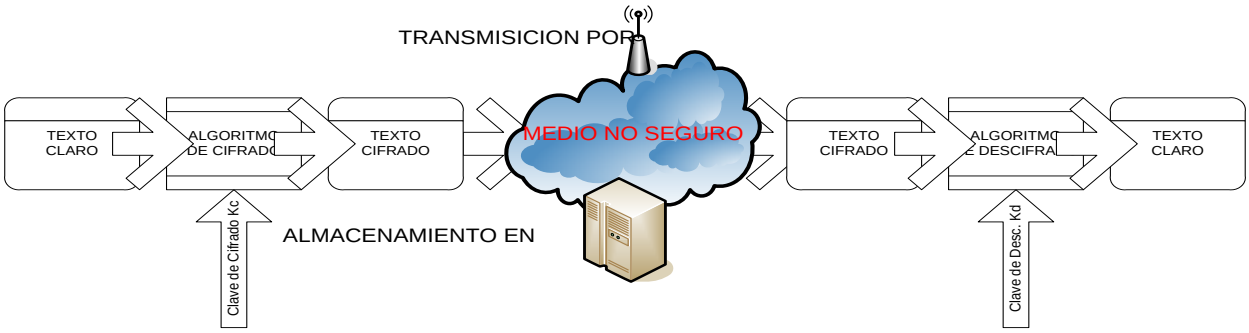
UNIVERSIDAD NACIONAL DE JUJUY (UNJu)
Licenciatura en Sistemas
Seguridad en Sistemas **Ing. José Daniel Zakhour**

UNIDAD 7
PROCESO DE ENCRIPTACION

Denominaremos como proceso de encriptación al sistema compuesto de las siguientes etapas (Criptosistema):



El concepto de ***texto claro*** hace referencia a todo tipo de información, no únicamente texto, que presenta un significado para cualquiera que pueda accederla. Frente al de ***texto cifrado*** que representa información ilegible o ininterpretable para quien la acceda. Se destaca la existencia de uno, o de dos, algoritmos de transformación de texto claro a texto cifrado, y del uso de claves para pasar de uno a otro y viceversa.

Si la clave de cifrado es igual a la clave de descifrado ($K_c = K_d$) el proceso de encriptación de denomina simétrico o de clave privada. En caso contrario ($K_c \neq K_d$) el proceso de encriptación se denomina asimétrico o de clave pública (K_c).

Surge además que la necesidad de aplicar un proceso de encriptación es imprescindible en aquellos casos en que nuestra información debe ser almacenada o transmitida por medios "NO SEGUROS", entendiendo por tal aquellos de acceso público o compartido y cuando interesa salvaguardar de terceros el contenido de la misma.

Criptografía Clásica

Podemos llamar así a todos los sistemas de cifrado anteriores a la II Guerra Mundial, o lo que es lo mismo, al nacimiento de las computadoras.

La transición desde la Criptografía clásica a la moderna se da precisamente durante la II Guerra Mundial, cuando el Servicio de Inteligencia aliado rompe la máquina de cifrado del ejército alemán, llamada ENIGMA.

Todos los algoritmos criptográficos clásicos son simétricos, ya que hasta mediados de los años setenta no nació la Criptografía asimétrica.

Algoritmos Clásicos de Cifrado

Estudiaremos en esta sección algunos criptosistemas que en la actualidad han perdido su eficacia, debido a que son fácilmente criptoanalizables empleando cualquier computadora doméstica, pero que fueron empleados con éxito hasta principios del siglo XX. Algunos se remontan incluso, como el algoritmo de César, a la Roma Imperial. Sin embargo mantienen un interés teórico, ya que nos van a permitir explotar algunas de sus propiedades para entender mejor los algoritmos modernos.

Cifrados Monoalfabéticos

Se engloban dentro de este apartado todos los algoritmos criptográficos que, sin desordenar los símbolos del lenguaje, establecen una correspondencia única para todos ellos en todo el mensaje. Es decir, si al símbolo A le corresponde el símbolo D, esta correspondencia se mantiene a lo largo de todo el mensaje.

Criptografía Clásica

Algoritmo de César

El algoritmo de César, llamado así porque es el que empleaba Julio César para enviar mensajes secretos, es uno de los algoritmos criptográficos más simples. Consiste en sumar 3 al número de orden de cada letra. De esta forma a la A le corresponde la D, a la B la E, y así sucesivamente. Si asignamos a cada letra un número (A = O,B = 1...), y consideramos un alfabeto de 26 letras, la transformación criptográfica seria:

$C= (M+3) \bmod 26$

Obsérvese que este algoritmo ni siquiera posee clave, la transformación siempre es la misma.

