Introducció a la criptografia

Algorismes criptogràfics

Què és un algorisme criptogràfic?

Un algorisme criptogràfic és un algorisme que modifica les dades d'un document amb l'objecte d'aconseguir algunes característiques de seguretat com ara autenticació, integritat i confidencialitat.

L'objectiu d'un algorisme criptogràfic és fer tan difícil com sigui possible desencriptar les dades sense utilitzar la clau. Si s'utilitza un algorisme d'encriptació realment bo, llavors no hi ha cap tècnica significativament millor que intentar metòdicament amb cada clau possible.

Cal tenir en compte...

Els algorismes criptogràfics tendeixen a degradar-se amb el temps. A mesura que transcorre el temps, els algorismes d'encriptació es fan més fàcils de trencar a causa de l'avanç en velocitat i potència dels equips de computació.

Tots els algorismes criptogràfics són vulnerables als atacs per força bruta, és a dir, tractar sistemàticament amb cada possible clau d'encriptació.

La força bruta és més fàcil d'aplicar en la mesura que passa el temps.

Curiositats dels atacs per força bruta

Contrasenya: lletres minúscules i números.

- 6 caràcters: 2.25 bilions de combinacions
  - ***Cracking online*** utilitzant una aplicació web amb un rang de 1.000 intents per segon: 3.7 setmanes.
  - ***Cracking offline*** utilitzant servidors o ordinadors potents amb un rang de 100 bilions d'intents per segon: 0.0224 segons.
  - ***Cracking offline***, utilitzant multiprocessament en paral·lel, clúster de sistemes o *grid computing* amb un rang de 100 trilions d'intents per segon:0.0000224 segons.

http://www.itworld.com/security/280486/how-long-would-it-take-crack-my-password

Curiositats dels atacs per força bruta

Contrasenya: afegint símbols

- 6 caràcters: 7.6 trilions de combinacions
  - ***Cracking online*** utilitzant una aplicació web amb un rang de 1.000 intents per segon: 2.4 segles..
  - ***Cracking offline*** utilitzant servidors o ordinadors d'escriptori potents amb un rang de 100 bilions d'intents per segon: 1.26 minuts.
  - ***Cracking offline***, utilitzant multiprocessament en paral·lel, clúster de sistemes o *grid computing* amb un rang de 100 trilions d'intents per segon:0.0756 segons.

http://www.itworld.com/security/280486/how-long-would-it-take-crack-my-password

Classificació

Podem classificar els algorismes criptogràfics en tres grups:

- Hash o de resum.
  - MD5
  - SHA1
- Criptografia simètrica o de clau secreta.
  - DES
  - RC4
- Criptografia asimètrica o de clau pública.
  - RSA

Algorismes de hash

Els hash o funcions de resum són algorismes que aconsegueixen crear a partir d'una entrada (ja sigui un text, una contrasenya o un arxiu, per exemple) una **sortida alfanumèrica de longitud normalment fixa que representa un resum de tota la informació** que se li ha donat (és a dir, a partir de les dades de l'entrada crea una cadena que solament pot tornar-se a crear amb aquestes mateixes dades).

Característiques dels algorismes hash

Les funcions hash s'encarreguen de representar de forma compacta un arxiu o conjunt de dades que normalment és de major grandària que el hash independentment del propòsit del seu ús.

Aquest sistema de criptografia utilitza algorismes que asseguren que amb la resposta (o hash) mai es podrà saber quines han estat les dades inserides, fet que indica que és una **funció unidireccional**.

Sabent que es pot generar qualsevol resum a partir de qualsevol dada ens podem preguntar si es podrien repetir aquests resums (hash) i la resposta és que teòricament si, podria haver-hi col·lisions, ja que no és fàcil tenir una funció hash perfecta (que aconsegueixi que no es repeteixi la resposta), però això no suposa un problema, ja que si s'aconseguissin (un bon algorisme) dos hash iguals els continguts serien totalment diferents.

Exemples.

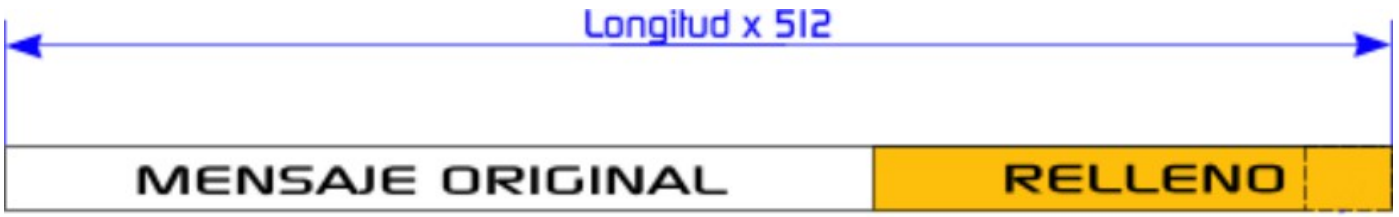
- **Protegir una contrasenya**, ja que podria estar en text pla i ser accessible per qualsevol i encara així no poder ser capaços de deduir-la. En aquest cas, per saber si una contrasenya que està guardada, per exemple, en una base de dades és igual a la qual hem introduït no es desxifra el hash (ja que deuria ser impossible fer-ho) sinó que s'aplicarà la mateixa funció de resum a la contrasenya que especifiquem i es comparará el resultat amb el qual tenim guardat (com es fa amb les contrasenyes dels sistemes basats en Linux).
- **Garantir la integritat de les dades** (cosa que haureu vist moltes vegades), per exemple en algunes webs que proporcionen descàrregues d'arxius grans, per exemple programari, donant al costat de la seva vegada el resum de l'arxiu i la funció usada (MD5 checksum).

Algorisme MD5

- MD5 (*Message-Digest Algorithm 5*, Algorisme de Resum del Missatge 5) és un Algorisme de reducció criptogràfic de 128 Bits àmpliament usat. El codi MD5 va ser dissenyat per Ronald Rivest en 1991.
- MD5 processa missatges de qualsevol mida (longitud variable) fins a concloure amb el missatge final, lliurant a la sortida un bloc "resumit" de 128 bits (longitud fixa).
- El processament consta de cinc passos:
  - Afegir bits.
  - Longitud del missatge.
  - Inicialitzar el buffer MD5.
  - Processar el missatge en blocs.
  - Sortida.

