

# Criptografía

## Criptografía

- Ciencia que estudia la transformación de un mensaje en código, de forma tal que solo algunas personas puedan obtener el mensaje original a partir de dicho código.
- Sufrió grandes avances en épocas de guerra, especialmente en la Segunda Guerra Mundial. Actualmente tiene un gran auge debido al auge de las comunicaciones digitales (Internet)

## Criptosistemas

- Es un conjunto de tres elementos
  - Un espacio de mensajes: PT. Es la colección de todos los posibles mensajes pt que se pueden enviar.
  - Un espacio de claves: K. Cada clave k determina un método de encriptación  $B_k$  y un método de desencriptado  $D_k$ , tales que  $B_k(pt)=\text{código}$  y  $D_k(\text{código})=pt$
  - Un espacio de códigos: CT. Es la colección de todos los posibles códigos ct.

## Criptoanálisis

- Ciencia que intenta 'romper' los criptosistemas desarrollados por los criptógrafos, para obtener el mensaje a partir del código cifrado.
- Se puede intentar algún tipo de análisis, o simplemente probar todas las claves posibles. A este método se lo denomina **ataque por fuerza bruta**.

## Aplicaciones

- Para proteger la información “crítica” almacenada en computadoras
- Para proteger los mensajes enviados a través de redes (locales, públicas, Internet)
- Para certificar la identidad de quienes envían mensajes a través de redes
- Para evitar que comunicaciones telefónicas, radiales o televisivas puedan ser interceptadas

## Algoritmos de encriptación

- Pueden subdividirse en dos grupos:
  - **Algoritmos de clave privada o simétricos**
  - **Algoritmos de clave pública o antisimétricos**
- Pueden ser:
  - **Incondicionalmente seguros**
  - **Computacionalmente seguros**

## One time pads

- Es el único método perfecto para encriptar
- Consiste en generar una tira de caracteres random. Cada caracter del mensaje se encripta con un caracter de la tira
- Quien descrypta tiene una copia de la tira
- NUNCA se vuelve a usar una tira
- Es imposible de quebrar si la tira es realmente al azar y no se utiliza mas de una vez.

## Algoritmos de clave privada

- Se utiliza un “password” para encriptar el mensaje, sin el cual no puede ser recuperado.
  - MENSAJE + PASSWORD = CODIGO
  - CODIGO + PASSWORD = MENSAJE

## Algoritmos de clave privada

- Sustitución:
  - Monoalfabética
  - Homofónica
  - Poligráfica
  - Polialfabética

## Algoritmos de clave privada

- Pueden clasificarse en:
  - **Monoalfabéticos:** cada ocurrencia de un mismo caracter en el mensaje original se reemplaza siempre por el mismo caracter en el código cifrado.
  - **Polialfabéticos:** cada ocurrencia de un mismo caracter en el mensaje original es reemplazada por distintos caracteres en el código cifrado.

## Criptosistema CAESAR

- Fue el primero que se utilizó. Es monoalfabético y muy malo.
- La encriptación se hace por sustitución. Cada caracter del mensaje original se reemplaza por un caracter en el mensaje cifrado, que se obtiene avanzando 'k' pasos en el alfabeto a partir del caracter original. 'k' es la clave.
- Ejemplo si k=1: 'Hola amigos como les va' => 'lpmb bnjhpt dpnp mft wb'

## Criptosistema DES

- DES: Data Encryption Standard. Usa una clave de 56 bits.
- Triple DES: usa dos claves DES para encriptar tres veces.
- IDEA: sucedió a DES. Usa claves de 128 bits. Está basado en el concepto de 'mezclar operaciones de distintos grupos algebraicos' (?!). Es más rápido que DES.

## Deficiencias de los algoritmos de clave privada

- Presentan una vulnerabilidad evidente: el password.
- Existen distintos métodos para averiguar el password:
  - Shoulder surfing
  - Caballos de troya
  - Ingeniería social

## Algoritmos de clave pública

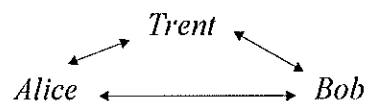
- Cada usuario genera dos claves: una pública y una privada. Debe conservar su clave privada, mientras distribuye su clave pública.
- Los mensajes encriptados con la clave pública de un usuario sólo pueden desencriptarse con la clave privada del mismo.
- El algoritmo de encriptación es público.