Sustitución Afmn

Es el caso general del algoritmo de Cesar. Su transformación seria:

$$Ek(M) = (aM+b) \bmod 26$$

siendo a y b dos números enteros menores que el cardinal N del alfabeto, y cumpliendo que rncd(a,N) = 1. La clave k es el par (a,b).

Cifrado Monoalfabético General

Es el caso más general de cifrado monoalfabétido. La sustitución ahora es arbitraria, siendo la clave k precisamente la tabla de sustitución de un símbolo por otro.

Criptoanálisis de los Métodos de Cifrado Monoalfabéticos

El cifrado monoalfabético constituye la familia de métodos más simple de criptoanalizar, puesto que las propiedades estadísticas del texto plano se conservan en el criptograma. Supongamos que, por ejemplo, la letra que más aparece en Castellano es la A. Parece lógico que la letra más frecuente en el texto codificado sea aquella que corresponde don la A. Emparejando las frecuencias relativas de aparición de cada símbolo en el mensaje cifrado con el histograma de frecuencias del idioma en el que se supone está el texto plano, podremos averiguar fácilmente la clave.

Cifrados Polialfabéticos

En los cifrados polialfabéticos la sustitución aplicada a cada carácter varía en función de la posición de éste dentro del texto plano. En realidad corresponde a la aplicación cíclica de ii cifrados monoalfabéticos.

Cifrado de Vigenere

Es un ejemplo típico de cifrado polialfabético, cuya clave es una secuencia de letras K = {ko, k1, .. kdl} y que emplea la siguiente función de cifrado:

Ek(mj) = m + k(~ mod d) (mod ii)

siendo m el i-ésimo símbolo del texto plano y ii el cardinal del alfabeto de entrada.

Criptoanálisis

Para criptoanalizar este tipo de claves basta con efectuar D análisis estadísticos independientes agrupando los símbolos según la k~ empleada para codificarlos. Para estimar D, buscaremos la periodicidad de los patrones comunes que puedan aparecer en el texto cifrado. Obviamente, para el criptoanálisis, necesitaremos al menos D veces más cantidad de texto que con los métodos monoalfabéticos.

Cifrados de Transposición

Este tipo de mecanismos de cifrado no sustituye los símbolos por otros, sino que cambia su orden dentro del texto. Un mecanismo de transposición podría consistir en colocar el texto en una tabla de 71 columnas, y dar domo texto cifrado los símbolos de una columna (ordenados de arriba a abajo) concatenados con los de otra, etc. La clave serla el numero 71 y el orden en el que se leen las columnas.

Por ejemplo, supongamos que queremos cifrar el texto "El perro de San Roque no tiene rabo", con ii = 5 y la permutación {3, 2,5,1, 4} como clave. Colocamos el texto en una tabla y obtenemos:

1	2	3	4	5
E	L		P	E
R	R	U		D
E	R	S		
0	A			
Q	N			
U				
E		N	U	
T	I			
R	E			
A	N			
B	E			
0				

Tendríamos como texto cifrado " Osonealr r irednu eoere et p aqonb".

Criptoanálisis

Este tipo de mecanismos de cifrado se puede criptoanalizar efectuando un estudio estadístico sobre la frecuencia de aparición de pares y tripletas de símbolos en el texto plano.
Máquinas de Rotores. La Máquina ENIGMA. En el año 1923, un ingeniero alemán llamado Arthur Scherbius patentó una máquina específicamente diseñada para facilitar las comunicaciones seguras. Se trataba de un instrumento de apariencia simple, parecida a una máquina de escribir. Quien deseara codificar un mensaje solo tenía que teclearlo y las letras correspondientes al mensaje cifrado se irían iluminando en un panel. El destinatario copiaba dichas letras en su propia máquina y el mensaje original aparecía de nuevo. La clave la constituían las posiciones iniciales de tres tambores o rotores que el ingenio pósela en su parte frontal.

Cifrado basado en funciones modulares

Dados tres números **a, b, θ ε N**, decimos que **a** es congruente con **b** modulo **θ**, y se escribe: a b(mod **θ**)

Si se cumple:

a = b + k **θ** k **ε** z

Los cifrados más actuales hacen uso intensivo de estos métodos básicos de cifrado para obtener algoritmos mas sofisticados.

