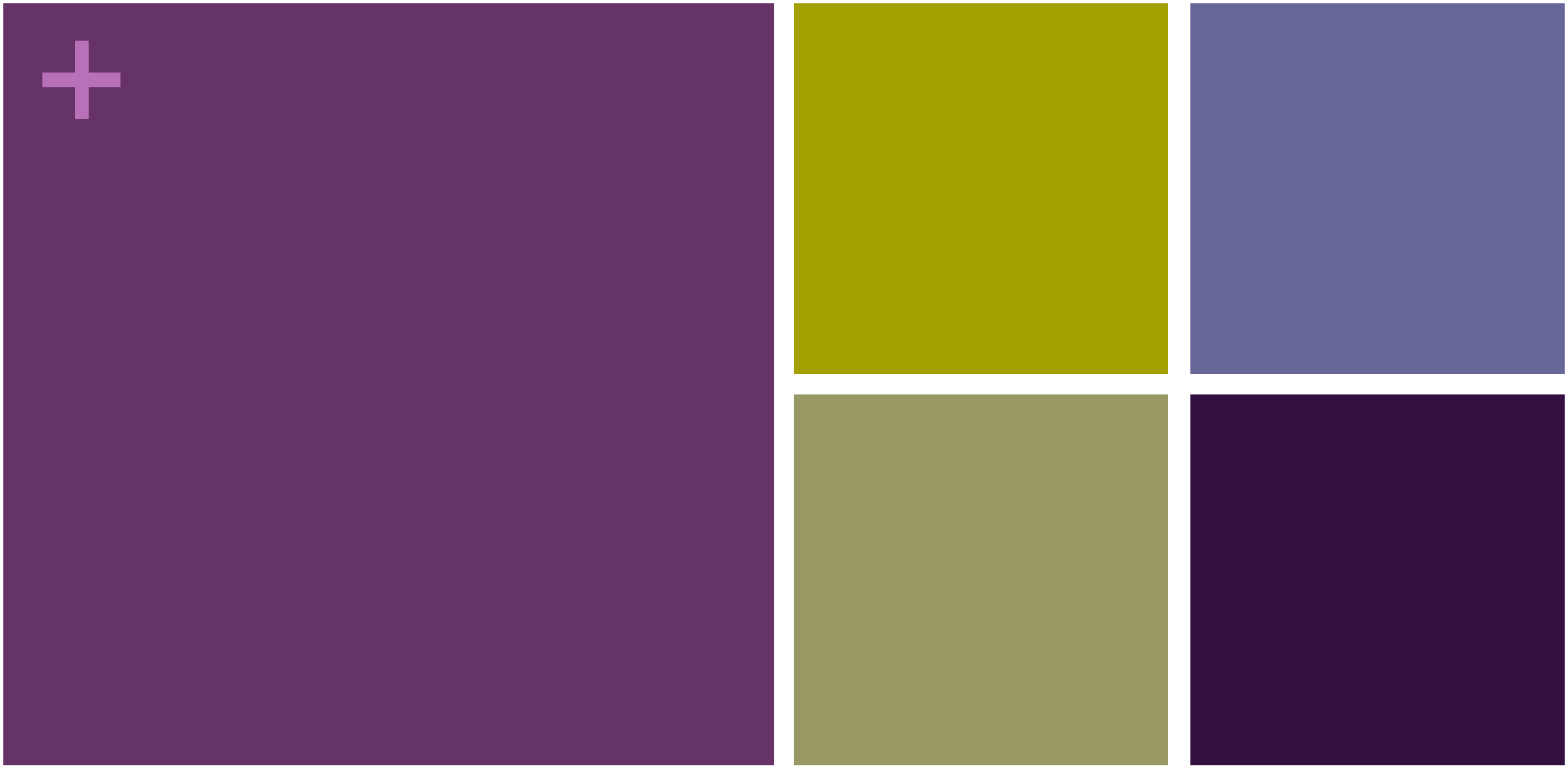


Asymmetric Key or Public Key Cryptography

Symmetric Key Problems

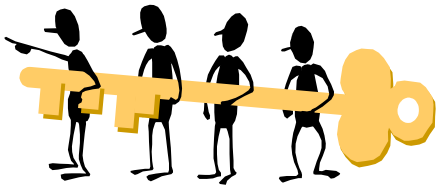
a) Key management
b) Key distribution.
c) No Digital Signature.

Why to use symmetric encryption?

- The speed of encrypting data is very high.
- With relatively small keys you can get a high-level of security.
- As they are based on non-linear functions the only practical attack is brute force one.

Asymmetric Encryption

- Each user has two keys, one secret and one public.
- Both keys are inverse within a field.
- To encrypt / decrypt trap one-way functions are used.



Trap One-way functions

- They are math functions that only allow to calculate results in an easy way to authentic users.
- The inverse direction is hard to non authentic users.
- For example:

  $f(M) = C$ it is always easy

  $f^{-1}(C) = M$ it is always hard but if you have the trap it becomes easy

Discussion

- Which one-way funcitons you know?
- How they work?

