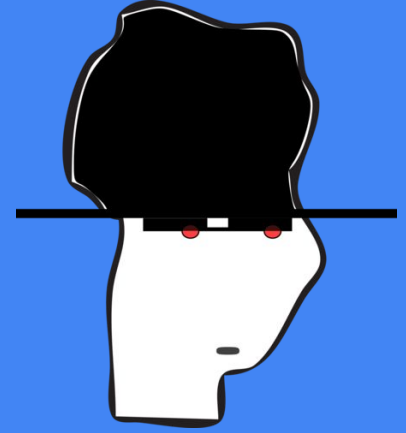


Criptografía básica

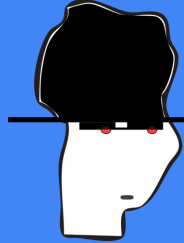
Seguridad Ofensiva



Universidad
Nacional
de Córdoba



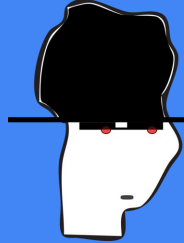
Facultad
de Matemática,
Astronomía, Física
y Computación



Crypto != Encoding

Encoding is the process of converting data from one form to another.

```
joe@zoidberg:~$ echo "HOLA MUNDO" | base64
SE9MQSBNVU5ETwo=
joe@zoidberg:~$ echo SE9MQSBNVU5ETwo= | base64 -d
HOLA MUNDO
```

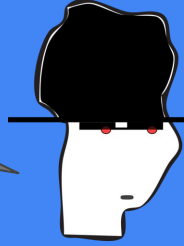


Encodings

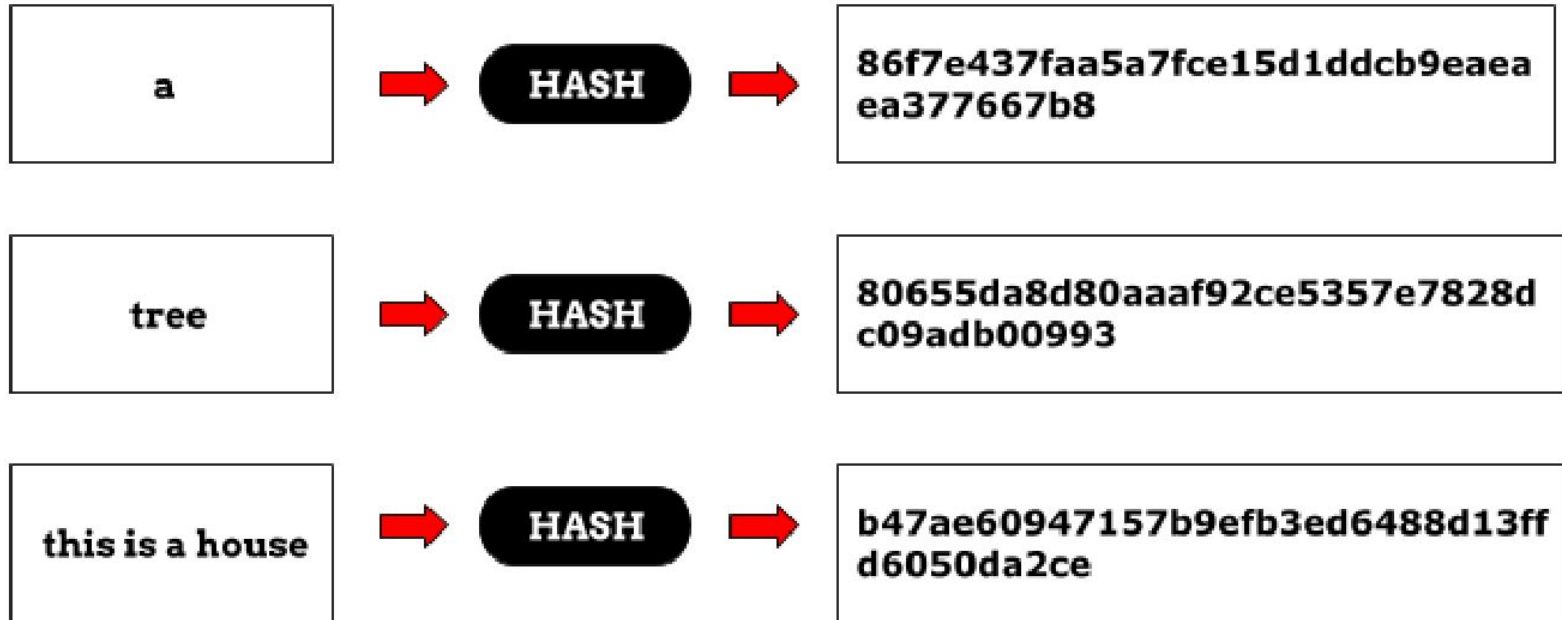
- base64 (SE9MQSBNVU5ETwo=)
- url (<https://es.wikipedia.org/wiki/Seguridad%20informatica>)
- utf-8 unicode (ñoño)
- latin1 (?o?o)
- hex (484f4c41)
- ...