Seguridad de un Criptosistema

Frente a lo que naturalmente parecería sugerir un proceso de encriptación, la seguridad no

esta directamente relacionada con la complejidad del algoritmo, ***sino con la baja probabilidad de obtener en un tiempo operativo valido y mediante métodos de criptoanálisis de búsqueda exhaustiva, la clave de cifrado y/o descrifrado del criptosistema.*** Donde cabe aclarar que tiempo operativo valido, representa un tiempo cercano, de nada me sirve que yo encuentre, si o si, por búsqueda exhaustiva, en la jerga "fuerza bruta", es decir probando todas las claves posibles, una clave en *2000 años*. **Cualquier método de cifrado por complejo que sea, si no se protege adecuadamente las claves, pierde su robustez**. Además cuando hablamos de métodos de cifrado estándar, estamos hablando de cifrado de dominio público para poder garantizar la estandarización.

Algoritmo DES

DES (Data Encrytion Standart):

Fue diseñado en USA(70) con la idea de que fuese un estándar y fuese usado por las empresas en las transacciones. Fue criticado ya que usa una clave secreta de 64 bits, pero de éstos sólo 56 son la clave ya que hay 8 bits de control (corrección de errores). Este sistema no ofrecía seguridad suficiente. El espacio de claves es enorme pero con ordenadores potentes se puede encontrar la clave.

Algoritmo de Cifrado

DES, es un algoritmo de cifrado en bloques simétrico, el tamaño del bloque es de longitud fija de 64 bits, el algoritmo consta de dos permutaciones, una al inicio conocida como *P1,* la cual se muestra a continuación:

Tabla antes la Permutación

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

Se separan el mensaje original de 64 bits en bloques de 8 bits, previo a la permutación, la tabla *P1* muestra el resultado de la permutación.

Permutación Inicial (P1)

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

En la parte superior se puede observar que se encuentran los números pares.

En la parte inferior se puede observar que se encuentran los números impares.

Al investigar información sobre DES, se puede encontrar la siguiente tabla:

Permutación Inicial (P1)															
58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

Después de recibir un bloque de entrada de 64 bits, el *primer paso* consiste en aplicar al bloque de entrada la permutación P1, teniendo como resultado un orden de salida que se identifica leyendo la tabla de izquierda a derecha y de arriba abajo. Significa que el bit del lugar 58 en el mensaje de entrada, después de la permutación, ocupara la posición 1 y así sucesivamente. Ejemplo:

Bloque de Entrada:	Bloque de Salida:
0..1 ..0 ..1 ..0 ..1 ..1 ..0	0..1 ..1 ..0 ..1 ..0 ..1 ..0
2..10..18..26..34..42..50..58	58..50..42..34..26..18..10.. 2

Una vez realizada la permutación, los 64 bits se dividen en dos sub-bloques Left y Right (L_i y R_i) de 32 bits, los bits que forma el sub-bloque L_i se encuentra formado por los primeros 32 bits y los bits restantes forma el sub-bloque R_i. En estas condiciones, el cifrado DES esta definido por las ecuaciones:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus f\,(R_{i-1}, K_i)$$

El valor de i = 16, representa el valor de las 16 vueltas del algoritmo. Lo anterior se explicará con un ejemplo, el cual se ira detallando a lo largo del trabajo, ejemplificando cada proceso del algoritmo.

Ejemplo Permutación P1

Mensaje a Cifrar = Denytamo

Decimal	Carácter	Binario
97	a	01100001
68	D	01000100
101	e	01100101
109	m	01101101
110	n	01101110
111	o	01101111
116	t	01110100
121	y	01111001

D	01000100
e	01100101
n	01101110
y	01111001
t	01110100
a	01100001
m	01101101
o	01101111

Utilizando la tabla Permutación Inicial P1, tenemos:

Tabla antes la Permutación

0	1	0	0	0	1	0	0
0	1	1	0	0	1	0	1
0	1	1	0	1	1	1	0
0	1	1	1	1	0	0	1
0	1	1	1	0	1	0	0
0	1	1	0	0	0	0	1
0	1	1	0	1	1	0	1
0	1	1	0	1	1	1	1

Permutación Inicial (P1)

1	1	1	1	1	1	1	1
0	0	0	1	1	0	0	0
1	1	0	1	0	1	1	1
1	1	1	0	1	0	1	0
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	0
1	1	0	0	1	1	0	0
1	0	0	0	0	1	0	0

Se muestra el resultado de haber realizado la permutación *P1*, la parte superior se encuentra marcada con el fin de indicar cuales son los bits que forman el sub-bloque L_0, los bits restantes forman el sub-bloque R_0, dando como resultado:

L_0 = 11111111 00011000 11010111 11101010

R_0 = 00000000 11111110 11001100 10000100

Sub-bloques iniciales

Permutación E

La salida de R_0 es de 32 bits, se utiliza la permutación E, con el propósito de expandir a 48 bits y así poder realizar la suma OR exclusiva con la clave K_i, lo anterior se encuentra esquematizado en la imagen. A continuación se muestra la tabla para realizar la permutación E.

