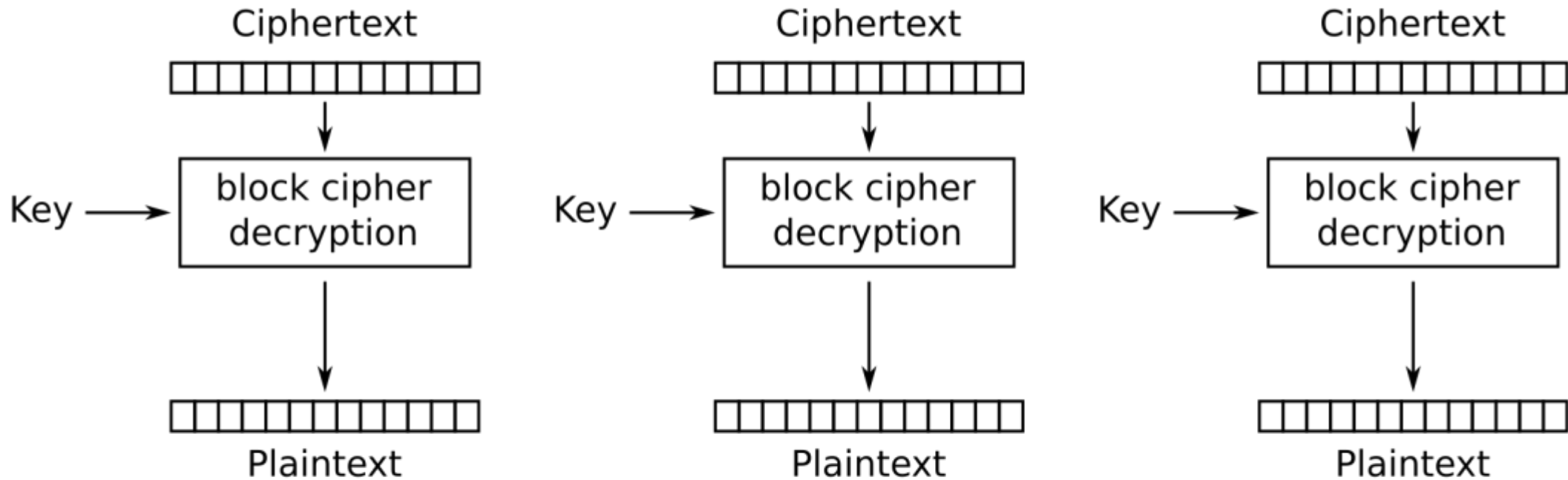
```
1  #pip install pycryptodome
2  ✓ from Crypto.Cipher import AES # type: ignore
3  from Crypto.Util.Padding import pad, unpad # type: ignore
4
5  BLOCK_SIZE = 16
6  KEY = b'd84ea284923ed09d' # 16 Bytes
7  plaintext = b'Hola, voy a cifrar este mensaje'
8  plaintext = pad(plaintext, BLOCK_SIZE) # Padding para alinear al size del bloque
9  cipher = AES.new(KEY, AES.MODE_ECB) # Nuevo cifrador en modo ECB
10 ciphertext = cipher.encrypt(plaintext) # Ciframos el mensaje
11
12 print(ciphertext)
13 print(unpad(cipher.decrypt(ciphertext), BLOCK_SIZE).decode())
14
15 ✓ ''' OUTPUT
16 b'p3\xf1\xb8\xf6i\xce+\xee\x9bK\xbd\xb9\x8bAj\xbf#\xe7\xfdw\xbb\xd7.%\xb0\xa3\xb6\x8a\xce\xbf_'
17 Hola, voy a cifrar este mensaje
18 '''
```

¿Qué es el modo ECB?



