Tecnología Digital IV: Redes de Computadoras

Clase 22: Seguridad en Redes de Computadoras - Parte 1

Lucio Santi & Emmanuel Iarussi

Licenciatura en Tecnología Digital
Universidad Torcuato Di Tella

3 de noviembre de 2025

Agenda

- Introducción a la seguridad en redes
- Criptografía
  - Simétrica: **DES**, **AES**
  - De clave pública: **RSA**
- Autenticación e integridad
  - Firmas digitales
  - Funciones de hash criptográficas
  - Certificados
- El protocolo **TLS**
- *Firewalls* e IDSs/IPSs

Objetivos

- Comprender los principios básicos de la seguridad en las redes de computadoras
  - Criptografía: sus múltiples usos además de ofrecer confidencialidad
  - Autentitcación
  - Integridad
- Seguridad en la práctica
  - Protocolos seguros en los distintos niveles del *stack* TCP/IP
  - Qué es un *firewall*

Introducción a la seguridad en redes

Principios de la seguridad en redes

Confidencialidad: sólo el emisor y el receptor deben poder entender el contenido de los mensajes intercambiados

- El emisor **encripta** mensajes
- El receptor los **desencripta**

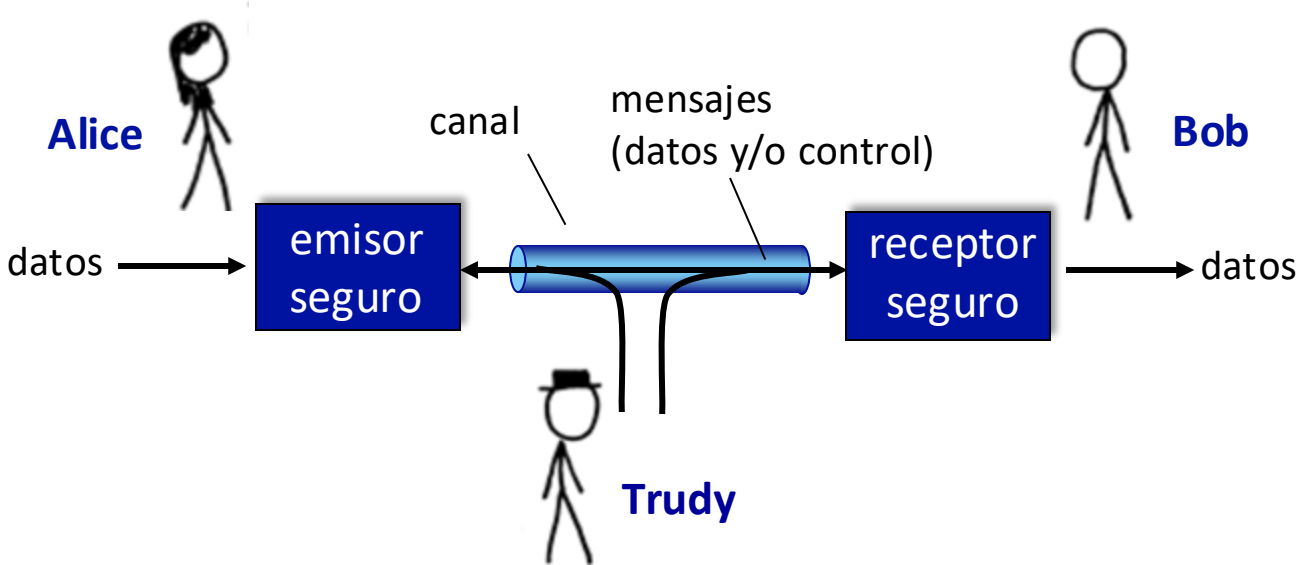
Autenticación: tanto el emisor como el receptor buscan confirmar la identidad de su interlocutor

Integridad: tanto el emisor como el receptor desean garantizar que los mensajes intercambiados no fueron manipulados en tránsito

Disponibilidad: los servicios deben ser accesibles y estar disponibles para los usuarios

Escenario y actores

- **Alice** y **Bob** desean comunicarse de forma segura
- **Trudy**, con intenciones maliciosas, puede interceptar, eliminar o agregar mensajes
- Terminología conocida y frecuente en el ámbito de la seguridad



Escenario y actores

¿Quiénes pueden adoptar los roles de Alice y Bob en la realidad?

