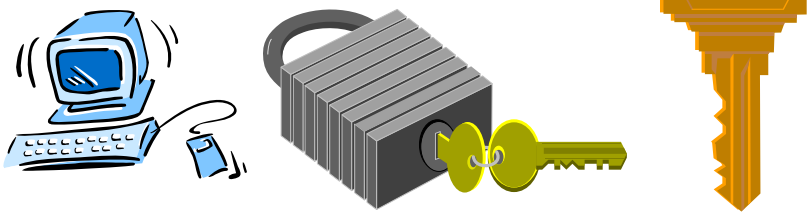
Criptologia Asimétrica

Criptografía de llave pública



Fecha última modificación: marzo 2009

Lámina 1 Roberto Gómez C.

Desventajas llave secreta

- Distribución de llaves
  - usuarios tienen que seleccionar llave en secreto antes de empezar a comunicarse
- Manejo de llaves
  - red de n usuarios, cada pareja debe tener su clave secreta particular, i.e. n(n-1)/2 claves
- Sin firma digital
  - no hay posibilidad , en general, de firmar digitalmente los mensajes

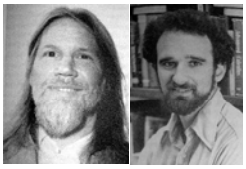
Lámina 2 Roberto Gómez C.

Solución al problema de intercambio de llaves: Criptosistema Diffie Hellman

Criptosistema intercambio llaves

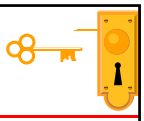
Diffie-Hellman

- Primer algoritmo de llave pública (1976)
  - Williamson del CESG[1] UK, publica un esquema identico unos meses antes en documento clasificado
  - asegura que descubrió dicho algoritmo varios años antes
- Varios productos comerciales utilizan este técnica de intercambio de llaves.
- Proposito del algoritmo
  - permitir que dos usuarios intercambien una llave de forma segura
  - algoritmo limitado al intercambio de llaves
- Basado dificultad para calcular logaritmos discretos



1: Communications-Electronic Security Group

El problema del logaritmo discreto



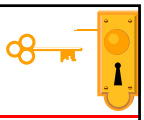
- Dados g, x y p en la formula:

$$y = g^x \bmod p$$

  - el valor de y se puede obtener fácilmente
- Sin embargo dado y,g y p es computacionalmente difícil calcular x, como el logaritmo discreto
- Por ejemplo
  - dado $y = 7^8 \bmod 13$ calcular y es fácil,
  - pero $3 = 7^x \bmod 13$ calcular x es muy difícil
- Conclusión:
  - es muy fácil calcular exponentes mod un primo
  - es muy pesado calcular un logaritmo discreto

Lámina 5 Roberto Gómez C.

Algoritmo de Diffie-Hellman



1. Los dos usuarios A y B seleccionan públicamente un grupo multiplicativo finito, G, de orden n y un elemento de G
2. A genera un número aleatorio X_a, calcula Y_a en G y transmite este elemento a B
3. B genera un número aleatorio X_b, calcula Y_b en G y transmite este elemento a A
4. A recibe Y_b y calcula $(Y_b)^{Xa}$ en G
5. B recibe Y_a y calcula $(Y_a)^{Xb}$ en G
6. La llave es $(Y_b)^{Xa} = (Y_a)^{Xb}$

Lámina 6 Roberto Gómez C.

Esquema Diffie Hellman

Elementos globales públicos: q (numero primo) y α $(\alpha < q)$


A

La llave de A y B es K

$K = (\alpha^{X_a X_b}) \bmod q$


B

Selecciona val. priv: X_a $(X_a < q)$

Calcula valor pub: $Y_a = \alpha^{X_a} \bmod q$

Selecciona val. priv: X_b $(X_b < q)$

Calcula valor pub: $Y_b = \alpha^{X_b} \bmod q$

Y_a →

← Y_b

Generando llave secreta A

$K = (Y_b)^{X_a} \bmod q$

$K = (\alpha^{X_b})^{X_a} \bmod q$

Generando llave secreta B

$K = (Y_a)^{X_b} \bmod q$

$K = (\alpha^{X_a})^{X_b} \bmod q$

Lámina 7 Roberto Gómez C.

Ejemplo Diffie Hellman

Elementos globales públicos: $q = 53$ $\alpha = 2$ $(2 < 53)$


A

La llave de A y B es 21


B

Selecciona val. priv: $X_A = 29$ $(29 < 53)$

Calcula valor pub: $Y_A = 2^{29} \bmod 53$

$= 45 \bmod 53$

Selecciona val. priv: $X_B = 19$ $(19 < 53)$

Calcula valor pub: $Y_B = 2^{19} \bmod 53$

$= 12 \bmod 53$

Y_A (45) →

← Y_B (12)

Generando llave secreta A

$K = 12^{29} \bmod 53 = 21 \bmod 53$

Generando llave secreta B

$K = 45^{19} \bmod 53 = 21 \bmod 53$

Lámina 8 Roberto Gómez C.

Comentarios ejemplo

- La clave privada o la información secreta que comparten ahora A y B es 21
- Un escucha, S, conoce del protocolo anterior:
  - Z_{53}^*, 2, 45 y 12
  - no puede conocer que la información secreta compartida por A y B es 21
- En general:
  - Un intruso que conozca las claves públicas q y α e intercepte el valor Y_a que ha transmitido A y el valor Y_b transmitido por B no podrá descubrir los valores de X_a, X_b ni $\alpha^{XaXb} \mod q$ ...

    Salvo que se enfrente al Problema del Logaritmo Discreto (PLD) que, se vuelve computacionalmente intratable para valores del primo p grandes

Lámina 9 Roberto Gómez C.

¿Es vulnerable el protocolo de DH?

- A elige un número X_a con: $1 < X_a < (q-1)$
  - y envía a B: $Y_a = \alpha^{Xa} \mod q$.
- C intercepta este valor, elige un número X_c con: $1 < X_c < (q-1)$
  - y envía a B: $Y_c = \alpha^{Xc} \mod q$.
- B elige un número X_b con: $1 < X_b < (q-1)$
  - y envía a A: $Y_b = \alpha^{Xb} \mod q$.
- C intercepta este valor
  - y envía a A: $Y_c = \alpha^{Xc} \mod q$ (valor anterior)
- A y B calculan sus claves
  - $K_A = (\alpha^{Xc})^{Xa} \mod q$
  - $K_B = (\alpha^{Xc})^{Xb} \mod q$
- C calcula también las claves:
  - $K_{CA} = (\alpha^{Xa})^{Xc} \mod q$
  - $K_{CB} = (\alpha^{Xb})^{Xc} \mod q$

Por lo tanto, a partir de ahora C tiene "luz verde" y puede interceptar todos los mensajes que se intercambian A y B

La solución a este problema es el sellado de tiempo

Lámina 10 Roberto Gómez C.

D.H. y autenticación (1)


A


B


E

Lámina 11 Roberto Gómez C.

D.H. y autenticación (2)


A


B

Elige X_a

$Y_a = \alpha^{X_a} \bmod q$

Y_a

Y_a


E

Elige X_c

$Y_c = \alpha^{X_c} \bmod q$

Y_c

Lámina 12 Roberto Gómez C.

D.H. y autenticación (3)

A

Elige X_a

$Y_a = \alpha^{X_a} \bmod q$

B

Recibe Y_c

Elige X_b

$Y_b = \alpha^{X_b} \bmod q$

Y_b

Y_b

E

Y_c

Elige X_c

$Y_c = \alpha^{Xc} \bmod q$

Lámina 13

Roberto Gómez C.

D.H. y autenticación (4)

A

Recibe $Y_c = (\alpha^{X_c}) \bmod q$

Calcula $K_{ca} = (\alpha^{X_c})^{X_a} \bmod q$

B

Recibió $Y_c = \alpha^{X_c} \bmod q$

Calcula $K_{cb} = (\alpha^{X_c})^{X_b} \bmod q$

E

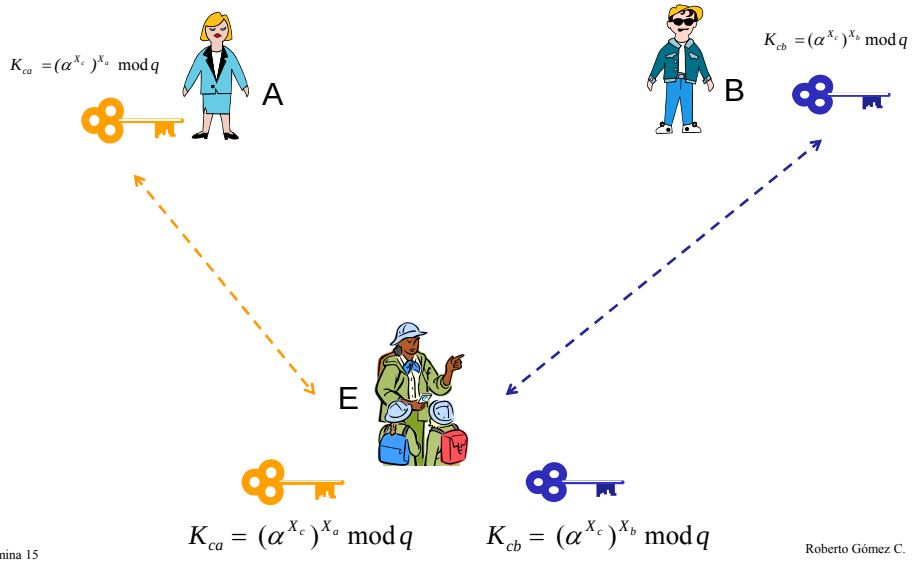
Calcula: $K_{ca} = (\alpha^{Xc})^{X_a} \bmod q$

Calcula: $K_{cb} = (\alpha^{Xc})^{X_b} \bmod q$

Lámina 14

Roberto Gómez C.

D.H. y autenticación (5)



$K_{ca} = (\alpha^{X_c})^{X_a} \bmod q$ $K_{cb} = (\alpha^{X_c})^{X_b} \bmod q$

Lámina 15 Roberto Gómez C.

Intercambio de clave DH entre n usuarios

Protocolo DH se puede generalizar para n usuarios: sea n = 3.

A, **B** y **C** seleccionan un grupo q y un generador α

- **A** genera un número aleatorio X_a y envía α^{Xa} mod q a **B**
- **B** genera un número aleatorio X_b y envía α^{Xb} mod q a **C**
- **C** genera un número aleatorio X_c y envía α^{Xc} mod q a **A**
- **A** recibe α^{Xc} mod p y calcula $(\alpha^{Xc})^{Xa}$ mod q y se lo envía a **B**
- **B** recibe α^{Xa} mod p y calcula $(\alpha^{Xa})^{Xb}$ mod q y se lo envía a **C**
- **C** recibe α^{Xb} mod p y calcula $(\alpha^{Xb})^{Xc}$ mod q y se lo envía a **A**
- **A** recibe α^{XbXc} mod p y calcula $(\alpha^{XbXc})^{Xa}$ mod q = α^{XbXcXa} mod q
- **B** recibe α^{XcXa} mod p y calcula $(\alpha^{XcXa})^{Xb}$ mod q = α^{XcXaXb} mod q
- **C** recibe α^{XaXb} mod p y calcula $(\alpha^{XaXb})^{Xc}$ mod q = α^{XaXbXc} mod q
- El secreto compartido por **A**, **B** y **C** es el valor α^{XaXbXc} mod q

Lámina 16 Roberto Gómez C.

Condiciones intercambio de llave D-H

Condiciones del Protocolo:

- Módulo q debe ser un primo grande, al menos 1024 bits.
- Interesa que q-1 debe tener factores primos grandes.
- Generador α debe ser una raíz primitiva del módulo q.

☹ Si el módulo es un primo pequeño, se puede hacer un ataque por fuerza bruta dentro de un tiempo razonable.

☹ Si el generador no es una raíz primitiva del grupo q, entonces la operación $\alpha^i \bmod q$ $(1 \leq i \leq q-1)$ no genera todos los restos del grupo y esto facilita el ataque por fuerza bruta.

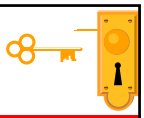
Lámina 17 Roberto Gómez C.

Criptosistemas de llave pública

Características y ejemplos

Lámina 18 Roberto Gómez C.

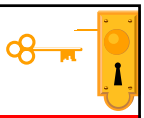
Background



- Concepto de llave pública fue inventado por Whitfield Diffie y Martin Hellman e independientemente por Ralph Merkle.
- Contribución fue que las llaves pueden presentarse en pares.
  - llave pública y llave privada
- Concepto presentado en 1976 por Diffie y Hellman.
- Desde 1976 varios algoritmos han sido propuestos, muchos de estos son considerados seguros, pero son imprácticos.

Lámina 19 Roberto Gómez C.

Background



- Algunos algoritmos solo son buenos para distribución de llaves.
- Otros solo son buenos para encripción.
- Algunos más solo son buenos para firmas digitales.
- Solo tres algoritmos son buenos para encripción y firmas digitales:
  - RSA,
  - ElGamal
  - Rabin.
- Los tres algoritmos son más lentos que los algoritmos simétricos.

Lámina 20 Roberto Gómez C.

Criptograma llave pública (asimétrico)

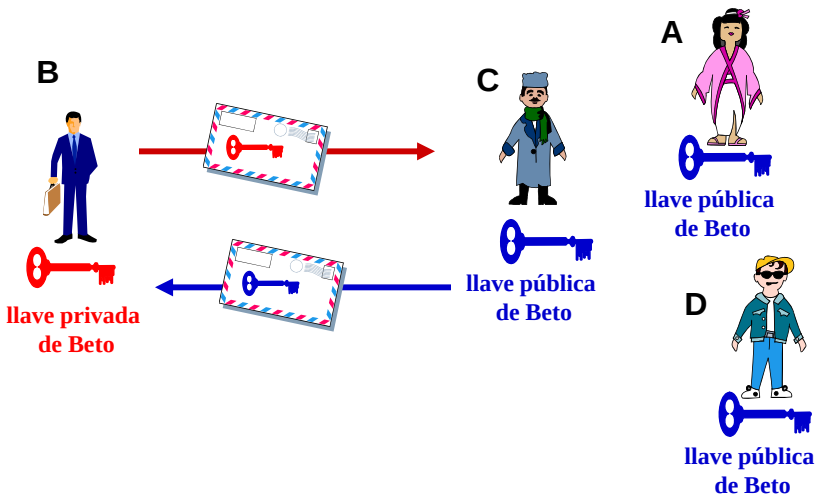


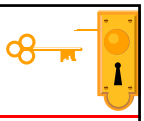
Lámina 21 Roberto Gómez C.

Encriptando con llave pública

- Emisor no usa sus llaves
- Necesario contar con la llave pública del receptor
- Llaves relacionadas matemáticamente
  - teoría de números
  - funciones unidireccionales con puerta trasera
- Dos funciones usadas
  - producto de números enteros, cuya inversa es la factorización del número obtenido (RSA)
  - la exponenciación discreta, cuya inversa es el logaritmo discreto (problema logaritmos discrteto, El Gammal)

Lámina 22 Roberto Gómez C.

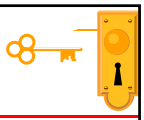
Función Unidireccional



- Una función One-Way Function (OWF):
  - f: M --> C
  - es fácil calcular f(m) = C
  - es difícil calcular $f^{-1}(c) = m$
- Función unidireccional con puerta trasera (Trapdoor One-way Function TOF) si puede ser invertida fácilmente cuando se conoce alguna información adicional extra
- Dicha información se conoce como *puerta trasera*

Lámina 23 Roberto Gómez C.

Criptosistema llave pública



- Familia funciones TOF, $\{f_k\}$, para cada llave k de K, de modo que la trampa t(k) sea fácil de obtener
- Para cada k de K se debe poder describir un algoritmo eficiente que permita calcular f_k pero de modo que sea intratable la determinación de k y t(k)

Lámina 24 Roberto Gómez C.

Implementación

- Dada una familia de TOFs, cada usuario U elige una llave aleatoria u de K y pública E_u que permite calcular f_u:E
  - llave u es su llave pública
  - puerta trasera para invertir f_u es su llave privada
- Si un usuario A desea enviar mensaje a B, mira la llave publica de B, Eb y le envía:

$$f_b(m) = c$$

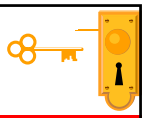
Lámina 25 Roberto Gómez C.

- Como B es el único capaz d invertir fb, es el único que puede recuperar el mensaje m

$$f_b^{-1}(c) = f_b^{-1}(f_b(m)) = m$$

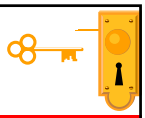
Lámina 26 Roberto Gómez C.

Uso de TOFs en la práctica



- No se ha demostrado existencia TOFs
- Hay dos funciones candidatas a serlo
  - producto de números enteros, cuya inversa es la factorización del número obtenido
  - la exponenciación discreta, cuya inversa es el logaritmo discreto
- Las dos funciones son fáciles de calcular, mientras que sus inversas no lo son

Lámina 27 Roberto Gómez C.



- En el caso de las funciones anteriores
  - dado un número n, es difícil determinar su descomposición en factores primos
  - dados a y b es difícil calcular x de modo que $a^x=b$
- La primera se utiliza en el criptosistema RSA, mientras que la segunda es la base del criptosistema de ElGamal

Lámina 28 Roberto Gómez C.

Encripción con llave pública

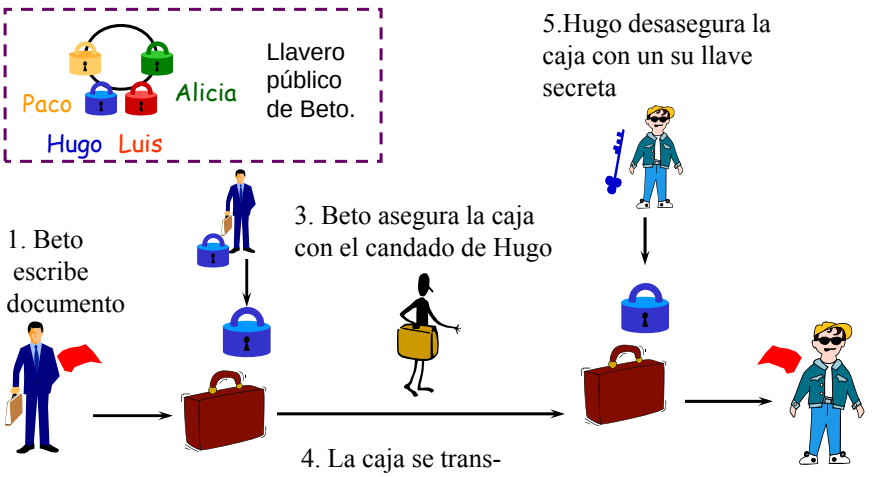


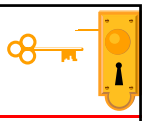
Lámina 29 Roberto Gómez C.

Aritmética modular y cifrado de números

- Utiliza enteros no negativos
- Realiza operaciones aritméticas ordinarias (suma, multiplicación).
- Reemplaza su resultado con el residuo cuando se divide entre n.
- El resultado es modulo n o *mod n*.

Lámina 30 Roberto Gómez C.

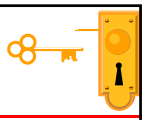
Ejemplo suma en módulo 10



- 5 + 5 = 10 mod 10 = 0
- 3 + 9 = 12 mod 10 = 2
- 2 + 2 = 4 mod 10 = 4
- 9 + 9 = 18 mod 10 = 8

Lámina 31 Roberto Gómez C.

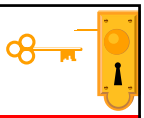
Tabla suma módulo 10



+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	0
2	2	3	4	5	6	7	8	9	0	1
3	3	4	5	6	7	8	9	0	1	2
4	4	5	6	7	8	9	0	1	2	3
5	5	6	7	8	9	0	1	2	3	4
6	6	7	8	9	0	1	2	3	4	5
7	7	8	9	0	1	2	3	4	5	6
8	8	9	0	1	2	3	4	5	6	7
9	9	0	1	2	3	4	5	6	7	8

Lámina 32 Roberto Gómez C.

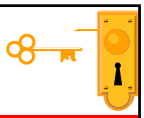
Cifrado usando suma módulo 10



- Suma modulo 10 puede usarse como esquema de encripción de dígitos.
- Encripción:
  digito + <constante> mod 10
- Se mapea cada dígito decimal a uno diferente de tal forma que es reversible.
- La constante es la llave de encripción
- Decripción:
  digito - <constante> mod 10
  si el resultado es menor a cero => sumar 10

Lámina 33 Roberto Gómez C.

Ejemplo encripción suma modular



- Llave secreta: 5
- Encripción:
  - 7 + 5 = 12 mod 10 = 2
  - 8 + 5 = 13 mod 10 = 3
  - 3 + 5 = 8 mod 10 = 8
- Decripción:
  - 2 - 5 = -3 + 10 = 7
  - 3 - 5 = -2 + 10 = 8
  - 8 - 5 = 3

Lámina 34 Roberto Gómez C.

Cifrado con inversa aditiva de x mod 10

- Aritmética regular:
  - substraer x puede hacerse sumando -x
- Inversa aditiva de x
  - número que se le tiene que sumar a x para obtener 0
- Por ejemplo:
  - inversa aditiva de 4 es 6
  - artimética mod 10: 4 + 6 = 10 mod 10 = 0
- Si la llave secreta es 4:
  - para encriptar se añade 4 mod 10
  - para decriptar se añade 6 mod 10

Lámina 35 Roberto Gómez C.

