ESTUDIO MONOGRÁFICO DE LOS SISTEMAS DE ENCRIPTAMIENTO Y DESARROLLO DE UNA APLICACIÓN

PROYECTO DE GRADUACIÓN PARA OPTAR AL GRADO DE

INGENIERO ELECTRÓNICO

PRESENTADO POR :

JOSÉ ARMANDO HERNÁNDEZ SAAVEDRA

CUIDADELA DON BOSCO AGOSTO DE 1999

RECTOR

ING. FEDERICO MIGUEL HUGUET RIVERA

SECRETARIO GENERAL:

PBRO. PEDRO JOSÉ GARCIA CARCÍA

DECANO DE LA FACULTAD DE INGENIERÍA:

ING. CARLOS GUILLERMO BRAN

ASESOR DEL PROYECTO DE GRADUACIÓN:

ING. ROBERTO CARLOS ALVARENGA URIAS

ING. ENRIQUE AGUILAR
JURADO EVALUADOR

ING. WENCESLAO RIVAS
JURADO EVALUADOR

CUIDADELA DON BOSCO

AGOSTO DE 1999

FACULTAD DE INGENIERIA

ESCUELA DE INGENIERIA ELECTRONICA

JURADO EVALUADOR DEL PROYECTO DE GRADUACIÓN

ESTUDIO MONOGRÁFICO DE LOS SISTEMAS DE ENCRIPTAMIENTO Y DESARROLLO DE UNA APLICACIÓN

ING. ENRIQUE AGUILAR
JURADO EVALUADOR

ING. WENCESLAO RIVAS
JURADO EVALUADOR

ING. ROBERTO CARLOS ALVARENGA
ASESOR DEL PROYECTO

CUIDADELA DON BOSCO AGOSTO DE 1999

DEDICATORIA:

A Dios Todopoderoso:

Que me da la fe, la vida y todo lo que Él sabe que necesito para desarrollar cada uno de los proyectos en mi vida.

A mis Padres:

Lic. José Armando Hernández Rivas
Dra. Ana Estela Saavedra de Hernández

Por su apoyo siempre constante
en este proyecto de Ingeniería
y en otros proyectos realizados
a lo largo de mi vida.

José Armando Hernández Saavedra.

INDICE

INFORMACIÓN PRELIMINAR

El contenido de la información digitalizada que se transmite a través de las redes de computadoras puede ser de suma importancia para las empresas o personas que la envían y la reciben , pues contiene datos sumamente confidenciales y transacciones monetarias importantes tales como la compra y venta de bienes y servicios a través de la red. No sólo es importante que la información llegue íntegra hasta su receptor, sino también que goce de privacidad durante el proceso de comunicación a fin de que ésta no sea conocida o desvirtuada deliberadamente por personas ajenas a el receptor o el transmisor.

El encriptamiento por su capacidad de alterar la información hasta hacerla aparentemente un conjunto de bits sin ningún sentido ni coherencia, minimiza este riesgo evitando que que la información sea conocida o manipulada por extraños, ya que la información encriptada aunque sea accesada por otro usuario distinto a su legítimo destinatario, es prácticamente inservible si no se tiene la clave para su desencriptamiento.

Un estudio amplio de todos aquellos factores relacionados con el encriptamiento se presenta a continuación en las páginas que suceden a este apartado. No obstante debe tenerse en consideración que este documento es simplemente el desarrollo de un proyecto académico para optar al título de Ingeniero Electrónico, y por lo tanto las directrices académicas que han manejado su desarrollo, tanto en el punto focal del estudio, como en la orientación de su desarrollo han determinado sus límites, sus alcances, sus objetivos y sus metas.

Con el fin de mostrar toda esta información de orden puramente académico, y para dejar claro al lector el contenido del documento, y los motivos origen de dicho contenido, se expone en este primer apartado toda aquella información que se detalló en la redacción del anteproyecto, documento que fue aprobado y que constituye una promesa en la realización de todo el proyecto.

1.- IMPORTANCIA Y JUSTIFICACIÓN DEL TEMA

No sólo la privacidad es el único objetivo del encriptamiento, pues en toda transferencia de datos es necesario verificar el origen, el destino, y la integridad de la información que se transmite. El encriptamiento

puede cumplir estas funciones siendo garantía de una correcta comunicación, evitando malos entendidos, fuga o desvirtuamiento de la información, o falsos documentos que se envíen bajo un nombre distinto al de su verdadero remitente.

El encriptamiento no es una labor que se la pueda considerar completamente terminada- El trabajo de los desencriptadores de información alcanza en algunas ocasiones los logros más elaborados de las compañías de informática dedicadas a esta labor de protección y seguridad.

Siempre es importante y valiosa la creación de nuevos sistemas de protección de información, -aún cuando los actuales no hayan sido violados -, ya que de esta manera se toma ventaja sobre los espías desencriptadores al no permitirles tiempo para que puedan encontrar la clave para el desencriptamiento. Aportes a la tecnología del encriptamiento como el que aquí se plantea siempre son útiles, ya que proporcionan un paso más en la evolución de los sistemas de seguridad de la información.

2.- DEFINICIÓN DEL TEMA

2.1 DEFINICIÓN DEL TEMA

EL PROBLEMA:

La necesidad de protección a la privacidad de la información que se transmite a través de medios de comunicación que enlazan equipos terminales de datos digitales plantea un serio problema en los sistemas de información; es necesario proteger la privacidad y la integridad de la información, la identificación de su origen, y la certeza de haber llegado íntegra hasta su correcto destinatario.

LA SOLUCIÓN:

La más completa solución la brinda el encriptamiento, que por su capacidad para hacer del formato de la información algo en apariencia ilegible e incoherente, proporciona una protección eficaz contra terceras personas no deseadas en la transferencia de datos. El desarrollo de esta solución obliga a desglosar el problema para poder desarrollar un sistema de protección efectivo. Esto conlleva a un estudio preliminar de la historia del encriptamiento y de los principios matemáticos en que se basa, así como también la situación actual de los sistemas en uso para desarrollar un nuevo sistema eficiente y competitivo.

2.2- OBJETIVO GENERAL

Identificar y describir los sistemas de encriptamiento que se han usado y que se usan , abarcando los principios matemáticos y técnicos en que se basan para hacer una valoración de cada uno de ellos en cuanto

a sus alcances y en cuanto a sus posibilidades de desencriptamiento por extraños. Todo esto en vistas a evaluar lo hasta hoy creado para el diseño e implementación de un nuevo sistema de encriptamiento completamente original y distinto.

2.3- OBJETIVOS ESPECÍFICOS

- Enumerar y dar a conocer en profundidad los sistemas de encriptamiento más usuales y que en su debida oportunidad fueron los más efectivos.
- Listar las formas más usuales con las cuales los desencriptadores (Hackers o cripto-analistas) logran descifrar los mensajes encriptados.
- Investigar el desarrollo de las técnicas de encriptamiento junto con el desarrollo tecnológico tanto del software como del hardware.
- Exponer todo el aspecto matemático asociado a los sistemas de encriptamiento con el propósito de plantear la creación de un nuevo sistema de encriptamiento para la protección de información en el actual entorno de hardware, software y cripto-análisis que se practica.
- Elaborar un informe de la situación actual del encriptamiento tanto como sea posible, teniendo en cuenta que lo último desarrollado en este campo es guardado sigilosamente por sus fabricantes.
- Crear un sistema de encriptamiento completamente original.

2.4 ENFOQUE DEL TEMA

Al abordar el tema de la seguridad de la privacidad de la información se debe tener en cuenta que se hace desde el objetivo de crear un sistema de encriptamiento, esto obliga a que el enfoque sea eminentemente técnico-científico y al mismo tiempo táctico-estratégico. Para llevar a buen término el objetivo planteado es necesario conocer datos técnicos acerca del funcionamiento de computadoras conectadas en forma de red, las técnicas utilizadas para la transmisión de datos, el aspecto matemático asociado con el encriptamiento y todo lo referente a el funcionamiento de redes de computadoras en cuanto a la intermigración de datos se refiere.

La valoración del sistema de protección de información se lo dan las entidades a quienes sirve con sus funciones, esto obliga a que se tome en cuenta todos aquellos factores sociales y mercantiles que resten efectividad a la valoración de su función de protección . Esto significa que el tema del encriptamiento también debe responder a los problemas actuales que enfrentan las empresas, tales como la competencia desleal y la fuga de información industrial.

2.5- ALCANCES Y LIMITACIONES

2.5.1 ALCANCES:

- La investigación acerca de los sistemas de seguridad alcanzará todo lo relativo al software y al hardware de redes de computadoras . Se pretende investigar qué mecanismos protegen la información en una red, pero no profundizar hasta el funcionamiento lógico de dichos mecanismos.
- En cuanto a tipos de encriptamiento y sistemas de encriptamiento, no se pretende conocer en detalle cada paso del proceso de encriptamiento y desencriptamiento, sino únicamente las generalidades que determinan su funcionamiento y diversificación.
- Al hablar de la técnicas para el desencriptamiento sin el conocimiento de la clave, no se investigarán los métodos completos y detallados para el ataque a un sistema de seguridad, se investigarán únicamente las técnicas más comunes que son utilizadas para este fin.

2.5.2 LIMITACIONES

- La información verdaderamente actual es inaccesible.
- Debido a su delicada labor , los sistemas de seguridad de última tecnología en uso actualmente son ocultados a todo extraño en cuanto a los detalles más recónditos y peculiares de su funcionamiento (el fabricante sólo expone generalidades de su producto) . La única información completamente disponible es la de la generación anterior de sistemas de seguridad, y todo lo anterior a ella.

3.- RESULTADOS ESPERADOS

Desde el punto de vista del anteproyecto de este trabajo de graduación, la investigación pretende recabar datos para el planteamiento de la implementación de un sistema de encriptamiento original y distinto. Se espera que toda la información aquí recolectada sea de utilidad para este fin, y sirva como base para el desarrollo de un criterio de selección de los mejores elementos que pueden ser utilizados para la creación de un sistema nuevo de encriptamiento.

La información proporcionada acerca del desarrollo del encriptamiento así como las técnicas que son utilizadas por los Cripto-analistas para descifrar la información, son una guía para rechazar aquellas herramientas de trabajo que han demostrado ser de poca utilidad en el desarrollo de sistemas de protección de información. basados en estos criterios se espera escoger lo mejor en cuanto a técnicas para el encriptamiento.

El proyecto definitivo pretende crear un sistema completamente nuevo de encriptamiento[A], el cual será presentado compuesto de dos partes: un elemento encriptador y un elemento desencriptador.

El elemento encriptador funcionará como un programa ejecutable que codifica la información de un archivo no ejecutable (archivo texto o de otra naturaleza), creando de esta manera un archivo encriptado. Este archivo podrá ser depositado ya sea en un medio de almacenamiento (diskdrive) o ser transferido a un puerto de entrada-salida tal como lo es cualquier medio de conexión con la red.

De manera semejante el elemento desencriptador funcionará como un programa ejecutable que decodifica la información de un archivo encriptado o la información encriptada procedente de un medio de almacenamiento o de un puerto de entrada-salida para convertirla en un archivo de lectura clara para el receptor de la información .

A Cabe destacar en este apartado, que al principio del proceso de graduación el proyecto únicamente se limitaba a la creación de un sistema de encriptamiento original y distinto. El anteproyecto de graduación contenía en si mismo mucha información para el planteamiento de la implementación del algoritmo.
Consideraciones de forma en el documento del proyecto obligaron a incluir un estudio monográfico que acompañe al desarrollo de la aplicación para crear un trasfondo para referencia de todos los elementos que se incluyen en la aplicación.
El estudio monográfico retomó muchos de los elementos conceptuales del anteproyecto, pero además mejoró el grado de detalles en la información con el fin no sólo de presentar todo lo útil para el desarrollo de la aplicación, sino también todo lo referente a el amplio mundo del encriptamiento.

INTRODUCCIÓN

En el actual ámbito de tecnología de comunicaciones, Internet se ha convertido en un valioso medio de comunicación que permite interacciones multimedia entre sus participantes, en este entorno el encriptamiento surge como la única alternativa confiable capaz de proteger efectivamente todos los datos que circulan a través de un medio tan amplio y al mismo tiempo tan inseguro.

Encriptamiento o Criptografía significa etimológicamente "ocultar lo escrito" (de 'Cripto': ocultamiento, y 'grafos': escrito). El encriptamiento de la información digital en su función más inmediata tiene la misión de hacer ilegible cualquier documento o mensaje. No obstante se puede ir un paso más allá en la seguridad de la información si se consigue que el formato de la información encriptada sea en apariencia una secuencia de bytes de comportamiento absolutamente aleatorio.

Este documento presenta de una manera detallada un estudio monográfico de los sistemas de encriptamiento, y además expone todos los elementos que propiciaron el desarrollo de una aplicación en software completamente original, tanto los elementos definitorios de los criterios de diseño, como aquellos que sirvieron como materia prima y herramientas para su construcción.

La parte del estudio monográfico es una útil ayuda para comprender de una manera clara todo lo relacionado con los sistemas de encriptamiento. Se puede resumir el contenido de este estudio como una respuesta amplia, clara y detallada a las siguientes interrogantes:

- ¿ Porqué y para qué existe la criptografía ?
- ¿ Cómo se ha desarrollado antes y cómo se realiza hoy el encriptamiento, de qué manera transforma y devuelve el contenido de la información, cuántas formas distintas existen para realizarlo?
-¿ Cómo se utilizan los procedimientos de encriptamiento en el entorno actual de información digital?

En cuanto a lo referente al desarrollo de la aplicación, se expone en primer lugar una teoría matemática del desarrollo de las funciones de encriptamiento. Se trata de observaciones a distintos algoritmos de encriptamiento sistematizadas en la forma de teoría del comportamiento de las funciones de encriptamiento. Sobre esta base conceptual se desarrolla en segundo lugar todo lo referente a la aplicación en software, comenzando por el planteamiento de las generalidades de su comportamiento, su algoritmo de encriptamiento, su manera de manipular los datos, hasta los más pequeños detalles de su funcionamiento como programa en lenguaje C para ser compilado en la forma de un archivo ejecutable.

1.- SEGURIDAD EN LOS SISTEMAS DIGITALES DE INFORMACIÓN:

Como un primer acercamiento al ámbito en el cual el encriptamiento se desarrolla como parte integral de los sistemas de seguridad de la informacion, esta sección expone brevemente la importancia de la seguridad en dichos sistemas, los niveles de seguridad que se pueden desarrollar, y las propiedades indispensables que un sistema de seguridad confiable debe proveer a la información protegida.

1.1 NECESIDAD DE SEGURIDAD EN LOS SISTEMAS DE INFORMACIÓN:

Los sistemas de que gozan de mayor relevancia para ser vigilados cuidadosamente por los sistemas de seguridad de la información pueden ser divididos en tres categorías, a continuación se expone las necesidades de seguridad de cada categoría para realizar confiablemente todas sus labores.

1.1.1 SEGURIDAD COMERCIAL:

Para poder llevar a cabo gestiones comerciales a través de medios de comunicación digitales es necesario que la información comercial , además de poseer estricta confidencialidad, sea correctamente autenticada para evitar situaciones peligrosas tales como:

- Falsas requisiciones de mercadería o servicios.
- Cobros o transacciones perjuras de dinero electrónico(web-banking).
- Duplicación de cobros o transacciones con fines de duplicar ganancias.
- Identificaciones ficticias del comprador o vendedor, realizando transacciones sin mercancía o sin dinero verdadero.

1.1.2 SEGURIDAD MILITAR Y/O POLICIAL:

Las gestiones de seguridad nacional que realizan las fuerzas armadas o policiales no deben ser conocidas en detalle por el ciudadano común. Las estrategias militares y policiales necesitan guardar su confidencialidad para poder surtir los efectos requeridos en el cumplimiento de una misión.

En todo patrullaje o ataque agresivo es necesario coordinar acciones conjuntas con ayuda de canales de comunicación seguros que impidan fugas o intrusiones de información por parte del enemigo. La autenticación y no repudio de la información son de vital importancia. De no existir una comunicación segura podría darse la posibilidad de ignoración de ordenes superiores, además de cualquiera de las siguientes falsas comunicaciones con mandato irrevocable de :

- Ataque (que guíe al conjunto hacia una emboscada).
- De retirada.
- De resistencia pasiva.
- De inicio de guerra nuclear.

1.1.3 SEGURIDAD EN LOS ARCHIVOS DEL SECTOR PUBLICO:

Es necesario proteger los archivos del sector público para evitar adulteraciones en los mismos, alteraciones que pueden ser frecuentes en las áreas de catastro (falsos registros de propiedad), identificación civil (falsas cédulas o pasaportes), sector judicial (actas judiciales), sector notarial (falsos poderes), corte de cuentas, etc. Estas modificaciones frauduluentas pueden ser efectuadas ya sea en el mismo sistema o durante una transferencia de datos con otros sistemas digitales de información en otras dependencias del sector público, su protección se hace necesaria para evitar situaciones comprometedoras de la vida de los ciudadanos.

1.2 NIVELES DE SEGURIDAD PARA LA INFORMACIÓN DIGITAL :

Los niveles de seguridad para la información se pueden enumerar , en orden ascendente en cuanto a la capacidad para proteger la información, de la siguiente manera [1] :

- Nivel 0 : No seguridad: sólo existe seguridad física para las unidades del sistema.
- Nivel 1: Bloqueo del teclado/diskdrive: (mediante una llave).
- Nivel 2: Loggin de Usuario o Password: cualquiera de los dos.
- Nivel 3: Loggin de Usuario y Password: dos identificaciones.
- Nivel 4: Loggin de Usuario, Password y Registro de Sesiones
- Nivel 5: Encriptamiento: La información confidencial guardada por la computadora en un medio de almacenamiento, o los archivos recibidos y enviados son ocultados bajo un código secreto, de

manera que estos datos sólo pueden ser conocidos por las personas poseedoras del algoritmo para descifrar la información que está bajo el código secreto y de la clave utilizada en el mismo.

- Nivel 6: Sistemas de Seguridad En el Hardware: Ofrecen todos los niveles anteriores sin necesidad de ser instalados en el sistema operativo o en programas especializados.

En cuanto a la seguridad por contraseña, se pueden tener dos modelos de niveles que puede proteger un password [2]:

1.- Nivel de Usuario: cuando el mecanismo que protege al sistema es la limitación del acceso del usuario a los sitios de la red que le han sido asignados.

2.- Nivel de recursos: cuando el password limita el acceso a los archivos o aplicaciones que protege sin tomar en cuenta el estatus del usuario:

Usualmente se utilizan ambos niveles para restringir el acceso dentro de la red a un usuario o a un conjunto de usuarios, según las necesidades de seguridad de la empresa que usa el sistema.

1.3 REQUISITOS FORMALES DE LA INFORMACIÓN SEGURA.

Para que toda información se pueda llamar segura, es necesario cumplir cuatro requisitos indispensables para garantizar que ningún daño se efectúe en la integridad de los sistemas de información de los participantes en la comunicación [3] :

a. CONFIDENCIALIDAD: Seguridad de que la información enviada no va a ser conocida por extraños.
b. AUTENTICACION: Seguridad que la información recibida ha sido verdaderamente enviada por quien dice que la envía.
c. INTEGRIDAD: Seguridad que la información llegue sin ninguna modificación hasta su destinatario.
d. NO REPUDIO: Seguridad que la información enviada ha llegado hasta su correcto destinatario por medio de un mensaje encriptado de confirmación de recepción.

2.- RECURSOS PARA LA SEGURIDAD DE LA INFORMACIÓN DIGITAL:

Además de la seguridad física del sistema (vigilancia, alarmas, instalaciones y medios de comunicación de difícil acceso físico, control de ingreso, etc.), es necesario contar con otros recursos de orden lógico si se quiere brindar una verdadera protección a la información digital. Para una conveniente ubicación conceptual del encriptamiento en el campo de la seguridad de la información, se expone en este capítulo los distintos recursos de orden lógico con los cuales se brinda esta protección.

Existen dos tipos de sistemas de seguridad para la información digital; los limitantes del acceso a la información ya sea mediante recursos físicos o lógicos, y aquéllos que impiden el acceso ocultando la información bajo un formato secreto imposible de ser descifrado por personas no autorizadas. De acuerdo a este criterio se pueden enumerar dos categorías en cuanto a recursos de seguridad:

- Firewalls: recurso lógico acotador del acceso a la información.
- Encriptamiento: recurso lógico que oculta la información cambiando su formato.

2.1 FIREWALLS.

Todo sistema de redes de computadoras (LAN O WAN)[1] necesita un enlace externo (con otras redes) para poder desarrollar sus labores en un forma veloz y efectiva, comunicando sus informaciones en una manera mucho más eficiente, y recolectando datos externos en una forma rápida y sin retardos en la introducción de la información. La única desventaja de este enlace externo radica en que se convierte en una puerta de entrada para personas o sistemas potencial y efectivamente dañinos a la red, debido a que pueden hacer pública la información confidencial, y más grave aun, alterar datos de la red causando daño a la organización.

[1] LAN: Local Area Network, WAN : Wide Area Network.

Para reducir este riesgo se utiliza un sistema de protección llamado Firewalls. Como su nombre lo indica[2], su misión es ser una barrera aislante, aunque se podría definir más correctamente como el componente de la red acotador del acceso entre una red, y otros conjuntos de redes [4].

Un firewalls es ante todo una aplicación, es decir un programa, el cual al mismo tiempo que crea lazos de comunicación entre los componentes de la red protegida, restringe estos lazos tomando como parámetros para la restricción la naturaleza de la información transferida por dichos lazos y la identidad de las personas que intercambian dicha información [4].

Un sistema firewalls puede residir en distintas partes del hardware de una red de computadoras, tales como [4]:

- Un servidor principal de la red (no es muy recomendable).
- Un enrutador ubicado entre los enlaces externos y la red interna.
- Uno o varios servidores en una subred de protección (red ubicada entre los enlaces con las redes externas y los enlaces de la red interna),
- Un anfitrión bastión (anfitrión ubicado entre la red interna y la internet, y que es el encargado de atender las peticiones de servicio y de archivos a todos los solicitantes de internet y a los clientes internos de la red).
- Un servidor proxi (servidor dedicado a tratar con servidores externos en nombre de clientes internos).

Un sistema de firewalls necesita protegerse a si mismo y a la información que contiene. Para lograr este objetivo de protección en una escala tan pequeña como lo son los bits (en lugar de niveles grandes como enlaces y dispositivos de hardware) , se utiliza una técnica distinta: el encriptamiento.

2.2 ENCRIPTAMIENTO.

Para poder estar conectado a la red, y gozar de todos sus servicios sin perder la privacidad, el encriptamiento de la información que transita por la red, y aún de la información utilizada para entablar los

[2] En la terminología de la construcción en los países de habla inglesa se entiende "firewalls" como una barrera incombustible y aislante que evita que el fuego se propague desde un ámbito a otro durante un incendio

enlaces, ofrece una alternativa segura la cual no resta mucha eficiencia a todos los servicios que se obtienen al estar conectado en red.

A los distintos métodos para ocultar la información a personas distintas de los legítimos remitente y destinatario en una transferencia de datos se le conoce como encriptamiento. Este ocultamiento de la información se realiza al cambiar su formato de tal manera que sea prácticamente ilegible para quien no conoce la forma de realizar dicho cambio [5].

Si se quiere llevar el encriptamiento un paso más adelante en cuanto a seguridad, no basta con hacer de la información una secuencia ilegible, sino también hacer que la secuencia de datos encriptados no ofrezca ninguna pista acerca de la información que oculta. Para lograr este efecto el texto encriptado debe ser en apariencia completamente aleatorio, sin patrones ni ciclos reconocibles que ofrezcan idea alguna (aun que sea muy vaga) del contenido del texto encriptado.

El encriptamiento protege la información a nivel de datos que transitan de sesión de usuario a sesión de usuario a través de tiempo o de espacio. Esto abarca la información transferida desde una máquina a otra, y la que no sale de la misma máquina, pero sí se transporta a través del tiempo, porque se desplaza a otra sesión de usuario, en un tiempo distinto, en la misma máquina, y probablemente con otra persona distinta.

Con el encriptamiento es posible crear una red privada virtual entre varios clientes de una red de área local o extensa, de manera que aun cuando los enlaces utilizados sean comunes a muchos más elementos en la red, la información que transita entre los elementos componentes del área virtual no pueda ser conocida ni alterada por elementos ajenos a la misma, al mismo tiempo que se puede evitar las intrusiones de información en ésta al permitir solamente la entrada a información y protocolos de comunicación encriptados [5].

La técnicas de encriptamiento no sólo son usadas para cambiar el formato de la información digital que se escribe bajo el código ASCII , su uso abarca toda la gama de comunicaciones digitales, como lo son el audio comercial, el vídeo, las imágenes digitalizadas, la modulación PCM [3] y modulación delta utilizada en los equipos telefónicos, etc.

[3] PCM : Pulse Code Modulation.

3.- HISTORIA DEL ENCRIPTAMIENTO:

Las técnicas de encriptamiento no persiguen un objetivo que haya sido planteado recientemente por la nueva tecnología informática, más bien se encaminan hacia las metas que desde el desarrollo del lenguaje escrito se ha tenido en toda transmisión de información confidencial: la privacidad de la información entre remitente y destinatario [7].

A continuación se listan algunos de estos intentos para brindar protección a la información de acuerdo a la evolución de los medios de comunicación a través de la historia:

1900 a.C.

En el antiguo Egipto se usaban jeroglíficos fuera del conjunto estándar para hacer escrituras de las cuales no se quería se supiera su contenido. Se puede considerar a esto como la forma más primitiva de encriptamiento [7].

50- 60 a.C.

El emperador Julio César mantiene control del imperio Romano por medio de informaciones encriptadas escritas utilizando sustitución de caracteres, intercambiando cada letra por la ubicada a cuatro letras de distancia hacia la derecha en la ordenación normal del alfabeto [8].

200 a.C. - 1800 d.C.

Cuando no existía otro medio de comunicación para distancias relativamente grandes que no fuera el medio escrito, la carta se protegía ya sea depositándola en un contenedor inviolable cuya llave la tenía el receptor, o bien escribiéndola en un tipo de clave secreta, para que aun cuando sea leída por una persona extraña no pudiera ser comprendida, en estos casos la clave para la decodificación la poseía el receptor.

En este período se utilizaron muchas técnicas, tales como la creación de alfabetos completos para encontrar una sustitución para cada letra, la sustición de fonemas por otros caracteres, números u otros

fonemas, o la sustitución de cada letra utizando varios alfabetos de sustitución en forma sucesiva para hacer un código casi indecifrable [8].

1840 -1910 :

Con el advenimiento de las comunicaciones eléctricas y electrónicas el problema siguió teniendo soluciones semejantes. Ya en la era del telégrafo, algunos telegrafistas utilizaban claves distintas a la morse adoptada universalmente, para poder transmitir informaciones secretas [7].

1917 :

Gilbert Vernam de AT&T crea un sistema de encriptamiento para teletipo, en el cual paralelamente a la trama de datos corría la trama de clave, la cual encriptaba los datos paralelos a cada elemento de la clave. Dos tramas de clave iguales hacían un sistema seguro entre dos teletipos. Con este importante avance se crea el sistema de encriptamiento de trama [6].

1919 :

Hugo A Koch crea en Holanda una máquina mecánica de encriptamiento que podía escribir un mensaje en papel, utilizando para ello transposición de caracteres tanto a lo largo de las columnas como de las filas de caracteres. La tecnología utilizada era rotores mecánicos que cambiaban la posición de cada caracter en el texto. Esta máquina fue comercializada en 1923 por Arthur Scherbies con el nombre de "Enigma Machine" [7].

1930 :

La armada de Estados Unidos crea "Sigaba", una máquina de transposición de caracteres basada en 15 rotores mecánicos que alteraban los caracteres de acuerdo al control ejercido por 5 rotores ajustables capaces de ser modificados para seguir una determinada clave de encriptamiento. Fue una máquina de relativa importancia para las comunicaciones durante la segunda guerra mundial [7].

1960 :

Dr. Horst Feistel crea "Lucifer Cipher", el cual encripta información digital en bloques de bits utilizando una clave secreta. Años mas tarde NSA (National Security Agency) lo haría evolucionar hasta convertirlo en el popular y oficial algoritmo DES (Data Encription Standar) [7].

1976:

Whitfield Diffie y Martin Hellman introducen la idea de la criptografía en clave pública al dar a conocer en su página web en UNIX el planteamiento del desarrollo de un proyecto de encriptamiento en el cual exista una negociación de la clave entre remitente y destinatario para crear una clave para desencriptar distinta de la clave para encriptar [7].

1977 :

Se crea el algoritmo DES (Data Encription Standard) según recomendaciones del gobierno de los Estados Unidos a través de la Agencia Nacional para la Seguridad (NSA : National Security Agency) [3].

1978 :

Tres investigadores; R. L. Rivest, A. Shamir y L. Adleman unieron sus esfuerzos para crear un algoritmo que se convertiría en el más popular y seguro para la transmisión de datos, el cual es conocido como RSA , y consiste en dos claves completamente distintas: una clave pública para encriptar el mensaje, y una clave privada para desencriptarlo. Con este algoritmo surgió un nuevo concepto en seguridad digital: el encriptamiento de clave pública [9].

1990:

Se crea IDEA (International Data Encription Algoritm) en Suiza, utilizando una clave de 128 bits para implementar un encriptamiento en clave secreta [7].

1997-1999:

En la actualidad los sistemas operativos y algunos Browser para internet llevan ya incluido dentro de sí programas de encriptamiento en los cuales pueden determinarse personalmente algunos parámetros del proceso; resultan relativamente seguros para aplicaciones esporádicas no demasiado importantes, debido a la poca posibilidad de que la clave sea encontrada en corto tiempo por un criptoanalista, además de no ofrecer un incentivo suficiente para darse a la tarea de hacerlo [1].

4.- CONCEPTOS FUNDAMENTALES DEL ENCRIPTAMIENTO:

Para la mejor comprensión del resto del documento se listan a continuación todos aquellos conceptos necesarios en el entendimiento del tema. Se exponen primero todos los términos que se involucran en el encriptamiento, y a continuación los algoritmos universales en la criptografía, su clasificación más general y algunos ejemplos que ayudan a comprender mejor cada una de las especies que este género abarca.

4.1- TERMINOS UTILIZADOS:

CRIPTOGRAFÍA (Cryptography) es el arte o la ciencia de mantener secreta la información mediante cambios en las unidades que componen el código utilizado para expresar dicha información [11].

CLAVE DE ENCRIPTAMIENTO: Es el parámetro mantenido fijo en cada sesión de comunicación o en cada usuario, el cual es usado por el algoritmo de encriptamiento para ser operado en combinación con el texto fuente para crear el texto cifrado. Se denota como: K

ANCHO DE CLAVE: (keyspace) Es la longitud de la clave de encriptamiento, ya se trate de una secuencia de bits o una secuencia de caracteres [8].

TEXTO FUENTE O MENSAJE (Cleartext or plaintext): Es la información que se desea encriptar escrita en un formato legible y estandarizado [10]. Se denota como: P o M .

ENCRIPTAMIENTO (Encryption or Cipher): Es el procedimiento mediante el cual el formato de la información es alterado de manera tal que no pueda ser comprendido por personas o sistemas ajenos a su remitente y destinatario [10]. Se denota esta operación como: $E(M)$, $E(P)$, $E_k(M)$, $E_k(P)$

TEXTO CIFRADO O TEXTO ENCRIPTADO (Ciphertext): es la información encriptada a la cual se le ha cambiado su formato en sus palabras, fonemas, caracteres o bits [11]. Se denota como: C , $C=E_k(M)$, $C=E_k(P)$, $C=E(M)$, $C=E(P)$

DESENCRIPTAMIENTO O DESCIFRAMIENTO (Decryption or decipher): Es el procedimiento mediante el cual el formato de la información encriptada es devuelto a su formato original [10]. Se denota esta operación como: $D(C)$, $D_k(C)$

CRIPTOANÁLISIS (Cryptanalysis) es el arte de descifrar el texto cifrado sin el conocimiento completo de la clave y del algoritmo de encriptamiento [6].

CRIPTOLOGÍA (Cryptology) es la rama de las matemáticas en la cual se funda el encriptamiento, brindando funciones y procedimientos matemáticos para desarrollar el algoritmo de encriptamiento [8].

ESTEGANOGRAFÍA (Steganography) es la acción de ocultar un mensaje dentro de otro mensaje sin cambiar su formato de presentación, por ejemplo: el uso de palabras claves y pseudónimos, caracteres diferenciados, dibujos claves, etc [8].

ENCRIPTAMIENTO EN CLAVE SECRETA Son los sistemas de encriptamiento en los cuales una única clave es utilizada para encriptar y desencriptar. Esta clave es guardada en secreto por sus usuarios [9].

ENCRIPTAMIENTO EN CLAVE PÚBLICA Son los sistemas en los cuales la clave para el encriptamiento es completamente distinta a la clave para el desencriptamiento, de tal manera que la clave para el desencriptamiento da muy pocas o casi nulas posibilidades para encontrar la manera de desencriptar la información encriptada. Según sea la utilidad que quiera darse (confidencialidad o autenticación de los mensajes), así se va a repartir una clave libremente a las personas con quienes se va a establecer comunicación encriptada. La clave repartida a las distintas personas se conoce como clave pública, y la otra conservada en secreto, se conoce como clave privada.

Si lo que se desea es confidencialidad en la información por recibir, entonces será conveniente repartir la clave de encriptamiento como clave pública, y dejar la clave para el desencriptamiento como clave privada.

Si lo que se desea es autenticación en la información por enviar, entonces será conveniente repartir la clave de desencriptamiento como clave pública, y dejar la clave para el encriptamiento como clave privada [3].

