Universidad de Oviedo
Departamento de Informática
Campus de Gijón

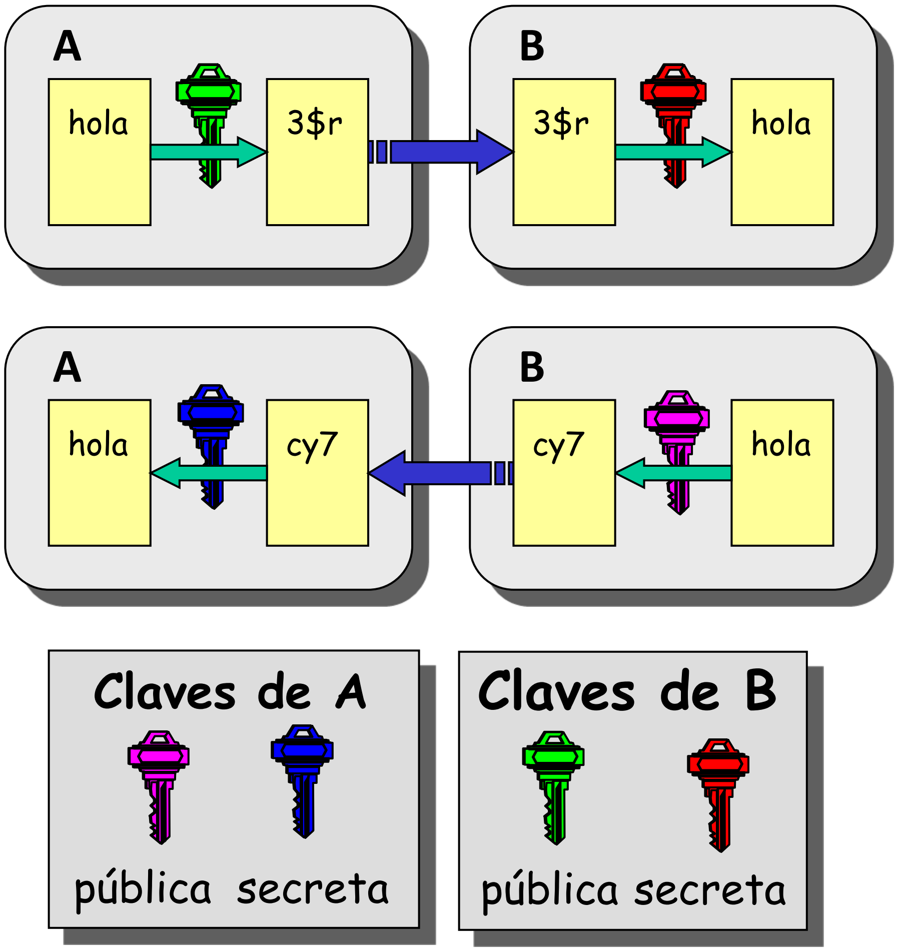
Cifrado de datos con algoritmos Asimétricos

Presentación

Daniel F. García

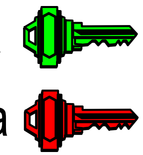
Funcionamiento de un sistema asimétrico

Ambas partes de una comunicación usan DOS claves distintas { Una para cifrar / Otra para descifrar



B desea que A le envíe un mensaje confidencial

1. B crea 2 claves { Pública / Secreta



2. B difunde su clave pública que lee A
3. A cifra el mensaje
   CON la clave pública de B
4. A envía el mensaje cifrado a B
5. B descifra el mensaje
   CON la clave secreta de B

Solo B puede descifrar el mensaje
¡El emisor A NO PUEDE descifrarlo!

Claves de un sistema asimétrico

Cada usuario genera sus **dos claves** (privada y pública) **a la vez**

- Las dos claves están relacionadas entre sí mediante funciones matemáticas
- Pero NO se puede obtener una clave a partir de la otra

VENTAJA → La gestión de claves es muy sencilla

- Cada usuario solo debe memorizar su clave privada
- NO es necesario intercambiar claves privadas a través de un "medio seguro"

INCONVENIENTE → El cifrado con clave pública es muy complejo

- Por tanto es muy lenta cuando se usa con un gran volumen de datos

Visión de algoritmos criptográficos asimétricos

1976 **DH** **Diffie-Hellman**

Permite acordar una clave secreta mediante un medio de comunicación inseguro

Es el algoritmo base (primero) de la criptografía asimétrica o de clave pública

1977 **RSA** **Rivest-Shamir-Adleman**

Fue el primer algoritmo que permitió cifrar información y firmar digitalmente

Su seguridad se basa en la dificultad de factorizar números enteros muy grandes

Es un algoritmo seguro si se usan claves de longitud apropiada

Publicado en 1978 A method for obtaining digital signatures and public-key cryptosystems
Communications of the ACM, Volume 21. Issue 2, Feb. 1978

Patentado por el MIT en US (expirada) RSA es de dominio público desde Sept 2000

1984 **El Gammal** Permite cifrado asimétrico y se basa en el Alg de Diffie-Hellman

Su seguridad se basa en la dificultad para calcular logaritmos discretos

1991 **DSA** **Digital Signature Algorithm**

Permite la firma digital (se estudiará posteriormente)

¡Hay nuevas versiones de DH, El Gammal y DSA basadas en **Curvas Elípticas**!