Bits duplicados

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
17	18	19	20
21	22	23	24
25	26	27	28
29	30	31	32

Permutación E

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

En la tabla del lado izquierdo se encuentran marcados los bits utilizados para expandir a 48 bits en la permutación E. Al realizar la expansión de R_0, se procede con la suma OR exclusiva que se explicara mas adelante.

Ejemplo Permutación E

Al tener la secuencia de R_0 de 32 bits, es necesario aplicar la permutación E, la cual se muestra a continuación.

R_0 = 0000 0000 1111 1110 1100 1100 1000 0100

32 Bits

0	0	0	0
0	0	0	0
1	1	1	1
1	1	1	0
1	1	0	0
1	1	0	0
1	0	0	0
0	1	0	0

Permutación E

0	0	0	0	0	0
0	0	0	0	0	1
0	1	1	1	1	1
1	1	1	1	0	1
0	1	1	0	0	1
0	1	1	0	0	1
0	1	0	0	0	0
0	0	1	0	0	0

El resultado de la permutación E(R_0) es:

E(R_0) = 000000 000001 011111 111101 011001 011001 010000 001000

Generación de la subclave K_i

La clave K_i tiene un valor inicial de 64 bits, su longitud es fija.

Tabla de 64 bits Inicial

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

Obteniendo como resultado la tabla que se utilizará para realizar la permutación PC1, donde se observa la falta de bits de paridad de cada byte, esto debido a que no aportan ninguna información.

Tabla de 56 bits

1	2	3	4	5	6	7
9	10	11	12	13	14	15
17	18	19	20	21	22	23
25	26	27	28	29	30	31
33	34	35	36	37	38	39
41	42	43	44	45	46	47
49	50	51	52	53	54	55
57	58	59	60	61	62	63

Permutación PC1

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

La tabla de la Permutación PC1, se utiliza para realizar la permutación inicial en la generación de la subclave K_i , para cada vuelta. Una vez realizada la permutación, los 56 bits se dividen en dos sub-bloques C_i y D_i de 28 bits. En estas condiciones, la clave esta definida por las ecuaciones:

$$C_i = LS\ (C_{i-1}) \quad D_i = LS(D_{i-1})$$

$$K_i = PC2\ (C_i\ ,\ D_i)$$

Ejemplo Permutación PC1

Clave K: Santiago

Decimal	Carácter	Binario
97	a	01100001
103	g	01100111
105	i	01101001
110	n	01101110
111	o	01101111
83	S	01010011
116	t	01110100

S	01010011
a	01100001
n	01101110
t	01110100
i	01101001
a	01100001
g	01100111
o	01101111

Utilizando la Permutación PC1 obtenemos:

Tabla de 64 bits de K_i Inicial

0	1	0	1	0	0	1	1
0	1	1	0	0	0	0	1
0	1	1	0	1	1	1	0
0	1	1	1	0	1	0	0
0	1	1	0	1	0	0	1
0	1	1	0	0	0	0	1
0	1	1	0	0	1	1	1
0	1	1	0	1	1	1	1

En la tabla de 56 bits se muestra la ausencia de los bits de paridad de cada byte que conforman la clave.

Tabla de 56 bits

0	1	0	1	0	0	1
0	1	1	0	0	0	0
0	1	1	0	1	1	1
0	1	1	1	0	1	0
0	1	1	0	1	0	0
0	1	1	0	0	0	0
0	1	1	0	0	1	1
0	1	1	0	1	1	1

Permutación PC1

0	0	0	0	0	0	0
0	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	0	0	0	0	0
1	1	0	0	0	1	0
1	1	1	0	0	1	1
0	0	1	0	0	1	0
1	0	0	1	0	0	1

Sub-bloque C_0 = 0000000 0111111 1111111 1100000

Sub-bloque D_0 = 1100010 1110011 0010010 1001001

Sub-bloques iniciales

Desplazamiento *LS*(…)

El desplazamiento *LS*(…) se aplica a los sub-bloques de longitud fija de 7 bits (C_i y D_i), donde *LS*(…) es un desplazamiento circular a la izquierda de 1 o 2 bits del entero binario que toma el argumento de acuerdo a la siguiente tabla:

Vuelta	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
No. Bits desplazados. Izda.	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Con el propósito de comprender mejor el desplazamiento *LS*(…) se realizara el proceso.