FUNCIONES DE UN SOLO SENTIDO (ONE-WAY FUNCTIONS): Son funciones matemáticas las cuales al ser aplicadas a los bits que representan una información digital, producen como salida un valor binario de extensión específica. Son de un sólo sentido porque el valor producido por ellas no dá ninguna información acerca del bloque de bits que la originó, y de esta manera la función no es reversible como para reconstruir el bloque origen. El ejemplo más sencillo de este tipo de funciones es la paridad de un bloque de bits.
Se puede identificar tres tipos de funciones de un sólo sentido [8] :

- FUNCIONES DE TOTALIZACIÓN (CHEKSUM): Son aquellas que producen un valor numérico expresando la totalidad de determinado parámetro en el bloque origen. Un ejemplo de esto puede ser la paridad de todos los bits, el total de caracteres en el bloque, el valor numérico de la suma de todos los códigos ASCII, etc.
- FUNCIONES HASH DE UN SOLO SENTIDO (ONE WAY HASH FUNCTIONS): Son funciones matemáticas o de otra naturaleza, que expresan en un número binario de longitud específica, una peculiaridad del bloque de bits origen, y que difícilmente puede ser obtenidos de otro bloque de bits. Al bloque de bits origen se le conoce como pre-imagen, y al producido como resultado se le conoce como valor Hash. Un ejemplo de este tipo de función es el valor de la operación en función EXOR en forma consecutiva de bloques de bits en el mensaje, reflejado en un bloque de igual longitud, otro ejemplo puede ser un bloque que exprese el total de caracteres seguido por el promedio de sus valores ASCII, etc. Este tipo de funciones generalmente se denota por :

  $$H \text{ (bloques)}$$

- CODIGO DE AUTENTICACIÓN DE MENSAJES (MAC: MESSAGE AUTENTICATION CODE): Es un tipo de función hash el cual devuelve un valor hash de uno o varios bloques consecutivos de información operándolos con una clave secreta, de manera que sólo los poseedores de esa clave secreta pueden desarrollar este valor hash en los bloques de información que han sido operados. Los mensajes auténticos producen el mismo valor Hash que el emisor original del mensaje ha mandado para autenticarlo, valor que sólo el genuino receptor puede desarrollar al operar el mensaje con la clave secreta. Este código se denota por :

  $$H_k \text{ (bloques)}$$

REGISTRO DE INSTANTE (TIMESTAMP): bloque de datos adosado e íntimamente relacionado con el mensaje mediante el desarrollo de un valor de función hash de un solo sentido que se aplica al texto, el cual expresa el instante en que dicho mensaje fue creado o enviado. Debido a que la computadora origen del mensaje puede perder el calendario y la hora de la red, y/o ser alterada en cuanto a su fecha y hora, generalmente este valor es adosado por un programa confiable alojado en un centro de servicios de autenticación [8]. Generalmente se denota simplemente como: T

FIRMAS DIGITALES (DIGITAL SIGNATURE) Bloque de datos adosado e íntimamente relacionado con el mensaje mediante el encriptamiento en clave privada.

En algunas aplicaciones este bloque es el valor de función hash de un solo sentido que se desarrolla a partir del texto autenticado y luego se encripta en clave privada. En este caso lo que sirve de verificación es el valor hash que se obtiene del texto.

Se puede verificar el origen del mensaje porque sólo la persona firmante del mensaje es la única poseedora de la clave privada para el encriptamiento, y además se puede verificar la integridad del mensaje porque el valor de la función hash de un sólo sentido adosado con el mensaje debe coincidir con el valor obtenido al aplicar la función al mensaje autenticado [6].

Muchos algoritmos de firmas digitales son una aplicación específica de los algoritmos de clave pública. La información autenticada puede o no ser información encriptada, pero el bloque adosado a dicha información va a ser siempre un encriptamiento de la información en clave privada especifica (clave especifica del emisor) [8].

Autenticar un mensaje con una firma digital de este tipo se denota por:

$$S_k\ (M)$$

Y verificar un mensaje con una firma digital de este tipo se denota por:

$$V_k\ (M)$$

De manera que el bloque adosado puede ser verificado al operar:

$$V_k\ (S_k(M)) = M$$

ENTIDAD ARBITRADORA (ARBITRATOR): Participante confiable del proceso de comunicación entre dos o más unidades de la red, el cual sirve como mediador y certificador en una comunicación segura. Este mediador en la comunicación debe poseer altos estándares de seguridad en sus bases de datos y en sus enlaces con cada uno de los huéspedes solicitantes de sus servicios, ya que sus certificaciones son tomadas

como garantía de seguridad en los enlaces y en la información otorgada a sus clientes. Dentro de sus funciones se puede enumerar:

- Identificación de cada participante y certificación de su participación en la sesión de comunicación.
- Asignación de claves secretas para una sesión de comunicación específica.
- Certificación de origen y destino de las claves o mensajes intercambiados en una sesión de comunicación.
- Asignación de claves públicas ya sea para encriptamiento o desencriptamiento, con la segura correspondencia de una clave privada que también es asignada al que la solicita.
- Asignación de firmas digitales a sus clientes, certificadas por su servicio.
- Certificación de registros de instante (timestamp) a los clientes que solicitan este servicio.

Generalmente las entidades arbitradoras son programas alojados en las máquinas de servicios en linea (internet) que la casa fabricante de los sistemas de seguridad proporciona a sus clientes [8].

NEGOCIACIÓN DE CLAVES (KEY EXCHANGE): Procedimiento mediante el cual dos o mas participantes en una sesión de comunicación intercambian una clave secreta, o sus claves públicas, o la clave de su firma digital, de una manera segura. Este tipo de acuerdos en cuanto al uso de una clave específica, son llevados a cabo mediante protocolos de seguridad que pueden o no involucrar a una entidad arbitradora [8].

4.2 ALGORITMOS DE ENCRIPTAMIENTO.

Para realizar el encriptamiento es imprescindible seguir una secuencia ordenada de transformaciones matemáticas a la representación digital de la información. Este proceso es conocido como algoritmo de encriptamiento. El proceso inverso es conocido como algoritmo de desencriptamiento.

Históricamente se pueden clasificar los algoritmos como:

- ALGORITMOS SECRETOS O RESTRINGIDOS: Son aquellos procedimientos de encriptamiento conocidos solamente por sus legítimos usuarios, y muchas veces ni siquiera por ellos mismos, sino solamente por su fabricante [8].
- ALGORITMOS NO SECRETOS DE ENCRIPTAMIENTO: Son aquellas funciones matemáticas utilizadas para encriptar y desencriptar mensajes, las cuales son conocidas en el dominio público.

Su fortaleza como medio de seguridad radica en su clave de encriptamiento, la cual sí es secreta, y además única para cada transferencia de datos, para cada usuario, o para cada grupo de usuarios [6].

En cuanto a las unidades transformadas en el encriptamiento se pueden clasificar los algoritmos como:

- ALGORITMOS PARA ENCRIPTAMIENTO DE CARACTERES.
- ALGORITMOS PARA ENCRIPTAMIENTO DE BITS.

Este tipo de algoritmos pueden ser comprendidos mejor en los siguientes sub-apartados.

4.2.1 ALGORITMOS PARA ENCRIPTAMIENTO DE CARACTERES:

Son aquellos que, al encriptar un mensaje, transforman el texto fuente mediante cambios en unidades de caracteres. De acuerdo a la forma como son transformados los mensajes, se pueden clasificar en dos grupos:

- Algoritmos cifradores por sustitución: los que alteran el texto sustituyendo un caracter por otro distinto.
- Algoritmos cifradores por transposición: los que alteran el texto cambiando de posición cada caracter por separado, ya sea dentro de la fila o de la columna en que está ubicado.

Estos tipos de algoritmos fueron muy utilizados antes del advenimiento de los sistemas de información digital electrónica. En la actualidad no son aprovechados como base de un sistema de encriptamiento, sin embargo pueden ser empleados para incrementar el grado de complejidad de la información encriptada por medios digitales binarios [8].

4.2.1.1 ALGORITMOS CIFRADORES POR SUSTITUCIÓN:

Son aquellos que alteran el texto sustituyendo uno o más caracteres por otro distinto. Los más usados pueden ser clasificados en las siguientes categorías [20]:

- ALGORITMOS CON SUSTITUCIÓN SIMPLE O MONOALFABÉTICA: Son aquellos en los cuales cada caracter del texto fuente tiene su único correspondiente caracter en el texto cifrado, el cual es asignado de acuerdo a una tabla de sustitución o alfabeto sustituto.[4]

[4]Un ejemplo de este tipo de tipo de función se tiene en el popular juego infantil de la clave murciélago, la clave semáforo, o las claves de letras con las manos.

- ALGORITMOS CON SUSTITUCIÓN POLIALFABÉTICA : En forma semejante a los anteriores suplantan un caracter por otro, pero con la particularidad de que se reemplazan caracteres del texto fuente de acuerdo a varias tablas de sustitución o alfabetos sustitutos. Cada tabla de sustitución es utilizada en forma sucesiva para reemplazar un caracter o un bloque de caracteres, y cuando todas las tablas han sido empleadas, se repite la sucesión en forma cíclica hasta terminar la sustitución de todos los caracteres del texto fuente.[5]
- ALGORITMOS CON SUSTITUCIÓN POLIEQUIVALENTE: Son aquellos en los cuales cada caracter del texto fuente tiene más de un valor sustituto correspondiente para el texto cifrado. Un ejemplo de este tipo de sustitución son las palabras asociadas con los números telefónicos, donde cada número (caracter del texto fuente) tiene varias letras asociadas con si mismo(caracteres del texto cifrado), de manera tal que se pueden dar números telefónicos encriptados en forma de palabras.
- ALGORITMOS CON SUSTITUCIÓN POLIGRÁMICA: Son aquellos en los cuales bloques de caracteres específicos en el texto fuente son sustituidos por otros bloques de caracteres específicos en el texto cifrado. Estos bloques de caracteres pueden ser sílabas o fonemas del lenguaje del texto fuente. Un ejemplo de este tipo de sustitución son las transliteraciones de las palabras que no tienen equivalente en otros idiomas (un caso especifico son las transliteraciones de los nombres propios escritos en un lenguaje extranjero tales como las representaciones de estas palabras españolas al chino o viceversa).

4.2.1.2 ALGORITMOS CIFRADORES POR TRANSPOSICIÓN:

Son aquellos en los cuales el texto es alterado cambiando de posición cada caracter por separado, ya sea dentro de la fila o de la columna en la cual está ubicado. También pueden realizar un encriptamiento por transposición de los caracteres, de filas a columnas, y de columnas a filas mediante la transposición de una matriz contenedora del mensaje [8]. Vease el siguiente ejemplo para comprender mejor la forma de llevar a cabo este tipo de encriptamiento:

Se tiene el siguiente mensaje como texto fuente a encriptar:

este es un ejemplo de encriptamiento por transposición de caracteres.

[5] Un ejemplo de este tipo de encriptamiento podría ser utilizar la clave murciélago para encriptar las líneas impares, y la clave del emperador Cesar Augusto para encriptar las líneas pares en un documento de varios párrafos. El encriptamiento podría ser aun más desconcertante si las claves se alternaran para cada palabra.

Se tiene la matriz de texto fuente:

e	s	t	e		e	s		u	n		e	j	e	m	p	l	o		d
e		e	n	c	r	i	p	t	a	m	i	e	n	t	o		p	o	r
	t	r	a	s	p	o	s	i	c	i	o	n		d	e		c	a	r
a	c	t	e	r	e	s	.												

Se obtiene el mensaje encriptado leyendo columnas de arriba hacia abajo:

[ee a s tc tert enae csr erpe sios ps. uti nac mi eio jen en mtd poe l opc oa drr]

Este tipo de encriptamiento fue muy usado a principios de este siglo, editado en máquinas de escribir mecánicas que utilizaban rotores para hacer esta transposición, las cuales no necesariamente imprimían las mismas letras tecleadas, sino que también hacían a la vez una sustitución de caracteres. El desciframiento del mensaje se llevaba a cabo escribiendo nuevamente el mensaje encriptado en la máquina, esta vez configurada como descifrador de mensajes [8].

4.2.2 ALGORITMOS PARA ENCRIPTAMIENTO DE BITS O BYTES:

Este género de algoritmos no discriminan grupos caracteres específicos, sino que operan el texto fuente en formato digital tomando el archivo donde éste se encuentra para encriptarlo completamente, incluyendo toda aquella información que acompaña al mensaje, tal como el registro del tipo de letra, márgenes, programa que lo desarrolla, ect. Este tipo de algoritmos no sólo pueden encriptar textos escritos en el alfabeto occidental, sino que pueden ser empleados todo tipo de información expresada en forma digital, tales como el sonido, imágenes digitalizadas, vídeo digital, enlaces telefónicos, etc. [10].

De acuerdo al manejo que hacen los algoritmos de la clave de encriptamiento, estos pueden ser clasificados en dos categorías [8]:

a. ALGORITMOS SIMÉTRICOS O DE CLAVE SECRETA: Los que usan una clave común para el encriptamiento y el desencriptamiento. Esta clave es mantenida en secreto por los usuarios de la misma.
b. ALGORITMOS ASIMÉTRICOS O DE CLAVE PUBLICA: Los que utilizan claves distintas para el encriptamiento y para el desencriptamiento, siendo la clave para el encriptamiento completamente inútil para descifrar la información encriptada por ella misma.

Las anteriores categorías no serán explicadas en detalle en este capítulo, pero serán ampliamente discutidas en capítulos posteriores.

4.3 ENCRIPTAMIENTO POR FUNCIÓN EXOR:

Cuando la información a encriptar está dada en un formato digital binario, la operación en función OR exclusiva (EXOR) de la representación del mensaje con un patrón específico puede constituir en si mismo un tipo de encriptamiento. Para desarrollar esto, los bits que representan el mensaje encriptado son operados como un término dentro de una función EXOR , el otro término lo constituye la clave de encriptamiento, El resultado de la función es el mensaje encriptado.

Una propiedad específica de la función EXOR es la de retornar el valor de uno de sus términos al ser operado el resultado con el término complementario, esto es:

$$A \oplus B = C$$

$$C \oplus B = A$$

Esto permite que el mensaje encriptado pueda ser desencriptado al ser operado con el término de clave en una función EXOR. En cuanto a la forma de intervenir el mensaje, esto se puede realizar de una manera continua bit a bit, o puede ser operado por partes de igual longitud de bits que la longitud de la clave [5].

ENCRIPTAMIENTO CON CLAVES DE USO ÚNICO (ONE TIME PAD)

Como pudo observarse en el procedimiento para operar datos con la función EXOR, es necesario tener una serie de datos de clave para encriptar en función EXOR con el mensaje.

Esta serie de datos puede ser una sola lista de bits para operar en EXOR con cualquier mensaje. Sin embargo esto brindaría muy poca seguridad ; con un sólo mensaje que se intercepte en su texto fuente y en su texto cifrado, la serie de datos quedaría al descubierto (Recuérdese que $K = C \oplus P$). Usar una serie de bits única para encriptar cada mensaje, sin repetirla nunca sería la alternativa ideal, ya que si alguna vez se interceptara un mensaje (texto fuente y cifrado), la clave encontrada sería inútil para desencriptar otro texto más.

La clave de uso único no sólo puede ser operada a nivel de bits con compuertas digitales, también puede ser implementada a nivel de caracteres. Si se asigna a cada letra del alfabeto hispano un valor numérico en orden correlativo (1 al 27), y se asocia también en doble equivalencia el mismo valor más 27, y en triple equivalencia el mismo valor mas 2*27, y así sucesivamente, se construiría la siguiente tabla de conversión de caracteres:

A	=	1	=	28	=	55	=	...
B	=	2	=	29	=	56	=	...
:		:		:		:		
Z	=	27	=	54	=	71	=	...

* [6]

Si se traduce cada mensaje a números, y cada número de cada caracter es sumado con otro número de un caracter clave (una clave que sólo se usa una vez), se obtiene un mensaje encriptado con números, el cual a su vez descifrado en letras con los valores de la tabla de conversión con períodos de 27 dá un mensaje encriptado en caracteres.

Para desencriptar el mensaje bastará con convertir el mensaje a su equivalente en números de la tabla (con la segunda o tercera asignación numérica), y restarle el valor de los números de los caracteres de la clave, y convertir cada valor a su correspondiente asignación literal [7].

Vease como ejemplo el encriptamiento de la palabra "ENCRIPTAMIENTO" , la cual será operada sumándole la clave KTAEMIMTYLGRB (una clave que a todas luces es una secuencia aleatoria de caracteres y la cual será utilizada sólo una vez), el procedimiento, tanto en cuanto a caracteres fuente, clave y resultado como en cuanto a valores numéricos asociados a manejar se muestra a continuación:

T. fuente	E	N	C	R	I	P	T	A	M	I	E	N	T	O
	5	14	3	19	9	17	21	1	13	4	5	14	21	16
Clave	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	K	T	A	E	M	N	I	M	T	Y	L	G	R	B
	11	21	1	5	13	14	9	13	21	26	12	7	19	2
Encriptado	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
	O	H	D	W	U	D	C	N	G	C	P	T	M	Q
	17	35	4	24	22	31	30	14	34	30	17	21	40	18

La palabra encriptada obtenida mediante esta técnica es "OHDWUDCNGCPTMQ" , la cual como puede comprobarse por simple inspección, no tiene ningún nexo con la palabra origen: "ENCRIPTAMIENTO".

[6] Esta forma de definir períodos con valores equivalentes se conoce como función mod (27) vease el capítulo 6 para mas detalles.

[7] Esta tecnología de encriptamiento fue creada por G. Vernam de AT&T laboratories en 1917, originalmente para encriptar mensajes en teletipos [8].

5.- ALGORITMOS SIMÉTRICOS O DE CLAVE SECRETA:

Existe gran diversidad de algoritmos de clave secreta si les analiza en sus propiedades de manejo de información, tales como: la cantidad de texto que toma en su ciclo de encriptamiento, la forma como maneja la información para transformarla (con o sin retroalimentación de lo anteriormente desarrollado), la forma como se desarrolla la clave (si cambia o no durante el proceso, y cómo se produce este cambio), y la forma en que la clave de encriptamiento se interrelaciona con el texto fuente para producir el texto encriptado.

La observación detenida tomando como referencia todos los parámetros anteriores despliega toda la clasificación que se expone en este apartado.

Como ya se explicó anteriormente, los algoritmos simétricos o de clave secreta son aquellos que usan una clave común para el encriptamiento y el desencriptamiento. Este parámetro es mantenido en secreto por los usuarios del mismo. El procedimiento para el encriptamiento es completamente reversible, siguiendo sus pasos en sentido inverso, puede obtenerse un algoritmo de desencriptamiento. Sin embargo la mayoría de los productos comerciales utilizan el mismo algoritmo para encriptar y desencriptar, sin revertir el orden de la secuencia de operación.

De acuerdo a la forma como manejan la información para desarrollar su encriptamiento, los algoritmos de clave secreta pueden ser clasificados en dos categorías [6]:

- ALGORITMOS CIFRADORES DE BLOQUE: Aquellos que toman la información en bloques de igual tamaño, uno a la vez, para aplicar el encriptamiento a cada bloque por separado.
- ALGORITMOS CIFRADORES DE TRAMA: Aquellos que encriptan la información toda de una sola vez en el momento en el cual ésta se traslada desde un dispositivo de memoria a otro tratando de manera individual a cada unidad que se traslada, ya sea bit a bit, o byte a byte.

5.1 ALGORITMOS CIFRADORES DE BLOQUE

5.1.1 MODOS DE OPERACION DE LOS ALGORITMOS CIFRADORES DE BLOQUE:

De acuerdo a la forma en que los bloques del texto fuente son encriptados, los algoritmos cifradores

de bloque pueden ser clasificados en los siguientes modos de operación que se explican a continuación.[8]

a). MODO TABLA ELECTRONICA DE CODIGOS (ELECTRONIC CODE BOOK: ECB)

Este modo de operación trabaja de la manera más obvia; cada bloque es encriptado de la misma manera que todos los demás bloques, usando la misma clave y los mismos parámetros en cada bloque [6], tal como lo muestra la figura 5.1 :

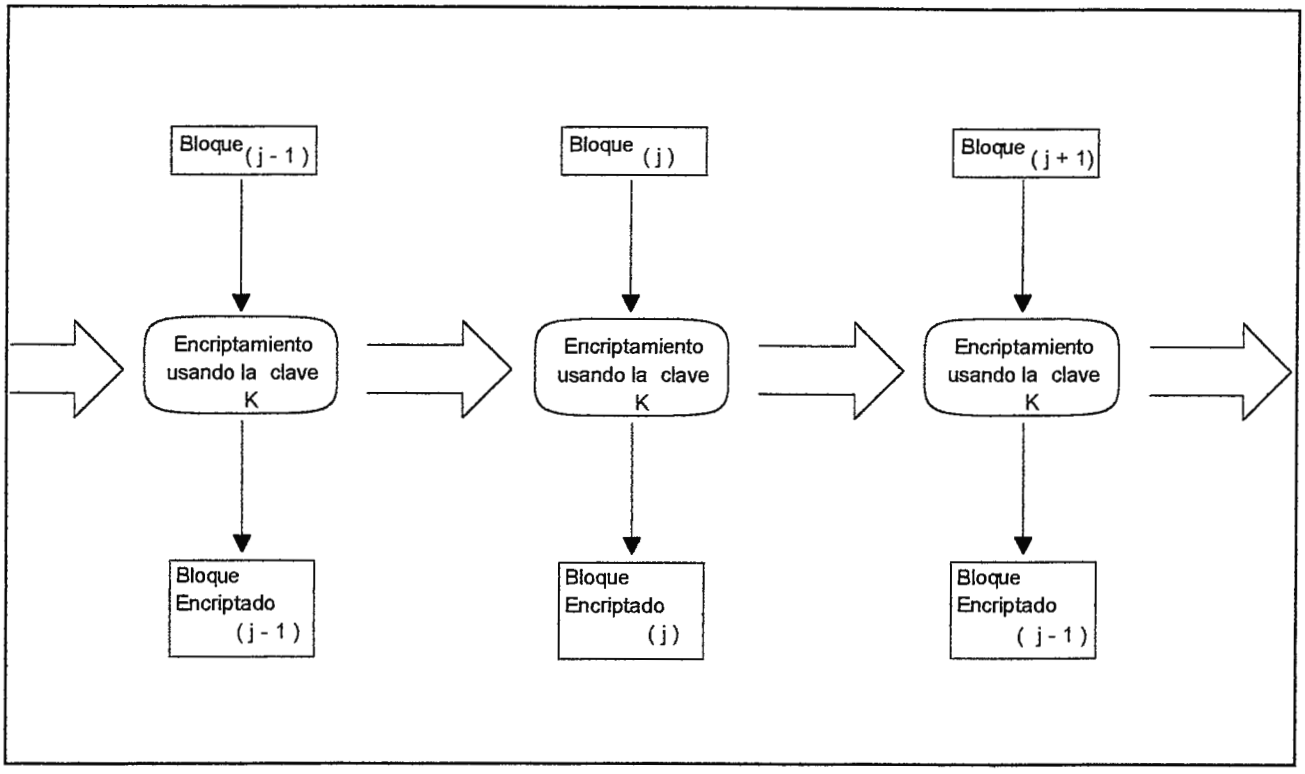


FIGURA 5.1

b) MODO ENCADENAMIENTO DE BLOQUES CIFRADOS (CIPHER BLOCK CHAINING:CBC)

En este modo de operación los parámetros para el encriptamiento de cada bloque van cambiando de bloque en bloque. Para desarrollar este proceso evolutivo, cada bloque de texto fuente es operado en función

[8]Esto no implica que no existan otras maneras distintas de operar bloques, hay infinitas maneras de hacerlo, pero las mostradas en este capítulo son las más comunes, y casi cualquier otra es simplemente un desarrollo posterior de una de ellas.

EXOR con el texto encriptado del bloque anterior, para luego ser este resultado cifrado con la clave y los parámetros de encriptamiento utilizados. Matemáticamente se puede expresar de la siguiente manera [8]:

$$C_i = E_k (P_i \oplus C_{i-1})$$
$$P_i = C_{i-1} \oplus D_k (C_i)$$

Recuerdese que C representa el bloque cifrado, y P representa el bloque de texto fuente base para producir C. Esto se explica con mayor claridad en la figura 5.2:

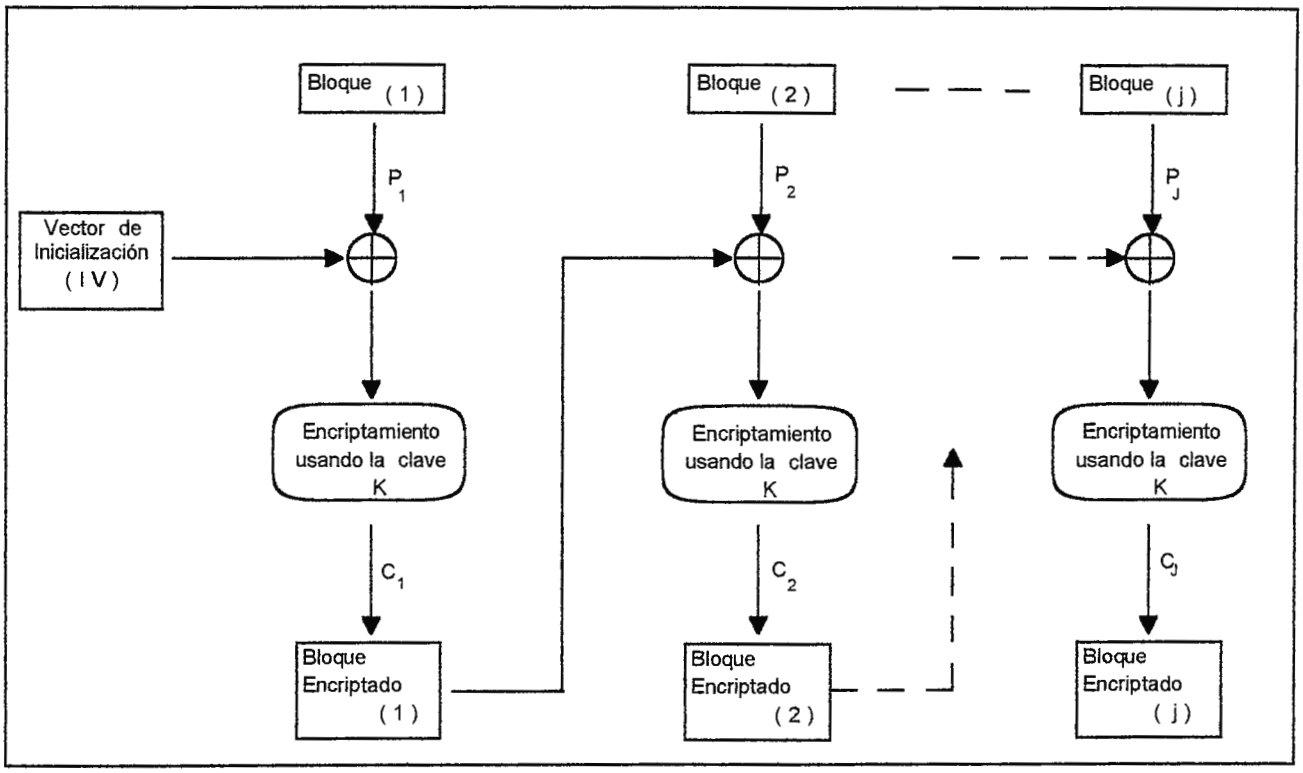


FIGURA 5.2

Como se puede apreciar, es necesario tener un primer bloque encriptado para iniciar el proceso, este primer parámetro de encriptamiento a definir es conocido como vector de inicialización (I V).

Todos los parámetros tanto los bloques de texto como los vectores de inicialización deben ser de la misma longitud del que opera la clave de encriptamiento. Si el último bloque de texto no tiene la longitud requerida, este va a ser rellenado con bits específicos que puedan ser comprendidos por el programa de desencriptamiento.

c) MODO RETROALIMENTACION DEL BLOQUE CIFRADO (CIPHER FEEDBACK (CFB) OR CIPHER AUTO KEY)

En este modo de operación los bloques del texto fuente son operados en función EXOR con el bloque encriptado del texto cifrado anterior. Nuevamente es necesario incluir un vector de inicialización para comenzar el proceso. Matemáticamente puede ser expresado este procedimiento como [8]:

$$C_i = P_i \oplus E_k(C_{i-1})$$
$$P_i = C_i \oplus E_k(C_{i-1})$$

Todo esto puede ser mejor comprendido observando con atención la figura 5.3:

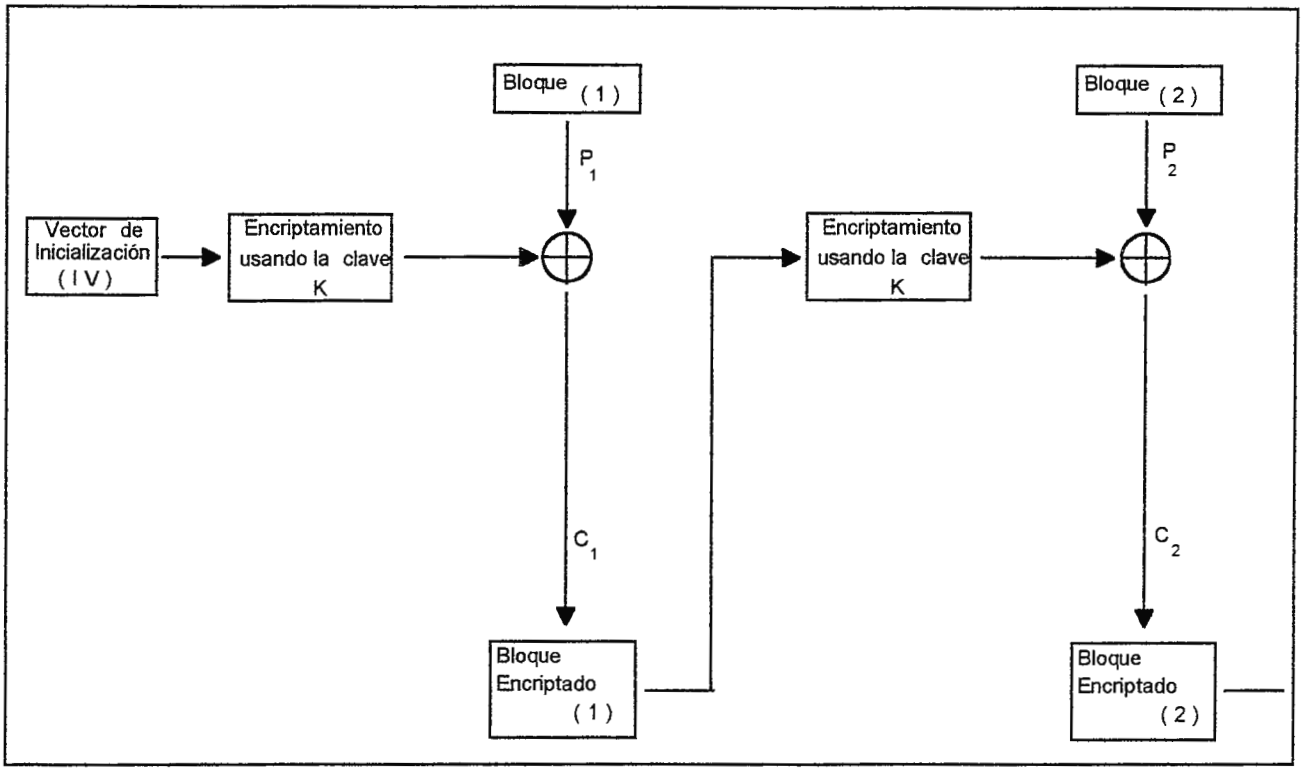


FIGURA 5.3

Como puede apreciarse, el texto cifrado puede ser de la misma longitud del texto fuente, pues aun cuando sea operado en función EXOR con otro término de mayor longitud, éste puede ser recortado en sus bits más significativos. Esto significa que el tamaño del bloque de texto (texto fuente y cifrado) es independiente del tamaño de la clave de encriptamiento. De esta forma el bloque de texto puede tener una longitud que puede oscilar desde 1 bit hasta el tamaño de la clave de encriptamiento.

d) MODO RETROALIMENTACION DE SALIDA DE CLAVE (OUTPUT FEEDBACK (OFB) O AUTOKEY.)

