UNIVERSIDAD DON BOSCO

“DISEÑO E IMPLEMENTACIÓN DE PROTOTIPO DE LABORATORIO DE CRIPTOGRAFÍA”

TRABAJO DE GRADUACIÓN PREPARADO PARA LA FACULTAD DE INGENIERÍA

PARA OPTAR AL GRADO DE
INGENIERO EN ELECTRÓNICA

PRESENTADO POR:
VÍCTOR MANUEL ESCOBAR AVILÉS
RAFAEL ANTONIO GALLARDO LARA
CARLOS VLADIMIR ZELAYA ZOMETA

AGOSTO DE 2005
SAN SALVADOR, EL SALVADOR, CENTRO AMÉRICA

ÍNDICE

Índices de Tablas y Figuras.

INTRODUCCIÓN

El cifrado de información, es una actividad que se ha realizado desde la antigüedad dado que siempre ha sido necesario ocultar la información. Así como Julio Cesar desarrollo el código Cesar, el cual consistía en sustituir un alfabeto por otro, y durante la segunda guerra mundial, las comunicaciones se realizaron utilizando códigos secretos. Estos métodos de ocultar información dieron origen a la creación de diversas técnicas de cifrado, logrando con esto garantizar la integridad y seguridad en la transferencia de datos en las comunicaciones actuales.

El programa desarrollado, muestra didácticamente el funcionamiento de los procesos internos seguidos por varios de los algoritmos de cifrado más comunes, de tal manera que los usuarios comprenderán de forma clara y objetiva cómo se establece la seguridad de la comunicación en redes de información privada.

El proceso se desarrolla de forma tal, que se pueda ver el contenido de todos los pasos seguidos por los algoritmos. De esta forma, cada algoritmo se muestra como una secuencia de bloques con entradas y salidas, donde se observa el proceso que realiza cada uno de éstos en forma independiente; teniendo como resultado final, un texto parcialmente cifrado o descifrado. En dicho proceso gráfico se incluye la información de los datos de entrada y los de salida.

I MARCO TEÓRICO

1.1 ANTECEDENTES

El origen de los programas de computadora que se utilizan para el cifrado se inicia junto al nacimiento de las mismas computadoras. En la época de los sesentas y setentas, inicialmente, los algoritmos de cifrado de tipo secreto no eran mostrados por cuestiones de seguridad. Actualmente se puede contar con diversos programas de distribución gratuita, así como programas licenciados que permiten cifrar la información.

Estos programas pueden dividirse en dos tipos, estos son: los programas con fines educativos y los programas con objetivos puramente aplicativos, se tomará entonces como referencia comparativa los programas de tipo educativo de libre distribución.

Como primer punto es posible hablar de los programas encontrados en la página Web www.criptored.upm.es, como “Alleged RC-4” [Ang-04]. En este caso, es un laboratorio que muestra el funcionamiento de un solo algoritmo de cifrado, mostrando paso a paso el proceso seguido, sin proporcionar información sobre dicho proceso, es decir, que cuando se muestran los resultados intermedios del proceso, no es comprensible para el principiante. Otro punto que desfavorece este programa es que sólo cuenta con un ejemplo y no es posible que el usuario lo cambie, por lo tanto el resultado será siempre el mismo.

También se puede hacer referencia al programa de computadora llamado “CriptProgram” [Men-04], que cuenta con siete algoritmos de cifrado para realizar prácticas de laboratorio, además incluye varias herramientas de cálculo, sin embargo, presenta las mismas desventajas que “Alleged RC-4” ya que tampoco especifica el significado de los resultados en cada uno de los procesos, lo cual es vital para la comprensión de los algoritmos.

Otra aplicación para computadora es “Criptominilab” [Val-04]. Este programa cuenta con tres algoritmos de cifrado, pero nuevamente no se observa el proceso que se utiliza para cifrar los textos y solamente muestra las entradas y las salidas, por otro lado la interfase con el usuario es poco agradable, lo que dificulta el proceso de aprendizaje.

En general, a través del Internet es posible encontrar una gran variedad de sistemas de cómputo que permiten cifrar textos, sin embargo, todos los que fueron probados presentan las mismas desventajas: son poco didácticos y carecen de una forma fácil de comprender la operación de los algoritmos que tratan.

1.2 JUSTIFICACIÓN

Debido a la carencia de aplicaciones de cifrado que permitan de manera didáctica comprender este proceso, es útil crear un programa para computadora que facilite el aprendizaje en laboratorios, a un bajo costo y de forma portátil, que ayuden a una mejor comprensión de los algoritmos de cifrado comunes, proporcionando una herramienta a los alumnos y maestros, para dar apoyo a los conocimientos teóricos adquiridos.

1.3 OBJETIVO GENERAL

Crear un software didáctico capaz de ejecutar algoritmos de cifrado, simétricos y asimétricos, así como la implementación de funciones Hash, de manera que el usuario pueda comprender paso a paso cada uno de los procesos.

1.4 OBJETIVOS ESPECÍFICOS

- Crear un sistema que permita la comprensión de los algoritmos de cifrado simétricos y asimétricos más comunes, así como las funciones Hash.

- Crear un sistema de visualización del proceso seguido por cada algoritmo utilizado.

1.5 ALCANCES

- Entrega de una aplicación de fácil aprendizaje y uso para los estudiantes de la UDB.
- Realizar una aplicación que contenga algoritmos de los siguientes tipos:
    - Simétricos.
        - DES
        - IDEA
    - Asimétricos
        - RSA
        - Rabin.
    - Funciones Hash
        - SHA
        - MD5
    - Firmas Digitales.
    - IPsec.
- Brindar una explicación exhaustiva del procedimiento seguido al cifrar y descifrar mensajes.

1.6 LIMITACIONES

- Los algoritmos a implementar son los de uso mas común, porque la información que describe cómo funcionan es de fácil acceso.
- La entrada de datos para los procesos será texto sin formato.
- El software final a entregar será compatible únicamente con el sistema operativo Windows. Del mismo modo las herramientas de programación podrán ser: Ensamblador de Intel, Visual Basic, Visual C/C++.
- Para la demostración de los algoritmos no se construirá ningún hardware o circuito.

1.7 FORMAS DE VALIDACIÓN

- Cumplir con todas las acciones matemáticas y lógicas que implica un proceso de cifrado.
- Demostrar la adecuada distribución en el programa de todos los algoritmos y las partes que compongan a cada uno de ellos, permitiendo una fácil comprensión y un manejo coherente de las herramientas que se proporcionarán, tanto para usuarios familiarizados con el tema como los que no.
- Comparar los resultados del programa a presentar con otros existentes.

I I CLASIFICACIÓN DE ALGORITMOS

2. ALGORITMOS SIMÉTRICOS.

2.1 DES (DATA ENCRYPTION STANDARD). [LUC-04]

DES, es un esquema de cifrado simétrico desarrollado en 1977 por el Departamento de Comercio y la Oficina Nacional de Estándares de EEUU en colaboración con la empresa IBM, que se creó con objeto de proporcionar al público en general un algoritmo de cifrado normalizado para redes de computadoras. Está basado en la aplicación de todas las teorías de cifrado existentes hasta el momento, y fue sometido a las leyes de Estados Unidos.

Se basa en un sistema mono-alfabético, con un algoritmo de cifrado consistente en la aplicación sucesiva de varias permutaciones y sustituciones. Inicialmente el texto en claro a cifrar se somete a una permutación, con bloque de entrada de 64 bits (o múltiplo de 64), para posteriormente ser sometido a la acción de dos funciones principales, una función de permutación y otra de sustitución, en un proceso que consta de 16 ciclos.

En general, DES utiliza una clave simétrica de 64 bits, de los cuales 56 son usados para el cifrado, mientras que los 8 restantes son de paridad, y se usan para la detección de errores.

Como la clave efectiva es de 56 bits, son posibles un total de 2 elevado a 56 = 72.057.594.037.927.936 claves, es decir, unos 72.000 billones de claves, por

lo que la ruptura del sistema por fuerza bruta o diccionario es sumamente improbable, aunque no imposible si se dispone de suerte y una gran potencia de cálculo.

Los principales inconvenientes que presenta DES son:

La clave es corta, tanto que no asegura la fortaleza adecuada. Con la potencia de cálculo actual y futura de las computadoras se pone en riesgo la seguridad del algoritmo.

No permite longitud de clave variable, con lo que sus posibilidades de configuración son limitadas, además de permitir con ello la creación de restricciones legales.

La seguridad del sistema se ve reducida considerablemente si se conoce un número suficiente de mensajes, ya que existe un sistema matemático, llamado Criptoanálisis Diferencial, que puede en ese caso romper el sistema en 2 elevado a 47 iteraciones.

Entre sus ventajas cabe citar:

Es el sistema más extendido del mundo, el que más máquinas usan, el más barato y el más probado.

Es muy rápido y fácil de implementar.

Desde su aparición nunca ha sido roto con un sistema práctico.

2.1.1 CIFRADO EN DES. [LUC-04]

El algoritmo DES cifra la información por bloques, es decir, el texto en claro debe ser dividido en bloques de 64 bits que serán cifrados uno tras otro.

DES utiliza claves de 56 bits, aunque estas suelen distribuirse en forma de números de 64 bits. De estos 64 bits, uno de cada ocho, es utilizado como bit de paridad (64-8=56).

A grandes rasgos el algoritmo DES sigue el esquema de la figura 2.1.

La tabla de permutación A:

Después de recibir un bloque de entrada de 64 bits, el primer paso consiste en aplicar al bloque de entrada una permutación A mediante la tabla 2.1:

Tabla de permutación A							
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

Tabla 2.1

El bit 58 del bloque de entrada será colocado en el bit 1 del bloque de salida. El bit 50 del bloque de entrada, será colocado en el bit 2 del bloque de salida y así sucesivamente.

Las 16 iteraciones:

Después de la permutación A, el algoritmo DES realiza 16 iteraciones, las cuales hacen lo siguiente:

1. Los 64 bits de entrada se dividen en dos partes de 32 bits.
2. La mitad de la derecha (R) se introduce en una función (F) que se describirá más adelante.
3. Se realiza un XOR entre la salida de la función y la parte izquierda (L).
4. El resultado (32 bits) pasará a ser la parte derecha de la siguiente iteración, mientras que la parte derecha inicial pasa a ser la parte izquierda del siguiente ciclo.

$$L_{i+1} = R_i \qquad Ec.1$$

$$R_{i+1} = L_i \text{ xor f}\left(\mathrm{R_i}, k\right) \qquad Ec.2$$

En la última iteración (i=16) no se realizará el paso 4.

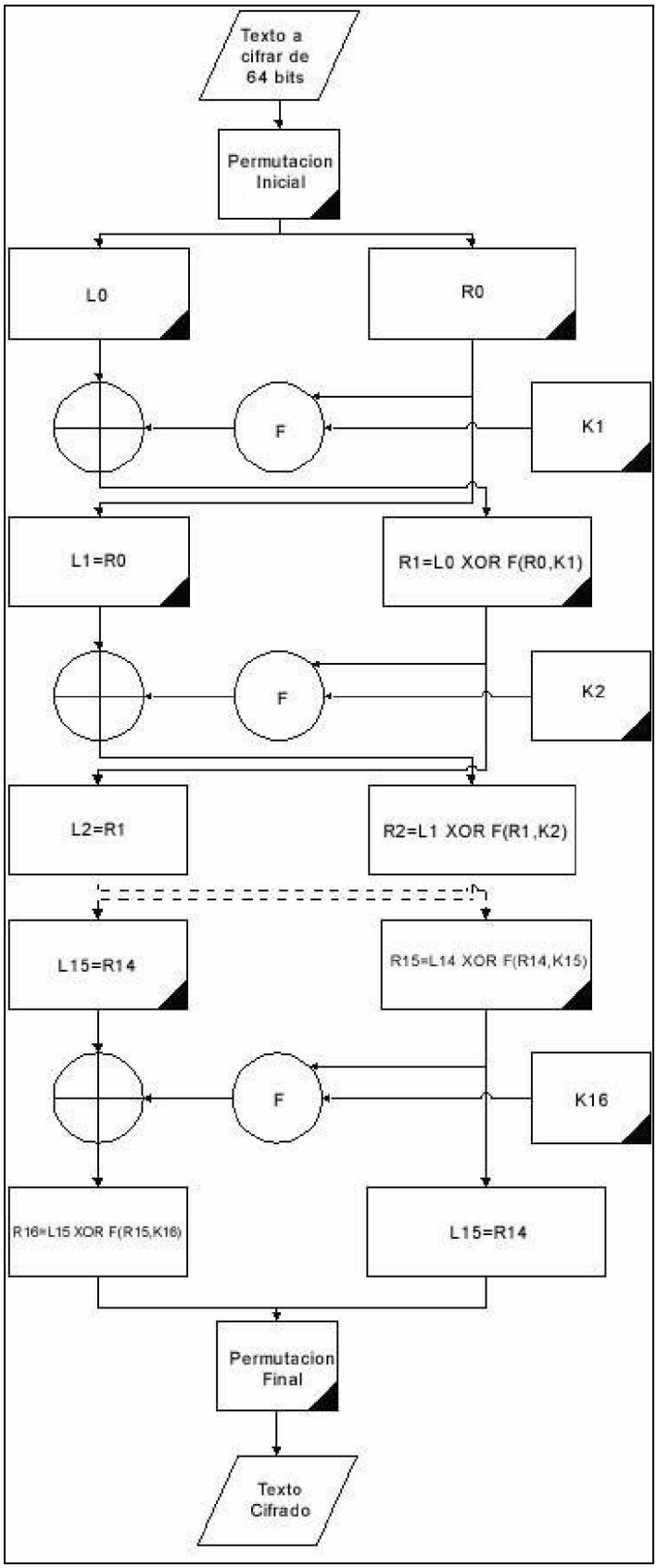


Figura 2.1 Algoritmo DES

La función F:

La función F se puede observar en la figura 2.2.

Los 32 bits de la parte derecha entran en la función F. El primer paso consiste en transformar la entrada de 32 bits a 48 bits mediante la tabla 2.2.

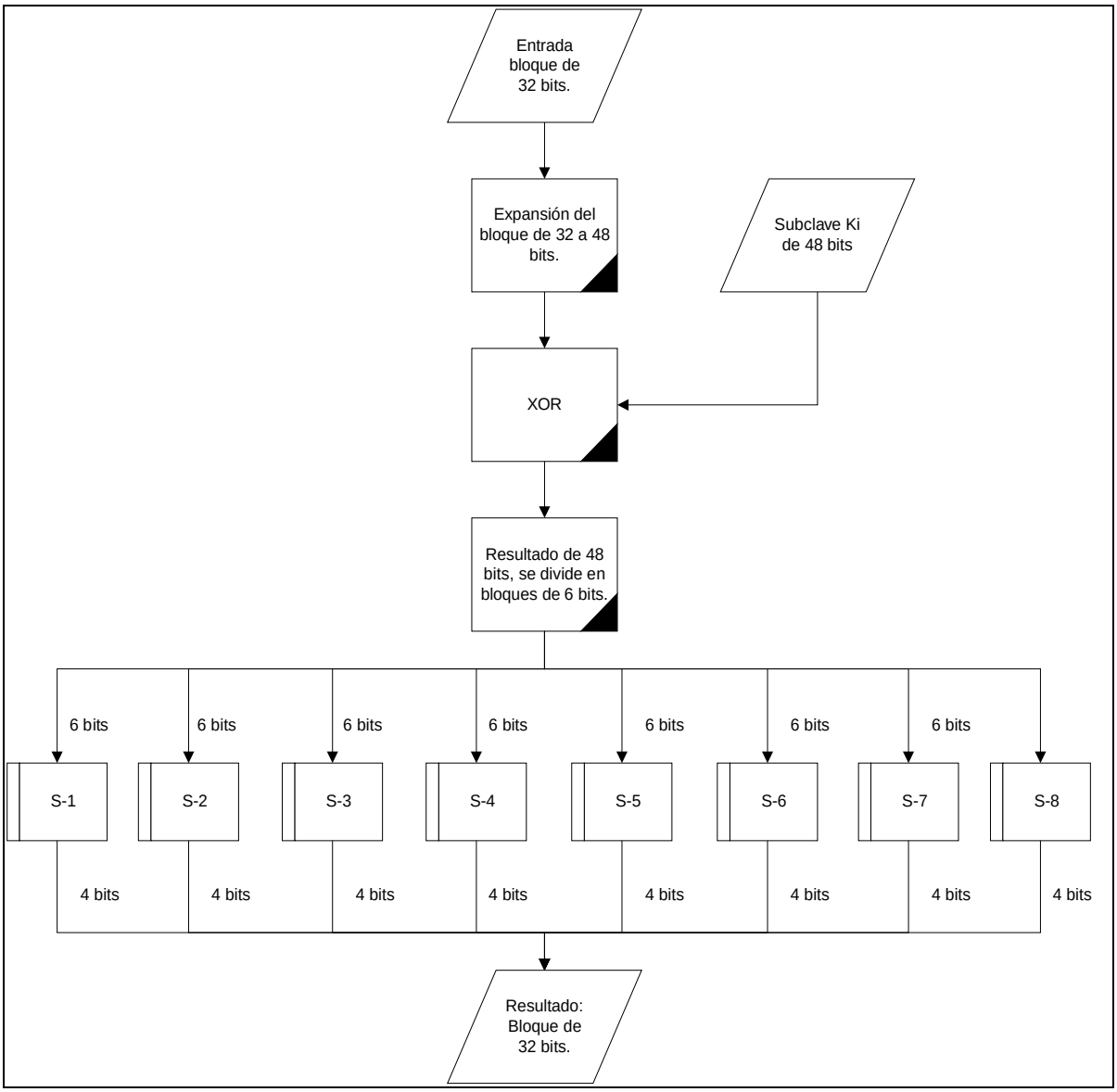


Figura 2.2. Secuencia de la función F.

Tabla de transformación C					
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

Tabla 2.2

A continuación se realiza un XOR con una subclave de 48 bits. Dado que la clave inicial era de 56 bits (representada como 64 bits) es necesario generar, a partir de esta, subclaves de 48 bits. Más adelante se describe detalladamente el proceso de generación de subclaves.

A continuación los 48 bits se dividen en bloques de 6 bits, cada uno de de los cuales es utilizado como entrada de lo que se conoce como una caja S. Observe la figura 2.2.

Los 6 bits que forman cada bloque b1, b2, b3, b4, b5 y b6, son utilizados para leer de cada caja S. b1 y b6 indican la fila mientras que b2, b3, b4, y b5 indican la columna. Es decir, si utilizamos como entrada 000011, b1b6 -> 01 indican la segunda fila y b2b3b4b5 -> 00001 indican la segunda columna, ya que las tablas inician desde cero. Las ocho cajas S existentes, se muestran en la tabla 2.3.

S-1															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S-2															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	4	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S-3															
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S-4															
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S-5															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S-6															
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S-7															
4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	7	5	0	15	14	2	3	12

S-8															
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Tabla 2.3

La tabla de permutación B.

De la misma manera, después de las 16 iteraciones se realiza otra permutación mediante la tabla 2.4.

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

Tabla 2.4

La tabla indica que el bit que ocupa la posición 40 del bloque de entrada, pasa a ocupar la posición 1 del bloque de salida y así sucesivamente.

2.1.2 Generación de subclaves K:

La secuencia de la figura 2.4 se realiza para cada una de las iteraciones:

El primer paso consiste en una permutación mediante la tabla 2.5.

Tabla de permutación D						
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

Tabla 2.5

Los 56 bits se dividen en dos partes de 28 bits. Cada una de estas partes rota hacia la izquierda un número X de bits en cada iteración, conforme a la tabla 2.6.

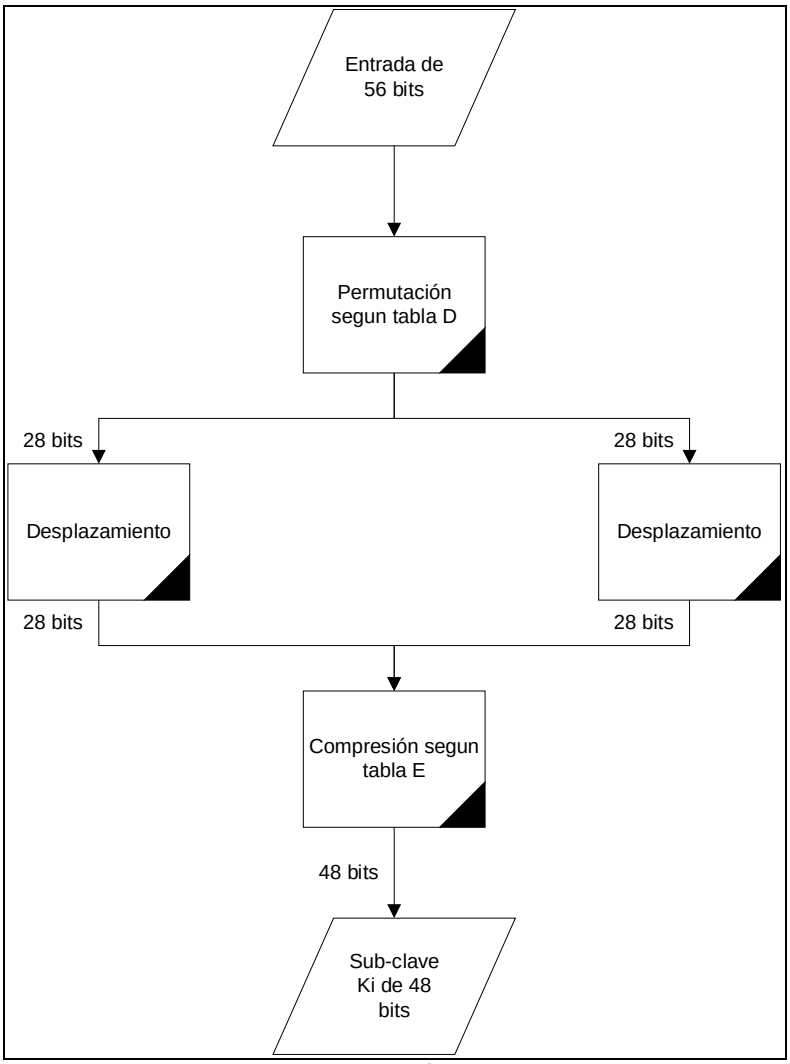


Figura 2.3. Generación de sub-claves

Iteración	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Desplazamiento	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Tabla 2.6. Desplazamiento de sub-claves

Finalmente, mediante la tabla 2.7 de compresión E los 56 bits (28+28) del resultado se comprimen formando una subclave de 48 bits (la utilizada en la función F).

Tabla de compresión E					
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

Tabla 2.7

2.1.3 Descifrado en DES

Una de las partes positivas de DES es que podemos descifrar los mensajes utilizando el mismo algoritmo que utilizamos para cifrar. La única diferencia consiste en el orden en que se aplican la subclaves, dado que en el descifrado se utilizan en orden inverso. Es decir, primero se utiliza la subclave k16, después la k15 ... k1.

2.1.4 Claves débiles en DES.[Sch-96]

Debido a la forma en que la clave inicial es modificada para formar subclaves, para cada ciclo del algoritmo, ciertas claves son débiles, esto debido a que los valores iniciales de cada ciclo son divididos en dos partes y cada mitad es desplazada de forma independiente; si todos los bits en cada mitad son 0 ó 1, entonces la clave usada para cada ciclo del algoritmo, es la misma para todos los ciclos. Esto puede ocurrir si la clave es, enteramente 1, enteramente 0 ó si en una mitad son 1 y en la otra mitad son 0.

Existen cuatro claves débiles conocidas para el algoritmo de cifrado DES. Son las siguientes [FRA-98]:

- 0101010101010101
- FEFEFEFEFEFEFEFE
- E0E0E0E0F1F1F1F1
- 1F1F1F1F0E0E0E0E

También cabe mencionar que existen seis parejas de claves semidébiles conocidas, las cuales generan las mismas dos claves durante todo el ciclo:

- (01FE01FE01FE01FE, FE01FE01FE01FE01)
- (1FE01FE01FE0F1E0, E01FE0F1E0F1E0F1E0)
- (01E001E001F101F1, E001E001F101F101)
- (1FFE1FFE0EFE0EFE, FE1FFE1FFE0EFE0E)
- (011F011F010E010E, 1F011F010E010E01)
- (E0FEE0FEF1FEF1FE, FEE0FEE0FEF1FEF1)

2.2 IDEA (International Data Encryption Algorithm). [Sch-96]

El algoritmo IDEA es bastante más joven que DES, pues data de 1992. Para muchos constituye el mejor y más seguro algoritmo simétrico disponible en la actualidad. Trabaja con bloques de 64 bits de longitud y emplea una clave de 128 bits. Como en el caso de DES, se usa el mismo algoritmo tanto para cifrar como para descifrar.

IDEA es un algoritmo bastante seguro, y hasta ahora se ha mostrado resistente a multitud de ataques, entre ellos el criptoanálisis diferencial. No presenta claves débiles, y su longitud de clave hace imposible en la práctica un ataque por la fuerza bruta.