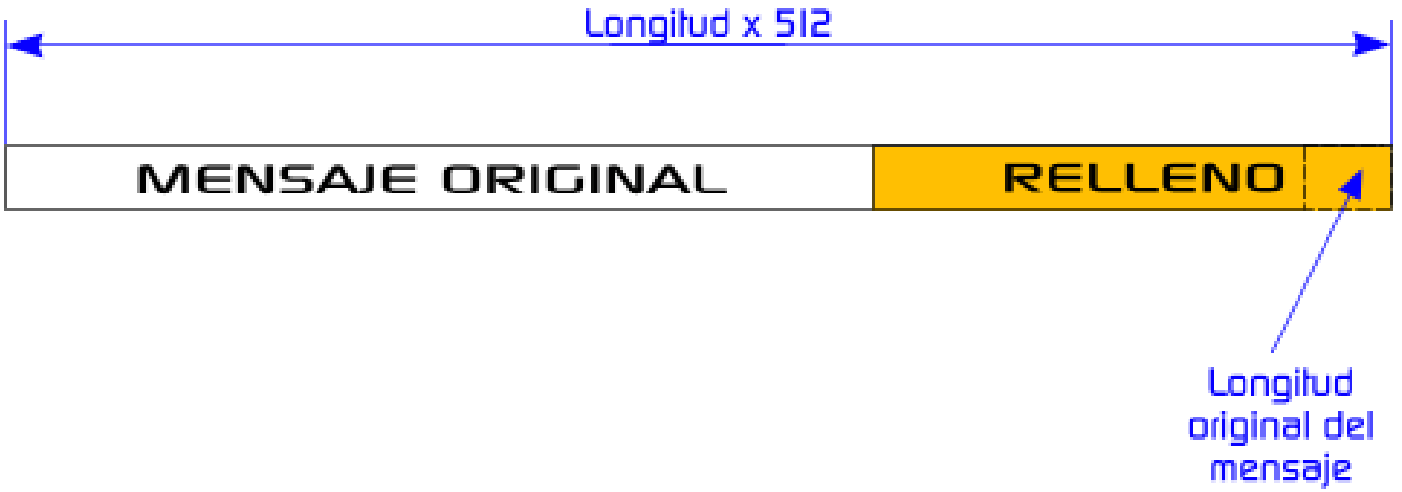
Pas 1. Afegir bits.

Cal reomplir o emplenar el missatge a una longitud congruent en mòdul **`448 mod 512`**. És a dir la longitud del missatge és 64 bits menys que un sencer múltiple de 512. El farciment consisteix en un bit en 1 seguit per tant bits en 0 siguin necessaris.



Pas 2. Afegir la longitud del missatge.

La longitud original del missatge és emmagatzemada en els últims 64 bits del farciment.



Pas 3. Inicialització del buffer

És fins a aquest pas que dóna inici el procés de reducció, per a això s'utilitza un registre de 128 bits que permet emmagatzemar i mantenir els resultats intermedis i final de la funció hash. El registre es divideix en quatre seccions de 32 bits (paraula) cadascuna, les quals corresponen a les variables A, B, C, i D que són inicialitzades amb els valors hexadecimals:

- **`Paraula A: 01 23 45 67`**
- **`Paraula B: 89 ab cd ef`**
- **`Paraula C: fe dc ba 98`**
- **`Paraula D: 76 54 32 10`**

Pas 4. Processar el missatge en blocs

El missatge es processa en blocs de 512 bits alhora a través de 16 blocs de 32 bits cadascun, per a això les quatre variables de concatenació es copien en variables diferents:

a = A

b = B

c = C

d = D

Pas 4. Processar el missatge en blocs

La part medul·lar de l'algorisme és una funció de compressió que consta de quatre rondes, les quals tenen una estructura similar; però cadascuna utilitza operacions diferents durant 16 iteracions; cada operació realitza una funció no lineal sobre tres de les variables a, b, c i d, i el resultat és sumat a la quarta variable que no va ser triada, un subbloque del text i una constant.

A aquest resultat se li aplica una rotació circular a l'esquerra un nombre variable de bits i se suma el resultat a una de les variables a, b, c o d. Finalment el resultat reemplaça a una de les variables a, b, c o d. La sortida de la quarta ronda se sumeixi a l'entrada de la primera en una operació modular 2^{32}

Pas 4. Processar el missatge en blocs

Primer definim quatre funcions auxiliars que prenen com a entrada tres paraules de 32 bits i la seva sortida és una paraula de 32 bits.

RONDA	FUNCIÓN OPERACIÓN
1	F(b, c, d) = (**b** AND **c**) OR (NOT **b** AND **d**)
2	G(b, c, d) = (**b** AND **d**) OR (**c** AND NOT **d**)
3	H(b, c, d) = **b** XOR **c** XOR **d**
4	I(b, c, d) = **c** XOR (**b** OR NOT **d**)

Pas 5. Sortida

El resum del missatge és la sortida produïda per A, B, C i D. Això és, es comença el byte de menor pes de **A** s'acaba amb el byte de major pes de **D**.

Algorisme SHA1

D'acord amb la Viquipèdia:

La família SHA (Secure Hash Algorithm, Algorisme de Hash Segur) és un sistema de funcions hash criptogràfiques relacionades de l'Agència de Seguretat Nacional dels Estats Units i publicades pel National Institute of Standards and Technology (NIST). El primer membre de la família va ser publicat en 1993 és oficialment anomenat SHA.

Cóm funciona?

Per a un missatge d'una longitud màxima de 2^{64} bits com a entrada, SHA-1 produeix com a sortida una cadena de 160 bits anomenada "Missatge Resum " (*message digest*). El missatge resum pot ser introduït a un algorisme de signatura digital el qual genera o verifica la signatura del missatge. Signar el missatge resum en lloc del missatge original proveeix a més, eficiència en el procés, a causa que el missatge resum és, usualment, molt menor en grandària que l'original. El mateix algorisme Hash amb el qual se signa el missatge, ha de ser utilitzat pel receptor per verificar la signatura digital.

El SHA-1 és segur a causa que, no és factible computacionalment, trobar un missatge que correspongui a un missatge resum donat, o trobar dos diferents missatges que produeixin el mateix missatge resum.