- Navegadores y servidores web (e.g. *home banking*)
- Notebook uniéndose a una LAN buscando un servidor DHCP
- Servidores DNS
- Routers BGP intercambiando actualizaciones de caminos
- …

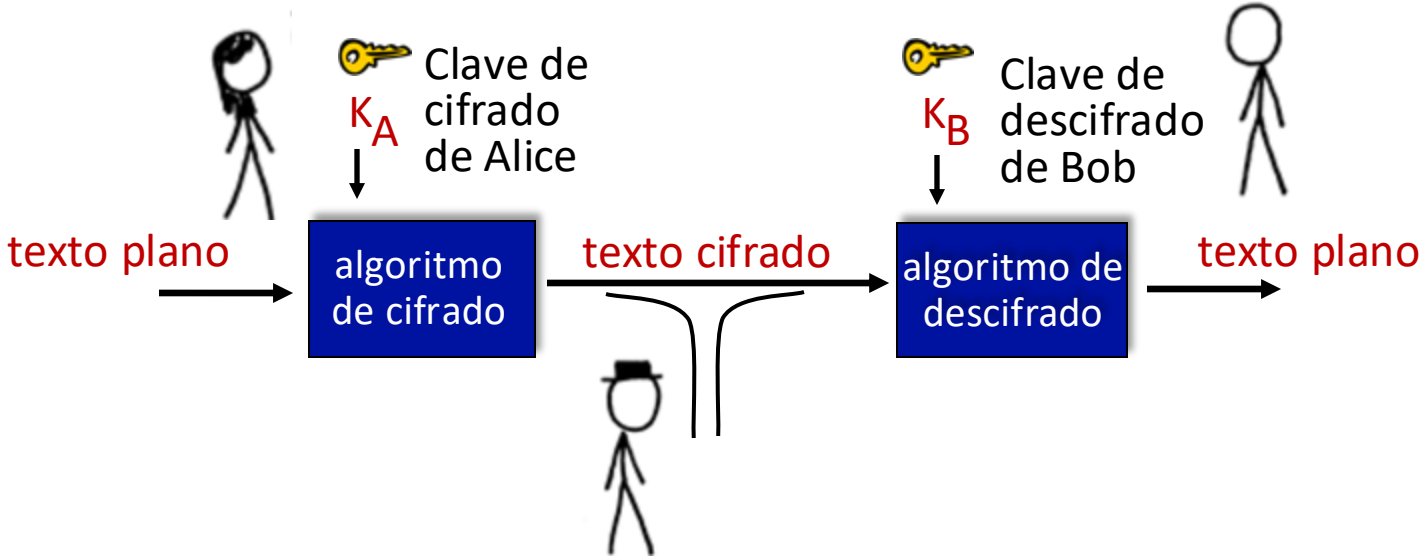
Actores maliciosos

¿Qué acciones puede llevar adelante Trudy en la realidad?

- Interceptar mensajes (*sniffing*)
- Inyectar mensajes en la conexión
- Impersonar otros actores (e.g. *spoofing* de direcciones en los paquetes)
- Tomar control de la conexión eliminando al emisor o al receptor y adoptando dicho rol (*hijacking*)
- Denegar servicio, prohibiendo que otros usuarios legítimos utilicen el servicio afectado
- …

Criptografía

Criptografía: escenario y terminología



m: mensaje en texto plano

$K_A(m)$: texto cifrado (encriptado con la clave K_A)