Algoritmo RSA: introducción

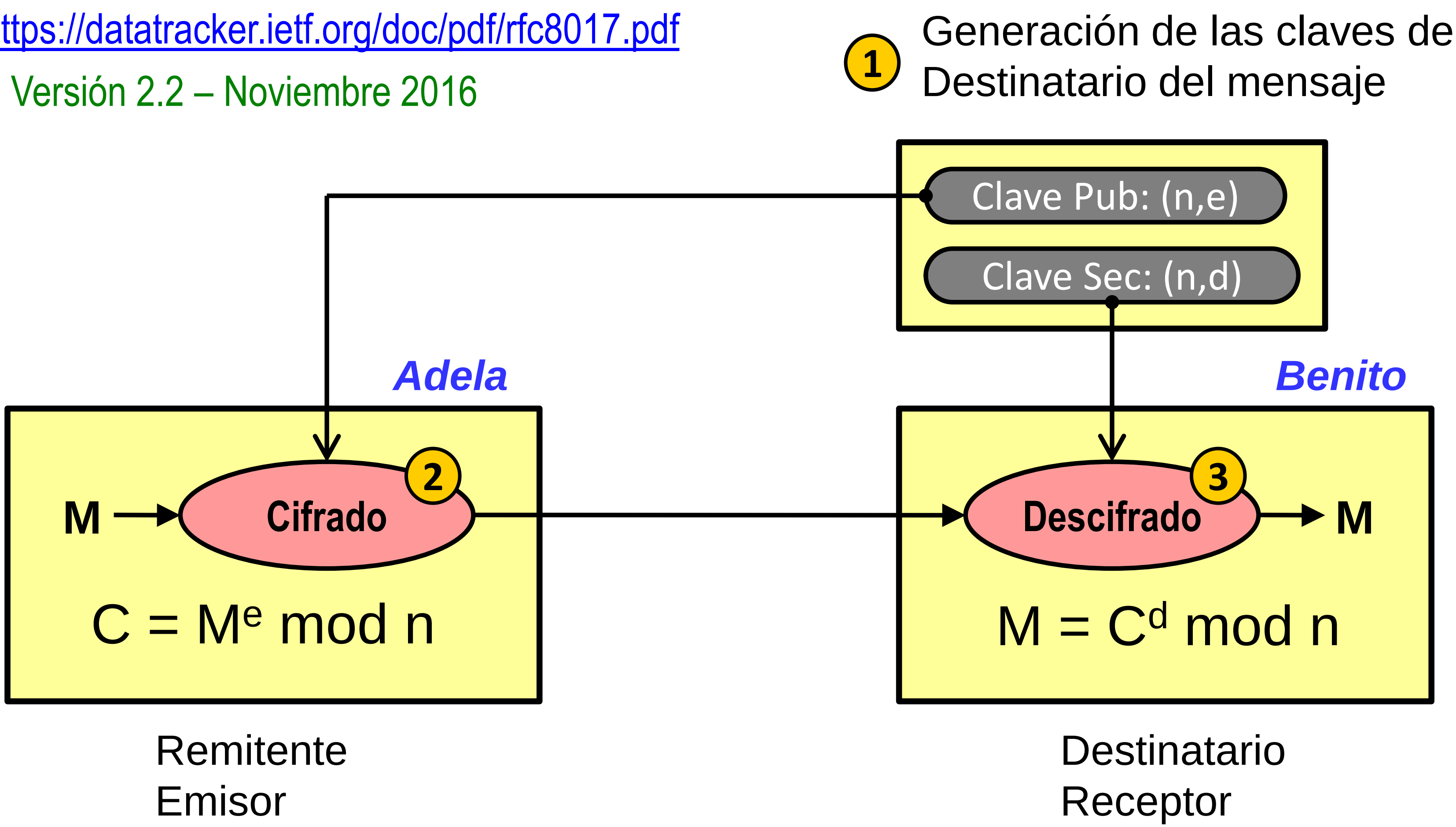
Desarrollado por **R**ivest, **S**hamir y **A**dleman (**RSA**) en 1977 en el MIT

Estándar PKCS#1

PKCS ←→ Public-Key Cryptography Standards

https://datatracker.ietf.org/doc/pdf/rfc8017.pdf

Versión 2.2 – Noviembre 2016



Algoritmo RSA: generación de claves

1) Elegir 2 números **primos** distintos p y q
Elegirlos aleatoriamente y de longitud en bits parecida

$p = 13$
$q = 7$

2) Calcular n, el **módulo** de las claves (pública y privada)
Es el producto $n = p \cdot q$

$n = 13 \cdot 7 = 91$

3) Calcular la función de **Euler** $\varphi(n) = (p-1) \cdot (q-1)$

$\varphi(n) = 12 \cdot 6 = 72$

4) Elegir e, el **exponente** de la clave **pública**
e debe estar en el rango ($1 \leq e < \varphi(n)$) y ser coprimo con $\varphi(n)$
Un valor de e muy pequeño (ej. e=3) puede ser un riesgo para la seguridad