Como ocurre con todos los algoritmos simétricos de cifrado por bloques, IDEA se basa en los conceptos de confusión y difusión, haciendo uso de las siguientes operaciones elementales (todas ellas fáciles de implementar):

1. XOR.
2. Suma módulo 2^{16}.
3. Producto módulo $2^{16} + 1$.

Este algoritmo es de libre difusión y no está sometido a ningún tipo de restricciones o permisos nacionales, por lo que se ha difundido ampliamente, utilizándose en sistemas como UNIX y en programas de cifrado de correo como PGP.

2.2.1 ALGOTIRMO IDEA.

El algoritmo IDEA consta de ocho rondas. Se dividirá el bloque X a codificar, de 64 bits, en cuatro partes X_1, X_2, X_3 y X_4 de 16 bits. Z_i será cada una de las 52 subclaves de 16 bits que se van a necesitar. Las operaciones que se llevaran a cabo en cada ronda son las siguientes:

1. Multiplicar X_1 por Z_1.
2. Sumar X_2 con Z_2.
3. Sumar X_3 con Z_3.
4. Multiplicar X_4 por Z_4.
5. Hacer un XOR entre los resultados del paso 1 y el paso 3
6. Hacer un XOR entre los resultados del paso 2 y el paso 4.
7. Multiplicar el resultado del paso 5 por Z_5.
8. Sumar los resultados de los pasos 6 y 7.
9. Multiplicar el resultado del paso 8 por Z_6.
10. Sumar los resultados de los pasos 7 y 9.
11. Hacer un XOR entre los resultados de los pasos 1 y 9.
12. Hacer un XOR entre los resultados de los pasos 3 y 9.
13. Hacer un XOR entre los resultados de los pasos 2 y 10.
14. Hacer un XOR entre los resultados de los pasos 4 y 10.

La salida de cada iteración serán los cuatro sub-bloques obtenidos en los pasos 11, 12, 13 y 14, que se convertirán en la entrada del siguiente ciclo, en el que emplearemos las siguientes seis subclaves, hasta un total de 48. Al final de todo se intercambian los dos bloques centrales (en realidad con eso se deshace el intercambio que se llevo a cabo en los pasos 12 y 13).

Después de la octava iteración, se realiza la siguiente transformación:

1. Multiplicar X_1 por Z_{49}.
2. Sumar X_2 con Z_{50}.
3. Sumar X_3 con Z_{51}.
4. Multiplicar X_4 por Z_{52}.

2.2.2 CLAVES DE IDEA.

Las primeras ocho subclaves se calculan dividiendo la clave de entrada en bloques de 16 bits. Las siguientes ocho se calculan rotando la clave de entrada 25 bits a la izquierda y volviendo a dividirla, y así sucesivamente.

Las subclaves necesarias para descifrar se obtienen cambiando de orden las Z_i y calculando sus inversas para la suma o la multiplicación. Puesto que 2^{16} + 1 es un número primo, nunca se obtendrá un cero como producto de dos números, por lo que no se necesita representar dicho valor. Cuando se calculen productos, se utilizara el cero para expresar el número 2^{16} -1 un uno seguido de 16 ceros. Esta representación es coherente puesto que los registros que se emplean internamente en el algoritmo poseen únicamente 16 bits.

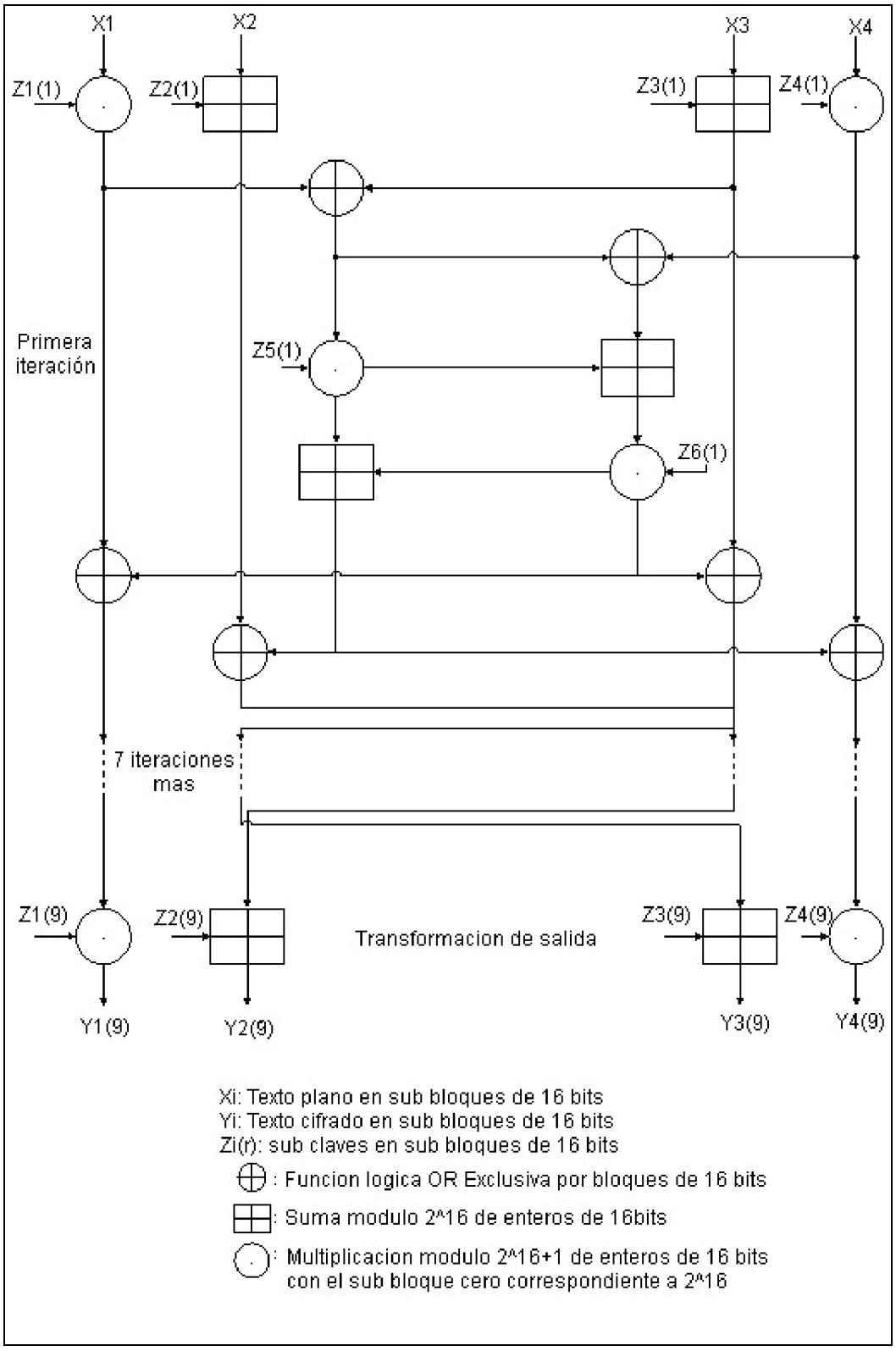


Figura 2.4 Algoritmo IDEA

3. Algoritmos de clave pública o asimétricos.

Al utilizar algoritmos de tipo simétricos, se sabe que la información cifrada estará segura mientras el atacante desconozca la clave, sin embargo para que esta información sea de utilidad para el receptor, es necesario que éste sepa cual es la combinación de bits con la que el mensaje fue cifrado, esta clave debe establecerse previamente entre los dos elementos que pretenden comunicarse, de lo contrario será necesario enviar la clave al receptor. Esta característica de los mensajes simétricos permite al criptoanalista dejar de lado el problema matemático que representa descifrar el mensaje e intentar interceptar la clave, puesto que ésta viaja en claro a través de los canales de comunicación.

Debido a esta falla de seguridad que presentan los algoritmos simétricos, Whitfield Diffie y Martin Hellman introducen en la década de los setentas el concepto de Algoritmos de clave asimétrica [Luc-04], con el objetivo de realizar comunicaciones a través de canales inseguros como el Internet.

Los algoritmos asimétricos fundamentan su seguridad en poseer dos claves, generadas a partir de un problema matemático que permite cifrar con una, que es pública y descifrar con la otra, conocida como privada, de tal forma que la clave con la que se realizó el cifrado no pueda resolver el problema matemático y el mensaje no pueda ser obtenido a partir de la misma.

Esta mecánica permite al emisor del mensaje cifrar utilizando la clave pública y al receptor descifrar utilizando la clave privada, cualquier otra incluyendo la clave pública producirá un mensaje erróneo.

La longitud de la clave de los algoritmos asimétricos es mucho mayor que los simétricos, mientras estos últimos utilizan claves de 128 bits, un equivalente a 16 letras en código ASCII, los sistemas asimétricos necesitan, para considerarse seguros longitudes de 1024 bits, que seria igual a escribir 128 letras en ASCII, es decir, que su longitud es ocho veces el de un simétrico.

3.1 Uso de los algoritmos asimétricos

Existen dos formas en las que se utilizan los sistemas de cifrado asimétrico, para proteger la información contra terceros cuando esta viaja por un canal inseguro y para autentificar la veracidad del mensaje recibido.

Protección de la información

Esta consiste en utilizar los sistemas asimétricos para ocultar la información cuando es enviada por medios inseguros, la mecánica con la que se realiza este proceso se inicia cuando el receptor “B” desea la información que posee el emisor “A”, entonces “B” genera dos claves a partir de un esquema matemático preestablecido, estas claves son conocidas como clave pública “KP” y clave privada “Kp”; el receptor “B” envía a través del canal de comunicación la clave pública “KP” la cual es recibida por el emisor “A”, éste utiliza la clave adquirida para cifrar la información que “B” desea adquirir, enviando ésta posteriormente por

el mismo canal. El receptor “B” toma el mensaje cifrado y lo decodifica utilizando la otra clave que el mismo genero, es decir la clave privada la cual está únicamente en manos de “B”.

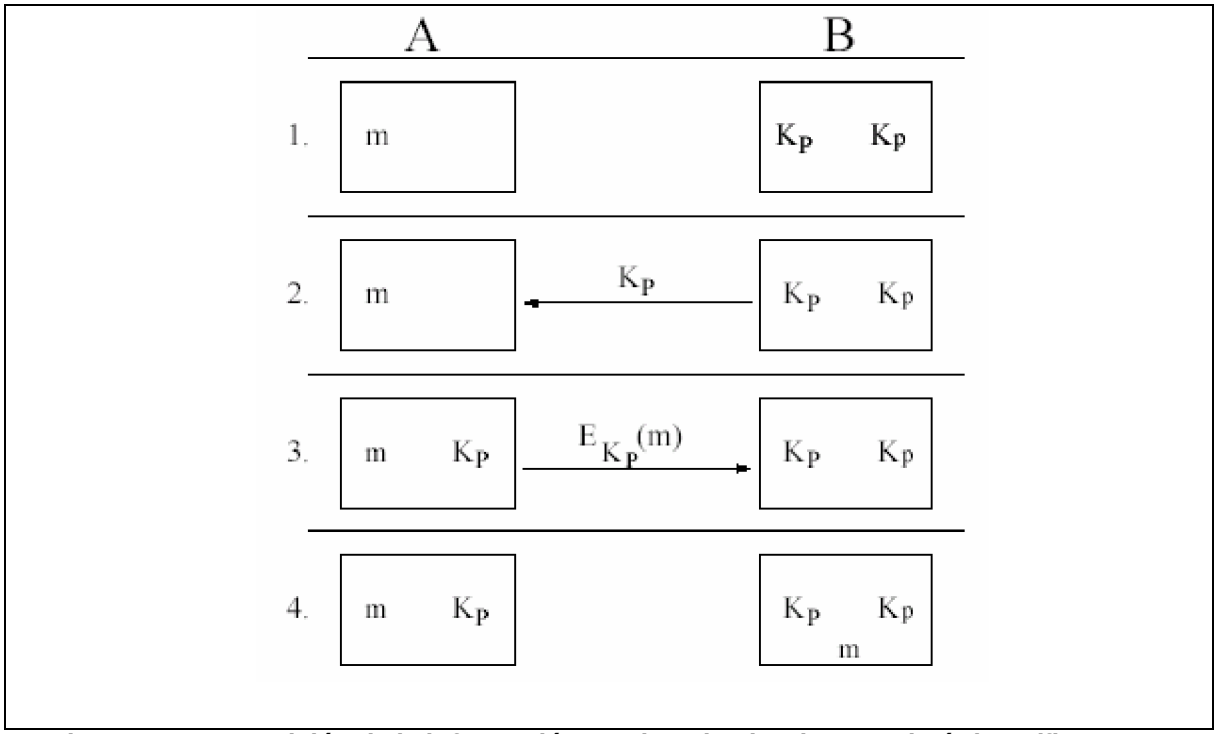


Figura 3.1. Transmisión de la información empleando algoritmos asimétricos (figura 12.1 [Luc-04])

El objetivo final de todo algoritmo de cifrado es mantener segura la información que será intercambiada entre emisor y receptor; sin embargo la aplicación de algoritmos de clave asimétrica genera grandes esfuerzos computacionales, lo que repercute en tiempos de cálculo demasiado grandes.

La ineficiencia que poseen estos sistemas para resolver el problema matemático en poco tiempo se resuelve codificando mensajes pequeños, lo cual no es práctico ya que muchos mensajes no son cortos, entonces es mejor utilizar

estos algoritmos solo para aquellos mensajes que no sobrepasen unas cuantas decenas de letras.

El limite en la longitud del mensaje a ocultar permite mezclar dos tipos de algoritmo, uno mas rápido pero cuya clave de descifrado debe viajar a través de los canales inseguros en forma clara, y otro mas lento cuya clave de descifrado esta solo en manos de receptor, por tanto es conveniente cifrar los mensajes utilizando sistemas simétricos y las claves de estos cifrados con algoritmos asimétricos.

Autenticación de la información

Esta se utiliza para determinar si la información recibida fue realmente enviada por el emisor “A” y no por un tercero que pudo alterar dicha información.

Se utiliza aplicando sistemas asimétricos en combinación de funciones resumen para generar una marca digital única (o muy difícil de duplicar) para el mensaje a intercambiar.

El esquema es similar al de la protección de la información, puesto que siempre existe un sujeto “B”, este trasmite un mensaje el cual es recibido por “A” el cual ha obtenido previamente la clave pública “KP”. En este momento “A” posee el mensaje pero no es seguro que el contenido de este mensaje provenga de “B” puesto que fue expuesto a un canal de comunicación inseguro y pudo ser modificado por un tercero. “B” utiliza una función matemática que produce un resumen del mensaje enviado a “A”, este resumen es cifrado utilizando la clave privada “Kp” y lo envía para que lo reciba “A”. Hasta el momento “A” posee un

mensaje y un resumen cifrado. Utilizando la clave pública, se descifra la información, obteniendo el resumen del mensaje enviado por “B”; Ahora “A” genera con la misma función un resumen del mensaje recibido, puesto que la probabilidad que una función resumen con diferentes mensajes produzcan el mismo resultado es muy baja, solo el resultado del mensaje enviado por “B” será igual al resumen que posee “A”.

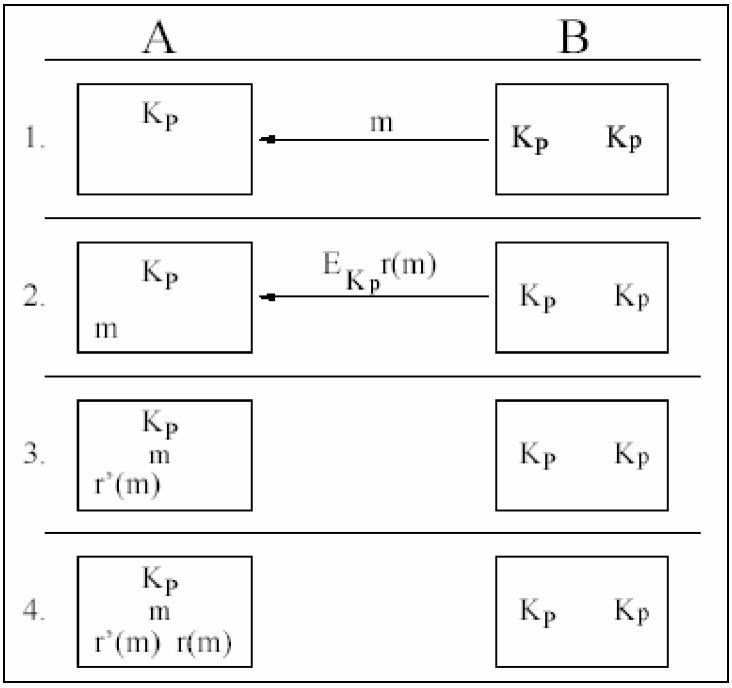


Figura 3.2. Autenticación de la información con algoritmos asimétricos

(Figura 12.2 [Luc-04])

Debe tomarse en cuenta que el uso de las claves pública y privada dependerá del tipo de aplicación que se este implementado, para el cifrado de la información se utilizará la clave pública “KP” para cifrar y la clave privada “Kp” para descifrar. Si la aplicación es para autenticar un mensaje el uso de las claves será inverso, la clave pública será para descifrar y la privada para cifrar.

3.2 Familias de algoritmos asimétricos

La gran variedad de algoritmos asimétricos se debe a los diferentes problemas matemáticos que se resuelven para obtener las claves y para cifrar los mensajes [Ang-04], con esta base se clasifican en:

- PLDE (problema del logaritmo discreto elíptico)
- PLD (problema del logaritmo discreto)
- PFE (problema de factorización entera)

3.3 Algoritmo de cifrado RSA

Debe su nombre a sus creadores Ronal Rivest, Adi Shamir y Leonard Adleman [Luc-04], éste basa su seguridad en la dificultad de factorizar grandes números, es decir que es del tipo PFE (problema de factorización entera). La clave pública y privada se obtiene del producto de dos números primos grandes y se calcula de la siguiente forma:

- Se eligen de forma aleatoria 2 números primos grandes que sean de la misma longitud y que numéricamente sean lejanos, a los cuales llamaremos “p” y “q” con los que se calculará su producto llamado “n”[Ang-04]

$$n = pq \qquad Ec.3$$

- Con este producto se escoge un número “e” que sea primo relativo a (p-1)(q-1) que cumpla 2^e>n.

- se calcula la inversa de *e mod ((p-1)(q-1))*, conocido como el teorema de Fermat [Ang-04], con lo que se obtiene un número “d” el cual cumple:

$$de \equiv 1 \bmod((p-1)(q-1)) \qquad Ec.4$$

Al resolver el problema matemático anterior, se obtienen dos números “e” y “d” de los cuales (e,n) será la clave pública y en consecuencia (d,n) la clave privada. Para cifrar se utilizará la siguiente ecuación:

$$C = m^e \pmod n \qquad Ec.5$$

Donde “m” es el mensaje a cifrar y “C” es el resultado de la operación matemática, es decir, el mensaje cifrado.

Para descifrar se utiliza la misma ecuación pero con la clave privada, así:

$$m = C^d \pmod n \qquad Ec.6$$

Note que para un atacante será muy difícil determinar el valor de (d,n) puesto que éste es producto de factorizar los números “p” y “q”, que no están en su poder, sin embargo para calcular las claves públicas y privadas se recomienda que se utilicen claves de 768 bit para actividades personales, 1024 bits para actividades corporativas y de 2048 bits para actividades de alto riesgo [Ang-04].

3.4 Algoritmo de cifrado de RABIN

El algoritmo de RABIN pertenece a la misma familia PFE (problema de factorización entera), sin embargo éste basa su seguridad no en el cálculo de los factores de un número sino en determinar las raíces cuadradas del módulo de un número compuesto.

Por ser de tipo asimétrico, es necesario poseer dos claves, una pública y una privada, que se utilizarán de la misma forma que las del RSA; su cálculo se realiza de la siguiente forma:

- Se eligen dos números primos “p” y “q”, ambos congruentes con 3(mod4) cuyos dos últimos bits sean 1 y se calcula el producto de estos.

$$n = pq \qquad Ec.7$$

De estos productos realizados los números “p” y “q” serán tomados como la clave privada, mientras que su producto “n” será la pública.

Con las claves previamente elegidas es posible iniciar la codificación utilizando la ecuación:

$$C = m^2 \bmod n \qquad Ec.8$$

Y para decodificar el mensaje debe resolverse el conjunto siguiente de ecuaciones.

$$m_1 = c^{\frac{p+1}{4}} \bmod p \qquad Ec.9$$

$$m_2 = \left(p - c^{\frac{p+1}{4}} \right) \bmod p \qquad Ec.10$$

$$m_3 = c^{\frac{q+1}{4}} \bmod q \qquad Ec.11$$

$$m_4 = \left(q - c^{\frac{q+1}{4}} \right) \bmod q \qquad Ec.12$$

Los valores de m_1 a m_4 no son el mensaje. Debido a la naturaleza matemática del algoritmo de RABIN surgen 4 posibles mensajes, los cuales pueden estar constituidos así:

$$m_a = (am_1 + bm_3) \bmod n \qquad Ec.13$$

$$m_b = (am_1 + bm_4) \bmod n \qquad Ec.14$$

$$m_c = (am_3 + bm_3) \bmod n \qquad Ec.15$$

$$m_d = (am_2 + bm_4) \bmod n \qquad Ec.16$$

Donde a y b deben calcularse de tal manera que cumplan la siguiente ecuación:

$$a = q\left(q^{-1} \bmod p\right) \qquad Ec.17$$

$$b = p\left(p^{-1} \bmod q\right) \qquad Ec.18$$

Al descifrar los cuatro mensajes, no existe ninguna forma de saber cual es el original, puesto que los cuatro mensajes m_a, m_b, m_c y m_d cumplen con la solución del problema por tanto el emisor del mensaje debe colocar alguna señal que le indique al receptor cual es el mensaje verdadero [Luc-04].

4. Funciones Hash (funciones resumen)

Existen diferentes formas de mantener la seguridad de un sistema, Por ejemplo: se pueden almacenar las contraseñas de los usuarios en forma de texto; sin embargo, quedarían expuestas porque serían legibles. Se puede adicionar seguridad si en lugar de colocar las contraseñas directamente, se utilizan algoritmos criptográficos, es decir, la contraseña es modificada de su forma original (cifrada) y luego almacenada.

Con estos métodos, las contraseñas quedan expuestas y probablemente se podrian descifrar y obtener sus valores originales. La utilización de una función hash hace posible que las contraseñas no queden expuestas [Per-00].

Las funciones hash se utilizan para “comprimir” un mensaje de longitud variable tomado como entrada a uno de tamaño fijo (valor hash) producido como salida, reduciendo el tiempo de generación de firmas por algoritmos de firmas digitales [Sta-98].

Cuando un usuario desea acceder al sistema, se le aplica una función hash a la contraseña dada por él y el valor hash obtenido se compara con el valor almacenado en la base de datos, si son iguales, el usuario tiene acceso. Si la contraseña almacenada en la base de datos estuviera “comprometida”, sería difícil obtener la contraseña dada por el usuario.

Para explicar cómo trabaja una función Hash o resumen se puede observar el siguiente ejemplo de una función Hash genérica.

Definamos la función Resumen como:

Resumen = Σ((primer numero+segundo numero)x(tercer numero–cuarto numero))

Texto	**U**	**N**	**i**	**V**	**e**	**R**	**S**	**i**	**d**	**a**	**D**
Valor numérico	85	110	105	118	101	114	115	105	100	97	100
Resultado de la función	**-2405**				**2150**				**6304**		
Texto	**D**	**O**	**n**		**B**	**O**	**S**	**c**	**o**		
Valor numérico	68	111	110	32	66	111	83	99	111		
Resultado de la función		**-7514**			**-2328**						

Resumen = -2405 + 2150 + 6304 - 7514 – 2328

Resumen = -3793

Dicho resumen identifica mediante la función establecida para el mensaje. Se puede además, comprobar cual será la variación en el resumen si se modifica el mensaje:

Texto	**U**	**N**	**i**	**V**	**e**	**R**	**S**	**i**	**d**	**a**	**D**
Valor numérico	85	110	105	118	101	114	115	105	100	97	100
Resultado de la función	**-2405**				**2150**				**6304**		
Texto	**D**	**O**	**n**		**B**	**O**	**S**	**Ć**	**o**		
Valor numérico	68	111	110	32	66	111	83	263	111		
Resultado de la función		**-7514**			**29488**						

Resumen = - 2405 + 2150 + 6304 - 7514 + 29488

Resumen = 28023

Como puede observarse, una pequeña variación en el mensaje puede afectar el resumen en gran medida.

4.1 Aplicaciones de las funciones Hash.

Son tres las aplicaciones principales de las funciones hash [Kam-98]:

- Contraseñas: Las funciones Hash son ampliamente usadas para almacenar contraseñas. Por su característica de irreversibilidad, almacenar un valor Hash de una contraseña es más seguro que almacenarla en forma criptográfica.

- Firmas Digitales: Realizar operaciones de firmas digitales sobre mensajes grandes puede consumir mucho tiempo por los algoritmos de firmas digitales. En su lugar, al mensaje se le aplica la función Hash y el algoritmo de firma digital se aplica al valor Hash obtenido de menor tamaño.