## Protocolos criptográficos

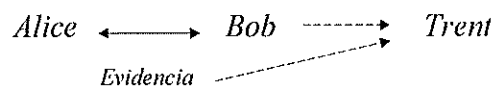
- Un protocolo es una serie de pasos que deben realizar dos o mas partes para llevar a cabo una tarea.
- Todas las partes intervinientes deben conocer el protocolo de antemano.
- Todas las partes tienen que acordar seguir el protocolo
- El protocolo no debe ser ambiguo
- El protocolo debe ser completo.
- *NO DEBE SER POSIBLE HACER MAS O SABER MAS QUE LO QUE ESTA ESPECIFICADO EN EL PROTOCOLO.*

## Tipos de protocolos

- Arbitrados



- Adjudicados



- Autosuficientes





## Comunicaciones con criptografía simétrica

- Alice y Bob se ponen de acuerdo sobre el criptosistema a utilizar
- Alice y Bob acuerdan una clave
- Alice encripta el mensaje usando la clave y lo envía a Bob
- Bob desencripta el mensaje con la clave y lee el mensaje

## Comunicaciones con algoritmo de clave pública

- Alice y Bob se ponen de acuerdo sobre el criptosistema a utilizar
- Bob envía a Alice su clave pública
- Alice encripta el mensaje con la clave pública de Bob y se lo envía
- Bob desencripta el mensaje con su clave privada y lo lee

## Criptosistema Híbrido

- Bob envía a Alice su clave pública
- Alice genera una clave temporal, la encripta utilizando la clave pública de Bob y se la envía
- Bob desencripta la clave temporal
- Ambos encriptan sus mensajes durante la sesión utilizando la clave temporal

## Firmas digitales

- Deben garantizar la autenticidad del documento
- Debe ser prueba de que quien firmó, y nadie más, fue realmente quien firmó el documento
- No deben ser reusables
- Nadie debe poder negar que firmó un documento si contiene su firma digital

### Firma de documentos con criptografía simétrica y un árbitro

- Alice encripta su mensaje para Bob con una clave  $K_a$  y la envía a Trent
- Trent desencripta el mensaje con  $K_a$
- Trent une el mensaje a una certificación de que recibió el mensaje de parte de Alicia, encripta todo esto con  $K_b$ , y se lo envía a Bob
- Bob desencripta lo recibido con  $K_b$ . Lee el mensaje y la certificación de Trent.

### Firma de documentos con algoritmos de clave pública

- Alice encripta el mensaje con su clave privada (firma)
- Alice envía el mensaje a Bob
- Bob desencripta el mensaje con la clave pública de Alice, verificando la firma

## Firmas de documentos encriptados

- Alice firma el mensaje con su clave privada
  - $Sa(M)$
- Alice encripta el mensaje firmado con la clave pública de Bob, y lo envía a Bob
  - $Eb(Sa(M))$
- Bob desencripta el mensaje con su clave privada
  - $Db(Eb(Sa(M)))=Sa(M)$
- Bob verifica la firma usando la clave pública de Alice
  - $Va(Sa(M))=M$

## Bit commitment

- Bob genera una cadena random  $R$  y se la envía a Alice
- Alice crea un mensaje con el bit por el cual va a comprometerse y la cadena random enviada por Bob. Lo encripta con una clave random  $K$ , y le envía el resultado. (Bob no puede desencriptarlo, así que no conoce el bit)
- Pasa el tiempo. Alice envía a Bob la clave  $K$
- Bob desencripta el mensaje para ver el bit. Verifica que la cadena random sea igual.