Ejemplo Permutación *LS*(..)

Al tener los sub-bloques C_0 y D_0, el siguiente paso es el desplazamiento *LS*, se marcan los dos primeros bits, con el propósito de poder identificar el resultado del desplazamiento.

Sub-bloque C_0 = 0000000 0111111 1111111 1100000

Sub-bloque D_0 = 1100010 1110011 0010010 1001001

Al ser la primer vuelta, el desplazamiento es de un bit a la izquierda como se indica en la tabla, dando como resultado C_1 y D_1.

Sub-bloque C_1 = 0000000 1111111 1111111 1000000

Sub-bloque D_1 = 1000101 1100110 0100101 0010011

Sub-bloques después del desplazamiento LS(..)

Permutación PC2

La permutación PC2 se conoce como permutación de compresión, dada por las operaciones de concatenar y permutar C_i y D_i, se va a comprimir de 56 bits a 48 bits para obtener la clave K_i, posteriormente será utilizada en la función ($f(R_{i-1}, K_i)$)
El orden de concatenar C_i y D_i, es utilizando primero los 28 bits de C_i y posteriormente los 28 bits de D_i, la tabla PC2 es una tabla de 8 x 6, dando como resultado 8 bloques de 6 bits.

Tabla (C_i, D_i)

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	32	33	34	35
36	37	38	39	40	41	42
43	44	45	46	47	48	49
50	51	52	53	54	55	56

Se muestra el orden para generar la tabla de permutación PC2, en la tabla de la izquierda se marcan los bits eliminados para comprimir a 48 bits.

Tabla de 8 x 6 bits

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36
37	38	39	40	41	42
43	44	45	46	47	48

Permutación PC2

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Ejemplo Permutación PC2

Continuando con el ejemplo se mostrará el proceso utilizado en la permutación PC2, para obtener la clave K_i.

Concatenando C_i, D_i

0000000 1111111 1111111 1000000 1000101 1100110 0100101 0010011

Los 56 bits de entrada en la permutación PC2

Tabla (C_i, D_i)

0	0	0	0	0	0	0
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	0	0	0	0	0	0
1	0	0	0	1	0	1
1	1	0	0	1	1	0
0	1	0	0	1	0	1
0	0	1	0	0	1	1

Permutación PC2

1	1	1	0	0	0
0	0	1	0	1	1
0	1	1	0	0	1
1	0	0	1	1	0
1	1	0	1	1	1
0	1	0	0	1	0
1	1	0	1	0	0
0	0	0	1	1	0

El resultado de haber realizado la permutación PC2 es la generación de la clave K_1 siendo:

K_1 = PC2 (C_1, D_1) =

111000 001011 011001 100110 110111 010010 110100 000110

Clave K_1

Las operaciones *LS*(..) y PC2, se repiten 15 veces para así obtener las 15 subclaves de cifrado restantes. Los procesos de cifrado y generación de claves se representan esquemáticamente en la siguiente imagen

Función *f*(R_{i-1} , K_i)

Al tener la clave K_i y la expansión de (R_0), el siguiente paso es la función *f*(R_{i-1} , K_i), la cual consta de tres procesos (Suma OR exclusivo, ocho funciones no lineales, Permutación P), siendo las ocho funciones lineales la mayor virtud del algoritmo, se han propuesto modificaciones en varios procesos, pero cualquier modificación realizada, en las funciones lineales hace débil al algoritmo. A continuación se explican los procesos.

Algoritmo RSA

Riverest-Shamir-Adleman , se piensa que el criptosistema RSA es tan seguro como lo era en 1978.

RSA = (P, C, K, E, D) donde:

$\mathbb{P}$ = $\mathbb{C}$ = Zn, con n = pq p y q primos

Este algoritmo se basa en la elección de un par de números primos P y Q que deben ser positivos y grandes, ya que cuanto más grandes sean más dígitos tendrá otro número llamado N, y más difícil será factorizar este N en un tiempo razonable. Algunos ejemplos son los siguientes:

- N=100 dígitos se tardaría una semana
- N=150 dígitos se tardaría 1000 años
- N>200 dígitos se tardaría 1 millón de años

Pero claro estos datos se quedan anticuados casi con el paso de los días. Es decir, no se puede asegurar que, debido al enorme desarrollo de las tecnologías informáticas, el computador más rápido actual, con el que se han sacado estos datos, no sea el más lento mañana.

