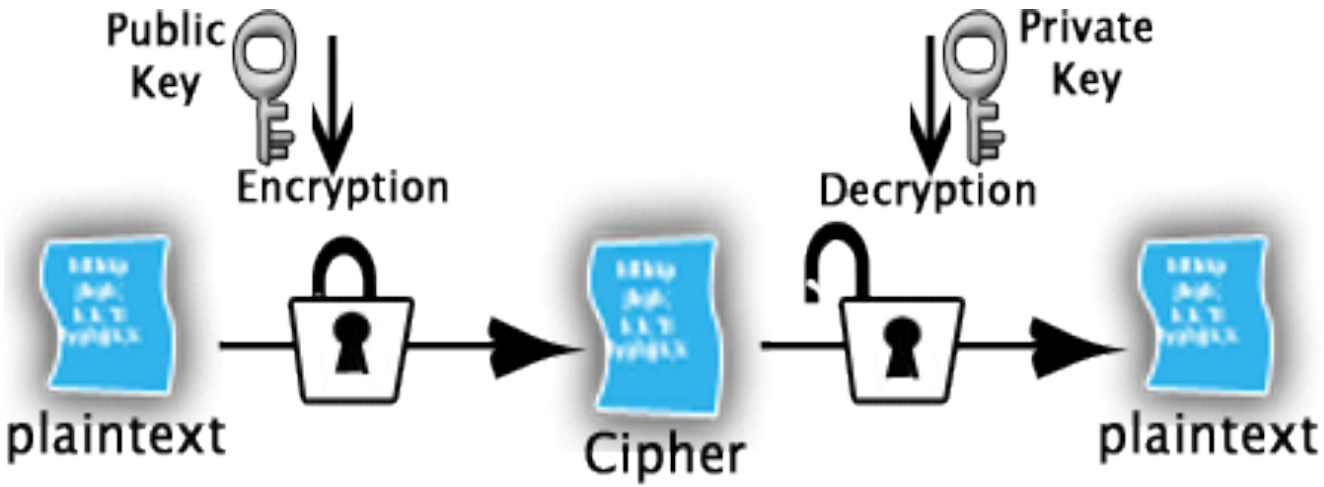
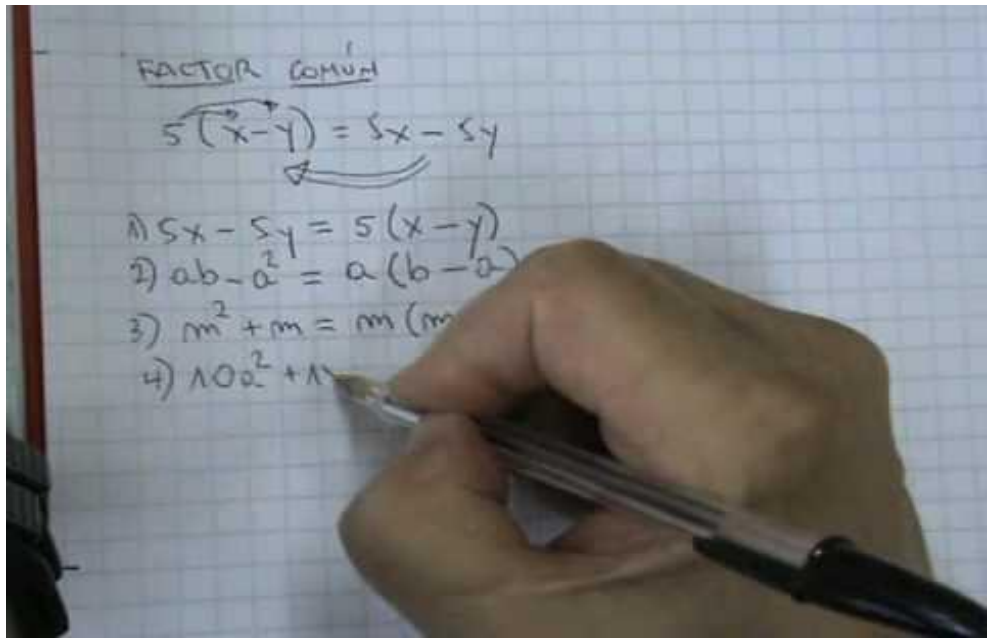


Cifrado Asimétrico

¿En qué consiste?