Trap One-way functions

Factorization Problem

Direct: product of two big prime numbers $p*q = n$
Inverse: factorization of a big number $n = p*q$

Discrete Logarithm Problem

Direct: discrete exponentiation $\beta = \alpha^x \bmod n$
Inverse: discrete logarithm $x = \log_\alpha \beta \bmod n$

Trap One-way functions

Knapsack Problem

Direct: Adding two elements with trap
Inverse: Adding two elements without trap

Primitive Root Modulo n Proble

Direct: Product Module n $x = a*a \bmod n$
Inverse: Root Module n $a = \sqrt{x} \bmod n$

Encrypting with receiver's public key

Sender

B

Keys: e_B, n_B, d_B

e_B, n_B: public

d_B: private

e_B and d_B are inverse over the same field n_B

if B uses A's public key (e_A, n_A), the transmitted information is <u>confidential</u>: only A can see it.

$C = E_{eA}(N) \bmod n_A$

Receiver

A

Keys: e_A, n_A, d_A

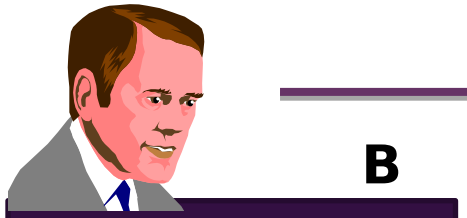
e_A, n_A: public

d_A: private

e_A and d_A are inverse over the same field n_A

Encrypt/Decrypt with receiver's public key

Encrypt:



B sends N encrypted to **A**

B

Keys: e_B, n_B, d_B

$C = E_{eA}(N) \bmod n_A$

A

Keys: e_A, n_A, d_A

Public Keys

Private Key

Decrypt:

$N = E_{dA}[E_{eA}(N)] \bmod n_A$

E_{dA} and E_{eA} are inverse

Confidentiality

Encrypting with sender's public key

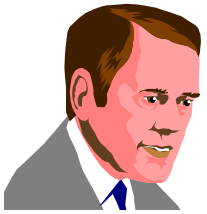
- This is no useful for transmitting data and guarantee confidentiality.
- It can be used to locally encrypt data.
- However, symmetric key algorithms are preferred given their higher efficiency.

Encrypting with sender's private key

- If now, the sender uses his private key to encrypt a message, the result will be a digital signature.
- This signature authenticates the sender.
- If, we also use a hash function, then we can also guarantee integrity of the message.
- Obviously, the sender cannot encrypt data using receiver's private key.

Encrypting with sender's private key

Sender



B

Now B encrypts using his private key d_B over the field n_B A will e able to verify that encryption using B's public key. In this way, A verifies message and entity authenticity.