Ejemplo cifrado con inversa aditiva en modulo 10

- Llave encripción: 4
- Encripción:
  - 7 + 4 mod 10 = 11 mod 10 = 1
  - 8 + 4 mod 10 = 12 mod 10 = 2
  - 3 + 4 mod 10 = 7 mod 10 = 7
- Decripción (llave: 6)
  - 1 + 6 mod 10 = 7 mod 10 = 7
  - 2 + 6 mod 10 = 8 mod 10 = 8
  - 7 + 6 mod 10 = 13 mod 10 = 3

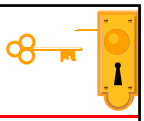
Llave encripción: 4

Llave decripción: 6

¿Es posible decriptar si solo se conoce la llave de encripción?

Lámina 36 Roberto Gómez C.

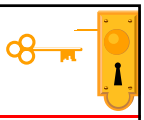
Cifrado con multiplicación modular



- Multiplicación modular: mismo principio que la suma:
  - 7 * 4 mod 10 = 8
  - 3 * 9 mod 10 = 7
  - 2 * 2 mod 10 = 4
  - 9 * 9 mod 10 = 1

Lámina 37 Roberto Gómez C.

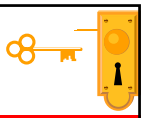
Tabla multiplicación modulo 10



*	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	0	2	4	6	8
3	0	3	6	9	2	5	8	1	4	7
4	0	4	8	2	6	0	4	8	2	6
5	0	5	0	5	0	5	0	5	0	5
6	0	6	2	8	4	0	6	2	8	4
7	0	7	4	1	8	5	2	9	6	3
8	0	8	6	4	2	0	8	6	4	2
9	0	9	8	7	6	5	4	3	2	1

Lámina 38 Roberto Gómez C.

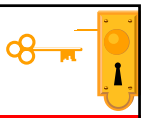
¿Cómo decriptar?



- No es posible aplicar el mismo principio de encripción que en la suma
- Inverso multiplicativo
  - aritmética normal: inverso de x es: x-1 = 1/x
  - número por el cual se debe multiplicar x para obtener el valor de 1: número fraccionario
  - en aritmética modular solo hay enteros
- ¿Cúales números se pueden elegir para encriptar y decriptar?

Lámina 39 Roberto Gómez C.

¿Es posible usar el 5 y el 8?



Encriptando con 5

- 1 * 5 mod 10 = 5
- 2 * 5 mod 10 = 0
- 3 * 5 mod 10 = 5
- 4 * 5 mod 10 = 0
- 5 * 5 mod 10 = 5
- 6 * 5 mod 10 = 0
- 7 * 5 mod 10 = 5
- 8 * 5 mod 10 = 0
- 9 * 5 mod 10 = 5

Encriptando con 8

- 1 * 8 mod 10 = 8
- 2 * 8 mod 10 = 6
- 3 * 8 mod 10 = 4
- 4 * 8 mod 10 = 2
- 5 * 8 mod 10 = 0
- 6 * 8 mod 10 = 8
- 7 * 8 mod 10 = 6
- 8 * 8 mod 10 = 4
- 9 * 8 mod 10 = 2

Lámina 40 Roberto Gómez C.

¿Entonces cuales se pueden usar?

*	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	0	2	4	6	8
3	0	3	6	9	2	5	8	1	4	7
4	0	4	8	2	6	0	4	8	2	6
5	0	5	0	5	0	5	0	5	0	5
6	0	6	2	8	4	0	6	2	8	4
7	0	7	4	1	8	5	2	9	6	3
8	0	8	6	4	2	0	8	6	4	2
9	0	9	8	7	6	5	4	3	2	1

- Se debe escoger con cuidado el multiplicador
- La llave puede ser 1,3,7 o 9 ya que realizan substitución uno a uno de los dígitos
- Problema: **¿Cómo decriptar?**

Lámina 41 Roberto Gómez C.

Ejemplos inversos multiplicativos

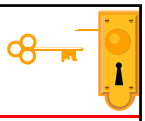
- Se van a usar los números que cuenten con un inveso multiplicativo: {1,3,7,9}
- Ejemplo 1:
  - 7 es el inverso multiplicativo de 3

    3 x 7 mod 10 = 21 mod 10 = 1
  - Entonces: encripción con 3 y decripción con 7

Encripción	Decripción
7 * 3 mod 10 = 1	1 * 7 mod 10 = 7
8 * 3 mod 10 = 4	4 * 7 mod 10 = 8
3 * 3 mod 10 = 9	9 * 7 mod 10 = 3

Lámina 42 Roberto Gómez C.

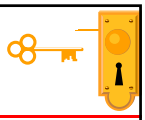
En general



- Criptosistema:
  - se puede modificar la información a través de un algoritmo y revertir el proceso para obtener la información original.
- Una multiplicación mod n por un número x es un criptosistema ya que:
  - se puede multiplicar por x mod n para encriptar
  - se puede multiplicar por x^{-1} mod n para decriptar

Lámina 43 Roberto Gómez C.

Primera observación

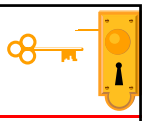


- No es tan simple encontar un inverso multiplicativo mod n, especialmente si n es muy grande,
- Si n = 100 dígitos
  - no es lógico realizar una busqueda de fuerza bruta para encontrar un inverso multiplicativo
- Algoritmo ecludiano
  - permite encontrar inversos mod n, dado x y n encuentra y tal que:

    x * y mod n = 1 (si existe)

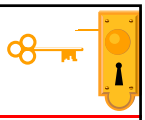
Lámina 44 Roberto Gómez C.

Segunda observación



- ¿Por qué los números {1,3,7,9} son los únicos que tienen inversos multiplicativos?
  - respuesta: son relativamente primos a 10.
- Relativamente primos a 10:
  - significa que no comparte ningún factor común aparte de 1, i.e. mcd(1,10) = 1
  - el entero más largo que divide 9 y 10 es 1
  - el entero más largo que divide 7 y 10 es 1
  - el entero más largo que divide 3 y 10 es 1
  - el entero más largo que divide 1 y 10 es 1

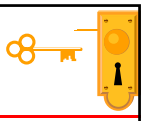
Lámina 45 Roberto Gómez C.



- En constraste 6, 2, 4, 5 y 8 son primos en 10 ya que:
  - 2 divide a 6 y 10, i.e. mcd(6,10) = 2
  - 2 divide a 2 y 10, i.e. mcd(2,10) = 2
  - 2 divide a 4 y 10, i.e. mcd(4,10) = 2
  - 5 divide a 5 y 10, i.e. mcd(5,10) = 5
  - 2 divide a 8 y 10, i.e. mcd(8,10) = 2
- Conclusión
  - cuando se trabaja con aritmetica mod n, todos los números relativos primos a n tienen multiplicativos inversos y los otros números no.

Lámina 46 Roberto Gómez C.

El mcd y los números relativamente primos a n

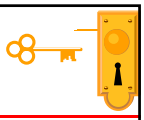


$\exists$ inverso a^{-1} en mod n *ssi* mcd (a, n) = 1

- Para poder determinar si un número cuenta con un inverso multiplicativo en aritmética modular n, es necesario encontrar el máximo común denominador, mcd, entre dos números a y b.
- Posible usar el algoritmo de Euclides para lo anterior

Lámina 47 Roberto Gómez C.

La función totient de Euler



- ¿Cuantos números a n pueden ser relativamente primos a n?
  - Respuesta: función totient $\Phi(n)$
  - to = total tient = quotient (cociente)
- Si n es primo:

$$\Phi(n) = n - 1$$

existen n-1 números relativamente primos a n

- Si n es un producto de dos números primos (p y q)

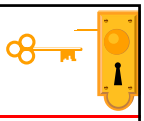
$$\Phi(n) = \Phi(pq) = \Phi(p) \times \Phi(q)$$

$$\Phi(n) = (p-1)(q-1)$$

existen (p-1)(q-1) números relativamente primos a n

Lámina 48 Roberto Gómez C.

Como se calcula el inverso de a en el cuerpo n

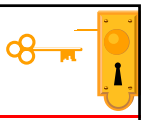


- Teorema de Euler/Fermat
  - basado en la función totient de Euler
- Algoritmo extendido de Euclides
  - es el método más rápido y práctico
- Teorema del Resto Chino TRC

ESTAMOS LISTOS PARA DISEÑAR UN ALGORITMO DE ENCRIPCION...

Lámina 49 Roberto Gómez C.

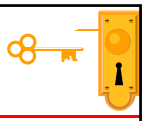
La función totient de Euler



- ¿Cuántos números a n pueden ser relativamente primos a n?
  - respuesta: función totient $\Phi(n)$
  - to = total tient = quotient (cociente)
- Conjunto reducido de restos CRR
  - el conjunto reducido de restos, conocido como CRR de n, es el subconjunto $\{0, 1, \ldots n_i, \ldots n-1\}$ de restos primos con el grupo n.
- Si n es primo, todos los restos serán primos con él
- Podremos representar cualquier número n de cuatro formas:
  - n es un número primo.
  - n se representa como $n = p^k$ con p primo y k entero.
  - n es el producto $n = p*q$ con p y q primos
  - n es un número cualquiera (genérico).

Lámina 50 Roberto Gómez C.

Función ϕ(n) de Euler (n = p)



- Caso 1: n es un número primo
- Si n es primo, ϕ(n) será igual a CCR menos el 0.

$$\phi(n) = n - 1$$ Se usará en sistemas ElGamal y DSS

- Si n es primo, entonces CRR = CCR - 1 ya que todos los restos de n, excepto el cero, serán primos entre sí
- Ejemplo:
  - CRR(7) = {1,2,3,4,5,6} seis elementos,

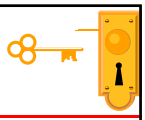
    ∴ $\phi(7) = n - 1 = 7-1 = 6$

    $\phi(11) = 11-1 = 10$

    $\phi(23) = 23-1 = 22$

Lámina 51 Roberto Gómez C.

Función ϕ(n) de Euler (n = p^k)



- Caso 2: $n = p^k$ (con p primo y k un entero)
- De los p^k elementos del CCR, restaremos todos los múltiplos $1*p, 2*p, 3*p, ...(p^{k-1}-1)*p$ y el cero.

$$\phi(n) = \phi(p^k) = p^k - p^{k-1} \qquad \phi(p^k) = p^{k-1}(p-1)$$

- Ejemplo
  - CRR(16) = {1,3,5,7,9,11,13,15} ocho elementos,

    ∴ $\phi(16) = \phi(2^4) = 2^{4-1}(2-1) = 2^3*1 = 8$

    $\phi(125) = \phi(5^3) = 5^{3-1}*(5-1) = 5^2*4 = 25*4 = 100$

Lámina 52 Roberto Gómez C.

Función ϕ(n) de Euler (n = p∗q) (1)

- Caso 3: n = p∗q (con p y q primos)

$$\phi(n) = \phi(p*q) = \phi(p)*\phi(q) = (p-1)(q-1)$$

- De los p∗q elementos del CCR, restaremos todos los múltiplos de p = 1∗p, 2∗p, ... (q - 1)∗p, todos los múltiplos de q = 1∗q, 2∗q, ... (p - 1)∗q y el cero

$$\phi(p*q) = p*q - [(q-1) + (p-1) + 1] = \underbrace{p*q - q - p + 1}_{(p-1)(q-1)}$$

Lámina 53 Roberto Gómez C.

Función ϕ(n) de Euler (n = p∗q) (2)

- Ejemplo
  - CRR(15) = {1,2,4,7,8,11,13,14} ocho elementos

    ∴ ϕ(15) = ϕ(3∗5) = (3-1)(5-1) = 2∗4 = 8

    ϕ(143) = ϕ(11∗13) = (11-1)(13-1) = 10∗12 = 120

Será una de las funciones más utilizadas ya que es la base del sistema RSA que durante muchos años ha sido un estándar de hecho.

Lámina 54 Roberto Gómez C.

Función ϕ(n) Euler (n = genérico)

- Caso 4: $n = p_1^{e1} * p_2^{e2} * \ldots * p_t^{et}$ (p_i son primos)

$$\phi(n) = \prod_{i=1}^{t} p_i^{ei-1} (p_i - 1)$$

- Ejemplo
  - CRR(20) = {1, 3, 7, 9, 11, 13, 17, 19} ocho elementos

    $\therefore \phi(20) = \phi(2^2 * 5) = 2^{2-1}(2-1) * 5^{1-1}(5-1)$

    $\phi(20) = 2^1 * 1 * 1 * 4 = 8$

    $\therefore \phi(360) = \phi(2^3 * 3^2 * 5) =$

    $\phi(360) = 2^{3-1}(2-1) * 3^{2-1}(3-1) * 5^{1-1}(5-1) = 96$

Lámina 55 Roberto Gómez C.

Euler's Totient Function

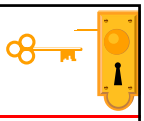
n	Φ(n)
1	1
2	1
3	2
4	2
5	4
6	2
7	6
8	4
9	6
10	4

n	Φ(n)
11	10
12	4
13	12
14	6
15	8
16	8
17	16
18	6
19	18
20	8

n	Φ(n)
21	12
22	10
23	22
24	8
25	20
26	12
27	18
28	12
29	28
30	8

Lámina 56 Roberto Gómez C.

¿Y cómo calculamos la inversa multiplicativa de un numero mod n?



Teorema de Euler

Dice que si mcd (a,n) = 1 $\Rightarrow$ $a^{\phi(n)} \bmod n = 1$

Ahora igualamos $a*x \bmod n = 1$ y $a^{\phi(n)} \bmod n = 1$

$\therefore \quad a^{\phi(n)} * a^{-1} \bmod n = x \bmod n$

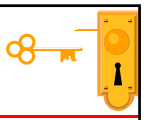
$\therefore \quad x = a^{\phi(n)-1} \bmod n$

El valor x será el inverso de a en el cuerpo n

Nota: Observe que se ha ***dividido*** por a en el cálculo anterior. Esto se puede hacer porque mcd (a, n) = 1 y por lo tanto hay un único valor inverso en el cuerpo n que lo permite.

Lámina 57 Roberto Gómez C.

Ejemplo cálculo inversos con Teorema de Euler



¿Cuál es el inverso de 4 en módulo 9? $\Rightarrow$ inv (4, 9)

Pregunta: ¿Existe $a * x \bmod n = 4 * x \bmod 9 = 1$?

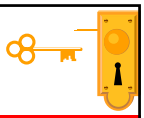
Como mcd (4, 9) = 1 $\Rightarrow$ Sí ... aunque 4 y 9 no son primos.

$\phi(9) = 6 \quad \therefore \quad x = 4^{6-1} \bmod 9 = 7 \quad \Rightarrow \quad 7*4 = 28 \bmod 9 = 1$

Resulta obvio que: inv (4, 9) = 7 e inv (7, 9) = 4

Lámina 58 Roberto Gómez C.

¿Y si no se conoce ϕ(n)?

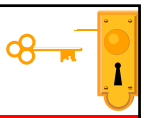


- Calcular a^i mod n cuando los valores de i y a son grandes, se hace tedioso pues hay que utilizar la propiedad de la reducibilidad repetidas veces.
- Si no conocemos ϕ(n) o no queremos usar el teorema de Euler/Fermat, siempre podremos encontrar el inverso de a en el cuerpo n usando el

Algoritmo Extendido de Euclides

Es el método más rápido y práctico

Lámina 59 Roberto Gómez C.



Tarea

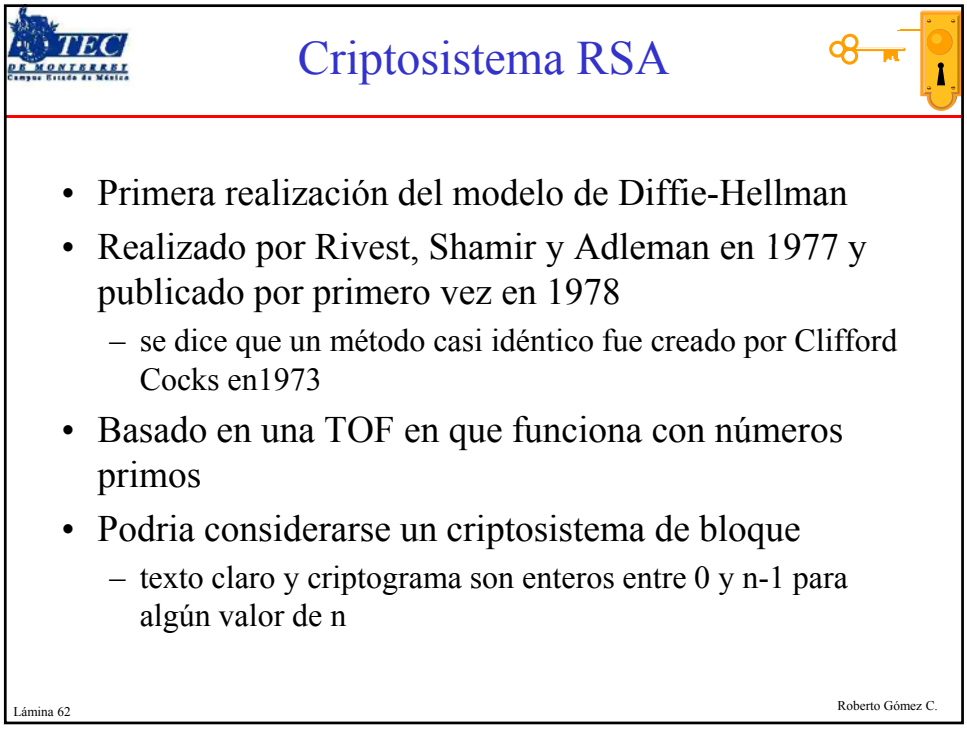
Investigar teorema del resto chino.

Dar dos ejemplos de obtención de la inversa de un número con este teorema.

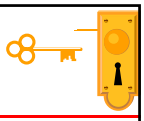
Describir otras aplicaciones.

Lámina 60 Roberto Gómez C.


Criptosistema RSA
Encripción/decripción
Lámina 61
Roberto Gómez C.


Criptosistema RSA
• Primera realización del modelo de Diffie-Hellman
• Realizado por Rivest, Shamir y Adleman en 1977 y publicado por primero vez en 1978
– se dice que un método casi idéntico fue creado por Clifford Cocks en1973
• Basado en una TOF en que funciona con números primos
• Podria considerarse un criptosistema de bloque
– texto claro y criptograma son enteros entre 0 y n-1 para algún valor de n
Lámina 62
Roberto Gómez C.

Características RSA

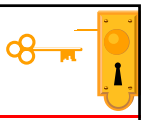


- El texto claro es encriptado en bloques
  - cada bloque tienen un valor binario menor a un número n
  - el tamaño del bloque debe ser igual o menor a $log_2(n)$
  - en la práctica el tamaño del bloque es de 2^k bits, donde

$$2^k < n \leq 2^{k+1}$$

Lámina 63 Roberto Gómez C.

El algoritmo RSA



- Dos etapas
  - la creación de las llaves
  - la encripción/decripción del mensaje
- La creación de llaves

1. Cada usuario elige un número n = p*q (pueden ser distintos).
2. Los valores p y q no se hacen públicos.
3. Cada usuario calcula $\phi(n) = (p-1)(q-1)$.
4. Cada usuario elige una llave pública e (e < n) y que cumpla:

   $\text{mcd}[e, \phi(n)] = 1$.
5. Cada usuario calcula la llave privada que cumpla:

   $d = \text{inv}[e, \phi(n)]$.
6. Se hace público el número n y la llave e.
7. Se guarda en secreto la llave d.

Podrían destruirse ahora p, q y ϕ(n).

Lámina 64 Roberto Gómez C.

Encripción/decripción mensajes

- Tomando en cuenta que las llaves son:

  Llave pública: (e, n)

  Llave privada: (d, n)

- Si se desea encriptar un mensaje M:
  - se tiene que cumplir: $M < n$
  - es necesario usar la llave pública (e,n) :

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

- Para decriptar el criptograma C es necesario usar la llave privada (d,n)

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

Lámina 65 Roberto Gómez C.

Ejemplo generación llaves RSA

- Alicia desea generar sus llaves

1. Elige un número n = 7 * 13 = 91
2. 7 y 13 permanecen secretos
3. $\phi(n) = \phi(7*13) = (7-1)(13-1) = 72$
4. Se elige una llave pública e=5 (5 < 91) que cumple:

   mcd [e, ϕ(n)] = mcd[5,72] = 1
5. Se calcula una llave privada

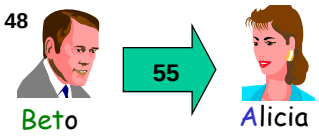
   d = inv [e,ϕ(n)] = inv [5,72] = 29
6. Se envía a Beto la llave pública (5,91)
7. Permanece en secreto 29

Lámina 66 Roberto Gómez C.

Ejemplo encripción/decripción RSA

- Mensaje a encriptar: M=48
- Para encriptar M, Beto toma la llave pública (5,91)

  $C = M^e \bmod n$
  $C = 48^5 \bmod 91$
  $C = 5245.803.968 \bmod 91$
  $C = 55$



- Se envía el mensaje 55 al receptor
- Para decriptar C, Alicia toma la llave privada (29, 91)

  $M = C^d \bmod n$
  $M = 55^{29} \bmod 91$
  $M = 2.954x10^{50} \bmod 91$
  $M = 48$

Lámina 67 Roberto Gómez C.