- Nuevo paradigma de cifrado publicado en 1976.
- Seguridad de estos algoritmos se basa en dificultad de factorizar números grandes.
- Dos llaves: una pública y una privada (secreta)
- Todo lo que se cifra con una llave sólo se puede descifrar con la otra llave.



Cifrador de Mochila

- Se da una sentencia de números positivos, supongamos, n = 2,4,6,7,8,10,14.
- Debemos de saber cuantos valores tendremos en la mochila, en este caso, c = 7.
- El problema consistirá en encontrar un objetivo ‘O’.
- Se necesitará un conjunto de números de la sentencia que sumados puedan representar el objetivo.

¿Cómo llegar al objetivo?

n = {2, 4, 6, 7, 8, 10, 14}, c = 7, O = 12
Si n1 = 2 <= 12? O = 12 – 2 será 10 = SI
Si n2 = 4 <= 10? O = O – 6 será 6 = SI
Si n3 = 6 <= 6? O = O – 6 será 0 = SI
Si n4 = 7 <= 0? O = O – 7 será 0 = No
Si n5 = 8 <= 0? O = O – 8 será 0 = No
Si n6 = 10 <= 0? O = O – 10 será 0 = No
Si n7 = 14 <= 0? O = O – 14 será 0 = No

Robustez de la mochila

Encontramos {1,1,1,0,0,0,0} equivalente a la suma de {2,4,6} =12
También existe:
{1,0,0,0,0,1,0} = {2,10} = 12
{0,1,0,0,1,0,0} = {4,8} = 12

Esta mochila es inapropiada para un sistema de cifrado ya que este solo va a necesitar una sola operación para realizar el descifrado que por lógica vendrá de un único mensaje y será inaudito que haya más de una solución para el objetivo.

Elección de la mochila

- Al existir múltiples soluciones se debe crear una nueva mochila.
- Esta nueva mochila se conoce por simple o super creciente, que nos servirá para poder crear una mochila más segura.
- Este tipo de mochila se basa en que el objetivo es mayor que cualquier número de la sentencia que se utiliza.
- Si tenemos una sentencia con n = {4,7,11,34,55,67,78,87,99,234} de 10 elementos y se tendrá que dar como objetivo un número mayor a 234 para que la mochila se considere segura, supongamos que objeto es 308 {0,1,0,0,0,1,0,0,0,1} y su solución será única dentro de la sentencia.

Ejemplo de mochila

Sea el M="HOLA"
Sea n = {2,4,5,34}
Transformemos nuestro mensaje a código ASCII ext:
H = 01001000, O = 01001111, L = 01001100, A = 01000001

Dividimos los valores ASCII en | n |
Mensaje = 0100 1000 0100 1111 0100 1100 0100 0001

Mensaje cifrado
C_i = 4, 2, 4, (2+4+5+34), 4, (2+4), 4, 34
C_i = 4, 2, 4, 45, 4, 6, 4, 34

Protocolo de Diffie y 

- Diffie-Hellman permite el intercambio secreto de claves entre dos partes que no han tenido contacto previo, utilizando un canal inseguro, y de manera anónima (no autenticada).
- Se emplea generalmente como medio para acordar claves simétricas que serán empleadas para el cifrado de una sesión.
- Siendo no autenticado, sin embargo provee las bases para varios protocolos autenticados.
- Su seguridad radica en la extrema dificultad de calcular logaritmos discretos.