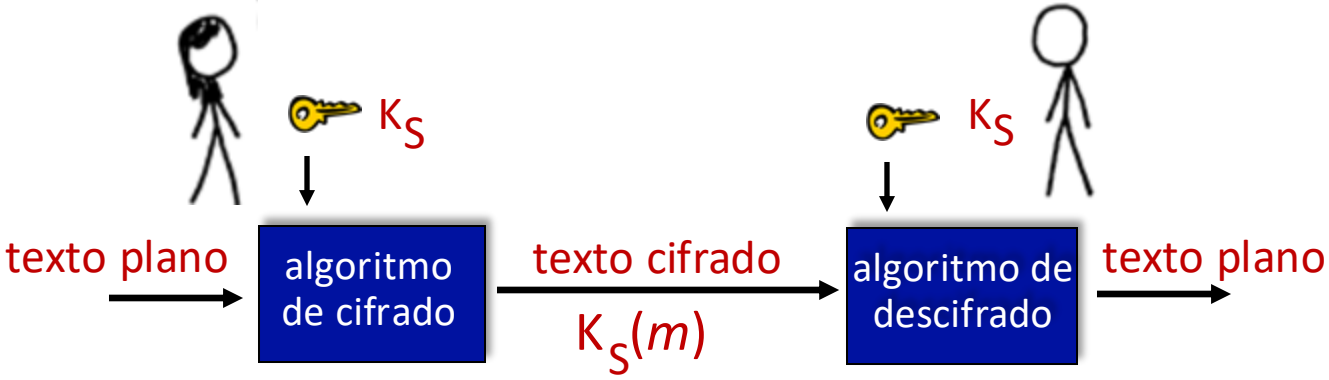
$m = K_B(K_A(m))$

Criptoanálisis - tipos de ataques

- Ataque de texto cifrado
  Supone que el atacante (Trudy) puede obtener y analizar textos cifrados
- Dos enfoques:
  - Fuerza bruta: explorar **todo** el espacio de claves
  - Análisis estadístico

- Ataque de texto plano conocido
  Supone que Trudy dispone del texto plano correspondiente a cierto texto cifrado
- Ataque de texto plano elegido
  Supone que Trudy puede obtener el texto cifrado de un texto plano a elección (*oráculo de encripción*)

Criptografía simétrica



Alice y Bob comparten la **misma clave**: K_S

¿Cómo hacen Alice y Bob para encriptar el mensaje?

¿Cómo hacen Alice y Bob para acordar una clave?

Esquema de cifrado por sustitución

En esencia, **reemplazar** una cosa por otra

- Cifrado monoalfabético: reemplazar una letra por otra:

texto plano: **`abcdefghijklmnopqrstuvwxyz`**

texto cifrado: **`mnbvcxzasdfghjklpoiuytrewq`**

e.g.:

texto plano: **`aguante td4!`**

texto cifrado: **`mzymjuc uv4!`**

Clave de cifrado: mapeo de un conjunto de 26 letras (el alfabeto) a sí mismo

Ejercicio!

- Descifrar el siguiente mensaje:

```
od fodvh pdv glyhuwlgd gh wg4
```

Un esquema de cifrado más sofisticado

- *n* cifrados por sustitución, M_1, M_2, … , M_n
- Patrón cíclico:
  - ej., *n* = 4: M_1,M_3,M_4,M_3,M_2; M_1,M_3,M_4,M_3,M_2; …
- Para cada símbolo del texto plano, utilizar el esquema de sustitución subsiguiente conforme al patrón cíclico
  - *hola*: *h* de M_1, *o* de M_3, *l* de M_4, *a* de M_3

Clave de cifrado: *n* cifrados de sustitución + patrón cíclico

Criptografía simétrica: DES

DES: *Data Encryption Standard*

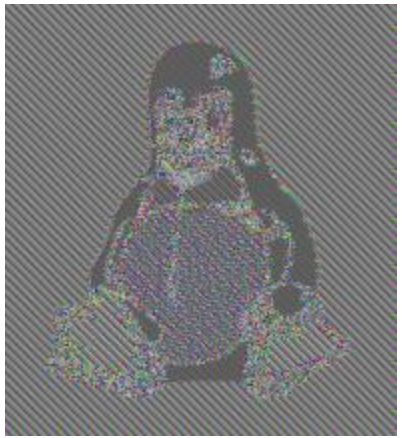
- Standard de cifrado de EEUU [NIST 1993]
- Clave simétrica de 56 bits; toma textos planos de 64 bits
- Cifrado por **bloques** (*block cipher*)

Criptografía simétrica: DES

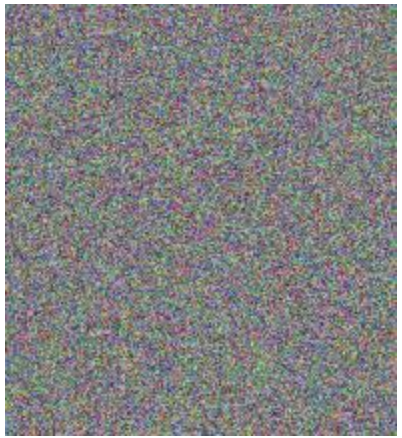
DES: *Data Encryption Standard*

- Standard de cifrado de EEUU [NIST 1993]
- Clave simétrica de 56 bits; toma textos planos de 64 bits
- Cifrado por **bloques** (*block cipher*)