Este modo de operación, al igual que el anterior, encripta los bloques operándolos en función EXOR con un parámetro el cual evoluciona de bloque en bloque a lo largo del mensaje, pero con la diferencia de que el parámetro evoluciona por si mismo con total independencia de el texto fuente y el texto cifrado, en una secuencia en la cual cada término de clave es obtenido con un encriptamiento del bloque de clave anterior. Las siguientes ecuaciones y la figura 5.4 dan una mejor idea del funcionamiento de este modo de operación [8]:

$$C_i = P_i \oplus S_i \qquad S_i = E_k(S_{i-1})$$

$$P_i = C_i \oplus S_i \qquad S_i = E_k(S_{i-1})$$

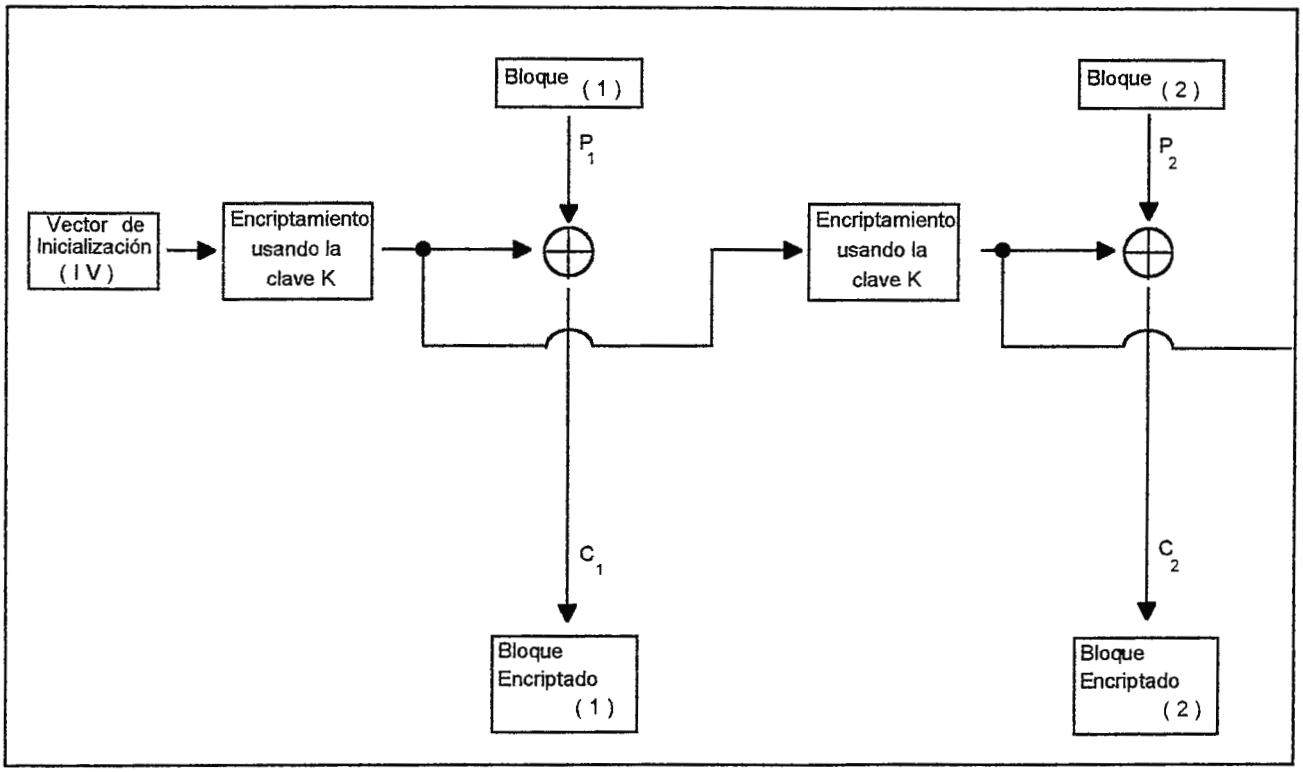


FIGURA 5.4

Al igual que el modo Cipher-Feedback, este modo de operación puede cifrar bloques desde 1 bit hasta el tamaño de la clave (operando en EXOR sólo los bits menos significativos hasta el ancho de bloque).

Una particularidad de este método radica en que el elemento de encriptamiento depende únicamente del vector de inicialización, el proceso evolutivo de los términos de la secuencia de encriptamiento puede ser desarrollado en otro momento distinto al de la operación en función EXOR. Esto permite tener series elaboradas de encriptamiento que pueden ser identificadas únicamente con el vector de inicialización. Convirtiendo el modo Cipher-Feedback (cuando se hace uso de esta ventaja) en el modo de encriptamiento más rápido (sólo desarrolla función EXOR de bloques).

e) MODO CONTADOR

En este modo de operación, los bloques de texto son encriptados operándolos en función EXOR con una serie de términos que evolucionan en forma secuencial matemática, progresiva, regresiva, aritmética, geométrica, exponencial etc. Sea cual fuere la forma de evolucionar de la secuencia, todas tienen como punto de inicio el vector de inicialización (I V) del algoritmo. Esto se puede ilustrar con mayor claridad con ayuda de la figura 5.5[8]:

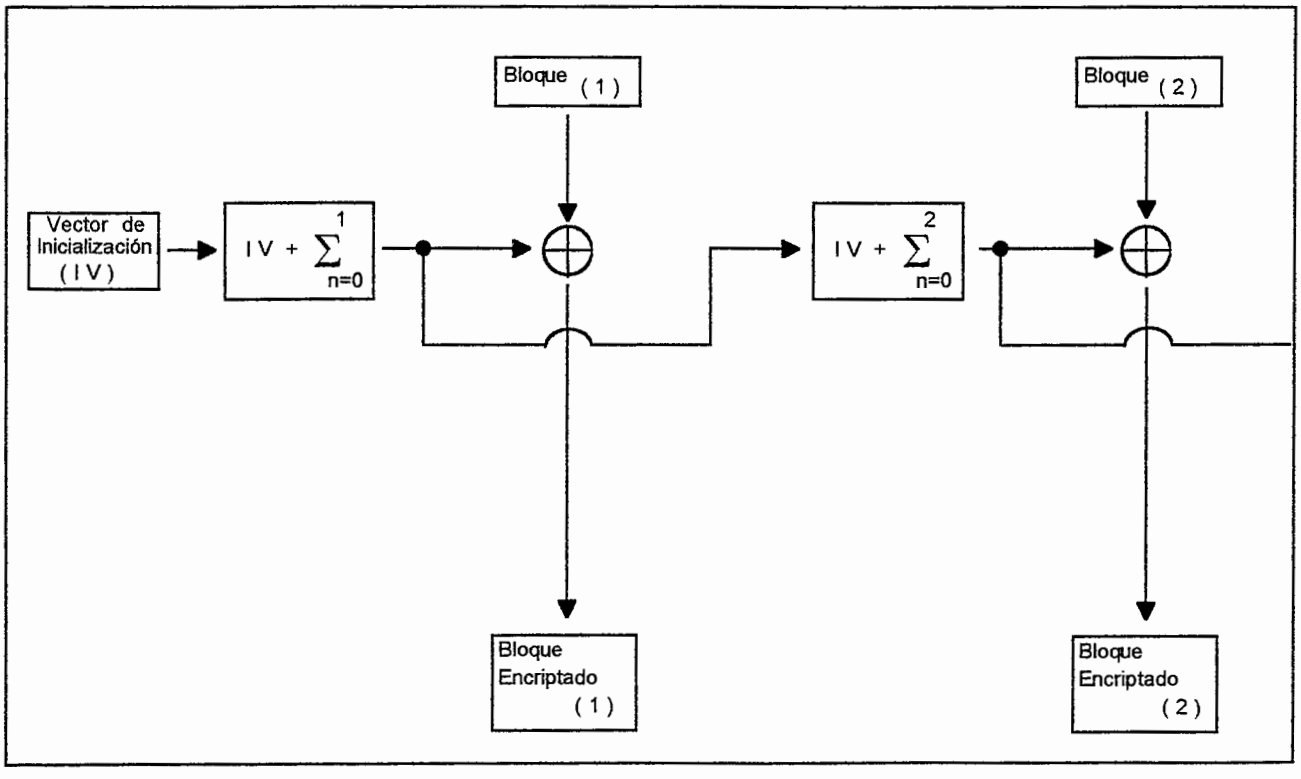


FIGURA 5.5

5.1.2 ALGORITMOS CIFRADORES DE BLOQUE CON FUNCIONES HASH

Como se explicó anteriormente, las funciones hash de un sólo sentido pueden producir un texto encriptado a partir de un texto fuente, pero no es posible revertir lo que producen para recuperar el texto fuente.

Sin embargo es posible encriptar y desencriptar un mensaje con funciones Hash utilizando los modos de encriptamiento de bloques Output Feedback y Cipher feedback, ya que en ellos el mensaje es simplemente operado en función EXOR con la evolución del elemento de clave. Obsérvese que en estos modos de encriptamiento la sucesión de bloques de texto fuente corresponde con una sucesión de bloques de clave, y en la operación EXOR de su mutua correspondencia producen una sucesión de bloques de texto cifrado. El desencriptamiento se produce operando en función EXOR la tercera y la segunda sucesión, lo cual producirá como resultado en forma íntegra la primera sucesión de bloques (vease la figura 5.6).

Obsérvese que en OFB y CFB la evolución del bloque de los bits de la clave es idéntica tanto para el encriptamiento como para el desencriptamiento. Si esta sucesión la dirige en su orden de cambio una función Hash H(m), se puede encriptar el mensaje con la sola aplicación sucesiva de la función, y aún en el mejor de los casos, utilizar una función MAC (Message Autentication Code) H (K, m) para regir el comportamiento de la función Hash con ayuda de una clave de encriptamiento K.

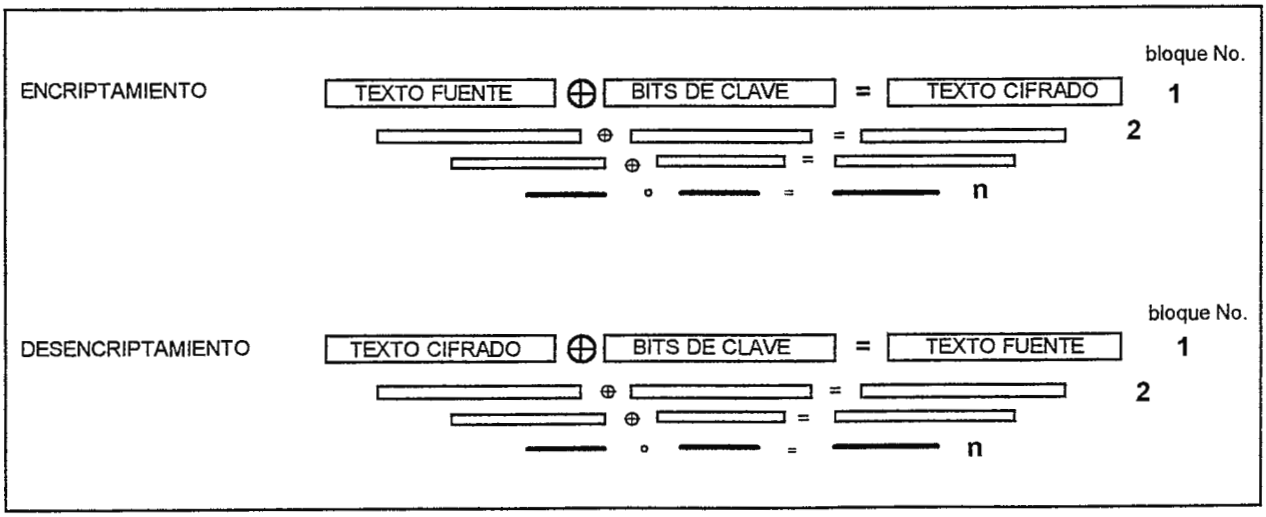


FIGURA 5.6

Utilizando el modo de encriptamiento CFB este encriptamiento puede ser expresado como:

$$C_i = P_i \oplus H_k(K, C_{i-1})$$

$$P_i = C_i \oplus H_k(K, C_{i-1})$$

Y utilizando el modo OFB este encriptamiento de bloques puede ser expresado como:

$$C_i = P_i \oplus S_i \qquad S_i = H_k(K, S_{i-1})$$

$$P_i = C_i \oplus S_i \qquad S_i = H_k(K, S_{i-1})$$

5.2 ALGORITMOS CIFRADORES DE TRAMA:

Son aquellos algoritmos que encriptan la información toda de una sola vez cuando ésta se traslada desde un dispositivo de memoria a otro, desarrollando el encriptamiento en el momento mismo de la transferencia de datos (aún cuando ésta sea en la misma máquina)[9].

Para comprender el funcionamiento de este tipo de algoritmos es necesario tener claro los siguientes conceptos:

TRAMA DE DATOS (DATASTREAM): Es la secuencia de datos (bits o bytes) que representan el texto fuente desde primer dato hasta el último [6]. Se denota cada elemento como P_i o M_i, y toda la trama como P o M.

TRAMA DE CLAVE (KEYSTREAM): es la secuencia de elementos de encriptamiento (bits o bytes) que servirán como parámetros para encriptar cada elemento de la trama de datos [6]. Cada elemento se denota como K_i , y la trama entera como K

TRAMA ENCRIPTADA: Es la secuencia de elementos encriptados que componen un mensaje completo encriptado [6]. Se denota cada elemento como C_i, y la trama entera como C.

Generalmente el encriptamiento se lleva a cabo mediante la operación en función EXOR de cada elemento de la trama de datos con cada elemento de la trama de clave (bit o bytes o la cantidad de bits que el microprocesador pueda operar en su puerto de datos) , a manera de crear una trama completa, que bit a bit es el resultado de la función EXOR de los elementos anteriores. Esto puede comprenderse mejor mediante las siguientes ecuaciones y observando la figura 5.7:

$$C_i = P_i \oplus K_i$$

$$P_i = C_i \oplus K_i$$

[9] El primero de estos algoritmos fue el de Gilbert Vernan con su encriptamiento para teletipos en 1917.

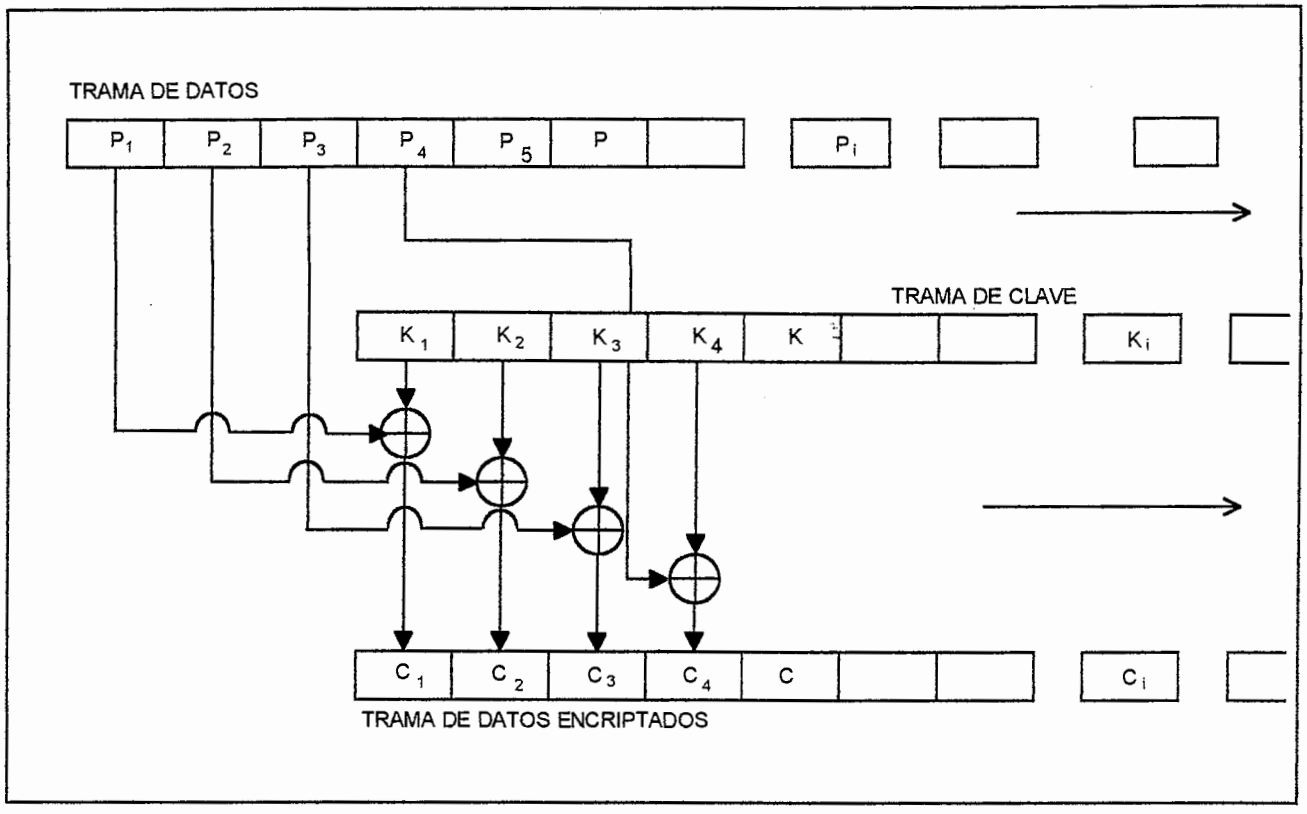


FIGURA 5.7

Como puede apreciarse en el diagrama , la trama de clave es el único elemento determinante en este tipo de encriptamiento. Su longitud debe ser lo suficiente larga como para poder encriptar un mensaje de varios Kilobytes de longitud como por ejemplo los archivos HTML grandes que circulan por internet.

Una opción para crear esta trama es por total aleatoriedad: se crea un set de tramas de claves de varios Kilobytes o Megabytes de longitud y se les reparte en CD a los participantes de la comunicación encriptada. Cada uno de ellos podrá encriptar mensajes y sólo dará al otro participante la identificación de la trama usada [6].

Sin embargo, este sencillo manejo de claves presenta un problema: alguna vez la trama se repetirá , y si un criptoanalista logra conseguir algunas de ellas (porque vé el texto cifrado y ha interceptado los documentos originales que van a ser encriptados, y los ha operado en función EXOR), la información habrá perdido su privacidad.

Otra opción la constituyen las claves de uso único (one time pad) , las cuales son absolutamente seguras, imposibles de ser descifradas de ninguna manera. El único problema con este tipo de clave es el manejo de la clave para hacerla llegar con absoluta confidencialidad hasta el emisor y el receptor antes de cada mensaje [6].

Existe una última opción para crear las tramas de clave: crear secuencias matemáticas que generen la clave en ambos lados del canal de comunicación.

Un generador de trama de claves puede visualizarse internamente en su funcionamiento como una función de salida que produce un valor a partir de un estado interno del generador. Este parámetro interno puede ser tomado de su mismo valor anterior, o de otro parámetro del sistema, tal como puede ser el mismo mensaje encriptado. Vease la figura 5.8 para tener una idea esquemática de un generador de tramas.

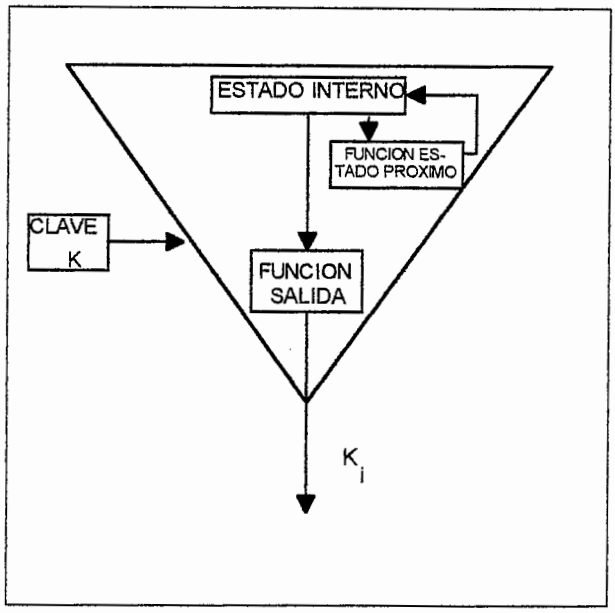


FIGURA 5.8

Esto arroja entre muchas, tres posibilidades que ya son conocidas por su aplicación en los modos de encriptamiento de los algoritmos cifradores de bloque [8]:

- Generar la trama de clave a partir de la "retroalimentación del bloque de salida de clave" (autokey).
- Generar la trama de clave a partir de la "retroalimentación del bloque cifrado" (cipher autokey).
- Generar la clave por cuenta progresiva (análogo al modo contador).

Para generar la trama de clave a partir de la retroalimentación del bloque de salida de la clave, se utiliza como estado interno, el mismo de la vez anterior (modificado por la clave K en la función estado próximo). Esto puede ser comprendido más fácilmente observando la figura 5.9 [8]:

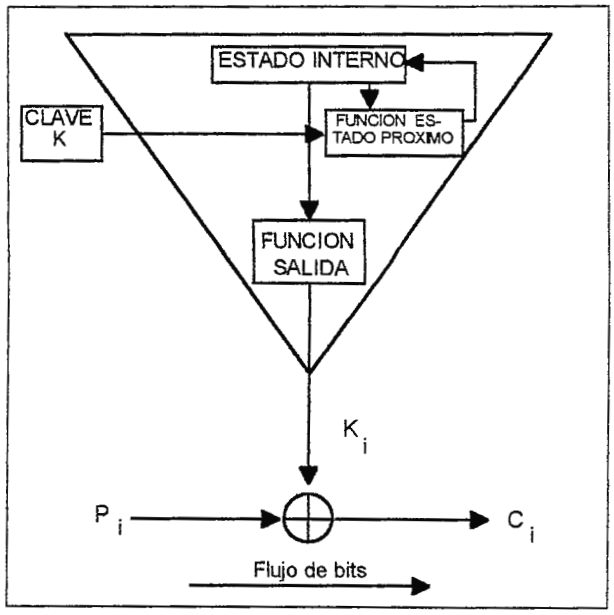


FIGURA 5.9

En forma similar, para generar una trama de clave a partir del bloque cifrado se toma como estado interno de la función de salida, el bloque anterior que se ha encriptado (un bit o un byte). Esto puede ser comprendido más fácilmente observando la figura 5.10:

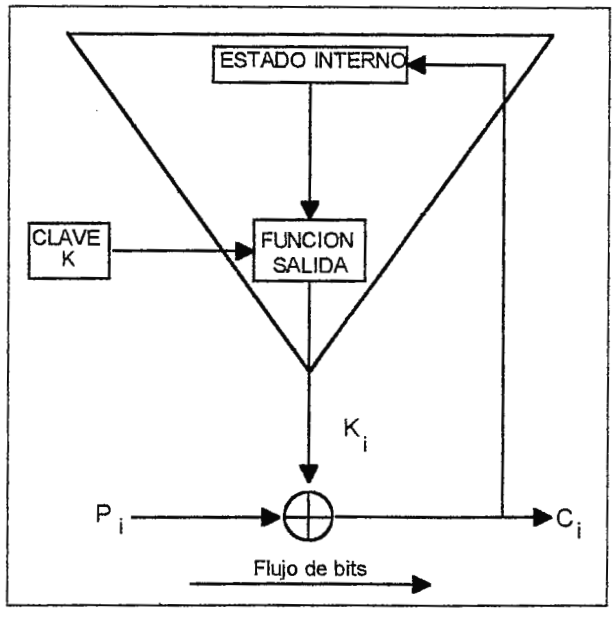


FIGURA 5.10

Si se desea utilizar como pauta de la evolución de la trama una cuenta progresiva de valores, se alimenta el estado interno de su mismo valor digital anterior, pero modificado por una función aritmética progresiva que hará las veces de clave de encriptamiento. Esto puede ser mejor comprendido observando la figura 5.11:

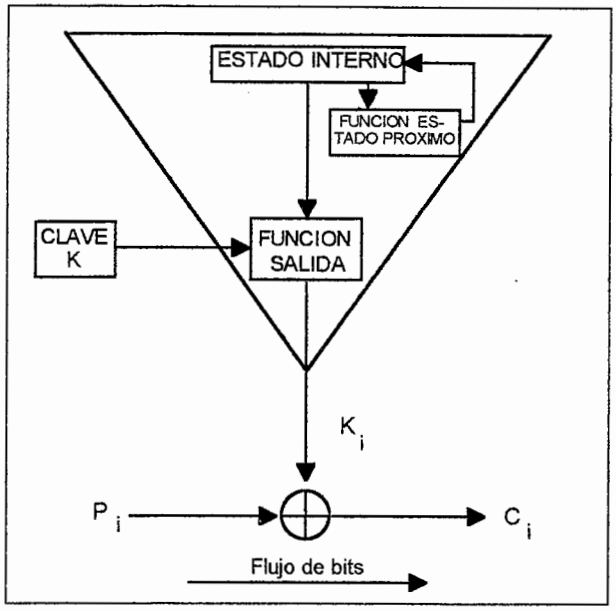


FIGURA 5.11

Existen también métodos extraordinariamente sencillos para generar secuencias de bits, tales como las combinatorias en el último byte que se ha encriptado (por ejemplo generar el nuevo bit mediante el tercero y el sexto bits operados en AND, y luego operados en OR con el cuarto...) todos ellos pueden ser aplicados a el modo Cipher Feedback o cipher autokey. La apariencia de aleatoriedad se logra fácilmente cuanto más caprichosa sea la forma se obtener el nuevo bit de clave.

Los algoritmos cifradores de trama pueden ser desarrollados mediante software, pero en su mayoría son implementados en bloques de hardware (en cuanto a aplicaciones de informática). En algunos sistemas de firewalls es el mismo sistema de protección el que realiza el encriptamiento de trama, en otros sistemas también puede ser realizado por un servidor proxi especialmente configurado para trabajar en esta forma. Estos sistemas de enlaces utilizan encriptamiento de trama por su rapidez, ya que encriptar bloques les tomaría mucho más tiempo, y se volverían lentos para realizar sus labores de enlace [9].

Existen también productos de hardware con encriptamiento de trama en linea [10],estos pueden ser instalados en los dispositivos de entrada/salida de una computadora de escritorio, como tarjetas internas o como equipo externo, de manera que para las máquinas que lo utilizan, su operación es completamente transparente, tanto para el que emite la información como para el que recibe la información [6].

En todo producto de hardware como el anteriormente descrito debe tenerse especial cuidado en las conexiones físicas del equipo: generalmente se utilizan las conexiones marcadas con rojo para el texto fuente, y las conexiones marcadas con negro para en texto cifrado. Los alambres que empalman dichas conexiones deben estar magnéticamente aislados uno del otro. Esto obedece a una anécdota de la guerra fría: Durante los años del muro de Berlín, el ejército soviético implementó un sistema de encriptamiento por hardware, pero no aisló nunca las conexiones rojas de las conexiones negras, esto provocó lo que los soviéticos denominaron como un "ruido aleatorio" en el texto encriptado, pero que en realidad era el texto fuente muy atenuado. Este error le costó caro a los soviéticos, pues sus enemigos tuvieron acceso a su información sin que ellos se dieran cuenta de ello [6].

[10]Information Resources Engineering Inc. (IRE) fabrica un dispositivo externo a la máquina llamado HS Remote Encription, el cual maneja en uno de sus puertos texto fuente (entrada o salida), y en otro de sus puertos sólo texto encriptado (entrada o salida), de manera que para los puertos a los cuales esta conectado parece simplemente un canal de comunicación para realizar la transferencia de datos.

6.- FUNCIONES MATEMATICAS USADAS POR LA CRIPTOGRAFIA

Para la mejor comprensión del material que se expondrá en los capítulos siguientes es necesario conocer algunas de las funciones matemáticas en las cuales se asientan los algoritmos de encriptamiento, y también algunos procedimientos para generar los elementos que las técnicas de encriptamiento en clave pública necesitan para crear sus sistemas de claves distintas pero complementarias.

Con el fin de satisfacer esta necesidad de conocimiento previo, y para presentar un trasfondo matemático de referencia y medida que ayude a la apreciación de cada algoritmo, se presenta en este capítulo una exposición rápida de la función módulo (sumamente necesaria para procesar datos encriptados), de sus propiedades y sus características, así como también de la generación de números primos con gran cantidad de cifras y de números primos relativos (de capital importancia en los algoritmos de clave pública y de autenticación digital). Todo el material mostrado a continuación ha sido editado a partir del capitulo "math Background" del libro "Applied Cryptography" (vease bibliografía [8]).

6.1 FUNCION MODULO

Son funciones periódicas aquellas expresiones algebraicas o trigonométricas que devuelven un valor de comportamiento cíclico cuando la variable independiente crece en forma lineal.

La función x Módulo(n) puede definirse mediante el comportamiento de su entrada y su salida, como una función x que devuelve siempre un valor periódico situado siempre entre 0 y n-1 , la variable independiente x puede tomar valores enteros desde 0 hasta infinito, y el valor de salida de la función para un x dado puede definirse como el residuo que deja la división de x entre n, no importando el valor que se obtenga como cociente. Por este motivo a esta función se le conoce también como función residuo, y este valor residual es siempre menor que n pero mayor o igual que cero, y recibe el nombre de reducción modular[11].

[11] En lenguaje C esta función se puede redactar como a % b , lo cual se interpreta como: a modulo b. Se puede apreciar una

Continúa en la página siguiente...

Otra manera de desarrollar la función consiste en encontrar el valor que se genera al restar de x el valor de n , revisar el resultado, y si este es mayor que n repetir la sustracción en operaciones sucesivas hasta conseguir un valor menor que n pero mayor que 0.

Esta función se denota como:

$$f(x) = x \text{ módulo } (n)$$

o simplemente como:

$$f(x) = x \bmod n$$

El valor que genera la función es conocido como "reducción modular de x mod n".

Un ejemplo de este tipo de función se encuentra en el tipo horario de 24 horas expresado en grupos de 12. Es un hecho: los relojes análogos sólo contienen 12 marcas en su pantalla, pero el día tiene 24 horas. El tipo horario de grupos de 12 es siempre una función: x módulo (12), de manera que en cuanto se dice "son las siete horas"; son las $7 \bmod (12) = 7$ (hora de la mañana). Y cuando se habla de "las 16 horas"; se quiere significar las $16 \bmod (12) = 4$ (hora de la tarde). En ámbitos militares y científicos, cuando se pretende medir el instante de realización de un evento en un proyecto que comenzó a las 0:00:00 horas del día, al hablar de las 27 horas se estaría dando a entender que son las $27 \bmod(12) = 3$ (hora de la mañana que marca el reloj civil).

Un concepto importante relativo a la función módulo es la congruencia de dos o más números. Dos o más números son congruentes si el valor de su reducción modular es igual. Esta propiedad se denota con el símbolo "≡ " (equivalencia), y puede expresarse de la siguiente manera:

$$a \equiv b$$

si: $$a \bmod (n) \equiv b \bmod (n)$$

Como estos dos números son congruentes con un tercero, que es el valor de su reducción modular, y este valor es el mismo para ambos números, esta propiedad también puede ser expresada de la siguiente manera [8]:

$$a \equiv a \bmod (n) \equiv b \bmod (n)$$

Viene de la página anterior...
aplicación interesante de esta función en la transposición que sufre la clave en cada iteración en el código fuente del programa de encriptamiento que se muestra en el capítulo 16.

$$b \equiv b \bmod (n) \equiv a \bmod (n)$$

entonces:

$$a \equiv b \bmod (n)$$

$$b \equiv a \bmod (n)$$

Las propiedades de esta función pueden enumerarse de la siguiente manera [8]:

x modulo (n) es conmutativa:

$$(a + b) \bmod n = [(a \bmod n) + (b \bmod n)] \bmod n$$

$$(a - b) \bmod n = [(a \bmod n) - (b \bmod n)] \bmod n$$

x modulo (n) es asociativa:

$$(a * b) \bmod n = [(a \bmod n) * (b \bmod n)] \bmod n$$

x módulo (n) es distributiva:

$$(a * (b + c)) \bmod n = \{ [(a * b) \bmod n] + [(a + c) \bmod n] \} \bmod n$$

6.1.1 INVERSA DE LA FUNCION MODULO.

Si un número a es el valor congruente de un número b tal que se cumple que:

$$a \equiv b \bmod n$$

Entonces el inverso de esta función módulo de dicho número es[8]:

$$a^{-1} \equiv (b \bmod n)^{-1}$$

$$1/a \equiv b \bmod n$$

$$1 \equiv (a * b) \bmod n$$

La anterior expresión expresada con palabras puede ser interpretada de la siguiente manera: " El inverso de b módulo (n) es un número tal que multiplicado por b produce un valor el cual operado en la función modulo (n) devuelve el valor de 1".

Se debe recordar que el valor 1 es producido en una función módulo cuando el valor a operar es n+1, 2n +1, 3n+1, ... Para producir este valor como operando de la función se deberá multiplicar b por un valor entero (puesto que esta función sólo opera valores enteros). Esto no siempre es posible, pues existen números abajo de n que nunca producirán una solución entera para la expresión:

$$a * b = n+1 = 2n+1 = \ldots$$

$$a = \frac{n}{b} + \frac{1}{b} = \frac{n+1}{b} = \frac{2n}{b} + \frac{1}{b} = \frac{2n+1}{b} = \ldots$$

Estas expresiones sólo retornarán un valor entero cuando n+1 o 2n+1 den como resultado un múltiplo de b, y esto se cumplirá cuando b y n no tengan ningún factor en común, ya que si lo tuvieran, cualquier múltiplo de n podría reducirse de alguna manera con b, y el término $\frac{1}{b}$,que es obviamente una fracción, no tendría ninguna oportunidad de resolverse, y el número "a" no podría ser un entero. Además existe la posibilidad que un múltiplo de n al cual se le sume 1 sea divisible entre b (tomando en cuenta que el factor que lo multiplica puede ser desde 1 hasta infinito).