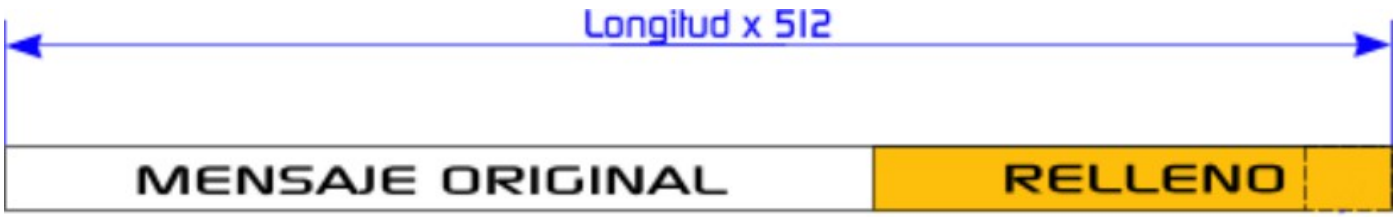
Cóm funciona?

El processament consta de cinc passos:

- Afegir bits.
- Longitud del missatge.
- Inicialitzar el buffer MD.
- Processar el missatge en blocs.
- Sortida.

Pas 1. Afegir bits

S'incorporen bits de farciment al missatge d'entrada de tal manera que compleixi: **`longitud = 448 mod 512`**. El farciment consisteix en un 1 de seguit dels zeros que siguin necessaris. Encara que el missatge ja tingui la longitud desitjada, s'ha d'efectuar el farciment, per la qual cosa el nombre de bits de dita farcida està en el rang d'1 a 512 bits.



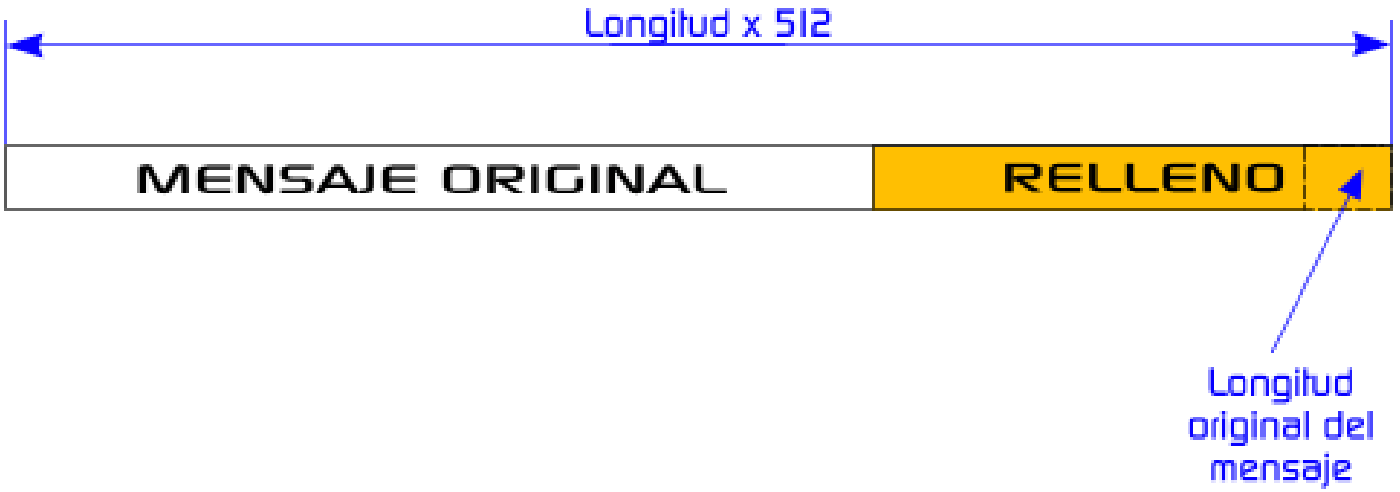
Pas 3. Inicialització del buffer

S'inicialitza la memòria temporal MD, la qual consta de 160 bits i la seva finalitat és emmagatzemar els resultats intermedis i finals de la funció de dispersió. La MD consta de 5 registres (A,B,C,D,I) de 32 bits cadascun, els valors amb els quals s'inicialitzen són els següents (valors hexadecimals):