Original



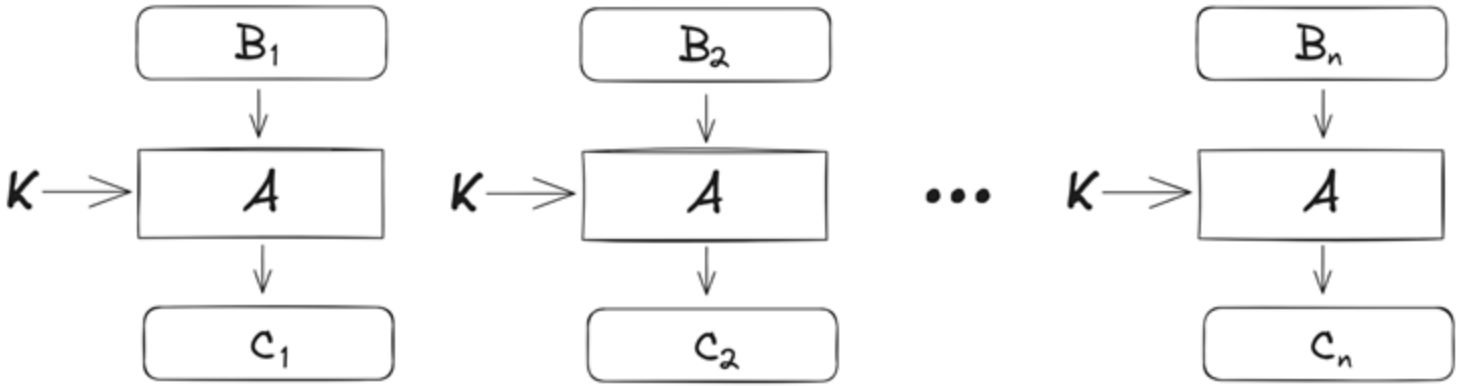
Cifrado ECB



Cifrado más seguro

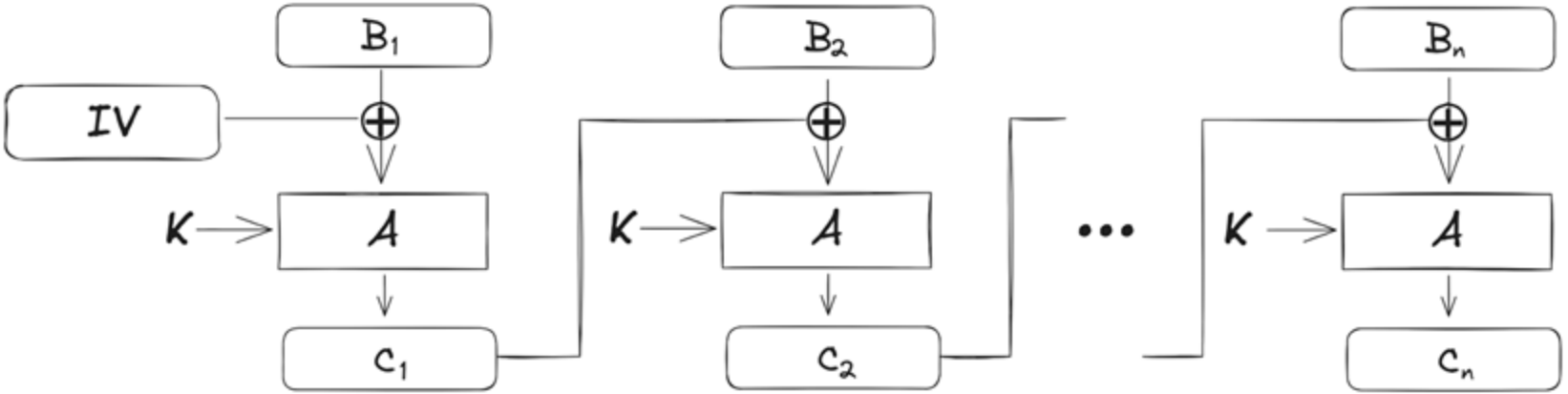
Block ciphers: modos de operación

- Supongamos un *block cipher A* con clave *K* y un texto plano conformado por *n* bloques $B_1,...,B_n$
- El modo **ECB** (*electronic codebook*) encripta cada bloque independientemente de los otros:



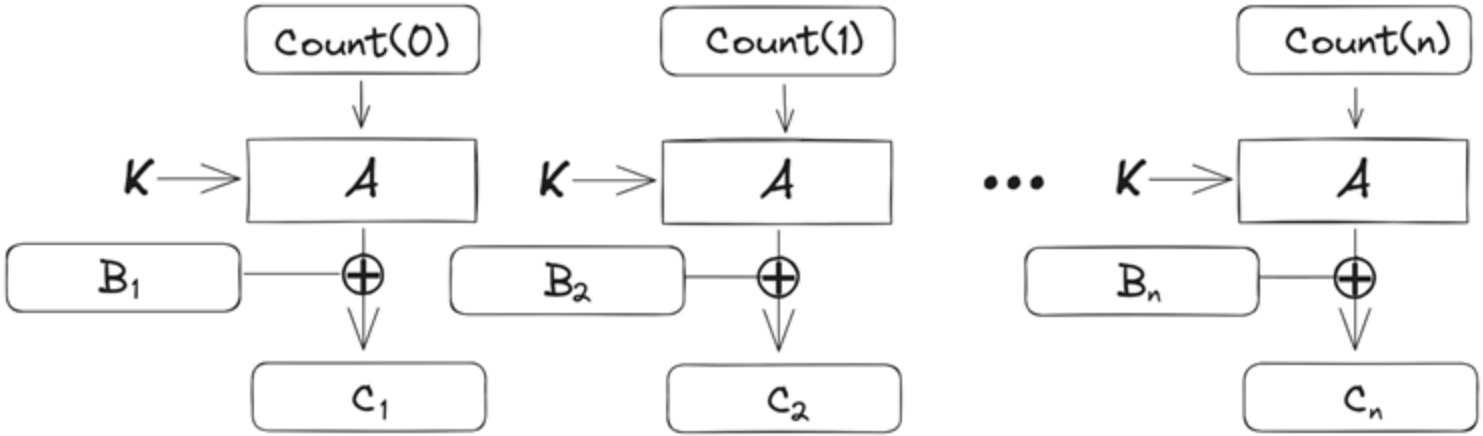
Block ciphers: modos de operación

- El modo **CBC** (*cipher block chaining*) “encadena” los bloques cifrados
- Cada bloque se cifra a partir del XOR entre el cifrado del bloque anterior y el bloque actual:



Block ciphers: modos de operación

- El modo **CTR** (*counter*) produce un *stream* de bits cifrados
- Se cifra sucesivamente el valor de una secuencia incremental y se calcula el XOR del valor cifrado obtenido con el bloque correspondiente:



Criptografía simétrica: DES

DES: *Data Encryption Standard*

- Standard de cifrado de EEUU [NIST 1993]
- Clave simétrica de 56 bits; toma textos planos de 64 bits
- Cifrado por bloques (*block cipher*)
- Seguridad de DES:
  - En 1999 se logró quebrar una clave por **fuerza bruta** en menos de un día (notar que el espacio de claves es “abordable”)
  - No se conocen ataques analíticos buenos
- Puede robustecerse el método vía **3DES**: cifrar tres veces con tres claves diferentes

AES: *Advanced Encryption Standard*

- Standard del NIST (reemplazó a DES en noviembre de 2001)
- Procesa datos en bloques de 128 bits
- Emplea claves de 128, 192, ó 256 bits
- Un ataque de fuerza bruta que tome 1 segundo para DES tomaría 149 trillones de años para AES