¿Cómo enviar caracteres en lugar de números?

- Primera opción
  - representar el mensaje en su valor ANSI/ASCII decimal
- Segunda opción
  - codificar el mensaje, expresarlo en base 26 de modo que el mensaje sea menor al número n elegido para generar la llave

A = 0	F = 5	K = 10	P = 15	U = 20	Z = 25
B = 1	G = 6	L = 11	Q = 16	V = 21	
C = 2	H = 7	M = 12	R = 17	W = 22	
D = 3	I = 8	N = 13	S = 18	X = 23	
E = 4	J = 9	O = 14	T = 19	Y = 24	

- En los dos casos hay que cuidar que el número resultante no sea mayor que el módulo de trabajo
  - n = pq

Lámina 68 Roberto Gómez C.

Ejemplo representación ASCII decimal

- Beto desea enviar mensaje Olé a Alicia
- Llave pública Alicia(5,91)
- LLave privada: (29,91)

Se recupera el mensaje agrupando en bloques de 2 dígitos excepto el último

M = Olé = 79 108 233 $\Rightarrow$ M = 79 10 82 33

ENCRIPCION

$C_1 = 79^5 \mod 91 = 53$
$C_2 = 10^5 \mod 91 = 82$
$C_3 = 82^5 \mod 91 = 10$
$C_4 = 33^5 \mod 91 = 24$

DECRIPCION

$M_1 = 53^{29} \mod 91 = 79$
$M_2 = 82^{29} \mod 91 = 10$
$M_3 = 10^{29} \mod 91 = 82$
$M_3 = 24^{29} \mod 91 = 33$

Lámina 69 Roberto Gómez C.

Ejemplo codificación base 26

- Beto desea enviar mensaje YES a Alicia
- Llave pública Alicia(5,91)
- Llave privada: (29,91)

$26^1 = 26$
$26^2 = 676$
$26^3 = 1756$

n = 91
26 < 91 < 676
solo una letra

M = YES

ENCRIPCION

$C_1 = Y = 24 \times 26^0 = 24^5 \mod 91 = 33$
$C_2 = E = 4 \times 26^0 = 4^5 \mod 91 = 23$
$C_3 = S = 18 \times 26^0 = 18^5 \mod 91 = 44$

DECRIPCION

$M_1 = 33^{29} \mod 91 = 24 = Y$
$M_2 = 23^{29} \mod 91 = 04 = E$
$M_3 = 44^{29} \mod 91 = 18 = S$

Lámina 70 Roberto Gómez C.

RSA como un criptosistema de bloques

- Texto claro y criptograma son enteros entre 0 y n-1 para algún valor de n
- Concepto bloque diferente a criptosistemas simétricos en bloques
- Por ejemplo: tomando en cuenta un valor de n = 32 => 5 bits

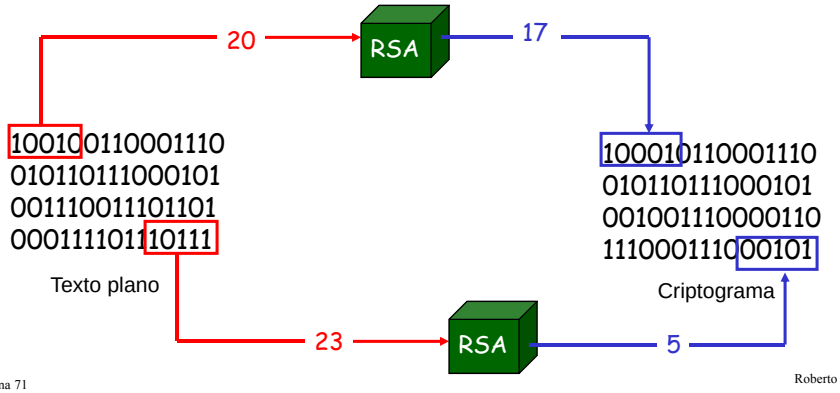


Lámina 71 Roberto Gómez C.

Requerimientos RSA

- Es posible encontrar valores de e,d y n tal que

$$M^{ed} = M \bmod n \quad \forall\, M < n$$

- Es relativamente fácil calcular M^e y C^d, para todos los valores $M < n$
- Es casi imposible determinar d si se conocen e y n

Lámina 72 Roberto Gómez C.

¿Porqué trabaja RSA?

- RSA trabaja con aritmética modulo n, donde $n = pq$
- Se sabe que $\phi(n) = (p-1)(q-1)$
- Se eligieron llaves d y e, tal que $de = 1 \bmod \phi(n)$, entonces:

para cada x: $x^{de} = \bmod n$

- Encripción: tomar x y elevarla a e
- Decripción: tomar resultado encripción y elevarlo a d:

$$(x^e)^d = x^{ed} = x$$

$\Rightarrow$ la decripción revierte la encripción

Lámina 73 Roberto Gómez C.

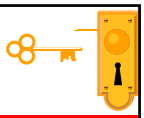
Velocidades

	512 bits	768 bits	1024 bits
Encriptar	0.03	0.05	0.08
Decriptar	0.16	0.48	0.93
Firmar	0.16	0.52	0.97
Verificar	0.02	0.07	0.08

Velocidades para diferentes longitudes de módulos con una llave pública de 8 bits en una Sparc II

Lámina 74 Roberto Gómez C.

Ataques RSA



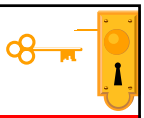
- Fuerza Bruta
  - intentar todas las llaves posibles
- Ataques matemáticos
  - factorizar n en dos números primos, calcular:

    $\Phi(n) = (p-1)(q-1)$ y $d = e^{-1} \bmod \Phi(n)$
  - determinar directamente $\Phi(n)$

    necesario determinar: $d = e^{-1} \bmod \Phi(n)$
  - determina *d* directamente
- Ataques de tiempo
  - comparación de tiempo de decripción

Lámina 75 Roberto Gómez C.

El problema de factorización



- En 1977 se lanzó un reto matematico
- Artículo *A New Kind of Cipher that Would Take Millions of Years to break*
- Columna Mathematical Games en Scientific American
- Criptosistema encriptado con llave pública

  114,381,625,757,888,867,669,235,779,926,146,612,010,218,296,721,
  242,362,562,561,842,935,706,935,245,733,897,830,597,123,563,958,
  705,058,989,075,147,599,290,026,879,543,541
- Se estima que la factorización tomo aproximadamente 4000 a 6000 MIPS años de computo sobre un periodo de seis a ocho meses.

Lámina 76 Roberto Gómez C.

La solución

- El 26 de abril de 1994, un equipo de 600 voluntarios anunciaron los factores de N
- El factor q

  3,490,529,510,847,650,949,147,849,619,903,898,133,417,764,638, 493,387,843,990,820,577
- El factor p

  32,769,132,993,266,709,549,961,988,190,834,461,413,177,642,967, 992,942,539,798,288,533
- El mensaje era:

  200805001301070903002315180419000118050019172105011309190800 151919090618010705

  "THE MAGIC WORDS ARE SQUEAMISH OSSIFRAGE"

Lámina 77 Roberto Gómez C.

Tiempos factorizacion

Numero Digitos decimales	Número de bits (aprox)	Fecha del logro	MIPS-año	Algoritmo
100	332	abril 1991	7	Quadratic sieve
110	365	abril 1992	75	Quadratic sieve
120	398	junio 1993	830	Quadratic sieve
129	428	abril 1994	5000	Quadratic sieve
130	431	abril 1996	500	Generalizado

MIPS-año: procesador de un millón de instrucciones por segundo corriendo un año, lo cual equivale a $2x10^{13}$ instrucciones ejecutadas. Un Pentium 200 MHz equivale aprox. a una máquina de 50 MIPS

Lámina 78 Roberto Gómez C.

¿Y hoy en día?

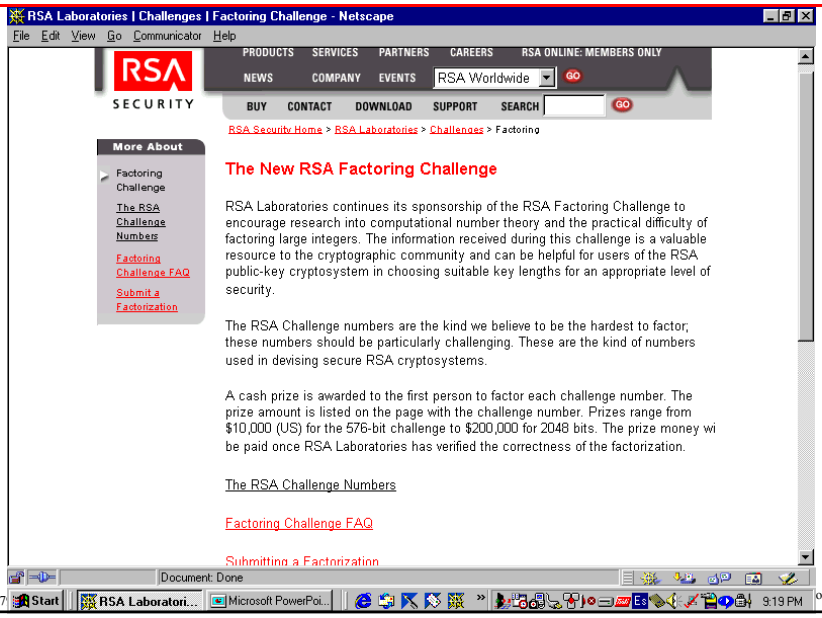


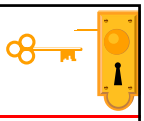
Lámina 7 o Gómez C.

Criptosistema ElGamal

y el problema del logaritmo discreto

Lámina 80 Roberto Gómez C.

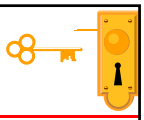
Criptosistema ElGamal



- Propuesto por Taher ElGamal en 1985
- Basado en el problema de los algoritmos discretos.
- Se le considera igual que Diffie-Hellman, excepto que uno de los factores es parte de la llave y se multiplica para la encripción.
- El criptosistema no esta patentado, pero se cree que esta bajo la patente de Diffie-Hellman.
  - la patente de DF expiró el 29 abril de 1997

Lámina 81 Roberto Gómez C.

Generación de llaves en ElGamal

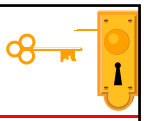


- Elegir un número primo, p y dos números aleatorios g y x tal que
  - g < p
  - x < p

  ⇒ $y = g^x \bmod p$
- Llave pública
  - la llave pública es: y, g y p
- Llave privada
  - la llave privada es: x

Lámina 82 Roberto Gómez C.

Encripción



- Se desea encriptar el mensaje M
- Elegir un número aleatorio k

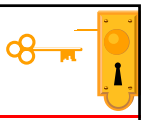
  $k < p-1$
- Calcular

$$a = g^k \bmod p$$
$$b = y^k M \bmod p$$

- El par a y b constituyen el criptograma
- Nota:
  - el criptograma es el doble de tamaño que el texto claro

Lámina 83 Roberto Gómez C.

Decripción



- Se recibe el mensaje a, b
- Para recuperar el mensaje M:

$$M = b/a^x \bmod p$$

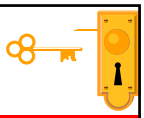
recordar:
x = inv{a, p} si y solo si
mcd(a,p) = 1
x*a mod p = 1

- es decir:

$$M = b/a^x \bmod p$$
$$M = b * (a^{x-1}) \bmod p$$
$$M = b * \{inv\ a^x,\ p\} \bmod p$$

Lámina 84 Roberto Gómez C.

Ejemplo generación llaves



- Beto elige los números:

  $p = 13$
  $g = 6$
  $x = 5$

  $y = g^x \bmod p$
  $y = 6^5 \bmod 13$
  $y = 2$

- Llave pública de Beto

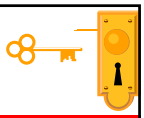
  (y, g, p)
  $(2, 6, 13)$

- Llave privada de Beto

  $X = 5$

Lámina 85 Roberto Gómez C.

Encripción mensaje



- Alicia va a enviar el mensaje `10` a Beto
- Llave pública Beto:

  $p = 13$
  $g = 6$
  $y = 2$

- Para encriptar:
  - selecciona valor arbitrario $k = 4$

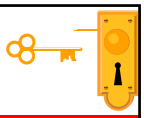
$a = g^k \bmod p$
$b = y^k M \bmod p$

$a = 6^4 \bmod 13 = 9$
$b = 2^4\ 10 \bmod 13 = 4$

- Alicia envía $[9,4]$ a Beto

Lámina 86 Roberto Gómez C.

Decripción del mensaje



- Beto recibe mensaje ***[9,4]***
- Su llave privada es:

  $x = 5$

  – y también conoce:

  $p = 13$

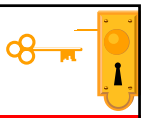
- Se llevan a cabo los siguiente cálculos:

$$M = b/a^x \bmod p$$
$$a^x \bmod p = 9^5 \bmod 13 = 59{,}049 \bmod 13 = 3$$
$$\{inv\ 9^5, 13\} = \{inv\ 3, 13\} = 9$$
$$M = b * \{inv\ a^x, p\} \bmod p = 4*9 \bmod 13$$

- Por lo que el mensaje es: M = 10

Lámina 87 Roberto Gómez C.

Velocidades criptosistema ElGamal



	512 bits	768 bits	1024 bits
Encripción	0.33 seg	0.80 seg	1.09 seg
Decripción	0.24 seg	0.58 seg	0.77 seg
Firma	0.25 seg	0.47 seg	0.63 seg
Verificar	1.37 seg	5.12 seg	9.30 seg

Velocidades para diferentes longitudes de módulos con exponentes de 160 bits en una Sparc II

Lámina 88 Roberto Gómez C.

Comparando las dos

RSA	512 bits	768 bits	1024 bits
Encriptar	0.03	0.05	0.08
Decriptar	0.16	0.48	0.93
Firmar	0.16	0.52	0.97
Verificar	0.02	0.07	0.08

ElGammal	512 bits	768 bits	1024 bits
Encripción	0.33 seg	0.80 seg	1.09 seg
Decripción	0.24 seg	0.58 seg	0.77 seg
Firma	0.25 seg	0.47 seg	0.63 seg
Verificar	1.37 seg	5.12 seg	9.30 seg

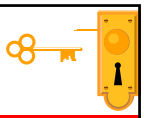
Lámina 89 Roberto Gómez C.

Knapsack cipher algorithms

o ¿cómo llenamos la mochila?

Lámina 90 Roberto Gómez C.

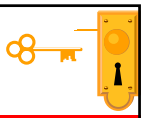
Antecedentes



- Primer algoritmo para cifrado
  - propuesto por Ralph Merkle y Martin Hellman
- Solo puede ser usado para cifrado
  - Adi Shamir lo adapta para firma digital
- Radica su seguridad en el problema de la mochila
  - problema NP-completo
- Tiempo después se demuestra que no es segura

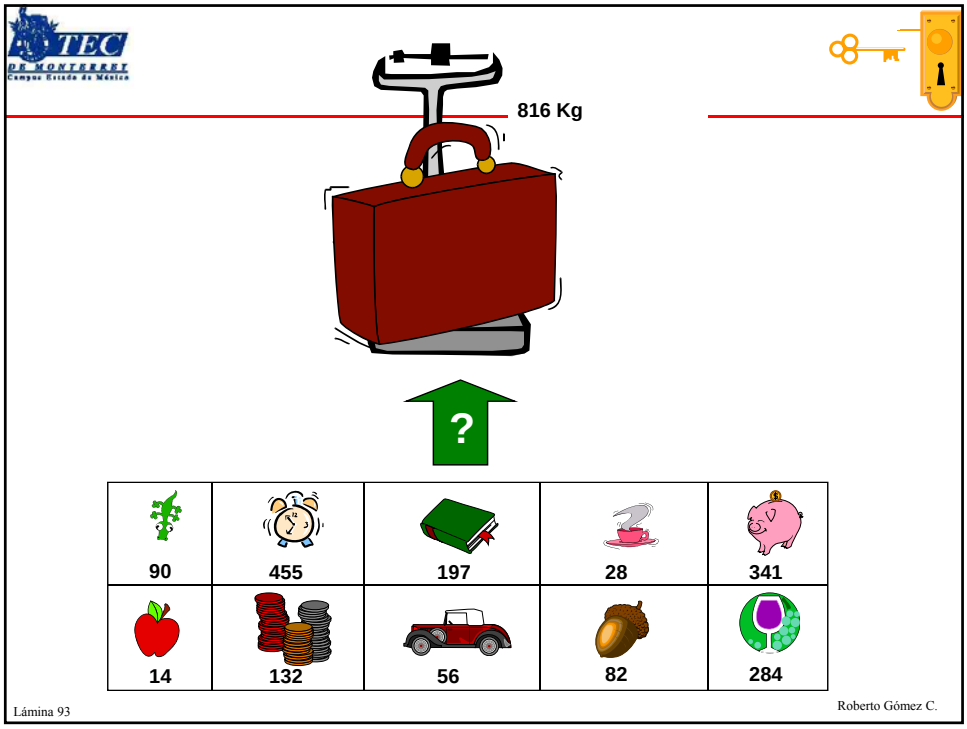
Lámina 91 Roberto Gómez C.

El problema de la mochila



- En inglés: *the Knapsack Problem*
- *Dados una mochila de volumen N y un conjunto de n elementos de volúmenes e_i con $i = 1,2,..,n$, que se desean meter en la mochila. El objetivo es llenar completamente la mochila.*
- Corresponde a problema de programación lineal
- Es la base para definir un criptosistema que es computacionalmente más rápido que RSA

Lámina 92 Roberto Gómez C.



Planteamiento del problema

- Dado un conjunto de valores $M_1, M_2, \ldots M_n$ y una suma S, calcular los valores de b_i, tal que:

$$S = b_1M_1 + b_2M_2 + b_3M_3 + b_4M_4 + b_5M_5 + b_6M_6$$

- La variable b_i puede tomar un valor de *1* ó *0*
  - *1* si el objeto se encuentra en la mochila
  - *0* si el objeto no se encuentra en la mochila
- Por ejemplo

$M_1 = 1, M_2 = 5, M_3 = 6, M_4 = 11, M_5 = 14, M_6 = 20$

Si S = 22, entonces:

$b_1 = 0, b_2 = 1, b_3 = 1, b_4 = 1, b_5 = 0, b_6 = 0$

Para un valor de S=24, no se puede empacar una mochila

Lámina 94 Roberto Gómez C.

Principio algoritmo encripción

- Idea base: encriptar mensaje como una solución a una serie de problemas de la mochila
- Tamaño bloque de texto claro es igual a la longitud del número de productos seleccionados en la mochila

bit texto claro = valor b_i

- El criptograma es la suma de los objetos de la mochila
- Ejemplo

Texto claro:	1 1 1 0 0 1	0 1 0 1 1 0	0 0 0 0 0 0	0 1 1 0 0 0
Mochila:	1 5 6 11 14 20	1 5 6 11 14 20	1 5 6 11 14 20	1 5 6 11 14 20
	1+5+6+20	5+11+14	0+0...+0	5+6
	32	**30**	**0**	**11**

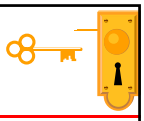
Lámina 95 Roberto Gómez C.

Tipos problema mochila

- Uno que se puede resolver en tiempo lineal
  - lista de pesos es super-incremental
- El otro se cree que no se puede resolver
  - lista de pesos no super-incremental
- El de la mochila fácil se puede modificar para crear uno difícil de mochila
- La llave pública: mochila difícil
  - puede ser usado fácilmente para encriptar, pero no puede usarse para decriptar
- La llave privada: mochila fácil
  - proporciona una forma fácil de decriptar mensajes

Lámina 96 Roberto Gómez C.

Problema mochila super-incremental

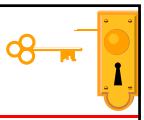


- Lista pesos super-incremental
  - cada termino es mayor que la suma de todos los anteriores

    {1,3,6,13,27,52} → Si
    {1,3,4,9,15,25} → No
- Solución a una lista super-incremental es fácil de encontrar
  - tomar el peso total y comparar con el número más grande de la lista/secuencia
  - si el peso total es menor que el número, entonces no esta en la mochila
  - si el peso total es mayor o igual, entonces si se encuentra en la mochila

Lámina 97 Roberto Gómez C.

Ejemplo solución super-incremental



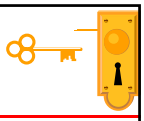
- Ejemplo de solución
- Peso de la mochila: 415
- Peso de los productos

1	3	7	13	26	65	119	267

- $415 \geq 267$ $x_8 = 1$
- $415 - 267 = 148 \geq 119$ $x_7 = 1$
- $148 - 119 = 29 < 65$ $x_6 = 0$
- $29 \geq 26$ $x_5 = 1$
- $29 - 26 = 3 < 13$ $x_4 = 0$
- $3 < 7$ $x_3 = 0$
- $3 \geq 3$ $x_2 = 1$
- $3 - 3 = 0 < 1$ $x_1 = 0$

Lámina 98 Roberto Gómez C.