Logaritmo Discreto

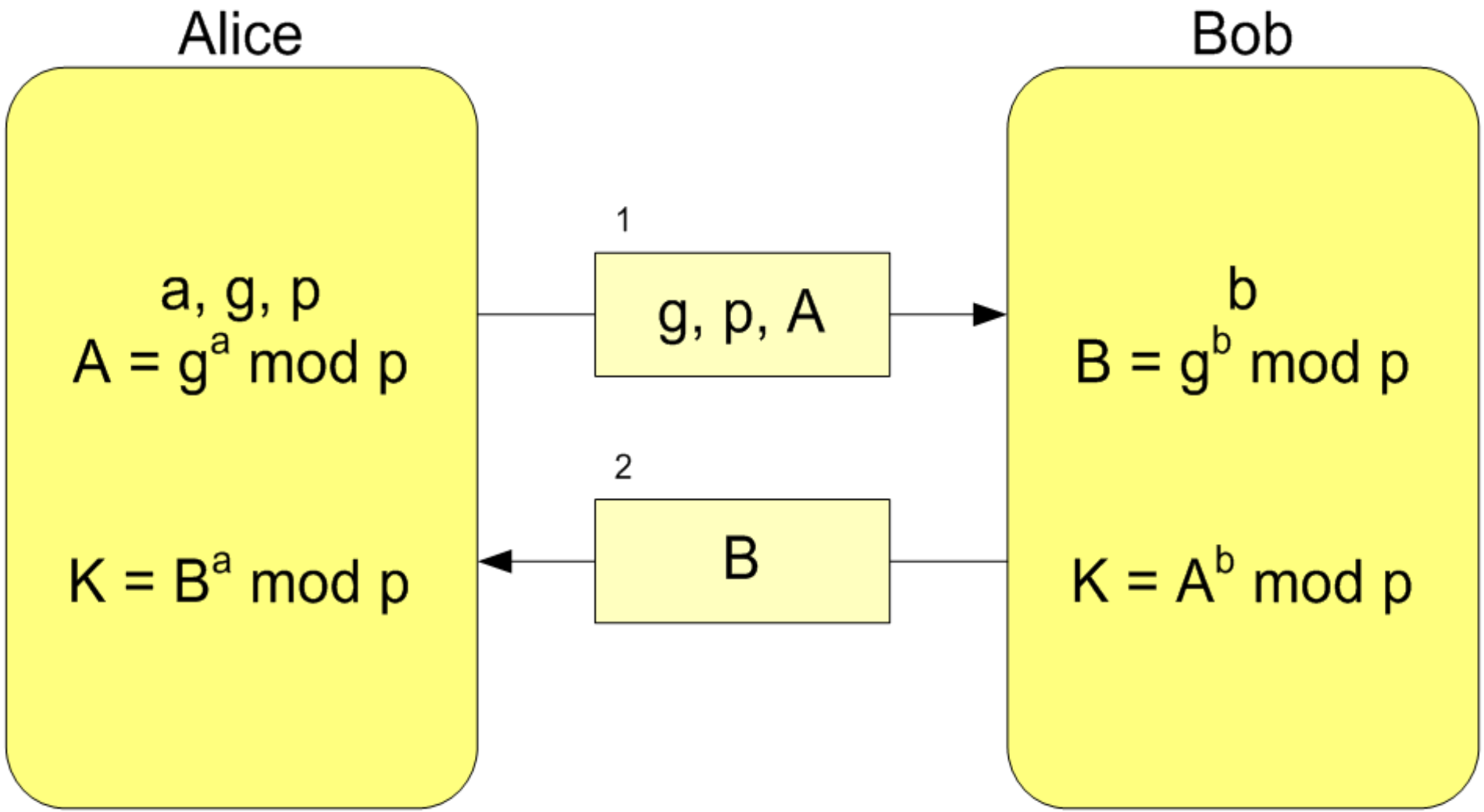
Se conoce como logaritmo discreto de x en base a módulo n a resolver la ecuación $x = a^y \bmod n$ donde x,n y a son constantes e y es la incógnita.

Se denota de la siguiente forma:
$y = \log \text{disc}_a(x)$

Soltando la mano...

A y B acuerdan usar el número primo p=23 y la base g=5.
A elige un número secreto a=6, luego envía a B (g^a mod p)
5^6 mod 23 = 8.
B elige un número secreto b=15, luego envía a A (g^b mod p)
5^{15} mod 23 = 19.
A calcula (g^b mod p)^a mod p
19^6 mod 23 = 2.
B calcula (g^a mod p)^b mod p
8^{15} mod 23 = 2.