- Integridad y autenticación de un mensaje: Un mensaje puede ser considerado íntegro si su valor Hash ya fue calculado antes de cualquier transmisión. Este valor es comparado con el valor Hash del mensaje recibido.

4.2 Características de las Funciones Hash.

Todos los números resumen generados con un mismo método tienen el mismo tamaño sea cual sea el texto utilizado como base.

- Dado un texto base, es fácil y rápido (para una computadora) calcular su número resumen.
- Es imposible reconstruir el texto base a partir del número resumen.
- Es imposible que dos textos base diferentes tengan el mismo número resumen.

4.3 Diferentes algoritmos Hash.

Varias funciones Hash han sido desarrolladas tratando de mejorar las versiones anteriores para tener una mayor seguridad y evitar que se lleven a cabo ataques con éxito.

- MD4 (Message Digest). Fue Inventado por Ron Rivest de la corporación de Seguridad RSA (RSA Security, Inc.). Produce un valor Hash de 128-bits. Se realiza una manipulación de bits para obtener el valor Hash, obteniéndolo de forma rápida, provocando que sea más riesgoso en un ataque. Se considera un estándar de Internet (RFC-1320) [Sta-98].

- MD5 Mensajes Digitales. Es una extensión de MD4. Produce como salida un valor Hash de longitud de 128-bits [Sta-98]. MD5 tiene optimizaciones y

sugerencias de varios revisores. La obtención del valor hash es lento pero considerado más seguro. Está especificado como un estándar de Internet (RFC-1321 [Riv-92]). Es usado por PGP (Pretty Good Privacy, Privacidad Considerada Buena).

- SHA-1 (Secure Hash Algorithm, Algoritmo Hash Seguro). Diseñado por NIST (National Institute of Standards and Technology), produce un valor hash de 160-bits. Su diseño tiene mucha relación con MD5 pero con ciertas diferencias (ej. salida de 160-bits) [Sta-98]. Se considera más seguro que MD4 y MD5 por su longitud de tamaño. También está considerado como un estándar [FIPS PUB 180-1].
- RIPEMD-160. Es una función Hash criptográfica diseñada por Hans Dobbertin, Antoon Bosselaers, and Bart Preneel dentro de un proyecto llamado RIPE (Race Integrity Primitives Evaluation), de 1988-1992. Produce una salida de 160 bits. Se intenta que sea usada como un reemplazo seguro de las funciones Hash de 128-bits como MD4, MD5.

4.3.1 SHA-1 (Secure Hash Algorithm).[Kam-98]

Este estándar fue introducido el 17 de Abril de 1995, y especifica un algoritmo Hash seguro para calcular la representación de un mensaje o archivo de datos. Para un mensaje de una longitud máxima de 2^{64} bits como entrada, SHA-1 produce como salida una cadena de 160 bits llamada “Mensaje Resumen”. El mensaje resumen puede ser introducido a un algoritmo de firma digital (o DSA por sus siglas en ingles), el cual genera o verifica la firma del mensaje. Firmar el

mensaje resumen en lugar del mensaje original provee eficiencia en el proceso, debido a que el resumen es, usualmente, mucho menor en tamaño que el mensaje original. El mismo algoritmo Hash con el que se firmó el mensaje debe ser utilizado por el receptor para verificar la firma digital.

El SHA-1 es llamado seguro debido a que computacionalmente no es factible encontrar un mensaje que corresponda a un mensaje resumen dado, o encontrar dos diferentes mensajes que produzcan el mismo mensaje resumen.

Cálculo:

SHA-1 es usado para calcular el resumen de un mensaje o archivo de datos suministrado como entrada del algoritmo. El mensaje o archivo de dato debe ser considerado como una cadena de bits. La longitud del mensaje es el número de bits del mensaje (Un mensaje vacío tiene longitud cero); si el número de bits en el mensaje es un múltiplo de 8, se puede representar el mensaje mediante notación hexadecimal.

SHA-1 procesa secuencialmente bloques de 512 bits cuando se calcula el mensaje resumen, por lo que se debe recurrir a una herramienta matemática para lograr que la longitud del mensaje sea un múltiplo de 512. Esta herramienta es el Relleno de Mensaje (Message Padding), realiza los siguientes paso:

- Se agrega un 1 lógico al final de la cadena original. Por ejemplo:

Mensaje original:	abc
Mensaje binario:	01100001 01100010 01100011
Agregando 1 lógico:	01100001 01100010 011000111

- Se agregan 0 lógicos, de acuerdo a la cantidad de bits después del primer paso y hasta completar 448 bits.

  Tomando el ejemplo del paso anterior: 448 – 25 = 423 ceros (se presenta en formato de palabras).

  61626380 00000000 00000000 00000000
  00000000 00000000 00000000 00000000
  00000000 00000000 00000000 00000000
  00000000 00000000

- Obtener, en notación de 2 palabras, el tamaño del mensaje original y agregarlos en los últimos 64 bits del bloque. Nótese que si la longitud del mensaje original es menor de 2^{32} bits, la primera palabra es 0.

  Siguiendo con el primer ejemplo, la longitud de la cadena original es 12, lo que en notación de dos palabras es 00000000 0000000C, por lo tanto:

  61626380 00000000 00000000 00000000
  00000000 00000000 00000000 00000000
  00000000 00000000 00000000 00000000
  00000000 00000000 00000000 0000000C

Puede observarse, que el mensaje rellenado, contendrá 16 x n palabras, para n>0. El mensaje rellenado estará distribuido como una secuencia de n bloques M_1, M_2, M_3,…. M_n, donde cada M_i contiene 16 palabras (512 bits), y M_1 contiene los primeros 512 bits del mensaje. El mensaje rellenado final es usado para calcular el mensaje resumen.

Una secuencia de funciones lógicas es usada en SHA-1, cada una de ellas utiliza tres palabras de 32 bits (B, C, D) como entrada y produce una palabra de 32 bits como salida. Esta secuencia se define como:

$$f(B,C,D) = \begin{cases} (B \text{ and } C) \text{ or } (\overline{B} \text{ and } D) & \text{para } 0 \le t \le 19 \\ B \text{ xor } C \text{ xor } D & \text{para } 20 \le t \le 39 \\ (B \text{ and } C) \text{ or } (B \text{ and } D) \text{ or } (C \text{ and D}) & \text{para } 40 \le t \le 59 \\ B \text{ xor C xor D} & \text{para } 60 \le t \le 79 \end{cases} \qquad Ec.19$$

Donde t es el ciclo de proceso.

Además de las funciones anteriores, son utilizadas 4 constantes, las cuales son:

$K = 5A827999$, *para* $t \le 19$.

$K_t = 6ED9EBA1$, *para* $20 \le t \le 39$.

$K_t = 8F1BBCDC$, *para* $40 \le t \le 59$.

$K_t = CA62C1D6$, *para* $60 \le t \le 79$.

El cálculo utiliza:

- Dos buffer de cinco palabras cada uno, los cuales son etiquetados como: “A, B, C, D, E” y “H0, H1, H2, H3, H4, H5”
- Una secuencia de 80 palabras de 32 bits cada una, la cual es etiquetada como: W_0, W_1,….. W_{79}.
- También se emplea un buffer Temporal de una sola palabra

Para generar el mensaje resumen, los bloques de 16 palabras (512 bits): M_1, M_2,… M_n, son procesados en orden y cada M_i involucra 80 pasos, los cuales son:

1. Se inicializa el buffer H:

   H_0 = 67452301

   H_1 = EFCDAB89

   H_2 = 98BADCFE

   H_3 = 10325476

   H_4 = C3D2E1F0.

2. Se toma M_1 y se divide en 16 palabras, desde W_0 hasta W_{15}, donde W_0 es la palabra más a la izquierda.

3. Hay que recordar que la secuencia W tiene 80 palabras. Desde t=16 hasta 79, Haciendo: $W_t = S^1(W_{t-3}$ XOR W_{t-8} XOR W_{t-14} XOR $W_{t-16})$. Con lo que la secuencia de 80 palabras está completa.

   En este paso se utiliza la función:

   $$W_t = S^n(X) = (X<<n) \text{ or } (X>>32-n) \qquad Ec.20$$

   Donde X es una palabra (32 bits) y n es un entero entre 0 y 32. Además: X<<n, se obtiene descartando los n bits que están mas a la izquierda y completando con 0's a la derecha (el resultado siempre será de 32 bits). La operación X>>n es obtenida de descartar los n bits que están mas a la derecha de X y rellenando el resultado con n ceros a la izquierda.

4. Haciendo que: $A = H_0$, $B = H_1$, $C = H_2$, $D = H_3$, $E = H_4$.

5. Comienza el ciclo de 80 pasos por cada bloque M. Desde t=0 hasta 79 se hace:

   $$TEMP = S^5(A) + f_t(B,C,D) + E + W_t + K; \qquad Ec.21$$

   E = D; D = C; C = S^{30}(B); B = A; A = TEMP;

6. Por ultimo:

   $H_0 = H_0 + A$, $H_1 = H_1 + B$, $H_2 = H_2 + C$, $H_3 = H_3 + D$, $H_4 = H_4 + E$.

7. Si el mensaje tiene más de un bloque, se repite desde el paso 2, con los nuevos valores de H, resultados de procesar el bloque anterior.
8. El mensaje resumen será la concatenación de los valores H finales después de procesar los n bloques: $H_0H_1H_2H_3H_4$.

4.3.2 MD5 (Message Digest Algorithm). [Riv-92]

Definición.

Este algoritmo toma como entrada un mensaje de longitud arbitraria y produce una salida de 128 bits, la cual representa la huella o un mensaje resumen de la entrada. Es además, computacionalmente improbable producir dos mensajes que tengan el mismo mensaje resumen, o de obtener el mensaje original a partir de una mensaje resumen.

El algoritmo MD5 fue diseñado para ejecutarse muy rápidamente en computadoras personales de 32 bits, además de que no requiere grandes tablas de substitución y puede ser codificado de forma muy compacta.

Este algoritmo es una extensión del algoritmo para resumir MD4. MD5 es un poco más lento que MD4 pero se considera mas seguro. MD5 se diseñó porque las revisiones críticas existentes señalaron que MD4 fue quizás adoptado por su rapidez y no por su seguridad. Debido a que MD4 fue diseñado para ser excepcionalmente rápido se encuentra “en el filo” en termino de riesgos de ataques cripto-analíticos exitosos. MD5 sacrifica un poco de velocidad en su cálculo por mucha más seguridad. Incorpora algunas recomendaciones hechas

por varios analizadores y contiene varias optimizaciones. El Algoritmo MD5 se hace de dominio público para revisiones y posibles adopciones como un estándar.

Cálculo.

Se empieza con la suposición que tenemos un mensaje de b-bits como entrada, y que deseamos encontrar el resumen. b es un entero cualquiera no negativo; puede ser 0, no es necesario que sea múltiplo de 8 y puede ser arbitrariamente largo. Podemos imaginar los bits del mensaje escrito así:

$m_0\ m_1 \ldots\ m_{b-1}$

Los siguientes 5 pasos son realizados para calcular el mensaje resumen del mensaje dado:

- Agregar bits para completar

  El mensaje es completado (extendido) para que su longitud (en bits), sea congruente con 448 modulo 512. Completar el mensaje siempre se lleva a cabo, aun cuando la longitud del mensaje ya sea congruente con 448 modulo 512.

  El relleno del mensaje se realiza así:

  - Un 1 lógico (1 bit) es agregado al final del mensaje.
  - Se agregan 0's lógicos hasta que la longitud en bits del mensaje completado sea congruente con 448 modulo 512. En resumen, al menos un bit debe ser agregado al mensaje, y como máximo 512.

- Agregar la longitud.

  La representación en 64 bits de b (longitud de mensaje original) es agregada al resultado del paso anterior. Si la longitud es menor que 2^{32}, entonces la primera palabra será cero.

  En este punto el mensaje resultante tiene una longitud que es un múltiplo exacto de 512 bits, o mejor dicho, este mensaje tiene una longitud que es un múltiplo de 16 palabras (32 bits).

- Inicializar buffer MD.

  Un buffer de 4 palabras (A, B, C, D) es usado para calcular el mensaje resumen. Cada una de las palabras es un registro de 32 bits. Estos registros son inicializados con los siguientes valores hexadecimales:

  A = 67452301

  B = EFCDAB89

  C = 98BADCEF

  D = 10325476

- Procesar el mensaje en bloques de 16 palabras.

  Primero se definen 4 funciones auxiliares, las cuales toman como entrada 3 palabras cada una, y producen como salida una palabra de 32 bits. Estas funciones son:

$$f(X,Y,Z) = XY \text{ or } \overline{X}Z \qquad Ec.22$$

$$G(X,Y,Z) = XZ \text{ or } \overline{Z} \qquad Ec.23$$

$$H(X,Y,Z) = X \text{ xor } Y \text{ xor } Z \qquad Ec.24$$

$$I(X,Y,Z) = Y \text{ xor (X or } \overline{Z}) \qquad Ec.25$$

Este paso utiliza una tabla de 64 elementos, construida con la función seno. La tabla se construye así:

$$T[i] = Parte\ Entera(4294967296 \times Abs(Sin(i))) \qquad Ec.26$$

Donde i esta en radianes.

Luego se hace lo siguiente, para cada bloque de 16 palabras, desde 0 hasta N/16-1 donde N es el número de bloques.

- Copiar el bloque i en X y hacer desde j=0 hasta 15

$$X[j] = M[i * 16+j] \qquad Ec.27$$

- Salvar los buffer A,B,C,D

$$AA = A \qquad Ec.28$$

$$BB = B \qquad Ec.29$$

$$CC = C \qquad Ec.29$$

$$DD = D \qquad Ec.30$$

- Se realiza una primera ronda donde la expresión [abcd k s i] denota la operación:

$$a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s) \qquad Ec.31$$

y se realizan las siguientes 16 operaciones.

[ABCD 0 7 1]	[DABC 1 12 2]	[CDAB 2 17 3]	[BCDA 3 22 4]
[ABCD 4 7 5]	[DABC 5 12 6]	[CDAB 6 17 7]	[BCDA 7 22 8]
[ABCD 8 7 9]	[DABC 9 12 10]	[CDAB 10 17 11]	[BCDA 11 22 12]
[ABCD 12 7 13]	[DABC 13 12 14]	[CDAB 14 17 15]	[BCDA 15 22 16]

- En la segunda ronda la expresión [abcd k s i] representa la operación de la ecuación 32 y se efectúan los 16 cálculos siguientes

$$a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s) \qquad Ec.32$$

[ABCD 1 5 17]	[DABC 6 9 18]	[CDAB 11 14 19]	[BCDA 0 20 20]
[ABCD 5 5 21]	[DABC 10 9 22]	[CDAB 15 14 23]	[BCDA 4 20 24]
[ABCD 9 5 25]	[DABC 14 9 26]	[CDAB 3 14 27]	[BCDA 8 20 28]
[ABCD 13 5 29]	[DABC 2 9 30]	[CDAB 7 14 31]	BCDA 12 20 32]

- En la tercera ronda el termino [abad k s t] denotara la expresión de la ecuación 33.

$$a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s) \qquad Ec.33$$

[ABCD 5 4 33]	[DABC 8 11 34]	[CDAB 11 16 35]	[BCDA 14 23 36]
[ABCD 1 4 37]	[DABC 4 11 38]	[CDAB 7 16 39]	[BCDA 10 23 40]
[ABCD 13 4 41]	[DABC 0 11 42]	[CDAB 3 16 43]	[BCDA 6 23 44]
[ABCD 9 4 45]	[DABC 12 11 46]	[CDAB 15 16 47]	[BCDA 2 23 48]

- En la ultima ronda la expresión [abad k s t] representa la operación de la ecuación 34.

$$a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s) \qquad Ec.34$$

[ABCD 0 6 49]	[DABC 7 10 50]	[CDAB 14 15 51]	[BCDA 5 21 52]
[ABCD 12 6 53]	[DABC 3 10 54]	[CDAB 10 15 55]	[BCDA 1 21 56]
[ABCD 8 6 57]	[DABC 15 10 58]	[CDAB 6 15 59]	[BCDA 13 21 60]
[ABCD 4 6 61]	[DABC 11 10 62]	[CDAB 2 15 63]	[BCDA 9 21 64]

- Al finalizar las 80 operaciones de los cuatro pasos anteriores se realizan las siguientes sumas, lo que es equivalente a incrementar el valor que tenían los registros antes de los calculos.

$$A = A + AA \qquad Ec.35$$

$$B = B + BB \qquad Ec.36$$

$$C = C + CC \qquad Ec.37$$

$$D = D + DD \qquad Ec.38$$

El mensaje resumen producido a la salida es A, B, C, D. Se empieza con el byte de menor orden de A y se termina con el byte de mayor orden de D.

Mensaje Resumen = ABCD

I I I APLICACIONES

5. Firma digital

El desarrollo de grandes redes ha propiciado la creciente cantidad de mensajes entre los usuarios, cuyos contenidos pueden ser desde un simple saludo hasta una transacción de capital. Como se muestra en la figura 5.1, todos estos mensajes se encuentran expuestos a cualquier modificación antes de llegar a su destino final, por lo cual el emisor debe proteger su información y asegurarse que ésta mantenga su integridad en todo momento.

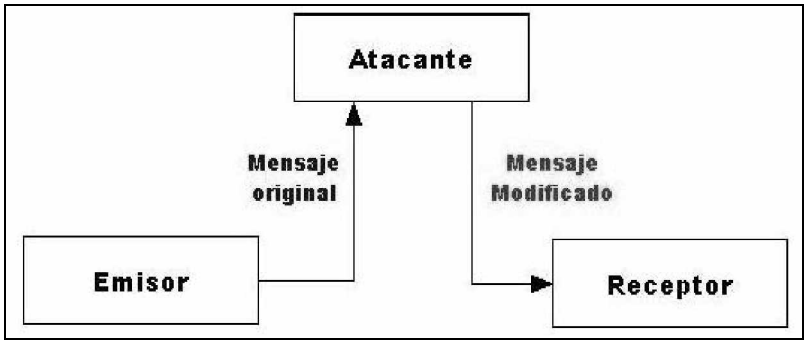


Figura 5.1 Ataque a un mensaje en una red.

Para garantizar la seguridad de la información se incorporan diversos sistemas de cifrado los cuales permiten mantener ocultos los mensajes con relativa efectividad, sin embargo, el hecho de proteger el mensaje no asegura que éste sea sustituido o modificado por otro, lo cual incorpora la necesidad de autenticar el mensaje.

Generalmente un mensaje se valida colocando un sello específico el cual puede ser emitido únicamente por el creador del mensaje. Esta señal es conocida como firma digital.

Una firma digital es un sello electrónico análogo a su versión manuscrita, es decir, que es una marca que se agrega a un documento para que el receptor verifique la autenticidad del documento recibido. En sentido estricto una firma digital será una secuencia única de bits calculados a partir de un algoritmo aplicado a un mensaje específico agregados al final de un mensaje para asegurar que su contenido provenga del emisor esperado [DNT-00].

El utilizar una firma digital implica que es posible adjudicar la creación del mensaje a un emisor específico sin que este pueda negar su emisión y que dicho mensaje no haya sufrido un cambio a través del canal de comunicación por el cual fue enviado.

Para aplicar una firma digital a un documento electrónico es necesario utilizar criptografía asimétrica y funciones resumen o funciones hash. El procedimiento por el cual se firma un documento sigue estos pasos:

- El emisor debe generar un resumen del mensaje a firmar utilizando una función hash y luego la cifra con algún algoritmo asimétrico.
- Se emite el mensaje junto con el grupo de datos que forman la firma digital.

- El receptor adquiere el mensaje y descifra la firma digital utilizando la clave pública previamente obtenida del emisor.
- El receptor genera su propio resumen del mensaje, sabiendo que únicamente el emisor puede crear un resumen como el que recibió, si ambos resúmenes son iguales el mensaje es auténtico [Luc-04].

6. IPSEC (Internet Protocol Security). [Igl-01]

IPSec es un estándar que proporciona servicios de seguridad a la capa IP y a todos los protocolos superiores basados en IP (TCP y UDP, entre otros).

Todas las soluciones anteriores a IPSec se basaban en soluciones propietarias que dificultaban la comunicación entre los distintos entornos empresariales, al ser necesario que éstos dispusiesen de una misma plataforma.

Entre las ventajas de IPSec destacan que está apoyado en estándares del IETF y que proporciona un nivel de seguridad común y homogéneo para todas las aplicaciones, además de ser independiente de la tecnología física empleada. IPSec se integra en la versión actual de IP (IP versión 4) y, lo que es todavía más importante, se incluye por defecto en IPv6.

Puesto que la seguridad es un requisito indispensable para el desarrollo de las redes IP, IPSec está recibiendo un apoyo considerable: todos los equipos de comunicaciones lo incorporan, así como las últimas versiones de los sistemas operativos más comunes. Al mismo tiempo, ya existen muchas experiencias que demuestran la interoperabilidad entre fabricantes, lo cual constituye una garantía para los usuarios.

Otra característica destacable de IPSec es su carácter de estándar abierto. Se complementa perfectamente con la tecnología PKI y, aunque establece ciertos algoritmos comunes, por razones de interoperabilidad, permite integrar algoritmos criptográficos más robustos que pueden ser diseñados en un futuro.

Entre los beneficios que aporta IPSec, cabe señalar que:

- Posibilita nuevas aplicaciones como el acceso seguro y transparente de un nodo IP remoto.
- Facilita el comercio electrónico de negocio a negocio, al proporcionar una infraestructura segura sobre la que realizar transacciones usando cualquier aplicación. Las extranets son un ejemplo.
- Permite construir una red corporativa segura sobre redes públicas, eliminando la gestión y el costo de líneas dedicadas.
- Ofrece al teletrabajador el mismo nivel de confidencialidad que dispondría en la red local de su empresa, no siendo necesaria la limitación de acceso a la información sensible por problemas de privacidad en tránsito.

Es importante señalar que cuando se cita la palabra "seguro" no se refiere únicamente a la confidencialidad de la comunicación, también se está refiriendo a la integridad de los datos, que para muchas compañías y entornos de negocio puede ser un requisito mucho más crítico que la confidencialidad.

Esta integridad es proporcionada por IPSec como servicio añadido al cifrado de datos o como servicio independiente.

6.1 Descripción del Protocolo IPSEC.

IPSec es, en realidad, un conjunto de estándares para integrar en IP funciones de seguridad basadas en criptografía.

Proporciona confidencialidad, integridad y autenticidad de datagramas IP, combinando tecnologías de clave pública (RSA), algoritmos de cifrado (DES, 3DES, IDEA, Blowfish), algoritmos de hash (MD5, SHA-1) y certificados digitales X509v3.

En la Figura 6.1 se observa como IPSec es el resultado de la complementariedad de varias de estas técnicas.

Figura 6.1 Tecnologías utilizadas en IPSEC

El protocolo IPSec ha sido diseñado de forma modular, de modo que se pueda seleccionar el conjunto de algoritmos deseados sin afectar a otras partes de la implementación. Han sido definidos, sin embargo, ciertos algoritmos estándar que deberán soportar todas las implementaciones para asegurar la interoperabilidad en el Internet. Dichos algoritmos de referencia son DES y 3DES, para cifrado, así como MD5 y SHA-1, como funciones de hash. Además es perfectamente posible usar otros algoritmos que se consideren más seguros o más adecuados para un entorno específico: por ejemplo, como algoritmo de

cifrado de clave simétrica IDEA, Blowfish o el más reciente AES que se espera sea el más utilizado en un futuro próximo.

Dentro de IPSec se distinguen los siguientes componentes:

Dos protocolos de seguridad: IP Authentication Header (AH) e IP Encapsulating Security Payload (ESP) que proporcionan mecanismos de seguridad para proteger tráfico IP.

Un protocolo de gestión de claves Internet Key Exchange (IKE) que permite a dos nodos negociar las claves y todos los parámetros necesarios para establecer una conexión AH o ESP.

6.1.1 El Protocolo AH

El protocolo AH es el procedimiento previsto dentro de IPSec para garantizar la integridad y autenticación de los datagrámas IP. Esto proporciona un medio al receptor de los paquetes IP para autenticar el origen de los datos y para verificar que dichos datos no han sido alterados en tránsito. Sin embargo no proporciona ninguna garantía de confidencialidad, es decir, los datos transmitidos pueden ser vistos por terceros.

Tal como indica su nombre, AH es una cabecera de autenticación que se inserta entre la cabecera IP estándar (tanto IPv4 como IPv6) y los datos transportados, que pueden ser un mensaje TCP, UDP o ICMP, o incluso un datagrama IP completo (ver la Figura 6.2).

AH es realmente un protocolo IP nuevo, y como tal el IANA (Internet Asigned Number Authority) le ha asignado el número decimal 51. Esto significa

que el campo Protocolo de la cabecera IP contiene el valor 51, en lugar de los valores 6 ó 17 que se asocian a TCP y UDP respectivamente. Es dentro de la cabecera AH donde se indica la naturaleza de los datos de la capa superior. Es importante destacar que AH asegura la integridad y autenticidad de los datos transportados y de la cabecera IP, excepto los campos variables: TOS, TTL, flags, offset y checksum (ver la Figura 6.2).