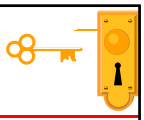
Pasando de lo fácil a lo difícil: creando las llaves



- Tomar una secuencia de mochila super-incremental
  - por ejemplo: {2,3,6,13,27,52}
- Multiplicar todos los valores por un número *n mod m*
- El valor de m debe ser un número más grande que la suma de todos los números en la secuencia:
  - por ejemplo 105
- El multiplicador no debe tener factores comunes con el módulo m
  - por ejemplo 31

Lámina 99 Roberto Gómez C.

Las llaves ...



- La secuencia normal de la mochila es:

$$2\times 31 \bmod 105 = 62$$
$$3\times 31 \bmod 105 = 93$$
$$6\times 31 \bmod 105 = 81$$
$$13\times 31 \bmod 105 = 88$$
$$27\times 31 \bmod 105 = 102$$
$$52\times 31 \bmod 105 = 37$$

recordar:
mcd(31, 105) = 1

- La mochila será: {62, 93, 81, 88, 102, 37}
- Secuencia super-incremental mochila: llave privada
- Secuencia normal mochila: llave pública

Lámina 100 Roberto Gómez C.

Encriptando

- Dividir los datos a encriptar en bloques iguales al número de objetos en la secuencia de la mochila
- Calcular el peso total de la mochila, recordar:
  - 1: el objeto esta presente
  - 0: el objeto esta ausente
- Se debe calcular un peso por cada bloque
- Ejemplo:

recordar, mochila: {62, 93, 81, 88, 102, 37}

Mensaje: 011000110101101110
Dividir: 011000 110101 101110

011000 corresponde a 93+81 = 174
110101 corresponde a 62+93+88+37 = 280
101110 corresponde a 62+81+88+102 = 333

Criptosistema: 174,280,333

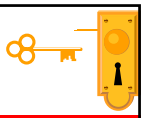
Lámina 101 Roberto Gómez C.

Decriptando

- Destinatario cuenta con llave privada: la secuencia super-incremental original, así como *m* y *n*, usados para transformarlos en una mochila normal
- Para decriptar el mensaje
  - determinar n^{-1} tal que $n(n^{-1}) = 1$
  - multiplicar cada uno de los valores del criptograma por n^{-1} *mod m*
  - descomponer los resultados para obtener el mensaje original.

Lámina 102 Roberto Gómez C.

Ejemplo decripción



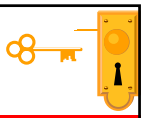
- Llave privada:
  - secuencia super-incremental: *{2,3,6,13,27,52}*
  - *m = 105*
  - *n = 31*
- Criptograma: *174, 280, 333*
- Calculando: $n^{-1} = inv(31,105) = 61$
  - criptograma debe ser multiplicado por *61 mod 105*

$$174 \times 61 \bmod 105 = 9 = 3 + 6, \quad \text{corresponde a } 011000$$
$$280 \times 61 \bmod 105 = 70 = 2 + 3 + 13 + 52, \quad \text{corresponde a } 110101$$
$$33 \times 61 \bmod 105 = 48 = 2 + 6 + 13 + 27, \quad \text{corresponde a } 101110$$

- Mensaje original: 011000 110101 10110

Lámina 103 Roberto Gómez C.

Implementación práctica



- En el mundo real las mochilas contienen al menos 250 objetos
- El valor de cada termino en la mochila de super-incremento debe ser entre 200 y 400 bits
  - el módulo debe contar con una longitud entre 100 y 200 bits
- Implementaciones reales del algoritmo usan generadores pseudo-aleatorios para producir estos valores.
- Con los valores anteriores no es práctico intentar resolverlos por fuerza bruta
  - si una computadora puede intentar millones de posibilidades por segundo, el probar todos los valores tomaría 10^{46} años

Lámina 104 Roberto Gómez C.

Seguridad del algoritmo

- No se necesito un millón de máquinas para romper el criptosistema
- Primero se recupero un bit de texto claro
- Shamir descubrió que puede ser roto bajo ciertas circunstancias
- Shamir y Zippel encontraron defectos en la transformación que permiten reconstruir la mochila super-incremental a partir de la mochila normal
  - en conferencia donde se presentó lo anterior, el ataque se demostró usando una computadora Apple II

Lámina 105 Roberto Gómez C.

Otros algoritmos de llave pública

- Pohling Hellman
  - similar RSA, pero llaves de encripción y decripción no son públicas
- McEliece
  - basado en una clase de códigos corrección de errores conocida como Códigos Goppa
- LUC
  - investigadores Nueva Zelanda, generalización RSA que usa varios polinomios de permutación en lugar de exponenciación
- Criptosistemas de llave pública de automatas finitos
  - desarrollado por investigador chino Tao Renji
  - es tan difícil factorizar la composición de dos autómatas finitos, como factorizar dos números primos grandes

Lámina 106 Roberto Gómez C.

Curvas elípticas

- Neal Koblitz y Miller proponen para usar curvas elípticas para criptosistemas llave pública
- No inventan un nuevo criptosistema
- Implementación de llave pública usando curvas elípticas: Diffie-Hellman, ElGamal, etc.
  - proporcionar mecanismos para construir "elementos" y "reglas para combinar" que producen grupos de combinación
  - grupos cuentan con propiedades familiares para construir algoritmos

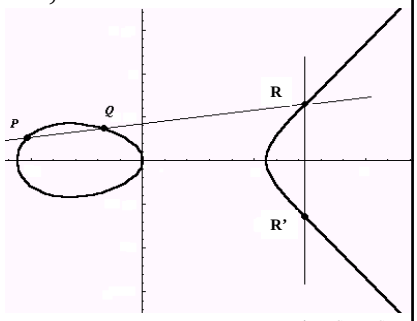


Lámina 107 Roberto Gómez C.

Algoritmos de intercambio de llaves

- Protocolo de estación-estación
- Protocolo de tres pasos de Shamir
- COMSET
- Encrypted Key Exchange
- Fortified Key Negotiation
- Conference Key Distribution and Secret Broadcasting

Lámina 108 Roberto Gómez C.

Sistemas Hibridos

- Un algoritmo simétrico con una llave de sesión aleatoria es usada para encriptar un mensaje.
- Un algoritmo de llave pública es usado para encriptar la llave de sesión aleatoria.

Lámina 109 Roberto Gómez C.

Encripción sistema hibrido

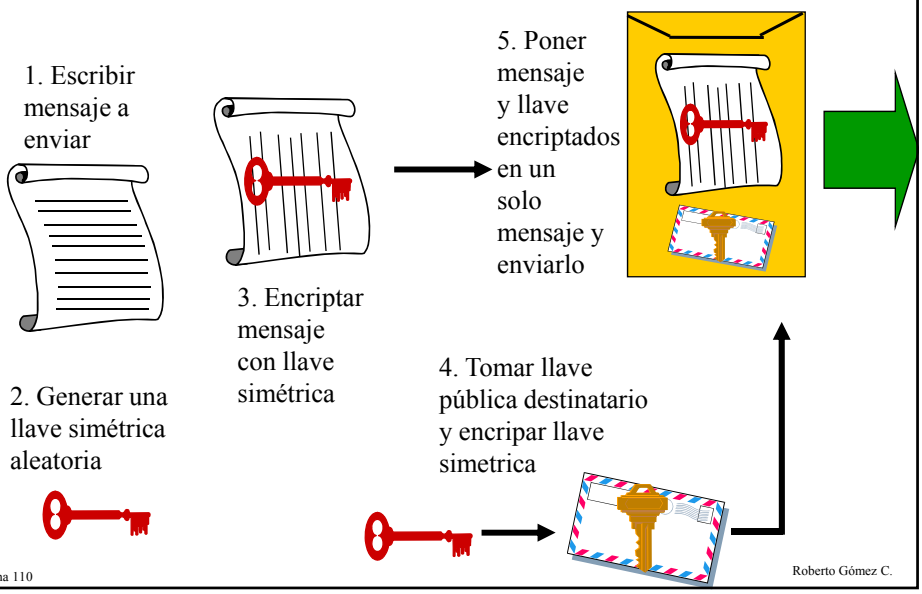
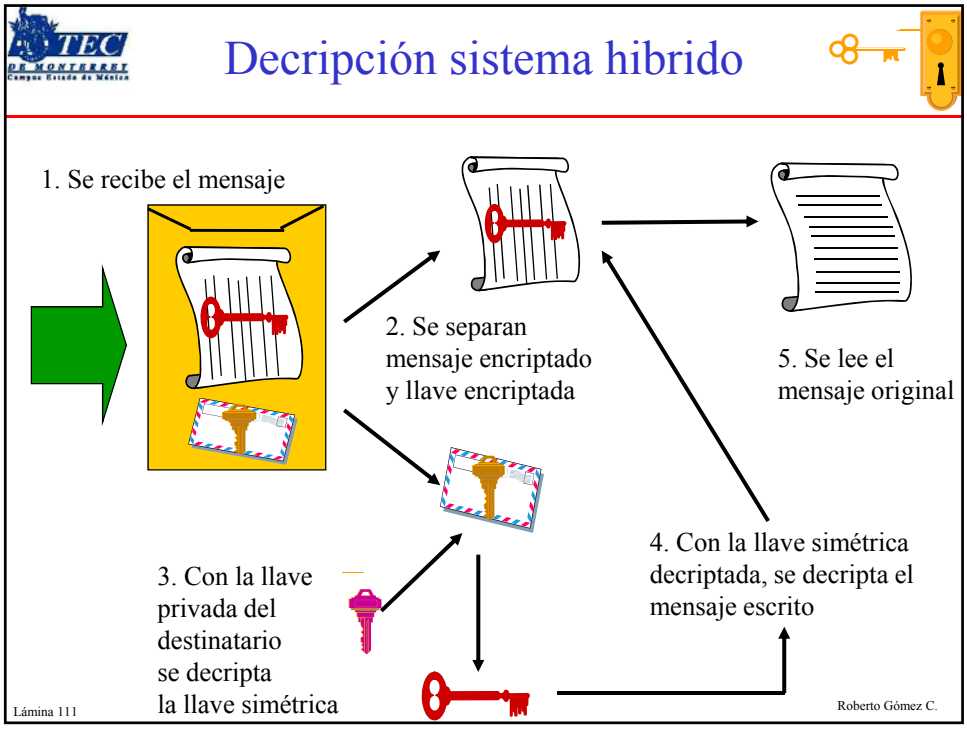
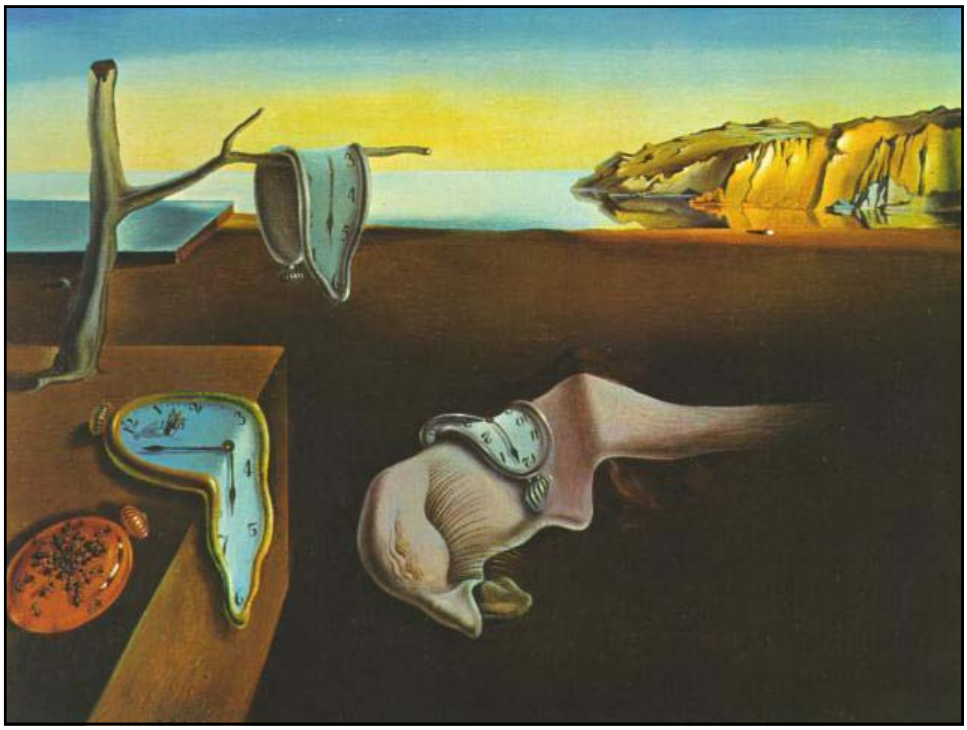
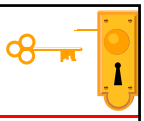


Lámina 110 Roberto Gómez C.


Decripción sistema hibrido
1. Se recibe el mensaje
2. Se separan mensaje encriptado y llave encriptada
3. Con la llave privada del destinatario se decripta la llave simétrica
4. Con la llave simétrica decriptada, se decripta el mensaje escrito
5. Se lee el mensaje original
Lámina 111
Roberto Gómez C.



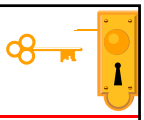
Posibles ataques en una comunicación



- Descubrimiento (disclosure)
  - dar a conocer el contenido mensaje a una persona o procesos que no poesee la llave apropiada.
- Analisis de tráfico
  - descubrimiento patrón tráfico en una red.
- Mascarada (masquerade)
  - inserción de mensajes en una red por parte de una parte fraudolenta.
- Alteración contenido
  - cambiar el contenido de un mensaje

Lámina 113 Roberto Gómez C.

Posibles ataques en una comunicación



- Modificación secuencia
  - cualquier modificación a la secuencia de mensajes entre las partes involucradas, (inserción, borrado, modificación o transposición)
- Alteración tiempo
  - retardo o reenvío de mensajes
- Repudiación
  - negación del envío de un mensaje por parte del emisor o negación de la recepción de un mensaje por parte del emisor

Lámina 114 Roberto Gómez C.

Funciones de autenticación

- Encripción de mensajes
  - el criptograma del mensaje entero sirve como su autenticador.
- Funciones hash
  - una función pública que mapea el mensaje de cualquier tamaño en un valor hash de tamaño fijo, el cual sirve de autenticador.
- Códigos de autenticación de mensajes
  - una función pública del mensjae y una llave secreta que produce un valor de longitud variable que sirve de autenticador

Lámina 115 Roberto Gómez C.

Firmas, huellas y MACs

características y usos

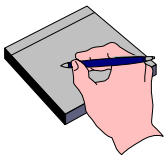
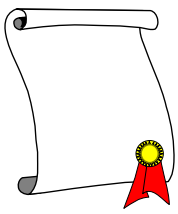




Lámina 116 Roberto Gómez C.

Un esquema de autenticación

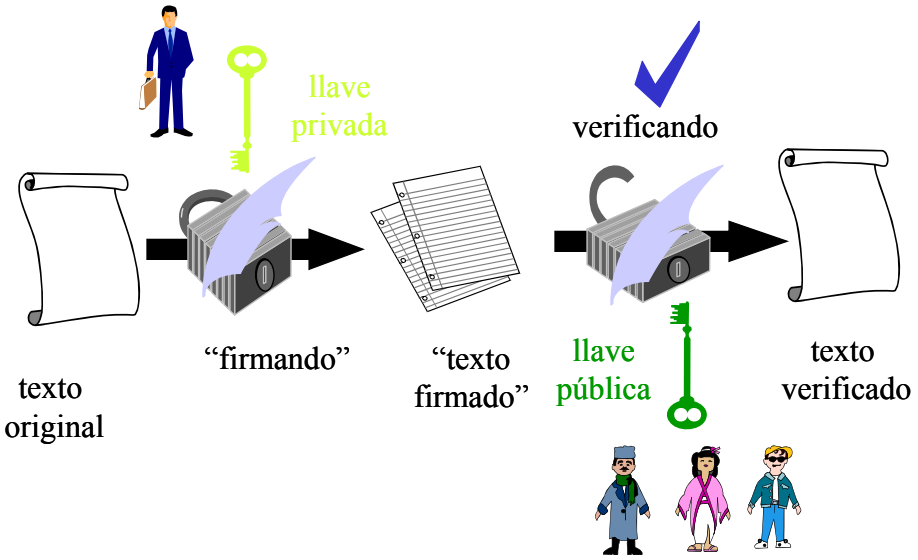


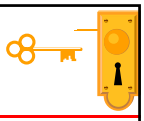
Lámina 117 Roberto Gómez C.

Las funciones hash

- Sistema anterior es lento y produce gran cantidad de información
- Mejoramiento: añadir una one-way hash function
  - función toma una variable de tamaño variable (cientos o miles de bits) y una salida de tamaño fijo (p.e. 160 bits)
- Función asegura que, si la información es cambiada (aún en sólo un bit) un valor completamente diferente es producido

Lámina 118 Roberto Gómez C.

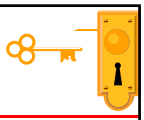
Características hash



- Libre de colisión:
  - difícil de generar dos entradas que generen la misma salida
- La salida es única
  - asegura que, si la información es cambiada (aún en sólo un bit) un valor completamente diferente es producido
- Es pública
  - no hay secretos en el proceso, la seguridad radica en su único sentido y en el hecho de que la salida no depende de la entrada

Lámina 119 Roberto Gómez C.

Ejemplo función hash



- Un ejemplo simple es tomar una entrada y y regresar un byte que consista de XOR de todos los bytes de entrada
- Función de un solo sentido: fácil de calcular un valor de hash de la entrada, pero difícil generar una entrada que corresponda a una salida

Lámina 120 Roberto Gómez C.

Código ejemplo función hash

- Función simple que utiliza el XOR de cada bloque de 128 bits de un archivo.
- Se presenta un ejemplo del código y de la ejecución de este.
- Fuente:
  - Dr. Dobb's Journal, Sep 1991, pp. 149

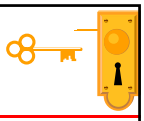
Lámina 121 Roberto Gómez C.

Ejemplo función simple

	Bit 1	Bit 2	•••	Bit n
Bloque 1	b_{11}	b_{21}		b_{n1}
Bloque 2	b_{12}	b_{22}		b_{n2}
	⋮	⋮	⋮	⋮
Bloque m	b_{1m}	b_{2m}		b_{nm}
Código hash	C_1	C_2		C_n

Lámina 122 Roberto Gómez C.

Ejemplo código función hash

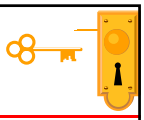


```
main(int argc, char *argv[])
{
   unsigned long hash[4] = {0, 0, 0, 0}, data[4];
   FILE *fp;      int i;

   if ((fp = fopen(argv[1], "rb")) != NULL) {
     while ( fread(data, 4, 4, fp) != NULL)
       for (i=0; i<4; i++)
         hash[i] ^= data[i];
      fclose(fp);
      for (i=0; i<4; i++)
        printf("%08lx",hash[i]);
      printf("\n");
   }
}
```

Lámina 123 Roberto Gómez C.

Salida del ejemplo código función hash



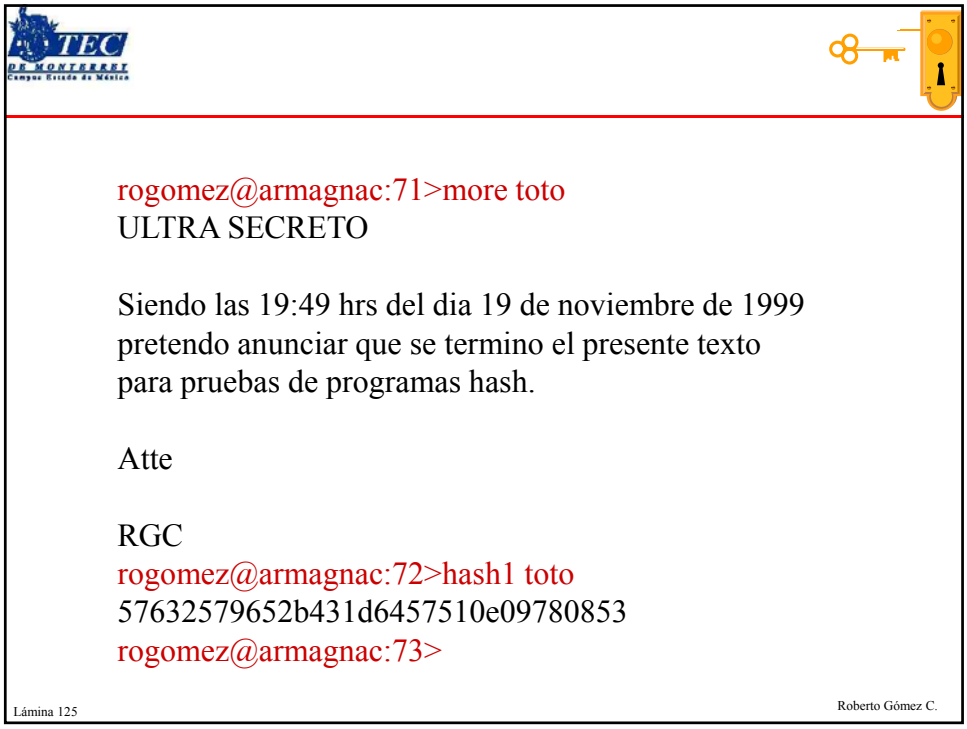
```
rogomez@armagnac:68>gcc hash1.c -o hash1
rogomez@armagnac:69>more toto
ULTRA SECRETO

Siendo las 19:49 hrs del dia 19 de noviembre de 1999
pretendo anunciar que se termino el presente texto
para pruebas de programas hash.

Atte;

RGC
rogomez@armagnac:70>hash1 toto
0f7621300e2b431d6457510e09780853
rogomez@armagnac:71>
```

Lámina 124 Roberto Gómez C.