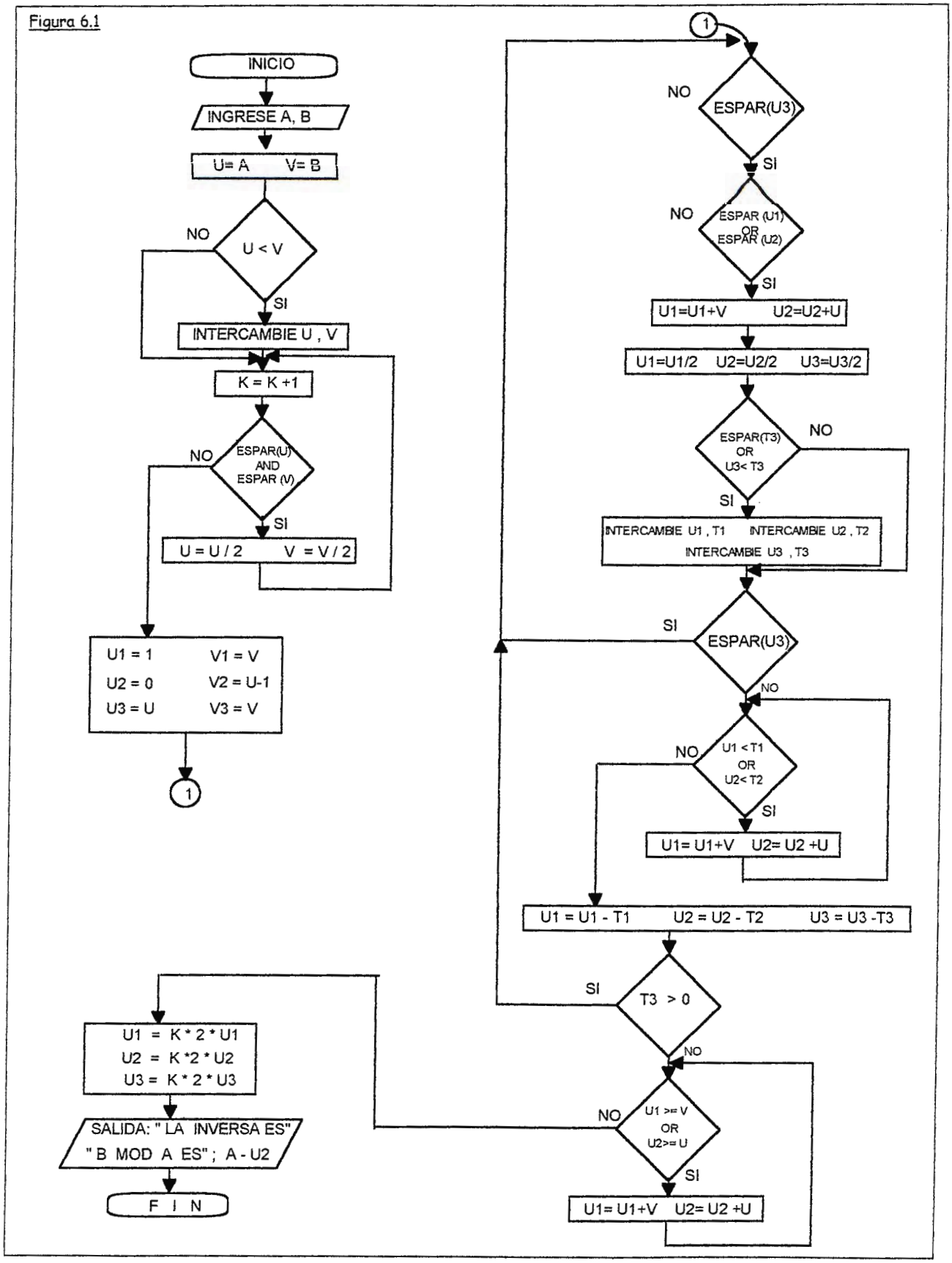
Todo lo anterior se puede expresar brevemente de la siguiente manera: "Una expresión b módulo (n) tendrá un valor inverso denotado por: a , si y sólo si b y n son números primos relativos entre si".

Recuérdese que dos números primos entre sí son aquellos que no comparten ningún factor entre ellos aparte del numero 1. O en otras palabras: dos números son primos entre si cuando su máximo común divisor es 1.

Existe un procedimiento para averiguar el valor inverso de la función módulo de un número b (b mod (n)) este procedimiento conocido como algoritmo de Euclides puede ser expresado como un diagrama de flujo, este diagrama se muestra a continuación[12]:

[12]Puede encontrarse este algoritmo escrito en forma de código fuente en lenguaje C++ en el apartado "math Background" del libro "Applied Cryptography" (vease bibliografía) , en este documento se ha expresado de esta manera para agilizar la comprensión en el lector poco familiarizado con dicho lenguaje de programación.

Figura 6.1


INICIO
INGRESE A, B
U= A V= B
NO
U < V
SI
INTERCAMBIE U , V
K = K +1
NO
ESPAR(U) AND ESPAR (V)
SI
U = U / 2 V = V / 2
U1 = 1 V1 = V
U2 = 0 V2 = U-1
U3 = U V3 = V
1
1
NO
ESPAR(U3)
SI
NO
ESPAR (U1) OR ESPAR (U2)
SI
U1=U1+V U2=U2+U
U1=U1/2 U2=U2/2 U3=U3/2
ESPAR(T3) OR U3< T3
NO
SI
INTERCAMBIE U1 , T1 INTERCAMBIE U2 , T2
INTERCAMBIE U3 , T3
SI
ESPAR(U3)
NO
NO
U1 < T1 OR U2< T2
SI
U1= U1+V U2= U2 +U
U1 = U1 - T1 U2 = U2 - T2 U3 = U3 -T3
SI
T3 > 0
NO
NO
U1 >= V OR U2>= U
SI
U1= U1+V U2= U2 +U
U1 = K * 2 * U1
U2 = K *2 * U2
U3 = K * 2 * U3
SALIDA: " LA INVERSA ES" " B MOD A ES" ; A - U2
F I N

6.2 GENERACION DE NUMEROS PRIMOS.

La generación de números primos es de suma importancia en los algoritmos de clave pública, no se trata ,como se pudiera pensar, de números como el 5 y el 7, sino de números cuya representación binaria abarca más de 100 o 200 cifras. El motivo de usar números de tal magnitud radica en la seguridad del algoritmo de clave pública, el cual toma parámetros para encriptamiento bazados en estos números pero sin revelar los números que lo originaron, algunos de ellos serán usados para encriptar, y otros para desencriptar, y su único nexo común serán estos números. Encontrar dos valores de tal magnitud que puedan relacionarse con los parámetros de encriptamiento y desencriptamiento es verdaderamente difícil, pues aunque algunos se relacionen con unos, pueden no relacionarse con otros.

Escoger dos números primos para iniciar el proceso de elaboración de los parámetros de encriptamiento puede ser también una tarea larga y difícil, para estos casos, el algoritmo de Rabin-Miller (desarrollado por Michael Rabin y Gary Miller) constituye una útil herramienta matemática. A continuación se lista el procedimiento a seguir para generar un número primo al cual se le llamará P [8].

(1) Se escoge al azar un número para analizarlo, para saber si es primo o no lo es .A este número se le designa con la letra { P }. Se inicializa la variable b con el valor de cero.

(2) Se divide entre dos en forma sucesiva el número (P -1), para cada división se incrementa el valor {b} en uno.

(3) Si el resultado de la operación anterior es par se regresa al paso (2). De lo contrario se continúa con el siguiente paso.

(4) Se guarda el valor de {b}, este representa el número de veces que (p-1) es divisible por dos.

(5) Se asigna a M el valor:

$$M = (P - 1)/ 2^b$$

(6) Se escoge un número {A} tal que : $A < P$

(7) Se inicializa: $J = 0$

(8) Se define la variable {Z} como:

$$Z = A^M \bmod P$$

(9) Si { Z =1 } o si { Z = P - 1 } entonces existen posibilidades que P sea primo, se continúa con el procedimiento, de lo contrario se interrumpe el proceso, se escoge otro A y se regresa al paso 6.

(10) Si { J < 0 } y además { Z = 1 },entonces P no es primo; se interrumpe el proceso, se escoge otro P y se comienza de nuevo desde el principio, en caso contrario se continúa el proceso.

(11) Se define: $J = J + 1$

(12) Si { J < b} y además { Z no es igual a (P-1) } entonces asígnese a la variable Z el valor definido por: $Z = Z^2 \bmod P$

y regrese al paso (10).

(13) Si { Z = (p-1) } entonces existen posibilidades que P sea primo, se continúa con el procedimiento, de lo contrario se interrumpe el proceso, se escoge otro P y se comienza de nuevo.

(14) Si { J = b } y además { Z no es igual a (P-1) } ,entonces P no es primo; se interrumpe el proceso, se escoge otro P y se comienza de nuevo, en caso contrario el número P es primo.

(15) fin.

Si se consigue llegar hasta el paso 15 es porque el número es primo. Este proceso puede parecer largo y difícil, pero en realidad es muy rápido si se compara con métodos rudimentarios.

Otro procedimiento más sencillo se desarrolla con la simple escogitación de los números impares cuya suma de sus cifras no sea múltiplo de 3, ni su última cifra sea cinco, y su posterior búsqueda de todos sus posibles múltiplos mediante divisiones con residuo cero. Con números de más de 200 cifras la búsqueda de posibles divisores puede tardar mucho tiempo, pues deberán examinarse una parte bastante grande de los números menores a el valor propuesto.

En la práctica, seguir el procedimiento de Rabin-Miller para cada numero aleatorio que se produzca, puede ser demasiado largo y complicado. Puede resultar un poco más fácil si se desarrolla un procedimiento previo, que acorta significativamente el proceso de prueba del número "primo". Este chequeo previo se lista a continuación:

- Se genera un número aleatorio de n bits.
- Este número debe tener un uno en el bit menos significativo (para que sea impar)
- Se asegura que dicho número no sea divisible entre pequeños números primos tales como el 3, 5, 7, 11, ... (lo normal es probar hasta los menores de 256, pero lo mejor es verificar hasta los primos menores de 2000).

- ❑ se aplica Rabin-Miller al numero, si pasa la prueba se toma como primo, de los contrario se comienza de nuevo.

Cuando se trata de encontrar dos números primos entre si la descomposición en factores y la consiguiente comparación de dichos factores para encontrar el máximo común divisor puede ser un procedimiento largo y difícil Para acortar dicho procedimiento se puede seguir los siguientes criterios para escoger dos números P y Q que sean primos entre si:

- El máximo común divisor de (P-1) y (Q-1) debe ser pequeño.
- ambos, (P-1) y (Q-1) , deben tener factores primos muy grandes (se les llamara a estos factores P´ y Q´).
- Los factores primos menos uno: (P´- 1) y (Q´- 1) a su vez deben ambos tener factores primos grandes
- Los factores primos más uno: (P´+ 1) y (Q´+ 1) a su vez deben ambos tener factores primos grandes
- Ambas expresiones: (P-1)/2 y (Q-1)/2 deben dar como resultado números primos.

7.- ALGORITMOS ASIMÉTRICOS O DE CLAVE PÚBLICA.

Este capítulo muestra una aproximación a el mundo real de los algoritmos asimétricos o de clave pública. Se detalla paso a paso el procedimiento que siguen dichos algoritmos, tanto para el encriptamiento como para la autenticación de mensajes digitales. Pero antes de listar estos algoritmos se expone una visión de conjunto para dejar claro el objetivo que se persigue con estos procedimientos, que en apariencia pueden resultar sumamente confusos cuando no se tienen en la mira los objetivos del encriptamiento.

Como se ha dicho anteriormente, este tipo de algoritmos desarrolla un encriptamiento el cual es casi imposible de revertir mediante el proceso que los formó. Podría pensarse que este tipo del algoritmos genera el texto cifrado a partir de una función matemática con una función inversa ambigua, esto es, tiene varios valores x capaces de generar un sólo valor en $f(x)$ (considerando a x como el texto fuente, y a $f(x)$ como el texto cifrado).

Pero el problema va mas allá de encontrar una simple función no inyectiva (a cada elemento del rango NO le corresponde uno y sólo uno elemento del dominio), el verdadero problema en el encriptamiento en clave pública es que para encriptar el texto fuente se hace uso de una función que no tiene función inversa, es decir, es difícil, y verdaderamente muy difícil de encontrar mediante los métodos tradicionales algebraicos de factorización y reducción de términos.

Para ilustrar mejor lo dicho anteriormente se tomará un ejemplo de este tipo de funciones en el álgebra tradicional de los número reales.

$$f(x) = e^{-x} - x$$

Es un ejemplo típico de funciones "sin solución" que en análisis numérico se usa para demostrar la utilidad de los métodos iterativos en la solución de ecuaciones en las cuales no puede ser despejada la variable x, pero sí puede ser encontrada con razonable aproximación mediante métodos iterativos.

En el mundo del encriptamiento no se trabajará con números reales, sino con números digitales binarios, o por lo menos con números enteros. El encriptamiento no puede hacer aproximaciones en sus transformaciones, tanto en las de encriptamiento como en las de desencriptamiento, por lo tanto sus funciones deben trabajar tanto en su dominio como en su rango exclusivamente con números enteros.

Como se verá mas adelante , aunque en apariencia la función de encriptamiento no tiene inversa, ésta se encuentra en algún lugar del método de desarrollo de dónde esta función fue obtenida.

Para lograr este objetivo se parte de un origen común que define las propiedades de un par ordenado, una de las propiedades de este par ordenado desarrolla la función de encriptamiento, y otra de las propiedades (inversa a la anterior) define el desarrollo de la función de desencriptamiento. Si luego el par ordenado y las propiedades que originaron las funciones se desechan, quedan solamente dos expresiones matemáticas, y si estas definen funciones algebraicas imposibles de despejar la variable independiente, se tiene entonces funciones sin posibilidad para encontrar su expresión inversa.

No existen técnicas generales en los métodos de encriptamiento por clave pública pues generar una función sin inversa puede tener muchos caminos, y también la aplicación de esta función para fines de encriptamiento.

La clave pública no sólo involucra el encriptamiento en sí, también incluye la autenticación, esto comprende las firmas digitales, las funciones Hash, y los códigos de autenticación de mensajes (MAC: Message Autentication Code), puesto que en todos ellos una parte del proceso da muy pocas pistas de la otra. A continuación se expondrá brevemente el funcionamiento de algunos algoritmos de clave pública, los cuales son usados con mucha frecuencia y han probado su relativa seguridad para los tiempos actuales.

7.1 ENCRIPTAMIENTO CON ALGORITMO RSA .

Este algoritmo debe su nombre a sus diseñadores: Ron Rivest,Adi Shamir y Leonard Adleman, fue creado en 1977, y publicado en 1978, fue el primer algoritmo verdaderamente efectivo para el encriptamiento en clave pública, muchos métodos anteriores necesitaban demasiado tiempo para crear dos claves, y para

desarrollar el proceso de desencriptamiento. Su seguridad se basa en la dificultad de factorar números verdaderamente grandes, con este objetivo define dos funciones para su sistema, una para encriptar, y otra para desencriptar, ambas son verdaderamente difíciles de invertir en cuanto a el despeje de sus variables en una manera razonablemente desarrollable [3].

RSA puede ser usado tanto para encriptar (cuando se hace pública la clave para encriptamiento) como para autenticar (cuando se hace pública la clave para el desencriptamiento), pues en cualquiera de los dos casos tanto una clave como la otra dan muy pocas pistas para deducir su contraparte [9].

GENERACIÓN DE LAS FUNCIONES DE ENCRIPTAMIENTO Y DESENCRIPTAMIENTO [8]:

1.- Se escogen dos números primos razonablemente grandes (100 o más cifras binarias), y se les denomina:

$$p \qquad y \qquad q$$

2.- Se calcula el producto de dichos números y se le denomina n:

$$n = p * q$$

3.- Se escoge al azar un número al cual se llamará {e}, y cumple la característica:

$$e \text{ es primo relativo del producto } (p-1)(q-1)$$

Esto quiere decir : se escoge aleatoriamente un numero, si éste no tiene ningún divisor común con el producto (p-1)(q-1) excepto 1 , se le llama {e} , de lo contrario se escoge otro número al azar y se repite el análisis hasta encontrar uno que cumpla con la característica deseada.

4.- Se calcula un número d que cumpla ser el valor inverso de:

$$e \bmod (p-1)(q-1)$$

Es decir:

$$1 \equiv d * e \bmod (p-1)(q-1)$$

Este número existe, puesto que {e} y (p-1)(q-1) son primos entre sí, y puede ser encontrado fácilmente aplicando el algoritmo de Euclides para encontrar la inversa de una función módulo (vease capítulo 6).

5.- La clave de encriptamiento para cada bloque de mensaje {m} se toma de la función:

$$C = m^e \bmod (n)$$

6.- La clave para el desencriptamiento de cada bloque de texto cifrado {C}se toma de la función:

$$m = C^d \bmod (n)$$

7.- Se desechan de cualquier medio de memoria los valores {p} y {q} , y se utilizan las funciones en forma de algoritmos.

Se puede demostrar la validez de las ecuaciones ya que :

$$C^d = (m^e)^d \bmod n$$

$$C^d = m^{ed} \bmod n$$

$$C^d = m^{K(p-1)(q-1)+1} \bmod n$$

$$C^d = m * m^{K(p-1)(q-1)} \bmod n$$

$$C^d = m * 1 \bmod n$$

$$m \equiv C^d \bmod n$$

La seguridad de este algoritmo es simple, existen dos números: {e} y {d} uno de ellos se hace público, el otro no. Muchas personas pueden saber que se usa este algoritmo para encriptar en clave pública, y muchas personas pueden tener esta clave pública , con la ecuación y uno de los dos números, pero teniendo uno de los números es muy difícil que puedan encontrar el otro.

Se puede tratar la manera de criptoananlizar este método de cifrado, se tiene n, se tiene e, se sabe que n= p*q , se sabe que e es primo relativo del producto: (p-1)(q-1), y se sabe que p y q son números primos. Cabe la posibilidad de probar todos los números primos desde el 3 hasta aquellos que tienen 250 cifras de longitud, si su producto es igual a n y además el producto (p-1)(q-1) es primo relativo con e, entonces se habrán encontrado p y q . Como puede verse, es verdaderamente difícil encontrar dicho número, pues es necesario examinar $2^{250} = 1.8093 \times 10^{75}$ números diferentes para saber si son o no primos, y encontrados los dos números debe de probarse si cumplen las condiciones requeridas. Es posible que cada prueba tome en promedio unos 5000 ciclos de reloj esto en una máquina de 400 Mhz sería 1.25×10^{-5} segundos, y en examinar todos los números se tardará $2.26 \times \times 10^{70}$ segundos, lo cual sería cerca de $7.17 \times \times 10^{62}$ años. Y debe tomarse en cuenta que puede ser que el número sea encontrado en el primer intento, o puede ser que sea encontrado hasta el último.

7.2 SECURE HASH ALGORITHM (SHA) [8].

SHA fue diseñado en forma conjunta por NSA (National Secure Agency) y NIST (National Institute of Standards and Technology) con el objetivo de crear una norma para reconocer en forma universal las funciones Hash y los valores que generan. En este sentido fue creado el estándar para funciones Hash llamado Secure Hash Standard (SHS), en el cual se utilizó Secure Hash Algorithm. SHS es el estándar, SHA es el algoritmo usado para implementarlo, pero no es el estándar.

Secure Hash Algorithm es un algoritmo para generar una función Hash de un solo sentido a partir de un texto fuente, el cual debe de poseer una longitud de n * 512 bits (n = número entero). Dado que cumplir esta condición no es siempre posible, se arregla el texto fuente mediante un relleno de bits, el cual comienza con un uno a continuación del bit final del texto fuente, y termina con un número de 64 bits que indica la longitud del mensaje. A todos los bits entre el uno y la longitud del texto se les asigna el valor de cero.

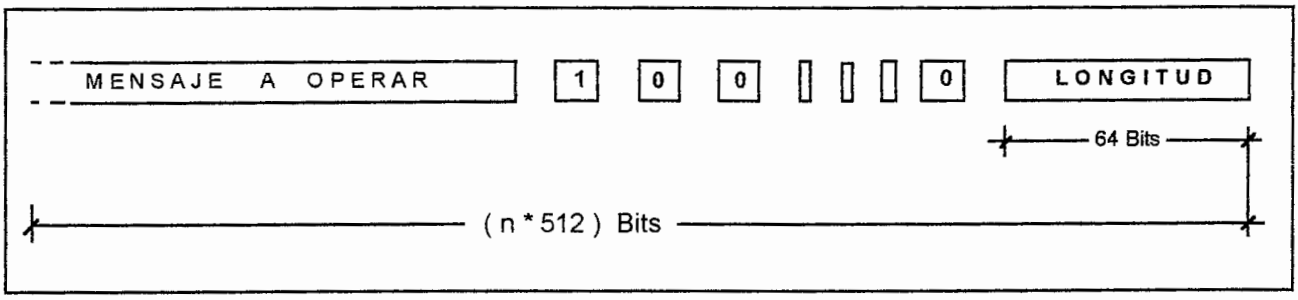


FIGURA 7.2.1

Por cada bloque de 512 bits en los cuales el mensaje será fragmentado, se producirá un valor Hash de 160 bits de longitud, de manera que el valor hash total del mensaje estará compuesto de la concatenación de todos los valores Hash que se obtuvieron de los bloques.

GENERACIÓN DEL VALOR HASH:

SHA es un algoritmo que genera valores Hash para cada bloque, trabaja cada bloque en una serie iterativa la cual se repite 80 veces, y produce un valor de 160 bits de longitud.

El algoritmo de esta función comienza por la asignación de valores a las variables de 32 bits que van a ser operadas con el mensaje durante el algoritmo. Se definen como valores iniciales los siguientes números hexadecimales:

A = 67452301

B = efcdab89

C = 98badcfe

D = 10325476

E = c3d2e1f0

Cada bloque de 512 bits es fraccionado en 16 sub-bloques de 32 bits, los cuales serán denotados por M_t , con t = 0 hasta 15. Además se inicializarán las variables a participar en las 80 iteraciones que desarrolla el algoritmo. Estas variables de 32 bits tendrán lo valores siguientes:

a = A

b = B

c = C

d = D

e = E

Una vez definidas e inicializadas las variables se desarrolla el siguiente procedimiento:

(1) Para t = 0 hasta 79.

(2) TEMP = a rot (5) + f_t(b, c, d) + e + W_t + K_t

(3) e = d

(4) d = c

(5) c = b rot (30)

(6) b = a

(7) a = TEMP

(8) si t < 79 incremente t en uno y regrese a (1).

(9) asígnese: A = A + a

B = B + b

C = C + c

D = D + d

E = E + e

(10) Salida: valor Hash del bloque: ABCDE (concatenación con un total del 160 bits).

(11) Si el anterior bloque no era el último, repítase todo el proceso.

DONDE:

a rot (5) significa rotar los bits del número a cinco espacios a la izquierda.

b rot (30) significa rotar los bits del número b treinta espacios a la izquierda.

$$
\begin{aligned}
f_t(b, c, d) &= (b \text{ AND } c) \text{ OR } (\text{NOT}(b) \text{ AND } d) && \text{para } 0 <= t <= 19 \\
&= b \text{ EXOR } c \text{ EXOR } d && \text{para } 20 <= t <= 39 \\
&= (b \text{ AND } c) \text{ OR } (b \text{ AND } d) \text{ OR } (c \text{ AND } d) && \text{para } 40 <= t <= 59 \\
&= b \text{ EXOR } c \text{ EXOR } d && \text{para } 60 <= t <= 79
\end{aligned}
$$

$$
\begin{aligned}
K_t &= 5a827999 && \text{para } 0 <= t <= 19 \\
&= 6ed9eba1 && \text{para } 20 <= t <= 39 \\
&= 8f1bbcdc && \text{para } 40 <= t <= 59 \\
&= ca62c1d6 && \text{para } 60 <= t <= 79
\end{aligned}
$$

$$
\begin{aligned}
W_t &= M_t && \text{para } 0 <= t <= 15 \\
&= [\ W_{(t-3)} \text{ EXOR } W_{(t-8)} \text{ EXOR } W_{(t-14)} \text{ EXOR } W_{(t-16)} \text{ EXOR }\ ] \text{ rot } (1) \\
& && \text{para } 16 <= t <= 79
\end{aligned}
$$

El valor total que se genera de este algoritmo puede ser usado para ser adosado a un mensaje cuando se quiere certificar su integridad, o bien para un posterior reconocimiento, o para distribuirlo como prueba de autenticidad para ser verificado mediante la operación del valor que lo originó (el cual es el que va a ser dado como prueba a quien tiene el valor hash y el algoritmo).

Este algoritmo es tomado como base para desarrollar el estándar de Firmas digitales que se explicará a continuación en el apartado siguiente.

7.3 DIGITAL SIGNATURE ALGORITHM (D S A).

Digital Signature Algorithm surge en 1992 como una propuesta para el proyecto de National Institute of Standards and Tecnogy (NIST) de crear un sistema estandarizado de firma digital, el cual sería llamado Digital Signature estándar (DSS). El algoritmo fue tomado en cuenta en el proyecto, asimilado íntegro, pero no le fue respetado su nombre. De esta fecha en adelante ha habido confusión en el verdadero nombre de una firma digital estandarizada, para algunos el estándar oficial es el DSS para otros es el DSA.

Para algunos autores, los más abiertos a aceptar los argumentos de ambos bandos, DSA es el algoritmo, DSS es el standard, el cual emplea el algoritmo, y el algoritmo es parte del standard, pero no es el estándar [8] .

DSA es un algoritmo para generación de firmas digitales, su funcionamiento se basa en las técnicas de encriptamiento en clave pública. En forma similar al RSA tiene una función matemática para transformar el mensaje en un texto encriptado, y una función para desencriptarlo

A continuación se muestra el desarrollo que se hace de las variables y la funciones para crear un par de expresiones matemáticas que cumplan la misión de crear un adosado de datos para comprobar la integridad y origen del mensaje, y la misión de verificar la validez del adosado de datos (verificación) [8].

1.- Se crean las variables con las cuales se va a construir la función de encriptamiento:

p : Se escoge un número { p } que sea primo cuya longitud sea entre 512 y 1024 bits, esta longitud debe ser un múltiplo de 64.

q : Se escoge un número { q } que sea primo y de longitud fija de 160 bits, y que además sea un factor de el número (p-1).

h : Se escoge un número { h } tal que sea menor que (p-1), y además cumpla el requisito de:

$$h^{(p-1)/q} \bmod p \text{ sea mayor que } 1$$

Para encontrar este número se hace al igual que el anterior, se propone uno, se opera la función, y si no cumple se descarta y se busca otro número.

g : Se desarrolla este número { g } resolviendo la función:

$$g = h^{(p-1)/q} \bmod p$$ puede verse que $g \equiv h^{(p-1)/q}$

x : Se escoge un número { x } cualquiera tal que sea menor que q (un número de 160 bits).

y : Se desarrolla un número { y } a partir de la expresión:

$$y = g^{x} \mod p$$

Las variables { p } , { q } y { g } pueden ser conocidas y hasta publicadas, pues su conocimiento no posibilita de ninguna manera la falsificación de una firma digital.

La variable { x } debe ser generada (o al menos conocida) solamente por el emisor de la firma digital.

La variable { y } será publicada por el firmante a todas aquellas personas o sistemas de información con quienes desea comunicarse en forma autenticada.

2.- El firmante desarrolla los parámetros del encriptamiento en forma particular en el momento de autenticar su mensaje:

(1) Genera un número aleatorio K tal que sea menor que q.

(2) Desarrolla los valores { r } y { s } que serán adosados al mensaje:

$$r = (g^{k} \mod p) \mod q$$

$$s = [(k \mod q)^{-1} * (SHA(m) + x*r)] \mod q$$

Como puede apreciarse, los valores { r } y { s } deberán ser menores que q , puesto que cualquier valor generado en mod q es siempre menor que q. Ademas, se hace uso de la función Hash SHA sobre el mensaje a autenticar para verificar su integridad.

3.- El receptor recoge los valores de { r } y { s } adosados con el mensaje y procede a calcular { r } por un camino distinto. Si el { r } generado es igual al recibido, entonces se verifica la autenticidad. Para este efecto se desarrolla el siguiente procedimiento:

(1) Se calcula $W = [(S \mod q)^{-1}] \mod q$

(2) Se definen los valores:

$$U1 = [SHA(m) * W] \mod q$$

$$U2 = (r * W) \mod q \qquad \{ r \} \text{ recibido}$$

(3) Se calcula el { r } de comprobación:

$$r = [(g^{u1} * y^{U2}) \bmod p] \bmod q$$

Puede dar la impresión que el algoritmo es un poco lento e ineficiente debido al tiempo empleado en generar números primos, factores y valores inversos de función módulo, sin embargo estos valores no necesariamente deben ser desarrollados en el momento de autenticar un documento. El único valor que si debe ser desarrollado en el momento de la autenticación es el valor {s} , ya que este requiere el valor SHA de el mensaje, todo lo anterior puede ser calculado con mucho tiempo de anticipación.

Nótese que DSA no encripta el mensaje, únicamente genera valores para ser adosados con el mensaje. Si además de autenticación se necesita confidencialidad, lo más recomendable es encriptar el mensaje antes o después de ser autenticado, especificando al receptor el momento en el cual se llevó a cabo el encriptamiento (encriptamiento en clave secreta es lo más aconsejable, pues la clave pública haría mucho más tardado todo el proceso).

8.- ALGORITMOS DE ENCRIPTAMIENTO EN CLAVE SECRETA USADOS ACTUALMENTE EN INTERNET.

A continuación se presentan dos de los algoritmos más usados para el encriptamiento de documentos de números binarios en las transferencias de datos actuales. No son algoritmos nuevos, sino que tienen ya algunos años de uso. Su mérito se encuentra en su utilización actual en sus versiones originales y con algunas modificaciones.

El origen de ambos algoritmos se encuentran en licitaciones para crear un estándar de seguridad, razón por la cual su funcionamiento ha servido para el desarrollo de nuevos productos para encriptamiento basados en sus principios básicos de su funcionamiento.

La fortaleza de ambos algoritmos se basa en el hecho de que, aún cuando su funcionamiento ha sido publicado, su capacidad para hacer indescifrable un texto no ha perdido validez. Este fenómeno se explicará con claridad después de haber descrito las características de su funcionamiento.

También se describen en este capitulo otros algoritmos ampliamente usados en ámbitos gubernamentales y comerciales, los cuales gozan actualmente de cierto reconocimiento y aceptación en algunos paquetes de software ampliamente usados en el internet actual.

8.1 DATA ENCRYPTION STANDARD (DES).

Data Encryption Standard (DES) es tal vez el más popular, y relativamente seguro, algoritmo de encriptamiento en clave secreta. Su desarrollo comenzó en 1973 promocionado por the National Bureau of Standards (Actualmente the National Institute of Standards and Technology NIST) quien puso a licitación pública un método de encriptamiento para proteger datos en las computadoras y las transmisiones de datos. IBM ganó la licitación en 1974 creando un algoritmo altamente seguro, el cual fue publicado abiertamente al público en 1975 por el Registro Federal, para ser adoptado como standard federal en 1976 [8] [9].

Su uso en la actualidad es todavía frecuente, y aunque los detalles de su funcionamiento son conocidos por mucha gente, sus múltiples variaciones en sus parámetros de control y modos de

encriptamiento en que puede ser aplicado, y además, sus 2^{56} distintos valores posibles de clave de encriptamiento, lo hacen verdaderamente difícil de ser criptoanalizado.

DES es un algoritmo cifrador de bloque el cual toma bloques de 64 bits de texto fuente y los transforma en bloques de igual longitud de texto cifrado. La clave de encriptamiento también tiene 64 bits, pero cada octavo bit es usado para chequear paridad, así que para fines de exactitud en cuanto a cantidad de claves, se puede decir que solamente tiene 56 bits de clave.

8.1.1 funcionamiento del DES [8].

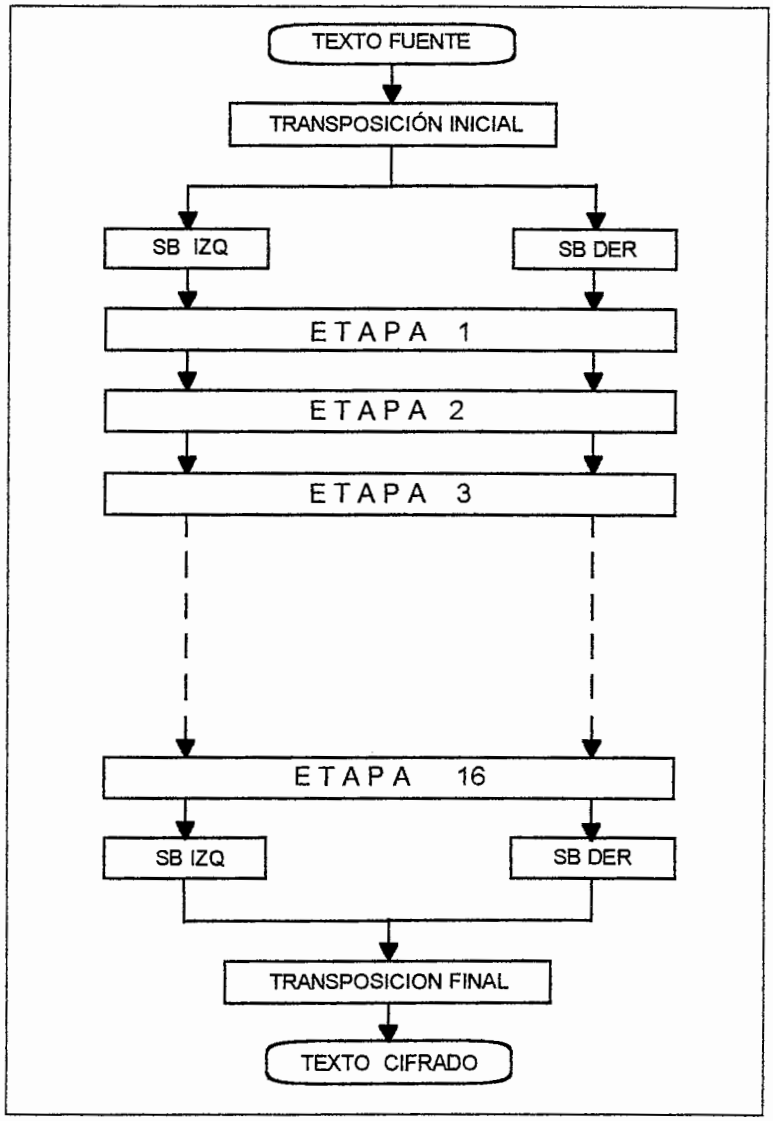


FIGURA 8.1.1

Para encriptar cada bloque el algoritmo realiza una transposición inicial de bits, seguida de una división inicial del bloque en dos sub-bloques: uno izquierdo SB IZQ , y uno derecho SB DER , ambos de igual longitud (32 bits). Luego realiza 16 etapas consecutivas de encriptamiento cada una con una entrada izquierda, una entrada derecha, una salida izquierda y una derecha. Y termina con una transposición final de bits para dar como salida un bloque encriptado. Todo esto puede ser comprendido con mayor facilidad observando la siguiente figura:

TRANSPOSICIÓN INICIAL:

La transposición inicial que se hace del texto fuente responde a una tabla de sustitución. Cada bit toma su nuevo estado del valor que posee el bit en la posición que le indica su número correlativo en la tabla (Se debe aclarar que la siguiente tabla, así como todas las posteriores, debe ser leída de izquierda a derecha, y de arriba hacia abajo, de manera que el número superior izquierdo es el primero, y el número inferior derecho es el sesenta y cuatro). La tabla estándar especifica las siguientes posiciones finales de los bits:

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 8.1.1

La tabla anterior indica lo siguiente: El bit número 1 tomará su estado del valor que tiene el bit de la posición 58, el bit número 2 será tomado de la posición 50, el bit número 3 de la 42, el 4 de la 34, ... y el bit 64 será tomado del valor en la posición 7.