## Fair coin flip

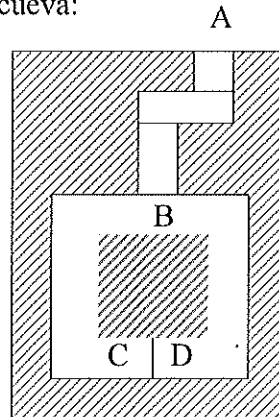
- Alice elige un bit en forma random, y utiliza el protocolo de bit-commitment
- Bob trata de adivinar el bit
- Alice muestra el bit a Bob. Bob gana si adivinó en forma correcta

## Fair coin flip con algoritmos de clave pública

- Alice y Bob generan un par de clave pública/privada cada uno.
- Alice genera dos mensajes distintos. Cada uno tiene que contener una cadena random para que después los pueda identificar. Encripta los dos mensajes con su clave pública y se los envía a Bob en un orden random.
- Bob elige uno al azar. Lo encripta con su clave pública y se lo envía a Alice.
- Alice lo desencripta con su clave privada y se lo envía a Bob.
- Bob desencripta el mensaje con su clave privada para revelar el resultado. Envía el mensaje desencriptado a Alice
- Alice verifica la cadena random. Los dos revelan los pares de claves para que puedan verificar que no hicieron trampa.

## Zero-knowledge proofs

- El problema de la cueva:



## Zero-knowledge proof of identity

- El problema del juego de ajedrez
- El fraude de la mafia
- El fraude del terrorista
- El fraude de las múltiples identidades
- Alquiler de pasaportes

## Dinero digital

- Alice prepara 100 órdenes anónimas por \$1000 c/u.
- Pone cada una en un sobre con papel carbónico.
- El banco abre 99 sobres y confirma que cada uno contiene una orden por \$1000.
- El banco firma el sobre que queda. A través del carbónico se pasa la firma a la orden. Devuelve el sobre a Alice y le descuenta \$1000 de su cuenta.
- Alice abre el sobre y gasta el dinero con un comerciante.
- El comerciante verifica la firma del banco. Lleva la orden al banco para cobrar el dinero.
- El banco verifica su firma y acredita \$1000 a la cuenta del comerciante.

## Knapsacks

- El problema de la mochila:
  - *“Se tiene una mochila con capacidad para ‘K’ kilos. Además se cuenta con una lista de ‘n’ objetos cuyos pesos se conocen y son ( $A_1, A_2, \dots, A_n$ ). El problema consiste en seleccionar una cierta cantidad de objetos de la lista de forma tal que la mochila quede completamente llena.”*
- Para resolverlo, hay que probar todas las combinaciones.

## Knapsacks

- Si los pesos de los objetos están en un vector, cada combinación puede escribirse como un número binario de  $n$  bits. Un uno en la posición  $i$  indica que el elemento  $i$  debe estar en la mochila.

–  $A=(3,45,6,7,21,12,9,90) \rightarrow$  a este vector se lo llama *Knapsack*

–  $C1=(00000001)=90$

–  $C2=(10101010)=3+6+21+9=39$

## Knapsacks

- Ejemplo de encriptación con clave privada, que utiliza knapsacks:

Con bloques de 8 bits y usando el código Ascii, si el vector es:  
(3,45,6,7,21,12,9,90)

"H" =  $01001000 = 45 + 21 = 66$

'o' =  $01101111 = 45 + 6 + 21 + 12 + 9 + 90 = 183$

'l' =  $01101100 = 45 + 6 + 21 + 12 = 84$

'a' =  $01100001 = 45 + 6 + 90 = 141$

'Hola' = 66,183,84,141



## Knapsacks

- Se simplifica el problema de la mochila usando un vector super-incrementante.

Ej: (1,3,5,11,21,44,87,175,349,701) es super-incrementante.