A

Receiver

Keys: e_B, n_B, d_B

e_B, n_B: public

d_B: private

e_B and d_B are inverse over the field n_B

$$C = E_{dB}(N) \bmod n_B$$

Keys: e_A, n_A, d_A

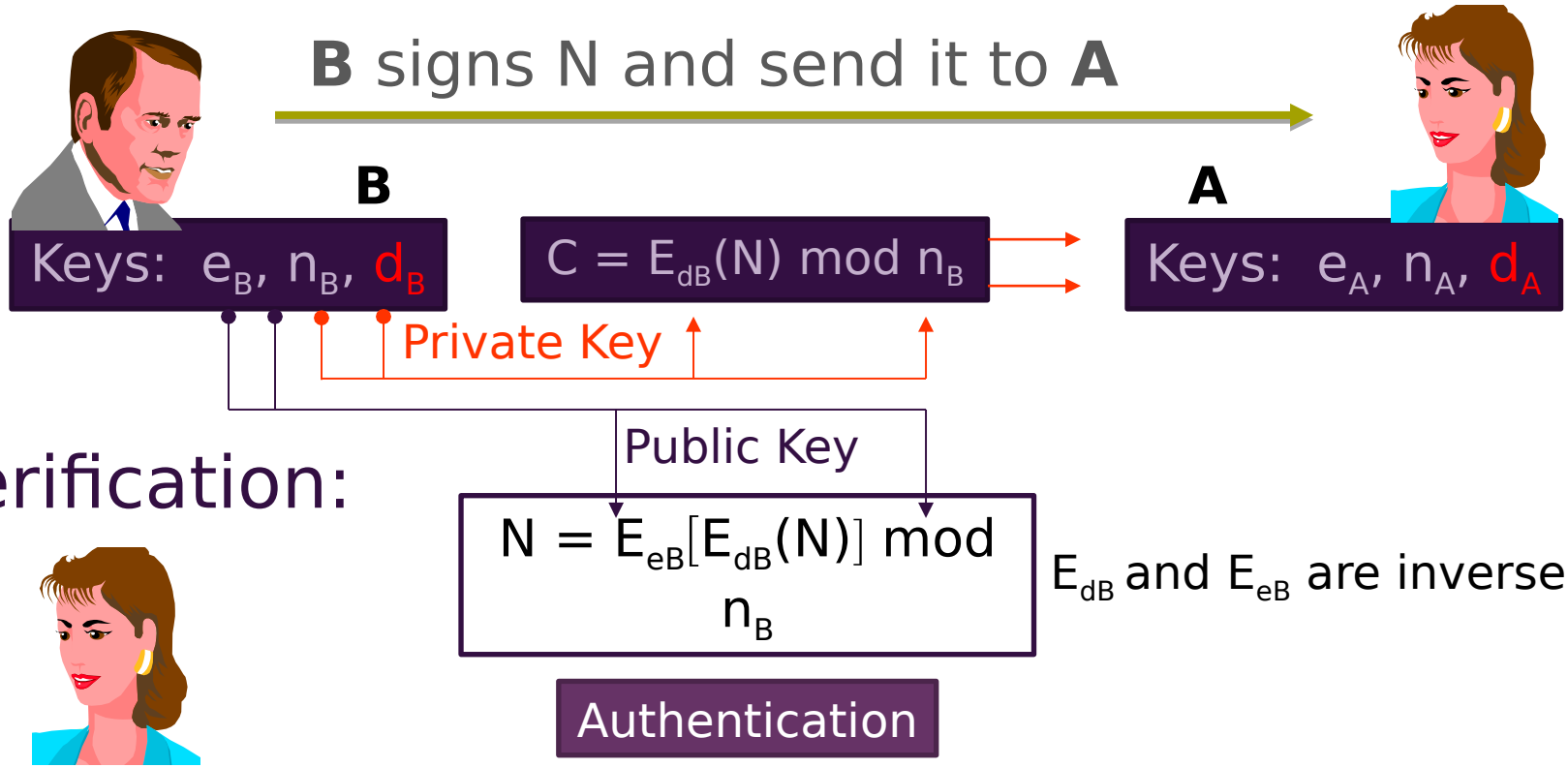
e_A, n_A: public

d_A: private

e_A and d_A are inverse over the field n_A

Encrypting with sender's private key

Digital Signature:



Verification:

Using Asymmetric Key Schemes

- By using receiver's public key, symmetric session keys can be securely exchanged.
- Using sender's private key, the hash value of messages can be signed.
- Remember using these schemes is slower than the symmetric ones.

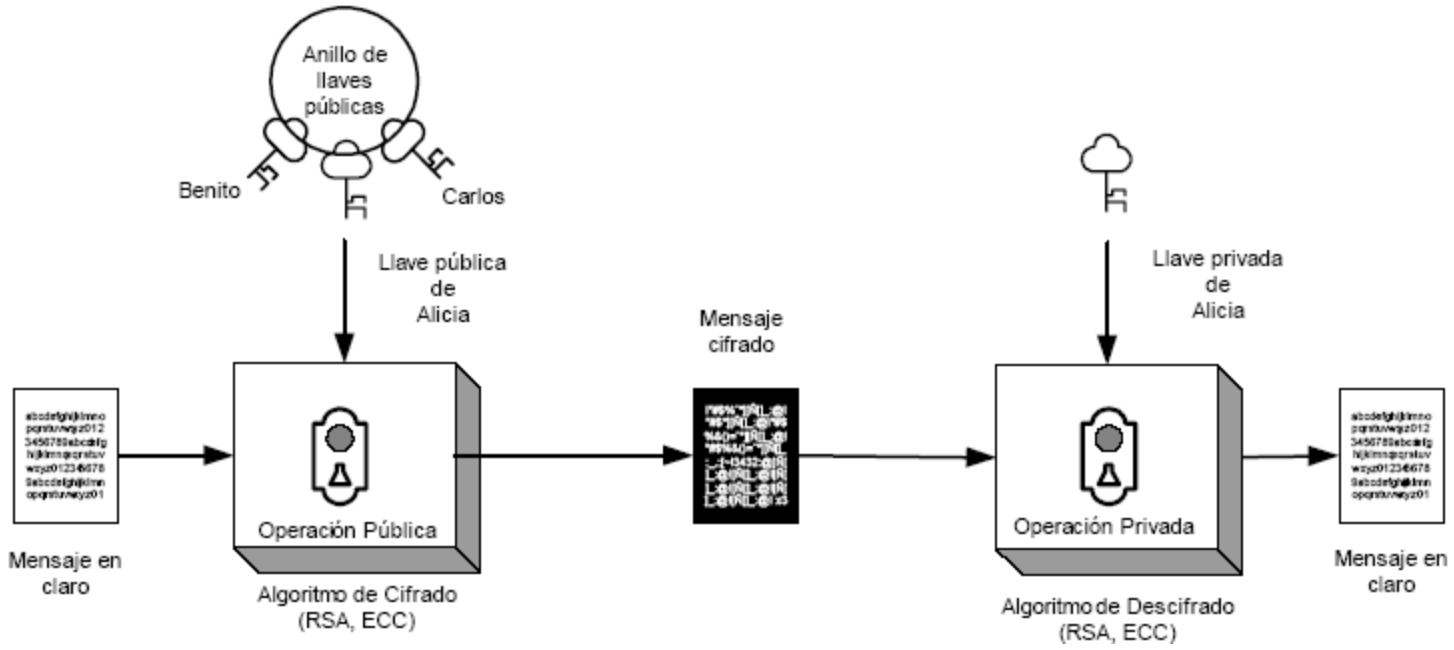
Comparing both schemes

Symmetric Key	Asymmetric Key
One has to memorize a big number of keys n^{2}.	One has to memorize one key (my private key).
Key is relatively small (hundreds).	Key is big (thousands).
Used as sessions keys, so life is short (hours).	Life is longer (months).
Provides message authentication.	Provides message and entity authentication.
Efficient for encrypting data.	Not very efficient with big amounts of data.