Criptografía de clave pública

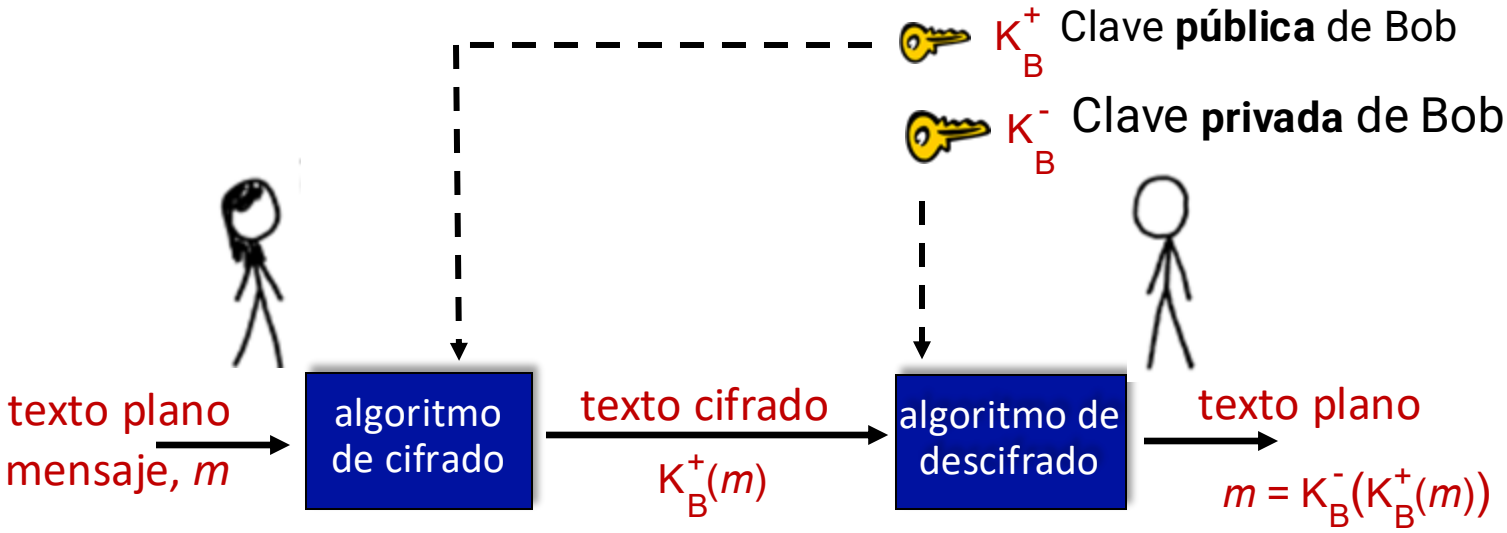
Criptografía simétrica

- Necesita que los interlocutores conozcan una clave compartida
- La pregunta es cómo acordar una –particularmente entre interlocutores que “no se conocen”

Criptografía de clave pública

- Adopta un enfoque muy diferente
- Los interlocutores no comparten una clave secreta
- La clave de cifrado es **pública** (todos la conocen)
- La clave de descifrado es **privada** (sólo la conoce el receptor del mensaje cifrado)

Criptografía de clave pública



Esta idea provocó una revolución en la criptografía –previamente, sólo basada en métodos simétricos

Algoritmos de clave pública

Requerimientos:

1. $K_B^+(\cdot)$ y $K_B^-(\cdot)$ tales que

$$K_B^-(K_B^+(m)) = m$$

2. Dada la clave pública K_B^+, debe ser **computacionalmente inviable** derivar la clave privada K_B^-

RSA: algoritmo de Rivest, Shamir y Adleman

Aritmética modular

- x mod n: el resto de dividir a x por n
- Propiedades:

  $[(a \bmod n) + (b \bmod n)] \bmod n = (a+b) \bmod n$

  $[(a \bmod n) - (b \bmod n)] \bmod n = (a-b) \bmod n$

  $[(a \bmod n) * (b \bmod n)] \bmod n = (a*b) \bmod n$
- Luego,

  $(a \bmod n)^d \bmod n = a^d \bmod n$
- Ejemplo: $x = 14$, $n = 10$, $d = 2$

  $(x \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$

  $x^d = 14^2 = 196; \quad x^d \bmod 10 = 6$

RSA: conceptos preliminares

- Mensaje: secuencia de bits
- Dicha secuencia puede representarse unívocamente como un **número entero**
- Luego, en RSA, cifrar un mensaje equivale a cifrar un número

Ejemplo:

- m = 10010001 → este mensaje se representa unívocamente como el número decimal 145
- Para cifrar m, se encripta el número correspondiente dando lugar a un nuevo número –el *texto cifrado*