Diagrama

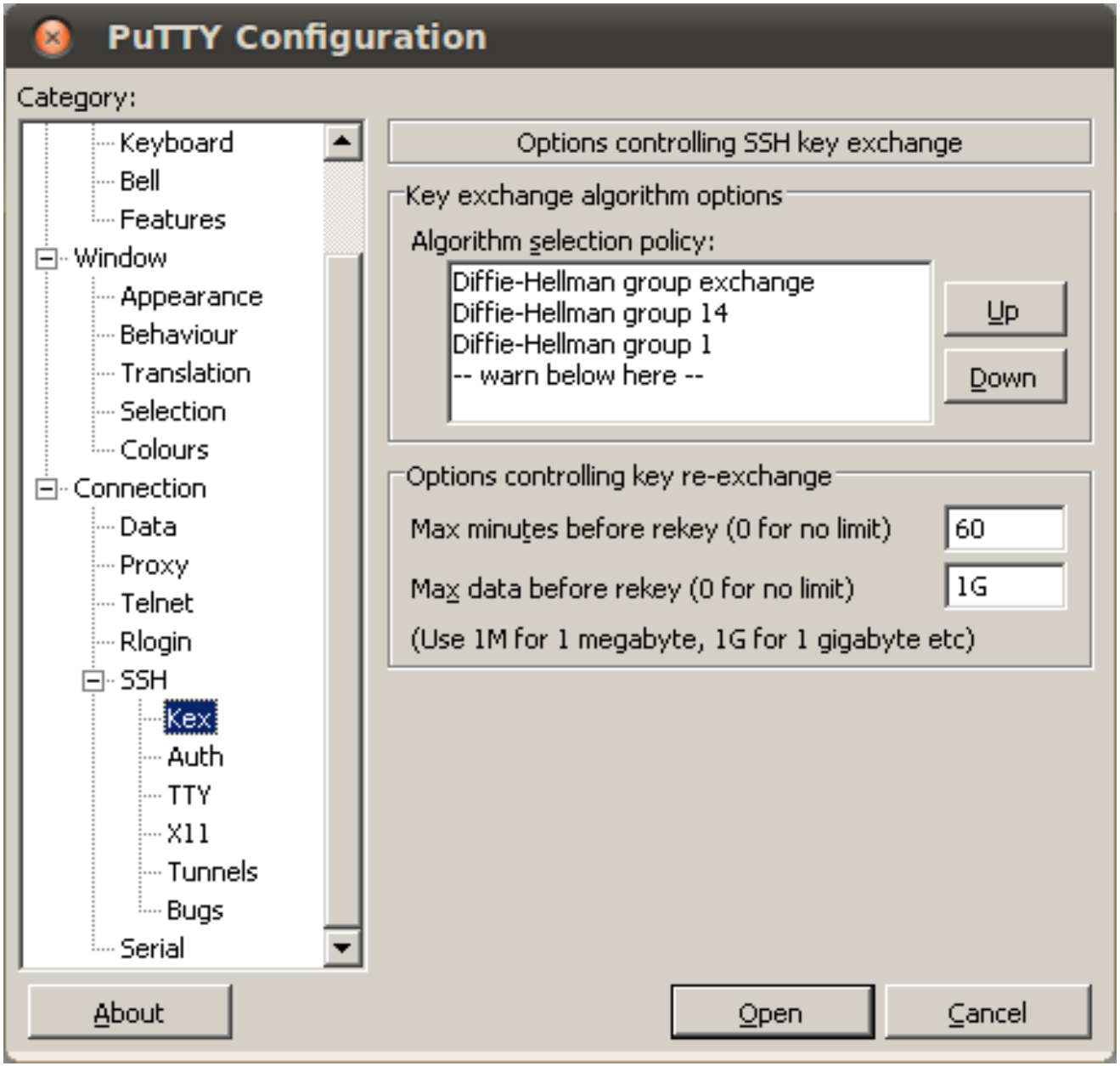


$$K = A^b \bmod p = (g^a \bmod p)^b \bmod p = g^{ab} \bmod p = (g^b \bmod p)^a \bmod p = B^a \bmod p$$

Como mejorar la fortaleza de cifrado?

- Valores mucho más grandes de a,b y p se necesitarían para hacer este ejemplo seguro.
- Dado que es muy sencillo probar todos los valores posibles de g^{ab} mod 23 (habrá, como máximo, 22 valores, inclusive si a y b son números grandes).
- Si p fuera un primo de más de 300 dígitos, y a y b fueran por lo menos de 100 dígitos, entonces hasta el mejor algoritmo para encontrar a dado g, p, y g^a mod p tomaría demasiado tiempo.
- g no necesita ser grande, y en la práctica es usualmente 2 o 5.