TRANSPOSICION FINAL:

De la misma manera que la inicial, trasladará de lugar todos los bits que se hayan obtenido al final de las 16 etapas de encriptamiento, esto lo hará siguiendo las directrices que indican la siguiente tabla:

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 8.1.2

ETAPAS DE ENCRIPTAMIENTO DE CADA BLOQUE:

Las 16 etapas de encriptamiento que siguen a la transposición inicial son todas absolutamente iguales: cada una pasa el sub-bloque derecho al lado izquierdo, y realiza varias transformaciones en el mismo sub-bloque derecho para terminar operándolo en función EXOR con el sub-bloque izquierdo, y depositando el resultado en el mismo sub-bloque derecho.

El orden de las operaciones en el sub-bloque derecho antes de ser operado en función EXOR con el izquierdo es el siguiente:

1. TRANSPOSICIÓN DE EXPANSIÓN: Los bits del sub-bloque son expandidos de 32 a 48, esto se logra repitiendo bits específicos y colocándolos un posición atrás o una adelante de su sitio actual.
2. OPERACIÓN EN FUNCIÓN EXOR CON LA CLAVE: Para cada etapa la clave es dividida, rotada en bits y compresionada a 48 bits siguiendo patrones específicos según el número de la etapa.
3. SUSTITUCIÓN DE GRUPOS DE 6 BITS POR GRUPOS DE 4 BITS: Con esta operación se obtiene nuevamente un sub-bloque de 32 bits de longitud.
4. TRANSPOSICIÓN DE LA ETAPA: Cada bit es trasladado a una posición distinta siguiendo las directrices trazadas en una tabla de transposición.

Todo este proceso realizado en cada una de las 16 etapas puede ser mejor comprendido en una forma gráfica con ayuda de la siguiente figura:

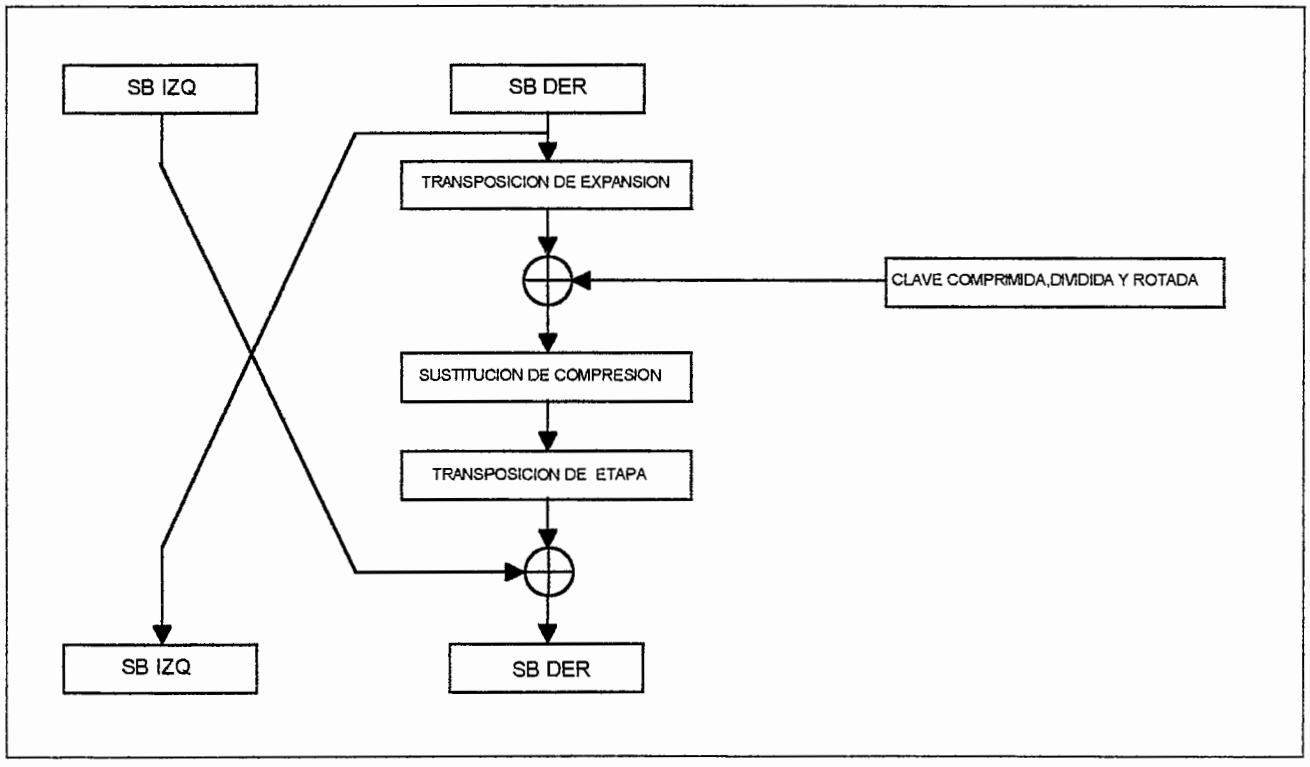


FIGURA 8.1.2

TRANSPOSICIÓN DE EXPANSIÓN:

En esta parte del proceso, se expanden los 32 bits del sub-bloque a 48 bits. Para realizar esta operación se realiza una transposición de bits en la cual cada cuarto bit de la serie junto con su bit siguiente son copiados dos veces: una vez seguido a su correlativo tercer bit y otra vez acontinuación de lo que se acaba de copiar (a los dos bits anteriores que son el cuarto y su siguiente) .

Todo esto puede sonar muy complejo, la tabla de transposición siguiente lo hace más fácil de comprender. Esta tabla especifica 48 posiciones de bits, cada posición contiene el número de bit que ha tomado de la ordenación original para llenar su espacio. Obsérvese que llena 48 casillas con 32 números, muchos de ellos se repiten.

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 8.1..3

SUSTITUCIÓN DE COMPRESIÓN:

En esta etapa del proceso, los 48 bit actuales serán comprimidos nuevamente a 32, para realizar esto, se dividirán los 48 bits en ocho grupos de seis bits, cada uno de estos grupos de seis bits será interpretado de la siguiente manera:

Los seis bits de cada grupo:	b 1	b 2	b 3	b 4	b 5	b 6

Numero de fila que se escoge en la tabla de sustitución:	b 2	b 3	b 4	b 5
Número de columna que se escoge en la tabla de sustitución:			b 1	b 6

Con los datos que se tienen en los 6 bits de cada grupo se escogerá un número ubicado en la tabla de sustitución, y este número (de 4 bits en notación binaria) se usará en lugar del grupo de 6 bits original. No existe una tabla común para todos los grupos de 6 bits, sino que cada grupo tiene su propia tabla, para hacer un total de 8 tablas de sustitución:

Tabla 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	1	3	14	10	0	6	13

TABLA 8.1.4

Tabla 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	43	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

TABLA 8.1.5

Tabla 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	5	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

TABLA 8.1.6

Tabla 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

TABLA 8.1.7

Tabla 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

TABLA 8.1.8

Tabla 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

TABLA 8.1.9

Tabla 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	9	5	0	15	14	2	3	12

TABLA 8.1.10

Tabla 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 8.1.11

TRANSPOSICIÓN DE LA ETAPA:

Una vez que se tienen 32 bits se opera una transposición de la etapa, al igual que en ocasiones anteriores, cada bit de la nueva disposición es tomado del bit que indica la tabla de transposición:

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

TABLA 8.1.12

Como puede observarse en la figura 8.1.2 , la clave de encriptamiento ha sido modificada antes de combinarse en función EXOR con el respectivo procedimiento de la etapa, estas modificaciones tienen lugar en cada etapa y son distintas para cada una de ellas. El orden de procedimientos para modificar la clave y hacerla de 48 bits para cada operación EXOR se muestra en el siguiente diagrama:

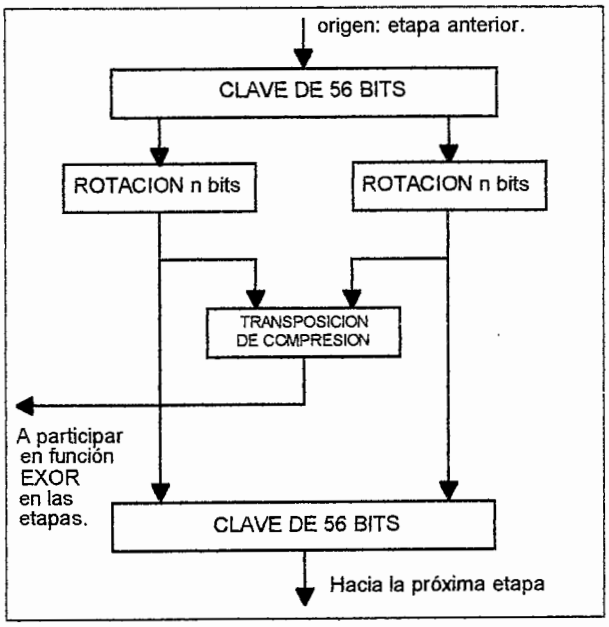


FIGURA 8.1.3

La clave originalmente tiene 64 bits, pero el algoritmo de encriptamiento ignora cada octavo bit (bit de paridad), con lo cual ésta queda con una longitud de 56 bits.

Esta clave es trabajada en bloques de 28 bits, los cuales son rotados hacia la izquierda n espacios (desplazados de su lugar un número n de espacios, el bit que llega al extremo izquierdo se desplaza al extremo derecho). El número de espacios que se desplaza cada rotación depende de la etapa a la cual se aplique la rotación, este numero es especificado por la tabla 8.1.13:

Etapa número	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
bits desplazados	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

TABLA 8.1.13

TRANSPOSICIÓN DE COMPRESION:

Para realizar esta transposición, cada bit de los 48 de la distribución de salida toma el valor de bit de entrada que le señala la tabla. Es necesario observar que la tabla de transposición no llama a los bits 8,16,24,32,40,48 y 56, todos estos bits son ignorados en la tabla de transposición; por este motivo la salida sólo tiene 48 bits. Vease a continuación la tabla de transposición de clave:

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 8.1.14

Nótese que aunque la clave le sean recortados 8 bits en el momento de su uso, ésta no pierde ninguno, ya que sólo ignora algunos de sus bits en el momento de la operación en función EXOR, pero sigue siendo rotada en sus 56 bits, y de estos 56 bits se ignoran 8 en la siguiente operación, pero que gracias a la rotación realizada con anterioridad no van a ser los mismos bits ignorados en la operación anterior.

8.1.2 Modos de operación de DES.

Data encryption Standard puede funcionar en cualquiera de los modos de operación mencionados anteriormente tales como ECB, CBC, OFB y CFB. El uso de los tres últimos modos de operación incrementa grandemente la confusión y la difusión de la información.

ANSI (American National Standards Institute) Recomienda que para fines de confidencialidad sean usados los modos ECB y CBC, y para fines de autenticación los modos de operación OFB y CFB con n bits por bloque de texto fuente.

8.1.3 Desencriptamiento del DES.

DES no necesita invertir el orden de su funcionamiento para desencriptar la información, ya que el mismo algoritmo de encriptamiento (siguiendo el mismo orden de operaciones en el mismo sentido) sirve para desencriptar.

Cada elemento del algoritmo realiza cambios en la información de tal manera que desarrollando el proceso en forma idéntica dos veces seguidas retornan el orden mismo de toda la información. Esto puede verse claramente en las tablas de transposición inicial y final de cada bloque, pues lo que una tabla cambia de posición, la otra lo devuelve[13].

Los elementos de transposición con expansión y transposición con compresión requieren un análisis mucho más cuidadoso para demostrar que su aparente ambigüedad al asignar valores con numerosos valores inversos posibles responden solamente a el hecho de que al aplicarse nuevamente se retornarán los valores originales. Se debe tener mucho cuidado al manejar las tablas de compresión y de expansión, ya que cada uno de esos valores ha sido calculado cuidadosamente para generar este comportamiento característico de $f(f(a)) = a$.

Los únicos elementos que sí deben ser manejados en orden inverso son la clave y sus valores para cada etapa de cada bloque. Para el desencriptamiento se debe iniciar el algoritmo con el valor final tomado por la clave, y éste debe rotado en orden inverso para cada etapa de bloque hasta llegar al valor inicial de la clave (obsérvese que los valores de rotación para cada etapa de cada bloque regeneran la clave al cabo de haber transcurrido las 16 etapas que tiene cada bloque.

8.1.3 SEGURIDAD DE EL DES.

Data Encription Standard puede ser considerado un algoritmo relativamente seguro: el tamaño de su clave arroja 2^{56} distintas posibilidades de números de clave, ademas de 4 posibles modos de operación, los cuales pueden ser inicializados con $3*2^{64}$ distintos vectores de inicialización crean $3*2^{122}$ alternativas distintas para escoger parámetros posibles que puedan interpretar el encriptamiento. Si cada alternativa examina su validez en $50*10^{-6}$ segundos, el tiempo para probar todas las posibles sería de 2.6585×10^{36} segundos, lo cual tomando en cuenta que el año tiene 3.15×10^{7} segundos, tomaría un total de 1.4×10^{27} años.

Debe recordarse que no necesariamente se deben probar todas las claves posibles. Si se usa el mismo generador de números aleatorios empleado para crear la clave, el número se reduce drásticamente, porque los generadores de números aleatorios no producen toda la gama de números, sino solamente un conjunto

[13] Vease el bit 1 de la tabla de transposición inicial, el cual es tomado de la posición 58, y el bit 58 de la tabla de transposición final, el cual es tomado de la posición 1.

relativamente pequeño, y si la clave es un nombre, este puede ser adivinado con relativa facilidad por el criptoanalista.

8.2 INTERNATIONAL DATA ENCRYPTION ALGORITHM (IDEA)

IDEA es un algoritmo creado en 1990 por Xuejia Lai y James Massey para el Instituto Federal Suizo de Tecnología, su prototipo fue llamado PES (Proposed Encryption Standar), y se desarrolló hasta convertirse en IDEA en 1991. Fue comercializado en 1992 bajo su configuración definitiva.

IDEA es un algoritmo de encriptamiento simétrico de bloques, al igual que Data Encription Standard, usa un mismo algoritmo para encriptar y para desencriptar el mensaje, es decir, puede ser concebido como una función $f(a)$ la cual :

$$f(\ f(a)\ ) = a$$

IDEA utiliza una clave de 128 bits de longitud, para encriptar bloques de 64 bits de longitud. En cuanto a su funcionamiento, es mucho más matemático y menos indexado que otros algoritmos. De hecho no incluye funciones digitales tales como la rotación de bits, su transposición, y su expansión o compresión. Su funcionamiento incluye únicamente tres funciones matemáticas:

- Función OR exclusiva (EXOR).
- Adición módulo (2^{16})
- Multiplicación modulo ($2^{16}+1$).

No tiene ninguna tabla que sirva como índice de alguna operación de encriptamiento; sin embargo la función módulo ($2^{16}+1$) puede ser considerada como una tabla índice expresada en términos matemáticos.

8.2.1 FUNCIONAMIENTO DE IDEA [8].

International Data Encryption Algorithm realiza el encriptamiento de un texto fuente sometiendo cada uno de sus bloques a ocho etapas idénticas y sucesivas de encriptamiento. Con este objetivo toma

bloques de texto fuente de 64 bits de longitud y devuelve un texto cifrado de igual longitud. Cada bloque de texto fuente es dividido en 4 sub-bloques de 16 bits. Para poder explicar claramente el funcionamiento del algoritmo se denotarán estos bloques como X_1 , X_2 , X_3, y X_4 . También la clave de encriptamiento se particiona en sub-claves de 16 bits cada una , las cuales se denotará como: K_1 , K_2 , K_3 , K_4 , K_5 , K_6 , K_7 , K_8 .

El encriptamiento de cada bloque de información se lleva a cabo mediante la operación de 8 etapas sucesivas de encriptamiento, cada una de las cuales es idéntica a las otras. La salida de cada etapa produce cuatro sub-bloques de información de 16 bits cada uno los cuales serán denotados SB_1 , SB_2 , SB_3, y SB_4 Entre etapa y etapa de encriptamiento se intercambian dos sub-bloques de información SB_2 y SB_3y con estas entradas intercambiadas se arregla la entrada para la siguiente etapa con los cuatro sub-bloques de 16 bits. Esto puede ser mejor comprendido con ayuda de la siguiente figura:

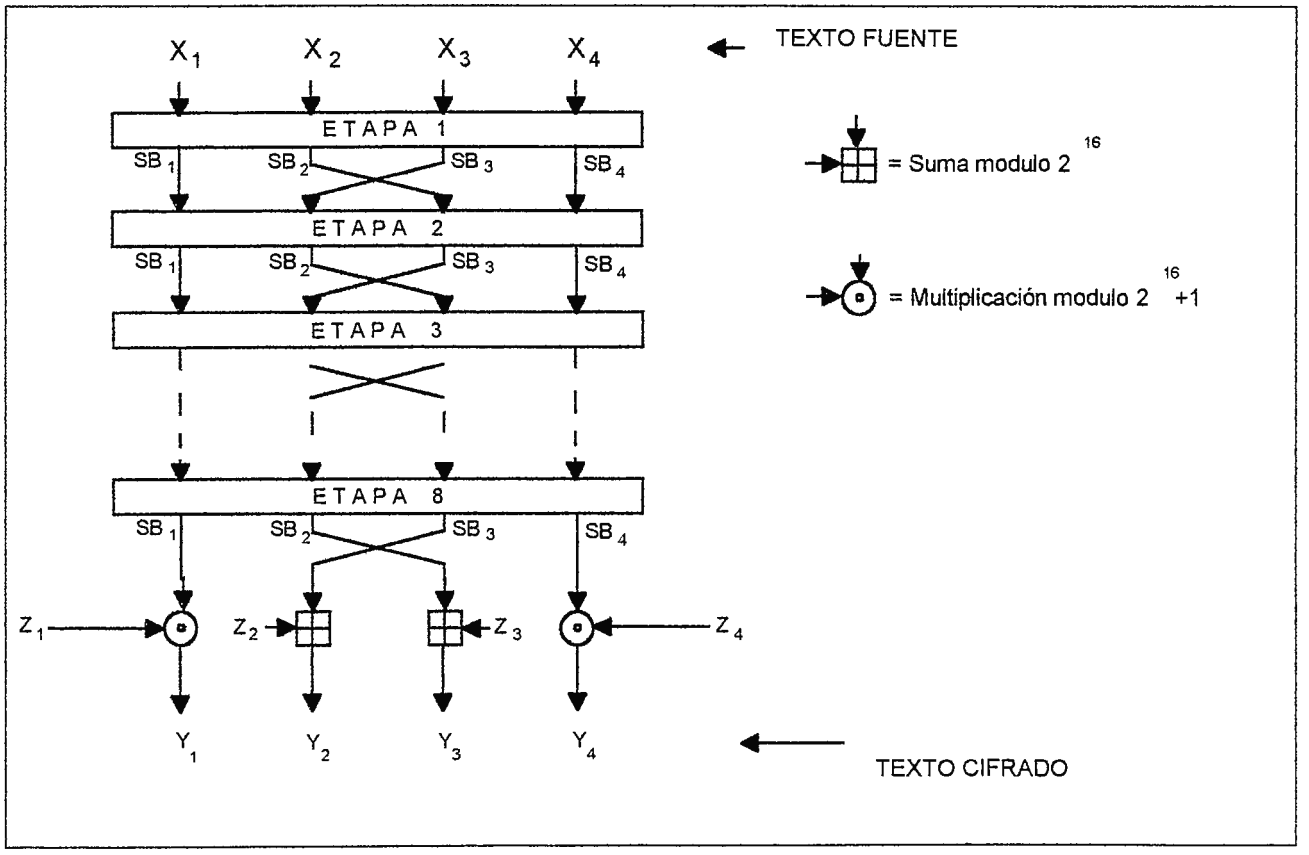


FIGURA 8.2.1

En cada una de las etapas se desarrolla una secuencia definida de operaciones, las cuales se listan a continuación:

1. Se efectúa la operación: $(SB_1 * Z_1)$ modulo $(2^{16} + 1)$ el resultado se guarda en la variable SB_1

   $SB_1 = SB_1 \boxplus Z_1$
2. Se opera: $(SB_2 + Z_2)$ modulo (2^{16}) el resultado se guarda en la variable SB_2

   $SB_2 = SB_2 \odot Z_2$
3. Se realiza la operación: $(SB_3 + Z_3)$ modulo (2^{16}) el resultado se guarda en la variable SB_3

   $SB_3 = SB_3 \odot Z_3$
4. Se opera: $(SB_4 * Z_4)$ modulo $(2^{16} + 1)$ el resultado se guarda en la variable SB_4

   $SB_4 = SB_4 \boxplus Z_4$
5. Se efectúa la operación: $OR_{13} = (SB_1 \oplus SB_3)$
6. Se efectúa la operación: $OR_{24} = (SB_2 \oplus SB_4)$
7. Se realiza la operación: $(OR_{13} * Z_5)$ modulo $(2^{16} + 1)$ el resultado se guarda en la variable ORZ_{135} $ORK_{135} = OR_{13} \boxplus Z_5$
8. Se opera: $(OR_{24} + ORZ_{135})$ modulo (2^{16}) el resultado se guarda en la variable S_{12345}

   $S_{12345} = OR_{24} \odot ORZ_{135}$
9. Se efectúa la operación: $(S_{12345} * Z_6)$ modulo $(2^{16} + 1)$ el resultado se guarda en la variable M_6 $M_6 = S_{12345} \boxplus Z_6$
10. Se opera: $(M_6 + ORZ_{135})$ modulo (2^{16}) el resultado se guarda en la variable F.

    $F = M_6 \odot ORZ_{135}$
11. Se realiza la operación: $SB_1 = (SB_1 \oplus M_6)$
12. Se efectúa la operación: $SB_3 = (SB_3 \oplus M_6)$
13. Se realiza la operación: $SB_2 = (SB_2 \oplus F)$
14. Se efectúa la operación: $SB_4 = (SB_4 \oplus F)$

Como se puede apreciar, el inicio del proceso de encriptamiento de cada etapa exige como entrada inicial los valores de: SB_1 , SB_2 , SB_3, y SB_4 , y arroja como salida valores para las mismas variables SB_1 , SB_2 , SB_3, y SB_4. Es importante recordar el intercambio a realizar entre SB_2 y SB_3 al final de cada etapa. El procedimiento anterior puede ser expresado gráficamente mediante la siguiente figura:

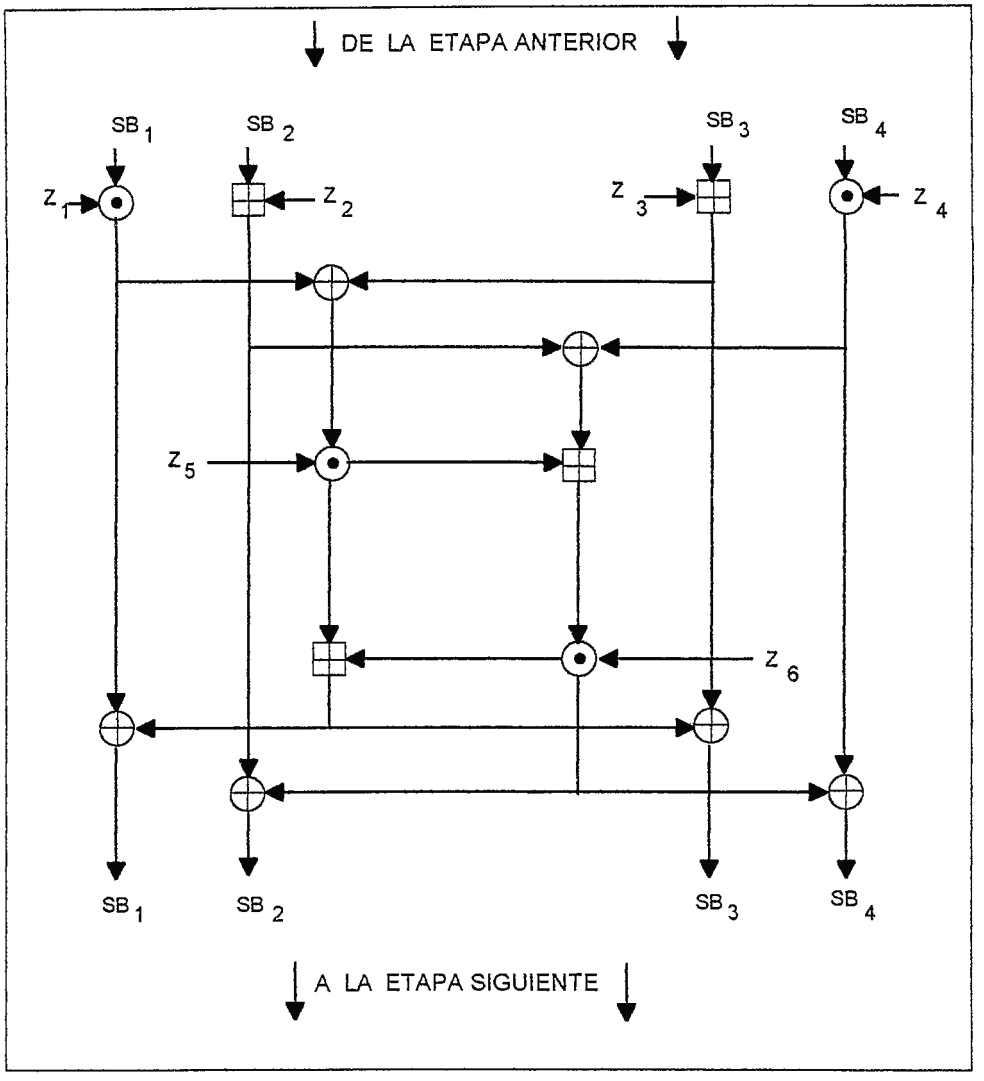


FIGURA 8. 2. 2

Al poner atención a el diagrama de encriptamiento se puede apreciar los parámetros Z_1 , Z_2 , Z_3 y Z_4. Estos términos, al igual que Z_5 y Z_6 los cuales se muestran en el desarrollo interno de cada etapa, proceden de la clave de encriptamiento, sus valores no son constantes a lo largo de todo el proceso, puesto que la clave va siendo rotada y fraccionada conforme se desarrolla el algoritmo, y es de cada una de las fracciones de la clave que los parámetros Z toman su valor para cada una de las etapas de encriptamiento.

A continuación se muestra el proceso de asignación de valores a Z_1 , Z_2 , Z_3 , Z_4 , Z_5 y Z_6 a lo largo del proceso de encriptamiento de un bloque de información.

ETAPA	Bits Rotados	VALORES DE LOS PARAMETROS Z_1	Z_2	Z_3	Z_4	Z_5	Z_6
1	0	K_1	K_2	K_3	K_4	K_5	K_6
2	0 / 25	K_7	K_8	// K_1	K_2	K_3	K_4
3	25 / 50	K_5	K_6	K_7	K_8	// K_1	K_2
4	50	K_3	K_4	K_5	K_6	K_7	K_8
5	75	K_1	K_2	K_3	K_4	K_5	K_6
6	75 / 100	K_7	K_8	// K_1	K_2	K_3	K_4
7	100 / 125	K_5	K_6	K_7	K_8	// K_1	K_2
8	125	K_3	K_4	K_5	K_6	K_7	K_8
Salida	150	K_1	K_2	K_3	K_4		
Siguiente bloque	150 /175	K_5	K_6	K_7	K_8	// K_1	K_2

Tabla 8.2.1

IDEA puede ser implementado en cualquiera de los modos de operación listados en el capítulo 5 lo cual multiplica las dificultades para desencriptarlo, además su clave de 128 bits genera un número muy grande de posibilidades de encriptamiento.

Si IDEA fuera atacado por fuerza bruta, es decir probando todas las distintas claves de encriptamiento, se necesitarían examinar $2^{128} = 3.4*10^{38}$ posibilidades distintas, y si en cada intento de prueba de clave se demoran $50*10^{-6}$ segundos, el probar todas las claves demoraría $1.7*10^{34}$ segundos, esto es $5.39*10^{26}$ años. Si a esto se le añade uno de los modos de encriptamiento en los cuales el elemento encriptador evoluciona, las variedades de los parámetros de encriptamiento se multiplicarían por cuatro, y con las 2^{64} posibles variedades de vector de inicialización el algoritmo se volvería mucho más difícil de romper por la fuerza bruta.

Este tipo de algoritmo, dependiendo de la máquina usada para su implementación, puede ser en ocasiones un poco más lento que el DES, ya que el este último sólo realiza operaciones con bits, las cuales pueden llegar a necesitar solamente 1 o hasta 5 ciclos de reloj, en cambio IDEA, por realizar operaciones matemáticas de suma, multiplicación y función módulo puede tardarse un poco más en realizar el encriptamiento de un sólo bloque, sin embargo esta diferencia se hace pequeña debido a que IDEA tiene menos etapas en cada bloque.

8.3 GENERALIDADES DE OTROS ALGORITMOS SIMÉTRICOS.

Existen muchos más algoritmos simétricos de encriptamiento, la explicación detallada de su funcionamiento podría ser de suma utilidad para su total comprensión, sin embargo el objetivo de este documento no es presentar un catálogo detallado de algoritmos. Por este motivo se listarán a continuación otros algoritmos de relativa importancia en el contexto de internet actual, sin profundizar demasiado en el funcionamiento de cada uno de ellos.

Téngase en consideración que todos los algoritmos siguientes son simétricos, es decir, utilizan un mismo algoritmo para encriptar y desencriptar, operan cada bloque de datos de texto fuente en forma idéntica, modificando los parámetros de la clave en cada uno de los bloques. Además, a semejanza de los anteriores, todos ellos utilizan varias etapas idénticas de encriptamiento en cada bloque para lograr su objetivo.

8.3.1 SKIP JACK [8]

Este es un algoritmo secreto desarrollado por NSA (National Security Agency) para ser implementado mediante Hardware, utilizando los circuitos integrados Cliper[14] y Capstone[15] , Su funcionamiento en detalle es completamente secreto. Tan sólo se sabe que es un algoritmo de encriptamiento de bloques de 64 bits, que utiliza una clave de 80 bits y opera en 32 etapas de encriptamiento en cada bloque. Puede ser usado en los modos ECB, CBC, 64bit OFB, y 1,8,16,32,o 64 bits en modo CFB.

8.3.2 GOST [8]

Su nombre es un acrónimo de " Gosudarstvennyi Standar Soyuza " , o Estándar del Gobierno. Creado por el gobierno soviético Ruso para sus comunicaciones militares, aunque actualmente su uso civil ha sido autorizado.

GOST es un algoritmo cifrador de bloques de 64 bits el cual utiliza una clave de 256 bits. Opera cada bloque mediante 32 etapas de encriptamiento, en cada una de las cuales se desarrolla el siguiente procedimiento:

- Se divide el bloque en dos semibloques de 32 bits.

[14] Cliper, También conocido como MYK-78T, es un circuito integrado para encriptamiento de trama, especialmente diseñado para encriptamiento de la voz digitalizada, puede obtenerse en el telefono de seguridad AT&T modelo 3600.

[15] Capstone, también conocido como MYK-80 es un circuito integrado para encriptamiento capaz de realizar cualquiera de los 4 modos de encriptamiento expuestos en el capítulo 5, además de generar DSS, SHA,KEA, y ser capaz de producir números aleatorios a partir del ruido que genera el equipo en algunos de sus dispositivos. Se puede obtener este IC en la tarjeta para encriptamiento llamada FORTEZZA ™ (Creada por Tessera Inc.).

- El medio bloque derecho se cambia al lado izquierdo.
- El medio bloque derecho se suma en módulo 16 con una subclave (tomada de los 8 segmentos de 32 bits en que puede ser descompuesta la clave) la cual es escogida según el número de etapa que se esté desarrollando[16].
- Este medio bloque es dividido en 8 números de 4 bits, y cada uno es sustituido de acuerdo a lo ordenado por 8 tablas de sustitución, una para cada número[17].
- El medio bloque anterior es rotado 11 bits hacia la izquierda.
- El medio bloque rotado es operado en función EXOR con el medio bloque izquierdo del principio de la etapa.
- El resultado de todo esto se toma como medio bloque derecho.

Es posible que de este lado del mundo GOST sea muy poco usado, pero eso no hace más que incrementar el grado de seguridad del mismo, ya que la cantidad de personas que conoce de su existencia es mucho más reducida.

8.3.3 BLOWFISH [8]

Blowfish es un algoritmo de encriptamiento que ha alcanzado un relativo grado de éxito comercial, el cual encripta bloques de 64 bits usando una clave de 576 bits, la cual para un manejo más sencillo es dividida en 18 elementos de clave, cada uno de 32 bits[18]. El proceso en si tiene 15 etapas idénticas de encriptamiento y una etapa final. En cada una de las primeras se desarrolla el siguiente procedimiento:

- El bloque de 64 bits es dividido en dos de 32.
- El semibloque izquierdo es operado en función EXOR con uno de los 18 elementos de clave de 32 bits.
- El semibloque anterior es considerado en adelante como el derecho.
- El semibloque anterior es sometido a una función F que lo transforma, este bloque transformado es operado en función EXOR con el semibloque derecho del principio de la etapa y es considerado en adelante como semibloque izquierdo.
- El procedimiento anterior se repite 15 veces.

---

[16] El orden para tomar las subclaves es: 1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8,8,7,6,5,4,3,2,1.

[17] Cada tabla de sustitución ordena un número sustituto para cada cifra, desde 0000 hasta 1111.