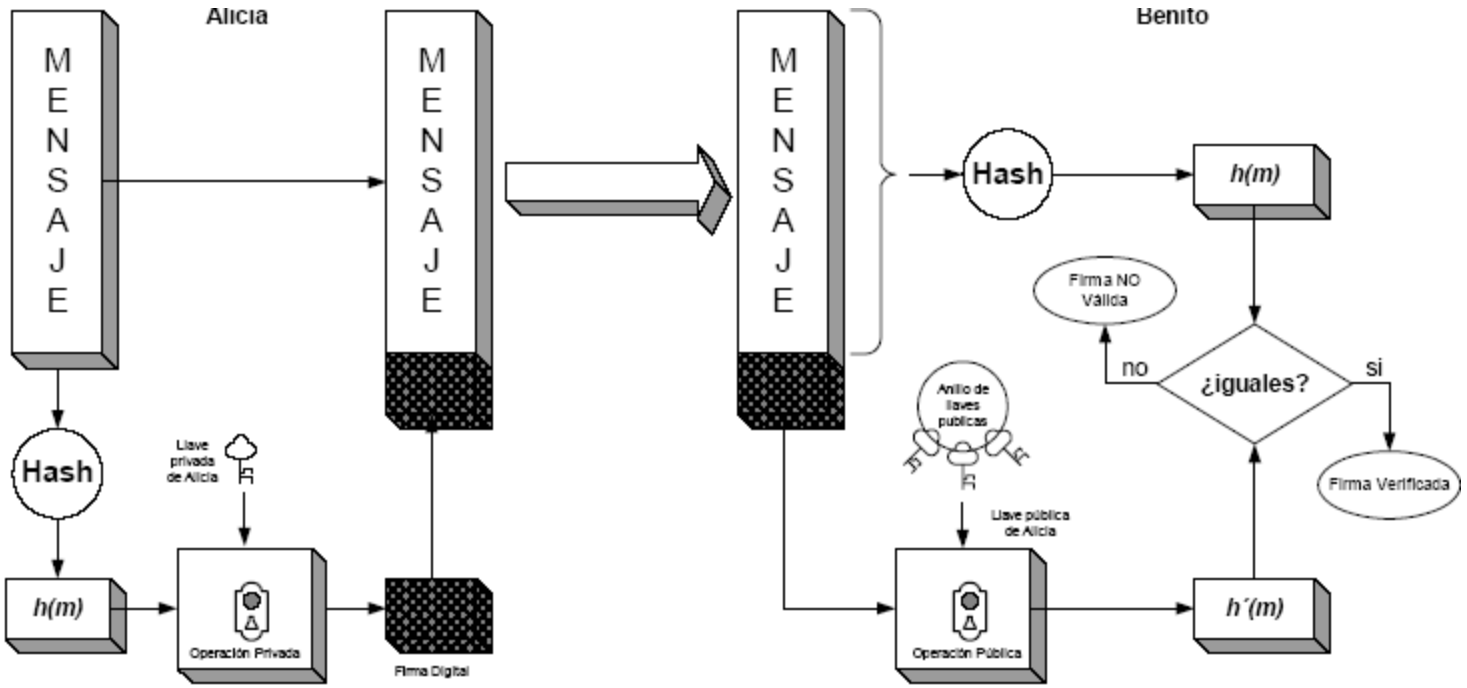
Comparing both schemes (properties)

Symmetric Key	Asymmetric Key
Confidentiality	Confidentiality
Partial Authentication	Total Authentication
No Digital Signature	Digital Signature
Keys: - Short length - Short life - Many keys	Keys: - Long length - Long life - Few keys
High Speed	Low Speed

Assymetric Encryption



Digital Signature



Activity

- Make a small research on the Internet and list 4 assymetric key schemes.
- From these algorithms, mention which trap one-way function they use.

+

RSA Algorithm

RSA Key exchange (B → A)

B

The key K will be exchanged

Be K = DA9F (16 bits)

A

Keys of B
$n_B = 65.669$
$e_B = 35$, $d_B = 53.77$

$2^{16} < 66.331 < 2^{17}$
Forcing to encrypt
a 16 bits block

Keys of A
$n_A = 66.331$
$e_A = 25$, $d_A = 18.377$

Encrypt

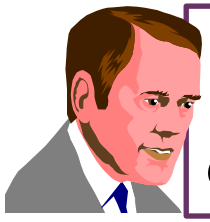
$K = DA9F_{16} = 55.967_{10}$

$C = K^{eA} \bmod n_A$

$C = 55.967^{25} \bmod 66.331 = 16.667$

B sent to A, C = 16.667

A recovers K



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

Keys
$n_A = 66.331$
$e_A = 25, d_A = 18.377$

B

A

$K = DA9F_{16} = 55.967_{10}$

$C = K^{eA} \bmod n_A \quad C = 55.967^{25} \bmod 66.331 = 16.667$

B sent to A: C = 16.667

A calculates:

- $C^{dA} \bmod n_A = 16.667^{18.377} \bmod 66.331 = 55.967$.
- Only A can calculate the result using her private key d_A.