¿Dónde se usa?


PuTTY Configuration
Category:
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Kex
Auth
TTY
X11
Tunnels
Bugs
Serial
Options controlling SSH key exchange
Key exchange algorithm options
Algorithm selection policy:
Diffie-Hellman group exchange
Diffie-Hellman group 14
Diffie-Hellman group 1
-- warn below here --
Up
Down
Options controlling key re-exchange
Max minutes before rekey (0 for no limit)
60
Max data before rekey (0 for no limit)
1G
(Use 1M for 1 megabyte, 1G for 1 gigabyte etc)
About
Open
Cancel

RSA

- Es el primer (1977) y más utilizado algoritmo asimétrico.
- Llave pública y privada se obtienen de números primos grandes (100-200 dígitos y más).
- Quebrar RSA equivale a factorizar el producto de dos de estos números primos.
- Para generar ambas llaves, se escogen 2 números primos grandes aleatorios p y q.
- Se calcula el producto n = pq
- Se escoge la llave de cifrado e tal que:

  e es menor y además es primo relativo de (p–1)(q–1)

Qué son los primos relativos?

Sean a, b pertenecientes al conjunto Z, se dice que son primos relativos (o coprimos) "a" y "b" si no tienen ningún factor en común, es decir, si no tienen otro divisor común más que 1.

¿Cómo obtengo d?

La llave de descifrado se obtiene con la siguiente fórmula:
$ed \bmod (p-1)(q-1) = 1$

La llave privada son los números e y n.
La llave pública son los números d y n.
Los números primos p y q ya no se necesitan y no deben ser revelados.

Cifrando m

- Dividirlo en bloques de tamaño menor que n (medido en bits, es la mayor potencia de 2 que sea menor que n).
- Si p y q son números primos de 100 dígitos, n tendrá un poco menos de 200 dígitos y cada bloque m_i será de tamaño similar (algo menos de 200 dígitos).
- Cada bloque cifrado c_i se calcula con la fórmula

  $c_i = m^e_i \bmod n$

- Para descifrar los bloques cifrados c_i se calcula

  $m_i = c^d_i \bmod n$

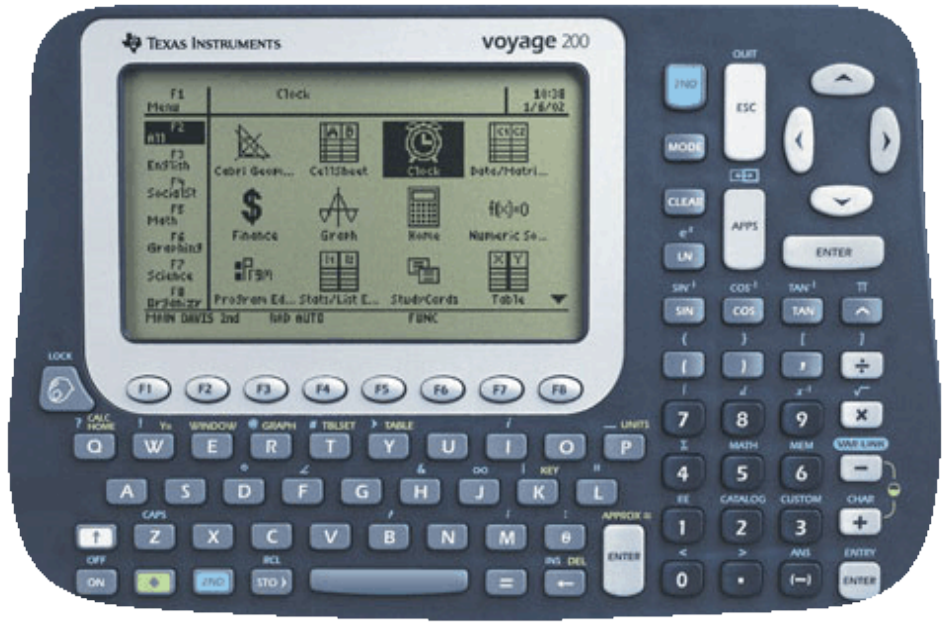
- Los roles de e y d se pueden intercambiar (se puede cifrar con d y descifrar con e)
- Si se cifra algo con e, sólo se podrá descifrar con d y viceversa
- Una de las llaves se mantiene en secreto (llave privada) y la otra se puede dar a conocer a terceros (llave pública).
- Alguien que conozca la llave pública (d y n) no puede deducir el valor de e, ya que:
  - Necesita encontrar los factores primos de n (p y q).
  - Sabemos que factorizar números es un problema exponencial con el tamaño del número.