$K = \{(n, e, d)/ ed \equiv 1 \pmod{\phi(n)}\}$

Para cada $k = (n, e, d)$

$Ek(x) := x^e \pmod n \ \forall\ x \in Zn$

$Dk(x) := x^d \pmod n \ \forall\ x \in Zn$

$\phi(n)$ = número de enteros positivos menores que n y primos con n.

$Zn = \{0, 1, ..., n-1\}$

En Zn sólo tienen inverso los x tales que $mcd(x,n) = 1$

$Z = \{x \in Zn / mcd(x,n) = 1\}$ es un grupo finito

$|Z| = \phi(n)$

En el caso del sistema RSA:

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

$|Z| = p-1$ si p es primo

$\forall\ x \in Zn \ \ Dk(Ek(x)) = x$

Teorema (Lagrange): Si G es un grupo finito y $x \in G$, entonces $x^{|G|} = 1$

Corolario (Euler): Si $x \in Z+$, $mcd(x,n) = 1$, entonces $x^{\phi(n)} \equiv 1 \pmod n$

Corolario (Teorema pequeño de Fermat): Si p es primo, $x^p \equiv x \pmod p$

$x \in Z$

$x^{p-1} \equiv 1 \pmod p$

$x^p \equiv x \pmod p$

$x \in Z$

$Ek(x) \equiv x^e \pmod n$

$Dk(Ek(x)) \equiv \pmod n \equiv x^{ed} \pmod n \equiv x^{t\phi(n)+1} \pmod n \equiv \pmod n \equiv$ $x \pmod n$

$ed \equiv 1 \pmod{\phi(n)} \Leftrightarrow ed = t\phi(n) +1$

El par (n,e) constituye la clave pública mientras que d, p y q se mantienen en secreto.

Cada usuario u elige dos primos distintos grandes (512 bits ≅ 154 dígitos decimales) pu y qu y calcula nu = puqu (n recibe el nombre de módulo).

Se calcula $\phi(nu) = (pu - 1)(qu - 1) = |Z|$. u elige un entero eu tal que $1 < eu < \phi(nu)$ y $mcd(eu,\phi(nu)) = 1$. A continuación se calcula el inverso de eu en Z, es decir, se calcula el entero du tal que $1 < du < \phi(nu)$ y $eudu \equiv 1 \pmod{\phi(nu)}$

eu $\rightarrow$ exponente de encriptación

du $\rightarrow$ exponente de desencriptación

Clave pública : (nu,eu)

Eu(x) : = x (mod nu)

Du(x) : = x (mod nu)

u mantiene secreto pu, qu y du

Los números más difíciles de factorizar son los que se factorizan como 2 primos que tienen el mismo número de dígitos.

Se presentan dos problemas:

- Reconocimiento de primos (primality test)
- Factorización de enteros.

Al usar el RSA queremos que otra parte no rompa el sistema, cualquier persona que conozca la descomposición de un en factores primos puede desencriptar, esto lleva al problema de la factorización de enteros; nadie conoce un método eficiente para factorizar enteros generales.

A $\rightarrow$ B

A busca (nB,eB)

A envía a B x (mod nB)

B calcula (x) (mod nB) = x

Un criptoanalista tiene x (mod nB), debe hallar la raíz eB de x, para esto debe conocer $\phi(nB)$ y para ello debe conocer qB y dB.

p = 47 q = 59

$n = pq = 47 \cdot 59 = 2773$

$\phi(n) = \phi(p) \cdot \phi(q) = 46 \cdot 58 = 2668$

e = 1225

	2	5	1	1	1	1	1	1
2668	1225	218	135	83	52	31	21	10
21	135	83	52	31	21	10	1	

mcd(1225,2668) = 1

1 = 257·1225 – 118·2668

257·1225 ≡ 1 (mod 2668)

d = 257

Clave pública (2773,1225)

Clave privada (2773,257)

x1225 (mod 2773)

Exponenciación binaria:

1225 = (10011001001)2 = 210 + 27 + 26 + 23 + 20

x1225 =

1 ↔ cx

0 ↔ c

c = elevado al cuadrado

x = multiplicar

cxcccxcxcccxccx

(((((((((x2)2)2·x)2·x)2)2)2·x)2)2)2)·x (mod 2773)

Haciendo esto calculamos directamente x1225 (mod 2773); hacemos esto con sólo 14 multiplicaciones.

Manejo de Claves

Esencialmente hay tres formas de atacar los esquemas de protección utilizados en los Sistemas Informaticos

Adivinar la clave de acceso.