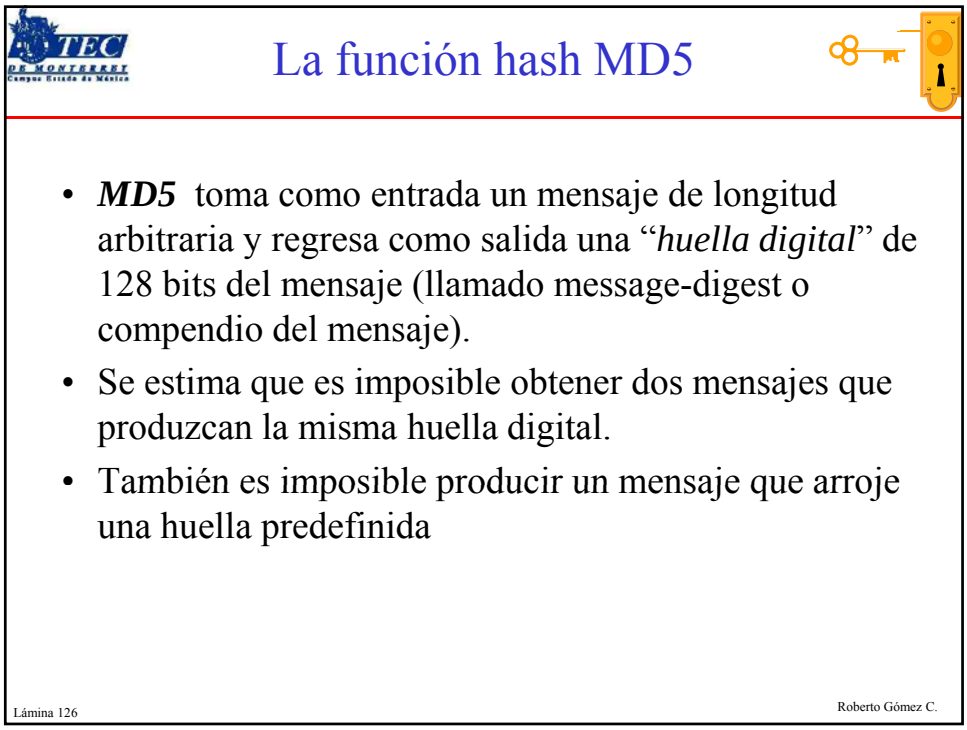
rogomez@armagnac:71>more toto
ULTRA SECRETO

Siendo las 19:49 hrs del dia 19 de noviembre de 1999
pretendo anunciar que se termino el presente texto
para pruebas de programas hash.

Atte

RGC
rogomez@armagnac:72>hash1 toto
57632579652b431d6457510e09780853
rogomez@armagnac:73>

Lámina 125 Roberto Gómez C.

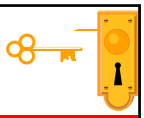


La función hash MD5

- ***MD5*** toma como entrada un mensaje de longitud arbitraria y regresa como salida una "*huella digital*" de 128 bits del mensaje (llamado message-digest o compendio del mensaje).
- Se estima que es imposible obtener dos mensajes que produzcan la misma huella digital.
- También es imposible producir un mensaje que arroje una huella predefinida

Lámina 126 Roberto Gómez C.

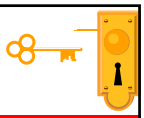
Descripción del algoritmo



- El mensaje de entrada puede tener cualquier longitud, no necesariamente debe ser múltiplo de 8.
- Los pasos que sigue el algoritmo son:
  - **Paso 1.** Agregado de bits de relleno (*Padding*).
  - **Paso 2.** Agregado de la longitud.
  - **Paso 3.** Inicialización del buffer del MD
  - **Paso 4.** Procesamiento del mensaje en bloques de 16 palabras.
  - **Paso 5.** Compendio del mensaje.

Lámina 127 Roberto Gómez C.

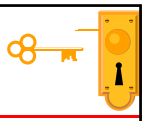
Paso 1: bits de relleno



- El mensaje es extendido de tal forma de que sea casi múltiplo de 512 bits de longitud.
- Casi porque se reservarán 64 bits.
- Estos 64 bits serán cubiertos con el tamaño del mensaje (expresado en 64 bits).

Lámina 128 Roberto Gómez C.

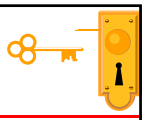
Paso 2: longitud mensaje



- El tamaño del mensaje es agregado al final del mensaje resultante del paso 1 (64 bits).
- En el caso de que la longitud del mensaje requiera más de 64 bits, sólo los 64 bits menos significativos se tomarían en cuenta.
- Como resultado de este paso se tiene un mensaje cuya longitud es múltiplo de 512.

Lámina 129 Roberto Gómez C.

Paso 3: Inicialización buffer



- Un buffer de 4 palabras es inicializado de la forma siguiente:

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

- Al final el buffer sea la huella digital.

Lámina 130 Roberto Gómez C.

Paso 4: procesamiento mensaje

- Función compresión consiste de cuatro iteraciones de procesamiento (HMD5)
- Cuatro funciones con similar estructura
  - cada uno usa una función diferente
- Cada iteración toma como entrada
  - bloque de 512 bits
  - un buffer de 128 bits
- Cada iteración usa
  - tabal de 64 elementos T[1…64] construida a partir de la función seno
- La salida de cada cuarta iteración se añade a la primera iteración.

Lámina 131 Roberto Gómez C.

Diagrama general hash

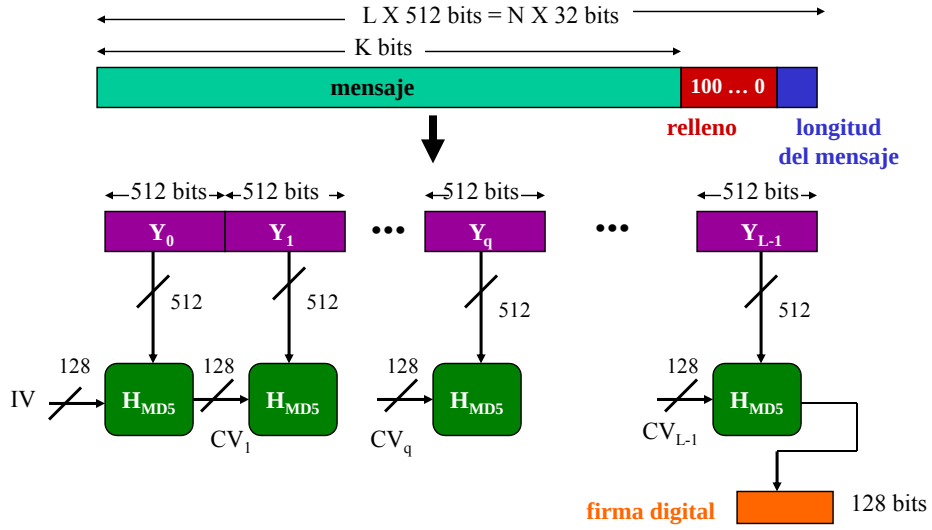


Lámina 132 Roberto Gómez C.

Diagrama procesamiento bloque 512 bits

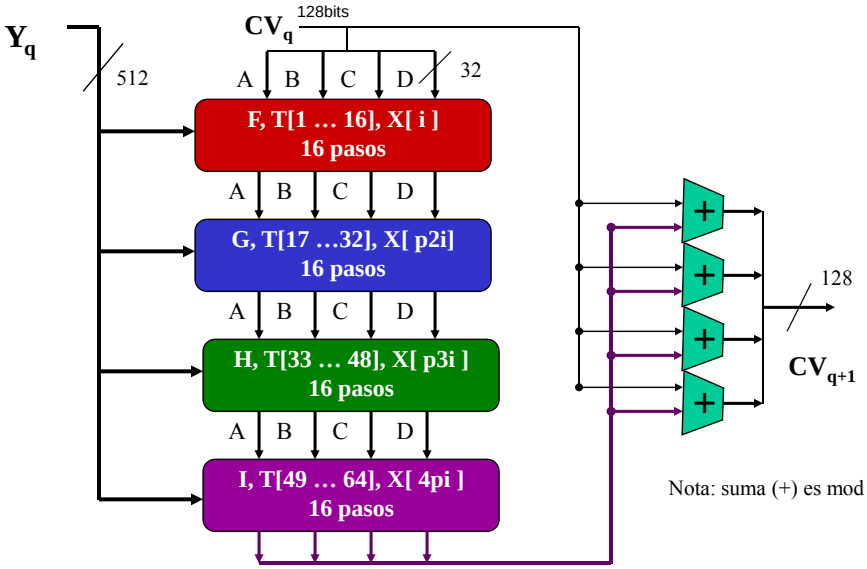
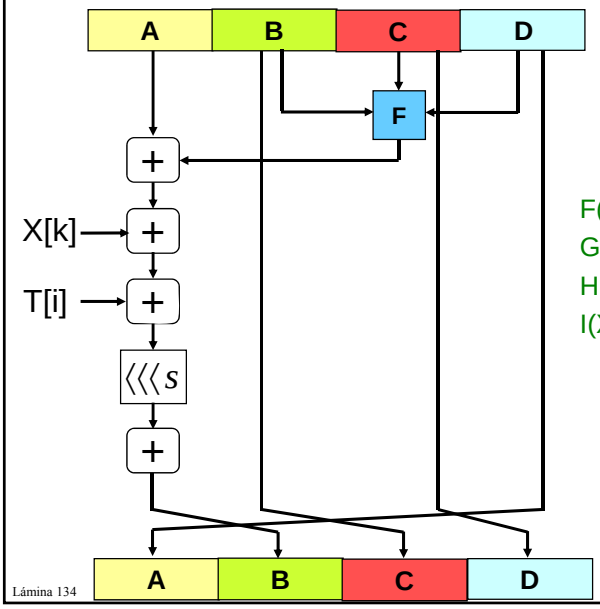


Lámina 133 Roberto Gómez C.

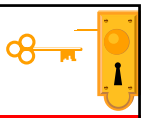
Funciones F,G,H e I



F(X,Y,Z) = XY v not (X) Z
G(X,Y,Z) = XZ v Y not (Z)
H(X,Y,Z) = X xor Y xor Z
I(X,Y,Z) = Y xor (X v not (Z))

Lámina 134 Roberto Gómez C.

Salida de MD5



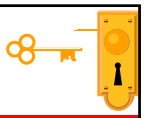
rogomez@armagnac:464>more toto
ULTRA SECRETO

Siendo las 19:49 hrs del dia 19 de noviembre de 1999
pretendo anunciar que se termino el presente texto
para pruebas de programas hash.

Atte;

RGC
rogomez@armagnac:465>md5 toto
MD5 (toto) = 0c60ce6e67d01607e8232bec1336cbf3
rogomez@armagnac:466>

Lámina 135 Roberto Gómez C.



rogomez@armagnac:467>more toto
ULTRA SECRETO

Siendo las 19:49 hrs del dia 19 de noviembre de 1999
pretendo anunciar que se termino el presente texto
para pruebas de programas hash.

Atte

RGC
rogomez@armagnac:468>md5 toto
MD5 (toto) = 30a6851f7b8088f45814b9e5b47774da
rogomez@armagnac:469>

Lámina 136 Roberto Gómez C.

Otra función hash SHA-1

- Desarrollado por el National Institute of Standards and Technology (NIST) y publicado como una FIPS.
- Toma un mensaje de entrada con una longitud máxima de 2^{64} bits y produce una salida de 160 bits.
- Entrada es procesada en bloques de 512 bits.
- Pasos que sigue:
  - Añadir bits de relleno (padding bits).
  - Añadir la longitud.
  - Inicializar el buffer de 160 bits MD
  - Procesar el mensaje en bloques de 512 palabras
  - Imprimir la salida

Lámina 137 Roberto Gómez C.

Buffer inicial

- Buffer inicial

palabra A: 67452301
palabra B: EFCDAB89
palabra C: 98BADCFE
palabra D: 10325476
palabra E: C3D2E1F0

Lámina 138 Roberto Gómez C.

Diagrama general hash SHA1

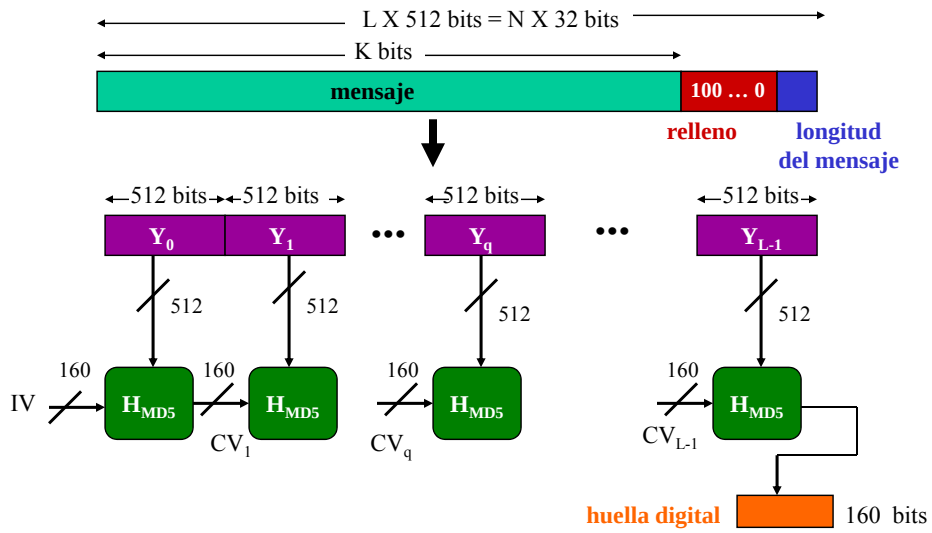


Lámina 139 Roberto Gómez C.

SHA-1 Compression Function

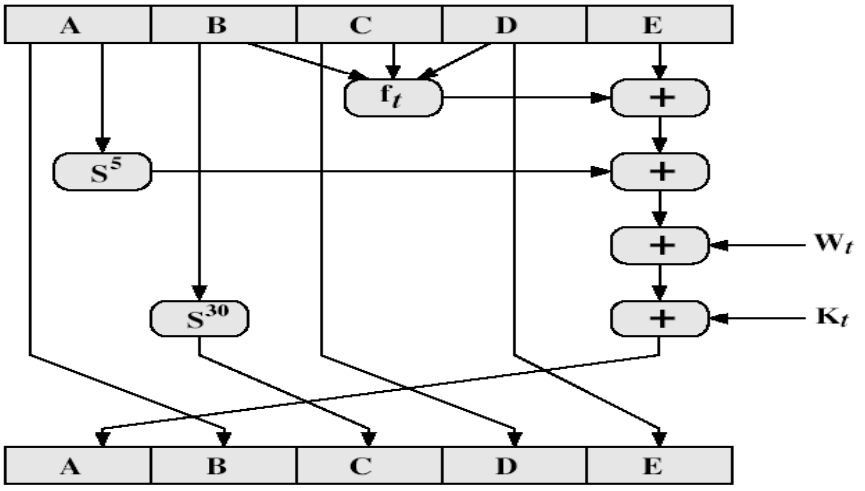


Lámina 140 Roberto Gómez C.

La función hash SHA-1

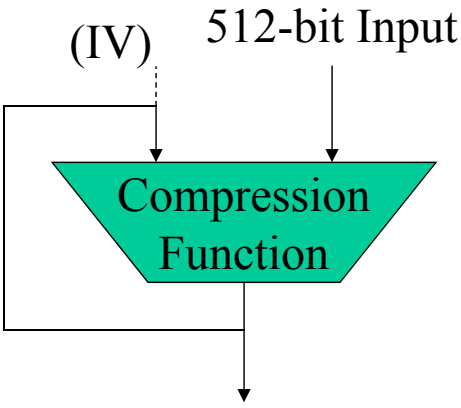


Lámina 141 Roberto Gómez C.

La función hash SHA-1

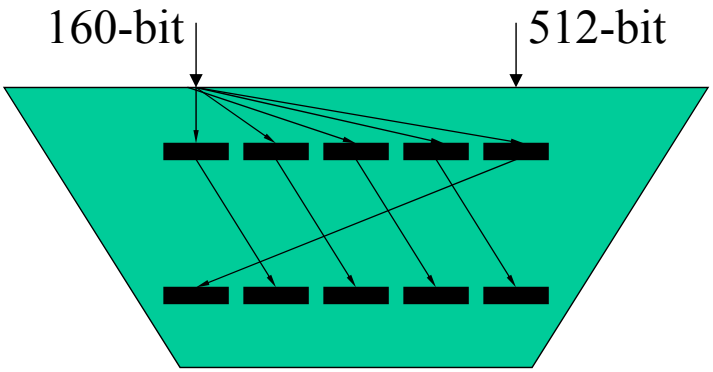
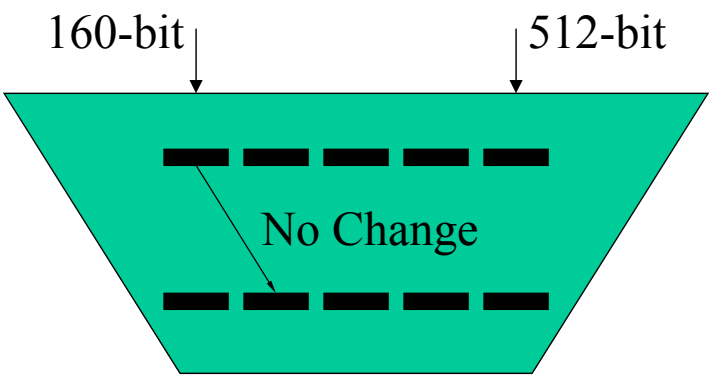


Lámina 142 Roberto Gómez C.

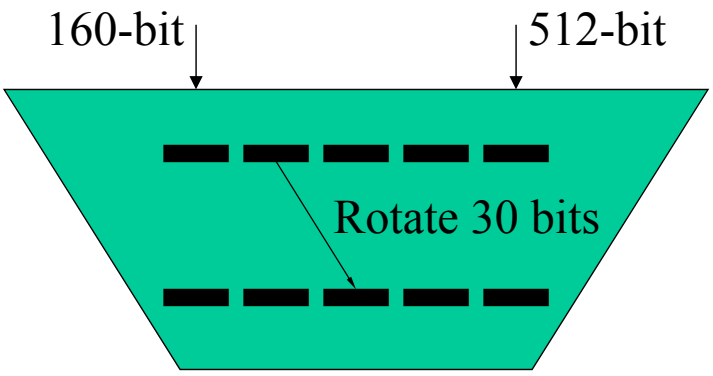
La función hash SHA-1



One of 80 rounds

Lámina 143 Roberto Gómez C.

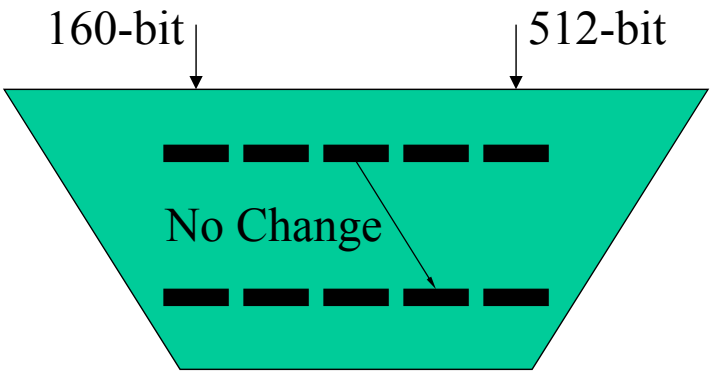
La función hash SHA-1



One of 80 rounds

Lámina 144 Roberto Gómez C.

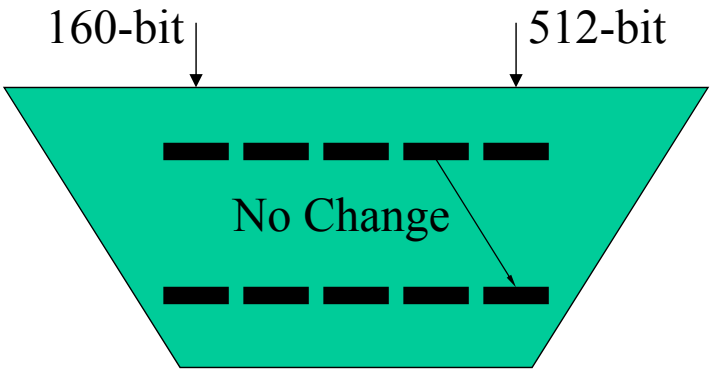
La función hash SHA-1



One of 80 rounds

Lámina 145 Roberto Gómez C.