- **A = 67 45 23 01**
- **B = EF CD AB 89**
- **C = 98 BA DC FE**
- **D = 10 32 54 76**
- **I = C3 D2 I1 F0**

Pas 2. Afegir la longitud del missatge

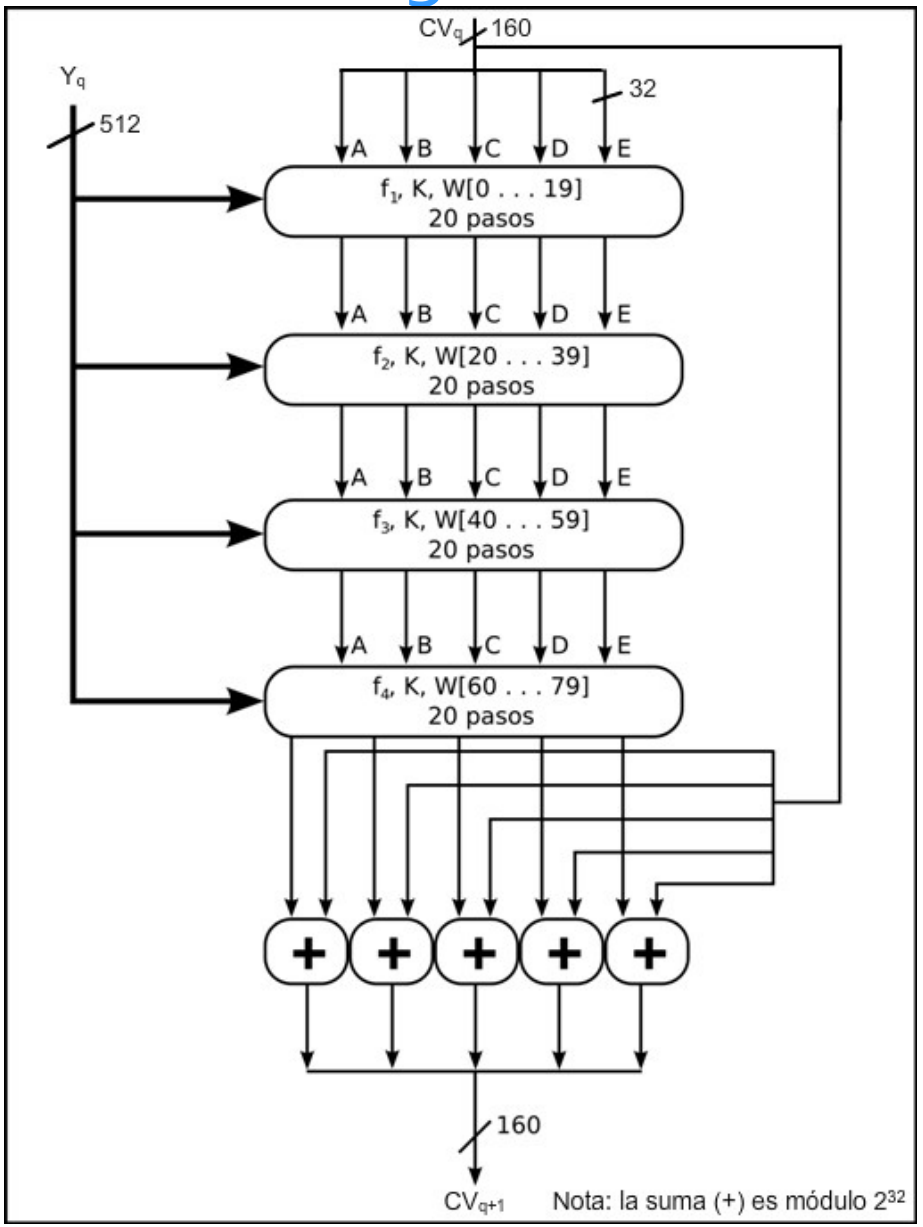
A la sortida del pas 1, se li afegeix un bloc de 64 bits que representi la longitud del missatge original abans de ser emplenat.



Pas 4. Processar el missatge en blocs

Es processa el missatge en blocs de 512 bits, cadascun d'ells passa per un mòdul que consta de 4 rondes de processament de 20 passos cadascuna. Les rondes tenen una estructura similar, amb l'excepció que cadascuna ocupa una funció lògica primitiva diferent (f_1, f_2, f_3 i f_4).

Pas 4. Processar el missatge en blocs



Pas 4. Processar el missatge en blocs

L'entrada a cada ronda consta del bloc de 512 bits que s'estigui processant ($\mathbf{Y}_q$) i els 160 bits de la memòria MD; cada bloc de 512 bits actualitzarà el valor de la memòria temporal. Cada ronda també fa ús de la constant additiva $\mathbf{K}_t$, on **0<=t<=79** indica un dels 80 passos al llarg de les quatre rondes.

Ronda	Número de paso	K_t (hexadecimal)
1	$0 \leq t \leq 19$	5A827999
2	$20 \leq t \leq 39$	6ED9EBA1
3	$40 \leq t \leq 59$	8F1BBCDC
4	$60 \leq t \leq 79$	CA62C1D6

Pas 5. Sortida

Una vegada que es processen els L blocs de 512 bits, el resum del missatge són els 160 bits de sortida de l'últim bloc.

Algorismes de clau simètrica

- Els algorismes simètrics o clau secreta encripten i desencripten amb la mateixa clau.
- Les dues parts que es comuniquen han de posar-se d'acord per endavant sobre la clau a utilitzar.
- Una vegada que ambdues parts tenen accés a aquesta clau, el remitent xifra un missatge usant la clau, ho envia al destinatari, i aquest ho desxifra amb la mateixa clau.
- Els principals avantatges dels algorismes simètrics són la seva seguretat i la seva velocitat.

Inconvenients dels algorismes de clau simètrica

- El principal problema amb els sistemes de xifrat simètric no està lligat a la seva seguretat, sinó a l'intercanvi i distribució de claus. Una vegada que el remitent i el destinatari hagin intercanviat les claus poden usar-les per comunicar-se amb seguretat, però quin canal de comunicació que sigui segur han usat per transmetre's les claus?
- Seria molt més fàcil per a un atacant intentar interceptar una clau que provar les possibles combinacions de l'espai de claus.
- Un altre problema és el nombre de claus que es necessiten.
- Per solucionar aquests problemes es podrien tenir centres de distribució de claus simètriques. Això podria funcionar per exemple per a organitzacions militar. Encara que sempre hi hauria un risc a possibles filtracions d'informació.

Un poc d'història

- Xifrat Cèsar
  - Algorisme de xifrat per substitució més antic i simple.
  - Consisteix a reemplaçar cada lletra de l'alfabet amb la lletra que està tres posicions més endavant en l'alfabet.
  - Desxifrar-lo utilitzant la força bruta és molt senzill. Solament calen 26 intents, un per cada lletra de l'alfabet.

```
text clar   : abcde ...
Clau        : 3 (desplaçament)
text xifrat: defgh...
```

Un poc d'història

- Xifrat de Vigenere
  - Ideat per Blaise de Vigenere al segle XVI.
  - Consisteix en 26 xifrats Cèsar (del 0 al 25).
  - Cada xifrat es denota per una lletra clau, que és la lletra xifrada que substitueix la lletra del text clar 'a'.
  - Per xifrar un missatge, es necessita una clau tan llarga com el missatge.
    - Normalment aquesta clau és una paraula clau repetida tantes vegades com sigui necessari.
  - Cada lletra de la clau indica quin dels xifrats monoalfabètics s'usarà; És a dir, cada lletra de la clau indica la substitució per a la lletra A,

Un poc d'història

Xifrat de Vigenere

ABCDEFGHIJKLMNOPQRSTUVWXYZ
BCDEFGHIJKLMNOPQRSTUVWXYZA
CDEFGHIJKLMNOPQRSTUVWXYZAB
DEFGHIJKLMNOPQRSTUVWXYZABC
EFGHIJKLMNOPQRSTUVWXYZABCD
.....
.....
XYZABCDEFGHIJKLMNOPQRSTUVW
YZABCDEFGHIJKLMNOPQRSTUVWX
ZABCDEFGHIJKLMNOPQRSTUVWXY

```
text clar   : wearediscoveredsaveyourself
clau        : deceptivedeceptivedeceptive
text xifrat: ZICVTWQNGRZGVTWAVZHCQYGLMGJ
```

Algoritme DES

- DES (*Data Encryption Standard*, estàndard de xifrat de dades) és un algorisme desenvolupat originalment per IBM a requeriment del NBS (National Bureau of Standards, *Oficina Nacional d'Estandardització*, en l'actualitat NIST, *National Institute of Standards and Technology*, Institut Nacional d'Estandardització i Tecnologia) d'EUA i posteriorment modificat i adoptat pel govern d'EUA en 1977 com a estàndard de xifrat de totes les informacions sensibles no classificades.