$e = 5$ vale pues $mcd(5,72) = 1$

5) Determinar d, el **exponente** de la clave **privada** o secreta
d debe satisfacer la congruencia $d \cdot e \equiv 1 \pmod{\varphi(n)}$
Condición equivalente: $(d \cdot e - 1)$ es divisible por $\varphi(n)$
Se suele calcular con el algoritmo de Euclides extendido

$d = 29$ vale pues
$29 \cdot 5 - 1 = 144 / 72 = 2$

Algoritmo RSA: **cifrado** y descifrado

Clave pública (n,e) = (91,5) - Utilizada para **CIFRAR** el mensaje

Primero hay que convertir las letras de un mensaje a números
Los números se pueden utilizar en las operaciones de exponenciación

Usaremos la siguiente "sustitución" de letras por números:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

Cifrado: $C = M^e \bmod n$

- M es el código numérico claro que representa cada letra
- C es el código numérico cifrado correspondiente

```
S  E  R  E  N  O  ← Letras
20 05 19 05 14 16 ← Códigos claros
76 31 80 31 14 74 ← Códigos cifrados
```

20^5 mod 91 = 3.200.000 mod 91 = 76
05^5 mod 91 = 3.125 mod 91 = 31
19^5 mod 91 = 2.476.099 mod 91 = 80
14^5 mod 91 = 537.824 mod 91 = 14
16^5 mod 91 = 1.048.576 mod 91 = 74

Algoritmo RSA: cifrado y **descifrado**

Clave privada (n,d) = (91,29) - Utilizada para **DESCIFRAR** el mensaje

"Sustitución" de letras por números utilizada:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

Descifrado: $M = C^d \bmod n$

- C es el código numérico cifrado de cada letra del mensaje
- M es el código numérico claro que representa cada letra

```
76 31 80 31 14 74 ← Códigos cifrados
20 05 19 05 14 16 ← Códigos claros
S  E  R  E  N  O  ← Letras
```

$76^{29} \bmod 91 = 3{,}54 \times 10^{54} \bmod 91 = 20$

$31^{29} \bmod 91 = 1{,}77 \times 10^{43} \bmod 91 = 5$

$80^{29} \bmod 91 = 1{,}54 \times 10^{55} \bmod 91 = 19$

$14^{29} \bmod 91 = 1{,}72 \times 10^{33} \bmod 91 = 14$

$74^{29} \bmod 91 = 1{,}61 \times 10^{54} \bmod 91 = 16$

Estas operaciones se pueden realizar con "La calculadora científica de Windows"

Algoritmo RSA: bases de su seguridad

RSA se basa en lo que se denomina "**funciones unidireccionales con trampa**"

- El uso de la función F en $\underbrace{\text{sentido “directo”}}_{\text{Uso legítimo: cifrar y descifrar}}$ es FÁCIL
- El uso de la función F en $\underbrace{\text{sentido “inverso”}}_{\text{Uso ilegítimo: hackers}}$ es DIFÍCIL

La seguridad de RSA se basa en "el problema de la factorización"

Cálculo directo → producto de dos primos grandes p x q = n

Muy fácil de calcular

Cálculo inverso → factorización de un número grande n = p x q

Muy difícil obtener dos primos p y q si n es muy grande

Algoritmo RSA: forma de romper el cifrado

Ataque obvio NO usado

Se conoce la clave pública (n,e) ?
Se desconoce el exponente de la clave secreta (n,d)

Se cifra un mensaje M

$C = M^e \bmod n$

Se descifra un mensaje encriptado C

$M' = C^d \bmod n$

↑

Probar valores de d hasta que M' = M

Si M, C y n son muy grandes, encontrar d mediante pruebas es casi imposible

Ataque estándar usado

1) Como se conoce n, se factoriza en dos primos p y q tal que $n = p \cdot q$

2) Calcular $\varphi(n) = (p-1)\cdot(q-1)$

3) Resolver la ecuación $d \cdot e \equiv 1 \ (\bmod\ \varphi(n))$ para obtener d

El problema intratable es el paso 1) de factorización

Algoritmo RSA: elección de parámetros (1)

La seguridad de RSA depende de la elección de los parámetros $\begin{cases} p, q \rightarrow n \\ e, d \end{cases}$

Elección de **p y q**

Longitud de p y q > 512 bits → n tendrá más de 1024 bits

Las longitudes de p y q deben diferir en unos pocos dígitos (relación bits/dígito ≈ 3,3)

p y q NO DEBEN ser primos muy cercanos

p-1 y q-1 DEBEN tener factores primos grandes

mcd(p-1, q-1) DEBE ser pequeño

Estas condiciones se cumplen calculando p y q con los denominados primos seguros

Consideraciones sobre la longitud de n

En Febrero de 2020 se completó la factorización del numero RSA-250 de 829 bits

https://en.wikipedia.org/wiki/RSA_numbers

Por ello las longitudes de n son generalmente de 1024 a 2048 bits

Si la longitud de n < 300, n puede ser factorizado en unas horas en un PC

https://en.wikipedia.org/wiki/Integer_factorization_records

Algoritmo RSA: elección de parámetros (2)