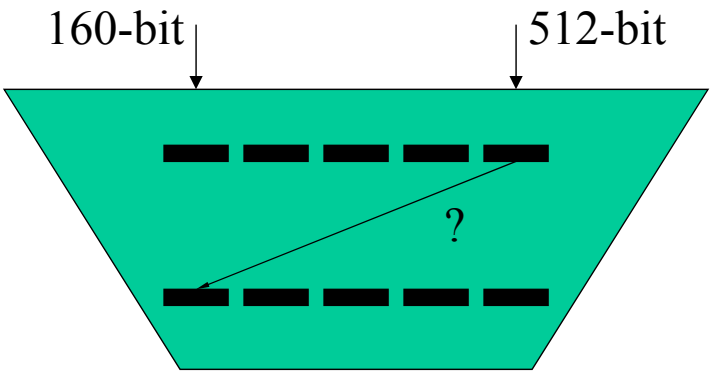
La función hash SHA-1



One of 80 rounds

Lámina 146 Roberto Gómez C.

La función hash SHA-1



One of 80 rounds

Lámina 147 Roberto Gómez C.

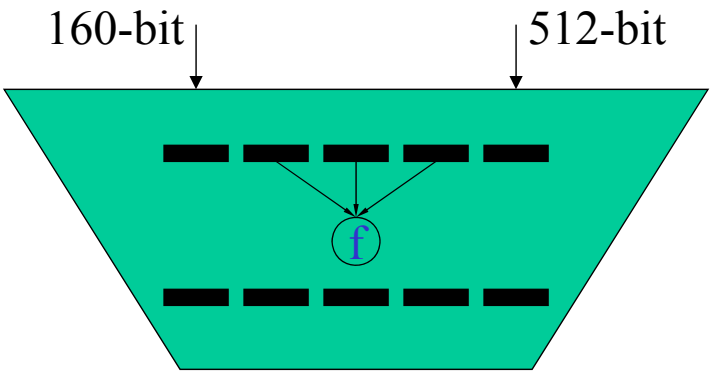
La función hash SHA-1

En que consiste la última transformación de 32 bits?

- Tomar la palabra más a la derecha.
- Añadir en la palabra más a la izquierda rotada 5 bits.
- Añadir en una función f dependiente de la vuelta de las tres palabras de enmedio.

Lámina 148 Roberto Gómez C.

La función hash SHA-1



One of 80 rounds

Lámina 149 Roberto Gómez C.

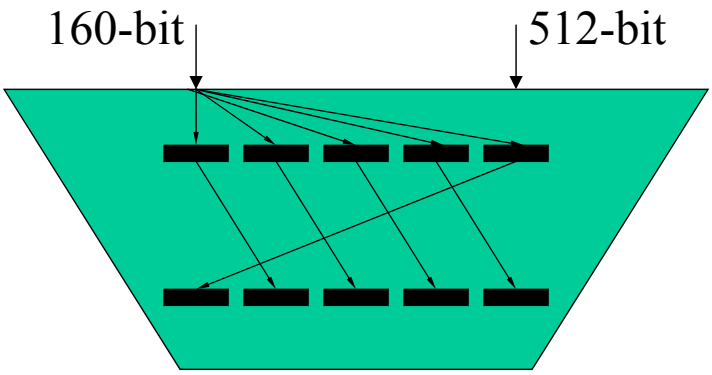
La función hash SHA-1

Dependiendo de la vuelta, la funcion no lineal f es una de las siguientes:

$$f(X,Y,Z) = (X \wedge Y) \vee ((\neg X) \wedge Z)$$
$$f(X,Y,Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$$
$$f(X,Y,Z) = X \oplus Y \oplus Z$$

Lámina 150 Roberto Gómez C.

La función hash SHA-1



One of 80 rounds

Ejemplo SHA-1: http://nsfsecurity.pr.erau.edu/crypto/sha1.html

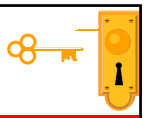
Lámina 151 Roberto Gómez C.

Revised Secure Hash Standard

- NIST revisa el FIPS 180-2
- Como resultado añade 3 algoritmos mas:
  - SHA-256, SHA-384, SHA-512
- Diseñados para compatibilidad con mayor seguridad proporcinada por el criptosistema AES.
- Estructura y detalles son similares a SHA-1
  - análisis es similar a lo presentado

Lámina 152 Roberto Gómez C.

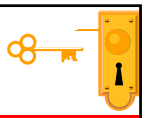
Comparación MD5 y SHA-1



- Seguridad en contra ataques fuerza bruta.
  - huella SHA es 32 bits más grande que la de MD5.
  - producir dos mensajes con la misma firma es de 2^{65} para MD5 y 2^{80} para SHA-1.
  - dificultad para producir cualquier mensaje teniendo una firma de mensaje es del orden de 2^{128} operaciones para MD5 y de 2^{160} para SHA-1.
- Seguridad en contra criptoanálisis
  - MD5 es vulnerable a ataques de criptoanalisis desde su diseño
  - Aparentemente SHA-1 no es suceptible a tales ataques

Lámina 153 Roberto Gómez C.

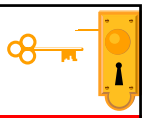
Comparación MD5 y SHA-1



  - Sin embargo se conoce muy poco acerca de los criterios de diseño de SHA-1, por lo ques es más difícil de juzgar que MD5.
- Velocidad
  - Debido operaciones modulo 2^{32} ambos trabajan bien en arquitecturas de 32 bits.
  - SHA-1 involucra más pasos (80 contra 64) y debe procesar un buffer de 160 bits en comparación con el buffer de 128 bits de MD5.
  - SHA-1 debe ser más lento que MD5 en el mismo hardware.

Lámina 154 Roberto Gómez C.

Comparación MD5 y SHA-1

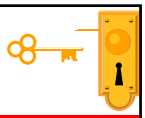


- Simplicidad y tamaño
  - ambos algoritmos son simples para describir y simples en su implementación
  - no requieren programas muy grandes o tablas de substitución
- Arquitectura little-endian versus big-endian
  - MD5 usa un esquema little-endian para interpretar un mensaje de una secuencia de 32 palabras
  - SHA-1 usa un esquema big-endian
  - esto no representa ninguna ventaja a ninguno de los dos

A = 01234567 (MD5) ⇒ 67452301 (SHA-1)
B = 89ABCDEF (MD5) ⇒ EFCDAB89 (SHA-1)
C = FEDCBA98 (MD5) ⇒ 98BADCFE (SHA-1)
D = 76543210 (MD5) ⇒ 10325476 (SHA-1)

Lámina 155 Roberto Gómez C.

Actualizaciones a SHA-1

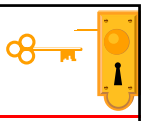


- Longitud de 160 bits no es suficiente para garantizar colisiones, o resistencia contra ataques de cumpleaños.
- Cuatro opciones
  - SHA-224
  - SHA-256
  - SHA-384
  - SHA-512

SHA-2

Lámina 156 Roberto Gómez C.

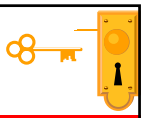
RIPEMD-160



- RIPEMD-160 fue desarrollado en Europa como parte del proyecto PIPE en 1996 por investigadores involucrados en ataques a MD4/5
  - inicialmente de 128 bits, pero propuestas iniciales de fortalecimiento al MD4/5 derivaron en RIPEMD-160
- En algún sentido similar a MD5/SHA
- Usa 2 líneas paralelas de 5 iteraciones de 16 pasos
- Crea un valor hash de 160 bits
- Más lento pero probablemente más seguro que SHA

Lámina 157 Roberto Gómez C.

RIPEMD-160 Overview



1. Rellenar mensaje para que longitud sea 448 mod 512
2. Añadir 64 bits de longitud del mensaje al final
3. Inicializar un buffer de 5 palabras (160 bits)

   A = 67452301 D = 10325476
   B = efcdab89 E = c3d2e1f0
   C = 98badcfe
4. Procesar mensajes en bloques de 16 palabras (512 bits
   - usa 10 iteraciones de 16 operaciones de bits en el bloque de mensaje y el buffer
     - en dos líneas paralelas de 5 iteraciones c/u
   - suma salida a entrada para formar un nuevo valor en el buffer
5. Valor de salida hash: valor que queda al final del buffer

Lámina 158 Roberto Gómez C.

Una iteración en RIPEMD-160

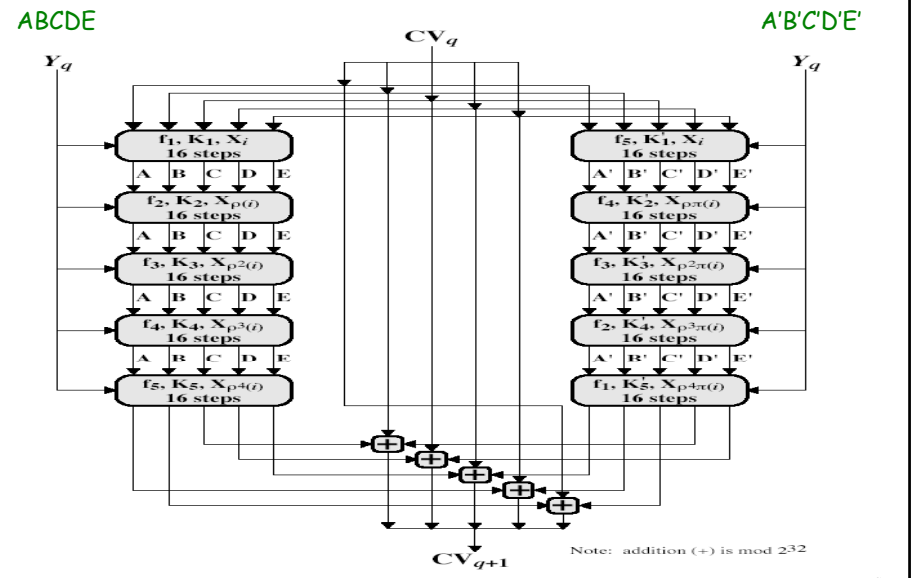


Lámina 159 Roberto Gómez C.

RIPEMD-160: función de compresión

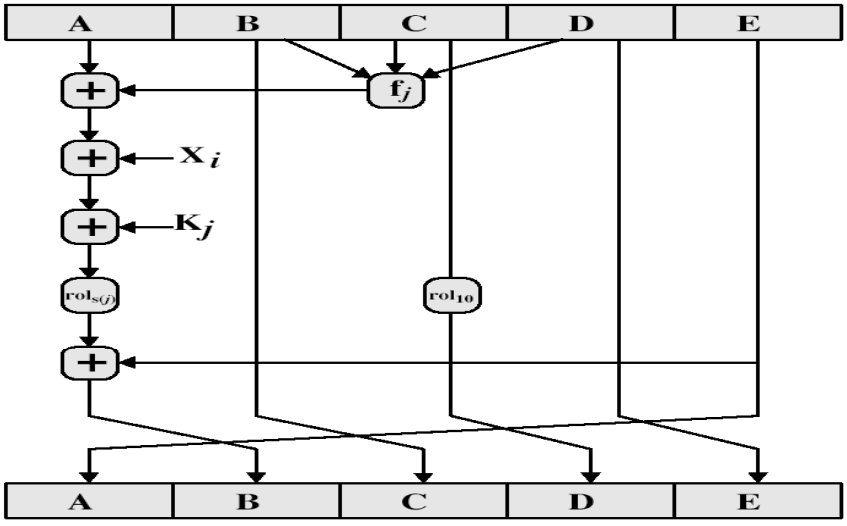


Lámina 160 Roberto Gómez C.

Funciones lógicas primitivas, fi

Nombre función	Valor función	Paso
$f_1 = f(j,B,C,D) = B \oplus C \oplus D$		$(0 \le j \le 15)$
$f_2 = f(j,B,C,D) = (B \wedge C) \vee (\overline{B} \wedge D)$		$(16 \le j \le 31)$
$f_3 = f(j,B,C,D) = (B \vee \overline{C}) \oplus D$		$(32 \le j \le 47)$
$f_4 = f(j,B,C,D) = (B \wedge D) \vee (C \wedge \overline{D})$		$(48 \le j \le 63)$
$f_5 = f(j,B,C,D) = B \oplus (C \vee \overline{D})$		$(64 \le j \le 79)$

Lámina 161 Roberto Gómez C.

RIPEMD-160: criterios diseño

- Usar dos líneas paralelas de 5 iteraciones para incrementar la complejidad
- Por simplicidad las dos líneas son muy similares
- Las operaciones de mezcla en cada paso son muy cercanas al de MD5
- Se permutan varias partes del mensaje
- Se usan shifts circulares para obtener mejores resultados

Lámina 162 Roberto Gómez C.

RIPEMD-160 versis MD5 & SHA-1

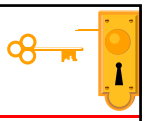
- Ataque de fuerza bruta más duro
  - 160 bits como SHA-1 vs 128 bits para MD5)
- No es vulnerable a ataques conocidos
  - como SHA-1 aunque más fuerte (comparado con MD4/5)
- Más lento que MD5 (más pasos)
- Todo el diseño es simple y compacto
- SHA-1 optimizado para CPUs tipo big endian
  - RIPEMD-160 & MD5 optimizados para CPU's little endian

Lámina 163 Roberto Gómez C.

Uso huellas para cuidar integridad

Lámina 164 Gómez C.

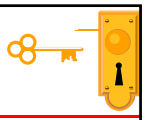
Otras funciones hash de un solo sentido



- Algoritmo MD2
- Algoritmo MD4
- RIPEMD-160
- HMAC
- N-Hash
- Haval
- Panama
- Tiger
- Vest
- Whirlpool

Lámina 165 Roberto Gómez C.

Comparativo funciones hash



Algoritmo	Tamaño salida	Internal state size	Tamaño bloque	Length size	Tamaño palabra
HAVAL	256/224/192/160/128	256	1024	64	32
MD2	128	384	128	No	8
MD4	128	128	512	64	32
MD5	128	128	512	64	32
PANAMA	256	8736	256	No	32
RIPEMD	128	128	512	64	32
RIPEMD-128/256	128/256	128/256	512	64	32
RIPEMD-160/320	160/320	160/320	512	64	32
SHA-0	160	160	512	64	32
SHA-1	160	160	512	64	32
SHA-256/224	256/224	256	512	64	32
SHA-512/384	512/384	5122	1024	128	64

Lámina 166 Roberto Gómez C.

Comparativo funciones hash

Algoritmo	Tamaño salida	Internal state size	Tamaño bloque	Length size	Tamaño palabra
TIGER(2)-192/160/128	192/160/128	192	512	64	64
VEST-4/8 (hash mode)	160/256	176/304	8	80	1
VEST-16/32 (hash mode)	320/512	424/680	8	88	1
WHIRLPOOL	512	512	512	256	8

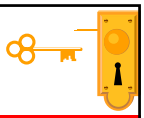
Lámina 167 Roberto Gómez C.

La huella digital

- La salida producida por una función hash aplicada a un documento, es conocida con el nombre de huella digital de dicho documento
- Cualquier cambio en el documento produce una huella diferente
- Huella digital también es conocida como compendio de mensaje (cuando el documento es un mensaje)
- En México:
  - Funciones de Digestión
  - resultado de la función: el digesto

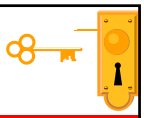
Lámina 168 Roberto Gómez C.

Características de una firma manuscrita



- La firma es autentica
  - el que firmante deliberadamente firmo el documento
- La firma es inolvidable
  - es prueba de que el firmante y no otra persona, deliberadamente firmo el documento
- La firma no es reutilizable
  - firma es parte del documento y otra persona no puede moverlo a otro documento

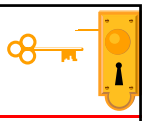
Lámina 169 Roberto Gómez C.



- El documento firmado es inalterable
  - después de que el documento fue firmado, no puede ser alterado
- La firma no puede ser repudiada
  - firma y documento son cosas físicas
  - el firmante no puede argumentar que el o ella no firmaron

Lámina 170 Roberto Gómez C.

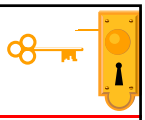
Problemas firma computacional



- No es posible aplicar lo anterior a computación directamente
- Archivos son fáciles de copiar
- Una imagen se puede cortar y pegar en otro documento
- Los archivos son fáciles de modificar una vez que son firmados, sin dejar evidencia de modificación

Lámina 171 Roberto Gómez C.

La firma digital



- Permiten al receptor verificar:
  - la autenticidad del origen de la información
  - que la información esta intacta (integridad)
  - no-repudiación: que el emisor argumente que no envío la información
- Tiene mismo propósito firma escrita
- Ventaja: no puede ser falsificada tan fácilmente como la escrita

Lámina 172 Roberto Gómez C.

Firmas digitales y huellas digitales

- Es posible usar la huella y la llave privada para producir una firma
- Se transmite el documento y la firma juntos
- Cuando el mensaje es recibido, el receptor utiliza la función hash para recalcular la huella y verificar la firma
- Es posible encriptar el documento si así se desea

Lámina 173 Roberto Gómez C.

Firma digital segura (envío)

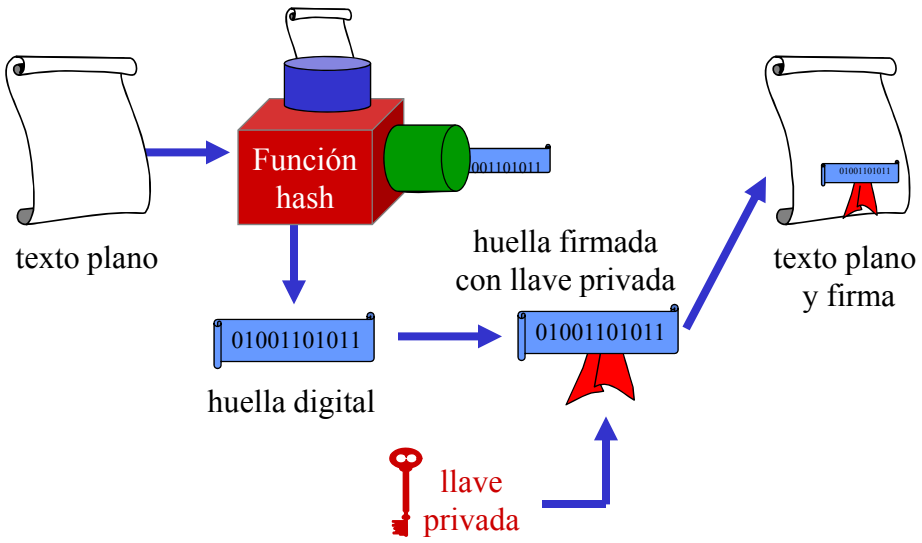
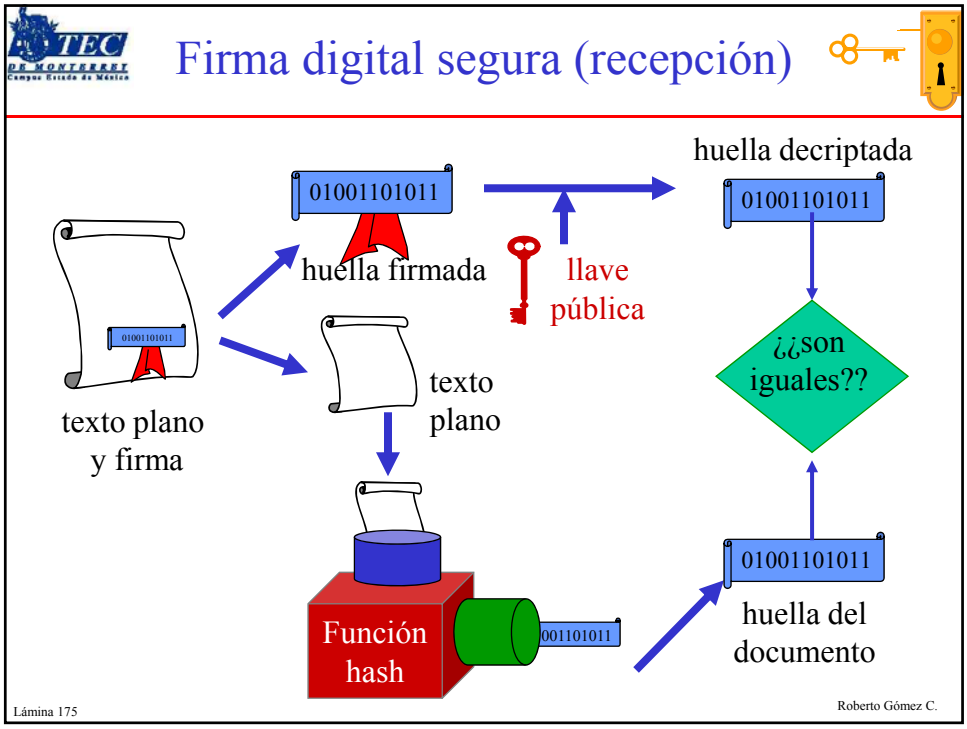
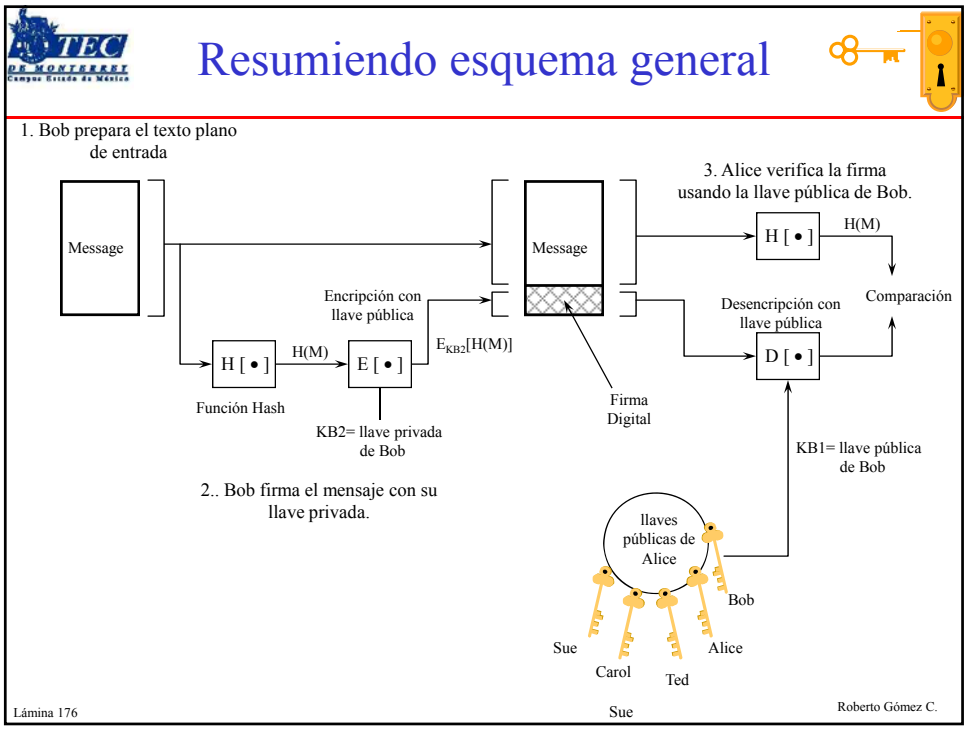


Lámina 174 Roberto Gómez C.