Un poc d'història

- En 1974, IBM presenta Llucifer. Després de l'estudi de l'algorisme, la NSA proposa una sèrie de canvis que són acceptats.
- El 17 de Març de 1975, el NBS publica els detalls del DES. Aquest algorisme va ser adoptat com a estàndard per a les comunicacions segures i l'emmagatzematge d'informació no classificada pel Govern dels EUA.
- En 1981, diversos organismes privats ho adopten com a estàndard.
- En 1983, el DES es ratifica com a estàndard sense problemes.
- En 1997, el DES és trencat com a conseqüència d'un repte que va llançar la RSA, es van examinar el 25% de les claus.

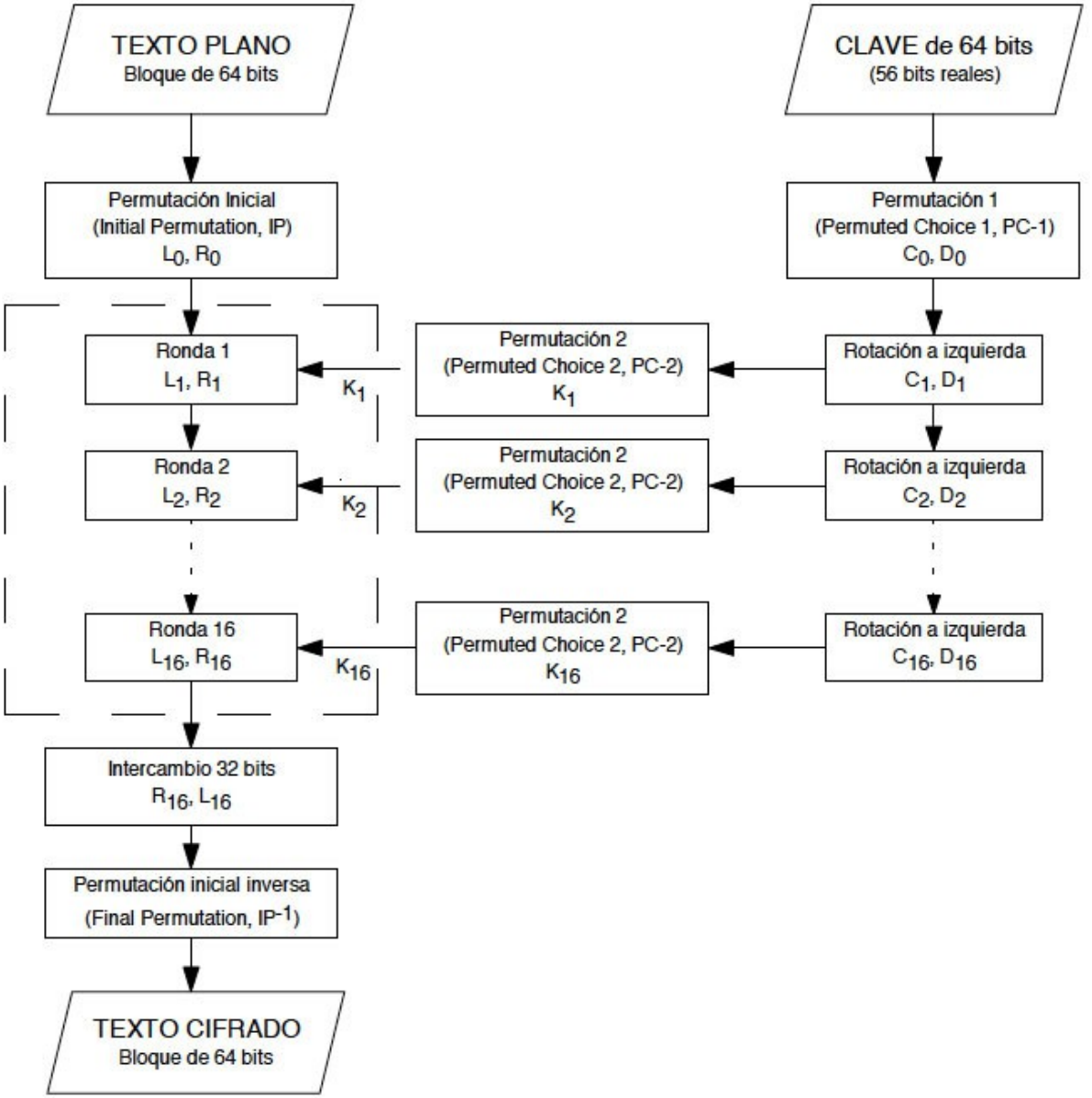
Un poc d'història

- Seguidament, en 1998 a causa de la curta longitud de la clau (56 bits) utilitzada per xifrar el DES, es va demostrar que per força bruta era possible trencar-lo. En 39 dies es va fer possible examinant el 85% de les claus. Poc més tard, la EFP (*Electronic Frontier Foundation*) va presentar la seva *DES cracker*, capaç de trencar el DES per força bruta en 4.9 dies.
- Finalment, en 1999 la EFP trenca el DES en menys de 23 hores.
- Avui dia existeixen sistemes capaços trencar el DES en 5 hores.
- Malgrat això, encara no s'ha trobat cap feblesa a nivell teòric pel que encara se segueix utilitzant, però amb variacions.