Cálculo de números primos p y q seguros

- Elegir R un número primo grande y calcular: $p = 2R+1$
- Elegir R' otro primo algo mayor que R y calcular: $q = 2R'+1$

Para elegir R y R' usar tablas o una herramienta:

https://www.walter-fendt.de/html5/mes/primenumbers_es.htm

Ejemplo: $R=1019$ y $R'=3863$

$p = 2\cdot1019 + 1 = 2039$ que es primo = 111 1111 0111 (11 bits)

$q = 2\cdot3863 + 1 = 7727$ que es primo = 1 1110 0010 1111 (13 bits)

$n = p\cdot q = 2039\cdot7727 = 15.755.353$

Además …

$$\left.\begin{array}{l} p - 1 = 2.038 = 2\cdot1019 \\ q - 1 = 7.726 = 2\cdot3863 \end{array}\right\} \rightarrow \text{mcd}(p - 1, q - 1) = 2 \text{ que es pequeño}$$

Algoritmo RSA: elección de parámetros (3)

Elección de e *(exponente público)*

e debe elegirse pequeño para facilitar su manejo y las operaciones a realizar

¡Pero NO debe ser muy pequeño!

El NIST (Publicación SP 800-78 Rev.4 de mayo 2015)

https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-78-4.pdf

NO permite e < 65.537 ← Mayor número primo de Fermat conocido

Primos de Fermat:
$$F_n = 2^{2^n} + 1$$
$$F_4 = 2^{2^4} + 1 = 2^{16} + 1 = 65537$$

Cálculo de d *(exponente secreto)*

Cumpliendo las restricciones anteriores

Como se debe verificar (d·e) mod $\varphi(n) = 1$

Se obtiene una longitud de d > 1000 bits (≈1024 bits)

Es importante que el exponente secreto sea lo suficientemente largo

Wiener demostró que SI $q<p<2q$ Y $d<n^{1/4}/3$ se puede calcular d eficientemente a partir de n y e

Algoritmo RSA: esquemas de relleno (1)

El ejemplo de cifrado RSA basado en la palabra SERENO es meramente didáctico

NO se puede realizar un cifrado real byte a byte individualmente

```
S  E  R  E  N  O  ← Letras
20 05 19 05 14 16 ← Códigos claros
76 31 80 31 14 74 ← Códigos cifrados
```

Esta forma de cifrar permite:

Ataques basados en el análisis de las frecuencias de los códigos cifrados

En cada idioma, la frecuencia de aparición de cada letra es diferente

En español, la letra más frecuente es la E

Se puede probar a reemplazar el código cifrado que más se repite por una E
Y continuar las sustituciones hasta inferir los códigos de las letras restantes

Solución POSIBLE

Combinar varios números en bloques:

```
 SER    ENO
200519 051416  ← Cifrar los números grandes
```

El módulo n debe ser siempre mayor que el mayor número a cifrar

Algoritmo RSA: esquemas de relleno (2)

Solución REAL

Las implementaciones reales de RSA incrustan bits de relleno seleccionados aleatoriamente en el mensaje claro M antes de cifrarlo

- El relleno asegura que el mensaje rellenado (M-padded)
  1) No es un texto inseguro (con patrones muy evidentes)
  2) Se cifrará a un C-padded entre **muchísimos** posibles

- Las especificaciones para rellenar están definidas en:
  Los estándares de criptografía pública PKCS
  PKCS = Public-Key Cryptographic Standards
  Han sido diseñados y publicados por los laboratorios RSA en California (USA)

Relleno PKCS#1 v1.5

Aunque se sigue usando se desaconseja su uso

EM = 0x00 || 0x02 || *PS* || 0x00 || *M*

PS (*Padding String*) son bytes generados aleatoriamente distintos de cero

Longitud de PS = Longitud del módulo RSA – Longitud del mensaje – 3

El mensaje rellenado (*EM, Encoded Message*) tiene una longitud igual al módulo RSA

PKCS#1 v2.2
RFC 3447

Relleno OAEP (*Optimal Asymmetric Encryption Padding*)