El funcionamiento de AH se basa en un algoritmo HMAC (Hashin for Message Authentication), esto es, un código de autenticación de mensajes. Este algoritmo consiste en aplicar una función hash a la combinación de unos datos de entrada y una clave, siendo la salida una pequeña cadena de caracteres que denominamos extracto. Dicho extracto tiene la propiedad de que es como una huella personal asociada a los datos y a la persona que lo ha generado, puesto que es la única que conoce la clave.

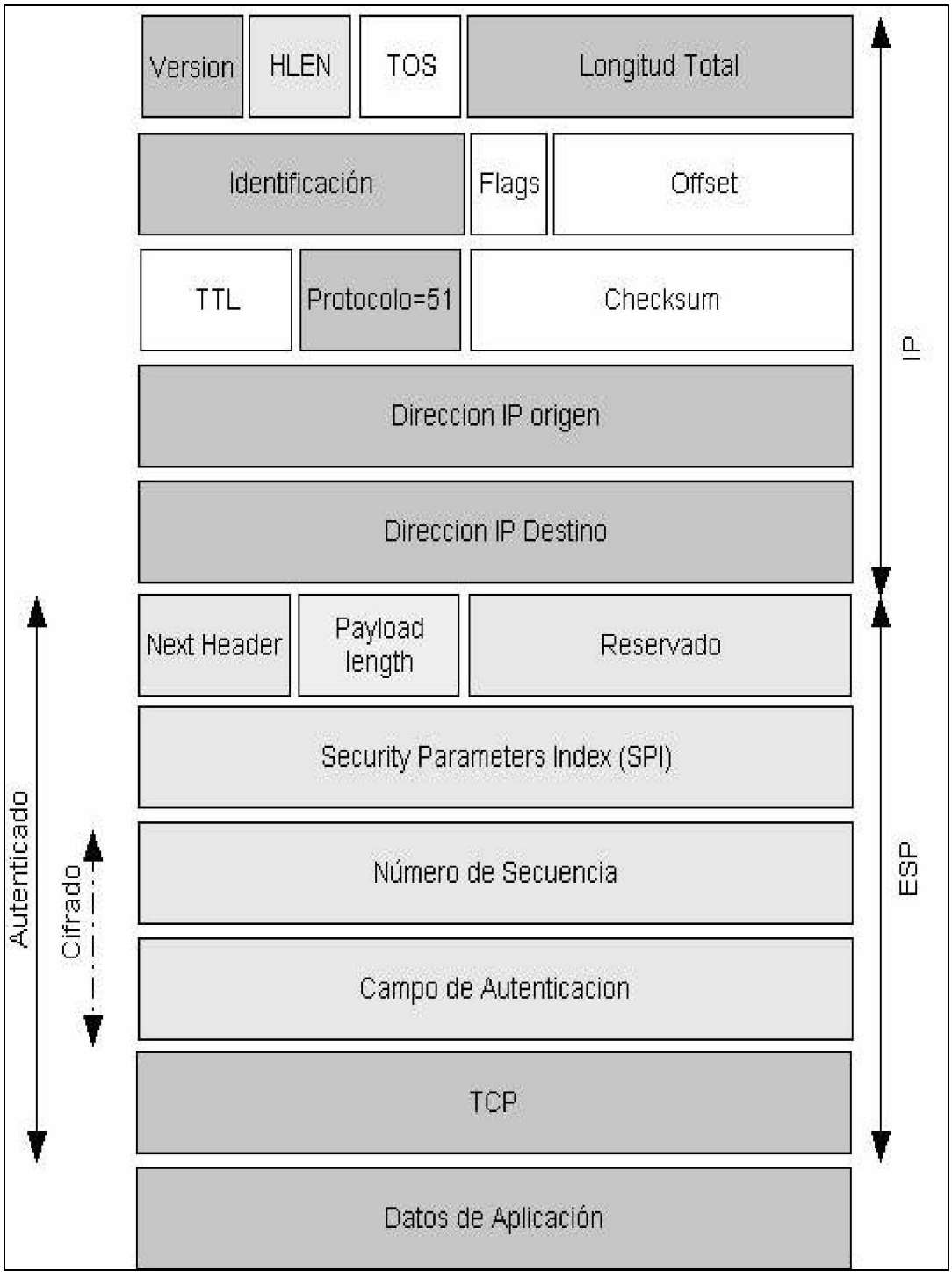


Figura 6.2 Estructura de un datagrama AH

En la Figura 6.3 se muestra el modo en que funciona el protocolo AH. El emisor calcula un extracto del mensaje original, el cual se copia en uno de los campos de la cabecera AH. El paquete así construido se envía a través de la red, repitiéndose en el extremo receptor el cálculo del extracto y comparándolo con el recibido en el paquete. Si son iguales, el receptor tiene la seguridad de que el paquete IP no ha sido modificado en tránsito y que procede efectivamente del origen esperado.

Si analizamos con detalle el protocolo AH, podemos concluir que su seguridad reside en que el cálculo del extracto (MAC) es imposible sin conocer la clave, y que dicha clave (en la Figura 6.3, clave AH) sólo la conocen el emisor y el receptor.

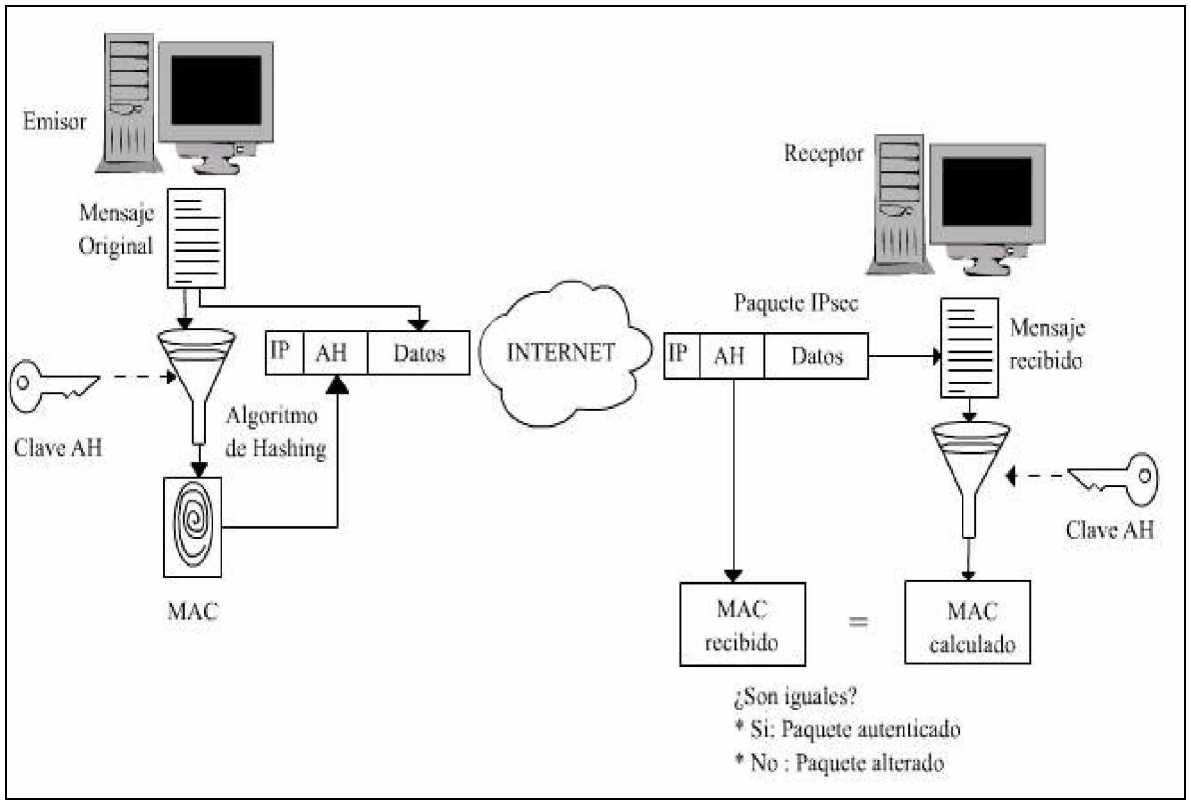


Figura 6.3 Funcionamiento del protocolo AH

6.1.2 El Protocolo ESP.

El objetivo principal del **protocolo ESP** (*Encapsulating Security Payload*) es proporcionar confidencialidad, para ello especifica el modo de cifrar los datos que se desean enviar y cómo este contenido cifrado se incluye en un datagrama IP. Adicionalmente, puede ofrecer los servicios de integridad y autenticación del origen de los datos incorporando un mecanismo similar al de AH.

Dado que ESP proporciona más funciones que AH, el formato de la cabecera es más complejo; este formato consta de una cabecera y una cola que rodean los datos transportados. Dichos datos pueden ser cualquier protocolo IP (por ejemplo, TCP, UDP o ICMP, o incluso un paquete IP completo). En la Figura 6.4 se muestra la estructura de un datagrama ESP, en la que se observa cómo el contenido o carga útil viaja cifrado.

El IANA ha asignado al protocolo ESP el número decimal 50. Esto implica que el campo *Protocolo* de la cabecera IP contendrá el valor 50, mientras que dentro del mensaje ESP se indica la naturaleza de los datos. Puesto que este campo, al igual que la carga útil, está cifrado, un hipotético atacante que intercepte el paquete no podrá saber si el contenido es TCP o UDP; esto es completamente normal ya que el objetivo que se persigue es, precisamente, ocultar la información.

La función de cifrado dentro del protocolo ESP es desempeñada por un algoritmo de cifrado de clave simétrica. Típicamente se usan algoritmos de cifrado bloque, de modo que la longitud de los datos a cifrar tiene que ser un múltiplo del tamaño de bloque (8 o 16 byte, en la mayoría de los casos). Por esta razón existe un campo de relleno, tal como se observa en la Figura 6.4, el cual tiene una función adicional: es posible añadir caracteres de relleno al campo de datos para

ocultar así su longitud real y, por tanto, las características del tráfico. Un atacante suficientemente hábil podría deducir cierta información a partir del análisis de ciertos parámetros de las comunicaciones, aunque estén cifradas, tales como el retardo entre paquetes y su longitud. La función de relleno está pensada para dificultar este tipo de ataques.

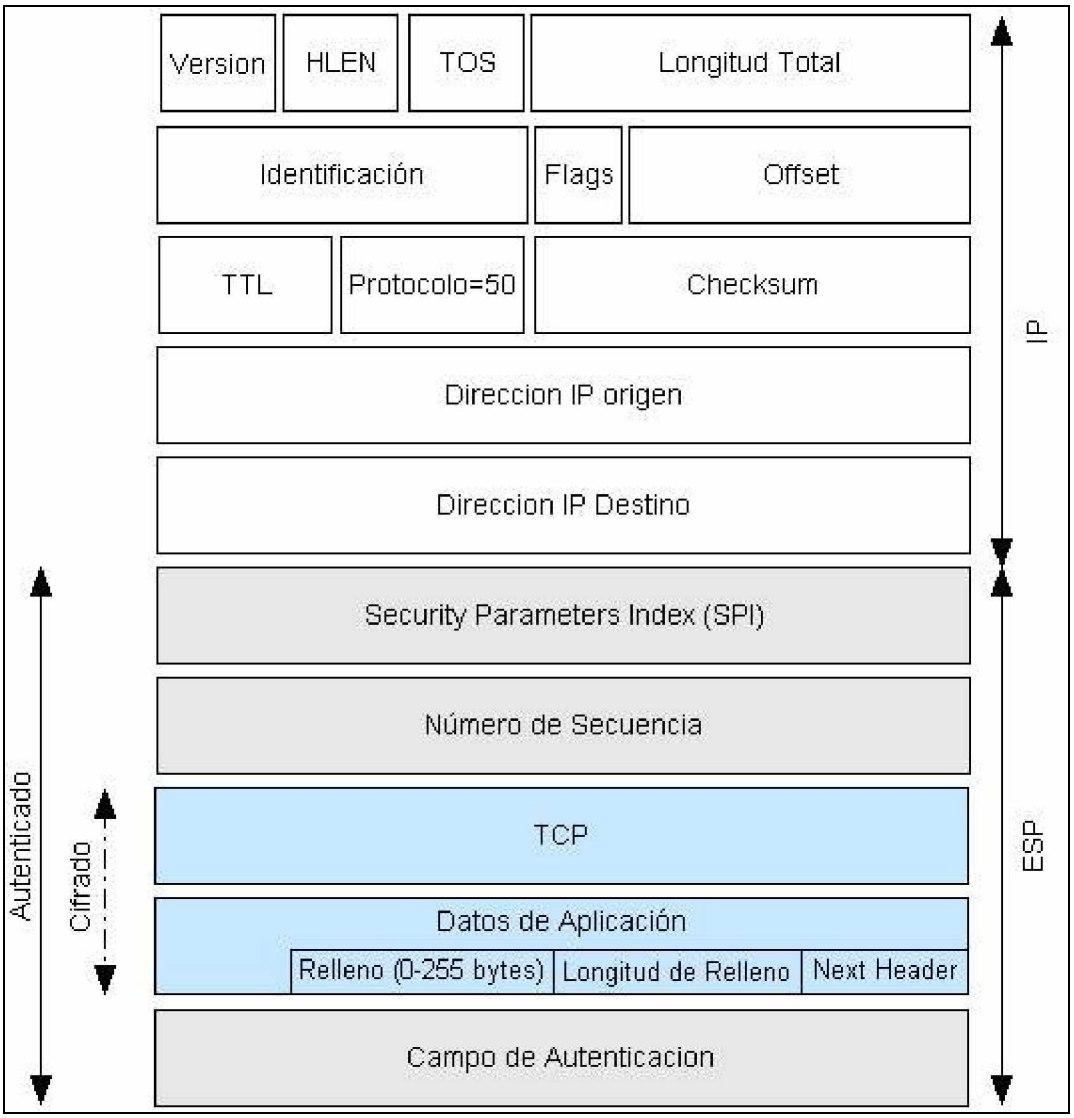


Figura 6.4 Estructura de un datagrama ESP

En la Figura 6.5 se representa cómo el protocolo ESP permite enviar datos de forma confidencial. El emisor toma el mensaje original, lo cifra, utilizando una clave determinada, y lo incluye en un paquete IP, a continuación de la cabecera ESP. Durante el tránsito hasta su destino, si el paquete es interceptado por un tercero sólo obtendrá un conjunto de bits ininteligible. En el destino, el receptor aplica de nuevo el algoritmo de cifrado con la misma clave, recuperando los datos originales.

Está claro que la seguridad de este protocolo reside en la robustez del algoritmo de cifrado, es decir, que un atacante no puede descifrar los datos sin conocer la clave, así como en que la clave ESP únicamente la conocen el emisor y el receptor.

La distribución de claves de forma segura es, por consiguiente, un requisito esencial para el funcionamiento de ESP y también de AH, como hemos visto anteriormente.

Asimismo, es fundamental que el emisor y el receptor estén de acuerdo tanto en el algoritmo de cifrado o de *hash* y como en el resto de parámetros comunes que utilizan. Esta labor de puesta en contacto y negociación es realizada por un protocolo de control, denominado IKE, se verá más adelante.

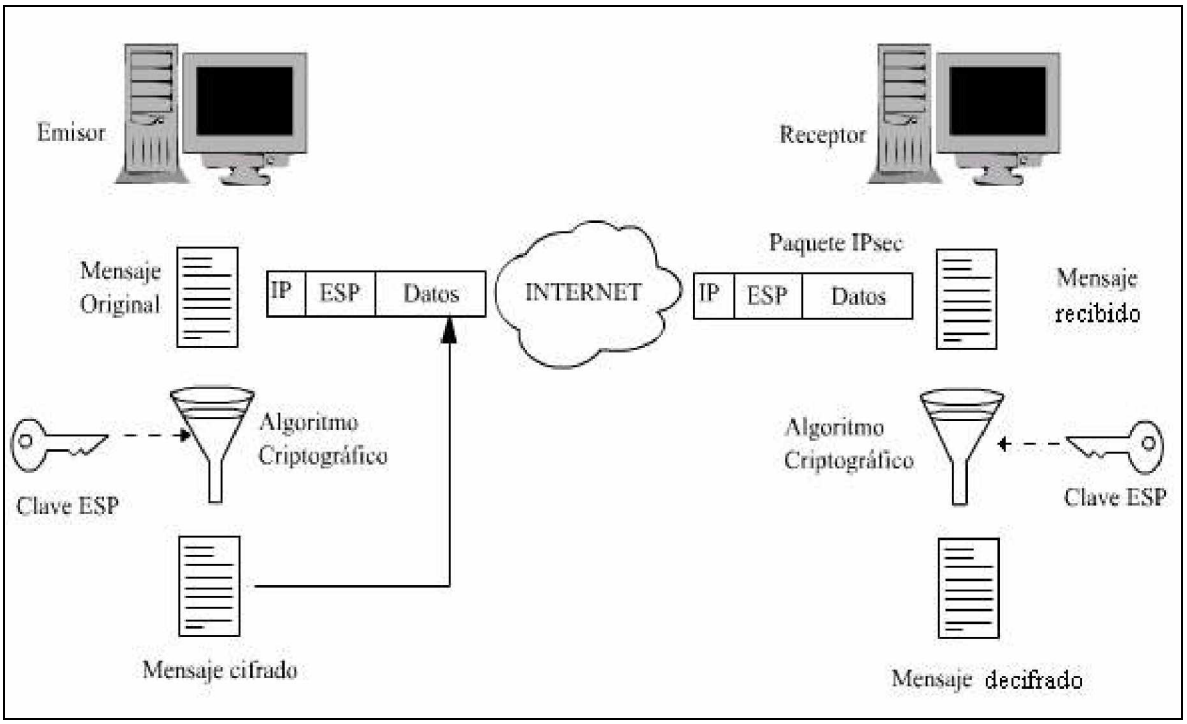


Figura 6.5 Funcionamiento del protocolo ESP

6.2 Los modos de Transporte y Túnel.

Antes de entrar en los detalles del protocolo IKE es necesario explicar los dos modos de funcionamiento que permite IPSec. Tanto ESP como AH proporcionan dos modos de uso:

1. ***El modo transporte***. En este modo el contenido transportado dentro del datagrama AH o ESP son datos de la capa de transporte (por ejemplo, datos TCP o UDP). Por tanto, la cabecera IPSec se inserta inmediatamente a continuación de la cabecera IP y antes de los datos de los niveles superiores que se desean proteger. El modo transporte tiene la ventaja de que asegura la comunicación extremo a extremo, pero requiere que ambos extremos entiendan el protocolo IPSec.

2. ***El modo túnel***. En éste el contenido del datagrama AH o ESP es un datagrama IP completo, incluida la cabecera IP original. Así, se toma un datagrama IP al cual se añade inicialmente una cabecera AH o ESP, posteriormente se añade una nueva cabecera IP que es la que se utiliza para encaminar los paquetes a través de la red. El modo túnel se usa normalmente cuando el destino final de los datos no coincide con el dispositivo que realiza las funciones IPSec.

El modo túnel es empleado principalmente por los *gateways* IPSec, con objeto de identificar la red que protegen bajo una misma dirección IP y centralizar de este modo el procesado del tráfico IPSec en un equipo. El modo túnel también es útil, cuando se utiliza junto con ESP, para ocultar la identidad de los nodos que se están comunicando. Otra aplicación del modo túnel, tanto con ESP como con AH, es poder establecer Redes Privadas Virtuales (RPV) a través de redes públicas, es decir, interconectar de forma segura redes de área local, incluso en el caso de que éstas usen direccionamiento privado o no legal en Internet.

IPSec puede ser implementado bien en un *host* o bien en un equipo dedicado, tal como un *router* o un *firewall,* que cuando realiza estas funciones se denomina *gateway* IPSec. La Figura 6.6 muestra los dos modos de funcionamiento del protocolo IPSec, donde: En la Figura 6.6a se representan dos *hosts* que entienden IPSec y que se comunican de forma segura. Esta comunicación se realiza en modo transporte, por tanto la información que se protege es únicamente el protocolo TCP o UDP, así como los datos de aplicación. En la Figura 6.6b se muestran dos redes que utilizan para conectarse dos

gateways IPSec y, por tanto, emplean una implementación en modo túnel. Se puede ver que la comunicación se realiza a través de una red de datos pública, entre un PC situado en una red local con otro PC situado en una red local remota, de modo que entre los *gateways* IPSec se establece un túnel a través del cual viajan protegidas las comunicaciones entre ambas redes locales.

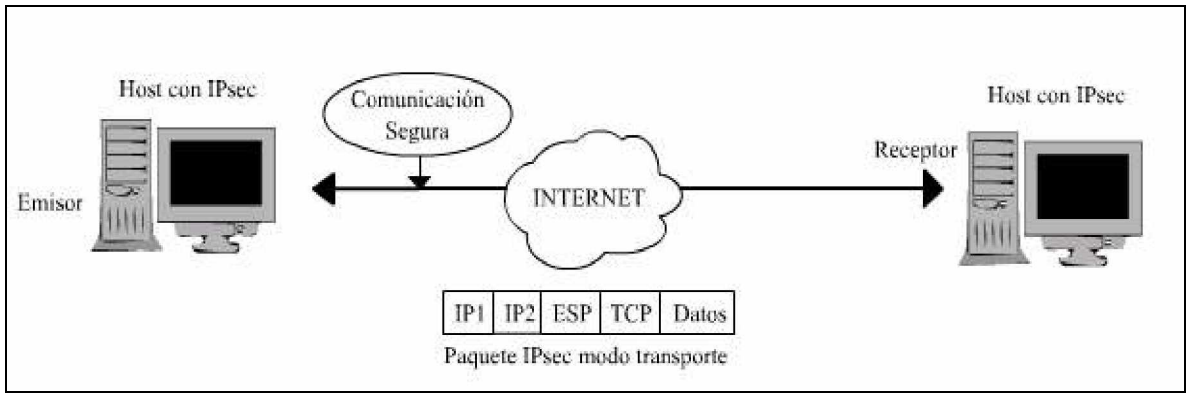


Figura 6.6a Modo de funcionamiento: transporte, IPSec

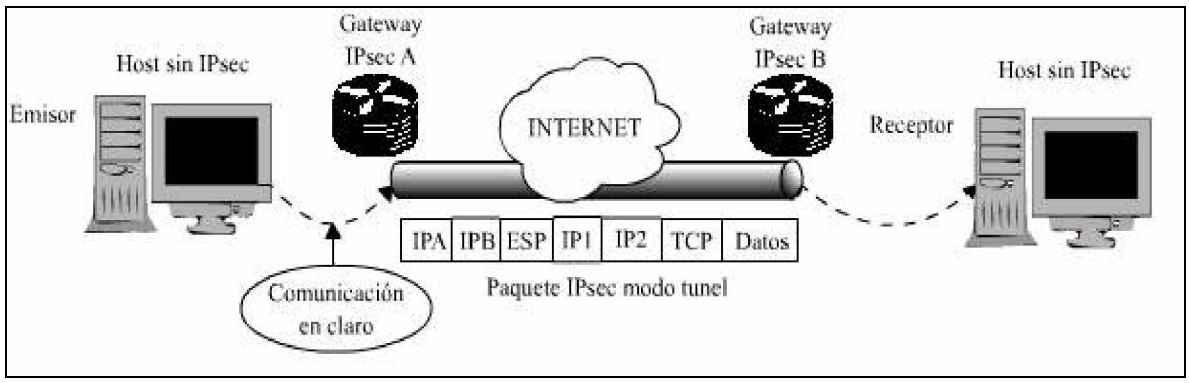


Figura 6.6b Modo de funcionamiento: tunel, IPsec

Sin embargo ambos PCs envían y reciben el tráfico en claro, como si estuviesen situados en la misma red local. Este esquema tiene la ventaja de que los nodos situados en redes separadas pueden comunicarse de forma segura y transparente, concentrándose, al mismo tiempo, las funciones de seguridad en un único punto, facilitando así las labores de administración.

6.3 IKE. El protocolo de control.

Un concepto esencial en IPSec es el de asociación de seguridad (SA): es un canal de comunicación unidireccional que conecta dos nodos, a través del cual fluyen los datagramas protegidos mediante mecanismos criptográficos acordados previamente. Al identificar únicamente un canal unidireccional, una conexión IPSec se compone de dos SAs, una por cada sentido de la comunicación.

Hasta el momento se ha supuesto que ambos extremos de una asociación de seguridad deben tener conocimiento de las claves, así como del resto de la información que necesitan para enviar y recibir datagramas AH o ESP. Tal como se ha indicado anteriormente, es necesario que ambos nodos estén de acuerdo tanto en los algoritmos criptográficos a emplear como en los parámetros de control. Esta operación puede realizarse mediante una configuración manual, o mediante algún protocolo de control que se encargue de la negociación automática de los parámetros necesarios; a esta operación se le llama negociación de SAs.

El IETF ha definido el protocolo IKE para realizar tanto esta función de gestión automática de claves como el establecimiento de las SAs correspondientes.

Una característica importante de IKE es que su utilidad no se limita a IPSec, sino que es un protocolo estándar de gestión de claves que podría ser útil en otros protocolos, como, por ejemplo, OSPF o RIPv2.