Funcionament

- Inicialment, IBM, el va denominar, Llucifer. Treballava sobre blocs de 128 bits, tenint la clau igual longitud. Es basava en operacions lògiques booleanes i podia ser implementat fàcilment, tant en programari com en maquinari.
- Després de les modificacions introduïdes pel NBS, consistents bàsicament en la reducció de la longitud de clau i dels blocs, DES xifra blocs de 64 bits, mitjançant permutació i substitució i usant una clau de 64 bits, dels quals 8 són de paritat (això és, en realitat utilitza claus de 56 bits), produint així 64 bits xifrats.

Esquema general de l'algorisme



Funcionament

- El text original de 64 bits i la clau inicial passen a través d'una permutació.
- Es realitzen 16 iteracions de la mateixa funció, la sortida de la iteració 16 conté 64 bits els quals són funció del text en clar i la clau, les meitats esquerra i dreta d'aquesta sortida són intercanviades per produir la *presortida*.
- La *presortida* passa a través d'una permutació per a produir el text xifrat de 64 bits, aquesta permutació és la inversa de la funció de permutació inicial.

Funcionament

- El procediment que es realitza en cadascuna de les iteracions:
  - Els 64 bits corresponents al missatge que entren en cadascuna de les iteracions són tractats com dos grups de 32 bits, els primers 32 i els últims 32, marcats com a L (part esquerra) i R (part dreta).
  - Els bits corresponents a la clau que entren a cadascuna de les iteracions es tracten com dos grups de 28 bits cadascun, marcats com a C i D.
  - En cada iteració C i D passen per separat per un desplaçament circular a l'esquerra d'1 o 2 bits.
  - Els bits resultants dels desplaçaments circulars serveixen com a entrada de la següent iteració i com a entrada a una funció de permutació la qual produeix una subclau Ki de 48 bits.

Variacions

- Va ser emès pel NIST al 1999 com una versió millorada de DES. Realitza tres vegades el xifrat DES utilitzant tres claus.
- Quan es va descobrir que una clau de 56 bits (utilitzada en el DES) no era suficient per evitar un atac de força bruta, el **3DES** va ser triat per engrandir la clau sense la necessitat de canviar l'algorisme de xifrat.
- Actualment el 3DES segueix sent utilitzat però cada vegada més està sent substituït per l'algorisme AES que ha demostrat ser molt robust i més ràpid.

Algorisme RC4

- Dins de la criptografia RC4 o ARC4 és el sistema de xifrat de flux (*Stream cipher*) més utilitzat i s'utilitza en alguns dels protocols més populars com ara *Transport Layer Security* (TLS/SSL) (per protegir el tràfic d'Internet) i *Wired Equivalent Privacy* (WEP) (per afegir seguretat a les xarxes sense fils).
- RC4 va ser exclòs de seguida dels estàndards d'alta seguretat pels criptògrafs i algunes maneres d'utilitzar aquest algorisme l'han portat a ser un sistema de criptografia molt insegur, incloent el seu ús WEP.
- No està recomanat el seu ús en els nous sistemes, no obstant això, alguns sistemes basats en RC4 són prou segurs per a un ús comú.
- El principal avantatge és que es tracta d'un algorisme de xifrat de flux, per tant l'encriptació la realitza byte per byte, és a dir, no ha d'esperar al proper bloc complet de dades. És molt útil si es te un ample de banda limitat.

Un poc d'història

- L'algorisme RC4 va ser dissenyat per Ron Rivest de la RSA l'any 1987.
- En un principi l'algorisme era un secret registrat, però al setembre de 1994 una descripció de l'algorisme va ser publicada anònimament en una llista de correu de Cypherpunks. De seguida va passar al grup de correu sci.crypt i d'aquí va ser publicat en nombrosos llocs d'Internet. A causa del coneixement de l'algorisme, aquest va deixar de ser un secret.
- No obstant això RC4 encara és una marca registrada i no es pot utilitzar comercialment sense pagar *royalties*.
- Actualment la implementació no oficial de RC4 és legal però no pot ser utilitzada amb el nom de RC4. Per aquest motiu, i amb la finalitat d'evitar problemes legals arran de la marca registrada, sovint podem veure-ho nomenat com ARCFOUR, ARC4 o Alleged-RC4.

Funcionament

- Algorisme:
  1. A partir de la clau secreta es tria una permutació del grup simètric S256 (un array) , és a dir, una forma particular d'ordenar els nombres del 0 al 255.
  2. Es pren el primer byte del missatge i es realitza el xifrat efectuant una XOR amb una part de la permutació, aquest procés se segueix per a cada byte del missatge.
- A la primera part s'anomena extensió de la clau, la segona descriu el procés de xifrat. Cal fer notar que el procés de desxifrat és igual al xifrat.

Funcionament

- Extensió de la clau (KSA, *Key Scheduling Algorithm*):

1. S'inicialitzen les variables:

```
for(i = 0; i < 256; i++)
    state[i] = i;

x = 0;
y = 0;
index1 = 0;
index2 = 0;
```

2. A partir de la clau privada, es reordena l'array:

```
for(i= 0; i < 256; i++)
{
    index2 = (key[index1] + state[i] + index2) mod 256;
    swapbyte(state[i], state[index2]);
    index1 = (index1 + 1) mod keylen;
}
```

Funcionament

- Recorrent cada element del array `state`, primer es recalcula la variable index2.
- Posteriorment es realitza una transposició (*swap*), és a dir, s'intercanvia l'element en `state[i]` per `state[index2]`.
- Finalment es torna a calcular la variable `index1`.
- L'anterior dóna un nou ordre als elements del array `state`, que no és gens més que una permutació calculada a partir de la clau privada, és a dir es tria un element del grup simètric S256

Funcionament

- Procés de xifrat (PRGA, *Pseudo-Random Generation Algorithm*)

```
for(i = 0; i < menlen; i++)
{
    x = (x + 1) mod 256;
    y = (state[x] + y) mod 256;
    swapbyte(state[x], state[y]);
    xorIndex = (state[x] + state[y]) mod 256;
    men[i] ^= state[xorIndex];
}
```

- Per a cada byte del missatge, és a dir des d'`i=0` fins a `i=longitud` del missatge (`menlen`), es calculen les variables `x`, `y` amb les formules exposades.
- Es realitza una transposició (*swap*) de l'element `state[x]` i `state[i]`. Es calcula la variable `xorIndex`. I finalment es xifra el byte `men[i]` amb la formula `men[i] xor state[xorIndex]`.