Localizar la clave de acceso.
Forzar la técnica de cifrado.

Las defensas posibles ante estos ataques son:

Seleccionar claves *difíciles+.
Tratar con cuidado las claves (anotaciones).
Utilizar técnicas de cifrado probadas.

Se puede ver como estas defensas están relacionadas con las tres preguntas hechas anteriormente:

) Qué constituye una buena clave de acceso?
) Cómo seleccionar y recordar las claves?
) Cómo funciona la protección con clave de acceso?

Hasta el momento este capítulo se ha centrado en la última de estas preguntas. Ahora es el momento de pasar a las dos primeras.

Selección y administración de claves de acceso

Una vez que se ha visto cómo funciona la protección con clave de acceso, se sabe que una de las principales debilidades es el uso de la clave *débiles+, es decir, aquellas que son fáciles de adivinar. Toda la fortaleza de la tecnología de cifrado más compleja queda sin utilidad si se puede obtener la clave de acceso.

Claves débiles

No tiene sentido utilizar la clave si ésta es * abretesesamo + o * clave + o los cuatro últimos dígitos del número de teléfono de casa. Aunque estas claves muy usadas pueden desanimar a un usuario sin experiencia que se encuentra de pronto con un archivo protegido, hay pocas posibilidades de que presenten problemas para un intruso con experiencia y determinación. Si se conoce a la persona correctas, se pueden comprar amplias vistas de claves de acceso habitualmente usadas, compiladas por piratas con experiencias. Consideremos el caso del famoso programa gusano de Internet creado por Robert Morris Jr., hijo de un oficial de la NSA. Este programa pudo acceder sin autorización a miles de potentes computadoras conectadas en una red mundial. Cuando el programa intentaba pasar de un sistema a otro, éste se encontraba con peticiones de claves de acceso diseñadas para mantener alejados a usuarios no autorizados. para sobrepasar este control de acceso el programa contaba con un módulo de clave de acceso, el cual incluía unas 400 claves muy usadas que probaba de formas sistemática, una tras otra. En una cantidad alarmante de casos la clave correcta estaba dentro de la lista.

Localización de la clave

Hay una clara relación entre la facilidad con que se puede recordar una clave y la facilidad con que se puede adivinar. Esta relación supone un dilema para los bancos que ofrecen cajeros automáticos que utilizan números de identificación personal (NIP). El dilema se pude formular de la siguiente forma: si el banco le permite a sus clientes seleccionar el número, éste seleccionará un número que sea fácil de recordar lo que puede facilitar que alguien que robe la tarjeta lo adivine; si el banco asigna un número aleatorio al cliente, a éste le será más fácil recordarlo, aunque probablemente le sea más difícil adivinarlo al usuario no autorizado. Por supuesto, se puede hacer todo tipo de disgresiones sobre esto. Cuando los usuario tienen que recordar un número sin sentido, algunos usuarios lo escriben y l.o mantienen a mano, un regalo para el ladrón que consigue a la vez la tarjeta y el número anotado sin cuidado. Por otro lado permitirle a los usuario seleccionar un número que un usuario no autorizado va a adivinar. La relación entre la facilidad para recordarlo y la facilidad para adivinarlo afecta a la selección de claves de acceso.

Selección de claves difíciles

La clave ideal es fácil de recordar y difícil de adivinar. Con el objetivo de obtener esta clave, es necesario observar que es lo que hace que una clave sea difícil de adivinar. Consideremos los siguientes criterios:

1. No debe de haber conexión lógica entre la clave y el usuario.
2. No debe de haber conexión lógica entre la clave y el contenido del archivo.
3. No debe de haber una relación visible entre la clave y la fecha del archivo.
4. La clave debería de incluir una mezcla de caracteres, tanto mayúsculas como minúscula, además de números y símbolos de puntuación y especiales.
5. La clave no debe de ser *palabra+.

Al cumplir los tres primeros criterios disminuye la posibilidad de que un atacante que conozca el tema del que trata el archivo, o cuando se creo y/o quien lo hizo, pueda utilizar esta información para adivinar la clave. Los dos últimos criterios aseguran que al intruso no le bastara con probar una serie de palabras.

Por ejemplo, una clave como PASS&6873 es mucho más oscura que PASS o Pass. Utilizados sin letras, números aleatorios de al menos cuatro dígitos de longitud suponen una clave aceptablemente oscura (1.000.000 de combinaciones posibles con cuatro dígitos de 0 a 9). Sin embargo, igual que hay palabras claves poco fiables, hay números poco fiables, como 1234, 1111, 2222, 0101, etc. Los números que se pueden deducir, como son los cumpleaños, partes del número telefónico, números del documento nacional de identificación o de la cartilla de la Seguridad Social, se deben evitar, a menos que se transformen o se usen en conjunción con texto aleatorio.