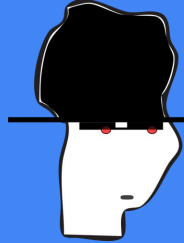
Fáciles de revertir, transformar, “re-codificar”.

Si es
criptografía



Funciones de Hash || one way





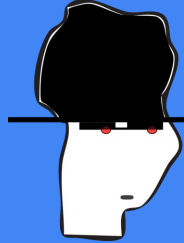
Funciones de Hash || one way

Algoritmo matemático cuyo conjunto de **entrada** son cadenas de datos de **longitud arbitraria** y su conjunto de salida son cadenas de datos de longitud fija.

Se espera que cumpla:

- Resistencia a primer preimagen
- Resistencia a la segunda preimagen
- Resistencia a Colisiones
- ...

Es determinista y se debería poder computar “eficientemente”.



Funciones de Hash || one way

Ejemplos:

MD2, MD4, MD5, RIPEMD-160, SHA-0, SHA-1

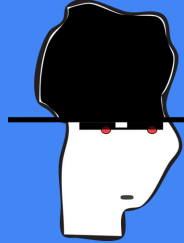
<= considerados weak

SHA-2 (256,384,512) , **SHA-3**, Whirlpool, **BLAKE**, **Keccak-256**, Shake, ...

<= considerados fuertes.

Criptografía





Usos

Comunicación Segura

- Trafico Web: HTTPS <=
- Tráfico Wireless: 802.11i WPA2 (and WEP), GSM, Bluetooth, RFID

Cifrado de Datos en Disco: EFS, TrueCrypt, GPG (análogo a comunicación)

Protección de Contenido: CSS, AACs

User authentication <=

Criptomonedas: BTC, XMR, ...



Ministerio de Ciencia,
Tecnología e Innovación
Argentina

Inicio



Iniciar Sesión

Email: *

Contraseña: *

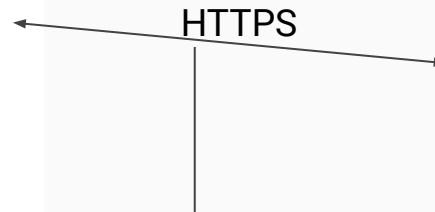
Recordarme ☐

✓ INGRESAR

[¿Olvidó su contraseña?](#)