[18] Estos elementos de clave pueden ser reducidos a nueve elementos de clave, los dos repetidos, o a seis elementos de clave repetidos tres veces, o a tres elementos de clave repetidos seis. De manera que la clave puede ser desde 32 bits hasta 576 bits. Por supuesto, el usar elementos de clave no repetidos incrementa la seguridad del algoritmo.

En la última etapa no se intercambian los bloques, sino que el izquierdo es operado en EXOR con el elemento 18 de la clave, y el derecho con el elemento 17 de la clave. La sucesión de los dos semibloques forma el bloque encriptado.

La función transformadora F utiliza 4 tablas de sustitución que transforman cada número de 8 bits (de los cuatro que pueden ser obtenidos en un bloque de 32) , en números de 32 bits, los correspondientes a el primero y el segundo números son operados en suma módulo 2^{16} , el resultado de esto es operado en EXOR con la sustitución del tercer número, y por último este resultado es operado en suma módulo 16 con la sustitución del cuarto número.

La ventaja de este algoritmo es su rapidez, debido a que las transformaciones son de orden muy sencillo y pueden ser desarrolladas en 26 ciclos de reloj por cada byte, o menos. Ademas, su sencillez hace que el archivo ejecutable básico tenga menos de 5 KB. Y la clave variable lo hace fácil de manipular, pues puede ser tan simple como 32 bits (repetidos 18 veces) , hasta 578 bits en 2^{576} ordenaciones distintas. todas estas ventajas le dan un amplio grado de popularidad, tanto por el fácil manejo de la clave como por la seguridad que brinda cuando el tamaño de la clave es pleno.

9.- ENCRIPTAMIENTO COMO MEDIO DE SEGURIDAD PARA LAS TAREAS CONJUNTAS.

Cuando la información es de vital importancia para el trabajo en equipo, ya sea entre distintas personas u organizaciones que comparten información para la toma de decisiones compartidas, o la ejecución de labores delicadas e importantes, el encriptamiento puede ser de invaluable ayuda para que las decisiones tomadas no sean saboteadas o manipuladas por unos pocos miembros del equipo de trabajo o por personas ajenas a la organización ejecutiva.

Para ilustrar mejor este tipo de tareas que se quieren dar a entender, se expondrán a continuación algunos ejemplos de situaciones concretas en que la mutua certificación de los participantes en la labor del equipo es crucial para la seguridad:

EJEMPLO 1:

La asignación a los herederos de una cuantiosa fortuna se verifica con la lectura del testamento por un notario certificado. No obstante por deseo de el testamentario, el documento deberá ser dividido en partes, cada una de las cuales deberá ser ilegible sin la combinación conjunta de todas las partes en que ha debido ser dividido el documento, de manera que si no se tienen todas las partes del documento, este no podrá ser conocido por nadie, y además, si alguna de las partes ha sido alterada, todo el documento no podrá ser leído por faltar alguna de sus partes integrales. De esta manera el testamentario se asegura de que ninguno de sus notarios alterará el contenido del testamento, y que éste será leído en forma íntegra el día de se proclamación legal.

EJEMPLO 2:

En un país en guerra, el mando para operar los misiles nucleares programados para ser lanzados hacia varias ciudades enemigas no puede ser responsabilidad de un solo oficial. Pues esta persona podría ser engañada por un enemigo común a ambos bandos para que los misiles sean lanzados sin responsabilidad de este tercer involucrado. También podrían ser lanzados en un momento de desesperación por un atentado

personal contra el militar responsable de los misiles. Los resultados de una mala decisión redundarían en una guerra nuclear que destruiría gran parte de la humanidad. Un recurso para evitar estas acciones imprudentes podría ser un mecanismo que para lanzar los misiles necesite la reunión de un determinado número de autorizaciones certificadas (no todas las autorizaciones, pues en tiempo de guerra algunos de los portadores podrían morir antes de emitirla), las cuales deben ser secretas, conocidas solamente por la persona con voto en la toma de decisiones de esta naturaleza, y que dicha autorización pueda ser transportada por un medio de comunicación seguro y en forma rápida hasta el centro de lanzamiento.

Labores administrativas de personal como las citadas anteriormente han obligado a que las técnicas de encriptamiento desarrollen procedimientos donde se involucran más de dos participantes para la puesta en operación de un dispositivo, o el encriptamiento y desencriptamiento de un documento clasificado. A continuación se presentan algunos procedimientos en los cuales las técnicas de encriptamiento se han destacado en la administración segura de operaciones que involucran tareas conjuntas.

9.1 CREACION DE CLAVES DE ENCRIPTAMIENTO MULTIPLES POR MEDIO DE LA FUNCION EXOR.

Como ya se explicó en el capítulo 4, un mensaje M redactado en código digital de bits puede ser encriptado por medio de la operación en función EXOR con otro parámetro K que será la clave de encriptamiento, el resultado puede expresarse como:

$$C = M \oplus K$$

$$M = C \oplus K$$

Si al realizar esta operación de encriptamiento se tomaran más de una clave para ser operada en función EXOR con el mensaje, entonces se tendría el mensaje cifrado C como resultado de la siguiente operación:

$$C = M \oplus K_1 \oplus K_2 \oplus K_3 \oplus K_4 \oplus .....$$

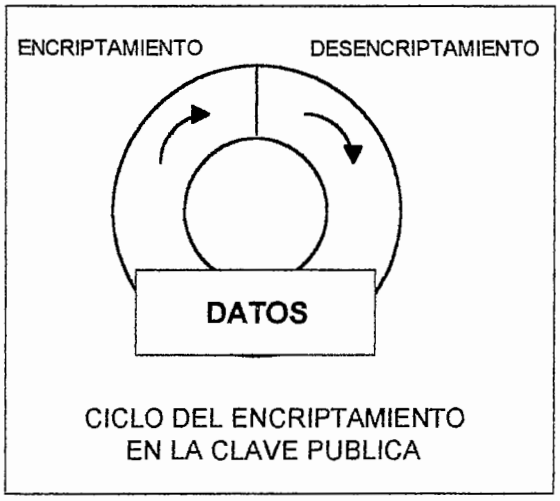
Si se agrupan todas las claves K_i para formar un sólo número binario especifico K , se comprenderá con claridad que la única operación realizada ha sido la descomposición de la clave en todos sus términos EXOR que la constituyen, pero dicha clave nunca estará completa si no tiene todos sus términos, de manera que el mensaje M sólo podrá ser desencriptado cuando se operen todos los términos de la clave como se muestra a continuación:

$$M = K_1 \oplus K_2 \oplus K_3 \oplus ..... K_n \oplus C$$

En este tipo de encriptamiento con clave múltiple, son absolutamente necesarias todas las partes para poder reconocer el dato contenido en el mensaje, cada sector o persona participante en el proceso puede tener una de las claves K_i y uno de ellos puede tener el elemento C. La forma de repartición de los elementos no es relevante para el correcto desarrollo del proceso, ya que tanto los elementos K_i como el elemento C tienen la misma longitud y la misma ilegibilidad, y ninguno es más importante que los otros ya que todos son absolutamente necesarios.

9.2 ENCRIPTAMIENTO EN CLAVE PÚBLICA CON MULTIPLES CLAVES PARA ENCRIPTAMIENTO Y EL DESENCRIPTAMIENTO.

El encriptamiento en clave pública, tal como se ha explicado anteriormente, no puede ser desencriptado con la misma clave con que se encriptó, ya que ésta resulta verdaderamente inútil para dar alguna pista importante que lleve a su desencriptamiento por parte de un criptoanalista.



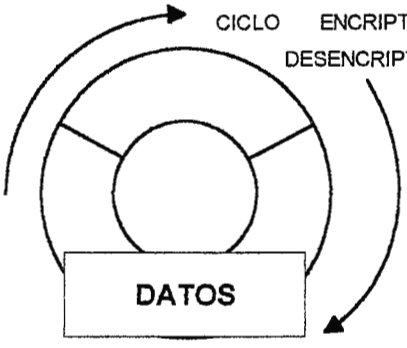
CICLO DEL ENCRIPTAMIENTO EN LA CLAVE PUBLICA

FIGURA 9.1

A diferencia de el encriptamiento en clave secreta, cuyo algoritmo puede ser entendido como un procedimiento reversible capaz de ser desarrollado hacia adelante y hacia atrás, El algoritmo de clave pública sólo puede ser desarrollado en un sentido. Por esta razón el procedimiento de encriptamiento no puede ser retrodesarrollado, sino que tiene que seguir su procedimiento establecido para llegar al texto desencriptado.

El encriptamiento en clave pública opera los datos del texto fuente en un ciclo completo de encriptamiento- desencriptamiento. Terminado el ciclo se tiene la misma información en el mismo formato que se tenía en el momento de iniciar. Cuando el ciclo de encriptamiento no está completado, al recoger la

información a medio camino del ciclo no se obtiene la información legible, sino que se obtiene una información con algún grado de encriptamiento.



CICLO DE TRES SEMENTOS DE ENCRIPTAMIENTO EN LA CLAVE PUBLICA

Si el ciclo de encriptamiento-desencriptamiento fuera fraccionado en tres segmentos como muestra la figura 9.2 , al final de cada segmento se obtendría la información que ha sido encriptada, pero con algún grado de encriptamiento. Solamente al final del desarrollo total del proceso podrá ser legible la información.

FIGURA 9.2

En una aplicación real del anterior ciclo de encriptamiento con tres segmentos, a cada participante del proceso de encriptamiento se le otorgaría una clave pública (correspondiente a un segmento) de las tres claves posibles (correspondientes a los tres segmentos). Esto obligaría a los poseedores de la claves a reunirse para hacer legible cualquier tipo de mensajes.

Las opciones para el encriptamiento y desencriptamiento utilizando tres claves de encriptamiento brindan múltiples posibilidades para manejar información cifrada entre los participantes de sistema. Las características de este tipo de encriptamiento obligarían a que en toda comunicación se hicieran presentes las tres personas para que ésta se lleve a cabo, ya sea que uno envíe y dos reciban, o que dos envíen y uno reciba. Esto puede ser comprendido con mayor claridad observando la siguiente tabla:

CLAVES USADAS PARA ENCRIPTAR	CLAVES USADAS PARA DESENCRIPTAR
K 1	K 2 Y K 3
K 2	K 1 Y K 3
K 3	K 1 Y K 2
K 1 Y K 2	K 3
K 1 Y K 3	K 2
K 2 Y K 3	K 1

TABLA 9.1

Se entregan tres claves públicas:

K 1

K 2

K 3

Para tres participantes:

A

B

C

Se comprenderá con claridad que todo mensaje será siempre conocido por los tres participantes A, B, y C, ya que nunca se podrá excluir a uno de los tres para entablar cualquier comunicación.

9.3 PARTICIPACION PARCIAL (N PARTES DE UN TOTAL DE M PARA FORMAR EL DOCUMENTO SECRETO)

Si no se desea que todas las partes sean indispensables para formar el dato secreto, puede arreglarse un método distinto a los anteriores para que los participantes del secreto sólo necesiten m partes de las n totales para formar el dato secreto.

Si se desea excluir a un participante como persona indispensable, ya sea porque existe la probabilidad de que no asista a la sesión de revelación del secreto, o bien por otro motivo, puede dársele un dato que en nada afecta a el encriptamiento y desencriptamiento de la información. Si se le da una clave que contenga el número binario 000000000.... , cualquier dato a encriptar que se opere en función EXOR con ese número producirá el mismo dato fuente. También puede asignarse una clave que contenga el número binario 1111111111111..... , de manera que cualquier dato a encriptar que se opere en función EXOR con ese número producirá el mismo dato fuente pero negado. (el sistema de seguridad puede ser alertado en este sentido, para que ante la falta de presencia de la clave nula presente el negado del resultado que se produce al aplicar sólo las claves valederas).

Para que la persona a quien se le asigna el número de clave nula no se de cuenta que no se le ha tomado en cuenta, todas las claves son encriptadas en cualquier forma para que nadie sepa en realidad cuál número binario opera como su clave, ya que este número está encriptado y sólo el sistema de seguridad puede desencriptarlo.

De esta manera, el mensaje podrá ser reconstruido aunque no asistan a su construcción todas las personas a quienes se les ha dado una clave para descifrarlo.

10- GENERACIÓN Y NEGOCIACIÓN DE CLAVES:

Al recurrir a la clave secreta para transmitir un mensaje es necesario tener en cuenta que de utilizar durante mucho tiempo la clave de encriptamiento, ésta puede ser conocida y descifrada por criptoanalistas experimentados, y ser utilizada para falsificar mensajes o simplemente para tener acceso a toda la información transmitida. Una manera para evitar este tipo de ataques consiste en cambiar la clave secreta en forma periódica. Esto dificulta la labor del criptoanalista al darle muy poco tiempo para encontrar la clave por fuerza bruta [19].

Este capítulo presenta una serie de procedimientos para facilitar el acuerdo periódico de una clave de encriptamiento utilizada por el emisor y receptor de la información. Esto no sólo incluye clave secreta, sino también clave privada y pública, ya que la autenticación de los mensajes es importante para verificar que se está intercambiando información con la persona correcta.

10.1 GENERACION DE CLAVES PUBLICAS Y PRIVADAS:

Cuando la clave de encriptamiento es adjudicada directamente por los usuarios cada vez que ellos lo consideren conveniente, pueden escoger dos formas para hacerlo:

- GENERACIÓN DE CLAVES POR COMUN ACUERDO: cuando los participantes en la comunicación encriptada escogen una clave secreta ya sea por moción de uno de ellos o porque tienen una sucesión de claves[20]. Si se trata de claves públicas, el interesado se encarga de publicar su clave entre todos los participantes en la comunicación.

[19] Esto es, probando todas las combinaciones posibles de los bists de la clave hasta encontrar la adecuada.

[20] Una sucesión sencilla de claves puede ser el uso de una clave distinta para cada día de la semana, una mucho más complicada el uso de claves matemáticamente calculadas por todos los integrantes del sistema de comunicación cada hora o cada media hora.

- GENERACION DE CLAVE COMPLETAMENTE ALEATORIA: Cuando la clave, ya sea pública o secreta, es generada por métodos de generación de números aleatorios[21] . Este procedimiento tiene la ventaja de ser menos fácil de adivinar, ya que el anterior puede ser encontrado probando claves que tengan algún significado, como palabras, nombres, o números que signifiquen algo para los participantes en la comunicación.

En ambos casos se tiene un nuevo problema en el acuerdo mutuo de la clave: hacerla llegar de un participante a otro con la total seguridad que este parámetro no va a ser conocido por personas no deseadas en la comunicación.

Sería sumamente ingenuo mandar el dato de clave por medio del correo normal, de telegramas, del correo electrónico desprotegido, o por medio del token de la red, esto equivaldría a tirar la llave del candado a la calle para que el otro participante la recoja, con el consiguiente riesgo que el delincuente la tome para si, la copie y la tire nuevamente para hacer creer que nadie la ha visto. De nada sirve asegurar con extrema solidez todas las puertas de entrada de un banco cuando las ventanas y el techo son endebles y fáciles de atravesar, y peor aún, si pueden ser fácilmente utilizadas como entrada y salida sin dejar rastro de la intromisión.

Diferentes medios seguros para hacer llegar la clave van desde las sesiones confidenciales, la tinta invisible, las tarjetas en código secreto, los sobres de seguridad con acuse de violación, la multitud de telegramas, cada uno con un fragmento de la clave, hasta la descomposición de la clave en términos EXOR para ser ensamblada por el receptor al operar todos los términos.

No obstante, en algunas ocasiones es necesario no sólo enviar la clave por un medio seguro, sino también saber si el otro participante ha recibido el mensaje. Para llevar a cabo este reconocimiento de que el otro ha recibido el mensaje es necesario ir un paso más allá de la distribución: se debe desarrollar una negociación completa de los parámetros a utilizar para establecer la comunicación.

[21] Recuérdese que no existen los generadores completamente aleatorios, existen generadores pseudoaleatorios que estadísticamente se aproximan un poco a la total aleatoriedad, pero que no la alcanzan a plenitud. Para efectos de encriptamiento y seguridad se debe escoger un generador de números aleatorios que sea lo más amplio posible en el rango de valores a producir, ademas de repetir sus valores en un ciclo lo más grande posible.

10.2 NEGOCIACION DE CLAVES.

Cuando no se dispone de cercanía física que permita acordar claves en sesiones confidenciales, se pueden asignar claves por medio de los canales mismos de comunicación, en este caso, de los enlaces mismos de la red. Esto involucra el establecimiento de procedimientos específicos que deben ser seguidos por cualquiera de los usuarios cuando se intente establecer una comunicación encriptada. A continuación se explican algunos de estos procedimientos:

- ASIGNACION DE CLAVES SECRETAS POR MEDIO DE UN ARBITRADOR:

En este caso cualquiera de los dos participantes recurre a un centro de asignación y distribución de claves (KDC : Key Distribution Center) para que actúe como arbitrador[22] y le sea asignada una clave, pública o privada, para realizar la sesión. Se recurre a un arbitrador para tener la seguridad de que el emisor y el receptor son realmente quienes dicen ser, pues el arbitrador puede reconocerlos y asignar claves que sólo los legítimos participantes pueden descifrar.

En caso de utilizar solamente claves secretas, se puede seguir el siguiente procedimiento:

(1) {A} llama al arbitrador y le pide una clave para comunicarse con {B}.

(2) El arbitrador genera una clave aleatoria K y encripta dos copias de dicha clave:

-Una para {B} encriptada en la clave secreta que comparten {B} y él mismo.

-Una para {A} encriptada en la clave secreta que comparten {A} y él mismo.

El arbitrador envía ambos mensajes a {A}.

(3) {A} desencripta el mensaje del arbitrador para asumir la clave K asignada[23].

(4) {A} envía a {B} una petición de comunicación y la clave de encriptamiento K encriptada con la clave que {B} comparte con el arbitrador.

(5) {B} desencripta el mensaje del arbitrador para asumir la clave asignada[24].

(6) {A} y {B} usan la clave para comunicarse con seguridad.

En el caso de contar con claves públicas los dos participantes en la comunicación, puede acortarse el procedimiento a los siguientes pasos:

---

[22] Recuérdese que el arbitrador es una tercera persona en la cual los participantes en la comunicación confían a ojos cerrados, vease el capítulo 4 como referencia para una idea más clara del concepto de entidad arbitradora.

[23] {A} sabe que proviene del arbitrador porque sólo él conoce esta clave de encriptamiento.

[24] {B} sabe que proviene de el arbitrador porque sólo él conoce esta clave de encriptamiento que con él comparte, y además sabe que el que pide la comunicación es {A} porque sólo el arbitrador pudo enviarle la misma clave a {A} encriptada en su respectiva clave para que {A} se la envíe a {B}, y como las claves coinciden, se comprueba la identidad de cada uno de los tres

(1) {A} obtiene la clave pública de {B} por medio de el Arbitrador.

(2) {A} genera una clave secreta de encriptamiento K, encripta esta clave utilizando la clave pública de {B} y envía este mensaje encriptado a {B}[25].

(3) {B} desencripta el mensaje de {A} utilizando su clave privada [26]

(4) {A} y {B} inician una sesión de comunicación segura por medio de la clave secreta generada.

- ASIGNACION DE CLAVES SECRETAS CON AYUDA DE FIRMAS DIGITALES:

Se puede ir un paso más allá en cuanto a seguridad utilizando firmas digitales para verificar el origen del mensaje sin ayuda de la certificación de un arbitrador, el cual puede se manipulado por falta de sistemas seguridad propios, o incluso falsificado en sus gestiones con sus clientes por un interceptor impostor. El siguiente procedimiento ofrece una opción muy segura contra la posibilidad anterior:

(1) {A} genera en forma aleatoria una clave K para el encriptamiento en clave secreta. Luego encripta el mensaje que tiene para enviar con dicha clave.

$$E_k (M)$$

(2) {A} obtiene la clave pública de {B} de la base de datos del Arbitrador.

(3) {A} encripta la clave que generó aleatoriamente con la clave pública de {B}.

$$E_B (K)$$

(4) {A} envía a {B} el mensaje encriptado y la clave de encriptamiento cifrada con la clave pública de {B}, todo esto autenticado con su firma digital S_A.

$$S_A (E_k(M), E_B (K))$$

(5) {B} desencripta el valor de la clave de encriptamiento K usando su clave privada.

(6) {B} desencripta el mensaje M usando la clave K.

Como puede deducirse, nadie más que {B} puede ver el mensaje, porque sólo {B} puede desencriptar la clave K, y además, {B} está seguro que el mensaje proviene de {A}, tanto la clave como el mensaje, ya que nadie más lo puede autenticar con su firma digital.

[25] {A} sabe que si inicia una comunicación con {B} utilizando dicha clave es porque el receptor verdaderamente es {B}, y además comprueba la identidad del arbitrador ya que solo él pudo haberle dado la clave pública correcta

[26] {B} sabe que {A} no pudo haber obtenido su clave pública de nadie más que del arbitrador, y que éste sólo la pudo haber dado a las personas que él ordeno que se la diera.

11.- PROCEDIMIENTOS PARA AUTENTICAR INFORMACIÓN DIGITAL:

El objetivo de la autenticación digital puede ser expresado con pocas palabras: Demostrar con certeza que quien solicita o envía una comunicación es realmente quien dice ser.

Reconocer el origen auténtico de un documento o de una comunicación no es fácil para quien recibe el mensaje. Sin embargo es fácil demostrar la propia identidad cuando se comparte una información secreta exclusiva con la persona que recibe el mensaje; el cuestionamiento de la información secreta contestada con la respuesta correcta, es suficiente prueba de identidad. Y cuando no se comparte un secreto, es posible compartir la confianza en una tercera persona que pueda autenticar los mensajes por el conocimiento de información secreta de las personas que pidan su arbitración, información que será cuestionada en el momento de certificar la identidad de los solicitantes.

A continuación se presentan varios ejemplos de autenticación digital en los cuales el encriptamiento es de vital importancia para el reconocimiento seguro de las personas o sistemas involucrados en una transferencia de datos.

11.1 AUTENTICACIÓN POR CONTRASEÑA.

En este sistema de autenticación, la identidad de la persona se verifica por su nombre y su contraseña. Es la forma de autenticación más sencilla de autenticación que se conoce.

No obstante es necesario proteger la lista de usuarios y contraseñas dentro del sistema de reconocimiento, ya que una vez que los datos se encuentren dentro del sistema, el programa reconocedor podría ser atacado por alguno de sus usuarios o por un intruso que haya conseguido entrar y robar la lista de usuarios con sus respectivas contraseñas. Esta protección se puede llevar a cabo utilizando técnicas de encriptamiento, guardando valores Hash de los datos en lugar de los datos mismos. De esta manera se va a poder llevar a cabo el reconocimiento comparando valores Hash de la base de datos con los que son obtenidos por medio de la misma función Hash al operar los datos de reconocimiento (nombre de usuario y password) en el momento en que estos son proporcionados por el solicitante.

11.2 AUTENTICACION POR FUNCION SKEY

En este tipo de autenticación se genera un número binario a partir de una clave de entrada, la cual es operada por una función Hash de un sólo sentido, y con este número generado se verifica la identidad del cliente al compararlo con el valor almacenado como su identidad.

Cada vez que se solicita una entrada al sistema, se pide la clave de entrada para la sesión actual; si el valor generado por la función de un sólo sentido a partir de esta clave de entrada corresponde con el valor que está almacenado en memoria; se verifica la autenticación y se asigna el valor obtenido por la función Hash como clave de entrada para la siguiente sesión. Este valor de clave de entrada no se almacena para la próxima sesión, sino que se opera nuevamente por la función Hash de un sólo sentido y el resultado es el que se guarda, de igual manera el proceso se repite en la siguiente sesión. El diagrama de flujo mostrado a continuación expone en forma gráfica la manera en que se desarrolla este procedimiento cada vez que se solicita entrada al sistema:

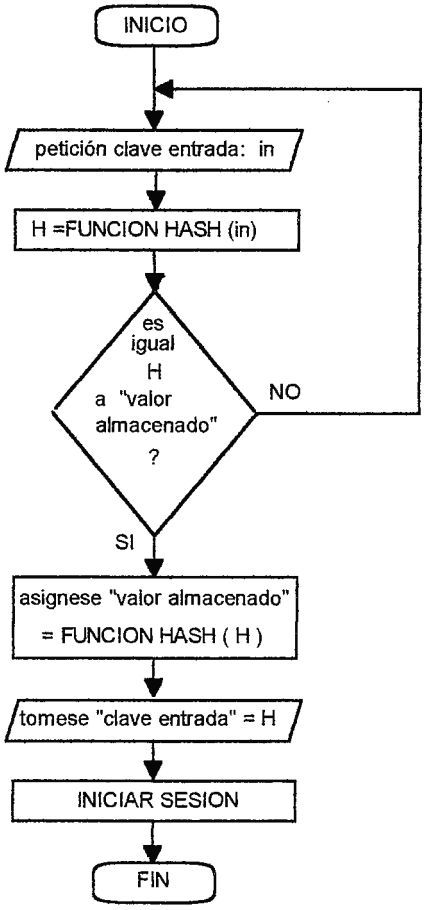


FIGURA 11.1

Nótese que cada clave es distinta para cada sesión, y cada sesión asigna una clave de entrada para la próxima sesión, para que sea guardada por el usuario y pueda tener acceso al sistema la próxima vez que lo intente.

11.3 AUTENTICACION POR ENCRIPTAMIENTO EN CLAVE PUBLICA.

Este tipo de reconocimiento verifica la identidad de la persona que solicita acceso por medio de la clave pública de desencriptamiento que publica la persona interesada en emitir mensajes. Para este efecto lleva a cabo el siguiente procedimiento:

(1) {A} envía a {B} un número aleatorio encriptado con su clave privada.

(2) {B} envía a {A} otro número aleatorio.

(3) {A} opera ambos números aleatorios (el generado y el recibido) en una función específica y envía el resultado a {B}, encriptado con su clave privada.

(4) {B} desencripta el resultado con la clave publica de {A}, y lo compara con el resultado que ha obtenido operando los números. Si el resultado es igual, la identidad ha sido verificada.

La ventaja de este procedimiento sobre la simple clave pública de desencriptamiento es que la función de operación de los dos números solamente es conocida por {A} y {B}, además verifica que quien dice ser "{A}" verdaderamente puede encriptar con la clave privada de {A}, y que la primera transmisión en verdad fue un número encriptado en clave privada de {A}.

11.4 AUTENTICACION SKID.

Este tipo de autenticación se lleva a cabo utilizando exclusivamente algoritmos de clave secreta (como son las funciones MAC) y sin la participación de un arbitrador. El reconocimiento mutuo por parte de los dos participantes en la comunicación se lleva a cabo mediante una función código de autenticación de mensaje (Message Autentication Code: MAC) el cual es operado por cada participante sobre parámetros comunes a ambos, si el valor de la función es igual para ambos, entonces la identidad de ambos participantes es verificada. El procedimiento a desarrollar es el siguiente:

(1) {A} genera un número aleatorio R_A y lo envía a {B}.

(2) {B} genera un número aleatorio R_B, opera este número, seguido del número aleatorio de {A} y su nombre en la función código de autenticación de mensajes H_k, y envía este resultado junto con su

número aleatorio a {A}:

$$R_B , H_k (R_A , R_B , B)$$

(3) {A} realiza la función código de autenticación de mensajes H_k operando la sucesión de números aleatorios y el nombre de {B}, y la confronta con el resultado recibido. Si ambos resultados son iguales, {A} sabe que realmente se ha comunicado con {B}.

(4) {A} opera la función código de autenticación de mensajes H_k en el número aleatorio de {B} R_B junto con su nombre, y envía este mensaje este resultado a {B}:

$$H_k (R_B , A)$$

(5) {B} recibe este resultado pero lo verifica operando su número aleatorio y el nombre de {A} en la función código de autenticación de mensajes. Si el resultado concuerda, {B} sabe que realmente se está comunicando con {A}.

Una vez terminado todo el procedimiento anterior, la comunicación puede establecerse, con la completa seguridad que no hay impostores ni falsos mensajes en el enlace creado.

12.- NEGOCIACIÓN DE CLAVES CON AUTENTICACIÓN DE REMITENTE Y DESTINATARIO.

Cuando se desea que la transferencia de datos se realice en una forma segura, la clave secreta brinda una opción confiable y rápida [27], aunque presenta un inconveniente: debe ser cambiada constantemente para evitar fugas o intrusiones en la información que pueden ser producidas por un delincuente digital que ha tenido suficiente tiempo para encontrar la clave secreta.

El problema del cambio constante de la clave de encriptamiento es encontrar la manera de llegar a un acuerdo con los demás participantes de la comunicación encriptada. Es posible tener una sesión privada con todos los interesados para tomar este acuerdo, pero en el caso de no poder realizarla puede tomarse la decisión de aceptar que uno de ellos u otra persona sea quien decida qué clave se va a usar para determinado período de tiempo.

Sin embargo aún existe el problema de hacer llegar hasta el otro participante la clave de encriptamiento que se utilizará para un determinado grupo de sesiones de enlaces encriptados. Esta comunicación puede ser llevada a cabo por medios no muy seguros: tales como los telegramas, las llamadas telefónicas o el envío de un fax. Puede incrementarse en alguna medida la seguridad de estos medios de comunicación si se usa tinta invisible y sustitución de caracteres en el telegrama, esteganografía en las llamadas telefónicas, o algún tipo de encriptamiento en el fax. Y todavía es posible llegar un poco más lejos si se envía la clave en fracciones las cuales al combinarse regeneren la clave en su integridad (esta combinación puede ser llevada a cabo ya sea por simple concatenación de las fracciones, o bien mediante la suma de los valores numéricos que representan cada caracter de cada fracción[28], o bien mediante la operación en función EXOR de las facciones).

---

[27] La clave pública puede ofrecer una mejor protección con autenticación de remitente y destinatario incluida, pero puede ser mucho más lenta.

[28] Utilizando el método de Gilbert Vernam para el encriptamiento de los mensajes en los teletipos de principios de siglo. Vease algoritmos de encriptamiento en el capítulo 5.

Existe una tercera vía para tomar el acuerdo de la clave de encriptamiento: realizarlo mediante el uso del mismo medio que se pretende encriptar, es decir mediante el enlace de datos que ya existe entre los participantes. Obviamente esto conlleva a la creación de procedimientos ordenados y seguros que impidan que un delincuente digital intercepte la clave o bien que proponga una al equipo de participantes, y cada uno de ellos la acepte inocentemente.

Es probable que el delincuente digital se haga pasar por un participante y envíe un mensaje como el siguiente "nuestra clave para este día es RJXGBH, atentamente: tu compañero". De no tener ninguna manera de verificar la autenticidad del mensaje, cualquiera de los participantes puede ser engañado por esta tercera persona y utilizar una clave que es conocida por el enemigo, con el riesgo que toda la comunicación sea percibida, y aún alterada en su integridad.

A continuación se presentan una serie de protocolos o procedimientos que impiden que situaciones de este tipo se generen entre la comunicación de dos o mas personas o sistemas de información.

12.1 ASIGNACION DE CLAVE SECRETA POR UN ARBITRADOR USANDO CLAVE SECRETA PARA AUTENTICAR A LOS PARTICIPANTES

Este protocolo o procedimiento se lleva acabo cuando los dos participantes de la comunicación se ponen de acuerdo en no aceptar claves directamente uno del otro, sino solamente con autenticación de un Centro de Distribución de Claves (KDC : Key Distribution Center), el cual actuará como arbitrador en la comunicación.

El siguiente procedimiento supone que los participantes {A} y {B} comparten cada uno una clave secreta con el arbitrador, estas claves serán denotadas como E_A y E_B, y nadie más aparte de la persona dueña de la clave y el arbitrador la conoce. {A} desea comunicarse con {B}, entonces realiza este sencillo procedimiento:

(1) {A} crea una petición de comunicación concatenando un registro de instante T_A el nombre de {B} y una clave secreta de encriptamiento K , Encripta todo lo anterior con la clave secreta que comparte con el arbitrador y envía este mensaje a el arbitrador junto con su nombre.

$$A,\ E_A(T_A, B, K)$$

(2) El arbitrador recibe el mensaje. desencripta lo encriptado y se da cuenta que en verdad se trata de {A} pues sólo entre ellos existe ésta clave, luego genera un registro de instante T_t, concatena este registro de instante con el nombre de {A} y la clave de encriptamiento K , Encripta toda la concatenación que acaba de realizar con la clave que comparte con {B} y envía el mensaje a {B}.

$$E_B(T_t, A, K)$$

(3) {B} recibe el mensaje y lo desencripta, sabe que sólo el arbitrador conoce esa clase y acepta la clave K como clave para comunicarse con {A}.

El procedimiento se puede interpretar de la siguiente manera: {A} le cuenta en secreto al arbitrador que quiere comunicarse con {B} con la clave K, y luego el arbitrador le cuenta en secreto a {B} que {A} quiere comunicarse con él por medio de la clave K, los secretos se los cuentan en un lenguaje que sólo el arbitrador y uno de ellos conoce, además , los secretos no son copias viejas de los mensajes, son comunicaciones actuales certificadas por los registros de instante (los mensajes podrían ser coleccionados por un delincuente digital, tomarse suficiente tiempo para desencriptarlos, y enviarlos nuevamente sin ningún cambio para obligar a los participantes a usar una clave conocida por él mismo).

El procedimiento anterior puede parecer muy confiable, pero las claves secretas que comparten los participantes con el arbitrador lo hacen inseguro, ya que si esta clave es descifrada por un criptoanalista, se podría interceptar el mensaje dirigido a {B}, cambiar la clave y mandar el mensaje ofreciendo un número para clave. Si todo lo anterior se realiza rápidamente, el registro de instante sería valedero y el participante aceptaría la clave k.

A continuación se presenta un procedimiento para asignar claves con autenticación de los participantes, el cual puede ser inmune a este tipo de ataque anterior. Algunos de sus pasos pueden resultar redundantes, pero son necesarios para evitar intrusiones en el procedimiento.