RSA: cifrado y descifrado

0. Dadas las claves (n,e) y (n,d),

1. Para *cifrar* el mensaje m ($< n$), calcular

$$c = m^e \bmod n$$

2. Para *descifrar* el patrón de bits c, calcular

$$m = c^d \bmod n$$

$$m = (\underbrace{m^e \bmod n}_{c})^d \bmod n$$

RSA: ejemplo

- Supongamos que tomamos $p = 5$ y $q = 7 \rightarrow n = 35$ y $z = 24$
- Supongamos que elegimos $e = 5$ (e y z son coprimos)
- Podemos instanciar $d = 29$ ($\rightarrow ed\text{-}1$ es divisible por z)
- Luego,

	entrada	m	m^e	$c = m^e \bmod n$
cifrado:	00011001	25	9765625	30

	c	c^d	$m = c^d \bmod n$
descifrado:	30	$68630377364883 \times 10^{29}$	25

RSA: generación de claves

1. Elegir **dos números primos** *p*, *q* **grandes** (e.g. de 1024 bits)
2. Calcular $n = pq$, $z = (p-1)(q-1)$
3. Elegir $e < n$ de modo tal que no posea factores comunes con z (e y z se dicen **coprimos**)
4. Elegir d de modo tal que $ed-1$ sea divisible por z (es decir, $ed \bmod z = 1$)
5. La clave **pública** es el par $\underbrace{(n,e)}_{K_B^+}$; la **privada**, el par $\underbrace{(n,d)}_{K_B^-}$

Demostrando que RSA funciona

- Para ver que el método es correcto, debemos comprobar que
  $c^d \bmod n = m$, con $c = m^e \bmod n$
- Necesitamos el teorema de Euler: para cualquier a coprimo con n,
  $a^{\varphi(n)} = 1 \bmod n$, siendo $\varphi(n)$ la función *phi* de Euler
- Vale que $\varphi(n) = \varphi(pq) = (p - 1)(q - 1) = z$
- Luego,

$$
\begin{aligned}
c^d \bmod n &= (m^e \bmod n)^d \bmod n \\
&= m^{ed} \bmod n \\
ed = 1 \bmod z \longrightarrow \quad &= m^{(zk+1)} \bmod n \\
&= (m^{zk}\, m) \bmod n \\
&= (m^z \bmod n)^k * m \bmod n \\
\text{teo. Euler} \longrightarrow \quad &= \text{m mod n}
\end{aligned}
$$

(si m no es coprimo con n: usar teorema chino del resto)

RSA: otra propiedad importante

$$\underbrace{K_B^-(K_B^+(m))}_{\text{emplear primero la clave pública y luego la privada}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{emplear primero la clave privada y luego la pública}}$$

Esta propiedad es importante para las **firmas digitales**

RSA: otra propiedad importante

Se demuestra fácilmente a partir de las propiedades de la aritmética modular:

$$(m^e \bmod n)^d \bmod n = m^{ed} \bmod n$$
$$= m^{de} \bmod n$$
$$= (m^d \bmod n)^e \bmod n$$

Seguridad de RSA

- Dada una clave pública (*n*, *e*), ¿cuán difícil es computar *d*?
- Necesitamos **factorizar** *n* (i.e. encontrar sus factores primos)
  - Problema computacionalmente **difícil**
  - No existen hasta ahora algoritmos tradicionales de tiempo polinomial para factorizar números suficientemente grandes (a excepción de algoritmos cuánticos –el algoritmo de Shor)
  - Los *semiprimos* (producto de dos primos) con factores de tamaño similar suelen ser más difíciles de factorizar (porque tienen factores grandes)

RSA en la práctica

- La exponenciación (modular) en RSA es **costosa**
- Los algoritmos simétricos (como AES) permiten encriptar considerablemente más rápido
- En la práctica, la criptografía de clave pública se utiliza para establecer una conexión segura y derivar una clave compartida de sesión para cifrar los datos posteriores con un método simétrico

Demo!

- Veamos cómo podemos implementar (de forma rudimentaria) RSA en Python mediante **sage**:
  - ¿Cómo elegimos *p*, *q* y *e*?
  - ¿Cómo calculamos *d*?
  - ¿Cómo podemos operar con tiras de bytes arbitrarias?
  - ¿Qué ocurre si el mensaje a encriptar termina siendo más grande que el módulo *n*?

Demo!

- Ahora veamos cómo encriptar con AES utilizando la librería **PyCryptodome**:
  - ¿Cómo configuramos el modo de encripción?
  - ¿Qué pasa si encriptamos 8 bytes en modo CBC?
  - ¿Qué pasa si los encriptamos en modo CTR?