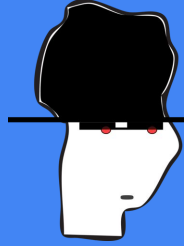
Comunicacion Insegura

Secure Communication



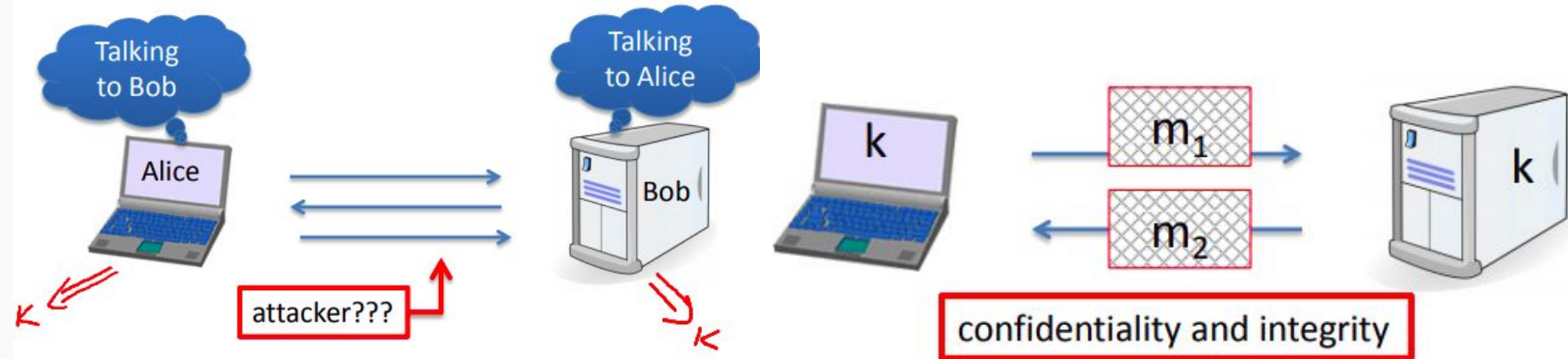
No hay manipulación (tampering)
No hay “espionaje” (eavesdropping)

Secure Communication



Secure Socket Layer / TLS serve para:

- Establish shared secret key using public-key cryptography
- Transmit data using shared secret key



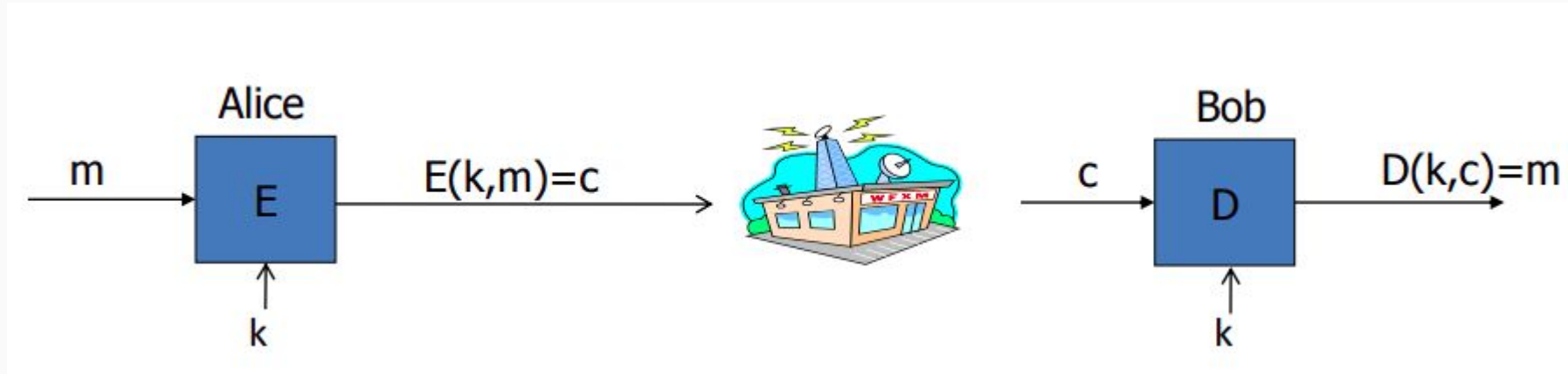
Criptografía Simétrica



Idea general



Transmitir un mensaje de forma segura.