(1) {A} genera un número aleatorio R_A, el cual concatena con un número índice I (número que indica al arbitrador que se trata de la relación de {A} con {B}), su nombre, y el nombre de {B}. Todo este concatenamiento lo encripta con la clave secreta que comparte con el arbitrador. Luego se comunica con {B} y le envía el numero índice I , su nombre , el nombre de {B} y el bloque encriptado.

$$I, A, B, E_A(R_A, I, A, B)$$

(2) {B} recibe el mensaje de A, y aunque no sabe que dice el bloque encriptado, sabe que A se quiere comunicar con él. Deja todo en manos del arbitrador para que él decida si {A} es realmente{A}. Para esto genera un número aleatorio R_B , lo concatena con el número índice I, el nombre de {A} y su nombre. Encripta toda la concatenación con la clave comparte con el arbitrador. Luego {B} se comunica con el arbitrador identificándose con la siguiente secuencia: El número índice, el nombre de {A}, su nombre, el bloque encriptado que le envió {A} y el bloque que él acaba de encriptar.

$$I, A, B, E_A(R_A, I, A, B), E_B(R_B, I, A, B)$$

(3) El arbitrador recibe el mensaje, sabe que se trata de {A} y {B} por el número índice, por que lo indican sus nombres, y porque los dos mensajes encriptados comprueban estos mismos datos, y sólo podían ser encriptados por {A} y {B}. Entonces genera una clave de encriptamiento K , y encripta la clave K con la clave secreta y los números aleatorios con la clave secreta que comparte con sus clientes: Encripta R_A y la clave K con la clave comparte con {A} y Encripta R_B y la clave K con la clave comparte con {B}. Para finalizar su función envía al que le solicitó la clave un mensaje con el número índice y los dos bloques encriptados.

$$I, E_A(R_A, K), E_B(R_B, K)$$

(4) {B} recibe el mensaje, verifica el número índice y sabe que ese mensaje es para su relación con {A}, desencripta el bloque encriptado que solo él puede encriptar y encuentra ahí la clave, y su número aleatorio. {B} sabe este encriptamiento sólo lo pudo hacer el arbitrador, sabe que no es una clave vieja y guardada porque recuperó su numero aleatorio R_B . {B} se comunica con {A} y le envía el número índice y su bloque encriptado.

$$I, E_A(R_A, K)$$

(5) {A} recibe el mensaje, verifica el número índice y sabe que ese mensaje es para su relación con {B} , desencripta el bloque encriptado que solo él puede encriptar y encuentra ahí la clave, y su número aleatorio. {A} sabe este encriptamiento sólo lo pudo hacer el arbitrador, sabe que no es una clave vieja y guardada porque recuperó su numero aleatorio R_A. Ahora sabe que {B} había enviado la petición, que él envió la suya y que el arbitrador les reconoció. La clave K es confiable porque nadie mas que {A} y {B} la han visto.

12.1 ASIGNACION DE CLAVE SECRETA POR UN ARBITRADOR USANDO CLAVE PUBLICA PARA AUTENTICAR A LOS PARTICIPANTES

De manera semejante a los procedimientos anteriores, el arbitrador comprobará las identidades de los participantes mediante un secreto conocido entre él y cada uno de los participantes. En este caso, cada uno de los participantes tiene una clave privada la cual es usada para encriptar el mensaje, y el arbitrador tiene las claves públicas certificadas de cada uno de sus clientes.

A continuación se muestra un procedimiento de asignación de clave secreta involucrando la clave pública de los participantes y del arbitrador. Se supone que los participantes {A} y {B} comparten cada uno una clave pública con el arbitrador, estas claves serán denotadas como E_A y E_B. También se tienen las firmas digitales de los participantes y del arbitrador (autenticación con encriptamiento del mensaje), . Todo lo anterior puede resultar redundante, pero cada redundancia incrementa la seguridad del procedimiento, ya que es un elemento más que el delincuente digital debe esforzarse en burlar. El siguiente algoritmo es conocido como Woo Lam creado y publicado por T.Y.C. Wom y S.S. Lam en 1992 [8].

(1) {A} envía una petición de clave al arbitrador la cual consiste solamente en su nombre y el nombre de {B}.

$$A , B$$

(2) El arbitrador envía a {A} la clave pública de {B} E_B,autenticado con su firma digital S_T

$$S_T(E_B)$$

(3) {A} recibe el mensaje de el arbitrador, puede confiar en que es del arbitrador porque viene encriptado con una clave que sólo el arbitrador puede usar, y que {A} puede desencriptar porque tiene la clave pública de autenticación del arbitrador para reconocerlo. Ahora {A} genera un número aleatorio R_A, y encripta su nombre seguido de este número con la clave pública de {B} E_B para asegurarse que sólo {B} pueda ver esta información. Luego {A} envía a {B} el bloque encriptado:

$$E_B (R_A , A)$$

(4) {B} recibe la anterior información desencripta su contenido con su clave privada y se da cuenta de que {A} desea comunicarse con él. Para estar seguro de ello envía a el arbitrador su nombre, el nombre de {A} , y el número aleatorio de {A} R_A encriptado con la clave pública del arbitrador K_T.

$$A, B, E_{K_T}(R_A)$$

(5) El arbitrador recibe el mensaje de {B} , se da cuenta que se trata de una petición de {B} para reconocer el llamado de {A}, desencripta lo encriptado con su clave privada K_T y prepara un mensaje de respuesta: autentica la clave pública de {A} E_A con su firma digital, y además genera una clave secreta de encriptamiento aleatoria K, luego forma un bloque con el número aleatorio de {A} R_A, seguido de la clave K, seguido de los nombres de {A} y de {B}. Todo este bloque lo autentica con su firma digital, y para que no sea conocido por nadie encripta este bloque autenticado con la clave pública de {B} y para finalizar envía ambos paquetes a {B}.

$$S_T(E_A) \qquad E_B(S_T(R_A, K, A, B))$$

(6) {B} verifica que la clave pública de {A} K_A es verdadera porque está autenticada por el arbitrador, luego desencripta con su clave pública el paquete encriptado y encuentra que se le ha asignado una clave K para que la usen {B} y {A}, este mensaje lo encuentra también autenticado con la firma del arbitrador, y como una garantía extra encuentra el número aleatorio de {A}, y que sólo podía ser conocido desencriptando con la clave pública del arbitrador. Ahora {B} genera un número aleatorio R_B y envía a {A} el mismo mensaje que recibió del arbitrador (autenticado con la firma del arbitrador), y además su número aleatorio R_B. Para que nadie más que {A} vea el contenido del paquete, lo encripta con la clave pública de {A} que acaba de recibir del arbitrador y envía toda esta información a {A}.

$$E_A(S_T(R_A, K, A, B), R_B)$$

(7) {A} desencripta el mensaje de {B}, verifica la firma digital del arbitrador, y revisa el contenido de lo autenticado. {A} se da cuenta que {B} envió su numero aleatorio al arbitrador porque venía dentro del paquete autenticado por el arbitrador, ademas se da cuenta que es el arbitrador quien asigna la clave K y que dicha clave ha sido creada para uso exclusivo de {A} y {B}, además se ha dado cuenta que ha sido el arbitrador quien ha proporcionado su clave pública a {B} porque ese número aleatorio que había mandado no podía ser conocido por nadie mas que por {B} por medio de su clave privada. En este punto {A} está seguro que se está comunicando con {B} y que es el arbitrador quien ha proporcionado la clave secreta K. Para que {B} tenga esta misma confianza le devuelve su número aleatorio R_B encriptado con su clave pública E_B:

$$E_B(R_B)$$

(8) {B} recibe su número R_B encriptado en su clave pública, sabe que nadie más que {A} pudo obtenerlo, ya que sólo podía ser recuperado por la clave privada de {A} y además se ha dado cuenta

que {A} poseía su clave de encriptamiento, con lo cual el inicio de las comunicaciones con él quedan confirmadas. Ahora {B} sabe que realmente se está comunicando con {A}.

Para que tanto este como otros procedimientos se realicen, estos deben seguirse desde el principio hasta el final, cualquier dato faltante interrumpe el proceso de negociación, la comunicación es cortada inmediatamente pues puede tratarse de un delincuente que está tratando de sabotear el sistema. Nunca hay petición de repetición por errores en el mensaje, sólo existe el corte definitivo.

Recuérdese que los procedimientos para realizar este tipo de autenticaciones de clave secreta son transparentes al usuario. El usuario sólo se da cuenta que solicita una clave con el software que él ha comprado para este fin, y el otro usuario sólo se da cuenta de que hay otro que se quiere comunicar con él. La única noción que ambos pueden tener de todo el proceso es la petición de espera por la negociación que se realiza, la cual es expresada por el programa que se está llevando a cabo.

Cabe la posibilidad que el receptor de la información no se encuentre disponible en el momento de realizar la negociación de claves. Por esta razón casi siempre el software que realiza estas labores de seguridad se encuentra alojado en un servidor de redes, tal como un servidor proxi, o en un dispositivo de seguridad tal como un firewalls para enlazar y protegerse de internet.

13.- NIVELES DE ENCRIPTAMIENTO EN UNA RED.

Hasta el capítulo anterior se ha discutido todo lo referente a los algoritmos de encriptamiento y a los cambios que se realizan en la información, también se han discutido los protocolos de transferencia de datos que brindan autenticación y no repudio a los sistemas de protección de información.

Para obtener una visión completa de los sistemas de encriptamiento es necesario conocer en qué parte de la máquina y de la red funcionan dichos sistemas. En otras palabras: conocer cómo se usan las aplicaciones de encriptamiento, qué elemento las ejecuta, de qué manera toma o envía la información, de quién depende la ejecución de las aplicaciones etc.

Con el fin de exponer toda esta información este capítulo explica de una manera rápida el ambiente en el cual se alojan y trabajan los elementos de software en una red, el lugar de desarrollo y el trabajo que un programa de encriptamiento desempeña dentro de este medio.

13.1 NIVELES OSI EN UNA RED DE COMPUTADORAS.

Con el objeto de ordenar en una jerarquía las diferentes labores de transferencia , procesamiento y presentación de datos en una red de computadoras, ISO (International Standard Organization) ha definido el modelo OSI (Open System Interconection) como una respuesta para la estandarización de la arquitectura de redes. Este modelo define la estructura de una red mediante 7 niveles operativos, cada uno de los cuales desarrolla determinadas labores que soportan con sus funciones el trabajo desarrollado por niveles superiores. Estos niveles pueden ser representados de la siguiente manera [2]:

7	Nivel de aplicación.	niveles de orden superior
6	Nivel de presentación.	
5	Nivel de sesión.	
4	Nivel de transporte.	
3	Nivel de red.	
2	Nivel de enlace de datos.	
1	Nivel físico.	niveles de orden inferior

Figura 13.1

Estos niveles funcionales ordenan la transmisión de datos entre sus elementos sin recargar a ninguno de ellos, además facilitan las tareas de configuración del sistema así como su mantenimiento. Cada uno de ellos realiza las siguientes funciones [2]:

1. Nivel Físico: Se compone de todo el hardware encargado de la transmisión de datos como ceros y unos, sin procesamiento alguno de datos. No define estándares físicos sino que usa los existentes (RS-232C , IEEE 802.3, etc.)
2. Nivel de datos: Prepara los datos en tramas que contienen información acerca de la fuente, destino, los datos mismos, control, y el tipo de trama para que los transporte el nivel físico.
3. Nivel de red: Determina la ruta del mensaje desde el elemento emisor hasta el elemento receptor. Su función es el direccionamiento de la comunicación para que sea ejecutado por el nivel de datos, su herramienta principal es el protocolo IP.
4. Nivel de Transporte: Provee los mecanismos para el manejo de la información, la detección y la corrección de errores, con ayuda de los protocolos TCP o UDP.
5. Nivel de Sesión: Inicia, mantiene, y termina cada sesión lógica entre usuarios, se encarga de hacer y mantener el enlace utilizando los niveles anteriores.
6. Nivel de Presentación: define el formato de la información que será presentada al usuario y es intercambiada entre ellos, así como la sintaxis usada entre aplicaciones.
7. Nivel de Aplicación: Se encarga de soportar directamente las aplicaciones del usuario, tales como programas que no requieren para su ejecución inmediata de otros elementos de la red.

Como puede deducirse del listado anterior, si existen mecanismos físicos y sistemas de software dedicados a la transferencia de datos en una red, debe existir protocolos, es decir procedimientos ordenados y detallados, que indiquen la manera correcta y estandarizada para lleva a cabo las comunicaciones entre los diferentes elementos de la red.

Existen gran variedad de protocolos para transmitir datos en una red, de hecho cada topología de red se ve en la necesidad de crear uno especializado para sus fines, pero si se considera a las redes de área local con conexiones a internet se puede hablar de un sólo estándar: TCP/IP.

13.2 TCP/IP.

TCP/IP (Transfer Control Protocol / Internet Protocol) define un conjunto de recomendaciones que hacen posible la transferencia de datos entre terminales de una red particular, y entre terminales de redes distintas conectadas a internet. Este conjunto de protocolos tiene su origen en la creación de ARPANET , el cual fue un intento exitoso para la creación de redes de computadoras descentralizadas, en las cuales no se dependía de un servidor central para accesar información de otro sistema de cómputo, sino que la conexión podía ser realizada mediante el acceso a cualquier elemento conectado al sistema que se desea accesar.

TCP es un protocolo confiable[29] que define una conexión lógica de extremo a extremo en una red de computadoras. Para llevar a cabo esta misión al momento de transferir datos, se realizan las siguientes labores [14]:

a. Una petición de transmisión por el emisor.
b. Un reconocimiento de lo anterior por el receptor.
c. Un reconocimiento de lo anterior por el emisor.
d. Envío de datos de control referentes a la información que se está transfiriendo (encabezado TCP) seguidos de la información misma.
e. Una señal de fin de transmisión.
f. Un reconocimiento de esta señal por el receptor.
g. Un ultimo reconocimiento del emisor para confirmar la recepción de los mensajes del receptor.

En el caso de existir un error se repite el procedimiento hasta que tanto emisor como receptor sean notificados de haber realizado la transmisión de datos con total perfección. Las anteriores funciones son llevadas a cabo mediante un encabezado TCP el cual contiene en 192 bits, y en orden sucesivo, la información referente a: puerto fuente, puerto destino, número de secuencia, datos de reconocimiento, datos de la información a transferir (desviación, banderas y una ventana para uso de las aplicaciones) , suma de verificación, apuntadores, opciones y rellenos [14]. La figura 13.2 ilustra esta información en una manera mucho más clara.

[29] Un protocolo confiable arroja información de transmisión y recepción de datos con sus respectivas verificaciones y repeticiones de ser necesarias. De manera distinta UDP (User Datagram Protocol) es un protocolo no confiable, pues sólo envía un encabezado con origen y destino del mensaje, su longitud y su suma de verificación, pero no lo verifica con un acuse de recibido, y consecuentemente no realiza retransmisiones en caso de haber errores.

0	4	16	28	31
puerto origen			puerto destino	
numero de secuencia				
numero de reconocimiento				
desviación	reservado	banderas	ventana	
suma de verificación			pointer	
Opciones				relleno

Figura 13.2

Cuando la información a transferir se encuentra en otra red distinta a la red local, pero se encuentra conectada a internet, es necesario, además de realizar todo el protocolo anterior con el servidor que conecta a internet, realizar el protocolo IP en el momento mismo de salida de la red local.[30]

En forma similar a TCP el protocolo de internet (IP) se auxilia de un encabezado para transmitir la información a través de todas las redes conectadas a internet, este bloque inicial para realizar la comunicación contiene información relativa a: la versión de IP que se utiliza, la longitud del encabezado, el tipo de servicio, la longitud total del mensaje, identificación de paquetes del remitente, banderas, número de fragmento de paquete que se está transmitiendo, tiempo de vida del encabezado, tipo de protocolo, suma de verificación del encabezado, dirección de origen, dirección de destino, opciones del usuario y bits de relleno. A continuación de este encabezado se coloca la información que se va a transferir hasta la otra red (esto incluye el usuario final de la otra red a quien se dirige el mensaje) [14]. Obsérvese la figura 13.3 para comprender mejor el orden de los contenidos del encabezado.

0	4	16	28	31
version	long. encabezado	tipo de servicio	longitud total	
identificación de paquetes			banderas	fragmento de paquetes
tiempo de vida	tipo de protocolo		suma de verificación	
dirección origen				
dirección destino				
opciones				relleno

Figura 13.3

Una vez que el mensaje ha salido de la red origen y se transporta a través de los diferentes nodos de internet que lo pueden llevar a su destino, la comunicación en forma de trama de datos toma la siguiente forma:

[30] IP también se utiliza en redes de área local para hacer más eficientes las transmisiones de datos sin centralizar el servicio de comunicación, este tipo de redes son conocidas como intranets.

Figura 13.4

Se puede considerar a cada uno de los encabezados anteriores y a los encabezados que añaden los medios físicos de transferencia de datos (desde el módem de la Computadora personal hasta la trama del canal del satélite) como ***Vehículos Lógicos*** para el transporte de la información digital. Esto es, un vehículo virtual al cual se afianza firmemente la información para viajar remolcada a través de los medios de comunicación físicos por los cuales estos vehículos se desplazan.

Si la red es medida bajo los parámetros de TCP/IP , se pueden considerar únicamente cuatro niveles operativos dentro de la red:

4	Nivel de aplicación.	niveles de orden superior
3	Nivel de transporte de anfitrión.	
2	Nivel de internet.	
1	Nivel de acceso a la red.	niveles de orden inferior

Figura 13.5

Más detalladamente cada uno de estos niveles contiene o desarrolla los siguientes protocolos y aplicaciones [14]:

1. Nivel de acceso a la red: Consiste en rutinas o programas para accesar redes físicas con sus respectivos protocolos, tales como Ethernet, FDDI , ATM, etc.
2. Nivel de Internet: define el datagrama y maneja el enrutamiento de la información a través de internet mediante la creación del encabezado IP y su manejo, y el uso de los protocolos ICMP (Internet Control Message Protocol).
3. Nivel de transporte de anfitrión: proporciona servicios de información de extremo a extremo mediante diversos protocolos, generalmente usará TCP, pero también puede usar UDP (User Datagram Protocol).
4. Nivel de aplicación: consiste en aplicaciones, protocolos y procesos que usan la red, tales como SMTP (Simple Mail Transfer Protocol), FTP (File Transfer Protocol), Telnet, Etc.

13.3 ENCRIPTMIENTO DE LOS ENLACES.

Cuando lo que se encripta son los canales de comunicación, entonces se está hablando de encriptamiento de los enlaces (link Encryption) [6]. Cuando se protege la información desde este nivel no

sólo la información que edita el usuario final es lo que se encripta, sino también todo aquellos protocolos que acompañan a la información.

En algunas ocasiones este tipo de encriptamiento es completamente transparente para el usuario, ya que la información es tratada o bien justo hasta el momento de salida de la terminal, o bien en el momento en que ésta abandona la red local o la subred que la aloja. Sin embargo es necesario que cada nodo de red por el cual transite la información conozca el destino de la información para tomarlo o para enrutarlo en la manera correcta. Eso obliga a que cada nodo que transporta información en la red desencripte el mensaje, lea el protocolo TCP y tome sus decisiones. En el caso de no ser este nodo el destino final del menaje, deberá de enrutarlo hacia un camino distinto, volverá a encriptarlo y lo transmitirá hacia el siguiente nodo [8]. Vease la siguiente figura para comprender mejor este nivel de encriptamiento:

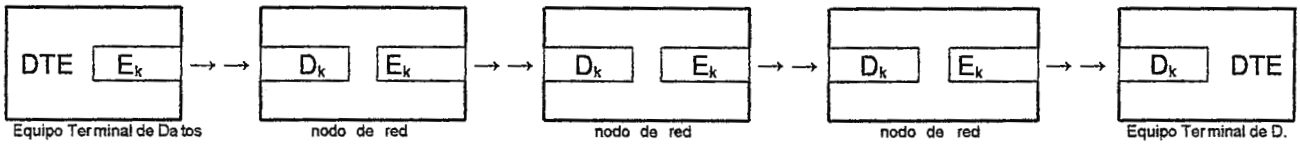


Figura 13.6

Resulta evidente que este tipo de encriptamiento no es posible realizarlo abarcando el ocultamiento del encabezado de los protocolos IP, puesto que si esto se realizara, no podría enrutarse el paquete hasta su destino. El encriptamiento y desencriptamiento sólo puede ser realizado por los nodos de la red privada, pero no por los nodos de internet, puesto que de poder hacerlo se perdería el efecto de seguridad deseado. Este tipo de encriptamiento de enlaces es implementado en su mayoría en redes WAN[31] , ya que frecuentemente utilizan canales de comunicación de fácil acceso para los delincuentes digitales debido a las grandes distancias que cubren.

Otra manera de realizar el encriptamiento de los enlaces es encriptando los protocolos de transferencia de datos y creando mecanismos ocultos de verificación de la información y sus protocolos que los acompañen en todo el camino por la red.

Se puede realizar un encriptamiento de los enlaces cifrando toda la información que acompaña a la información; el texto fuente que se envía puede o no estar encriptado, la transmisión siempre es segura, ya que al no tener el dato para asir el mensaje, para interpretarlo, o para modificarlo, el delincuente digital no tiene ningún recurso para saber qué es lo que se envía, su origen y su destino, y aún dónde comienza la información y donde termina [6].

[31] WAN : Wide Area Network.

Visto desde los niveles TCP/IP de arquitectura de redes, se podría decir que el encriptamiento se lleva a cabo mediante la inserción de un encabezado extra entre el nivel de transporte de anfitrión y el nivel de internet. De manera que toda la información desde el encabezado de transporte de red hasta el fin de la trama se oculte bajo un cifrado secreto (esto significa que todos los datos manejados por los niveles de transporte de anfitrión y de aplicación serán datos encriptados). Este tipo de protocolo es conocido como IPSEC (Internet Protocol SECurity) [6]. Analizado desde el punto de vista de los canales de transferencia de datos, la secuencia de todos los elementos anteriores formaría la trama que se expone a continuación:

Figura 13.7

Recuérdese que todos los nodos de enrutamiento en internet interpretan únicamente el encabezado IP para dirigir la trama de datos hasta su destino, lo que sigue a continuación de dicho encabezado no es leído por estos nodos, únicamente es transferido como simples bits especificados por la longitud del mensaje en el encabezado IP. La red destino si interpreta todo lo que sigue a IP. En el caso de ser una red protegida por IPSEC interpretará la información otorgada por dichos encabezados de seguridad y desencriptará todo lo que encuentra a continuación, interpretará el encabezado TCP y dirigirá la información descifrada hacia la sesión de usuario destino que se especifica.

Existen dos tipos de encabezados IPSEC que se utilizan dependiendo de las necesidades de servicios de seguridad que se requieran [6]:

a) ENCABEZADO DE AUTENTICACIÓN (AH : Autencation Header).

Este tipo de encabezado autentica la información utilizando clave secreta, para encriptar un valor Hash de la información y los protocolos de transporte de datos. Este protocolo se ilustra a continuación:

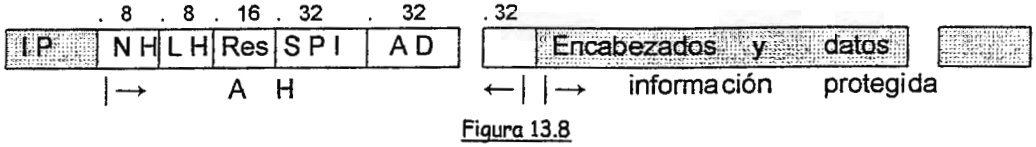


Figura 13.8

Los elemento de este encabezado son:

- Identificación y localización del próximo encabezado (N H : Next Header): Este sub-bloquede 8 bits contiene información acerca del tipo del siguiente encabezado de protocolo y de su ubicación a partir de el primer bit.
- Longitud del encabezado de autenticación en bits (L H : Length Header).
- Datos reservados (Res).
- Información acerca del tipo de encriptamiento utilizado, la clave, y la función Hash de autenticación (SPI: Security Parameter Index).
- Datos de autenticación encriptados (AD: Autentication Data): valores Hash de los datos protegidos. Generalmente estos valores son obtenidos mediante la función Hash MD5 [6].

b) CARGA ENCAPSULADA DE SEGURIDAD: (ESP : Encapsulating Security Payload)

Este encabezado proporciona información acerca del encriptamiento al cual están sometidos los protocolos de transporte, los de aplicación y los datos. Su formato es mucho mas sencillo:

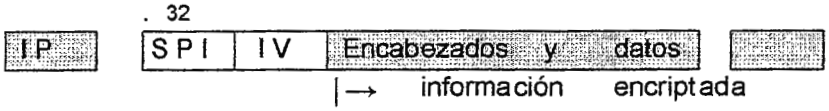


Figura 13.9

Los elemento de este encabezado son:

- Información acerca del tipo de encriptamiento utilizado (SPI: Security Parameter Index) especifica el tipo de encriptamiento utilizado para el resto de la información. Generalmente se usa DES (Data Encription Standard) trabajando en modo CBC (Cipher Block Chaining).
- Vector de Inicialización (I V : Initialization Vector) del algoritmo de encriptamiento.

En la actualidad (año 1999), esta serie de protocolos IPSEC son adosados por las redes protegidas a sus protocolos de conexión a continuación de IP, de esta manera IPSEC es ignorado por los nodos de comunicación IP, pero reconocido por la red de la dirección destino (la versión 4.0 IP no incluye estos protocolos, pero la nueva versión 6.0 de IP a implantarse en el año 2000 los llevará dentro de sí como parte del estándar).

13.4 ENCRIPTAMIENTO PUNTO A PUNTO.

Este tipo de encriptamiento no toma en cuenta para nada el ambiente en el cual el paquete encriptado va a ser transferido, simplemente la información es transformada por voluntad del usuario en el momento mismo en que esta va a ser transferida o guardada.

La mayoría de las aplicaciones de software utilizado popularmente por usuarios aislados conectados por voluntad propia a una red utilizan este tipo de encriptamiento. Su instalación y su uso podría ser comparado con un accesorio del sistema operativo, el cual procesa archivos una vez que están ya almacenados en algún medio (como podría ser un compresor de archivos).

El paquete encriptado generado por este tipo de aplicaciones tiene un nombre y tamaño específicos, y su tráfico por cualquier medio de transferencia de datos lo hace como cualquier otro archivo, viajando por toda la red remolcado por cualquier vehículo lógico, como pudiera ser un token, un encabezado de ethernet, un encabezado TCP o un IP conocido por otros componentes de la red. Al llegar a su destinatario, este lo desencripta y lo utiliza para los usos acordados por ambos participantes.

Si bien este tipo de encriptamiento es más económico y fácil de implementar, este cuenta con la desventaja de que todos los datos referentes a los protocolos de comunicación son conocidos por otros componentes de la red, como podría ser el origen y destino del archivo encriptado. Esto da lugar a que otros sujetos aprovechen la información y puedan llegar a violar la privacidad de la comunicación, o aún puedan llegar a criptoananlizar el sistema al tener la manera de asir archivos cifrados y almacenarlos.

14.- TÉCNICAS DE LOS CRIPTOANALISTAS PARA VIOLAR SISTEMAS DE SEGURIDAD.

No existen los ataques a la información sin ninguna pista acerca de su origen, de su sistema de seguridad, y de su utilidad. Casi siempre los criptoanalistas (ya sean espías, investigadores privados o delincuentes digitales) utilizan un plan muy elaborado de ataque. El presente capítulo expone de una manera rápida las generalidades que toman en cuenta este tipo de especialistas para encontrar la manera de burlar la seguridad de un determinado sistema.

Ningún atentado contra un sistema de información es llevado a cabo fijando exclusiva atención a los mensajes y protocolos de transferencia encriptados. Aunque este es el blanco del ataque, éste no puede ser alcanzado a menos que se tenga suficiente conocimiento del sistema al cual se va a agredir [12] [13].

Todo atacante explora el campo a asaltar antes de tomar acciones específicas. Esta exploración casi siempre involucra espionaje de los sistemas (hardware y software), las personas y las operaciones que anidan en la organización blanco del ataque. Este espionaje puede involucrar: Intercepto de llamadas telefónicas y de comunicaciones de la red de información, investigación de las compañías que proporcionan hardware y software (incluyendo sistemas de encriptamiento y productos de seguridad) a la empresa, investigación de compañías que proporcionan enlaces de comunicación, investigación de las personas que laboran en la empresa, investigación de horarios y procedimientos administrativos, listado de posibles contraseñas y otras investigaciones que usualmente son conseguidas mediante invasión a la privacidad, mentiras o sobornos a las personas involucradas [12] [13].

Con toda la anterior exploración el asaltante puede conocer la manera de interceptar la comunicación (tipo de enlaces, protocolo de tranmisión de datos, puntos de transferencia susceptibles de ser interceptados etc.), la configuración de los sistemas de protección (su marca comercial, sus niveles de encriptamiento,sus técnicas de autenticación, sus métodos y algoritmos de encriptamiento, etc), así como también pequeños detalles que faciliten el acceso a la información protegida (tales como la frecuencia del

cambio de la clave, la frecuencia del cambio de sistema de encriptamiento, la frecuencia del cambio de password, la manera como se almacena toda la información concerniente a el manejo de la seguridad, etc.).

Por último, debe tenerse en cuenta que no es una persona la que hace el asalto, sinó una máquina capaz de actuar con rapidez y en completo sigilo. Esto supone que la persona dedicada a estas labores no es solamente un usuario final de sofware comercial, sino más bien un experto programador, conocedor de la mayoría de los productos y algorimos de encriptamiento, y capaz de crear programas que hagan el trabajo de criptoanálisis de una manera rápida, y que puedan interactuar como impostores en una transmisión de datos [12] [13].

14.1 PISTAS NECESARIAS PARA INICIAR CUALQUIER ATAQUE:

- ENCABEZADOS DE LA INFORMACIÓN: Todo buen criptoanalista conoce todos los encabezados que identifican el formato de la información para que esta pueda ser accesada de la manera correcta (encabezados de WORD, TXT, WORD PERFECT, E-MAIL,ETC.). Y sabe que todo mensaje encriptado corresponde en su primer bloque a uno de estos encabezados, con esto tiene ya en su poder un bloque de texto fuente y un bloque de texto encriptado, encontrar la clave aunque es tarea difícil, no es imposible cuando se dispone de muchas máquinas y muy veloces [8].
- GENERADORES DE NUMEROS PSEUDO ALEATORIOS: Como se ha mencionado anteriormente, los generadores pseudo aleatorios no son en realidad aleatorios, no generan en teoría (cuando n tiende al infinito) todos los números dentro de los límites establecidos, sino que poseen un ciclo, y además sólo producen un conjunto limitado de esos números. Con este conocimiento y una buena colección de generadores pseudoaleatorios pueden generarse fácil y rapidamente toda la colección de claves aleatorias que podrían ser generadas por el sistema de encriptamiento, el cual supuestamente genera sus claves en forma aleatoria [6].
- DICCIONARIO DE ATAQUE : Esta herramienta de ataque es un gran compendio es datos y procedimientos entre los cuales se debe incluir: una abundante colección de password y claves comunes, listas de nombres de hombres, de mujeres, de mascotas, de apodos, de organizaciones, de empresas comerciales, de nombres código militares, de números de seguro social, de números de pasaportes, de números de carnets, etc. así como también un programa que genere todas las posibles siglas y abreviaciones cuando se le introduce un nombre completo, todas las posibles

combinaciones con las letras de ese nombre, todos los posibles recursos mnemotécnicos que esas letras puedan generar, así como todas las posibles asociaciones que ese nombre tenga con distintos personajes célebres de la historia. Lo anterior no implica una lista infinita, tomando en cuenta que los números y nombres pueden ser robados de archivos de registro civil, lo cual descarta todos los nombres y los números que no existen en un determinado país. De reunirse todo lo anterior puede ser construido un ***diccionario de ataque*** para sistemas de seguridad capaz de acertar muchas de las posibles claves de autenticación y de encriptamiento que la mayoría de la gente ocupa para sus labores cotidianas[32] [8].

14.2 TIPOS DE ATAQUES QUE SE DAN A LA INFORMACION:

Los ataques que se dan a la información encriptada pueden se agrupados en cuatro categorías de acuerdo a la cantidad de información que se tiene acerca de los mensajes mismos y de su forma y métodos de encriptamiento [11]:

a) ATAQUE SOLO AL TEXTO CIFRADO

En este tipo de ataque lo único que se tiene es el mensaje encriptado. Se presume que el mensaje es de relativa importancia, estas inferencias han sido deducidas del contexto social y político que envuelve al mensaje. Además se tienen algunas pistas acerca del tipo de encriptamiento que se está usando, puesto que si se ha logrado interceptar el mensaje, es por que se tiene alguna información de los canales de comunicación que se están empleando, del lenguaje en que se ha redactado el mensaje, de los sistemas de encriptamiento que usualmente son utilizados por el sistema, y aún de las personas que están llevando a cabo la comunicación.

Este tipo de ataque es el más difícil de todos, ya que se cuenta con muy poca información para comenzar un ataque. El método de la fuerza bruta resultará muy lento, otros métodos como generadores de números pseudoaleatorios y el uso de diccionario de claves, puede resultar exitosos[33].

[32] David Klein, creador de uno de estos diccionarios asegura ser capaz de encontrar con sus datos el 40 por ciento de todos los password de los E.E. U.U.[8].

[33] Vease el apartado 14.4 para comprender estos métodos para violar el encriptamiento.

b) ATAQUE CON EL TEXTO FUENTE CONOCIDO.

Puede resultar ilógico realizar un ataque a un sistema de encriptamiento cuando se tiene ya un mensaje encriptado, y el mensaje original que llevó a cabo el encriptamiento. Lo que se pretende con este tipo de ataque es conocer detalles acerca del sistema de encriptamiento que se está utilizando.