IKE es un protocolo híbrido que ha resultado de la integración de dos protocolos complementarios: ISAKMP y Oakley [HAR-98]. ISAKMP define de forma genérica el protocolo de comunicación y la sintaxis de los mensajes que se

utilizan en IKE, mientras que Oakley especifica la lógica de cómo se realiza de forma segura el intercambio de una clave entre dos partes que no se conocen previamente.

El objetivo principal de IKE consiste en establecer una conexión cifrada y autenticada, a través de la cual se negocian los parámetros necesarios para establecer una asociación de seguridad IPSec. Dicha negociación se lleva a cabo en dos fases:

1. La fase común a cualquier aplicación, en la que ambos nodos establecen un canal seguro y autenticado. Dicho canal seguro se consigue mediante el uso de un algoritmo de cifrado simétrico y un algoritmo HMAC. Las claves necesarias se derivan de una clave maestra que se obtiene mediante un algoritmo de intercambio de claves Diffie-Hellman. Este procedimiento no garantiza la identidad de los nodos, para ello es necesario un paso adicional de autenticación.

Existen varios métodos de autenticación, los dos más comunes se describen a continuación: El primer método de autenticación se basa en el conocimiento de un secreto compartido que, como su propio nombre indica, es una cadena de caracteres que únicamente conocen los dos extremos que quieren establecer una comunicación IPSec. Mediante el uso de funciones *hash* cada extremo demuestra al otro que conoce el secreto sin revelar su valor; así los dos se autentican mutuamente. Para no debilitar la seguridad de este mecanismo de autenticación, debe configurarse un secreto distinto para cada par de nodos, por lo que el número de secretos crece muy rápidamente cuando aumenta el número de nodos. Por esta razón en entornos en los que se desea interconectar muchos nodos IPSec la gestión de claves es muy complicada. En este caso no se

recomienda el uso de autenticación mediante secreto compartido, sino autenticación basada en certificados digitales.

En los estándares IPSec está previsto el uso de un método de autenticación que se basa en utilizar certificados digitales X509v3. El uso de certificados permite distribuir de forma segura la clave pública de cada nodo, de modo que éste puede probar su identidad mediante la posesión de la clave privada y ciertas operaciones de criptografía pública. La utilización de certificados requiere de la aparición de un elemento más en la arquitectura IPSec, la PKI (Infraestructura de Clave Pública), cuya integración se tratará con detalle más adelante.

2. En la segunda fase el canal seguro IKE es usado para negociar los parámetros de seguridad específicos asociados a un protocolo determinado, en nuestro caso IPSec.

Durante esta fase se negocian las características de la conexión ESP o AH y todos los parámetros necesarios. El equipo que ha iniciado la comunicación ofrecerá todas las posibles opciones que tenga configuradas en su política de seguridad y con la prioridad que se hayan configurado. El sistema receptor aceptará la primera que coincida con los parámetros de seguridad que tenga definidos. Asimismo, ambos nodos se informan del tráfico que van a intercambiarse a través de dicha conexión.

En la Figura 6.7 se representa de forma esquemática el funcionamiento del protocolo IKE y el modo en que se obtiene una clave de sesión, que es la que se utiliza para proteger las conexiones ESP o AH.

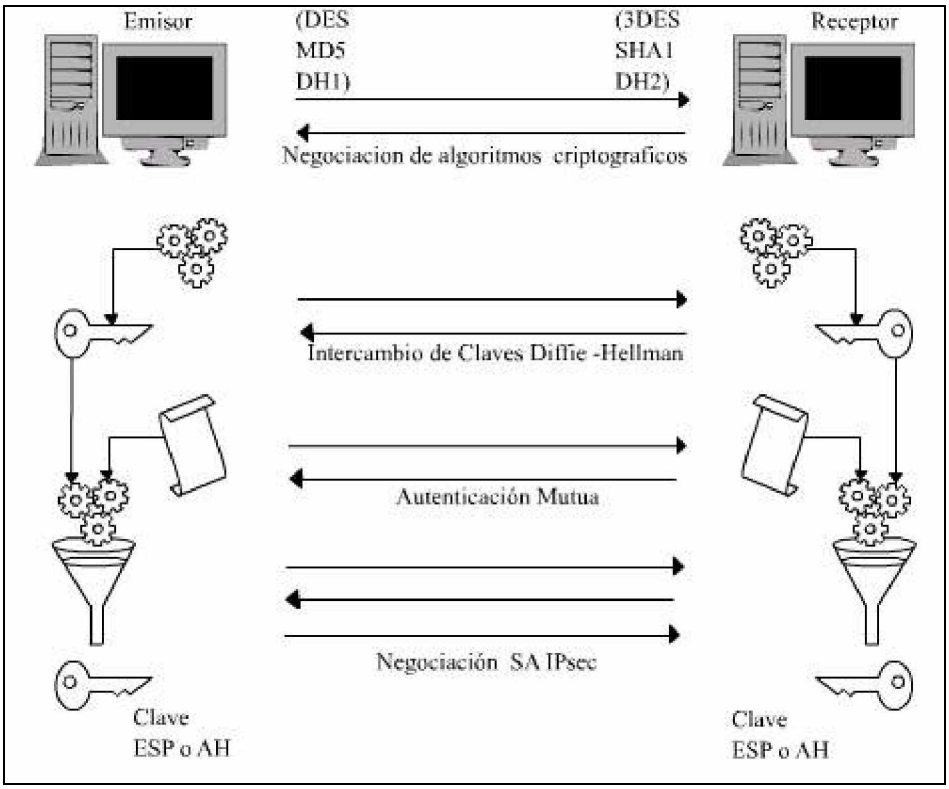


Figura 6.7 funcionamiento del protocolo IKE

6.4 Servicios de Seguridad ofrecidos por IPsec.

En este apartado se analizan las características de los servicios de seguridad que ofrece IPSec. Dichos servicios son:

Integridad y autenticación del origen de los datos El protocolo AH es el más adecuado si no se requiere cifrado. La opción de autenticación del protocolo ESP ofrece una funcionalidad similar, aunque esta protección, a diferencia de AH, no incluye la cabecera IP. Como se comentó anteriormente, esta opción es de gran importancia para aquellas aplicaciones en las cuales es importante garantizar la invariabilidad del contenido de los paquetes IP.

Confidencialidad. El servicio de confidencialidad se obtiene mediante la función de cifrado incluida en el protocolo ESP. En este caso es recomendable activar la opción de autenticación, ya que si no se garantiza la integridad de los datos el cifrado es inútil. Esto es debido a que aunque los datos no pudiesen ser interpretados por nadie en tránsito, éstos podrían ser alterados haciendo llegar al receptor del mensaje tráfico sin sentido que sería aceptado como tráfico válido.

Además de ofrecer el cifrado del tráfico, el protocolo ESP también tiene herramientas para ocultar el tipo de comunicación que se está realizando; para ello permite introducir caracteres de relleno en el contenido de los datos del paquete, de modo que se oculta la verdadera longitud del mismo. Ésta es una protección útil contra las técnicas de análisis de tráfico, que permiten a un atacante deducir información útil a partir del estudio de las características del tráfico cifrado. El análisis de tráfico es un riesgo que debe considerarse seriamente, recientemente se ha documentado la viabilidad para deducir información a partir del tráfico cifrado de una conexión SSH. Es previsible que este tipo de ataques se hagan más habituales y sofisticados en el futuro, conforme se generalice el cifrado de las comunicaciones.

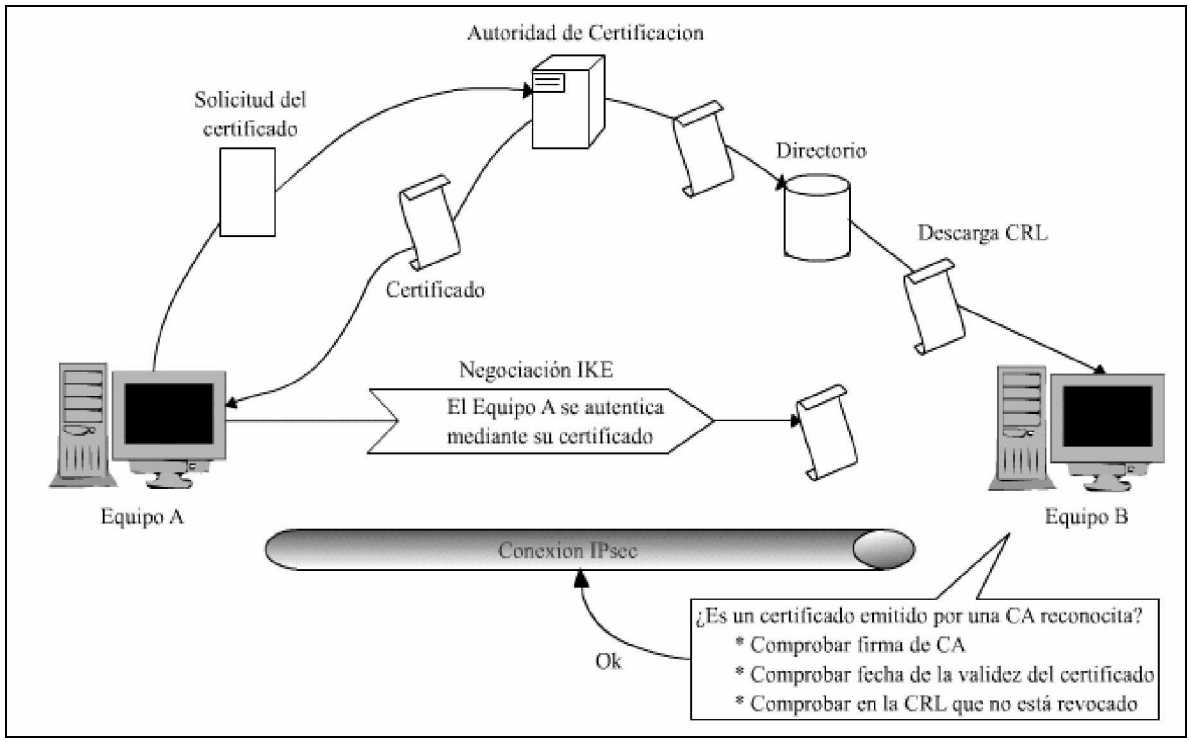


Figura 6.8 integración de una PKI en IPSec

Detección de repeticiones. La autenticación protege contra la suplantación de la identidad IP, sin embargo un atacante todavía podría capturar paquetes válidos y reenviarlos al destino. Para evitar este ataque, tanto ESP como AH incorporan un procedimiento para detectar paquetes repetidos. Dicho procedimiento está basado en un número de secuencia incluido en la cabecera ESP o AH, el emisor incrementa dicho número por cada datagrama que envía y el receptor lo comprueba, de forma que los paquetes repetidos serán ignorados.

Esta secuencia no podrá ser modificada por el atacante, debido a que se encuentra protegida por medio de la opción de integridad para cualquiera de los dos protocolos (AH y ESP) y cualquier modificación en este número provocaría un error en la comprobación de la integridad del paquete.

Control de acceso: autenticación y autorización. Dado que el uso de ESP y AH requiere el conocimiento de claves, y dichas claves son distribuidas de modo seguro mediante una sesión IKE en la que ambos nodos se autentican mutuamente, existe la garantía de que sólo los equipos deseados participan en la comunicación. Es conveniente aclarar que una autenticación válida no implica un acceso total a los recursos, ya que IPSec proporciona también funciones de autorización. Durante la negociación IKE se especifica el flujo de tráfico IP que circulará a través de la conexión IPSec. Esta especificación es similar a un filtro de paquetes, considerándose el protocolo, las direcciones IP de los puertos origen y destino, el byte "TOS" y otros campos.

Por ejemplo, puede utilizarse IPSec para permitir el acceso desde una sucursal a la red local del centro corporativo, pero impidiendo el paso de tráfico hacia máquinas especialmente protegidas.

No repudio. El servicio de no repudio es técnicamente posible en IPSec, si se usa IKE con autenticación mediante certificados digitales. En este caso, el procedimiento de autenticación se basa en la firma digital de un mensaje que contiene, entre otros datos, la identidad del participante. Dicha firma, gracias al vínculo entre la clave pública y la identidad que garantiza el certificado digital, es una prueba inequívoca de que se ha establecido una conexión IPSec con un equipo determinado, de modo que éste no podrá negarlo. En la práctica, sin embargo, esta prueba es más compleja, ya que requeriría almacenar los mensajes de negociación IKE y, además, no está definido un procedimiento para referenciar este evento a una fecha concreta.

LABORATORIO DE CRIPTOGRAFIA.

Manual del Usuario.

INTRODUCCIÓN.

El Manual del Usuario describe los pasos necesarios para el manejo del Programa de Criptografía.

Para cada algoritmo se detallan cada uno de los paneles en los que han sido divididos los algoritmos para su adecuada comprensión. Incluyendo la operación que realiza cada botón y casilla de datos.

La distribución para cada algoritmo presenta las opciones de ejecución y animación necesarias para el desarrollo del cifrado/descifrado según sea el caso.

El Panel Principal permite al usuario elegir con cual de los algoritmos desea iniciar su aprendizaje, incorporando también opciones avanzadas y aplicaciones de cada uno.

Para una mejor comprensión de cada algoritmo de Cifrado, Funciones Hash y Aplicaciones, el usuario deberá tener conocimientos básicos en los algoritmos ó haber leído la teoría proporcionada en la ayuda de este Manual.

PARTE 1. ALGORITMOS DE CLAVE SIMETRICA.

1.a DES (Data Encryption Standard).

Manual de uso y operación.

Introducción.

DES, es un esquema de cifrado simétrico, que se creó con el objeto de proporcionar al público en general un algoritmo de cifrado normalizado para redes de computadoras. Está basado en la aplicación de todas las teorías de cifrado existentes hasta el momento, y fue sometido a las leyes de Estados Unidos.

Se basa en un sistema mono-alfabético, con un algoritmo de cifrado consistente en la aplicación sucesiva de varias permutaciones y sustituciones. Inicialmente el texto en claro a cifrar se somete a una permutación, con bloque de entrada de 64 bits (o múltiplo de 64) y posteriormente a la acción de dos funciones principales, una de permutación y otra de sustitución, en un proceso que consta de 16 ciclos.

En general, DES utiliza una clave simétrica de 64 bits, de los cuales 56 son usados para el cifrado, mientras que los 8 restantes son de paridad, y se usan para la detección de errores.

Como la clave efectiva es de 56 bits, son posible un total de 2 elevado a 56 = 72.057.594.037.927.936 claves posibles, es decir, unos 72.000 billones de claves, por lo que la ruptura del sistema por fuerza bruta o diccionario es improbable, aunque no imposible si se dispone de suerte y una gran potencia de cálculo [Sch-96].

Activando el Algoritmo DES.

Para tener acceso al sistema de cifrado DES posicione el Mouse en la pestaña ***DES*** del menú principal y luego seleccione con un clic la opción de ***Algoritmo.*** Ver Figura 1.a.1.

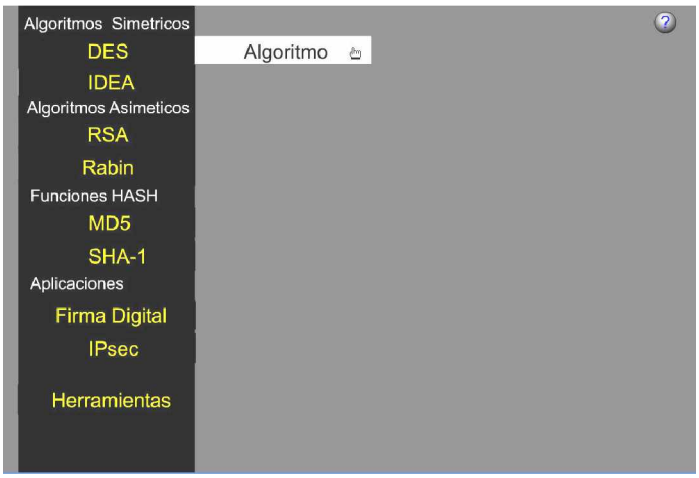


Figura 1.a.1 Menú General para activar el Algoritmo DES.

El panel de trabajo de DES.

El panel de trabajo DES permite al usuario tener acceso a todas las operaciones que se realizan durante los ciclos de cifrado, incluyendo algunas opciones que cambian la forma de operación del algoritmo ver figura 1.a.2.

A continuación se detallan los componentes con los que cuenta el panel de trabajo del algoritmo DES y la forma en la que deben utilizarse.

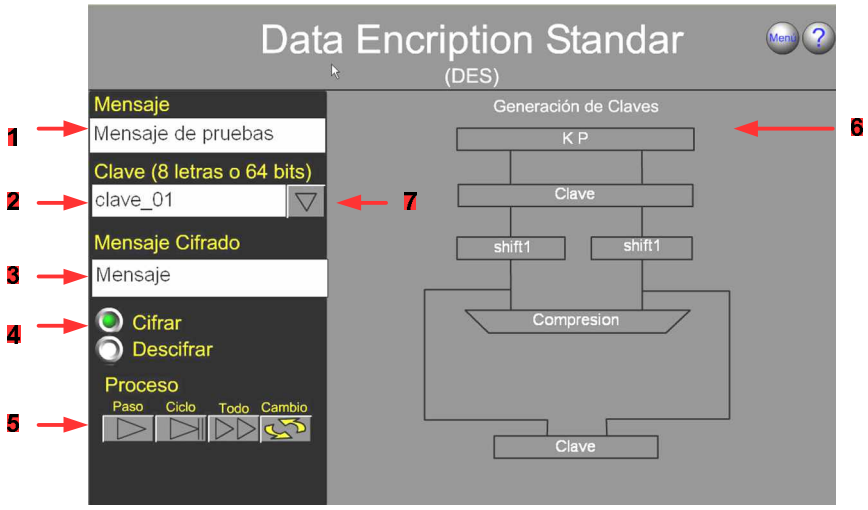


Figura 1.a.2 Panel de trabajo del algoritmo DES.

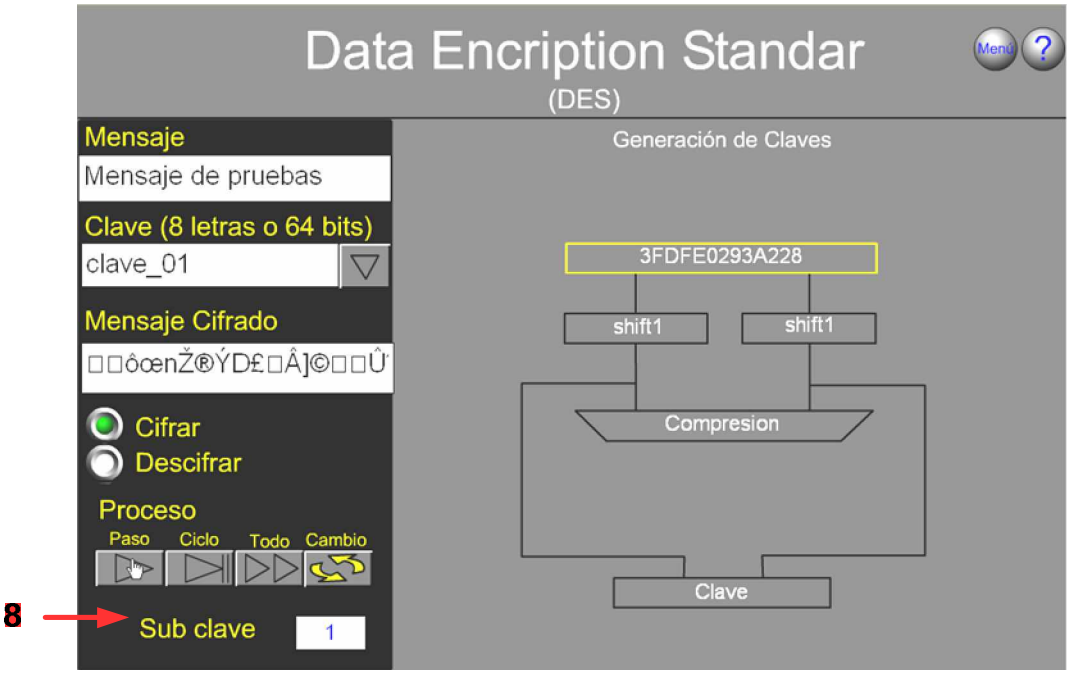


Figura 1.a.3 Flujograma de la generación de subclaves para DES.

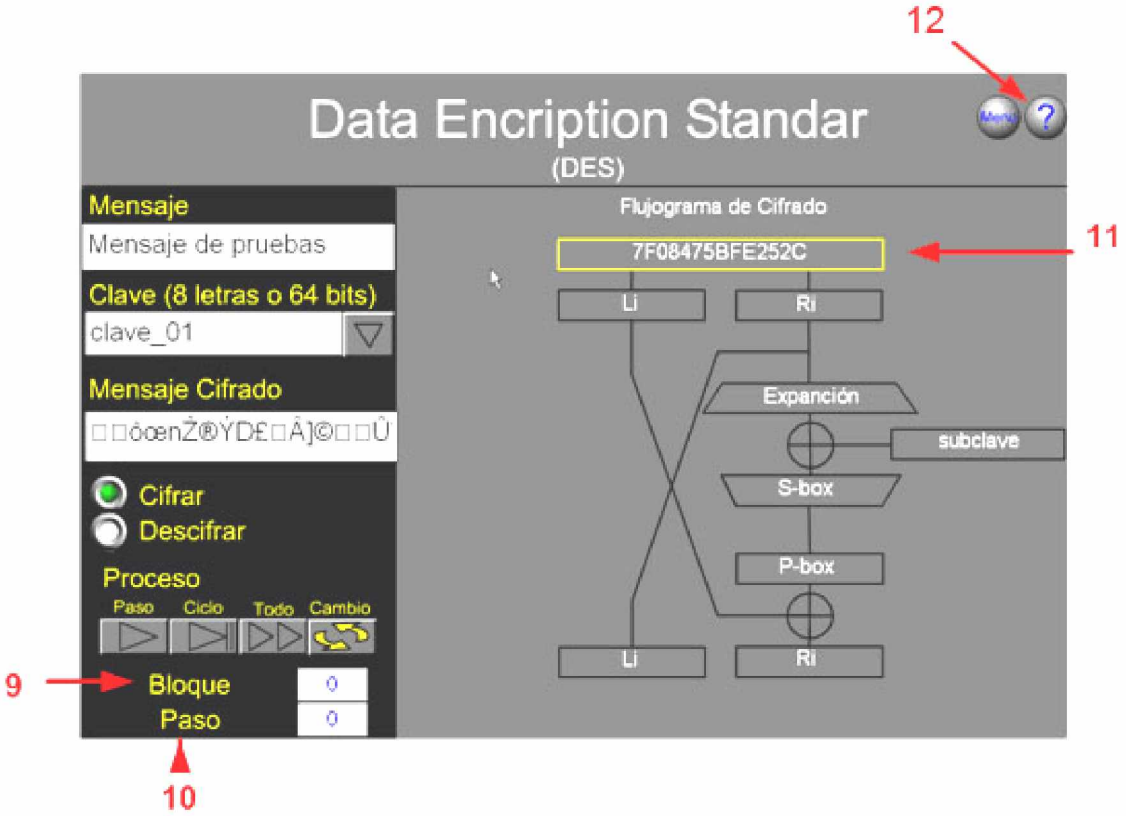
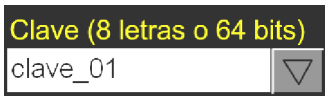


Figura 1.a.4 Flujograma de la secuencia de cifrado/descifrado en DES.

1. **Casilla de mensaje.** (Fig. 1.a.2)

		Permite el acceso de un mensaje a cifrar o descifrar. Acepta cualquier tipo de carácter desde el teclado o utilizando las teclas de método abreviado de Windows (Ctrl-V y Ctrl-C). ï Mientras el algoritmo esté procesando un mensaje, ésta casilla se mantendrá deshabilitada. Se recomienda un mensaje de 15 caracteres para efectos de prueba.

2. **Casilla de clave.** (Fig. 1.a.2)

		La casilla de clave permite al usuario escribir una clave para utilizarla en el cifrado del mensaje que se encuentra en la casilla de mensaje (1), esta casilla acepta cualquier combinación de caracteres del teclado. Deben escribirse 8 caracteres (el equivalente a 64 bits) en esta casilla. ï Mientras el algoritmo esté procesando un mensaje ésta casilla se mantendrá deshabilitada.

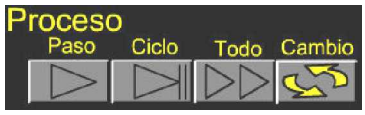
3. **Casilla de mensaje cifrado.** (Fig. 1.a.2)

		Es una casilla de texto únicamente de salida, no puede ser modificada y permite al usuario ver el resultado del cifrado después que el algoritmo ha terminado de procesar el mensaje en la casilla de texto.

4. **Selección cifrado/descifrado.** (Fig. 1.a.2)

		Los botones de opción Cifrado/Descifrado permiten al usuario elegir el tipo de operación que se realizará utilizando los parámetros introducidos en la casilla de clave (2), rotación de bits (11) y casilla de mensaje (1). Para cifrar de un clic en el botón de opción marcado con la palabra cifrar.
		Para descifrar de un clic sobre el botón de opción que esta indicado con la palabra descifrar.