Soltando la mano...

p = 3, q = 11, n = p*q = 33
(p-1)(q-1) = 2 * 10 = 20

Escoger e que sea < 20 y primo relativo con 20:
e = 13
d = 17 ya que 13*17 = 221
221 mod 20 = 1
cifrar m = 10010011011000110
Mayor potencia de 2 menor que 33 es 2^5.
m = 10010 | 01101 | 10001 | 00010
$m_1 = 18$, $m_2 = 13$, $m_3 = 17$, $m_4 = 2$

Cifrado



Se calcula:

$c_1 = 18^{13} \bmod 33 = 24$

$c_2 = 13^{13} \bmod 33 = 19$

$c_3 = 17^{13} \bmod 33 = 29$

$c_4 = 2^{13} \bmod 33 = 8$

Texto cifrado = 2419298

Descifrado

Se calcula:

$m_1 = 24^{17} \bmod 33 = 18$

$m_2 = 19^{17} \bmod 33 = 13$

$m_3 = 29^{17} \bmod 33 = 17$

$m_4 = 8^{17} \bmod 33 = 2$

Texto en claro = 1813172

RSA vs DES

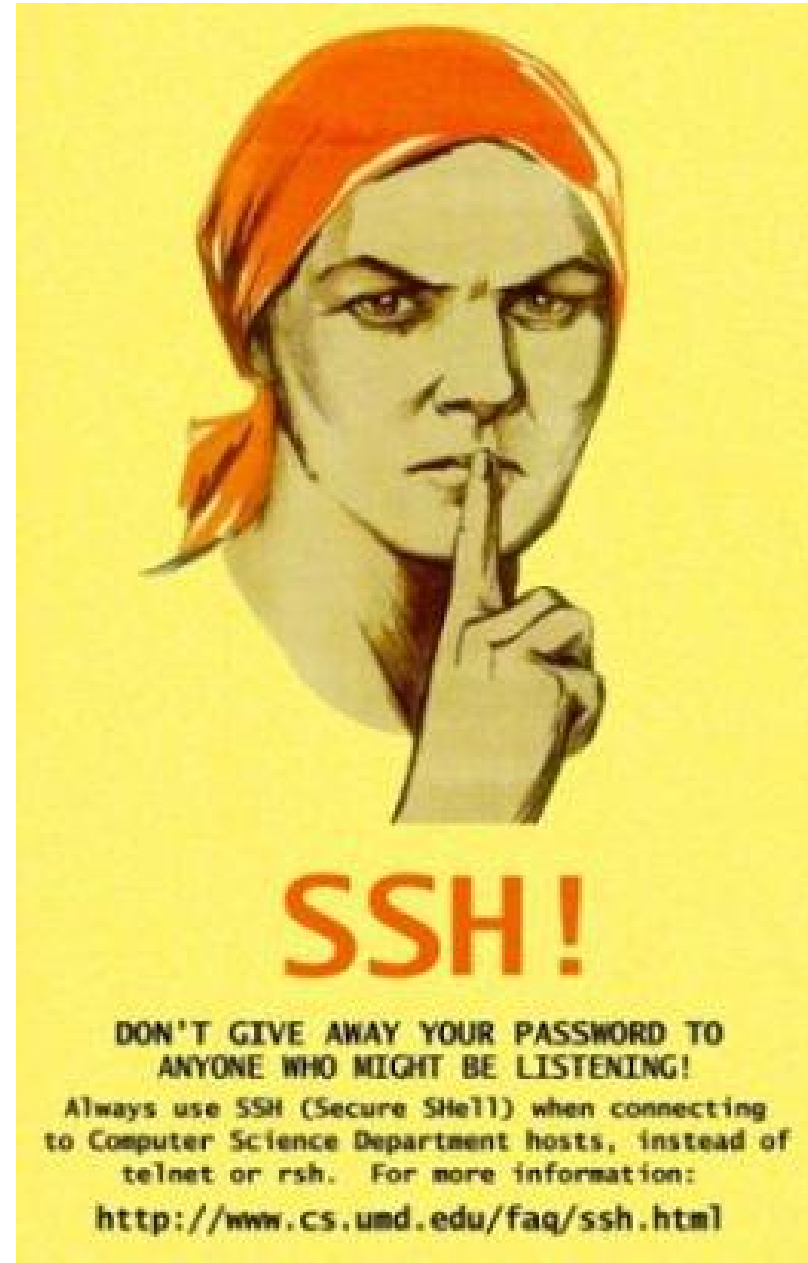
- Las llaves RSA (512, 1024, 2048 bits) son mucho más grandes que las llaves DES.
- Una llave RSA de 1024 bits equivale más o menos a una llave 3DES (112 bits) en cuanto a la dificultad en adivinar la llave.
- RSA en software es 100 veces más lento que DES.
- Nunca podrá ser tan rápido como DES por los algoritmos involucrados y el tamaño de las llaves.
- RSA no se puede usar directamente para cifrar tráfico en una red de alta velocidad.