1 Se parte de una etiqueta *L* (puede estar vacía) y se obtiene su resumen (hash) → *lHash* (de longitud *hLen*)

2 Añadir *PS* (*Padding String*) que son bytes a cero

3 Añadir byte *01* y el mensaje *M* → *DB* (*Data Block*)

4 Generar una semilla aleatoria (de longitud igual al hash, *hLen*)

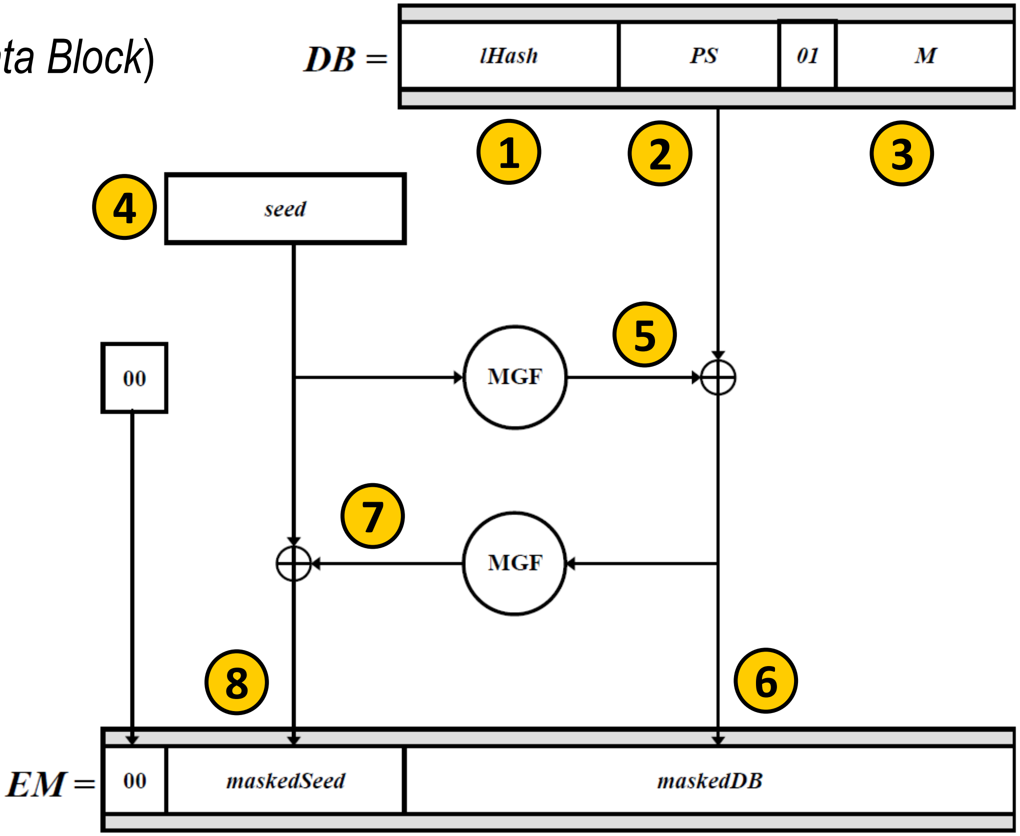
5 dbMask = MGF(seed)

6 maskedDB = DB ⊕ dbMask

7 seedMask = MGF(maskedDB)

8 maskedSeed = seed ⊕ seedMask

El mensaje rellenado (*EM, Encoded Message*) tiene una longitud igual al módulo RSA



Algoritmo RSA: SU GRAN PROBLEMA

El cifrado exponencial con clave pública (como la usada en RSA)
Es muy complejo y su coste computacional es muy elevado

NO se puede cifrar un volumen de información grande en un tiempo razonable

Entonces … ¿Qué utilidad tiene?

Cifrar, transmitir, y descifrar la clave secreta que comparten dos usuarios
(Una clave → Volumen de información reducido)

Los usuarios usan la clave secreta para cifrar información con un algoritmo simétrico

Actualmente se usan CRIPTOSISTEMAS HÍBRIDOS:

- Criptografía **asimétrica** para intercambiar claves simétricas cifradas

+

- Criptografía **simétrica** para intercambiar información cifrada

Aplicación de RSA: intercambio de claves (1)

PASOS:

Variante: Pasos 1 a 3

Usar algoritmo de clave pública para intercambio de claves

El más usado: Diffie-Hellman

1. B crea 2 claves: Pública + Privada
   B difunde su clave pública

2. A recupera la clave pública de B
   A genera una lista de números aleatorios Na
   A cifra la lista de Na con la clave pública de B y los envía a B

3. B descifra los números aleatorios con su clave privada
   Solo puede hacerlo B pues solo B tiene la clave privada

4. A y B usan los números aleatorios como clave simétrica compartida
   Todos los intercambios de información los hacen con criptografía simétrica
   (a un coste computacional bajo)

$$A \xrightarrow[\text{una clave secreta K}]{\text{desea enviar}} B$$

$K = DA9F_h = 55.969_d$

$n_B = 65.669$
$e_B = 35$
$d_B = 53.771$

A cifra la clave secreta K con la clave pública de B

$C = K^{e_B} \bmod n_B = 55.969^{35} \bmod 65.669 = 45.213$

A envía C = 45.213 en un mensaje a B

B descifra la clave secreta K con su clave privada

$K = C^{d_B} \bmod n_B = 45.213^{53.771} \bmod 65.669 = 55.969$

Aplicación de RSA: autenticación (informal)

El algoritmo RSA puede **trabajar al revés** (de cómo se ha explicado hasta ahora)

Nueva forma de trabajo

Información CIFRADA con una clave privada → Puede ser DESCIFRADA con la clave pública correspondiente

Alicia se autentica (prueba su identidad) a **B**enito

1. A cifra una información (conocida por A y B) CON la clave privada de A
2. A envía la información cifrada a B
3. B descifra la información CON la clave pública de A
4. B compara las informaciones (descifrada y conocida) Si Coinciden B sabe que la ha enviado A