Criptografia asimètrica

- Coneguda també com a criptografia de clau pública.
- En la criptografia de clau pública, l'usuari té un parell de claus una **clau pública** i una **clau privada**. La **privada** es manté **secreta**, mentre que la **pública** es pot distribuir a tothom.
- Els missatges s'han de xifrar amb la clau pública i només es poden desxifrar amb la clau privada corresponent.
- Les claus es relacionen matemàticament, però la clau privada a la pràctica no es pot obtenir a partir de la clau pública.

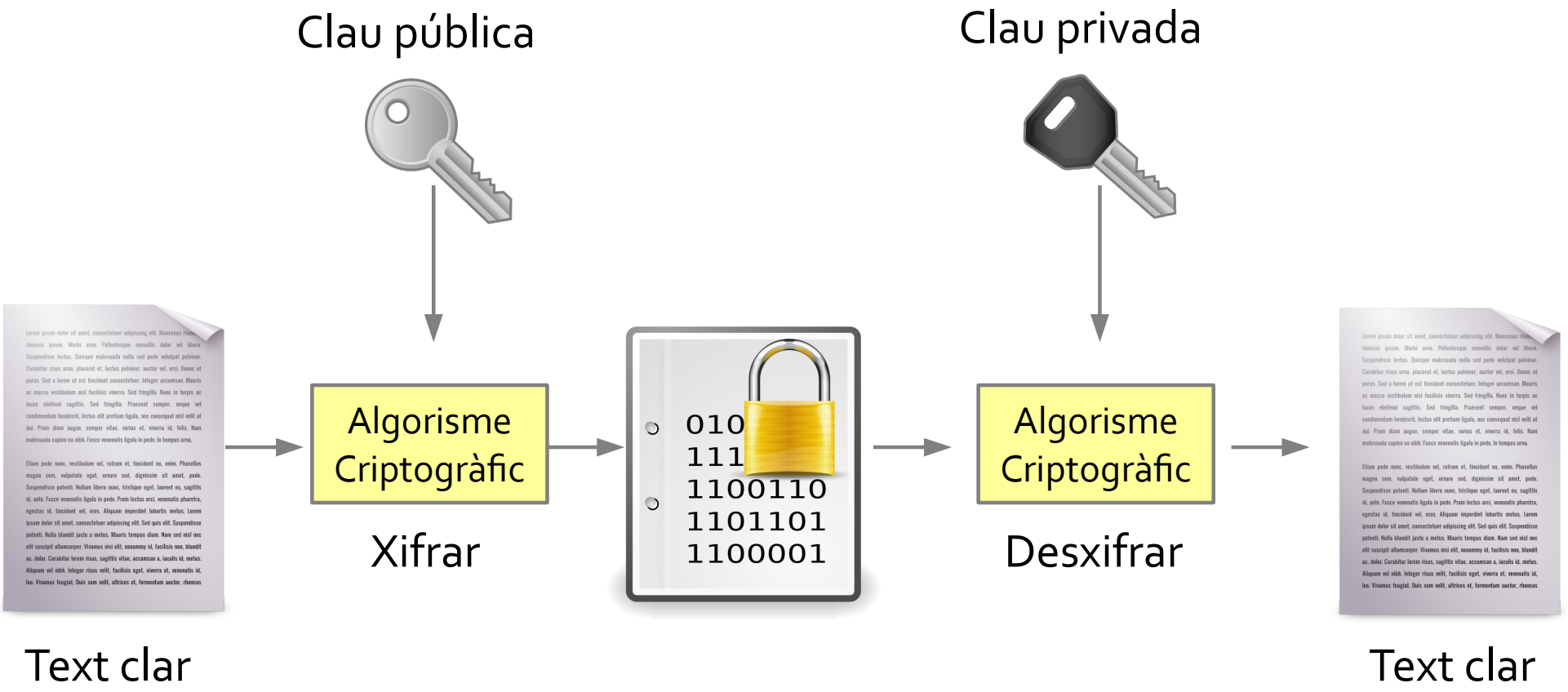
Criptografia asimètrica

- Una **analogia** per al xifratge de clau pública és una bústia tancada amb una ranura per al correu. La ranura de correu exposada i completament accessible al públic és en essència la clau pública. Qualsevol que sap l'adreça de la bustia pot anar i deixar un missatge escrit a través de la ranura; tanmateix, només la persona que posseeix la clau pot obrir la bústia i llegir el missatge.

Principals usos

- **Protecció de la informació:**
  - Suposem que **A** vol enviar un missatge a **B**. Per a això sol·licita a B la seva clau pública **Kp**. **A** genera llavors el missatge xifrat **m**. Una vegada fet això únicament qui posseeixi la clau **Kp,** és a dir B, podrà recuperar el missatge original **m**.