Cipher



Def:

Un cipher definido sobre $(\mathcal{K}, \mathcal{M}, \mathcal{C})$
es un par de algoritmos eficientes (E,D)

$E: \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$ $D: \mathcal{K} \times \mathcal{C} \rightarrow \mathcal{M}$ tales que

$\forall m \in \mathcal{M}, k \in \mathcal{K}: D(k, E(k, m)) = m$

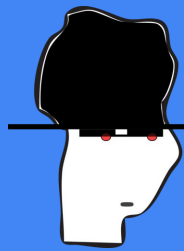




Ciphers Históricos

Algoritmos de sustitución:

- Caesar Cipher (shift)
- Sustitución Simple
- Vigenere (Desplazamiento por password paddeada)[polyalphabetic substitution]
- Enigma Machines



One Time Pad (Vernam 1917)

Cipher seguro. Tiene Perfect Secrecy.

$$c = E(k,m) = k \oplus m$$

$$D(k,c) = k \oplus c$$

Es cipher:

$$D(k, E(k,m)) = D(k, k \oplus m) = k \oplus k \oplus m = (k \oplus k) \oplus m = 0 \oplus m = m$$

msg: 0 1 1 0 1 1 1

key: 1 0 1 1 0 1 0

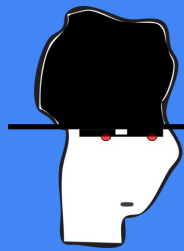


CT:



One Time Pad (Vernam 1917)

m	H	O	L	A	...
k	A	S	D	F	...
m (Hexa)	0x48	0x4f	0x4c	0x41	...
k (Hexa)	0x41	0x53	0x44	0x46	...
m (Bin)	01001000	01001111	01001100	01000001	...
k (Bin)	01000001	01010011	01000100	01000110	...
c = m \oplus k	00001001	00011100	00001000	00000111	...
c (Hexa)	0x09	0x1c	0x08	0x07	...

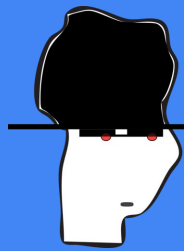


Stream Cipher

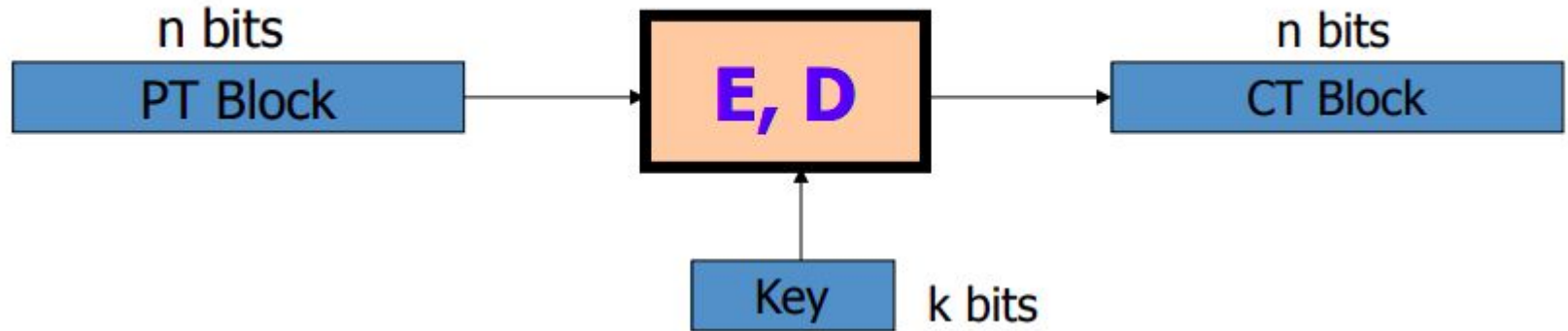
Making OTP practical using a PRG: $G: K \rightarrow \{0,1\}^n$

Stream cipher: $E(k,m) = m \oplus G(k)$, $D(k,c) = c \oplus G(k)$

Security: PRG must be unpredictable (better def in two segments)