Aplicación RSA: autenticación +formal → firma digital

Alicia "firma" el mensaje M que envía a **B**enito (M puede enviarse cifrado o no)

PASOS:

1. A calcula un resumen (hash) h(M) del mensaje M
   (las funciones hash se ven posteriormente)
2. A cifra el resumen h(M) con su clave RSA privada
   Resumen cifrado ← → Firma digital (**Específica para cada M**)
3. A envía el mensaje M + resumen h(M) cifrado a B
4. B descifra el resumen usando la clave RSA pública de A, obteniendo h(M)
5. B calcula el resumen (hash) h'(M) del mensaje recibido M
6. B comprueba: SI h(M) = h'(M) ENTONCES A es el autor de M

Aplicación RSA: cifrado local (inútil)

Hasta ahora se han visto 2 utilidades del algoritmo RSA:

1 ***Cifrando con la clave pública del destinatario***

Uso: intercambio de claves

2 ***Cifrando con la clave privada del emisor***

Uso: autenticación (firma digital)

¿Qué posibilidad queda?

3 ***Cifrar información con la clave pública del emisor***

SOLO el propio emisor puede descifrar la información
pues es el único que tiene acceso a su clave privada

Posible Uso: cifrar archivos locales

¡ Pero los algoritmos simétricos son 100-1000 veces más rápidos !

Diffie-Hellman: Introducción

Permite el intercambio confidencial de claves entre dos personas que no han tenido un contacto previo, usando un canal de comunicación inseguro y de forma anónima

Fue inventado por Whitfield Diffie y Martin Hellman en 1976
→ Para intercambiar (acordar) claves

Aunque es un protocolo sin autenticación de los intervinientes constituye la base para otros protocolos con autenticación

Estándar PKCS#3

https://datatracker.ietf.org/doc/pdf/rfc2631.pdf

Está integrado en el Estándar ANSI (ANS X9.42)
Agreement of Symmetric Keys Using Discrete Logarithm Cryptography

Diffie-Hellman: Fases

Fases del algoritmo, para un intercambio de claves: Alicia ←→ Benito

1 Ambos, Alicia y Benito seleccionan y comparten dos números públicos:
- Un módulo p
- Un generador g

Alicia

2 Elige su clave secreta K_{SA} (entero aleatorio largo)

3 Calcula su clave pública
Y la envía a Benito

$$K_{PA} = g^{K_{SA}} \bmod p$$

4 Recibe la clave pública de Benito
Calcula la clave secreta común

$$K_{SC} = K_{PB}^{K_{SA}} \bmod p$$

Benito

2 Elige su clave secreta K_{SB} (entero aleatorio largo)

3 Calcula su clave pública
Y la envía a Alicia

$$K_{PB} = g^{K_{SB}} \bmod p$$

4 Recibe la clave pública de Alicia
Calcula la clave secreta común

$$K_{SC} = K_{PA}^{K_{SB}} \bmod p$$

$$K_{SC} = (g^{K_{SB}} \bmod p)^{K_{SA}} \bmod p = g^{K_{SB}K_{SA}} \bmod p$$

$$K_{SC} = (g^{K_{SA}} \bmod p)^{K_{SB}} \bmod p = g^{K_{SA}K_{SB}} \bmod p$$

Diffie-Hellman: Ejemplo numérico

1 Ambos, Alicia y Benito seleccionan y comparten:
Módulo p = 23
Generador g = 5

Alicia

2 Elige $K_{SA} = 6$

3 Calcula $K_{PA} = g^{K_{SA}} \bmod p$

$K_{PA} = 5^6 \bmod 23 = 8$

(15 625)

4 Calcula $K_{SC} = K_{PB}^{K_{SA}} \bmod p$

$K_{SC} = 19^6 \bmod 23 = 2$

(47 045 881)

Benito

2 Elige $K_{SB} = 15$

3 Calcula $K_{PB} = g^{K_{SB}} \bmod p$

$K_{PB} = 5^{15} \bmod 23 = 19$

(30 517 578 125)

4 Calcula $K_{SC} = K_{PA}^{K_{SB}} \bmod p$

$K_{SC} = 8^{15} \bmod 23 = 2$

(35 184 372 088 832)

Diffie-Hellman: Seguridad (1)

El objetivo de un atacante es descubrir las claves secretas de Alicia y Benito
Conociéndolas, puede calcular la clave secreta común directamente

El atacante debe resolver una de estas ecuaciones →
$$\begin{cases} K_{PA} = g^{K_{SA}} \bmod p \\ K_{PB} = g^{K_{SB}} \bmod p \end{cases}$$

K_{PA}, K_{PB}, p y g con conocidos ¿K_{SA} ? ¿K_{SB}?