Para 734: Como  $734 > 701 \Rightarrow$  el bit numero 10 es 1

$734 - 701 = 33$

Para 33:  $33 > 21 \Rightarrow$  el bit 5 es 1

$33 - 21 = 12$

Para 12 :  $12 > 11 \Rightarrow$  el bit 4 es 1

$12 - 11 = 1$

Para 1 : El bit 1 es 1

Luego  $734 = (1001100001) = 1 + 11 + 21 + 701$

## Knapsacks

- Una vez que tenemos un vector super-incrementante, se eligen dos números  $t$  y  $m$ , de forma tal que no tengan factores en común.
  - El número  $t$  es el multiplicador
  - El número  $m$  es el módulo
- Además se tiene  $t'$ , inverso multiplicativo de  $t$  en aritmética modulo  $m$ .
- A cada elemento  $A_i$  del knapsack se le aplica:  $A_i' = A_i * t \mod m$ . El vector obtenido es la clave pública (ya no es super-incrementante)
- La clave privada está compuesta por  $t'$  y  $m$ .

## Knapsacks

- Ejemplo: Para encriptar:

Sea  $m=1590$ ,  $t=43$ ,  $t'=37$  ( $37*43=1591$ ,  $1591 \bmod 1590=1$ )  
(1,3,5,11,21,44,87,175,349,701) $\Rightarrow$ (43,129,215,473,903,302,561,1165,697,1523)

Supongamos que queremos encriptar 'Hola'

'H' = 01001000 =  $129 + 903 = 1032$

'o' = 01101111 =  $129 + 215 + 903 + 302 + 561 + 1165 = 3275$

'l' = 01101100 =  $129 + 215 + 903 + 302 = 1549$

'a' = 01100001 =  $129 + 215 + 1165 = 1509$

'Hola' = 1032,3275,1549,1509

## Knapsacks

- Ejemplo: Para desencriptar:

$$1032 * 37 \bmod 1590 = 24$$

$$3275 * 37 \bmod 1590 = 335$$

$$1549 * 37 \bmod 1590 = 73$$

$$1509 * 37 \bmod 1590 = 183$$

Se convierte el vector publico en un vector super-incrementante utilizando la misma transformación.

Luego con los números (24,335,73,183) y el vector super incrementante se puede desencriptar fácilmente el mensaje.

## RSA

- Denominado así debido a sus autores: Rivest, Shamir y Adleman.
- Sean dos números  $p$  y  $q$  primos de aprox. 100 dígitos c/u.
- $n=p*q$  y  $\phi(n) = (p-1) * (q-1)$
- Se elige un número random  $d$  (exponente de descryptación), tal que  $d$  y  $\phi(n)$  son relativamente primos. Y un número  $e$  (exponente de encriptación),  $1 < e < \phi(n)$  tal que  $e*d=1$  usando aritmética módulo  $\phi(n)$
- La clave pública está formada por  $n$  y  $e$ .
- La clave privada está formada por  $p$ ,  $q$ ,  $\phi(n)$  y  $d$ .

## RSA

- Para encriptar, se pasa el mensaje a binario y se lo divide en bloques de un cierto tamaño. Cada bloque se encripta elevando el número a la potencia  $e$  y reduciéndolo módulo  $n$ . Para descryptar, se eleva el código a la potencia  $d$  y se lo reduce módulo  $n$ .
- Ejemplo:  $p=5$ ,  $q=11$ ,  $n=p*q=55$ ,  $\phi(n)=40$ .
  - Elegimos  $d=23$  ( $23$  y  $40$  son relativamente primos)
  - $e=7$  pues  $7*23=161$  ( $161 \bmod 40=1$ )
    - 1       $1(1^7 \bmod 55)$        $1^{23} \bmod 55=1$
    - 2       $18(2^7 \bmod 55)$        $18^{23} \bmod 55=2$
    - 3       $42(3^7 \bmod 55)$        $42^{23} \bmod 55=3$

## PGP

- PGP: Pretty Good Privacy
- Trabaja con un algoritmo RSA con claves de 256, 512 o 1024 bits.
- Genera las claves publicas y privadas del usuario utilizando un algoritmo de pseudoaleatorización que mide los tiempos transcurridos entre lo que se tipea en un teclado. Dichas claves se almacenan en disco
- Las claves publicas de otros usuarios, se almacenan en un conjunto de claves públicas (Public-key-ring).