SSH

origen# ssh-keygen -t rsa

destino# mkdir ~/.ssh

origen# cat .ssh/id_rsa.pub | ssh usuario@servidordestino 'cat >> .ssh/authorized_keys'



RSA vs DSA (algoritmo firma digital)

DSA es más rápido en crear y firmar.
RSA es más rápido en verificar.

Algorithm	Key Generation * 1(ms.)	Sign * 100 (ms.)	Verify*100(ms.)
RSA 512	544.61	915	160
RSA 1024	1120.46	4188	263
DSA 512	6.62	634	988
DSA 1024	17.87	1775	3397

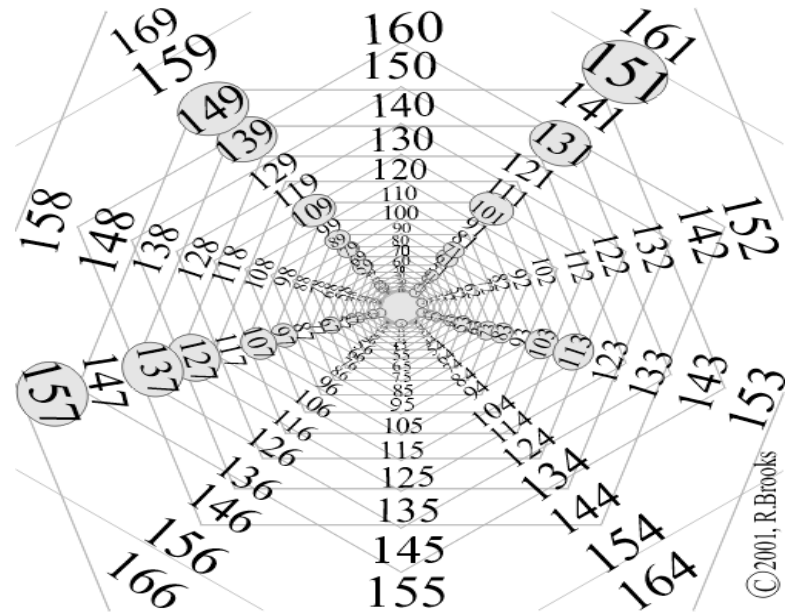
¿Dónde se usa todo esto?

For Windows on Intel x86

PuTTY:	putty.exe	(or by FTP)	(RSA sig)	(DSA sig)
PuTTYtel:	puttytel.exe	(or by FTP)	(RSA sig)	(DSA sig)
PSCP:	pscp.exe	(or by FTP)	(RSA sig)	(DSA sig)
PSFTP:	psftp.exe	(or by FTP)	(RSA sig)	(DSA sig)
Plink:	plink.exe	(or by FTP)	(RSA sig)	(DSA sig)
Pageant:	pageant.exe	(or by FTP)	(RSA sig)	(DSA sig)
PuTTYgen:	puttygen.exe	(or by FTP)	(RSA sig)	(DSA sig)

Primos

- Hasta ahora no existe un algoritmo que permita factorizar un número (encontrar sus factores primos) que no sea exponencial.
- El tiempo que se demora en encontrar los factores primos crece exponencialmente con el tamaño del número.
- Números muy grandes no se pueden factorizar (por el momento).