Firma digital segura (recepción)
huella decriptada
01001101011
01001101011
huella firmada
llave
pública
¿¿son
iguales??
texto plano
y firma
texto
plano
Función
hash
001101011
01001101011
huella del
documento
Lámina 175
Roberto Gómez C.


Resumiendo esquema general
1. Bob prepara el texto plano
de entrada
Message
Message
3. Alice verifica la firma
usando la llave pública de Bob.
H [•]
H(M)
Encripción con
llave pública
Desencripción con
llave pública
Comparación
H [•]
H(M)
E [•]
$E_{KB2}[H(M)]$
D [•]
Función Hash
KB2= llave privada
de Bob
Firma
Digital
KB1= llave pública
de Bob
2.. Bob firma el mensaje con su
llave privada.
llaves
públicas de
Alice
Bob
Sue
Carol
Ted
Alice
Sue
Lámina 176
Roberto Gómez C.

Estándares firmas digitales

- Existen tres algoritmos aprovados como FIPS para producir una firma digital

1. Digital Signature Algorithm (DSA)
2. RSA (ANSI X9.31) y
3. Elliptic Curve DSA (ECDSA -ANSI X9.62).

Lámina 177 Roberto Gómez C.

Ejemplo de firma digital: RSA

Clave Pública (n_A, e_A) Clave Privada (d_A)


Alicia

Algoritmo:

Rúbrica: $r_A H(M) = H(M)^{dA} \bmod n_A$

A envía el mensaje M en claro (o cifrado) al destinatario B junto a la rúbrica: $\{M, r_A H(M)\}$


Beto

El destinatario B tiene la clave pública e_A,n_A de A y descifra $r_A H(M) \Rightarrow \{(H(M)^{dA})^{eA} \bmod n_A\}$ obteniendo así H(M). Como recibe el mensaje M', calcula la función hash H(M') y compara:

Si H(M') = H(M) se acepta la firma.

Lámina 178 Roberto Gómez C.

Ejemplo firma RSA: creando firma

Hola. Te envío el documento. Saludos, Beto.

Beto

Alicia

Sea H(M) = F3A9 (16 bits)

Claves Beto
$n_B = 65.669$
$e_B = 35$, $d_B = 53.771$

$2^{16} < 65.669 < 2^{17}$
Forzaremos firmar bloques de 16 bits

Claves Alicia
$n_A = 66.331$
$e_A = 25$, $d_A = 18.377$

Firma $H(M) = F3A9_{16} = 62.377_{10}$

$r_{H(M)} = H(M)^{dB} \bmod n_B$

$r_{H(M)} = 62.377^{53.771} \bmod 65.669 = 24.622$

Beto envía el par (M, r) = (M, 24.622)

Nota: los primos que usa Beto son 97, 677 y Alicia 113, 587

Lámina 179 Roberto Gómez C.

Ejemplo firma RSA: verificando firma

Beto

Claves Beto
$n_B = 65.669$
$e_B = 35$, $d_B = 53.771$

Claves Alicia
$n_A = 66.331$
$e_A = 25$, $d_A = 18.377$

Alicia

Teníamos que: $H(M) = F3A9_{16} = 62.377_{10}$

$r_{H(M)} = H(M)^{dB} \bmod n_B$ $r_{H(M)} = 62.377^{53.771} \bmod 65.669 = 24.622$

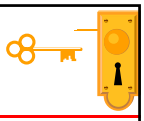
Beto había enviado el par (M, r) = (M, 24.622)

Alicia recibe un mensaje M' junto con una rúbrica r = 24.622:

- Calcula $r^{eB} \bmod n_B = 24.622^{35} \bmod 65.669 = 62.377$.
- Calcula el resumen de M' es decir H(M') y lo compara con H(M).
- Si los mensajes M y M' son iguales, entonces H(M) = H(M') y se acepta la firma como válida.
- NOTA: No obstante, H(M) = H(M') no implica que M = M'.

Lámina 180 Roberto Gómez C.

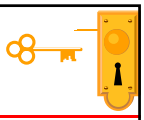
Firmando con ElGamal



- El Gamal también puede ser usado (con un pequeño cambio) para firmar mensajes
- Como en el esquema de encripción, la llave pública será: $y = g^x \bmod p$, junto con g y p
- Para firmar un mensaje:
  - Beto elige un número aleatorio k menor a $p-1$
  - Beto calcula: $a = g^k \bmod p$
  - Beto resuelve la ecuación $M = x*a + k*b \pmod{p-1}$ para b
  - La firma es el par a y b

Lámina 181 Roberto Gómez C.

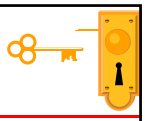
Verificando la firma



- Se recibe el mensaje M junto con la firma a y b
- Para verificar la firma
  - comprobar que $y^a a^b \bmod p = g^M \bmod p$
- Si son iguales la firma es válida
- En caso contario la forma es inválida

Lámina 182 Roberto Gómez C.

Ejemplo de firma con ElGamal



Firmando

p = 11, g = 2.
Beto elige x = 8.
$y = 2^8 \bmod 11 = 3$
Llave pública: y = 3, g = 2, p = 11
Beto quiere firmar M = 5
Elige k = 9 (mcd(9, 10) = 1)
$a = 2^9 \bmod 11 = 6$
Resuelve: 5 = 8*6 + 9 * b mod 10
b = 3
La firma es a = 6, b = 3

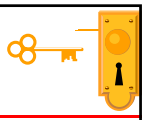
Verificando Firma

Se recibió mensaje 10
Con firma 6,3
Entonces se calcula:

$3^{6}6^{3} \bmod 11 = 2^5 \bmod 11$

Lámina 183 Roberto Gómez C.

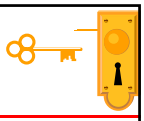
DSS: Digital Signature Standard



- Gobierno USA aprueba el esquema de firma FIPS 186
- Usa el algoritmo hash SHA
- Diseñado por el NIST y la NSA al comienzo de los 90s.
- DSS es el estándar y DSA es el algoritmo
- Referencia:
  - NIST FIPS 186 Digital Signature Standard (DSS)

Lámina 184 Roberto Gómez C.

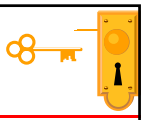
El esquema DSA



- DSA fue diseñado por la NIST y la NSA y el esquema federal de firma en USA (usado con el algoritmo de hash SHA)
- DSA es el algoritmo y DSS es el estándar
- DSA es una variante de los algoritmos ElGamal y Schnorr
- Crea una firma de 320 bits, pero con una seguridad de 512-1024 bits
- Se dio una considerable reacción por este anuncio
  - debate acerca de si RSA debio ser usado
  - debate acerca la provisión de un solo algoritmo de firma

Lámina 185 Roberto Gómez C.

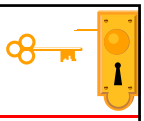
Reacciones anuncio DSA



- DSA no puede ser usado para encripción o distribución de llaves
- DSA fue desarrollado por la NSA y puede contar con un trapdoor en el algoritmo
- DSA es más lento que RSA
- RSA es un estándar de facto
- El proceso de selección de DSA no fue público
  - no se contó con suficiente tiempo para su análisis
- DSA puede infringir en otras patentes
- El tamaño de la llaves es muy chico.

Lámina 186 Roberto Gómez C.

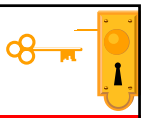
Algoritmo DSA



- Generación llaves en DSA
  - p de longitud es un número primo
    - número entre 512 y 1024 bits
  - q es un factor primo de 160 bits de p-1
  - $g = h^{(p-1)/q}$
    - donde h es cualquier número menor a p-1 con $h^{(p-1)/q} \bmod p > 1$
  - x es un número menor a q
  - $y = g^x \bmod p$
  - La llave pública es: (p,q,g,y)
  - La llave privada es: x

Lámina 187 Roberto Gómez C.

DSA: firmando y verifando



Proceso de Firman

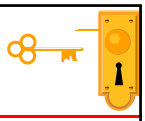
Firmar un mensaje M
Generar número random k
$k < q$
Calcular:
$r = (g^k \bmod p) \bmod q$
$s = k^{-1}*(SHA(M)+ x* r) \bmod q$
La firma es (r,s)

Proceso verificación firma

Verificar $(0<r<q)$ y $(0<s<q)$,
si no, inválido
$w = s^{-1} \bmod q$
$u1 = SHA(M)*w \bmod q$
$u2 = r*w \bmod q$
$v = (g^{u1} * y^{u2} \bmod p) \bmod q$
si v=r entonces
la firma es válida

Lámina 188 Roberto Gómez C.

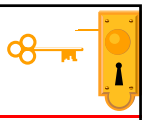
Ejemplo DSA: generación llaves



- Selección números primos: q= 101
  - p = 78 q + 1 = 7879
  - q divide a (p-1): (p-1)/q = 101
- Selección número aleatorio h = 3
  - 3 es un elemento primivito de Z_{7879}
- Se calcula
  - $g = h^{78} \bmod 7879 = 170$
- Selección número x=75 (75 < 101) y se calcula
  - $y = g^x \bmod p = 170^{75} \bmod 7879$
  - y = 4567
- La llave pública de A es: (p,q,g,y) = (7879,101,170,4567)
- La llave privada es: x = 75

Lámina 189 Roberto Gómez C.

Ejemplo de firma con DSA



p= 7879, q= 101, g= 170, y= 4567

Llave privada Beto x = 75.

Beto quiere firmar M = 22

Se genera número k =50

$k^{-1} \bmod 101 = 99$

Se calcula:

$r = (g^k \bmod p) \bmod q$

$r = (170^{50} \bmod \bmod 7879) \bmod 101$

$r = 2518 \bmod 94$

r= 94

$s = k^{-1}*(SHA(M)+ x* r) \pmod q$

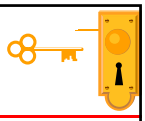
$s = 99* (22 + 75 * 94) \bmod 101$

s = 97

Firma mensaje M: (94, 97)

Lámina 190 Roberto Gómez C.

Ejemplo verificación firma DSA



LLave pública: p= 7879, q= 101, g= 170, y= 4567

Mensaje: 22

Firma: (r,s) ➔ (94 97)

Se cálcula:

$s = 97 \rightarrow w = s^{-1} = 97^{-1} \bmod 101 = 25$

u1= SHA(M)*w mod q

u1 = 22*25 mod 101 = 45

u2= r*w mod q

u2= 97*25 mod 101 = 27

$v = (g^{u1} * y^{u2} \bmod p) \bmod q$

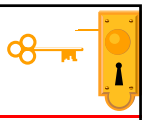
$v= (170^{45} * 4567^{27} \bmod 7879) \bmod 101$

v=94

94 = 94 ➔ v = r
→ la firma es válida

Lámina 191 Roberto Gómez C.

Comentarios DSA



- Originalmente habia una sugerencia para usar módulos comunes,
- Posible emplear tanto ElGamal como RSA como algoritmos de encripción usando rutinas DSA,
- DSA esta patentado como de uso libre
  - sin embargo la patente ha sido objetada, situación no clara
- Gus Simmons ha encontrado un canal subliminal en DSA, que puede ser usado para descubrir la llave privadas de una biblioteca
  - se recomienda estar seguros en la confianza de la implementación de la biblioteca

Lámina 192 Roberto Gómez C.

Algunos tiempos de cálculo

DSA Speeds for Different Modulus Lengths with a 160-bit Exponent (on a SPARC II)

	512 bits	768 bits	1024 bits
Sign	0.20 sec	0.43 sec	0.57 sec
Verify	0.35 sec	0.80 sec	1.27 sec

Comparison of RSA and DSA Computation Times

	DSA	RSA	DSA with Common *p, q, g*
Global Computations	Off-card (P)	N/A	Off-card (P)
Key Generation	14 sec	Off-card (S)	4 sec
Precomputation	14 sec	N/A	4 sec
Signature	.03 sec	15 sec	.03 sec
Verification	16 sec	1.5 sec	10 sec
	1–5 sec off-card (P)	1–3 sec off-card (P)	

Off-card computations were performed on an 80386 33 mHz, personal computer. (P) indicates public parameters off-card and (S) indicates secret parameters off-card. Both algorithms use a 512-bit modulus.

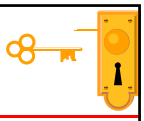
Lámina 193 Roberto Gómez C.

Diferencias RSA y DSS

	RSA	DSS
Algoritmo para cálculo del hash	MD5	SHA-1
Algoritmo de cifrado/descifrado	RSA	DSA
Desarrollador	RSA	NIST

Lámina 194 Roberto Gómez C.

¿Qué es una curva elíptica?



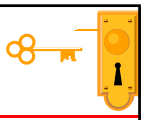
- Una curva elíptica E sobre Zp (p>2) esta definida por una ecuación de la forma

$$y^2 = x^3 + ax + b, \quad (1)$$

  - donde:
    - a, b∈ Zp
    - $4a^3 + 27b^2 \neq 0 \pmod p$
    - Y existe un punto especial **0**, llamado el "punto en el infinito".
- El conjunto E(Zp) consiste en todas las puntos (x, y), x∈Zp, y∈Zp, que satisfacen la ecuación (1), junto con el punto **0**.

Lámina 195 Roberto Gómez C.

Firma digital y curvas elípticas



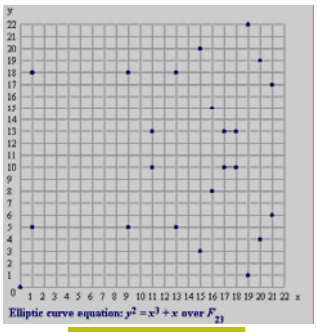
- ECDSA: Elliptic Curve DSA
- Modificación de algunos pasos en el algoritmo DSS en la selección de los números a usar
- El algoritmo de hash es el mismo
  - SHA-1
- El 15 febrero del 2000, NIST anuncia la publicación del FIPS 186-2, que substituye al FIPS 186-1 así como:
  - la aprobación del ECDSA
  - lista de curvas elípticas recomendadas para uso gubernamental

Lámina 196 Roberto Gómez C.

Ejemplo curva elíptica

- Curva elíptica sobre Z_{23}
- E: $y^2 = x^3 + x$
- Puntos en E(Z_{23})

(0, 1)	(0, 5)	(1, 18)
(9, 5)	(9, 18)	(11,10)
(11,13)	(13, 5)	(13,18)
(15, 3)	(15,20)	(16, 8)
(16,15)	(17,10)	(17,13)
(18,10)	(18,13)	(19, 1)
(19,22)	(20, 4)	(20,19)
(21, 6)	(21,17)	



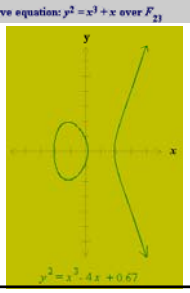
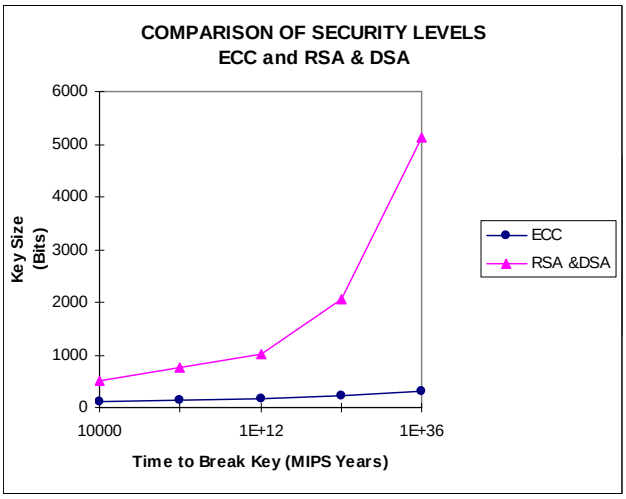


Lámina 197 Roberto Gómez C.

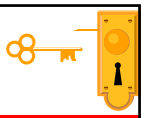
¿Porqué las curvas elípticas?



Referencia: Certicom white paper. Remarks on fhe Security of The

Lámina 198 Roberto Gómez C.

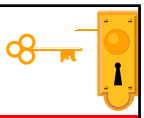
Seguridad de la firma



- Seguridad depende de lo seguro de la función hash
- No existe ninguna forma de tomar la firma de alguien de un documento y ponerla en otro
- No es posible alterar un mensaje firmado
- El más simple cambio en el documento firmado se verá en la verificación

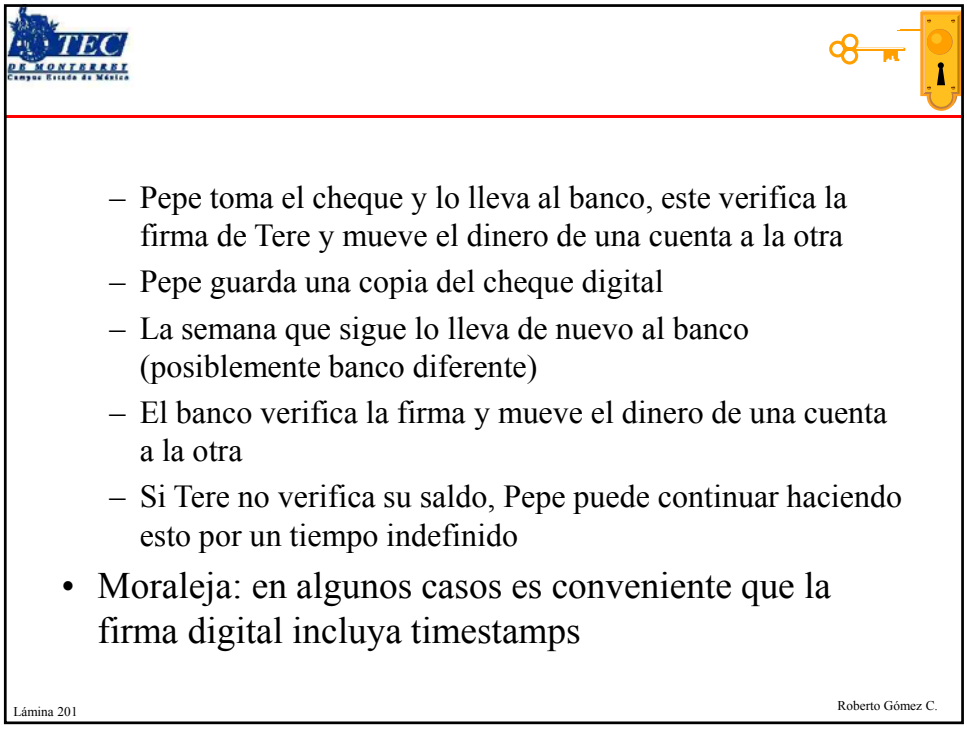
Lámina 199 Roberto Gómez C.