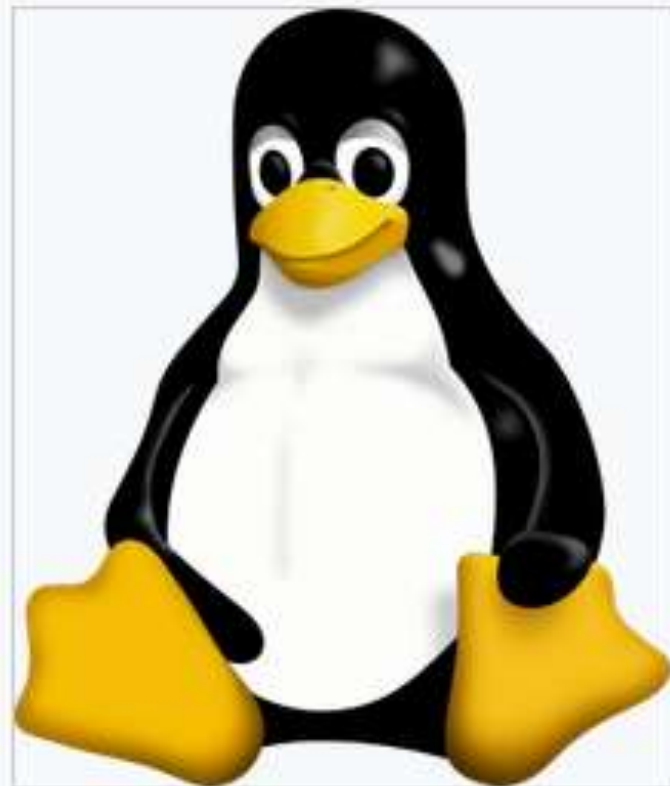
Electronic Codebook (ECB) mode encryption

¿Qué es el modo ECB?



Electronic Codebook (ECB) mode decryption

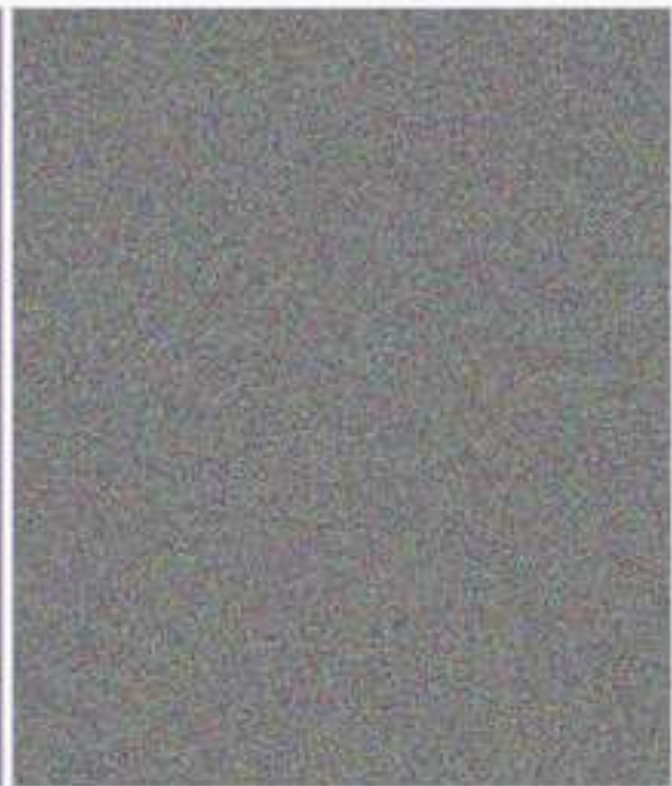
¿Qué es el modo ECB?