B used prime numbers are (97, 677) and A prime numbers are (113, 587)

RSA Algorithm

- In February 1978, Ron Rivest, Adi Shamir y Leonard Adleman propose the public key algorithm called RSA.
- Steps

1. Each user choose a group $n = p*q$
2. p and q are private values.
3. Each user calculates $\phi(n) = (p-1)(q-1)$.
4. Each users choose a public key e such that $1 < e < \phi(n)$ that holds the condition $mcd\ [e, \phi(n)] = 1$
5. Each user calculate the private key $d = inv\ [e,\phi(n)]$.
6. The user makes public the group n and the key e.
7. d is kept in secret including p and q.
   - Encrypt: $C = N^{eR} \bmod n_R$
   - Sign: $C = h(M)^{dS} \bmod n_S$

https://www.youtube.com/watch?v=vgTtHV04xRI&feature=youtu.be

Ejemplo

Generación de Llaves

1. Cada usuario elige dos números primos distintos p y q tal que $p \neq q$ ambos deben ser del mismo tamaño.
   - Estos números deben escogerse de forma aleatoria (test de primarilidad).
2. Se calcula $n = pq$
3. Selecciona un entero impar e que es primo relativo de $\phi(n)$, esto será igual a $(p-1)(q-1)$
4. Se calcula d como el inverso multiplicativo de e modulo $\phi(n)$.
5. Se hace público el par $P = (e, n)$ como la **llave pública** del usuario.
6. Se mantiene en secreto el par $S = (d, n)$ como la **llave privada** del usuario.

Ejemplo

Encontrar dos números primos distintos, **p** y **q**, y multiplicarlos entre sí para obtener un nuevo número **n**.

p = 3

q = 11

n = p * q = 3 * 11 = 33

phi(n) = 2 * 10 = 20

Ejemplo

- La clave pública será un número aleatorio que sea primo relativo entre **1** y **phi(n)** al cual llamaremos **e**, y usaremos el **7**.
- A la inversa la llamaremos **d** y surge de obtener el número que multiplicado por **e** nos de **1 mod(phi(n))**.

Ejemplo

- Para encontrar **d** tenemos que iterar desde **1** hasta **20** (phi(n)) y ubicar un número tal que multiplicado por **7** nos de **1** en módulo **20**.

$7 * d = 1 \bmod (20)$

$7 * 3 = 21$

$21 \bmod 20 = 1$

d = 3 (exponente privado)

Llave Pública: *(7, 33)*

Llave Privada: *(3, 33)*

Cifrar y descifrar

- Para este esquema el dominio es el conjunto $\mathbb{Z}_n$
- Si queremos cifrar un mensaje M usando la llave pública P, calculamos:

$$P(M) = M^e \ mod \ n$$

- Si queremos descifrar un texto cifrado C asociado con la llave secreta S, calculamos:

$$S(C) = C^d \ mod \ n$$

- Este esquema se puede usar también para implementar firma digital.

Ejemplo

- Para cifrar un mensaje, lo que debemos hacer es elevar el dato dado a la **e** (parte de la llave pública).
- El procedimiento para descifrar un mensaje se apoya en la propiedad de que *si multiplicamos un número x por sí mismo* ***phi(n)*** *veces, el resultado es 1 en módulo n.*
- Esta propiedad es la base del algoritmo RSA.

Cifrar y descifrar

- Tomando los datos del ejemplo anterior. Supongamos que queremos cifrar el número 2.
- *dato = 2*
- Cifrar:
  - $dato^{e} \bmod n$
  - $2^{7} \bmod 33 = 128 \bmod 33 = 29$
  - Este es el dato cifrado.
- Descifrar:
  - $datoCifrado^{d} \bmod n$
  - $29^{3} \bmod 33 = 24389 \bmod 33 = 2$
  - Este es mi dato original

Inversos módulo n:

- Existencia de inversos: $\forall x \in \mathbb{Z}_n$

$$\exists x^{-1} |\, x \cdot x^{-1} \equiv 1 \bmod n \Leftrightarrow \mathrm{mcd}(x, n) = 1$$

- Cálculo: algoritmo extendido de Euclides.
- Ejemplo de cálculo manual de $17^{-1} \bmod 65 = 23$:

(1)	**65**	=	3 ·	**17**	+	14
(2)	17	=	1 ·	14	+	3
(3)	14	=	4 ·	3	+	2
(4)	3	=	1 ·	2	+	**1**

$$1 = \overbrace{3 - 1 \cdot 2}^{(4)} = 3 - \overbrace{(14 - 4 \cdot 3)}^{(3)} = -14 + 5 \cdot 3 =$$

$$= -14 + 5\overbrace{(17 - 1 \cdot 14)}^{(2)} = 5 \cdot 17 - 6 \cdot 14 =$$

$$= 5 \cdot 17 - 6\overbrace{(65 - 3 \cdot 17)}^{(1)} = \mathbf{23} \cdot 17 - 6 \cdot 65$$

- Teorema de Fermat para p primo:

$$0 < x \leq p - 1 \Longrightarrow x^{p-1} \equiv 1 \bmod p$$

Teorema Chino del Resto (TCR)

Si $n_1, \ldots, n_k$ son enteros primos entre sí dos a dos, $n = n_1 \cdots n_k$ y $a_1, \ldots, a_k \in \mathbb{Z}$, se trata de resolver el sistema de ecuaciones:

$$\begin{cases} x \equiv a_1 \bmod n_1 \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ x \equiv a_k \bmod n_k \end{cases}$$

- El sistema tiene solución única $\bmod n$.
- Se calculan: $c_i = \frac{n}{n_i}, d_i = c_i^{-1} \bmod n_i \ \ \forall i = 1, \ldots, k.$
- Todas las soluciones son:

$$x = a_1 c_1 d_1 + \cdots + a_k d_k c_k + \lambda \cdot n, \qquad \forall \lambda \in \mathbb{Z}$$

Cálculo de potencias enteras

- La idea:

$$m^e = \begin{cases} m & \text{si } e = 1 \\ \left(m^{\left\lfloor \frac{e}{2} \right\rfloor}\right)^2 & \text{si } e \text{ es par} \\ m \cdot m^{e-1} & \text{si } e \text{ es impar} \end{cases}$$