Encontrar K_{SA} ó K_{SB} ←→ Resolver el Problema del Logaritmo Discreto (PLD)
El PLD es computacionalmente intratable si p, g, K_{SA} y K_{SB} han sido bien elegidos

Reglas para la elección correcta

- El **módulo p** debe ser un número primo grande (de al menos 1024 bits)

  Interesa que el indicador de Euler $\varphi(p)$ = p-1 tenga factores primos grandes (además de incluir el factor 2)

  Si p es pequeño se puede hacer un ataque por fuerza bruta en un tiempo razonable

Diffie-Hellman: Seguridad (2)

- El **generador g** debe ser una raíz primitiva del módulo p

  Si el generador g NO es una raíz primitiva del grupo p
  entonces la operación g^{Ks} mod p (1<Ks<p-1) NO genera todos los restos del grupo
  Y esto facilita un ataque por fuerza bruta

Ejemplo de una MALA elección de parámetros p=13, g=3 $K_P = 3^{Ks} \bmod 13$

Cálculo de las claves públicas que se pueden generar

$3^1 \bmod 13 = 3$	$3^2 \bmod 13 = 9$	$3^3 \bmod 13 = 1$ ← MAL
$3^4 \bmod 13 = 3$	$3^5 \bmod 13 = 9$	$3^6 \bmod 13 = 1$
$3^7 \bmod 13 = 3$	$3^8 \bmod 13 = 9$	$3^9 \bmod 13 = 1$
$3^{10} \bmod 13 = 3$	$3^{11} \bmod 13 = 9$	$3^{12} \bmod 13 = 1$

Solo se debe obtener 1
en el caso $g^{p-1} \bmod p = 1$

Se repiten los restos 3, 9 ,1 porque g=3 no es un generador del grupo p=13

Un ataque por fuerza bruta deberá buscar solo en 1/4 del espacio de claves

La probabilidad de encontrar la clave secreta aumenta de 1/12 a 1/3
(usando la ecuación $K_P = g^{Ks} \bmod p$)

Diffie-Hellman: Seguridad (3)

Ejemplo de una BUENA elección de parámetros p=13, g=2 $K_P = 2^{K_s} \bmod 13$

Cálculo de las claves públicas que se pueden generar

$2^1 \bmod 13 = 2$	$2^2 \bmod 13 = 4$	$2^3 \bmod 13 = 8$
$2^4 \bmod 13 = 3$	$2^5 \bmod 13 = 6$	$2^6 \bmod 13 = 12$
$2^7 \bmod 13 = 11$	$2^8 \bmod 13 = 9$	$2^9 \bmod 13 = 5$
$2^{10} \bmod 13 = 10$	$2^{11} \bmod 13 = 7$	$2^{12} \bmod 13 = 1 \leftarrow$ BIEN

Solo se debe obtener 1 en el caso $g^{p-1} \bmod p = 1$

Se generan TODOS los restos multiplicativos del grupo p=13
(porque 2 es un generador dentro de este grupo)

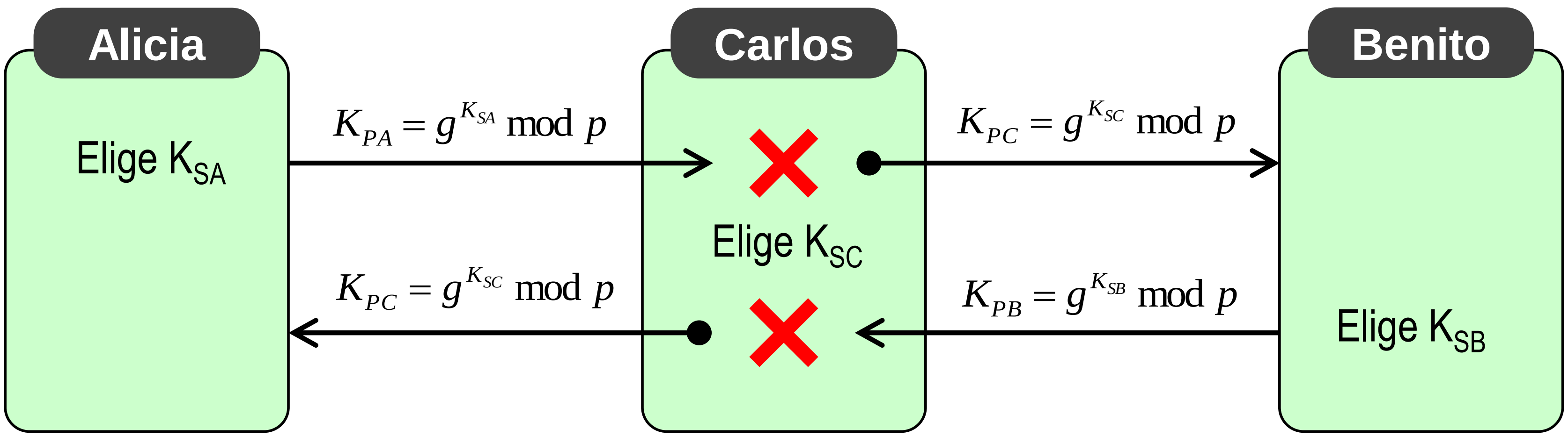
Observar que el valor 1 SOLO se obtiene para $g^{p-1} \bmod p$