Original image

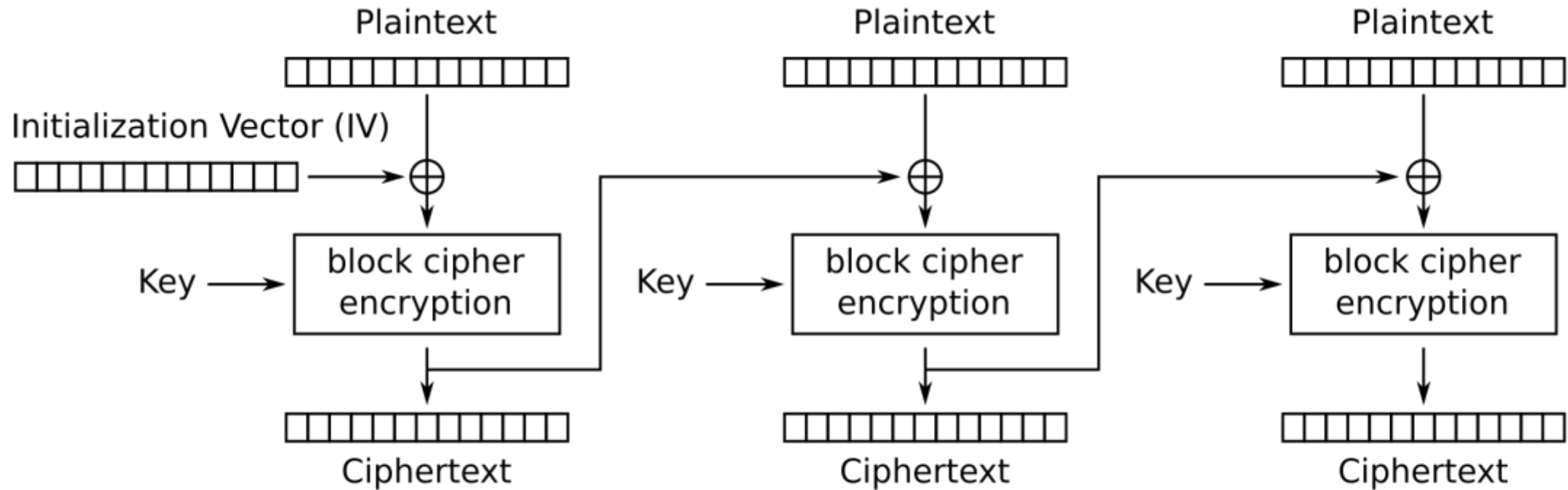


Using ECB allows patterns to be easily discerned



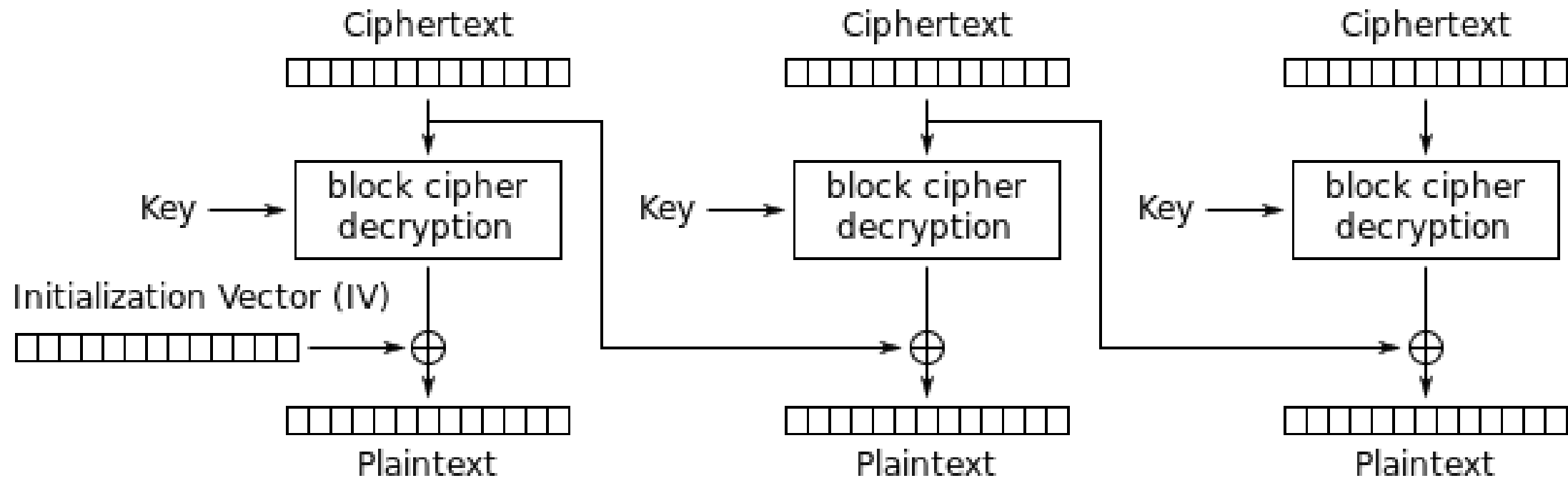
Modes other than ECB result in pseudo-randomness

¿Qué es el modo CBC?



Cipher Block Chaining (CBC) mode encryption

¿Qué es el modo CBC?



Existen muchos modos más...

- PCBC
- OFB
- CTR
- CFB
- ...