La capacidad de un sistema de cifrado de utilizar los caracteres de puntuación y

*especiales+ es un elemento valorable. Por ejemplo, hay 256 caracteres ASCII, de los que solo 62 son letras (mayúsculas y minúsculas) y números. Quedan otros 194 caracteres. No todos los sistemas de cifrado reconocen los códigos ASCII como caracteres válidos, aunque muchos lo hacen. Esto permite utilizar claves de acceso como la cara feliz del código ASCII 1. Los códigos ASCII que reconocen un sistema de clave de acceso de deben describir en las especificaciones o manuales del programa, y se deben de probar. Una frase como ' Nuevamente aquí ' es fácil de recordar y no es difícil de escribir. Sin embargo, es bastante difícil adivinarla porque tiene 15 caracteres, la longitud máxima para algunos esquemas con clave. Sin embargo, igual que se ha de evitar que las frases hechas y los juegos de palabra. De nuevo, añadiendo números y otros caracteres se fortalece la clave de acceso, como con Chinatown, 1975.

Sistemas de selección

A los usuarios que deban introducir una gran cantidad de claves de accesos, puede que les interese fijar un sistema de selección de claves. Un método simple pero efectivo para un usuario aislado es utilizar las palabras de una pagina concreta de un libro. Esto permite identificar la palabra que hay en el nombre del archivo sin revelárselo a otros que no conozcan el sistema. Por ejemplo, muchas personas tienen a mano un diccionario en su mesa de trabajo. Con este sistema se seleccionan una pagina al azar y se utiliza la primera palabra de la pagina. Supongamos que se ha elegido la pagina 67 y la primera palabra es cataclismo. Para el primer archivo que haya que cifrar se utilizaría como clave de acceso 67cataclismo01. El nombre del archivo debe de contener 671 como un código para recordar la palabra, como en BENTA671.WK1. El código no tiene utilidad sin la propia palabra. El segundo archivo podría ser utilizar la segunda palabra de la pagina, por ejemplo catacumba, siendo 67catacumba02, utilizando 672 con el archivo como código. Siguiendo este sistema se puede disponer de una amplia fuente de claves de acceso, que se pueden volver a recordar sin problemas.

Gestión continuada de un sistema de clave de acceso

Una vez que ha pasado por el trabajo y los gastos que suponen establecer un sistema de clave de acceso querrá estar seguro de que no se instaura simplemente y se deja correr a su aire. Los sistemas de claves de acceso pierden rápidamente su efectividad si no se refuerzan, mantienen y monitorizan. Los siguientes puntos pueden ser de interés cuando se haga una comprobación de los sistemas de seguridad con clave de acceso:

Asegúrese de que los empleados han recibido la formación adecuada sobre la selección de claves de acceso y las reglas relativas como la de una longitud mínima.

Desarrolle una lista de claves de acceso prohibidas para cruzarlas con las que estén en uso.

Actualice la lista de forma regular, y realice el cruce según períodos aleatorios.

Compruebe que existe un procedimiento para cancelar rápidamente las claves de acceso de las personas que se marchan de la organización. Compruebe que se cumple el procedimiento, eliminando todas las claves redundantes.

Compruebe que se modifica la clave de acceso de un empleado que pasa de un departamento a otro.

Si el sistema de clave de acceso no posee el requisito de una longitud de clave de acceso mínima, verifique la longitud de las claves que se estén usando.

Algunos sistemas de clave de acceso se pueden activar desde macros o secuencias de conexión, lo que significa que la clave de acceso puede estar en un archivo que no esté protegido.

Compruebe que los usuarios están utilizando dicho sistema y compruebe que es seguro.

Guarde un registro de las claves de acceso utilizadas previamente por los usuarios, listado por nombre de empleado.

Compruebe que dispone de una lista actualizada y fiable de usuarios autorizados.

Si los usuarios tienen dificultades en recordar sus claves de acceso, compruebe la seguridad de los procesos de asignación de nuevas claves y muéstrele a los empleados métodos seguros para recordar las claves.

Monitorice el número de intentos ilegales de acceder a los archivos y tenga en cuenta cualquier aumento.

Compare las claves utilizadas por los empleados de un mismo departamento. Asegúrese de que nadie utiliza la misma clave de acceso.