- El seudocódigo de una versión no recursiva sería así:

```
s = 1, t = m, q = e;
mientras (q > 0)
    si (q & 1) // si q es impar
        s = s * t;
    q = q / 2; // división entera(desplazamiento)
    t = t * t;
retorna s;
```

- Sin ordenador es un poco diferente. Ejemplo de cálculo de 2^{100}:

$100 = 1100100_2 \longrightarrow 1C1C0C0C1C0C0 \longrightarrow MCMC0C0CMC0C0 \longrightarrow MCMCCCMCC$

1 —M—▶ 2 —C—▶ 4 —M—▶ 8 —C—▶ 64
—C—▶ 4096 —C—▶ 16777216 —M—▶ 33554432
—C—▶ 1125899906842624
—C—▶ 1267650600228229401496703205376

No son 100 multiplicaciones, sino sólo 9 (máximo = 2 · nº de bits del exponente - 1).

Construcción de primos grandes

Se construye un entero impar de k bits:

El primero y el último bit se ponen a 1 y los bits intermedios se eligen al azar.

La probabilidad de que el número así construido sea primo es de $2 \,/\, (k \cdot \ln 2)$

Si el número obtenido no es primo volvemos a sortear los bits intermedios y repetimos hasta obtener un primo.

¿Cómo saber si hemos encontrado un primo?

Test de primalidad de Miller-Rabin

Entrada: n impar, con $n = 2^s \cdot r + 1$ con r impar.

Salida: *'compuesto'* si el número no es primo; *'probable primo'* si el número es probablemente primo, con probabilidad de error de ¼.

- Se elige al azar un entero a con $1 < a < n$.
- Si $a^r \equiv 1 \bmod n$, retorna *probable primo*.
- Para $j = 0 \ldots s - 1$
  - Si $a^{2^j r} \equiv -1 \bmod n$, *retorna probable primo*.
- Retorna *compuesto*.

Si se obtiene k veces la respuesta *probable primo*, entonces la probabilidad de error es inferior a $1/2^{2k}$, lo que lo hace *muy* fiable.

Búsqueda de primos fuertes: test de Gordon

1. Elige dos primos r, s de tamaño adecuado.
2. Calcula t como el menor primo de la forma $t = a \cdot r + 1$ con $a \in \mathbb{Z}$.
3. Calcula $p_0 = (s^{t-1} - t^{s-1}) \bmod (s \cdot t)$
4. Calcula p como el menor primo de la forma $p = p_0 + a \cdot t \cdot s$ con $a \in \mathbb{Z}$.

Datos:

- Un entero n, que se sabe que es producto de dos primos p y q (desconocidos).
- Un entero e, que sabemos que es primo con $p-1$ y con $q-1$.
- Un entero c, que sabemos que es el resultado de elevar un número desconocido m a e módulo n.

Problema:

- Encontrar un entero m tal que $m^e \equiv c \ (mod\ n)$.

1. Si sabemos factorizar $n = p \cdot q$, entonces el problema RSA se resuelve fácilmente:
   - Calculamos $\phi = (p-1)(q-1)$
   - Hallamos $d = e^{-1}$ en $\mathbb{Z}_\phi$
   - La solución es $m = c^d \ mod \ n$.

2. Conociendo ϕ sería fácil encontrar p y q con sólo resolver una ecuación de segundo grado.

3. No está demostrado que el problema RSA sea equivalente al de la factorización, pero el problema de determinar d sí lo es, ya que conocido éste existe un sencillo ataque que nos daría la factorización de n.

- Como $ed \equiv 1 \ (mod \ \phi)$, existe un entero k con $ed = 1 + k\phi$.
- Si $mcd(m, p) = 1$entonces, por el Teorema de Fermat, $m^{p-1} \equiv 1 \ (mod \ p)$. Elevando ambos miembros a $k(q-1)$ y multiplicando por m queda

$$m^{ed} = m^{1+k(p-1)(q-1)} \equiv m \ (mod \ p).$$

- Si $mcd(m, p) = p$ entonces ambos miembros son $0 \ (mod \ p)$, con lo que la congruencia anterior también se verifica.
- Por un argumento similar $m^{ed} \equiv m \ (mod \ q)$, y como p y q son primos, de aquí se sigue que

$$m^{ed} \equiv m \ (mod \ n)$$

Idea: trasladar las operaciones de $\mathbb{Z}_n$ a $\mathbb{Z}_p \times \mathbb{Z}_q$

Se usa el Teorema Chino del Resto para descifrar c y obtener m:

1. $a_p = q^{p-1} \, mod \, n, a_q = p^{q-1} \, mod \, n$
2. $d_p = d \, mod \, (p-1), d_q = d \, mod \, (q-1)$
3. $c_p = c \, mod \, p, c_q = c \, mod \, q$
4. $m = a_p \left(c_p^{d_p} mod \, p \right) + \, a_q \left(c_q^{d_q} mod \, q \right) mod \, n$

Se hacen más operaciones pero más rápidas; además los valores de los pasos 1 y 2 pueden estar precalculados.