CHALLENGE



Universidad
Rey Juan Carlos

¿Qué es un Encryption Oracle Attack?

- Oracle / Oráculo: Funcionalidad del programa que filtra información.
- Acceso a un oráculo que devuelve cifrado
- Podemos controlar el plaintext
- Podemos conocer / deducir tamaño de bloque / formato
- Ejemplos clásicos: Byte-at-a-time EBC o Padding attack

byte-at-a-time ECB

- AES ECB
- Block size = 16
- AES (input + secret)

```
introduce input: testing  
## se va a cifrar lo siguiente: testingsecreto_maximo  
8a29c2e679ed6ff5395f9ff12f125066be777b7c76fb0aee2ae9ad781e9f5d8300fd25897844159ef10cc7e250e7cade
```

t	e	s	t	i	n	g	s	e	c	r	e	t	o	_	m
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

a	x	i	m	o	0x100	0x100	0x100	0x100	0x100	0x100	0x100	0x100	0x100	0x100	0x100
---	---	---	---	---	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

byte-at-a-time ECB

- Input 1: "a"*15

a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	s	e	c	r	e	t	o	_	m	a	x	i	m	o	0x00	0x00	0x00
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	------	------	------

```
introduce input: aaaaaaaaaaaaaaaaaa
## se va a cifrar lo siguiente: aaaaaaaaaaaaaaasecreto_maximo
8110984cf37db03908615db4ae2d1fea2f1368989353ab2b14070d5ffd1b030ae32b4b7a03adec06d08a557f075f8c22
```

- Input 2: "a"*15 + carácter a probar

a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	s	s	e	c	r	e	t	o	_	m	a	x	i	m	o	0x00	0x00
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	------	------

```
introduce input: aaaaaaaaaaaaaaaaaa
## se va a cifrar lo siguiente: aaaaaaaaaaaaaaasecreto_maximo
8110984cf37db03908615db4ae2d1fea2f1368989353ab2b14070d5ffd1b030ae32b4b7a03adec06d08a557f075f8c22
introduce input: aaaaaaaaaaaaaaaaaa1
## se va a cifrar lo siguiente: aaaaaaaaaaaaaa1secreto_maximo
7042f9a6eb7e01c6e89834c2ba767903165c68699c0be8e4d02a61a77f2b284e7f1ec99373e0658960c3f8b373d064e9
introduce input: aaaaaaaaaaaaaaaas
## se va a cifrar lo siguiente: aaaaaaaaaaaaaaassecreto_maximo
8110984cf37db03908615db4ae2d1fea165c68699c0be8e4d02a61a77f2b284e7f1ec99373e0658960c3f8b373d064e9
```

byte-at-a-time ECB

- Input 1: "a"*14

a	a	a	a	a	a	a	a	a	a	a	a	a	a	s	e	c	r	e	t	o	_	m	a	x	i	m	o	0x00	0x00	0x00	0x00
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	------	------	------	------

```
introduce input: aaaaaaaaaaaaaa
## se va a cifrar lo siguiente: aaaaaaaaaaaaaasecreto_maximo
e3b50003be77e48f7600a0eb44de0c76fb177a6a2038dbf6aacedb2eea57df2d5e261655285ca3a5618f6f897eaab382
```

- Input 2: "a"*14 + carácter conocido + carácter a probar

a	a	a	a	a	a	a	a	a	a	a	a	a	a	s	e	s	e	c	r	e	t	o	_	m	a	x	i	m	o	0x00	0x00
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	------	------

```
introduce input: aaaaaaaaaaaaaa
## se va a cifrar lo siguiente: aaaaaaaaaaaaaasecreto_maximo
e3b50003be77e48f7600a0eb44de0c76fb177a6a2038dbf6aacedb2eea57df2d5e261655285ca3a5618f6f897eaab382
introduce input: aaaaaaaaaaaaaasx
## se va a cifrar lo siguiente: aaaaaaaaaaaaaasxsecreto_maximo
8b17c685b926958ac3ff89856662dc42165c68699c0be8e4d02a61a77f2b284e7f1ec99373e0658960c3f8b373d064e9
introduce input: aaaaaaaaaaaaaase
## se va a cifrar lo siguiente: aaaaaaaaaaaaaasecreto_maximo
e3b50003be77e48f7600a0eb44de0c76165c68699c0be8e4d02a61a77f2b284e7f1ec99373e0658960c3f8b373d064e9
```