5. **Botones de control.** (Fig. 1.a.2)

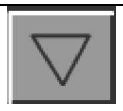
		Estos botones permiten realizar las operaciones de cifrado y descifrado así como mover el texto en la casilla de mensaje cifrado (3) a la casilla de mensaje (1)
		El botón de paso, permite realizar el proceso de cifrado/descifrado, de forma tal, que se observan todas y cada una de las operaciones matemáticas.
		El botón de ciclo, muestra los resultados parciales del cifrado/descifrado para cada ronda

		El botón todo, realiza el proceso de cifrado/descifrado sin mostrar ningún paso intermedio, dando el mensaje resultante según la opción seleccionada.
		El botón cambio, mueve el mensaje resultante en la casilla de cifrado (3) a la casilla de mensaje (1). ¡ Mientras el algoritmo esté procesando un mensaje ésta casilla se mantendrá deshabilitada.

6. **Panel de visualización del proceso de generación de claves.** (Fig. 1.a.2)

KP		Clave Digitada en la casilla de mensaje (2), en nomenclatura hexadecimal.
Clave		Clave resultado de una Permutación que reduce de 64 bits a 56 bits.
Shift 1		División de la clave en dos bloques de 28 bits que son rotados en forma circular.
Compresión		La unión de los dos bloques de 28 bits rotados se comprime mediante una tabla para obtener un bloque de 48 bits que se utilizará para el proceso de cifrado/descifrado.
Clave		Clave generada para el siguiente ciclo.

7. **Botón acceso a claves débiles.** (Fig. 1.a.2)

		Acceso a elegir las claves débiles y semi-débiles para el algoritmo.

8. **Indicador de subclave generada.** (Fig. 1.a.3)

		Indica el número de Subclave generada.

9. **Indicador de bloque.** (Fig. 1.a.4)

		Indica el numero de bloques de 64 bits del Mensaje a Cifrar/Descifrar.

10. **Numero del paso en el proceso de cifrado/descifrado.** (Fig. 1.a.4)

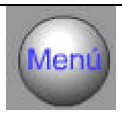
		Indica el número de ciclos para el cifrado de cada Bloque de 64 bits.

11. **Flujograma de cifrado/descifrado.** (Fig. 1.a.4)

Li, Ri		Resultado de Permutar el Mensaje original de 64 bits y dividido en dos bloques de 32 bits. Izquierdo (Li), Derecho (Ri).
Expansión		El bloque Ri se expande de 32 bits a 48 bits.
S-box		Tablas por medio de las cuales se trasforman bloques de entrada de 6 bits a 4 bits.
P-box		Permutación que mezcla los bits de salida de las S-box.

- Al ubicar el cursor sobre cualquier recuadro del proceso que se este ejecutando, aparecerá un mensaje indicando la entrada y la salida de la operación realizada en el bloque, nótese que cuando ocurre la operación el recuadro está en color amarillo.

12. Menú y Ayuda. (Fig. 1.a.4)

	Cancela la ejecución del algoritmo y regresa al menú principal.
	Muestra la ayuda de usuario para el algoritmo que esté en ejecución.

Bibliografía.

[Sch-96]

Schneier, Bruce. “Applied Cryptography”, Segunda Edición, Mineapolis, 1996

[STA-98]

William STALLINGS. “Cryptography and Network Security: Principles and Practice”. Segundo Edición. Prentice Hall, 1998.

[Val-04]

Valeiras Reina, Gerardo. "Red Temática Iberoamericana de Criptografía y Seguridad en la Información", Universidad de Sevilla, España, 2004

1.b IDEA (International Data Encryption Algorithm).

Manual de uso y operación.

Introducción.

IDEA es un algoritmo bastante seguro, y hasta ahora se ha mostrado resistente a multitud de ataques. No presenta claves débiles, y su longitud de clave hace imposible en la práctica un ataque por la fuerza bruta. Trabaja con bloques de 64 bits de longitud y emplea una clave de 128 bits. Utiliza el mismo algoritmo tanto para cifrar como para descifrar.

IDEA se basa en los conceptos de confusión y difusión, haciendo uso de las siguientes operaciones elementales:

- XOR.
- Suma módulo 2^{16}.
- Producto módulo $2^{16} + 1$.

El calculo de las subclaves se realiza dividiendo la clave de entrada en bloques de 16 bits, tomando estos bloques como las primeras ocho subclaves. Las siguientes ocho se calculan rotando la clave de entrada 25 bits a la izquierda y volviendo a dividirla, y así sucesivamente. Las subclaves necesarias para descifrar se obtienen cambiando de orden las Z_i y calculando sus inversas para la suma o la multiplicación. [Sch-96]

Activando el Algoritmo IDEA.

Para tener acceso al sistema de cifrado IDEA posicione el Mouse en la pestaña ***IDEA*** del menú principal y luego seleccione con un clic la opción de ***Algoritmo.*** Ver figura 1.b.1.

Figura 1.b.1 Menú general para activar el algoritmo IDEA.

El panel de trabajo de IDEA.

El panel de trabajo de IDEA permite al usuario tener acceso a todas las operaciones que se realizan durante los ciclos de cifrado, incluyendo algunas opciones que cambian la forma de operación del algoritmo.

A continuación se detallan los componentes con los que cuenta el panel de trabajo del algoritmo IDEA y la forma en la que deben utilizarse.

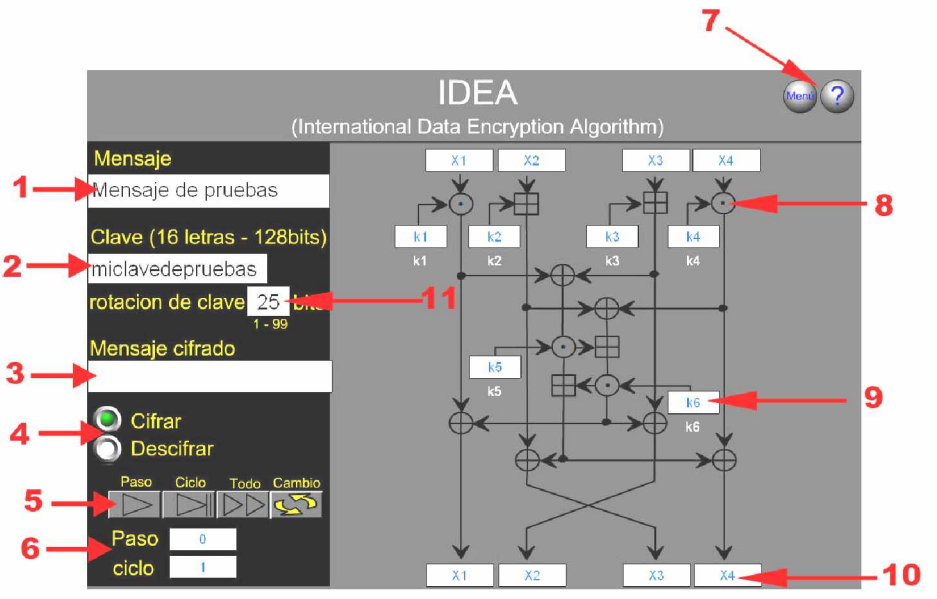


Figura 1.b.2 Panel de trabajo del algoritmo IDEA.

1. **Casilla de mensaje.** (Fig. 1.b.2)

		Permite el acceso de un mensaje a cifrar o descifrar. Acepta cualquier tipo de caracter desde el teclado o utilizando las teclas de método abreviado de Windows (Ctrl-V y Ctrl-C). ï Mientras el algoritmo esté procesando un mensaje, esta casilla se mantendrá deshabilitada

2. **Casilla de Clave.** (Fig. 1.b.2)

		La casilla de clave permite al usuario escribir una clave para utilizarla en el cifrado/descifrado del mensaje que se encuentra en la casilla de mensaje (1), ésta acepta cualquier combinación de caracteres del teclado. Deben escribirse 16 caracteres (el equivalente a 128 bits). ï Mientras el algoritmo esté procesando un mensaje ésta casilla se mantendrá deshabilitada.

3. **Casilla de mensaje cifrado.** (Fig. 1.b.2)

		Es una casilla de texto únicamente de salida, no puede ser modificada y permite al usuario ver el resultado del cifrado/descifrado después que el algoritmo ha terminado de procesar el mensaje en la casilla de texto.

4. **Selección Cifrado/Descifrado.** (Fig. 1.b.2)

		Los botones de opción Cifrado/Descifrado permiten al usuario elegir el tipo de operación que se realizara utilizando los parámetros introducidos en la casilla de clave (2), rotación de bits (11) y casilla de mensaje (1). Para cifrar de un clic en el botón de opción marcado con la palabra cifrar.
		Para descifrar de un clic sobre el botón de opción que esta indicado con la palabra descifrar.

5. **Botones de control.** (Fig. 1.b.2)

		Estos botones permiten realizar las operaciones de cifrado y descifrado así como mover el texto en la casilla de mensaje cifrado (3) a la casilla de mensaje (1)
		El botón de paso, permite realizar el proceso de cifrado/descifrado, de forma tal que se observan todas y cada una de las operaciones matemáticas.
		El botón de ciclo, muestra los resultados parciales del cifrado/descifrado cada 14 pasos.

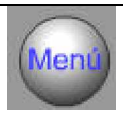
		El botón todo, realiza el proceso de cifrado/descifrado sin mostrar ningún paso intermedio, dando el mensaje resultante de la opción seleccionada.
		El botón cambio, mueve el mensaje resultante en la casilla de cifrado (3) a la casilla de mensaje (1). ï Mientras el algoritmo este procesando un mensaje ésta casilla se mantendrá deshabilitada

.

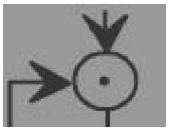
6. **Indicadores de Paso/Ciclo.** (Fig. 1.b.2)

		Muestra el estado del proceso de cifrado/descifrado cuando se realiza paso a paso o por ciclo.

7. **Menú y Ayuda.** (Fig. 1.b.2)

		Cancela la ejecución del algoritmo y regresa al menú principal.
		Muestra la ayuda de usuario para el algoritmo que esté en ejecución.

8. **Operaciones.** (Fig. 1.b.2)

		Permite observar las operaciones matemáticas realizadas en el proceso de cifrado/descifrado cuando se desarrolla el algoritmo paso a paso.

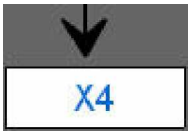
- Al ubicar el cursor sobre cualquier símbolo de la operación matemática que se esté ejecutando, aparecerá un recuadro indicando los argumentos y el resultado de dicha operación, nótese que cuando ocurre la operación está en color amarillo.

		Representa una multiplicación modulo $2^{16}+1$
		Representa una suma modulo 2^{16}
		Representa la operación XOR

9. Casillas de Subclaves. (Fig. 1.b.2)

		En estas casillas se observarán las subclaves que se calculan después de cada operación matemática en el proceso de cifrado/descifrado.

10. Casillas de mensajes intermedios. (Fig. 1.b.2)

		En estas casillas se observará el mensaje dividido en 4 partes que se calcula después de cada operación matemática en el proceso de cifrado/descifrado.

11. Casilla de rotación de bits. (Fig. 1.b.2)

		Permite modificar la generación de las subclaves. El valor por defecto es 25 pero puede cambiarse por cualquier número entre 1 y 99. İ Mientras el algoritmo esté procesando un mensaje, esta casilla se mantendrá deshabilitada.

Generando Claves para el Algoritmo IDEA.

Coloque el puntero del Mouse sobre la pestaña ***IDEA*** del menú principal y luego, del menú emergente, de clic sobre la opción ***Claves.*** Ver figura 1.b.3

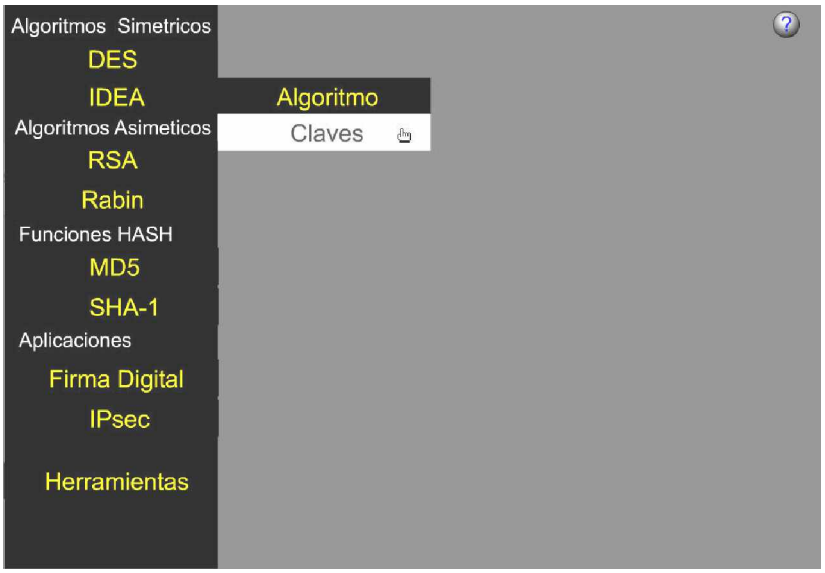


Figura 1.b.3 Menú general para activar visualizador de sub-claves para IDEA.

Panel de trabajo del generador de Claves de IDEA.

El generador de Claves de IDEA muestra al usuario una tabla de subclaves, generadas a partir de una clave de 16 símbolos que pueden ser utilizadas para el cifrado/descifrado utilizando el algoritmo IDEA.

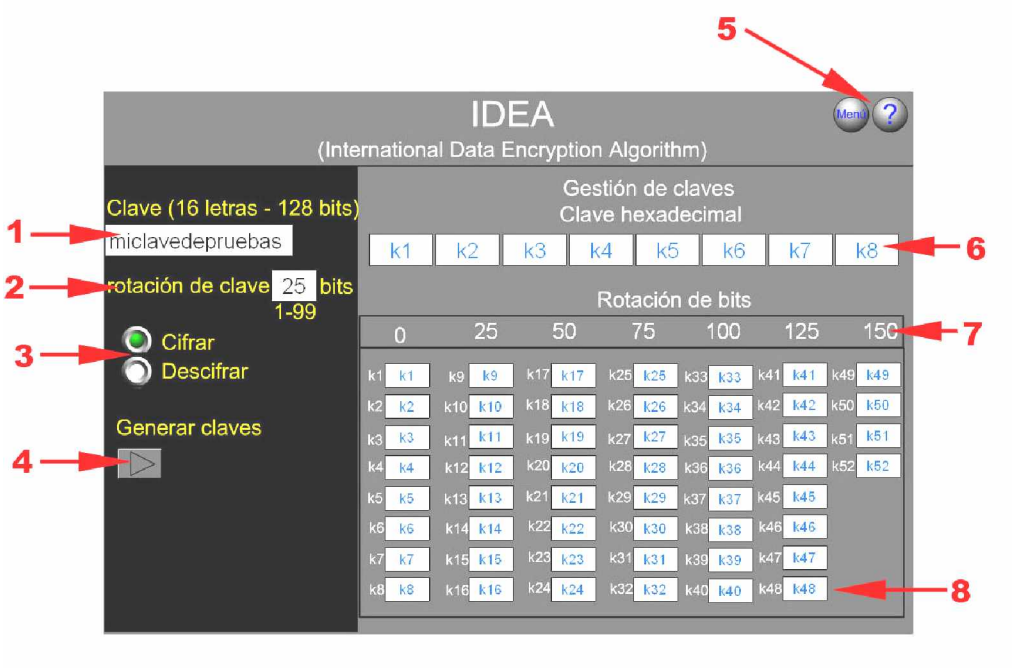


Figura 1.b.4 Funciones del panel de generación de claves de IDEA.

1. **Casilla de clave.** (Fig. 1.b.4)

Clave (16 letras - 128bits) miclavedepruebas		La casilla de clave permite al usuario escribir una clave para utilizarla como base en la generación de las subclaves para el cifrado/descifrado de un mensaje. Esta casilla acepta cualquier combinación de caracteres del teclado. Deben escribirse 16 caracteres (el equivalente a 128 bits). ¡ Mientras el algoritmo esté procesando un mensaje esta casilla se mantendrá deshabilitada

2. **casilla de rotación de bits.** (Fig. 1.b.4)

		Permite modificar la generación de las claves. El valor por defecto es 25 pero puede cambiarse por cualquier número entre 1 y 99. ì Mientras el algoritmo esté procesando un mensaje, esta casilla se mantendrá deshabilitada.

3. **Selección Cifrado/Descifrado.** (Fig. 1.b.4)

		Los botones de opción Cifrado/Descifrado permiten al usuario elegir el tipo de subclaves que serán generadas, ésta operación se realiza utilizando los parámetros introducidos en la casilla de clave (1), rotación de bits (2). Para generar las subclaves de cifrado de un clic en el botón de opción marcado con la palabra cifrar.
		Para generar las subclaves de descifrado de un clic sobre el botón de opción que está indicado con la palabra descifrar.

4. **Generador de claves.** (Fig. 1.b.4)

		Crea todas las subclaves de cifrado/descifrado utilizando los valores obtenidos de las casillas de clave (1) y de rotación de bits (2). Las subclaves generados dependerán de la opción elegida con los botones de selección cifrado/descifrado (3)

5. **Menú y Ayuda.** (Fig. 1.b.4)

		Cancela la ejecución del algoritmo y regresa al menú principal.
		Muestra la ayuda de usuario para el algoritmo que esté en ejecución.

6. **Clave hexadecimal.** (Fig. 1.b.4)

		Muestra la clave introducida por el usuario en formato hexadecimal y dividida en 8 bloques de 16 bits cada uno. Estos 8 bloques forman las primeras 8 subclaves.

7. **contador de rotación de bits.** (Fig. 1.b.4)

		Este numero muestra al usuario la cantidad de bits que se han rotado circularmente, que han sido necesarios para generar las subclaves que se encuentran debajo de él. Este valor cambia según se modifique la casilla de rotación de bits (2) y toma efecto en la tabla al dar clic en el botón de generación de claves (4).

8. **Subclaves.** (Fig. 1.b.4)

		Muestra la K-esima subclave generada para la clave que ha sido introducida en la casilla de clave (1).

Bibliografía.

[Sch-96]

Schneier, Bruce. “Applied Cryptography”, Segunda Edición, Mineapolis, 1996

[Luc-04]

López, Manuel José. "Criptografía y Seguridad en Computadores". Tercera Edición (Versión 2.15). Marzo de 2004.

PARTE 2. ALGORITMOS DE CLAVE ASIMETRICA.

2.a RSA.

Manual de uso y operación.

Introducción.

RSA es un algoritmo asimétrico que basa su seguridad en la dificultad de factorizar grandes números, consta de dos tipos de claves, una publica y una privada las cuales se utilizan según sea la aplicación del algoritmo.

Las claves pública y privada se obtienen al operar números primos grandes y el cifrado/descifrado de la información depende de estos valores. [Luc-04].

Activando el panel de trabajo de RSA paso a paso.

En el menú principal, coloque el puntero del Mouse sobre la pestaña **RSA**, luego seleccione del menú emergente **Paso a Paso.** Ver figura 2.a.1

Figura 2.a.1 Menú general para activar algoritmo RSA paso a paso.

Panel de trabajo de RSA paso a paso.

El panel de trabajo RSA paso a paso permite al usuario realizar una secuencia ordenada de pasos para el correcto uso del algoritmo. Esta dividida en cuatro pasos fundamentales que se irán activando según se desarrolle el cifrado/descifrado.

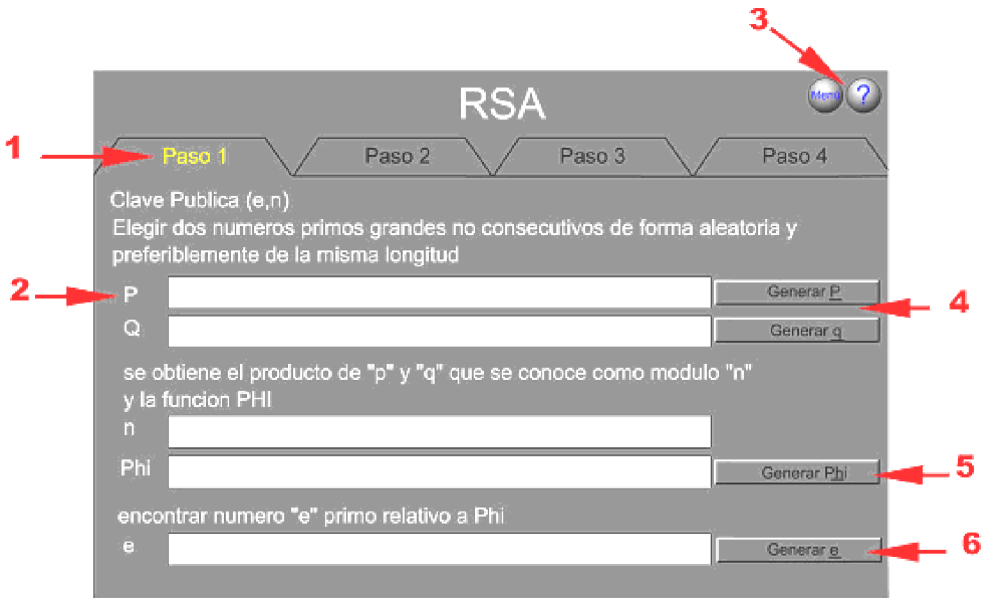


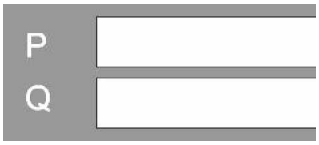
Figura 2.a.2 Panel de trabajo de RSA paso a paso.

Funciones del panel de trabajo RSA paso a paso.

1. **Pestaña de pasos.** (Fig. 2.a.2)

		Permite desplazarse a través de los cuatro pasos necesarios para cifrar y descifrar un mensaje. Cuando la etiqueta de la pestaña se encuentre en amarillo, significa que es el paso que se esta ejecutando y no podrán activarse los pasos siguientes a menos que el actual sea terminado. Para cambiar de un paso a otro, de un clic sobre la pestaña etiquetada con el paso deseado o presione la tecla TAB.

2. **Casillas de datos/resultados.** (Fig. 2.a.2 y Fig. 2.a.3)

		Estas casillas cumplen con dos funciones, muestran los resultados de las operaciones matemáticas que se efectúan en el proceso de cifrado/descifrado. También se utilizan para introducir valores que modifican el resultado del cifrado/descifrado

3. **Menú y ayuda.** (Fig. 2.a.2 y Fig. 2.a.3)

		Cancela la ejecución del algoritmo y regresa al menú principal.
		Muestra la ayuda de usuario para el algoritmo que esté en ejecución.

4. **botones para generación de datos/Procesos.** (Fig. 2.a.2 y Fig. 2.a.3)

		Son botones que ejecutan los procesos matemáticos del RSA. Su forma de operar dependerá del tipo de botón
Generar p		Genera un número primo con una longitud de 5 dígitos. i Puede utilizar la tecla P como método abreviado para generar el número o puede escribirlo directamente en la casilla de datos a un lado del botón.
Generar q		Genera un número primo con una longitud de 5 dígitos. i Puede utilizar la tecla Q como método abreviado para generar el número o puede escribirlo directamente en la casilla de datos a un lado del botón.

Generar Phi		Realiza las operaciones matemáticas para obtener los números “Phi” y “n”. i Puede utilizar la tecla “h” como método abreviado para obtener estos resultados
Generar e		Permite obtener el valor de “e” que representa parte de la clave pública. i Puede utilizar la tecla “e” como método abreviado para obtener este valor.
Generar d		Permite obtener el valor de “d” que representa parte de la clave pública. i Puede utilizar la tecla “d” como método abreviado para obtener este valor.
Cifrar		Activa el algoritmo de cifrado para el mensaje decimal propuesto por el usuario.
Descifrar		Activa el algoritmo de descifrado para el mensaje decimal generado por el programa o el propuesto por el usuario. i El usuario puede cambiar el mensaje cifrado en la caja de cifrado durante el paso 4
Convertir		Permite al usuario escribir un mensaje en caracteres ASCII (caracteres del teclado) y convertirlo en decimal (formato requerido por RSA)

Convirtiendo un mensaje ASCII a Decimal.

Las casillas de mensaje de RSA están divididas en 3 partes para facilitar la lectura y la escritura de mensajes.

La primera casilla denominada mensaje acepta caracteres ASCII y permite convertir su contenido en valores hexadecimales, los cuales se presentan en la casilla de texto “HEX” y en valores decimales en la casilla de texto “DEC” los cuales se actualizan después de utilizar el botón convertir.

Si el usuario no desea escribir un mensaje en ASCII puede escribir directamente sobre la casilla de texto “DEC”, el programa interpretara el numero escrito como un mensaje valido, pero solo deben escribirse valores decimales.

Activando el panel de trabajo de RSA.

En el menú principal, coloque el puntero del ratón sobre la pestaña **RSA**, luego seleccione del menú emergente **Algoritmo.**

Figura 2.a.3 Menú general para activar algoritmo RSA.

Panel de trabajo RSA.