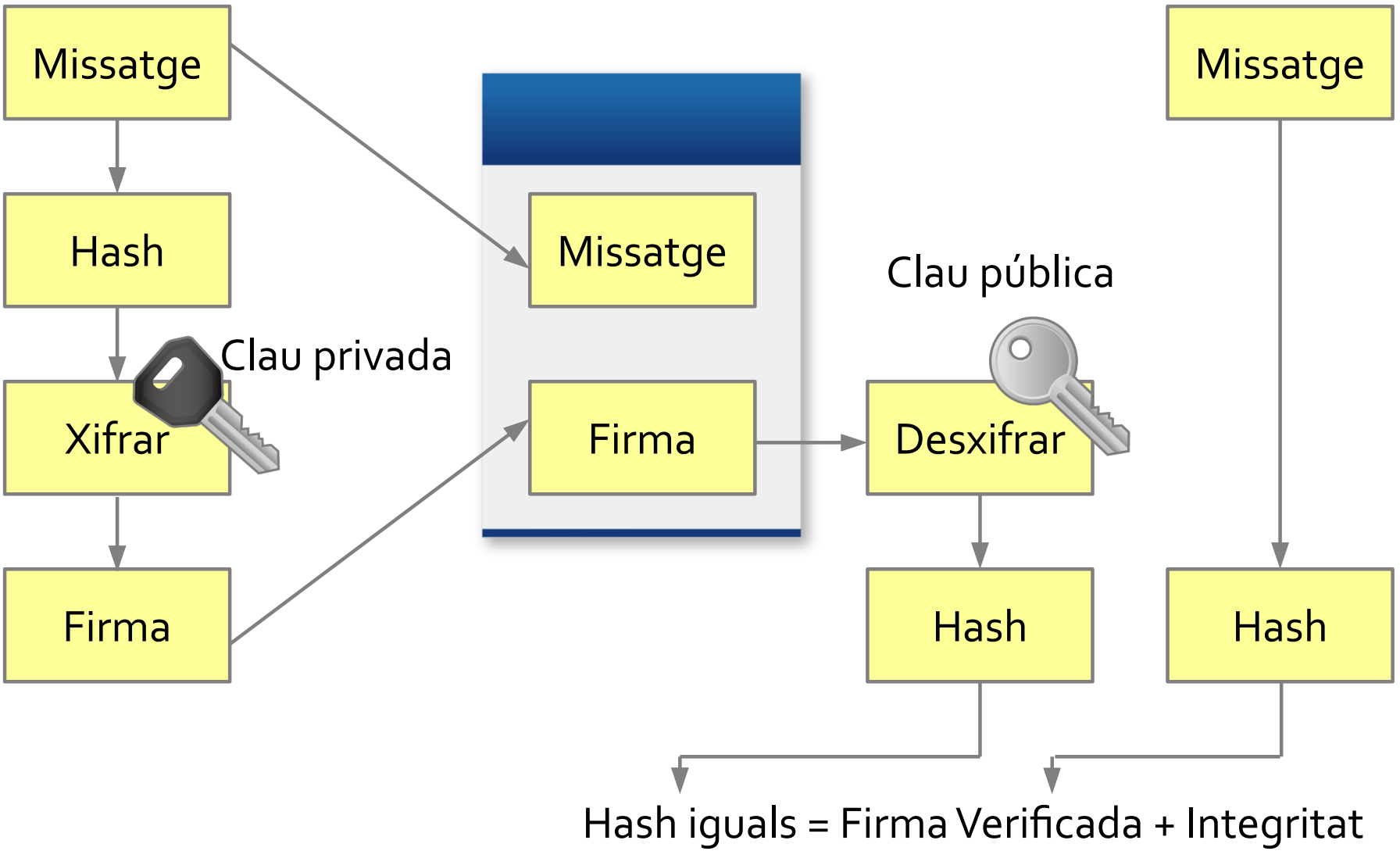
Assegurar la confidencialitat amb claus públiques



Principals usos

- **Autenticació**
  - La segona aplicació dels algorismes asimètrics és l'autenticació de missatges, amb ajuda de funcions resum (hash), que ens permeten obtenir una signatura digital a partir d'un missatge. Aquesta signatura és molt més petita que el missatge original, i és molt difícil trobar un altre missatge que doni lloc a la mateixa.
  - Suposem que **A** rep un missatge **m** de **B** i vol comprovar la seva autenticitat. Per a això **B** genera un resum del missatge **r(m)** i ho codifica emprant la clau de xifrat, que en aquest cas serà privada.
  - La clau de desxifrat s'haurà fet pública prèviament, i ha d'estar en poder de **A**. **B** envia a **A** el criptograma corresponent a **r(m)**. **A** pot ara generar la seva pròpia **r'(m)** i comparar-la amb el valor **r(m)** obtingut del criptograma enviat per **B**. Si coincideixen, el missatge serà autèntic, ja que l'únic que posseeix la clau per codificar és precisament **B**.

Autenticar amb claus públiques



Inconvenients

- Lentitud.
- Ineficient per a grans volums de dades.
- Necessitat d'un tercer implicat (Autoritat de Certificació) dins el procés.
- Necessitat d'una gran infraestructura independentment del nombre d'usuaris.

Algorisme RSA

- Dissenyat en 1977 per Rivest, Shamir i Adleman.
- És molt utilitzat avui en dia.
- Serveix tant per a xifrar missatges com per a autenticar documents (firma digital).
- La seguretat d'aquest algorisme es fonamenta en que no existeix una manera ràpida i senzilla de factoritzar nombres sencers molt grans.

Algorisme RSA

- Un missatge que és a punt de ser xifrat és tractat com un nombre gran. Al encriptar el missatge, aquest s'eleva a la potència de la clau i el que queda és dividit per un producte fix de dos nombres primers. En repetir el procés amb l'altra clau, el text simple o clar pot ser recuperat de nou.
- El millor mètode conegut actualment per trencar el xifrat requereix factoritzar el producte utilitzat en la divisió.
- Actualment, no és possible calcular aquests factors per a nombres majors que 768 bits. No obstant això, els cripto-sistemes moderns utilitzen una longitud mínima de clau de 3072 bits.

http://www.slideshare.net/amadapa/el-algoritmo-rsa

Una descripció més detallada

Vulnerabilitats

- **Claus petites.**
  - Actualment es considera segura una clau RSA amb una longitud d'almenys 768 bits, si bé es recomana l'ús de claus no inferiors a 1024 bits. Fins a fa relativament poc es recomanaven 512 bits. Tenint en compte els avanços de la tecnologia, i suposant que l'algorisme RSA no sigui trencat analíticament, haurem d'escollir la longitud de la clau en funció del temps que vulguem que la nostra informació romangui en secret. Efectivament, una clau de 1024 bits sembla sens dubte massa talla com per protegir informació per més d'uns pocs anys.

Vulnerabilitats

- Atac de l'intermediari.
  - L'atac d'intermediari pot donar-se amb qualsevol algorisme asimètric.
  - Suposem que **A** vol establir una comunicació amb **B**, i que **C** vol espiar-la. Quan **A** li sol·liciti a **B** la seva clau pública **KB**, **C** s'interposa, obtenint la clau de **B** i enviant a **A** una clau falsa **kC** creada per ell. Quan **A** codifiqui el missatge, **C** ho interceptarà de nou, desxifrant-lo amb la seva clau pròpia i emprant **KB** para a recodificar-lo i enviar-ho a **B**. Ni **A** ni **B** són conscients que els seus missatges estan sent interceptats.
  - L'única manera d'evitar això consisteix a assegurar a **A** que la clau pública que té de **B** és autèntica. Per a això gens millor que aquesta estigui signada per un amic comú, que certifiqui l'autenticitat de la clau.

Atac de l'intermediari (*man in the middle*)