byte-at-a-time ECB

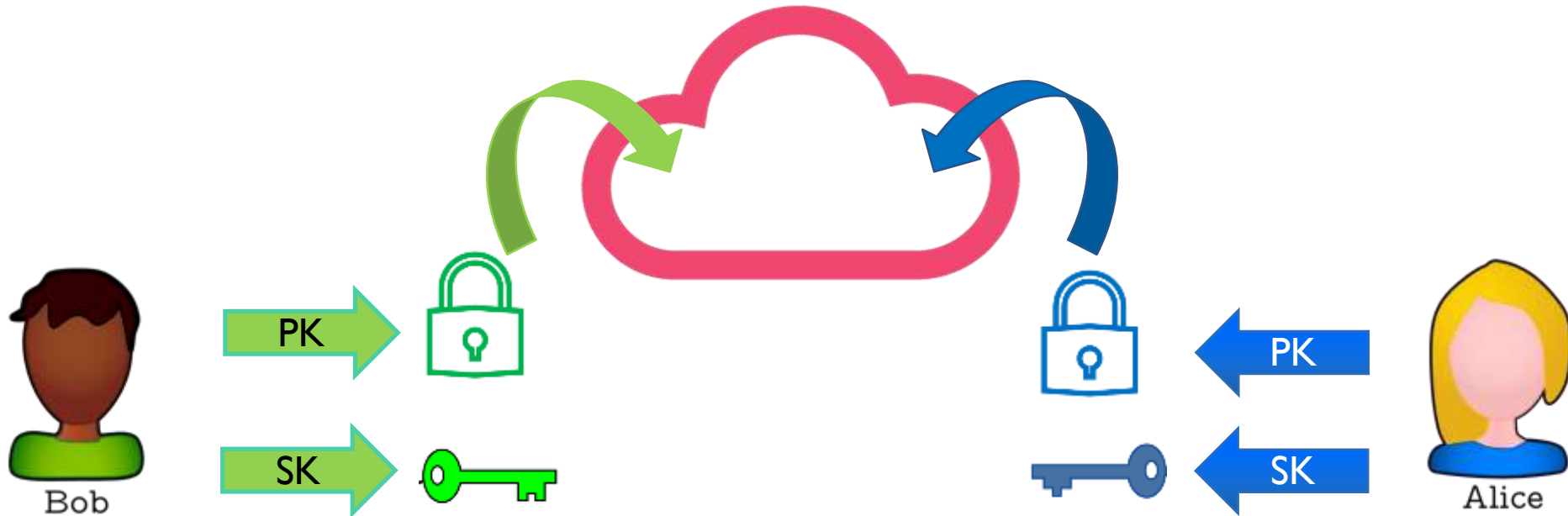
- aaaaaaaaaaaaaaas
- aaaaaaaaaaaaaaase
- aaaaaaaaaaaaaaasec
- aaaaaaaaaaaaaaasecr
- aaaaaaaaaaaaaaasecret
- ...
- aasecreto_maximo