El panel de trabajo RSA opera de igual forma que el panel de trabajo paso a paso, pero reúne todas las variables que componen el proceso de cifrado/descifrado en una sola pantalla, dando mayor libertar de pruebas y operaciones al usuario, sus funciones y su uso son similares al modo paso a paso.

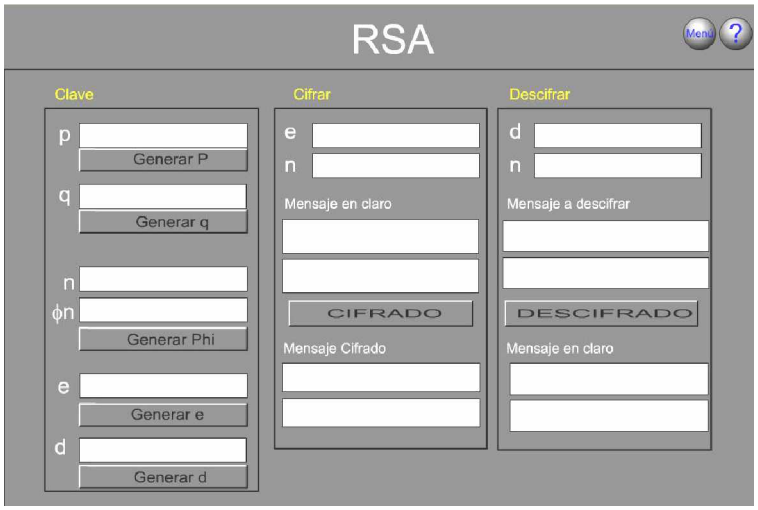


Figura 2.a.3 Panel de trabajo en RSA para usuarios avanzados.

Bibliografía.

[Ang-04]

Ángel, Jesús Ángel. "Red Temática Iberoamericana de Criptografía y Seguridad en la Información", Seguridata, México, 2004.

[Luc-04]

López, Manuel José. "Criptografía y Seguridad en Computadores". Tercera Edición (Versión 2.15). Marzo de 2004.

[Fra-98]

Franco, Jose Pastor. “Criptografia digital fundamentos y aplicaciones”. Prensas Universitarias de Zaragoza. 1998

2.b Rabin.

Manual de uso y operación.

Introducción.

Rabin es un algoritmo asimétrico, basa su seguridad en la dificultad para determinar raíces cuadradas de grandes números, al igual que RSA consta de dos tipos de claves, una publica y una privada las cuales se utilizan según sea la aplicación del algoritmo.

Las claves pública y privada se obtienen al operar números primos grandes y el cifrado/descifrado de la información depende de estos valores, sin embargo Rabin es poco usado por la dificultad que presenta al descifrado, ya que no es posible determinar con facilidad el mensaje original. [Luc-04]

Activando el panel de trabajo de Rabin paso a paso.

En el menú principal, coloque el puntero del Mouse sobre la pestaña **Rabin**, luego seleccione del menú emergente **Paso a Paso.** Ver figura 2.b.1.

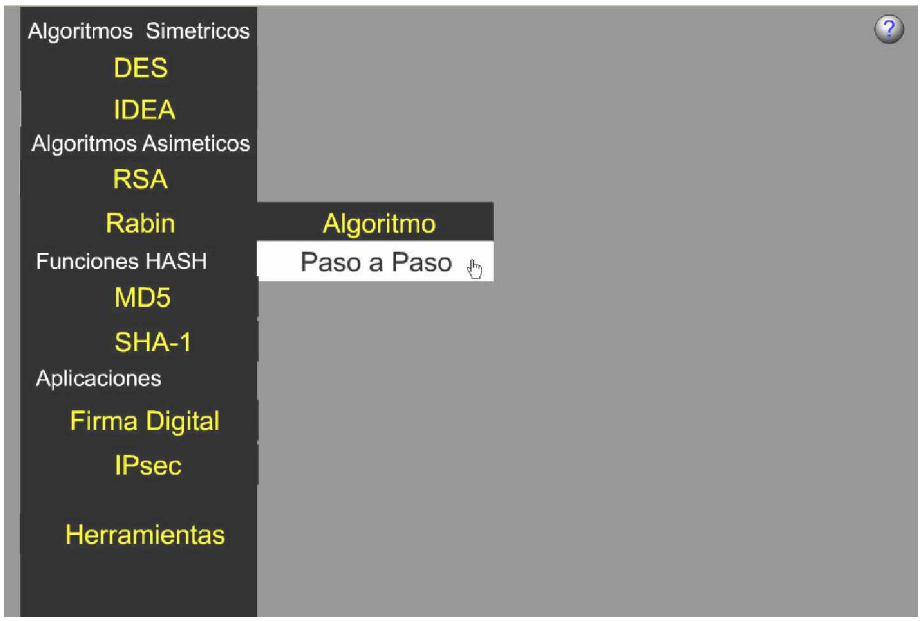


Figura 2.b.1 Menú general para activar algoritmo Rabin paso a paso.

Panel de trabajo de Rabin paso a paso.

El panel de trabajo de Rabin paso a paso permite al usuario realizar una secuencia ordenada de pasos para el correcto uso del algoritmo. Está dividida en tres pasos fundamentales que se irán activando según se desarrolle el cifrado/descifrado.

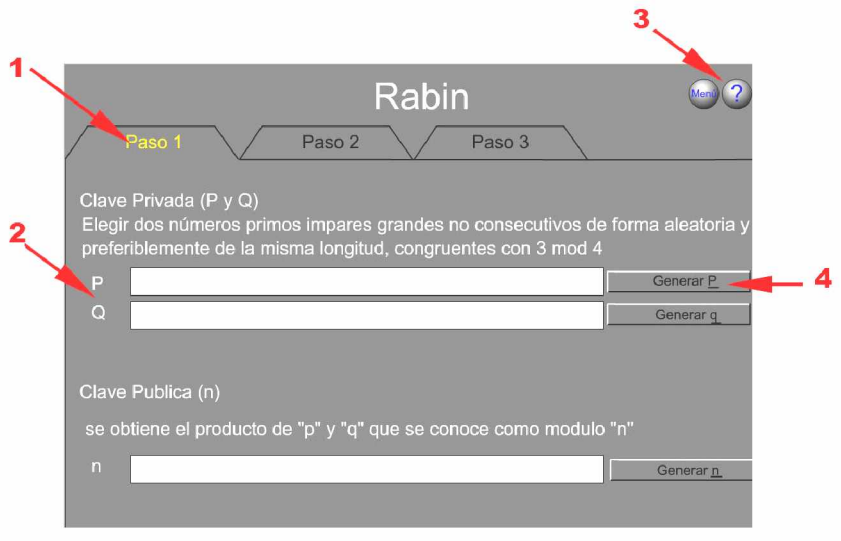


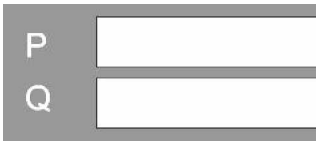
Figura 2.b.2 Panel de trabajo en Rabin paso a paso.

Funciones del panel de trabajo Rabin paso a paso.

1. **Pestaña de pasos.** (Fig. 2.b.2)

		Permite desplazarse a través de los tres pasos necesarios para cifrar y descifrar un mensaje. Cuando la etiqueta de la pestaña se encuentre en amarillo, significa que es el paso que se está ejecutando y no podrán activarse los pasos siguientes a menos que el actual sea terminado. Para cambiar de un paso a otro de un clic sobre la pestaña etiquetada con el paso deseado o presione la tecla TAB.

2. **Casillas de datos/resultados.** (Fig. 2.b.2 y Figura 2.b.4)

		Estas casillas cumplen con dos funciones, muestran los resultados de las operaciones matemáticas que se efectúan en el proceso de cifrado/descifrado. También se utilizan para introducir valores que modifican el resultado del cifrado/descifrado

3. **Menú y ayuda.** (Fig. 2.b.2 y Figura 2.b.4)

		Cancela la ejecución del algoritmo y regresa al menú principal.
		Muestra la ayuda de usuario para el algoritmo que esté en ejecución.

4. **Botones para generación de datos/Procesos.** (Fig. 2.b.2 y Figura 2.b.4)

		Son botones que ejecutan los procesos matemáticos de Rabin. Su forma de operar dependerá del tipo de botón
Generar p		Genera un número primo con una longitud de 5 dígitos. i Puede utilizar la tecla “P” como método abreviado para generar el número o puede escribirlo directamente en la casilla de datos a un lado del botón.
Generar q		Genera un número primo con una longitud de 5 dígitos. i Puede utilizar la tecla “Q” como método abreviado para generar el número o puede escribirlo directamente en la casilla de datos a un lado del botón.

Cifrar		Activa el algoritmo de cifrado para el mensaje decimal propuesto por el usuario.
Descifrar		Activa el algoritmo de descifrado para el mensaje decimal generado por el programa o el propuesto por el usuario. i El usuario puede cambiar el mensaje cifrado en la caja de cifrado durante el paso 3
Convertir		Permite al usuario escribir un mensaje en caracteres ASCII (caracteres del teclado) y convertirlo en decimal (formato requerido por Rabin)

Convirtiendo un mensaje ASCII a Decimal.

Las casillas de mensaje de Rabin están divididas en 3 partes para facilitar la lectura y la escritura de mensajes.

La primera casilla denominada “Mensaje” acepta caracteres ASCII y permite convertir su contenido en valores hexadecimales, los cuales se presentan en la casilla de texto “Hex” y en valores decimales en la casilla de texto “Dec” los cuales se actualizan después de utilizar el botón convertir.

Si el usuario no desea escribir un mensaje en ASCII puede escribir directamente sobre la casilla de texto “DEC” el programa interpretará el número escrito como un mensaje válido, pero solo deben escribirse valores decimales.

Activando el panel de trabajo de Rabin.

En el menú principal, coloque el puntero del Mouse sobre la pestaña **Rabin,** luego seleccione del menú emergente **Algoritmo.** Ver figura 2.b.3.

Figura 2.b.3 Menú general para activar panel de trabajo avanzado en Rabin.

Panel de trabajo Rabin

El panel de trabajo Rabin opera de igual forma que el panel de trabajo paso a paso, pero reúne todas las variables que componen el proceso de cifrado/descifrado en una sola pantalla, dando mayor libertar de pruebas y operaciones al usuario, sus funciones y su uso son similares al modo paso a paso.

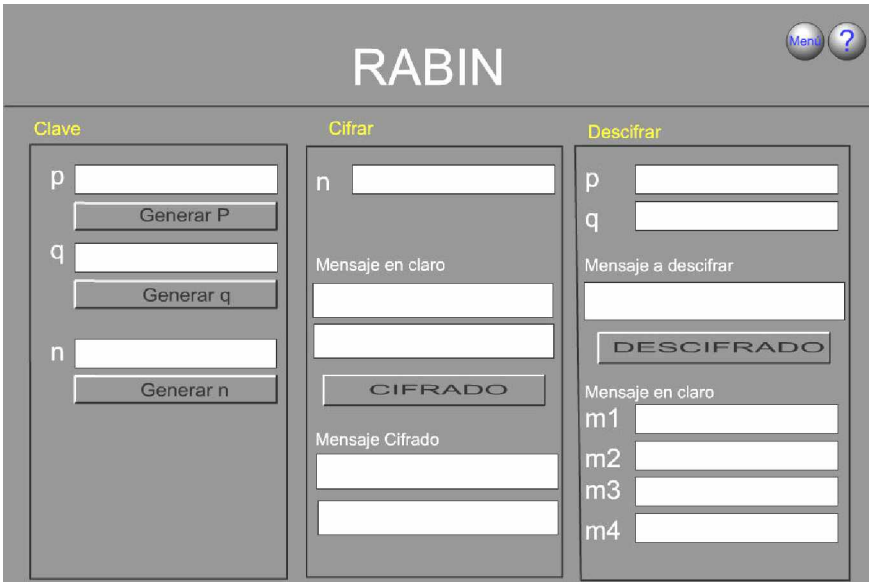


Figura 2.b.4 Panel de trabajo en Rabin para usuarios avanzados

Bibliografía.

[Ang-04]

Ángel, Jesús Ángel. "Red Temática Iberoamericana de Criptografía y Seguridad en la Información", Seguridata, México, 2004.

[Luc-04]

López, Manuel José. "Criptografía y Seguridad en Computadores". Tercera Edición (Versión 2.15). Marzo de 2004.

[Fra-98]

Franco, Jose Pastor. “Criptografia digital fundamentos y aplicaciones”. Prensas Universitarias de Zaragoza. 1998

PARTE 3. FUNCIONES HASH.

3.a SHA1(Secure Hash Algoritm)

Manual de Operación y uso

Introducción

SHA-1 es un algoritmo HASH que toma una cadena de bits (máximo 2^{64} bits) y devuelve un resumen de esta cadena de entrada de 160 bits.

Las principales características de las funciones HASH son:

- No hay dos mensajes diferentes entre si que tengan un mismo resumen.
- Dado un mensaje resumen es imposible regresar al mensaje original. [Kam-98]

Activando SHA-1

Para tener acceso al sistema de resumen SHA-1 posicione el Mouse en la pestaña ***SHA-1*** del menú principal y luego seleccione con un click**.** Ver figura 3.a.1.

Figura 3.a.1 Menú general para activar la función SHA-1.

El panel de trabajo de SHA-1.

El panel de trabajo de SHA1 permite al usuario tener acceso a todas las operaciones que se realizan durante los ciclos de cálculo del resumen, incluyendo algunas opciones que cambian la forma de operación del algoritmo.

A continuación se detallan los componentes con los que cuenta el panel de trabajo de SHA-1 y la forma en la que deben utilizarse.

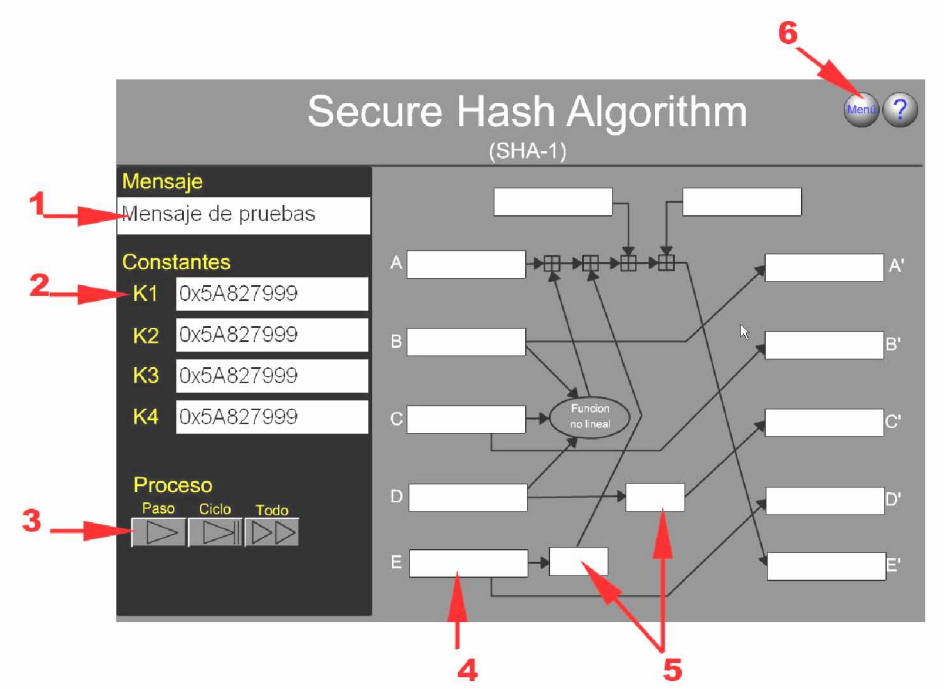


Figura 3.a.2 Panel de trabajo en SHA-1

1. **Casilla de mensaje.** (Fig. 3.a.2)

Mensaje Mensaje de pruebas		Permite escribir el mensaje a resumir. Acepta cualquier tipo de carácter desde el teclado o utilizando las teclas de método abreviado de Windows (Ctrl-V y Ctrl-C). İ Mientras el algoritmo esté procesando un mensaje, esta casilla se mantendrá deshabilitada

2. **Casilla de constantes.** (Fig. 3.a.2)

		Son casillas de textos que permiten al usuario modificar los valores iniciales del SHA-1.

3. **Botones de control.** (Fig. 3.a.2)

		Estos botones permiten realizar y controlar las operaciones matemáticas del resumen del mensaje escrito en la casilla de mensaje (1)
		El botón de paso realiza el proceso de resumen de forma tal que se observan todas y cada una de las operaciones matemáticas.
		El botón de ciclo muestra los resultados parciales del resumen para cada ronda
		El botón todo realiza el proceso de resumen sin mostrar ningún paso intermedio.

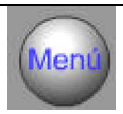
4. **valores entrada/salida por ronda.** (Fig. 3.a.2)

		Las casillas de la “A” a la “E” y de la “A^1” a la “E^1” representan resultados parciales en el proceso de resumen del mensaje.

5. **Casillas de desplazamiento circular.** (Fig. 3.a.2)

		Permite al usuario modificar los desplazamientos que se realizan para las rondas del SHA-1 en el transcurso de un resumen

6. **Menú y Ayuda.** (Fig. 3.a.2)

		Cancela la ejecución del algoritmo y regresa al menú principal.
		Muestra la ayuda de usuario para el algoritmo que esté en ejecución.

Bibliografía.

[Kam-98]

Dimitrios L. KAMBANIS. “Des, one-way hash functions and the md5”.

http://www.cs.nyu.edu/~yingxu/privacy/0303/0303_dkambanis.html, Marzo 1998.

[Per-00]

Peter PERSITS. “Protecting passwords with a one-way hash function”.

http://local.15seconds.com/issue/000217.htm, Febrero 2000.

[STA-98]

William STALLINGS. “Cryptography and Network Security: Principles and Practice”. Segundo Edición. Prentice Hall, 1998.

3.b MD5 (Message Digest).

Manual de Operación y uso

Introducción

MD5 es un algoritmo HASH que toma una cadena de bits (máximo 2^{64} bits) y devuelve un resumen de esta cadena de entrada de 128 bits.

Las principales características de las funciones HASH son:

- No hay dos mensajes diferentes entre si que tengan un mismo resumen.

Dado un mensaje resumen, es imposible regresar al mensaje original. [Riv-92]

Activando MD5

Para tener acceso al sistema de resumen MD5 posicione el Mouse en la pestaña ***MD5*** del menú principal y luego seleccione con un click**.** Ver figura 3.b.1.

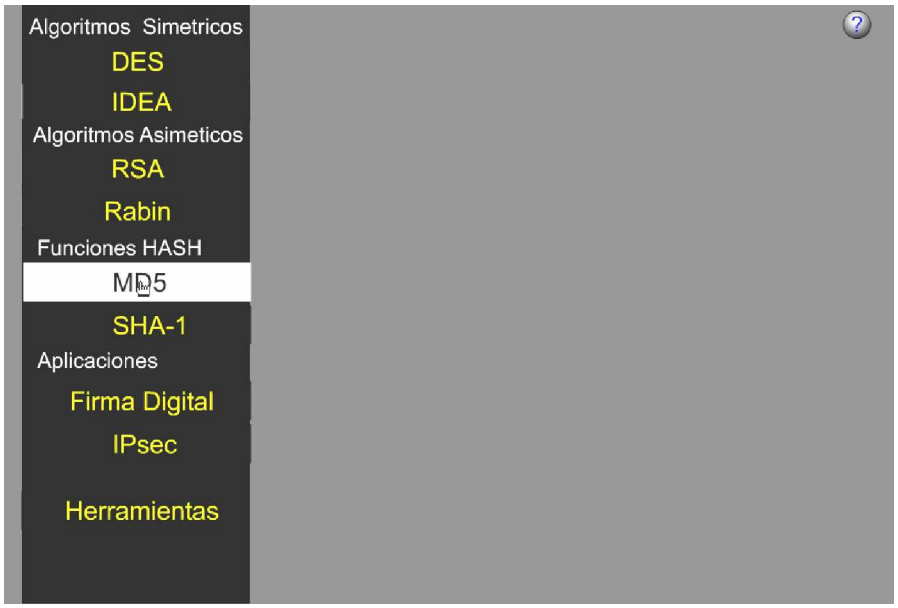


Figura 3.b.1 Menú general para activar la función MD5.

El panel de trabajo de MD5.

El panel de trabajo de MD5 permite al usuario tener acceso a todas las operaciones que se realizan durante los ciclos de cálculo del resumen, incluyendo algunas opciones que cambian la forma de operación del algoritmo.

A continuación se detallan los componentes con los que cuenta el panel de trabajo de MD5 y la forma en la que deben utilizarse.

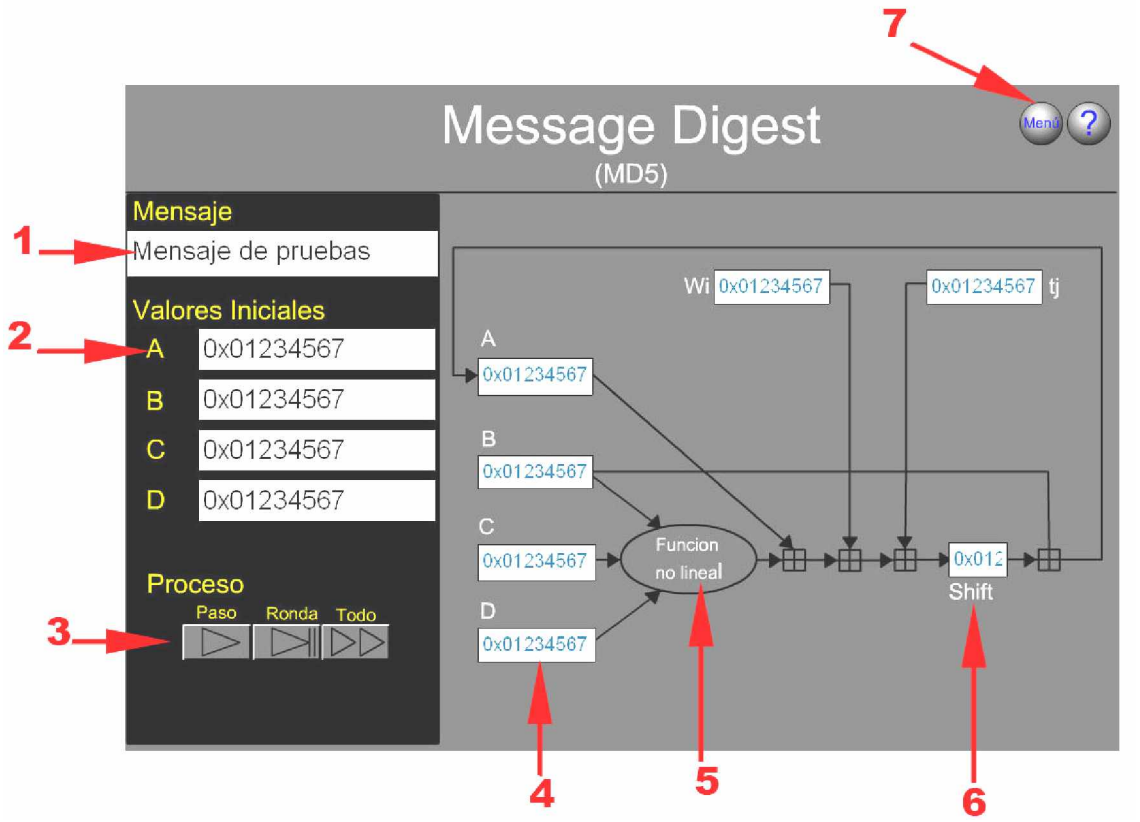


Figura 3.b.2 Panel de trabajo en MD5.

1. **Casilla de mensaje.** (Fig. 3.b.2)

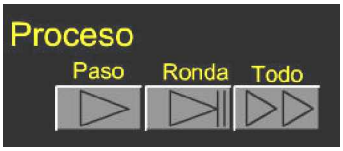
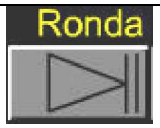
Mensaje Mensaje de pruebas		Permite escribir el mensaje a resumir. Acepta cualquier tipo de caracter desde el teclado o utilizando las teclas de método abreviado de Windows (Ctrl-V y Ctrl-C).

ï Mientras el algoritmo esté procesando un mensaje, esta casilla se mantendrá deshabilitada

2. **Casilla de Valores Iniciales.** (Fig. 3.b.2)

		Son casillas de textos que permiten al usuario modificar los valores iniciales del MD5.

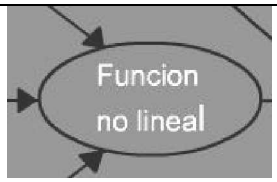
3. **Botones de control.** (Fig. 3.b.2)

		Estos botones permiten realizar y controlar las operaciones matemáticas del resumen del mensaje escrito en la casilla de mensaje (1)
		El botón de paso realiza el proceso de resumen de forma tal que se observan todas y cada una de las operaciones matemáticas.
		Este botón realiza cada una de las 4 rondas del MD5 que se realizan por cada bloque de entrada.
		El botón todo realiza el proceso de resumen sin mostrar ningún paso intermedio.