- **En 1977 Ron Rivest profetizó que factorizar un número de 125 dígitos llevaría 40.000 billones de años.**
- **El desafío RSA-129 (129 dígitos, 425 bits):**
  - 114381625757888867669235779976146612010218296721242362562561842935706935245733897830597123563958705058989075147599290 026879543541
    = 3490529510847650949147849619903898133417764638493387843990820577 * 327691329932667095499619881908344614131776429679929425397982885 33
  - Resuelto en abril de 1994 por un equipo dirigido por D. Atkins, M. Graff, A. Lenstra y P. Leyland usando 600 ordenadores conectados a través de Internet.

- **Con el siguiente escenario:**
  - Capacidad de cálculo (mips por año):

Particular (P)	Gran empresa (E)	Gobierno (G)
10.000	10.000.000	1.000.000.000

  - La potencia de cálculo se multiplica por 10 cada 5 año y las Matemáticas avanzan cada año.
- **Tamaños recomendados de claves:**

Año	Contra P	Contra E	Contra G
1995	768	1280	1536
2000	1024	1280	1536
2005	1280	1536	2048
2010	1280	1536	2048
2015	1536	2048	4096
2045	8192	16384	32768

1. **Algunos valores del mensaje pueden dar cifrados inseguros:**
   - Los valores m=0 y m=1 siempre se cifran en sí mismos.
   - Con exponentes pequeños y valores pequeños de m, el cifrado podría ser estrictamente menor que el módulo y el texto en claro podría obtenerse haciendo la raíz e-ésima, sin tener en cuenta el módulo.

2. **Siempre hay mensajes que se cifran en sí mismos (al menos 9). De hecho su número exacto es :**

$$(1 + \text{mcd}(e-1, p-1)) \cdot (1 + \text{mcd}(e-1, q-1))$$

3. **RSA es determinista**, por lo que es viable un ataque de texto elegido: el atacante construye un diccionario de textos probables y sus cifrados. Interceptando un texto cifrado, el atacante puede usar este diccionario para descifrar el mensaje.

- Protección: combinar RSA con algún esquema de relleno (como por ejemplo estándares como PKCS).

- **Fuerza bruta:**

  Si p y q son primos cercanos, p está cerca de $\sqrt{n}$; así sería viable un ataque de fuerza bruta dividiendo n entre impares menores que $\sqrt{n}$.

- **Factorización de Fermat:**

  Si llamamos $x = \frac{p+q}{2}, y = \frac{p-q}{2}$ entonces $x^2 - n = y^2$. Como $x^2 > n$, la idea es probar *todos* los $x > \sqrt{n}$ hasta dar con uno que $x^2 - n$ sea cuadrado perfecto. Cuanto más cercanos sean p y q, más pequeño será y, y harán falta menos iteraciones.

  Con x e y, calculamos $p = x + y,\ q = x - y$.

- **Mersenne propuso a Fermat el problema de decidir si era primo el número** 2027651281
- **Fermat contestó rápidamente:**

$$2027651281 = 44021 \cdot 46061$$

- **El método:**

$$\sqrt{n} = 45029{,}45\ldots$$

x	$\sqrt{x^2 - n}$
45030	222,75…
45031	373,74…
45032	479,32…
.....	.....
45041	1020,00…

$$2027651281 =$$
$$45041^2 - 1020^2 =$$
$$(45041 + 1020) \cdot$$
$$(45041 - 1020) =$$
$$44021 \cdot 46061$$

Además de la clave privada d pueden encontrarse otras claves que también sirvan para descifrar, aunque no sean $d = e^{-1} (\mathrm{mod}\, n)$:

Llamando:

$$\gamma = \mathrm{mcm}\,(p-1, q-1)$$

$$d_\gamma = e^{-1}\,(\mathrm{mod}\,\gamma)$$

$$\lambda = \left\lfloor \frac{n - d_\gamma}{\gamma} \right\rfloor$$

Habrá λ claves d_k que descifran (además de d):

$$d_k = d_\gamma + k \cdot \gamma, \quad k = 0,1,\ldots,\lambda$$

Ejemplo:

Clave pública $(n, e) = (3053 = 71 \cdot 43, 157)$.

Clave privada $d = 157^{-1} \bmod (70 \cdot 42) = 1573$.

$$\gamma = \text{mcm}(70,42) = 210.$$

$$d_\gamma = 157^{-1} \bmod 210 = 103.$$

Claves que descifran:

$$103 + k \cdot 210, \quad k = 0, \ldots, \frac{3053 - 103}{210} = 14$$

103, 313, 523, 733, 943, 1153, 1363, **1573**,

1783, 1993, 2203, 2413, 2623, 2833, 3043

Algunos criterios para una buena clave:

- p y q deben ser suficientemente grandes, del mismo tamaño, pero no muy cercanos.
- Debe maximizarse $\text{mcm}\,(p-1, q-1)$
- Deben minimizarse
  $$\text{mcd}(e-1, p-1) \text{ y } \text{mcd}\,(e-1, q-1)$$
- p y q deben ser primos *fuertes*. Un primo p es fuerte si:
  - $p+1$ tiene un factor primo grande.
  - $p-1$ tiene un factor primo r grande.
  - $r-1$ tiene un factor primo grande.

Existen algoritmos para generar buenas claves y pruebas para validar su calidad.

- **Error: elegir un mismo módulo n y distribuir distintos pares (e, d) a los usuarios de la organización.**
  - Por lo delicado de la buena elección del módulo, es un error bastante frecuente.
  - El problema es que el conocimiento de un simple par (e, d) puede revelar la factorización de n y romper todo el sistema.
- **Debe evitarse el uso repetido de un pequeño exponente de cifrado.**
  - Por ejemplo, un mensaje enviado a 3 destinatarios que han elegido e=3, es muy probable que pueda ser descifrado usando el TCR.