CRIPTOGRAFÍA ASIMÉTRICA

¿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

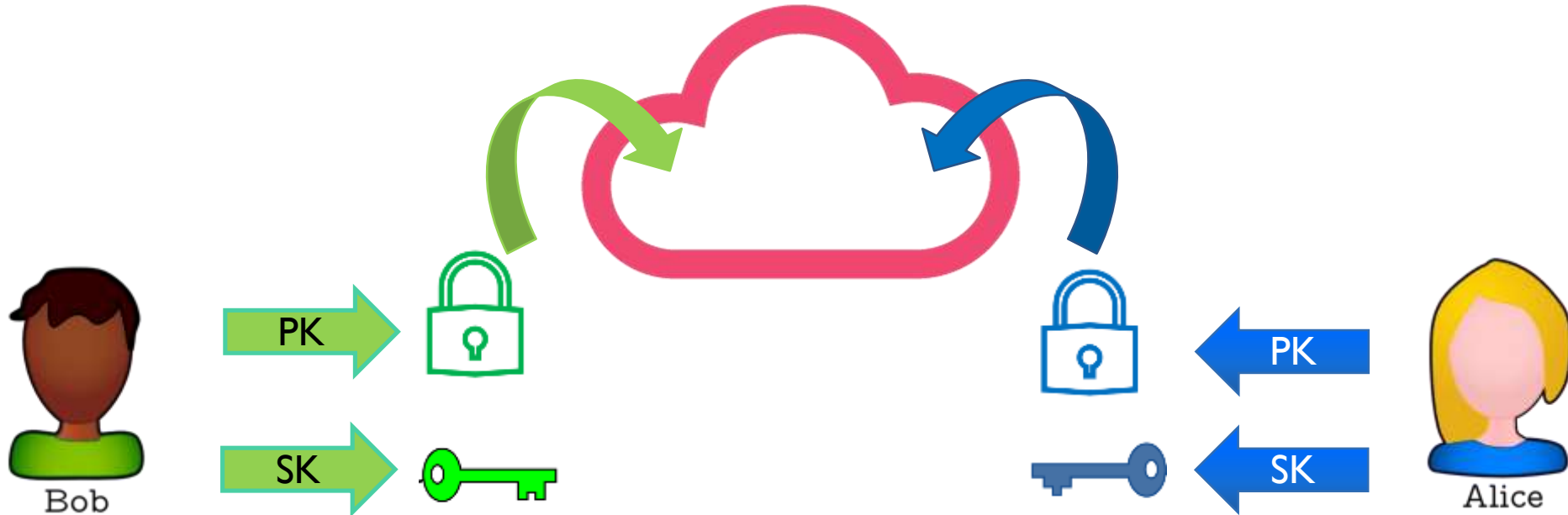


CRIPTOGRAFÍA ASIMÉTRICA

¿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

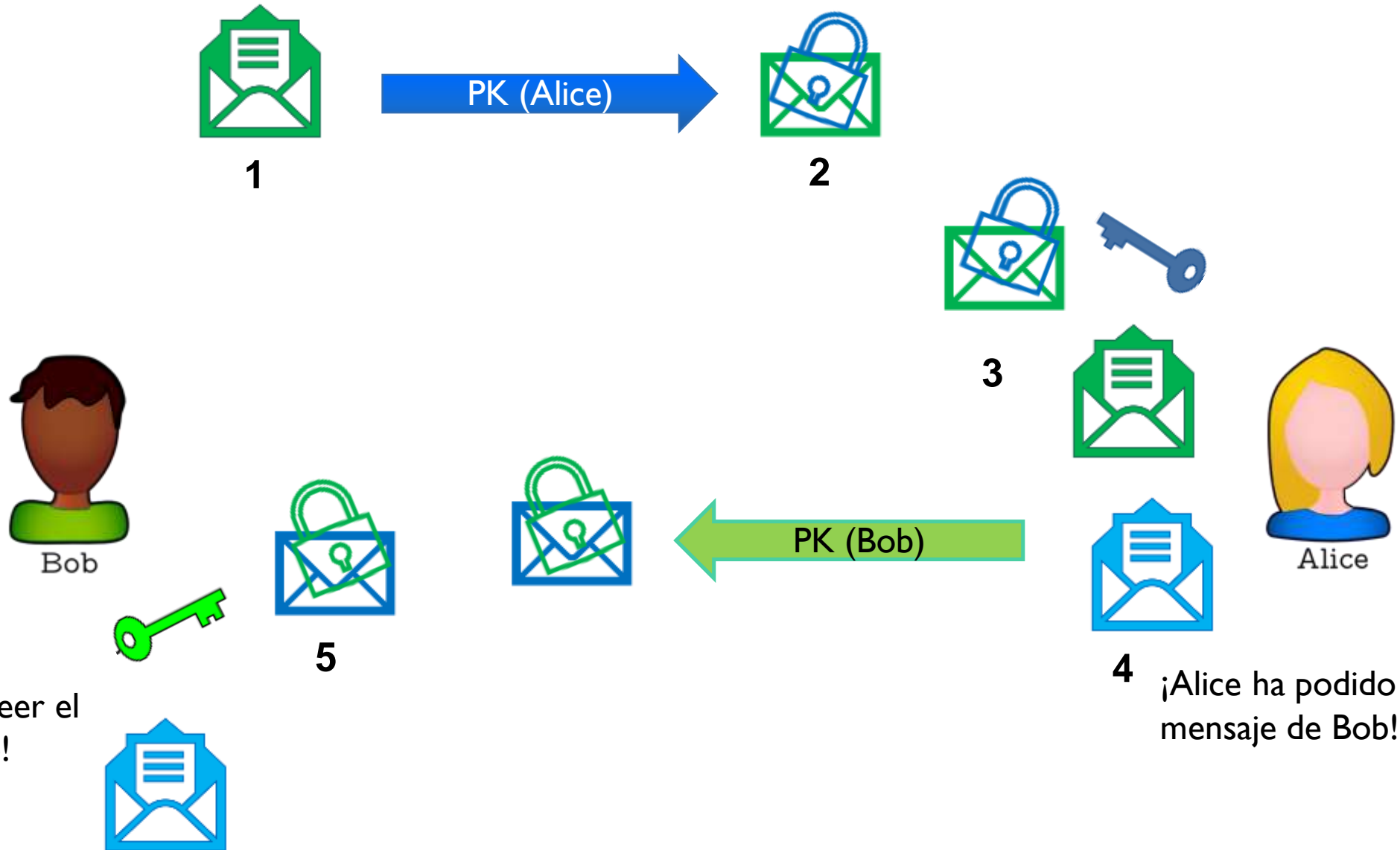


CRIPTOGRAFÍA ASIMÉTRICA

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

¿Cómo lo hacen?

CRIPTOGRAFÍA ASIMÉTRICA



¡Bob ha podido leer el mensaje de Alice!

4 ¡Alice ha podido leer el mensaje de Bob!

BASES DE LA ARITMÉTICA MODULAR

Modular/Clock Arithmetic



Modulus 12

$$a \equiv b \pmod{n}$$

$$63 \equiv 83 \pmod{10}$$

$$64 \bmod 5 = 4$$

$$\begin{array}{r} 64 \\ 14 \\ 4 \end{array} \quad \begin{array}{r} 5 \\ \hline 12 \end{array}$$

Inverso: $a^{-1} * a = 1 \bmod N$

¡¡SOLO TIENE INVERSO SI SON COPRIMOS
 $\text{GCD}(a, N) = 1$!!



RSA

Creado por Ron **Rivest**, Adi **Shamir** y Leonard **Adleman** en 1977

GEN (generación de claves)

p y $q \rightarrow$ dos números primos

$$N = p * q$$

Clave Pública
(N,e)