4. **valores entrada/salida por ronda.** (Fig. 3.b.2)

		Las casillas de la “A” a la “D” presentan los valores iniciales e intermedios que se realizan durante el cálculo del MD5

5. **Función no lineal.** (Fig. 3.b.2)

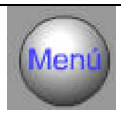
		Esta imagen representa el cálculo de cada nueva variable que se realiza durante el proceso.

i Al posicionar el puntero del ratón sobre esta imagen se observarán los valores de entrada y salida de dicha función.

6. **Casillas de desplazamiento circular.** (Fig. 3.b.2)

		Permite al usuario modificar los desplazamientos que se realizan para las rondas del MD5 en el transcurso de un resumen

7. **Menú y Ayuda.** (Fig. 3.b.2)

		Cancela la ejecución del algoritmo y regresa al menú principal.
		Muestra la ayuda de usuario para el algoritmo que esté en ejecución.

Bibliografía.

[Riv-92]

R. RIVEST. “Rfc: 1321, the md5 message-digest algorithm”.

http://www.faqs.org/rfcs/rfc1321.html, Abril 1992.

PARTE 4. APLICACIONES.

4.a IPSec (Internet Protocol Security).

La animación de IPSec permite al usuario observar todas las operaciones que se realizan durante el uso del Protocolo IPSec. [Igl-01]

A continuación se detallan los componentes con los que cuenta el panel de IPSec y la forma en la que debe utilizarse.

Figura 4.a.1 Panel de animación para IPSec.

1. **Casilla Nombre del Canal.** (Fig. 4.a.1)

Nombre del canal Canal		Muestra el nombre del Canal que se utilizará para la Primera Fase de IPSec.

2. **Dirección IP Remota.** (Fig. 4.a.1)

Direccion IP remota IP		Muestra la Dirección IP con la que se requiere establecer una comunicación segura, mediante IPSec.

3. **Botón Clave/Certificado.** (Fig. 4.a.1)

		Muestra que es posible elegir la forma de autenticado durante la Primera Fase de IPSec, esto puede hacerse mediante un Certificado Digital ó una Clave (Cadena de caracteres).

4. **Botón Clave/Certificado.** (Fig. 4.a.1)

		Esta casilla permite colocar una Clave si es elegida la opción Clave según ítem 3.

5. **Panel para elegir los Algoritmos de Cifrado y Autenticado.** (Fig. 4.a.1)

		Esta casilla permite elegir el algoritmo de Cifrado y la función Hash para autenticado.

6. **Panel para elegir los Algoritmos de Cifrado y Autenticado.** (Fig. 4.a.1)

		Este botón permite iniciar la animación del proceso que lleva a cabo IPSec para establecer una comunicación segura mediante un medio inseguro; además de llevar paso a paso el proceso.

7. **Fase del Proceso de IPSec.** (Fig. 4.a.1)

   Muestra la Fase en la que se encuentra IPSec.

8. **Panel de Proceso de las Fases de IPSec.** (Fig. 4.a.1)

   Muestra las imágenes de cada paso que lleva el proceso de IPSec.

9. **Casilla de Registro.** (Fig. 4.a.1)

   Muestra una descripción del paso que se está visualizando en la Fase actual de IPSec.

Bibliografía.

[HAR-98]

D. HARKINS and D. CARREL. “The Internet Key Exchange”. RFC 2409, noviembre de 1998.

[Igl-01]

Iglesias, Santiago Pérez. “Análisis del protocolo ipsec: el estándar de seguridad en IP”. Telefónica investigación y desarrollo. Noviembre 2002.

4.b Firma Digital.

Manual de uso y operación

Introducción

Una firma digital es un sello electrónico análogo a su versión manuscrita, es decir que es una marca que se agrega a un documento para que el receptor verifique la autenticidad del documento recibido. En sentido estricto una firma digital será una secuencia única de bits calculados a partir de un algoritmo aplicado a un mensaje específico agregados al final de un mensaje para asegurar que su contenido provenga del emisor esperado.

Para aplicar una firma digital a un documento electrónico es necesario utilizar criptografía asimétrica y funciones Hash. El procedimiento por el cual se firma un documento sigue estos pasos:

- El emisor debe generar un resumen del mensaje a firmar utilizando una función hash y luego la debe cifrar con algún algoritmo asimétrico.
- Se emite el mensaje junto con el grupo de datos que forman la firma digital.
- El receptor adquiere el mensaje y descifra la firma digital utilizando la clave pública previamente obtenida del emisor.

El receptor genera su propio resumen del mensaje, sabiendo que únicamente el emisor puede crear un resumen como el que recibió, si ambos resúmenes son iguales el mensaje es auténtico. [Luc-04]

Activando el panel de trabajo de Firma Digital.

En el menú principal, coloque el puntero del Mouse sobre la pestaña **Firma**, luego de un click. Ver figura 4.b.1

Figura 4.b.1 Menú general para activar la aplicación de Firma Digital.

Panel de trabajo de Firma digital.

El panel de trabajo de Firma digital paso a paso permite al usuario realizar una secuencia ordenada de pasos para la correcta generación de la firma. Está dividida en tres pasos fundamentales que se irán activando según se desarrolle el proceso.

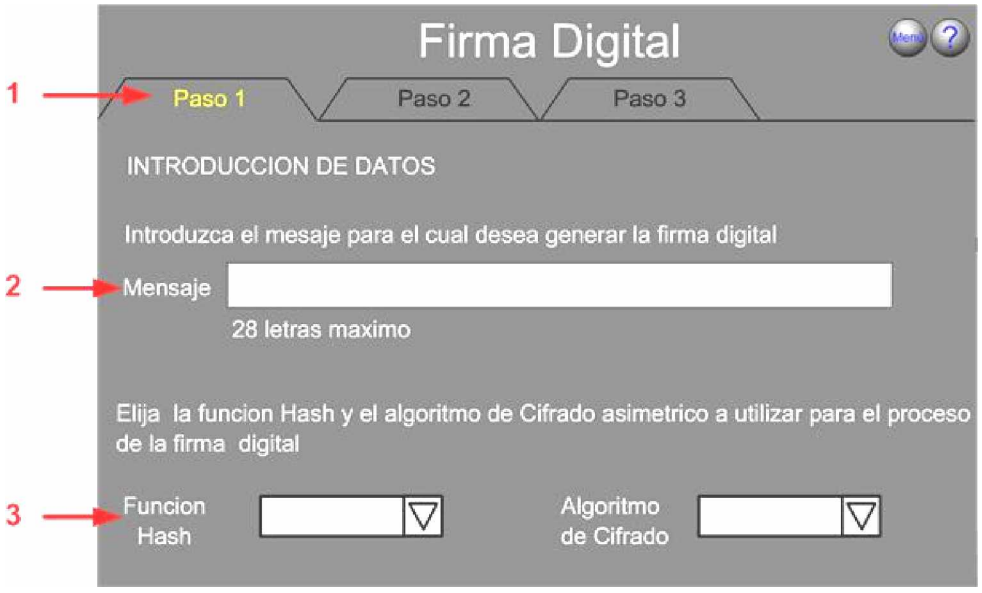


Figura 4.b.2 Paso 1 de la secuencia de la generación de una Firma Digital.

Figura 4.b.3 Paso 2 de la secuencia de la generación de una Firma Digital.

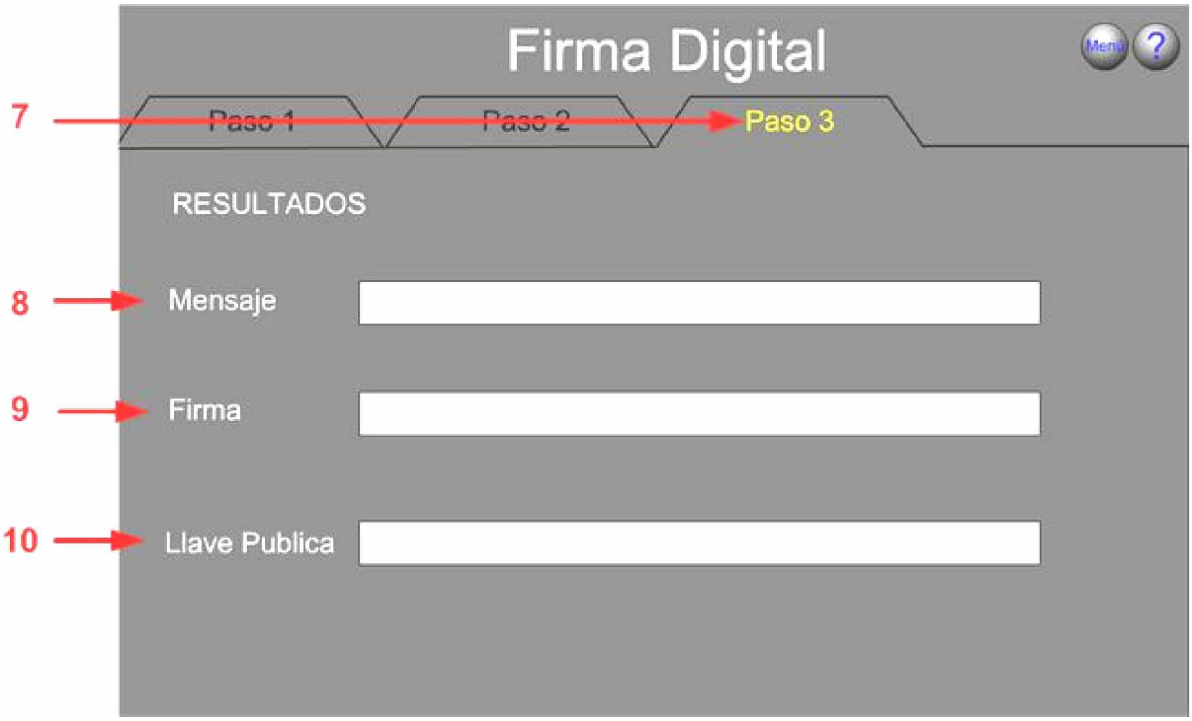


Figura 4.b.4 Paso 3 de la secuencia de la generación de una Firma Digital.

Funciones del panel de trabajo Firma digital paso a paso.

1. **Pestaña de pasos.** (Fig. 4.b.2)

		Permite desplazarse a través de los tres pasos necesarios para cifrar y descifrar un mensaje. Cuando la etiqueta de la pestaña se encuentre en amarillo significa que es el paso que se está ejecutando y no podrán activarse los pasos siguientes a menos que el actual sea terminado. Para cambiar de un paso a otro de un clic sobre la pestaña etiquetada con el paso deseado o presione la tecla TAB.

2. **Casilla de mensaje.** (Fig. 4.b.2)

		Permite escribir el mensaje que se desea firmar Puede dejarse en blanco o escribir hasta un máximo de 28 caracteres.

3. **Selección de algoritmos.** (Fig. 4.b.2)

		Permite seleccionar el algoritmo de cifrado y la función Hash a utilizar en el proceso de creación de la firma.

4. **Paso 2.** (Fig. 4.b.3)

		Permite el acceso al proceso de generación de datos, seleccionados con los controles de función Hash y algoritmo de cifrado (3)

5. **Botón Resumir.** (Fig. 4.b.3)

		Toma el mensaje escrito en la casilla de mensaje y aplica la función Hash seleccionada por el usuario.

6. **Botón Cifrar.** (Fig. 4.b.3)

		Toma el resumen del mensaje y aplica el algoritmo de cifrado seleccionado por el usuario.

7. **Paso 3.** (Fig. 4.b.4)

		Permite el acceso al resultado del proceso de la creación de la firma digital.

8. **Mensaje.** (Fig. 4.b.4)

Muestra el mensaje original introducido por el usuario.

9. **Firma Digital.** (Fig. 4.b.4)

Muestra la firma digital resultante del proceso de creación utilizando los valores seleccionados por el usuario.

10. **Llave Pública.** (Fig. 4.b.4)

Muestra la llave publica, la cual se enviara al receptor para el proceso de autenticado.

Bibliografía.

[DNT-00]

Derecho de las Nuevas Tecnologías - La firma digital

http://www.tuguialegal.com/firmadigital1.htm, octubre de 2000

[Luc-04]

López, Manuel José. "Criptografía y Seguridad en Computadores". Tercera Edición (Versión 2.15). Marzo de 2004.

BIBLIOGRAFÍA

[Ang-04]

Ángel, Jesús Ángel. "Red temática iberoamericana de criptografía y seguridad en la información", Seguridata, México, 2004.

[DNT-00]

TuGuiaLegal.com “Derecho de las nuevas tecnologías - La firma digital” http://www.tuguialegal.com/firmadigital1.htm, octubre de 2000

[FIPS PUB 180-1]

FIPS PUB 180-1. “Announcing the standard for secure hash standard”. http://www.itl.nist.gov/fipspubs/fip180-1.htm, Mayo 1993.

[Fra-98]

Franco, Jose Pastor. “Criptografia digital fundamentos y aplicaciones”. Prensas Universitarias de Zaragoza. 1998

[HAR-98]

Harkins, D. “The Internet Key Exchange”. RFC 2409, noviembre de 1998.

[Igl-01]

Iglesias, Santiago Pérez. “Análisis del protocolo ipsec: el estándar de seguridad en IP”. Telefónica investigación y desarrollo. noviembre 2002.

[Kam-98]

Kambanis, Dimitrios L.. “Des, one-way hash functions and the md5”.

http://www.cs.nyu.edu/~yingxu/privacy/0303/0303_dkambanis.html, Marzo 1998

[Luc-04]

López, Manuel José. "Criptografía y seguridad en computadores". Tercera Edición (Versión 2.15). Marzo de 2004.

[Men-04]

Méndez Medina, Yerai. "Red Temática Iberoamericana de criptografía y seguridad en la información", Universidad de la Laguna, España, 2004.

[Mor-04]

Moreno, Luciano. “Criptografía”. Cap VII. BJS Software.

http://www.htmlweb.net/seguridad/cripto/cripto_1.html. Septiembre 2000.

[Per-00]

Persits, Peter. “Protecting passwords with a one-way hash function”.

http://local.15seconds.com/issue/000217.htm, Febrero 2000.

[Riv-92]

Rivest R. “Rfc: 1321, the md5 message-digest algorithm”.

http://www.faqs.org/rfcs/rfc1321.html, Abril 1992

[Sch-96]

Schneier, Bruce. “Applied cryptography”, Segunda Edición, Mineapolis, 1996

[STA-98]

Stallings, William. “Cryptography and Network Security: Principles and Practice”. Segundo Edición. Prentice Hall, 1998.

[Val-04]

Valeiras Reina, Gerardo. "Red Temática iberoamericana de criptografía y seguridad en la información", Universidad de Sevilla, España, 2004

ANEXO I

Herramientas Matemáticas

Los números naturales y los números enteros

Los números naturales y los números enteros son objetos conocidos. El lector está familiarizado con dichos objetos y con sus operaciones:

$$\tilde{a} = \{1, 2, 3, 4, ...\} \qquad Ec.39$$

$$\grave{a} = \{0, 1, -1, 2, -2.....\} \qquad Ec.40$$

Como primer objetivo se introducirán de forma rigurosa estos objetos. Existen varias maneras de hacerlo, una de ellas es la axiomática. Consiste en caracterizar el conjunto N de los números naturales por algunas de sus propiedades que se imponen como axiomas, de manera que cualquier otra propiedad se deduce de estos axiomas.

Se puede usar la caracterización de N como cualquier conjunto que verifique los llamados Axiomas de Peano:

- En **ã** hay un elemento distinguido que denominamos 1.
- Para cada $n \in$ **ã** se define de manera única el *siguiente de n*.

  Se denota *s*(*n*). Es un elemento de **ã** y verifica que $s(n) \neq 1$ para cada $n \in$ **ã**
- Si $s(n) = s(m)$ entonces *n* = *m*.

En este conjunto se puede definir una primera operación denominada suma, de manera que dados dos naturales $a,b \in$ **ã** cualesquiera se puede construir el número natural $a+b \in$ **ã** que es la suma de los dos. Esta operación se define por las reglas:

- Para cada $a \in$ **ã** se define $a+1=s(a)$
- Para cada $a,b \in$ **ã** se define $a+s(b)=s(a+b)$

Las propiedades que verifican los números naturales con esta operación de suma son las siguientes:

- Propiedad asociativa: $(a+b)+c=a+(b+c)$ para cualesquiera $a,b,c \in$ **ã** .
- Propiedad conmutativa: para cualquier pareja $a,b \in$ **ã** se tiene que $a+b=b+a$.

También se puede definir una segunda operación en **ã** que es el producto, asignando a cada par $a,b \in$ **ã** el producto de ambos $a¥b$ (ó $a \times b$ ó simplemente *ab*). Las reglas que se presentan a continuación son suficientes para definir el producto de dos números naturales $a,b \in$ **ã** .

- Para cada $a \in$ **ã** se tiene que $a \times 1=a$.
- Para cada $a,b \in$ **ã** se tiene que $a \times s(b)=a \times b+a$.

Los números naturales con esta operación de producto tienen las siguientes propiedades y reglas:

- propiedad asociativa: $(a\times b)\times c = a\times(b\times c)$ para cualesquiera $a,b,c\in$ **ã** ,
- existencia de elemento neutro: existe $1\in$ **ã** tal que, para cualquier número natural $a\in$ **ã** se tiene que $a\times 1 = 1\times a = a$
- propiedad conmutativa: para cualquier pareja $a,b\in$ **ã** se tiene que $a\times b = b\times a$.

Las dos operaciones, la suma y el producto, quedan relacionadas por la propiedad distributiva:

- propiedad distributiva: para cualesquiera $a,b,c\in$ **ã** se tiene que $a\times(b+c) = a\times b + a\times c$

Aritmética Modular. Propiedades

La aritmética modular es una parte de las Matemáticas extremadamente útil en Criptografía, ya que permite realizar cálculos complejos, manteniendo siempre una representación numérica compacta y definida, puesto que sólo maneja un conjunto finito de números enteros.

En forma rigurosa: Dados tres números $a,b,n\in$ **ã** decimos que a es congruente con b módulo n, y se escribe:

$$a \equiv b \bmod n \qquad Ec.41$$

Si se cumple:

$$a = b + kn \text{, para algún } k\in \textbf{à} \qquad Ec.42$$

Se define una clase de equivalencia como aquel conjunto de números que sean congruentes con un módulo n cualquiera. De esta forma un número entero pertenece necesariamente a alguna de esas clases, y en general

tendremos n clases de equivalencia módulo n (números congruentes con 0, números congruentes con 1, . . . , números congruentes con n-1). Por razones de simplicidad, representaremos cada clase de equivalencia por un número comprendido entre 0 y n - 1.

Las operaciones suma y producto en este tipo de conjuntos se pueden definir así:

$$a+b \equiv c \bmod n \Leftrightarrow a+b=c+kn \qquad k \in \mathbf{à} \qquad Ec.43$$

$$a \times b \equiv c \bmod n \Leftrightarrow a \times b=c+kn \qquad k \in \mathbf{à} \qquad Ec.44$$

Propiedades de la suma:

- Asociativa: $\forall a,b,c \in \mathbf{à} \quad (a+b)+c \equiv a+(b+c) \bmod n$
- Conmutativa: $\forall a,b \in \mathbf{à} \quad a+b \equiv b+a \bmod n$
- Elemento Neutro: $\forall a \in \mathbf{à} \quad \exists\, 0 \text{ tal que } a+0 \equiv a \bmod n$
- Elemento Simétrico: $\forall a \in \mathbf{à} \quad \exists\, b \text{ tal que } a+b \equiv 0 \bmod n$

Propiedades del producto:

- Asociativa: $\forall a,b,c \in \mathbf{à} \quad (a \times b) \times c \equiv a \times (b \times c) \bmod n$
- Conmutativa: $\forall a,b \in \mathbf{à} \quad a \times b \equiv b \times a \bmod n$
- Elemento Neutro: $\forall a \in \mathbf{à} \quad \exists\, 1 \text{ tal que } a \times 1 \equiv a \bmod n$

Propiedades del producto con respecto de la suma:

- Distributiva: $\forall a,b,c \in \mathbf{à} \quad (a+b) \times c \equiv (a \times c)+(b \times c) \bmod n$

Teorema de la división

Aunque los números enteros no se pueden dividir, porque los cocientes no son enteros, sí se puede hacer una división entera, obteniéndose un cociente y un resto. Esta división nos va a permitir por un lado construir un algoritmo para el cálculo del máximo común divisor de dos números sin necesidad de factorizarlos y por otro establecer una relación de equivalencia, la congruencia módulo un entero, Antes de enunciar y demostrar el teorema del resto debemos detenernos un momento a reflexionar sobre el orden que presentan los números naturales y los números enteros es necesario definir lo que es una relación de orden.
Sea *A* un conjunto, se define una relación en el conjunto *A* como un subconjunto *R* del producto cartesiano $A\times A=\{(a,b):a,b\in A\}$, de modo que dos elementos $a,b\in A$ están relacionados si y solamente si la pareja (*a, b*) está en *R.*

Así, de esta forma el enunciado del teorema del resto es el siguiente: Sean a y b dos números enteros positivos, entonces existen dos enteros q (cociente) y r (resto) únicos tales que $q\geq 0,\qquad 0\leq \mathrm{r}<\mathrm{b}$ y se tiene la expresión

$$a=bq+r \qquad Ec.45$$

Esto es, dividendo es igual a divisor por cociente más resto. A éste teorema podemos agregarle la siguiente definición:

Sean a y b dos números enteros, se dice que *a* divide a *b* y se escribe *a*|*b* si existe un número entero *c* tal que *b* = *a x c*. También decimos que *b* es múltiplo de *a* ó que *a* es un factor de *b*. Es decir, en el caso divisible, se puede efectuar la división *b* entre *a* y se obtiene un número entero, esto es, al hacer la división entera entre *b* y *a* el resto es 0.

Utilizando la definición de la división entera se puede decir que un número primo es un entero positivo $a\neq 1$ si sus únicos factores positivos son *a* y 1. Si *a* tiene un factor positivo distinto de *a* ó de 1 se dice que es compuesto.

Algoritmo de Euclides

Quizás sea el algoritmo más antiguo que se conoce, y a la vez es uno de los más útiles. Permite obtener de forma eficiente el máximo común divisor de dos números.

Sean a y b dos números enteros de los que queremos calcular su máximo común divisor m. El Algoritmo de Euclides explota la siguiente propiedad:

$$m \mid a \ \wedge \ m \mid b \Rightarrow m \mid (a - kb) \quad \text{con} \quad k \in \mathbf{à} \Rightarrow m \mid (a \bmod b) \qquad Ec.46$$

En esencia estamos diciendo que puesto que m divide tanto a *a* como a *b*, debe tambien dividir a su diferencia. Entonces si restamos *k* veces *b* de *a*, llegará un momento en el que obtengamos el resto de dividir *a* por *b*, o sea *a mod b*.

Si llamamos *c* a *(a mod b),* podemos aplicar de nuevo la propiedad anterior y tenemos:

$$m \mid (b \bmod c) \qquad Ec.38$$

Sabemos pues, que m tiene que dividir a todos los restos que vayamos obteniendo. Es evidente que el último de ellos será cero, puesto que los restos siempre son inferiores al divisor. El penúltimo valor obtenido es el mayor número que divide tanto a *a* como a *b*, o sea el máximo común divisor de ambos.

Cálculo de Inversas en à

Dos números enteros a y b se denominan primos entre sí (o co-primos), si su máximo común divisor es igual a 1. la definición anterior conduce al siguiente teorema:

Si mcd(a, n) = 1, a tiene inversa módulo n.

Es decir que cualquier par a,n de números cuyo máximo común divisor sea 1, a tendrá inversa modulo n

Conclusiones

- No es posible mostrar los algoritmos asimétricos en forma de flujograma, por lo tanto es mejor estructurarlos como un conjunto de pasos.

- El desarrollo de una aplicación demostrativa de los algoritmos, permite la comprensión de diversas técnicas de comunicación que se utilizan actualmente para garantizar la seguridad e integridad de la información.

- Un complemento necesario para comprender los algoritmos es estudiar o conocer las reglas básicas de la matemática modular, ya que los procesadores que se utilizan realizan todos sus cálculos en este sistema. Por tanto es conveniente aprender matemática discreta antes del estudio de criptografía.

- Para facilitar el aprendizaje de los algoritmos fue necesaria la utilización de diferentes lenguajes de programación y visualización; unos para realizar los cálculos de forma eficiente tales como rotaciones y desplazamiento, otros para presentar los gráficos y un tercero para controlar y enlazar a los dos anteriores.

- Para la comprensión de IPsec es necesario el entendimiento de otros protocolos los cuales generalmente no son difíciles de comprender. Por lo tanto se prefiere una animación de IPsec para explicar su funcionamiento y hacer más fácil la comprensión del mismo.

- Durante la implementación de los algoritmos se hizo evidente que algunos de estos, necesitan un relleno en los mensajes para complementar las longitudes requeridas. Por lo que se añade a la aplicación entregada un apartado para la comprensión de estos complementos.

- El programa desarrollado implementa una diversidad de algoritmos mostrando la gran cantidad de características las cuales resulta difícil encontrar en otros programas didácticos.