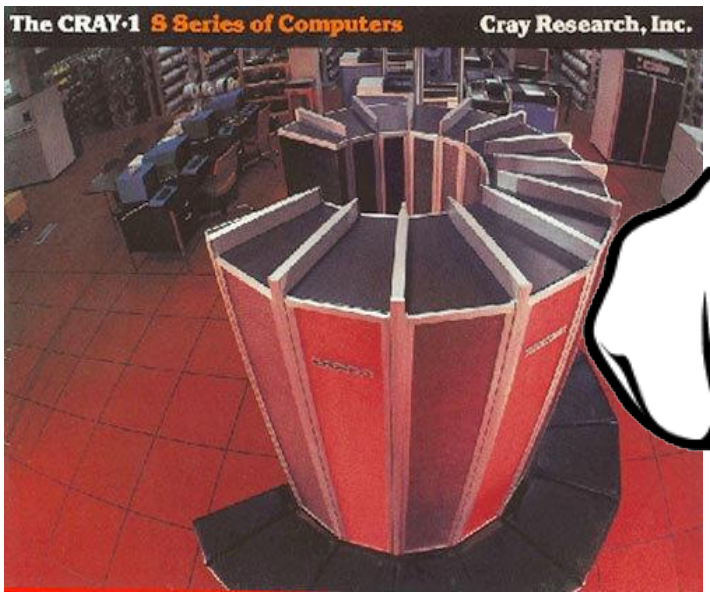
Números Primos Grandes

¿Cómo encontrar uno?

- Existen 10^{151} primos que tienen <= 512 bits.
- La probabilidad que dos personas tengan los mismos factores primos (p y q) es muy cercana a 0.
- Se puede saber con 100% de certeza si un número no es primo, y con un 100% de certeza si es primo cuando es < $3,4 * 10^{14}$ y con probabilidad > 99, 9% si es mayor.
- Se pueden combinar distintas heurísticas para llegar a 99,999...%

RSA Challenge

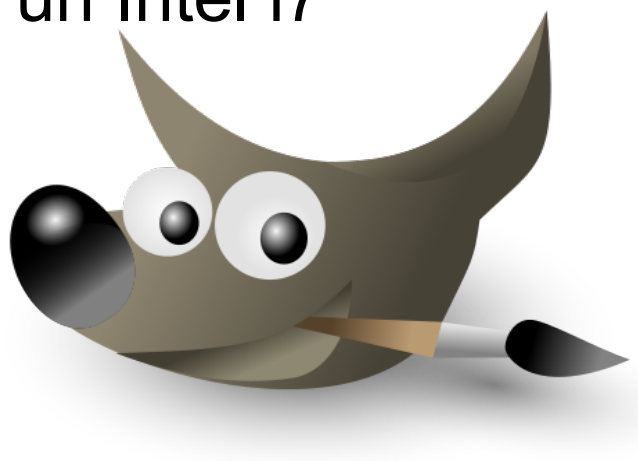
- Concurso organizado entre 1991 y 2007 por RSA para factorizar números grandes.
- Un número de 155 dígitos decimales, fue factorizado el 22/08/99 después de 7 meses.
- Se usaron 300 PC's Pentium y 1 Cray (80 Mhz 64-bits)



http://www.rsa.com/rsalabs/node.asp?id=2092

El número primo más grande conocido

- Descubierto este año por GIMPS (no confundir con el editor gráfico)
- En vez de buscar vida extraterreste como SETI, GIMPS busca números primos de Mersenne en forma distribuida.
- Los primos de Mersenne son los de la forma $2^n - 1$.
- El número es $2^{257,885,161}-1$
- Alrededor de 17.4 millones de dígitos.
- 4.5 días se demoraron en verificarlo con un Intel i7



http://www.mersenne.org/
http://primes.utm.edu