$$\Phi(N) = (p-1)*(q-1)$$

$$\text{Exp } e \rightarrow 1 = \text{mcd}(e, \Phi(N))$$

$$d = \text{inverso}(e) \mod(\Phi(N))$$

Clave Privada



RSA – Encriptar y Desencriptar

ENCRYPTAR UN MENSAJE

$$C = m^e \bmod(N)$$

e = Exponente

N = Producto de 2
primos P y Q

DESENCRIPTAR UN MENSAJE

$$m = c^d \bmod(N)$$

d = Clave privada

¿Por qué el inverso?

$d = \text{Clave privada } (e^{-1} \bmod \phi(N))$

$$e * d \bmod \phi(N) = 1$$

$$m = (m_e)_d = m^{e*d} = m^1$$

Python Useful Functions

Comando para instalar módulos extra de python --> **pip install pycryptodome gmpy2**

Después de ejecutar el comando **python3**, podemos usar estas funciones para hacer operaciones con números grandes.

```
pow(base,exponente,modulo) -->  $x^e \bmod N$   
pow(base,-1,modulo) --> calcular el inverso de "base" modulo  
gmpy2.iroot(x,i) --> raíz i de x  
long_to_bytes(mensaje) --> transforma un numero grande en bytes  
bytes_to_long(mensaje) --> transforma un mensaje a un numero entero grande
```