Block Ciphers

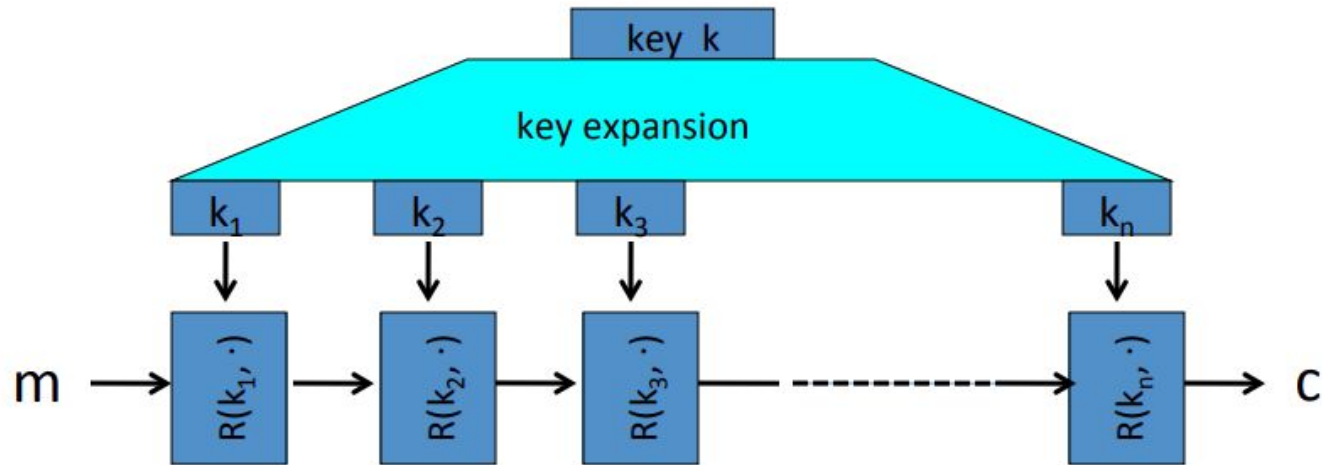
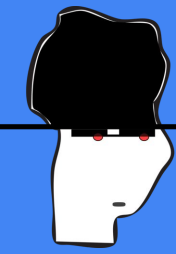


AES: $n=128$ bits, $k=128, 192, 256$ bits

DES $n=64$, $k=64$ bits

3DES $n=64$ bits, $k=168$ bits

Block Ciphers

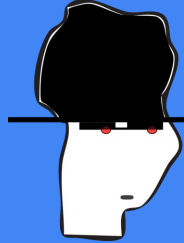


$R(k, m)$ is called a round function

for 3DES ($n=48$), for AES-128 ($n=10$)



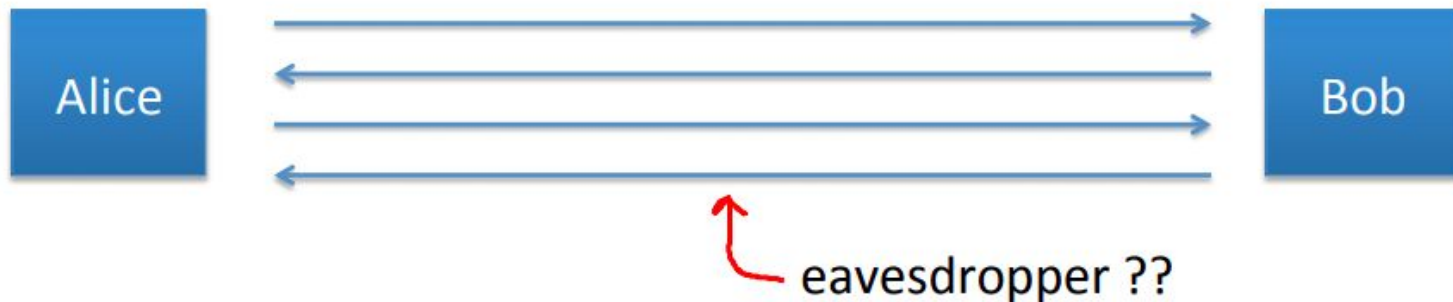
Criptografía Asimétrica



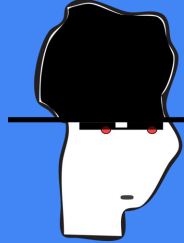
Cripto de clave pública

Goal: Alice and Bob want shared key, unknown to eavesdropper

- For now: security against eavesdropping only (no tampering)



Can this be done using generic symmetric crypto?