Documentos firmados y timestamps



- Actualmente Pepe puede comunicarse con Tere en ciertas circunstancias
- Pueden tomar un documento y firmarlo juntos
- Esto puede no ser muy importante si Tere firma un contrato, pero puede ser más interesante si Tere firma un cheque digital
- Escenario:
  - Tere le envía a Pepe un cheque de $1000 firmado digitalmente

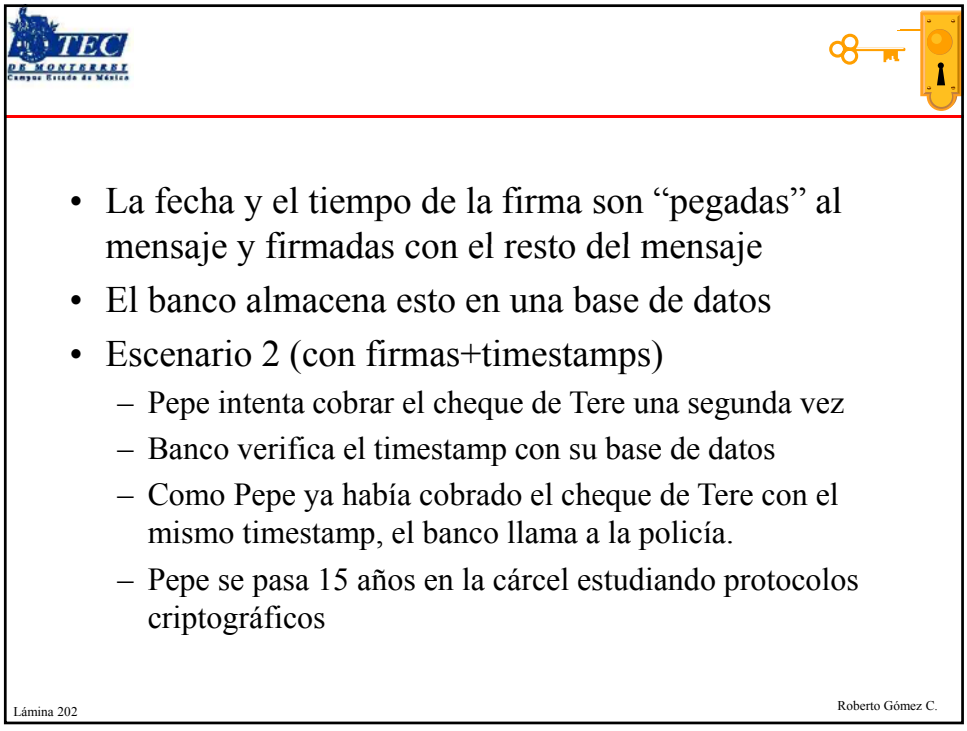
Lámina 200 Roberto Gómez C.



- Pepe toma el cheque y lo lleva al banco, este verifica la firma de Tere y mueve el dinero de una cuenta a la otra
- Pepe guarda una copia del cheque digital
- La semana que sigue lo lleva de nuevo al banco (posiblemente banco diferente)
- El banco verifica la firma y mueve el dinero de una cuenta a la otra
- Si Tere no verifica su saldo, Pepe puede continuar haciendo esto por un tiempo indefinido

- Moraleja: en algunos casos es conveniente que la firma digital incluya timestamps

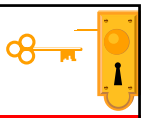
Lámina 201 Roberto Gómez C.



- La fecha y el tiempo de la firma son "pegadas" al mensaje y firmadas con el resto del mensaje
- El banco almacena esto en una base de datos
- Escenario 2 (con firmas+timestamps)
  - Pepe intenta cobrar el cheque de Tere una segunda vez
  - Banco verifica el timestamp con su base de datos
  - Como Pepe ya había cobrado el cheque de Tere con el mismo timestamp, el banco llama a la policía.
  - Pepe se pasa 15 años en la cárcel estudiando protocolos criptográficos

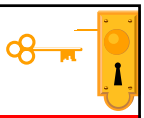
Lámina 202 Roberto Gómez C.

Firmas múltiples



- ¿Cómo podrían firmar Tere y Pepe el mismo documento?
- Sin huellas digitales, existen dos opciones
  1. Pepe y Tere firmen copias separadas del documento, documento final es el doble del original
  2. Tere firma el documento y después Pepe firma la firma de Tere, funciona pero es difícil verificar la firma de Tere sin también verificar la de Pepe

Lámina 203 Roberto Gómez C.



- El uso de huellas digitales simplifica todo:
  1. Tere firma la huella del documento
  2. Pepe firma la huella del documento
  3. Pepe envía su firma a Tere
  4. Tere envía el documento, su firma y la firma de Pepe a Raquel
  5. Raquel verifica la firma de Tere y de Pepe

Lámina 204 Roberto Gómez C.

Esquema firmas múltiples

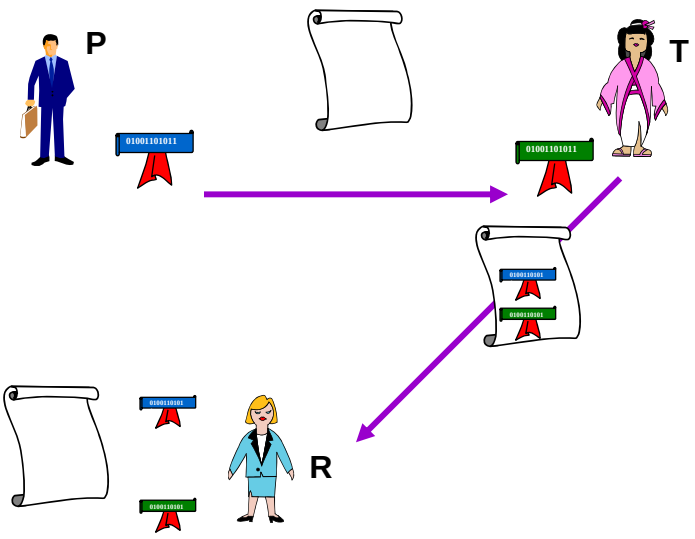
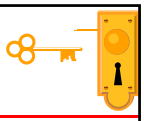


Lámina 205 Roberto Gómez C.

- Detalles
  - Tere y Pepe pueden hacer pasos 1 y 2 paralelamente o en secuencialmente
  - En paso 5 Raquel puede verificar una firma sin verificar la otra

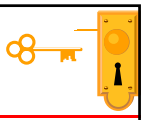
Lámina 206 Roberto Gómez C.

No repudiación y firmas digitales



- Tere puede firmar un documento y después argumentar que no lo hizo:
  - primero firma el documento normalmente
  - después publica anónimamente su llave secreta, la pierde en un lugar público o pretende ser otra persona
  - Tere alega que su firma esta comprometida y que otros la están usando pretendiendo ser ella
  - ella no reconoce la firma de ningún documento y cualquier otro que haya firmado usando su llave privada

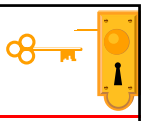
Lámina 207 Roberto Gómez C.



- Lo anterior se conoce como repudiación
- Las timestamps pueden ayudar pero Tere puede reclamar que su llave se comprometió antes
- Resumiendo, Tere puede firmar un documento y después decir que no lo hizo
- No es posible evitar lo anterior, pero es posible tomar acciones para garantizar que viejas firmas no se invaliden por acciones tomadas debido a las disputas de varias de nuevas
- Base solución: un servidor de confianza (Toto)

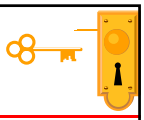
Lámina 208 Roberto Gómez C.

Protocolo de no repudiación



1. Tere firma un mensaje
2. Tere procede a formar un paquete:
   - genera un encabezado con información de su identidad
   - concatena el mensaje firmado con el encabezado,
   - firma todo y se lo envía a Toto
3. Toto recibe el mensaje y:
   - verifica la firma y confirma la identidad,
   - añade un timestamp al mensaje firmado por Tere

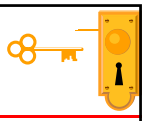
Lámina 209 Roberto Gómez C.



   - añade la información de identidad
   - firma todo y se lo envía a Tere y a Pepe
4. Pepe verifica la firma de Toto, la información de identidad y la firma de Tere
5. Tere verifica el mensaje que Toto le envió a Pepe:
   - si ella no origino el mensaje, lo anuncia rápidamente

Lámina 210 Roberto Gómez C.

Códigos Autenticación Mensaje



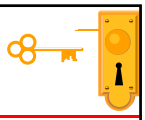
- MAC por sus siglas en inglés
- También conocido bajo el nombre de DAC (Data Authentication Code)
- Función de un solo sentido junto con una llave secreta:

$$MAC = C_K(M)$$

- M es el mensaje, K es una llave que conoce tanto el emisor como el receptor, y C_K es una función hash basada en K.

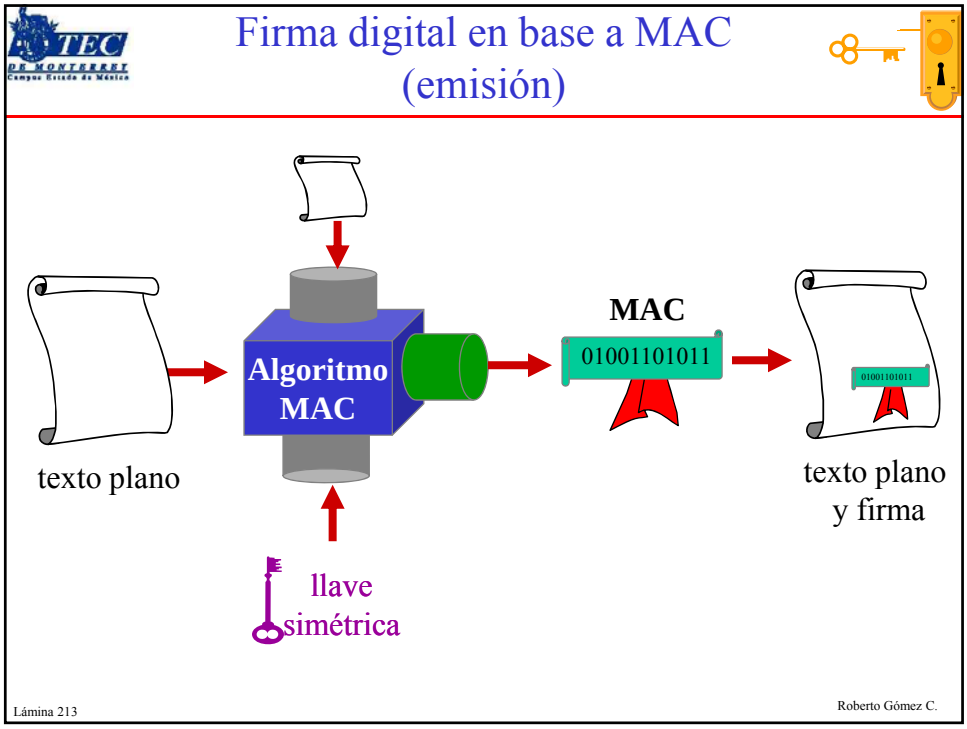
Lámina 211 Roberto Gómez C.

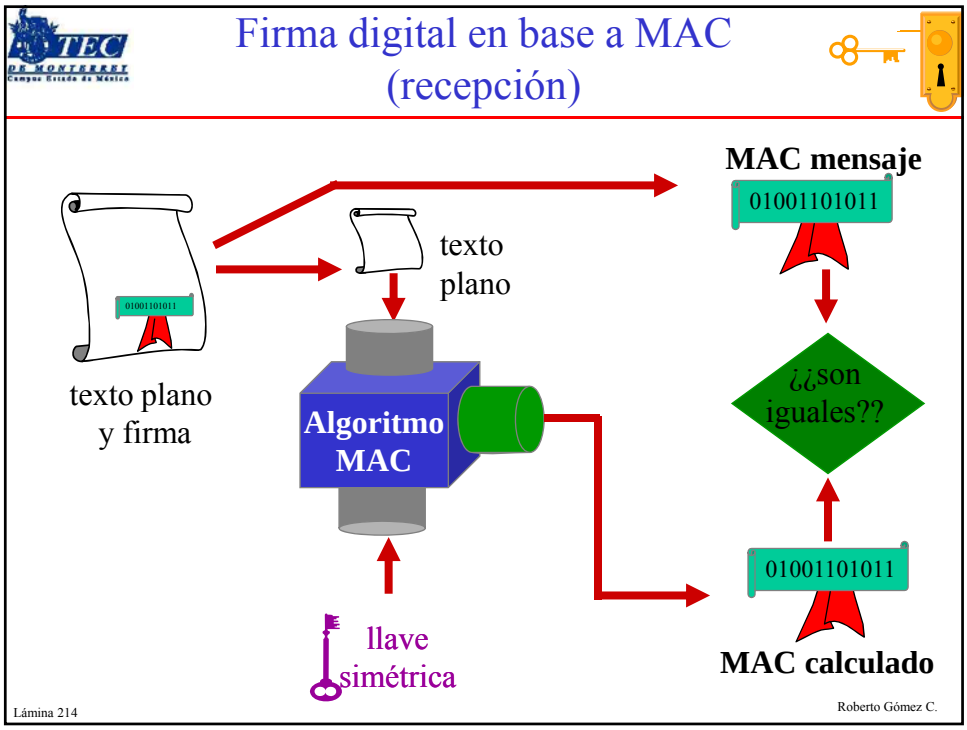
Códigos Autenticación Mensaje



- El principio es el mismo, pero si alguien conoce la llave puede verificar el valor hash.
- El sitio emisor pega el MAC al mensaje cuando se decide que el mensaje es correcto.
- El receptor auténtica re-calculando el MAC

Lámina 212 Roberto Gómez C.


Firma digital en base a MAC
(emisión)
texto plano
Algoritmo
MAC
MAC
01001101011
texto plano
y firma
llave
simétrica
Lámina 213
Roberto Gómez C.


Firma digital en base a MAC
(recepción)
MAC mensaje
01001101011
texto
plano
texto plano
y firma
Algoritmo
MAC
¿¿son
iguales??
01001101011
llave
simétrica
MAC calculado
Lámina 214
Roberto Gómez C.

Ejemplo MAC

- Uno de los más usados es el Data Authentification Algorithm.
- Esta basado en DES.
- Es una publicación FIPS (FIPS PUB 113) y un estándar ANSI (X9.17).
- Consiste en una versión de DES operando en modo CBC con un vector de inicialización cero.
- Los datos a autenticar son agrupados en bloques contiguos de 64 bits
- Si es necesario el último bloque se puede rellenar de bits con valor de cero, para completar el bloque de 64 bits.

Lámina 215 Roberto Gómez C.

Funcionamiento DAA

- Usando el algoritmo de encripción DES, una llave secreta K, el MAC se calcula:

$$O_1 = E_K(D_1)$$
$$O_2 = E_K(D_2 \oplus O_1)$$
$$O_3 = E_K(D_3 \oplus O_2)$$
$$\vdots$$
$$O_N = E_K(D_N \oplus O_{N-1})$$

Lámina 216 Roberto Gómez C.

Esquema general DAA

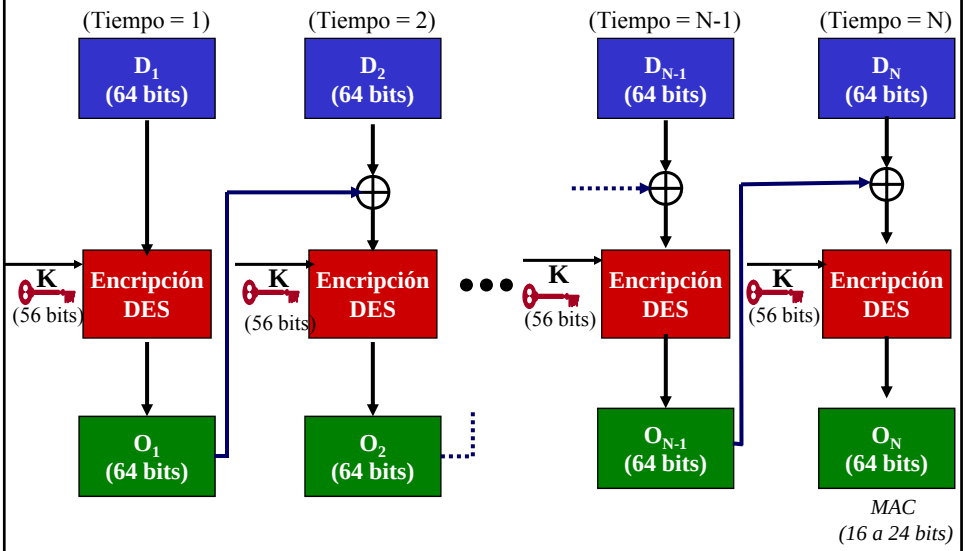


Lámina 217 Roberto Gómez C.

Usando Keyed Hash Functions para generar MACs

- Se desea crear un MAC usando una función hash más que un bloque de encripción
  - funciones hash son generalmente más rápidas
  - no estan limitadas por controls de exportación como algunos bloques de encripción
- Hash incluye una llave junto con el mensaje.
- Propuesta original

```
KeyedHash = Hash(Key|Message)
```

  - algunas debilidades se encontraron en esta propuesta
- Eventualmente llevaron al desarrollo de HMAC

Lámina 218 Roberto Gómez C.

HMAC

- Especificado con un Internet standard RFC2104
- Usa una función hash en el mensaje:

$$HMAC_K = Hash[(K^+ \text{ XOR opad}) \,||\, Hash[(K^+ \text{ XOR ipad})||M)]]$$

  - donde K^+ es un bloque creado por la llave, a la que se le añaden ceros a su izquierda, para que su tamaño sea de b bits
  - opad, ipad: constantes de relleno (36_h y $5C_h$ respectivamente)
- El overhead solo implica tres operaciones tipo hash extras que el mensaje necesita
- Puede usarse cualquier algoritmo de hash
  - MD5, SHA-1, RIPEMD-160

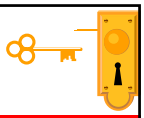
Lámina 219 Roberto Gómez C.

Algoritmo HMAC (variables)

- Sea
  - K llave secreta
  - IV valor inicial de entrada a la función hash
  - n: longitud del código hash producido por la función hash
  - H función hash
  - M mensaje de entrada a HMAC
  - L número de bloques en M
  - Y_i : iésimo bloque de M
  - b: número de bits en el bloque
  - ipad = 00110110 (36_h)
  - opad = 01011100 ($5C_h$)

Lámina 220 Roberto Gómez C.

Algoritmo HMAC (1)

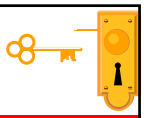


$HMAC_K = Hash[(K^+ XOR opad) || Hash[(K^+ XOR ipad)||M)]]$

1. Añadir ceros a la izquierda de K para crear un string de b bits llamado K^+
   - si longitud K es de 160 bits y b = 512, entonces se le añadirán a K 44 bytes de valor 0
2. Suma binaria de K^+ para producir el bloque se b-bits S_i
3. Añadir M a S_i
4. Aplicar H al stream generado en paso 3

Lámina 221 Roberto Gómez C.

Algoritmo HMAC (2)



$HMAC_K = Hash[(K^+ XOR opad) || Hash[(K^+ XOR ipad)||M)]]$

5. Suma binaria K^+ con opad para producir bloque S_0 de tamaño b
6. Añadir el resultado de la función hash del paso 4 a S_0
7. Aplicar H al stream generado en el paso 6 y sacar el resultado

Lámina 222 Roberto Gómez C.

Diagrama operación HMAC

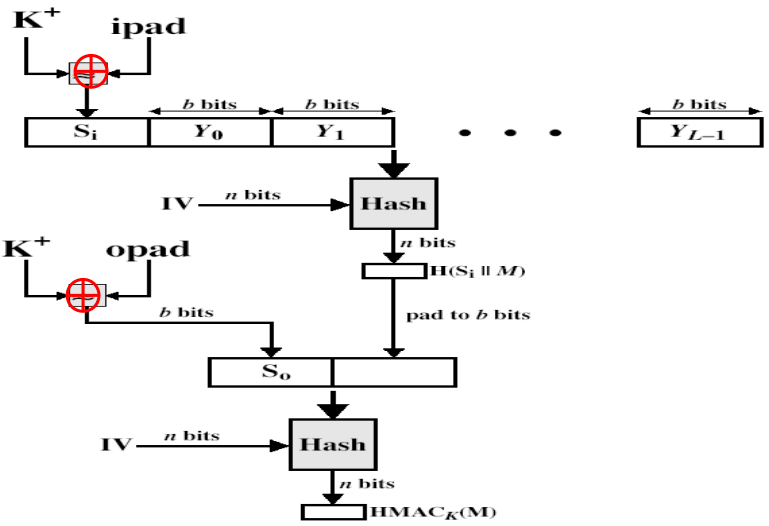


Lámina 223 Roberto Gómez C.

Seguridad HMAC

- La seguridad de HMAC esta relacionada con el algoritmo de hash usado.
- Atacar HMAC requiere, ya sea:
  - ataque de fuerza bruta en la llave usada
  - ataque de cumpleaños, pero es necesario ver un gran numero de mensajes
- Elegir la función hash tomando en cuenta velocidad procesamiento contra requerimientos de seguridad.

Lámina 224 Roberto Gómez C.

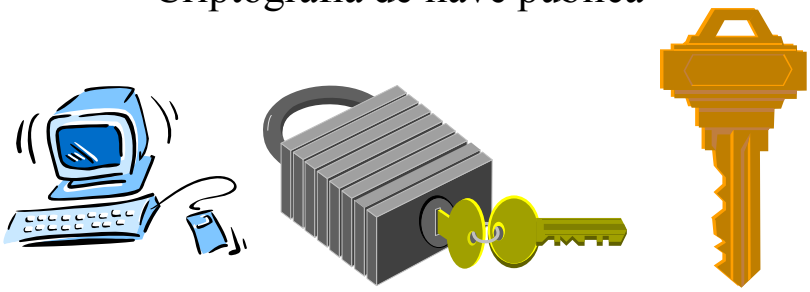
Otros algoritmos de firma digital de llave pública

- Variantes DSA
- Gost Digital Signature Algorithm
- Discrete Logarithm Signature Schemes
- Ong-Schorr-Shamir
- ESIGN
- Automatas celulares

Lámina 225 Roberto Gómez C.

Criptologia Asimétrica

Criptografía de llave pública



Fecha última modificación: marzo 2009

Lámina 226 Roberto Gómez C.