Propiedades

- Cifrando y descifrando
- ¿Qué propiedad le añado a la información?
- ¿Qué otras propiedades podría garantizar con este esquema?

+

El Gammal Algorithm

ElGamal

- In 1985 Taher ElGamal proposed an algorithm that uses the Discrete Logarithm Problem.
- A multiplicative group Z_p^* is chosen, where p is a big prime.
- From p, a root α is chosen as group generator.
- Each user choses a random number λ inside p.
  - λ is the private key.
- Each user calculates α^{λ} mod p
  - The values (α^{λ} mod p) y p are the public key

El problema del logaritmo discreto

- Un grupo $(G, \cdot)$ de n elementos es cíclico si existe un $g \in G$ *(generador)* tal que $G = \{1, g, g^2, \cdots, g^{n-1}\}$. Entonces para todo $a \in G$ existe un entero k con $0 \leq k \leq n-1$ tal que $g^k = a$. Se dice que k es el logaritmo discreto de a de base g.

- El problema del logaritmo discreto consiste en hallar k conocidos G, g y a.

- En grupos finitos grandes (p. e. en el grupo multiplicativo de $\mathbb{Z}_p$ con p un primo fuerte), este problema es computacionalmente intratable.

Búsqueda de un generador

Entrada: G cíclico, con $|G| = n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$.

Salida: un generador g de G.

1. Elige al azar $g \in G$
2. Para $i = 1$ hasta k
   - Si $g^{n/p_i} = 1$ vuelve al paso 1.
3. Retorna g.

La probabilidad de que g sea generador es $\phi(n)/n$, y un valor del número estimado de intentos necesarios para encontrarlo es $6 \cdot \ln(\ln(n))$.

Cada usuario crea su clave pública y la correspondiente clave privada:

1. Construye un primo grande p y un generador del grupo multiplicativo de $\mathbb{Z}_p$.

2. Elige aleatoriamente un entero a, $1 \leq a \leq p - 2$ y calcula $\beta = g^a \bmod p$.

3. La clave pública es (p, g, β). La clave privada es a.

B cifra un mensaje para A

1. Obtiene la clave pública de A (p, g, β) .
2. Representa el mensaje como un entero m en el intervalo $[0, p-1]$.
3. Elige al azar un entero k, $1 \leq k \leq p - 2$.
4. Calcula $\gamma = g^k \bmod p$ y $\delta = m \cdot \beta^k \bmod p$.
5. Envía el texto cifrado $c = (\gamma, \delta)$ a A.

A descifra el mensaje de B

1. Usa su clave privada para recuperar $m = \gamma^{p-1-a} \cdot \delta \bmod p$.

- El proceso de descifrado hace sólo una potencia, pero el de cifrado requiere dos, de exponente *k*, que pueden hacerse más rápidas eligiendo adecuadamente el exponente.

- Una desventaja es que el mensaje cifrado es el doble de largo que el original.

- Una gran ventaja es que al elegirse *k* de forma aleatoria, el cifrado de un mismo texto dará resultados diferentes, aunque esto es transparente para el descifrado.

- La seguridad de Elgamal se basa en la dificultad de resolver un problema de logaritmo discreto.
- Resulta crítico que se usen diferentes enteros aleatorios k para cada mensaje. Si se usara el mismo k para m_1 y m_2, dando lugar a los cifrados (γ_1, δ_1) y (γ_2, δ_2), entonces $\delta_1 / \delta_2 = m_1 / m_2$ y el conocimiento de uno de los mensajes revelaría el otro.

ElGamal Encryption

- A encrypts N to send it to B
- B has chosen its private key b within a multiplicative group an the prime number p, which is public.
- B makes public the value of α^b mod p.
- A generates a random number ν (used for a session) and calculates α^ν mod p.
- With B's public key (α^b) A calculates:
  - $(\alpha^b)^\nu$ mod p and $N*(\alpha^b)^\nu$ mod p
- A sends B the pair: $C = [\alpha^\nu \text{ mod } p, N*(\alpha^b)^\nu \text{ mod } p]$

ElGamal Decryption

- B decrypts C:
- B receives C = [α^v mod p, N$*$(α^b)^v mod p].
- B takes α^v mod p and calculates (α^v)^b mod p.
- B calculates:
  - [N$*$(α^b)^v mod p] / [(α^v)^b mod p]
  - Remember: (α^b)^v = (α^v)^b
- This is possible as exist the inverse of (α^v)^b in the group p. Then:

$$[N*(\alpha^b)^v * \{inv\ (\alpha^v)^b, p\}] \bmod p = N$$

Resumen de los sistemas de clave pública

Pros y contras de los Sistemas de Clave Pública

- Emisor y receptor generan un par de claves, pública y privada, relacionadas por una función con trampa (trap-door).
- Emisor y receptor de un mensaje usan claves diferentes para las operaciones de cifrado, descifrado y firma.
- La seguridad del sistema va asociada a la resolución de un problema matemático de difícil solución en el tiempo.
- Firma digital completa: autentican al mensaje y al emisor.
- Es necesario contar con mecanismos de certificación para asegurar la veracidad de las claves públicas.
- Son sistemas de cifra son muy lentos comparados con los sistemas de llave simétrica.