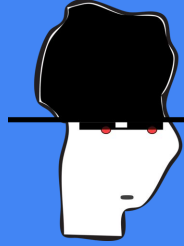


Cripto de clave pública

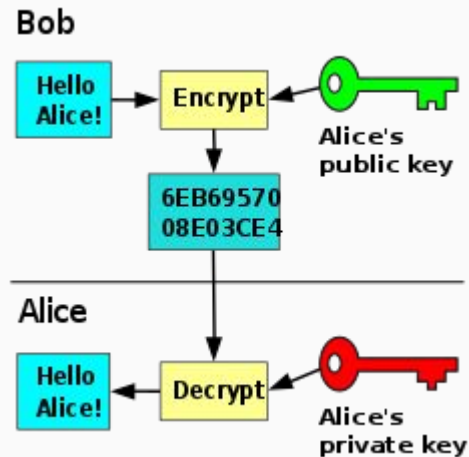
Utiliza claves públicas y privadas para cifrar y descifrar datos.

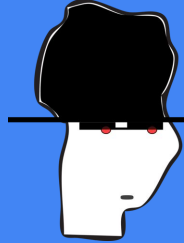
Los algoritmos de cifrado asimétrico utilizan un par de claves relacionadas matemáticamente para el cifrado y descifrado; una es la clave pública y la otra es la clave privada. Si la clave pública se usa para cifrar, la clave privada relacionada se usa para descifrar y si la clave privada se usa para cifrar, la clave pública relacionada se usa para descifrar.

Cripto de clave pública



Utiliza claves públicas y privadas para cifrar y descifrar datos.

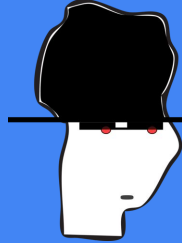




Cripto de clave pública

Beneficios:

- El problema de distribución de claves se elimina porque no hay necesidad de intercambiar claves.
- La seguridad se incrementa ya que las claves privadas nunca tienen que transmitirse o revelarse a nadie.
- El uso de firmas digitales está habilitado para que un destinatario pueda verificar que un mensaje proviene de un remitente en particular.
- Permite el no repudio para que el remitente no pueda negar el envío de un mensaje.



Cripto de clave pública

Ejemplos:

- Diffie - Hellman (1976):

Used discrete logarithms in a finite field. It allows two endpoints to swap over with a secret key on an insecure medium without any prior knowledge of each other.

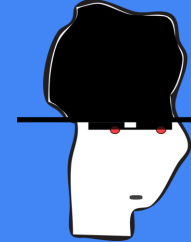
- RSA (1977):

This is the most widely used asymmetric algorithm. The RSA algorithm is used for both encrypting data and signing, providing confidentiality, and non-repudiation.

The algorithm uses a series of modular multiplications to encrypt the data

- ECC (1985?): Requires less computing power for its, encryption and decryption process. Use discrete logarithms in elliptic curves

RSA



$$C \equiv M^e \pmod{N}$$

$$M \equiv C^d \pmod{N}$$

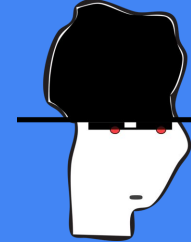
Donde:

- $N = p.q$
- $e / 1 < e < (p-1)(q-1) = \phi(N)$, e coprimo de $\phi(N)$.
- d es el inverso multiplicativo de e módulo $\phi(N)$
(es decir, debe cumplir: $e.d \equiv 1 \pmod{\phi(N)}$)

(e, N) es la clave pública

(d, N) es la clave privada.