RSA – Encriptar y Desencriptar

```
from Cryptodome.Util.number import bytes_to_long, long_to_bytes, getPrime

p = getPrime(512) #Asigna un numero primo de 512 bits
q = getPrime(512)
N = p*q          #Se calcula el modulo
e = 65537

#ENCRIPtar UN MENSAJE
mensaje = b"rsa es facil" #mensaje en bytes
mensaje_long = bytes_to_long(mensaje) # Se transforma el mensaje a un numero entero
mensaje_encriptado = pow(mensaje_long, e, N)
print(mensaje_encriptado)

#DESENCRIPTAR UN MENSAJE
phi = (p-1)*(q-1) #Se calcula el totient
d = pow(e,-1,phi) #Se calcula el inverso de e modulo phi
mensaje_desencriptado = pow(mensaje_encriptado, d, N) #Calculas el valor original del mensaje
print(long_to_bytes(mensaje_desencriptado)) #Pasas el resultado a bytes para poder ser interpretado
```


RSA – Encriptar y Desencriptar



Search for a tool

★ SEARCH A TOOL ON DCODE BY KEYWORDS:
e.g. type 'boolean'

★ BROWSE THE [FULL DCODE TOOLS' LIST](#)

Results

- ❌ Wiener's attack: failure
- ❌ (Self-Limited) Prime Factors Decomposition: failure
- ✅ P,Q computed with N (FactorDB database)
- ✅ D computed with P,Q,E
- ✅ Decryption using C,D,N

Hola mundo

RSA Cipher - [dCode](#)

Tag(s) : Modern Cryptography, Arithmetics

Share

[+](#) [f](#) [t](#) [r](#) [e](#)

dCode and more

dCode is free and its tools are a valuable help in games,

RSA CIPHER

Cryptography › Modern Cryptography › RSA Cipher

RSA DECODER

Indicate known numbers, leave remaining cells empty.

★ VALUE OF THE CIPHER MESSAGE (INTEGER) C=
37629675427502492008492393023411334963114383741303 ...

★ PUBLIC KEY E (USUALLY E=65537) E=
65537

★ PUBLIC KEY VALUE (INTEGER) N=
88256459553622414063962598765941602942623923080461 ...

★ PRIVATE KEY VALUE (INTEGER) D=

★ FACTOR 1 (PRIME NUMBER) P=

★ FACTOR 2 (PRIME NUMBER) Q=

★ INTERMEDIATE VALUE PHI (INTEGER) ϕ =

★ DISPLAY ☒ PLAINTEXT AS CHARACTER STRING
☐ COMPUTED VALUES (C,D,E,N,P,Q,...)
☐ PLAINTEXT AS INTEGER NUMBER
☐ PLAINTEXT AS HEXADECIMAL FORMAT

▶ CALCULATE/DECRYPT

dCodeFR ----> <https://www.dcode.fr/rsa-cipher>

RSA – Ataque Exponente pequeño y Modulo Grande

$e \Rightarrow 3$

$N \Rightarrow$

17920302547451456255260628601727787230172785014692541565373018303626923200144183
95774329698747061027147739143566410586761477690456773509856029277075469025914383
28530840229118555963282468638588243456874311075732609267271548495053483974016854
08690680392631146761855793479884844297557829348192912981737152331029681972235731
933154889782680523867681705997376633312277

$m \Rightarrow$

123920310321213321233213231212672136867326723673126732

m^e


\Rightarrow

190295043635636787334664801249623051718205871126260174796304068080172301486
402873771262184351748127806309044976187422625748234599636095254735855176834
3285695168

Como m^e es mas pequeño que N podemos calcular el mensaje haciendo la raíz e de m

RSA – Un único primo

e \longrightarrow 65537

N \longrightarrow 857504083339712752489993810777  Es un número primo

m \longrightarrow 123920310321213321233213231212672136867326723673126732

Como N es un número primo podemos calcular Phi de N y con ello la clave privada.

$$\Phi(N) = N - 1 \text{ SOLO SI N ES PRIMO}$$

RSA – Encriptar y Desencriptar

Recursos Útiles

- <http://factordb.com/>
- <https://aurea.es/demos/criptografia/pag/calculadoraRSA.html>
- <https://github.com/jvdsn/crypto-attacks>
- <https://asecuritysite.com/rsa/>
- [RSA Calculator \(tausquared.net\)](https://tausquared.net/)
- [RsaCtfTool: https://github.com/Ganapati/RsaCtfTool](https://github.com/Ganapati/RsaCtfTool)

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>

- Vídeo AES:

<https://www.youtube.com/watch?v=tzjIRoqRnv0>



I. Criptografía avanzada

Alumnos Ciberseguridad