Según la teoría de números, para p=13 serán generadores g=2, 6, 7 y 11

- La elección de las **claves secretas K_S**
  - Elegir K_S con un buen generador de enteros aleatorios en el rango 1 a p-1
  - Si las claves no son completamente aleatorias se facilita la tarea a un posible atacante

Diffie-Hellman: Ataque del hombre en medio

Cuando las claves públicas son enviadas mediante mensajes el protocolo DH es sensible al **ataque del hombre-en-medio** que intercepta los mensajes intercambiados



$$K_{SI} = K_{PC}^{K_{SA}} \bmod p \qquad K_{SIA} = K_{PA}^{K_{SC}} \bmod p \qquad K_{SIB} = K_{PB}^{K_{SC}} \bmod p \qquad K_{SI} = K_{PC}^{K_{SB}} \bmod p$$

$$K_{SI} = (g^{K_{SC}})^{K_{SA}} \bmod p \qquad K_{SIA} = (g^{K_{SA}})^{K_{SC}} \bmod p \qquad K_{SIB} = (g^{K_{SB}})^{K_{SC}} \bmod p \qquad K_{SI} = (g^{K_{SC}})^{K_{SB}} \bmod p$$

Carlos, recibe y descifra cada mensaje que le llega, y luego lo cifra y lo reenvía al destinatario

Para evitar este tipo de ataque se necesita un método para autenticar cada parte de una comunicación a la otra

Calculo de inversos modulares

El algoritmo extendido de Euclides permite encontrar soluciones a la identidad de Bezout

$MCD(a,b) = a \cdot x + b \cdot y$

El máximo común divisor de dos números se puede expresar como una combinación lineal de ambos números

El Alg-Ext de Euclides permite calcular x e y

El **inverso modular** de un número a se obtiene como la solución de:

$a \cdot x \equiv 1 \pmod{m}$

x es el número que multiplicado por a y calculado el módulo m del producto da 1

Por la definición de congruencia m es divisor de $(a \cdot x - 1)$

$(a \cdot x) \bmod m = 1$

Entonces ···

$a \cdot x - 1 = K \cdot m$

$a \cdot x - K \cdot m = 1$ ← Identidad de Bezout particular: $MCD(a,m) = 1$

a y m son conocidos
x y K se calculan con el Alg-Ext de Euclides → { x es el inverso de a
K se descarta }

Algoritmo extendido de Euclides

Cálculo de MCD(a,b) = g   a>b

K=0	a / b	$a = q_0 \cdot b + r_0$	$r_0 = a - q_0 \cdot b$
K=1	b / r_0	$b = q_1 \cdot r_0 + r_1$	$r_1 = b - q_1 \cdot r_0$
K=2	r_0 / r_1	$r_0 = q_2 \cdot r_1 + r_2$	$r_2 = r_0 - q_2 \cdot r_1$
K=3	r_1 / r_2	$r_1 = q_3 \cdot r_2 + r_3$	$r_3 = r_1 - q_3 \cdot r_2$
K=k	r_{k-2} / r_{k-1}	$r_{k-2} = q_k \cdot r_{k-1} + r_k$	$r_k = r_{k-2} - q_k \cdot r_{k-1}$
K=n-3	r_{n-5} / r_{n-4}	$r_{n-5} = q_{n-3} \cdot r_{n-4} + r_{n-3}$	$r_{n-3} = r_{n-5} - q_{n-3} \cdot r_{n-4}$
K=n-2	r_{n-4} / r_{n-3}	$r_{n-4} = q_{n-2} \cdot r_{n-3} + r_{n-2}$	$r_{n-2} = r_{n-4} - q_{n-2} \cdot r_{n-3}$
K=n-1	r_{n-3} / r_{n-2}	$r_{n-3} = q_{n-1} \cdot r_{n-2} + r_{n-1}$	$\underbrace{r_{n-1}}_{= g} = r_{n-3} - q_{n-1} \cdot r_{n-2}$
K=n	r_{n-2} / r_{n-1}	$r_{n-2} = q_n \cdot \underbrace{r_{n-1}}_{= g} + \underbrace{r_n}_{= 0}$	

Algoritmo extendido de Euclides (ejemplo)

Resolver $d \cdot 5 \equiv 1 \pmod{72}$ ←→ $(d \cdot 5) \bmod 72 = 1$

Cálculo del MCD(72,5)

m e

			Despejar los restos	
K=0	72 / 5	$72 = 14 \cdot 5 + 2$	$2 = \underline{72} - 14 \cdot \underline{5}$	
K=1	5 / 2	$5 = 2 \cdot 2 + 1$	$1 = \underline{5} - 2 \cdot 2$	$1 = \underline{5} - 2 \cdot (\underline{72} - 14 \cdot \underline{5})$
K=2	2 / 1	$2 = 2 \cdot 1 + 0$		$1 = \underline{5} - 2 \cdot \underline{72} + 28 \cdot \underline{5}$
				$1 = \mathbf{29} \cdot \underbrace{\underline{5}}_{e} - 2 \cdot \underbrace{\underline{72}}_{m}$

MCD(72,5) = 1 ¡OK!   0 → FIN

Inverso de e

Comprobar → $29 \cdot 5 = 145 \bmod 72 = 1$