RSA



$$C \equiv M^e \pmod{N}$$

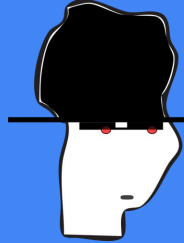
$$M \equiv C^d \pmod{N}$$

Donde:

- $N = p \cdot q$
- $e / 1 < e < \text{mcm}(p-1, q-1) = \lambda(N)$, e coprimo de $\lambda(N)$.
- d es el inverso multiplicativo de e módulo $\lambda(N)$
(es decir, debe cumplir: $e \cdot d \equiv 1 \pmod{\lambda(N)}$)

(e, N) es la clave pública

(d, N) es la clave privada.



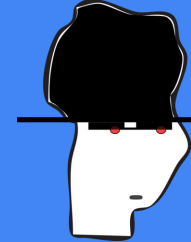
RSA (Ejemplo)

Cifrado con Clave Pública: (3,253)

A	B	C	D	E	F	G	H	I	J	K	L
0	1	2	3	4	5	6	7	8	9	10	11

M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
12	13	14	15	16	17	18	19	20	21	22	23	24	25

RSA (Ejemplo)



Queremos cifrar:

S	E	G	U	R	I	D	A	D
18	4	6	20	17	8	3	0	3

$$18^3 = 5832 \bmod 253 = 13$$

$$6^3 = 216 \bmod 253 = 216$$

$$17^3 = 4913 \bmod 253 = 106$$

$$3^3 = 27 \bmod 253 = 27$$

$$3^3 = 27 \bmod 253 = 27$$

$$4^3 = 64 \bmod 253 = 64$$

$$20^3 = 8000 \bmod 253 = 157$$

$$8^3 = 512 \bmod 253 = 6$$

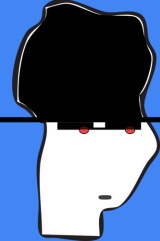
$$0^3 = 0 \bmod 253 = 0$$

$$C = 13 \ 64 \ 216 \ 157 \ 106 \ 6 \ 27 \ 0 \ 27$$



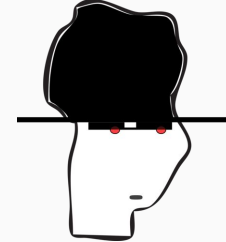
TOOLS

Terminal



```
joe@zoidberg:~$ echo "hola" | base64
aG9sYQo=
joe@zoidberg:~$ echo "hola" | sha256sum
133ee989293f92736301280c6f14c89d521200c17dcdcecca30cd20705332d44 -
joe@zoidberg:~$ openssl prime -generate -bits 24
16537267
joe@zoidberg:~$ openssl prime -generate -bits 24
15946393
joe@zoidberg:~$ wc -l secret.txt
3 secret.txt
joe@zoidberg:~$ head -1 secret.txt
Ultra Secret Message
joe@zoidberg:~$ openssl enc -aes-256-cbc -e -iter 1000 -salt -in secret.txt -out secret.enc
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
joe@zoidberg:~$ cat secret.enc
XK?j????????java?oidberg:~$ ?????
```


CyberChef



From Base64

To Hex

From Hex

To Hexdump

From Hexdump

URL Decode

Regular expression

Entropy

Fork

Magic

Data format

Encryption / Encoding

Recipe

To Base64

Alphabet
A-Za-z0-9+/=

Entropy


Visualisation
Shannon scale

Input

```
{"username": "Admin"}
```

Output

Shannon entropy: 4.351823225551766



English text

<https://gchq.github.io/CyberChef>

"A pair of Russia-designed cryptographic algorithms -- the Kuznyechik block cipher and the Streebog hash function -- have the same flawed S-box that is almost certainly an intentional backdoor. It's just not the kind of mistake you make by accident, not in 2014."

- [Bruce Schneider](#) -