Algunos de estos detalles pueden ser revelados por el texto cifrado con mucha facilidad: viendo los espacios en blanco en el texto fuente, y las letras más frecuentes en el texto, se puede saber si se está utilizando un algoritmo sencillo (transposición de caracteres, función EXOR simple , sustitución de caracteres), o si se está utilizando un algoritmo sofisticado(DES, RSA, IDEA, DSA, ect.).

En cuanto a la forma de conseguir el texto fuente, existen muchas técnicas, desde la más obvia como sería el robo de papelería confidencial por medios muy sigilosos, hasta la influencia y la manipulación de la persona u organización que transmite los mensajes. Un ejemplo de ello podría ser dar información irrelevante a un espía para que este la transmita a sus clientes bajo el supuesto de haber encontrado un importante secreto de estado, aquí se conoce el texto fuente, y al interceptar el mensaje se tiene un texto cifrado.

Un ejemplo comercial más sofisticado podría tenerse al hacer varias de cuentas transferencias completamente iguales de una cuenta hacia otra, al interceptar las comunicaciones del banco se podrá observar varios mensajes idénticos, entonces se tendrá el texto fuente y el texto cifrado. Este tipo de ataque tiene un objetivo práctico: realizar una transferencia falsa para provecho del delincuente digital. Si ésta operación se realiza por montos millonarios y el autor abandona el país antes del corte del día, se habrá tenido un perfecto fraude bancario.

c) ATAQUE CON TEXTO FUENTE ESCOGIDO PARA EL ATAQUE.

En este tipo de ataque se conoce el texto fuente, y los métodos de encriptamiento que utilizan las personas u organizaciones que se están atacando. Puede o no pretenderse romper el encriptamiento, en algunas ocasiones esto no es necesario.

La mayoría de veces con este tipo de ataque se pretende únicamente alterar el texto que se va a transmitir. Por ejemplo si se trata de una orden de cobro o de una transferencia de dinero entre cuentas, se puede alterar inteligentemente el monto del cobro o el número de las cuentas. Se supone que dichos mensajes van autenticados, y certificados por algún tipo de función Hash de un sólo sentido, por lo que las alteraciones deben ser mínimas, tales como la transposición de cifras o de caracteres, cambios que la función de un sólo sentido puede ignorar, ya que los valores ASCII del mensaje no han sido alterados, sino solamente cambiados de lugar. Un criptoanalista experimentado puede saber que tipo de datos y puestos en

que forma no alteran el sistema de autenticación que la organización atacada no puede detectar. Generalmente este ataque es rápido, para no dar tiempo a las víctimas de alertar a su contraparte en el sistema de comunicación.

d) ATAQUE "HOMBRE EN MEDIO DE COMUNICACIÓN"

En este tipo de ataque, el agresor se sitúa en el medio de comunicación que une los dos sistemas que se van a violar en su privacidad, no sólo se ve la comunicación, sino que se roba, se altera, y se manda información falsa a través de la linea de comunicación.

Para realizar este tipo de ataque es necesario tener la mayor cantidad de información posible acerca de los sistemas que va a ser atacados, ante todo se necesita saber qué métodos de encriptamiento se utilizan, y qué procedimientos se llevan a cabo para verificar certificaciones, autenticaciones y notificaciones de no repudio.

Lo que se falsifica en este tipo de ataque es a las personas. Se manda información a nombre de otra persona, o se recibe información a nombre de otra persona, y para ello se utilizan los sistemas de encriptamiento y de autenticación que utilizarían las personas a las que se está suplantando.

Es incorrecto el nombre de "hombre en el medio de comunicación", más bien debería decirse "máquina en el medio de comunicación", ya que obviamente el ataque no se hace a mano, sino que se fabrica un programa que detecte la información que se quiere interceptar, que tome sus propias decisiones y que altere la información que se quiere alterar. Un humano no podría desarrollar todo este trabajo en una forma rápida, un programa si , pero hay que diseñarlo cuidadosamente tomando en cuenta todos los posibles contratiempos que puedan surgir en el camino.

14.3 TECNICAS PARA VIOLAR LA INFORMACION DURANTE PROCESOS DE COMUNICACIÓN.

Todas las técnica siguientes basan casi siempre su funcionamiento en el ataque del tipo "hombre en el medio de comunicación". Esta es una recopilación de muchas de las técnicas que hackers de la vida real han ocupado para burlar sistemas de seguridad basados en el encriptamiento [6] [12] [13].

a) FALSA REQUISICION DE LAS CLAVES DE ENCRIPTAMIENTO.

Cuando se tiene conocimiento que dos sistemas de información basan sus comunicaciones en un determinado procedimiento de negociación de claves, una tercera persona puede hacerse pasar por uno de los dos participantes que solicita una clave de encriptamiento. Lógicamente esta requisición es encriptada, pero el criptoanalista puede averiguar el sistema de encriptamiento de requisición de claves, sabiendo que este no cambiará, o puede también guardar una requisición de clave original, encriptada, y utilizarla para iniciar el proceso de encriptamiento, una vez que la clave es otorgada, el impostor podrá hacerse pasar por uno de los participantes sin que nadie note que se está tratando con un delincuente [6].

b) ADQUISISION DE CLAVES PUBLICAS DE LOS PARTICIPANTES EN EL ENCRIPTAMIENTO.

Muchos de los métodos para entablar una sesión de comunicación encriptada se basan en la adquisición de una clave secreta de encriptamiento por parte de los participantes. Dicha clave es negociada por medio de peticiones y certificaciones en clave pública, lo cual asegura la autenticidad de los participantes. Si una tercera persona averigua la clave privada de encriptamiento usada para estas requisiciones y logra hacer una requisición de clave, empezará una sesión de comunicación con un sistema sin que éste se de cuenta que está tratando con un impostor. En ocasiones puede no ser necesario ni siquiera averiguar la clave privada de una de los participantes, sino solamente copiar la petición de clave encriptada, si esta clave se obsequia en forma clara al pedirla con clave pública, bastará con tener una copia de la petición aunque no se sepa qué cosa dice la petición [10].

c) REPETICION DE MENSAJES.

En algunas ocasiones no es necesario conocer cómo se encriptó el mensaje, sino solamente lo que significa un bloque ilegible. En el caso de un empleado listo que quiera duplicar, o multiplicar n veces su salario puede pedir a su empleador que simplemente transfiera el valor de su salario a su cuenta. Si este empleado se da cuenta cual transacción en el sistema es relativa a su salario, este empleado puede copiar esta transferencia de datos (vaya encriptada o no) y repetirla las veces que desee, de manera que con cada repetición su salario se multiplicará [6].

d) EL ATACANTE SE HACE PASAR POR ARBITRADOR.

Cuando se conoce que los participantes en una comunicación encriptada utilizan un arbitrador para generar su clave de encriptamiento secreta, la forma más fácil de atacar esta comunicación es atacando al arbitrador. Este ataque puede ser desarrollado suscribiendose a los servicios de asignación de claves del

arbitrador en el cual se está interesado. Una vez teniendo acceso al arbitrador puede hacerse todo lo posible para atacar su sistema de base de datos (para un cliente autorizado a ser servido por un servidor le es posible atacar a su servidor para encontrar información) y conseguir toda la información referente a el ataque que desea realizar.

El ataque se llevará a cabo cuando uno de los dos participantes de la comunicación solicite clave de encriptamiento. El atacante se hará pasar por el arbitrador, reconocerá al solicitante, reconocerá al receptor de la comunicación, y el solicitante lo identificará como "el arbitrador". Cuando la clave sea asignada, la comunicación encriptada dará inicio. El atacante se hará pasar como "el otro" participante, pues conoce la clave de encriptamiento con la que se está trabajando [8].

14.4 TECNICAS PARA DESCIFRAR EL ENCRIPTAMIENTO.

Se presentan a continuación una serie de técnicas para descifrar el encriptamiento. Algunas veces esta labor se puede reducir solamente al desciframiento de la clave con la cual se ha encriptado el mensaje, en cambio en otras ocasiones puede ser solamente dar pistas acerca de el sistema de encriptamiento, para luego continuar con otra técnica más avanzada hasta el desciframiento total del texto. Se presentan algunas opciones que posiblemente no hayan sido llevadas a la práctica, pero que no pueden ser descartadas como posibles alternativas, que con un poco más de refinamiento podrían convertirse en herramientas del criptoanálisis.

a) FRECUENCIA DE LOS CARACTERES USUALES DE UN LENGUAJE:

Este tipo de análisis basa sus observaciones en los caracteres más usados por un lenguaje en particular. Se sabe que en español una de las letras más frecuentes es la letra "a", seguida por la "s", y que una de las más inusuales es la "x", así mismo en inglés se sabe que una de las letras más usuales son la "e" , la "t" y la "a". Si se está en presencia de un algoritmo de encriptamiento sencillo, que encripta caracter por caracter o bit a bit con un mismo patrón, aquellos caracteres o conjuntos de bits que resulten más frecuentes pueden corresponder a las anteriores letras. Por otro lado cuando se trata de un mensaje de otra naturaleza, tal como un informe de ventas o un pedido comercial, se sabe que los caracteres o códigos más comunes van a ser una determinada letra, función o número, el cual es común por la naturaleza del mensaje [6].

b) FUERZA BRUTA.

Esta técnica es tal vez la más sencilla de explicar de todas, aunque la más difícil de ejecutar. Si se tiene conocimiento de qué algoritmo o algoritmos de encriptamiento utilizado por el adversario, se desarrollará el desencriptamiento con dicho algoritmo probando todas las claves posibles hasta encontrar la que desencripta el mensaje del adversario [10].

c) FUERZA BRUTA CON GENERADORES DE NUMEROS ALEATORIOS.

Esta no es más que un refinamiento de la técnica anterior. Ya no se va a examinar todas las posibles claves que se puedan utilizar, sólo se van a examinar los números que producen los distintos métodos de generación de números aleatorios que se sospecha que usa el adversario[34] [6].

d) MAQUINAS ESPECIALIZADAS.

Es posible reunir todas las técnicas anteriores, y los recursos para violar la seguridad de los sistemas, en un solo programa que se aloje en una máquina de múltiples microprocesadores, todos ellos trabajando en paralelo para examinar un sistema de encriptamiento sin que un micro procesador repita el procedimiento realizado por otro. Esta máquina no necesita más que compartir una memoria de entrada para el texto a examinar, y una única salida para publicar la clave encontrada y el texto fuente. Esta es una propuesta de Michael Wiener de Bell-Northern research en 1994 [6] , aunque posiblemente no ha sido llevada a la práctica, ha sido muy tomada en cuenta por muchos de los investigadores de sistemas de encriptamiento, hasta el punto de hacer mención frecuente de esta máquina[35].

e) OPERACION PARALELA EN UNA RED.

En el caso de una emergencia nacional en una guerra, todas las redes de información del sector público pueden ser mandadas a operar en paralelo con un único objetivo: encontrar la clave de un mensaje encriptado. Esta técnica ya ha sido ocupada por U.S. ARMY durante la guerra fría, con la única diferencia de que no involucró a todas las computadoras del gobierno, sino sólo a un pequeño sector de la ARMY. Este tipo de operación también puede ser realizada en forma ilegal por el director de un centro de cómputo cuando las computadoras no están siendo utilizadas para trabajar en sus labores cotidianas, de manera que

[34] Recuérdese que los generadores de números aleatorios no desarrollan todo el rango de valores posibles, sino solamente un conjunto reducido de valores, y en la mayoría de los casos repiten los mismos números en forma periódica, todo ello disminuye considerablemente la cantidad de valores a examinar.

[35] Usualmente se menciona este proyecto como la máquina de 1 millón de Dólares para criptoanálisis (the 1 millon Dollars Criptanalisys Engine).

aprovechando la existencia de la red se utilicen todos sus recursos para la consecusión de éste único objetivo [8].

f) PROGRAMA VIRUS.

Cuando no se disponga de una red para realizar criptoanálisis, puede utilizarse internet para este efecto. Por supuesto, voluntariamente nadie va a colaborar, pero si el programa de criptoanálisis y los datos se propagan en forma de virus, esto obligaría a los contagiados a ayudar. El programa virus podría permanecer en cada computadora aún cuando no esté conectada a intenet, y mantenerse funcionando cuando el protector de pantalla (screen saver) está funcionando, y transmitir la información por medio de petición de conexión al Browser del software de la máquina cuando ésta se conecte a internet[36] [8].

g) LOTERIA CHINA.

Esta sólo es una propuesta, es bastante improbable pero es posible, y sería la opción más rápida. Si cada equipo de T.V. contara con un microprocesador y memoria capaces de recibir información (para procesar: programas, datos...) por medio de la antena normal de T.V. comercial, se podría transmitir un programa para desencriptar un texto cifrado específico en la emisora de T.V. durante las horas de ocio(pues en este momento se tienen el mayor número de aparatos en recepción de la señal comercial). Si esto se hace en China, mas de docientos millones de equipos harían el trabajo en forma paralela[37].

Para realizar lo anterior, el programa criptoanalista deberá probar claves aleatorias, puesto que en una transmisión simultánea no se pueden asignar claves especificas para cada equipo receptor, sin embargo se pueden crear distintas opciones de aleatoriedad según sea la repetidora de T.V. que lo transmite.

El problema de recuperar la clave correcta encontrada podría resolverse poniendo en pantalla el mensaje: "llame gratis al xxxxxxxx diga la siguiente clave y gane $100.00". Bruce Schneier[38], reconocido investigador y creador de sistemas de seguridad y encriptamiento, opina que con este método se podría romper el encriptamiento DES en unos 10 minutos (si existiera y si se realizara en china) [8] [comentario del libro citado al artículo de la revista computer v 24 , nov 1991].

---

[36]Recientes virus de internet aprovechan la libreta de direcciones del usuario final de servicios de correo eléctrónico para propagarse con extraordinaria rapidez hacia todas las máquinas que alojan el servicio de correos de los clientes. Este recurso puede ser muy bien empleado para propagar el algoritmo de criptoanálisis y aún para retornar el valor de clave buscado hasta el origen de la infección digital.

[37]Tomando en cuenta la posobilidad que existan más de 1 televisor por cada 5 habitantes.

[38]Autor del algoritmo de encriptamiento "Blowfish".

15.- ELEMENTOS MATEMÁTICOS A CONSIDERAR EN EL DESARROLLO DE UNA APLICACIÓN.

Con el objetivo de comprender mejor el desarrollo de la aplicación de encriptamiento que se desarrolla en este proyecto de graduación, se expone en este capítulo todas aquellas funciones y propiedades matemáticas que se utilizarán para crear el algoritmo de encriptamiento original. Se dedica este capítulo a la explicación de ciertas propiedades observadas en algoritmos de encriptamiento simétricos los cuales sirven tanto para encriptar como para desencriptar sin modificar la secuencia de sus procedimientos.

Los conceptos expuestos a continuación no serán encontrados en ningún libro, son únicamente apelativos que el autor de este documento considera adecuados para describir en una forma nominativa las propiedades matemáticas de ciertos procedimientos de aritmética, álgebra, y trigonometría tradicional, y del álgebra booleana.

La mayoría de los fenómenos matemáticos expuestos son de invención exclusiva del autor. Existen algunos procedimientos o funciones que si son extraídos de algoritmos de encriptamiento comerciales (vease el ejemplo de la función iterativa recursiva de tercer orden), pero todos los demás son observaciones que la experiencia en el manejo de números binarios en sistemas con microprocesador proporciona a quien practica la manipulación de bits en dichos sistemas.

15.1 FUNCIONES ITERATIVAS RECURSIVAS.

Se le llamará funciones iterativas recursivas a todas aquellas funciones las cuales al ser operadas en forma iterativa n veces producen a su salida el valor que sirvió como origen para la primera operación (esto es tomando como abcisa el valor generado por la ordenada en la operación anterior). Esto puede ser expresado de una manera mucho más clara de la siguiente manera:

$f(x)$ es una función iterativa recursiva si y solo si:

$$f(f (f (\ldots f(x) \ldots))) \quad = \quad x$$

El número de veces que la función debe ser operada en forma recursiva debe ser un número entero finito. Existen algunas funciones algebraicas que cumplen la propiedad iterativa recursiva en el límite cuando el número de iteraciones tiende a infinito; estas funciones no se podrán considerar como funciones iterativas recursivas.

Para los efectos que el presente documento abarca, se hará énfasis en este apartado en las funciones iterativas recursivas que operan números enteros, y más específicamente, las que operan números binarios.

ORDEN DE LAS FUNCIONES ITERATIVAS RECURSIVAS:

El número de veces que es necesario operar la función en forma iterativa define el orden de la función. De esta manera , una función de primer orden será aquella que devuelve el valor x en la primera iteración. Por ejemplo:

$f(x) = (x * x)^{0..5}$ Para números positivos.

de esta manera: $f(2) = (2*2)^{0.5} = 2$

Una función iterativa recursiva de segundo orden devolverá el valor original al cabo de dos iteraciones.

EJEMPLOS:

Función $f(x)$ = negado (x) Para x = número hexadecimal o binario.

valor inicial: x = 8C

1ª iteración $f(8C) = 74$

2ª iteración $f(74) = 8C$

Función $f(x)$ = (1011) EXOR (x) Para x = número binario de 4 cifras.

valor inicial: x = 0011

1ª iteración $f(0011) = 1000$

2ª iteración $f(1000) = 0011$

Función $f(x) = \text{rot}(x, 4)$ Para números de ocho bits. Esto es la rotación de bits en la mitad de la longitud de bits del número que se opera.

valor inicial: $x = 00110100$

1ª iteración $f(00110100) = 01000011$

2ª iteración $f(01000011) = 00110100$

Función $f(x) = \text{transposición}(x)$ Transposición simétrica de pares de bits. Cada bit se intercambia con otro en pares definidos. Para este ejemplo se intercambia bit 3 con bit 1 y bit 2 con bit 0.

valor inicial $x = 1001$

1ª iteración $f(0011) = 0110$

2ª iteración $f(0011) = 1001$

Funciones iterativas recursivas de segundo orden con manipulaciones de bloques de bits:

EJEMPLO:

EXOR de medios bloques sucesivos:

Se tiene un bloque B dividido en dos sub-bloques SB1 y SB2. La función operará en EXOR el segundo sub-bloque con su anterior y almacenará el resultado en el segundo sub-bloque, sin alterar en nada el primero. Esto se puede ilustrar mejor de la siguiente manera:

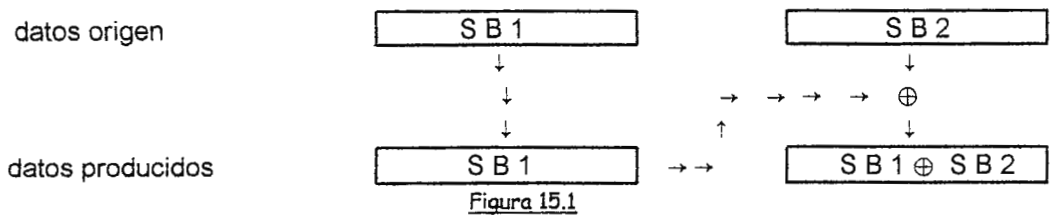


Figura 15.1

operando datos de ejemplo:

primera iteración:

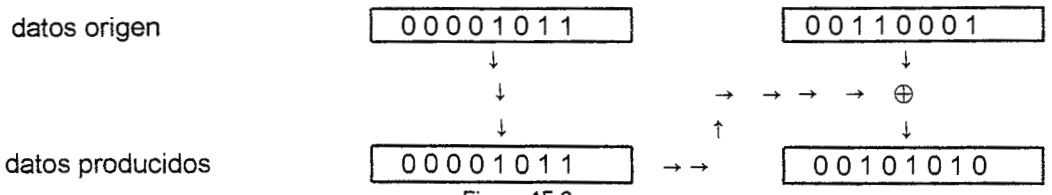


Figura 15.2

segunda iteración:

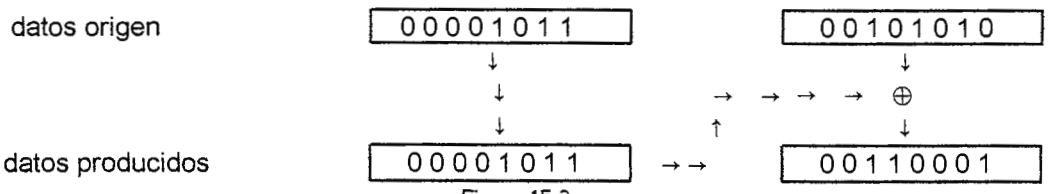


Figura 15.3

Una función iterativa recursiva de tercer orden devolverá el valor original al cabo de tres iteraciones:

EJEMPLO:

EXOR de medios bloques con intercambio de bloques[39]:

Se tiene un bloque B dividido en dos sub-bloques SB1 y SB2. La función pondrá SB2 en el lugar de SB1, operará en EXOR SB2 con su sub-bloque anterior colocando el resultado en el lugar de SB2. Obsérvese la siguiente ilustración para comprender mejor la operación:

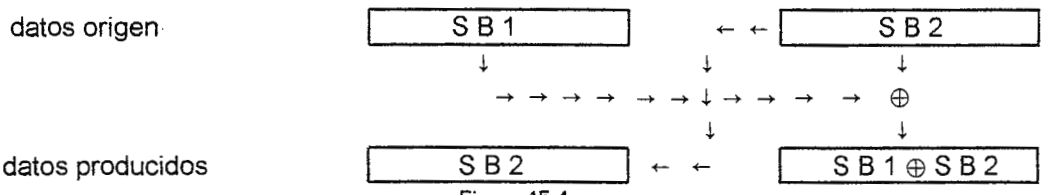


Figura 15.4

[39]Esta operación es desarrollada por Data Encription Standard en cada una de sus etapas de encriptamiento, con la salvedad que en este algoritmo el segundo sub-bloque sufre algunas modificaciones antes de su operación en EXOR con su compañero.

operando datos de ejemplo:

primera iteración:

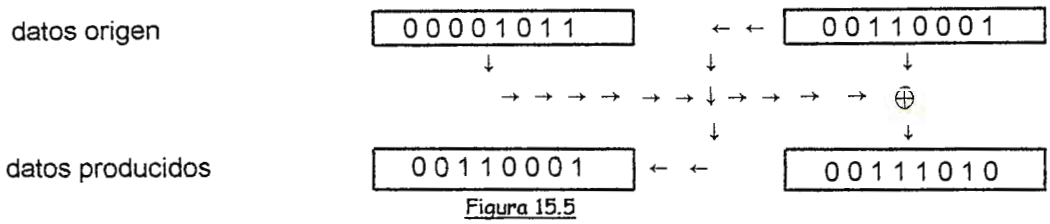


Figura 15.5

segunda iteración:

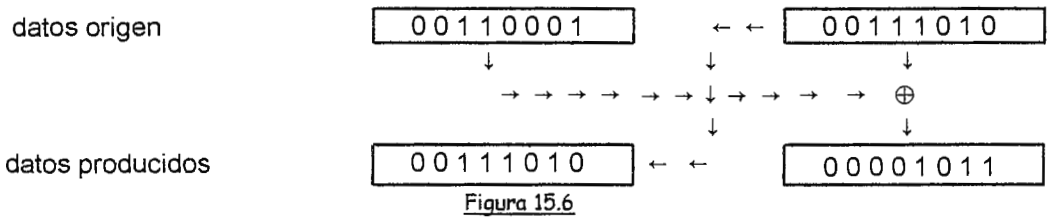


Figura 15.6

tercera iteración:

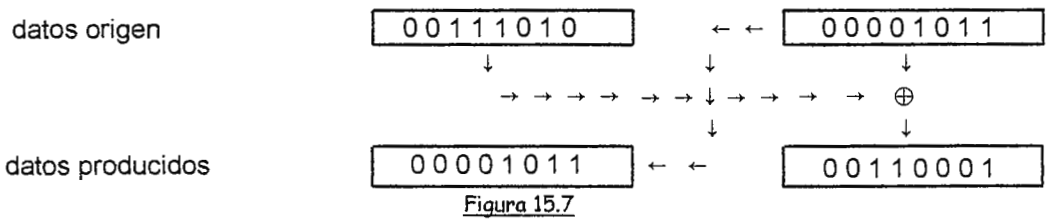


Figura 15.7

Una función iterativa recursiva de cuarto orden devolverá el valor original al cabo de cuatro iteraciones.

EJEMPLOS:

EXOR de 4 sub-bloques sucesivos:

Este tipo de función no es más que la duplicación de la función EXOR de medios bloques sucesivos pero con la particularidad de que los bloques son más pequeños, se trabajan cuatro sub-bloques.

A continuación se muestra la manera de operar los datos de cada sub-bloque utilizando variables algebraicas que representan el contenido de cada uno de ellos:

primera iteración:

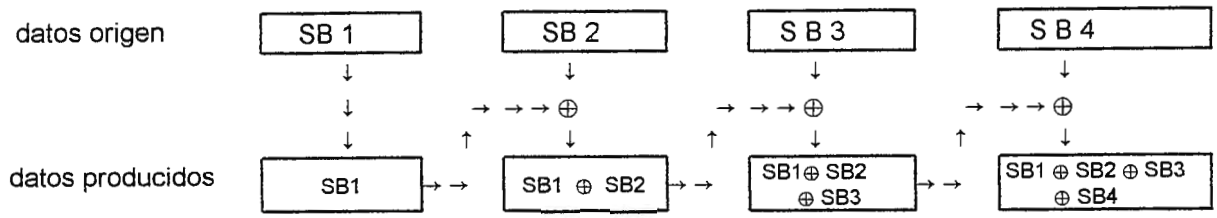


Figura 15.8

segunda iteración:

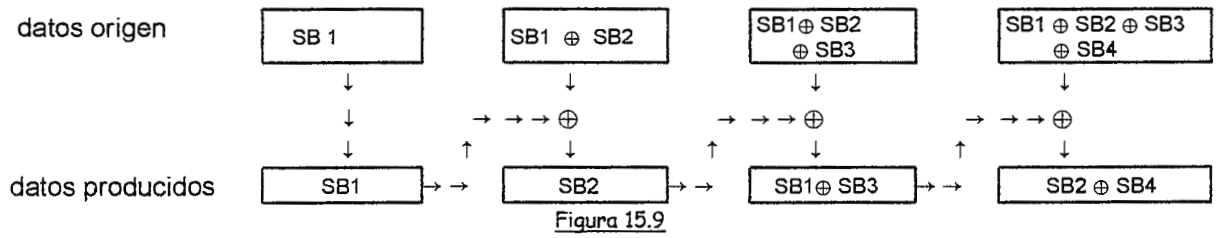


Figura 15.9

tercera iteración:

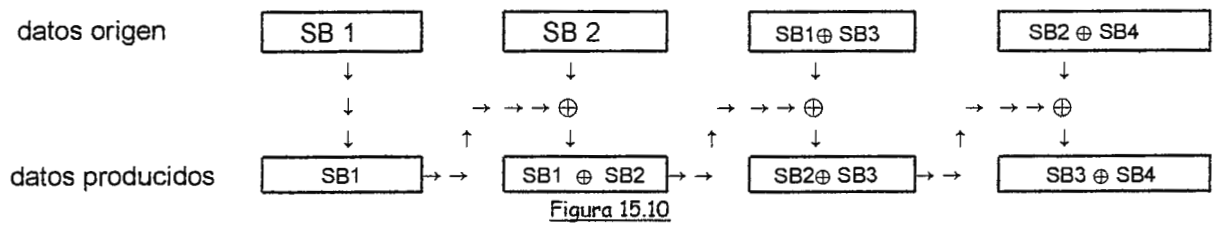


Figura 15.10

cuarta iteración:

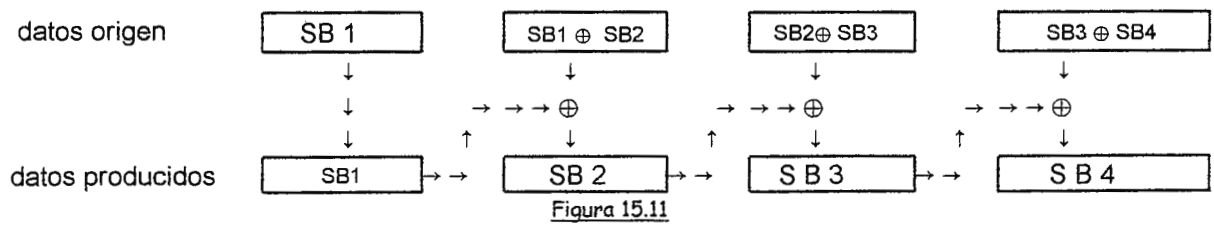


Figura 15.11

Podría pensarse que si la operación en EXOR de bloques sucesivos de dos sub-bloques desarrolla su propiedad recursiva a las dos iteraciones, y la de cuatro sub-bloques la desarrolla en cuatro, entonces la operación de tres sub-bloques la desarrollará en tres. Esta deducción es falsa. Obsérvese su desarrollo algebraico para comprenderlo mejor.

EXOR de 3 sub-bloques sucesivos:

Este tipo de función opera en función EXOR cada sub-bloque con su anterior (a excepción del primer bloque que no sufre ninguna operación), se trabajan tres sub-bloques como se muestra a continuación:

primera iteración:

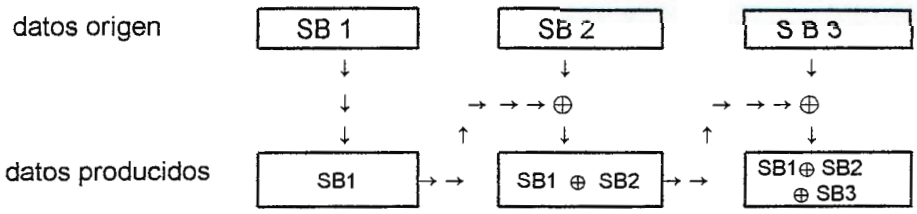


Figura 15.12

segunda iteración:

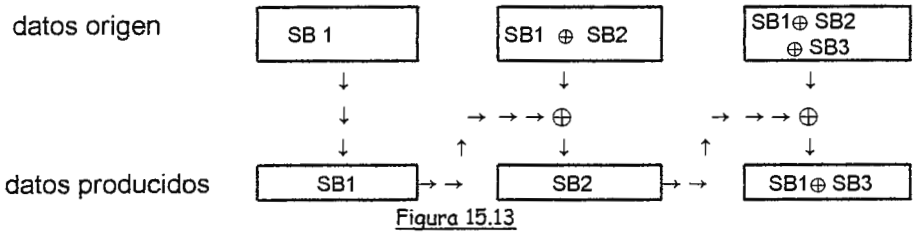


Figura 15.13

tercera iteración:

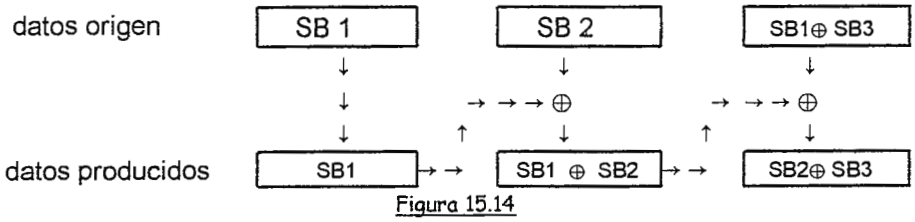


Figura 15.14

cuarta iteración:

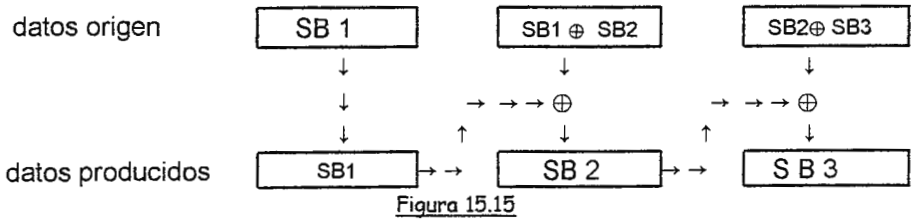


Figura 15.15

15.2 CONCATENACIÓN DE FUNCIONES ITERATIVAS RECURSIVAS.

Es posible crear nuevas funciones a partir de la sucesión de una función iterativa recursiva que opere el resultado de otra función de su misma naturaleza, el resultado de esta concatenación es siempre una función iterativa recursiva, la cual retornará el valor origen al cabo de n iteraciones siempre y cuando el orden de la concatenación de funciones se mantenga constante a lo largo de todas las iteraciones.

Es fácil demostrar la validez de la afirmación anterior, si se considera a la función iterativa recursiva como una función periódica, la cual tiene como dominio el número de iteraciones que se han realizado, y como rango los distintos valores enteros que se proporcionan como salida en cada iteración. De esta manera el ciclo de periodicidad será numéricamente igual al orden de la función.

Es conocido que el comportamiento de una función periódica la cual toma como valores de entrada los producidos por otra función de igual naturaleza es siempre periódico. Un ejemplo de lo anterior se puede visualizar en el comportamiento de la función trigonométrica: $f(x) = \text{sen}\ (\ \cos\ (\ 2x)\ )$.

En forma similar, en el caso de funciones iterativas recursivas digitales como las que se han discutido en las páginas anteriores, la concatenación de dos o más producirá un comportamiento periódico a lo largo del avance lineal del número de iteraciones.

Una concatenación de funciones iterativas recursivas siempre retornará el dato fuente en alguna de sus iteraciones. Puesto que las unidades de su dominio y rango son números enteros, y su comportamiento es periódico, existe la certeza que cuando ésta haya trazado todos los elementos que definen su período, el valor inmediato siguiente a obtener sea el mismo con el cual empezó su período, el cual podría ser el dato fuente.

Se puede ilustrar este comportamiento desarrollando la concatenación de dos funciones sencillas, y mostrando el resultado obtenido al final de cada iteración:

EJEMPLO:

Primera Función $f_1(x) = (10110001)$ EXOR (x) para un número binario de 8 cifras.

Segunda Función $f_2(x) = \text{rot}\ (x, 4)$ rotación de bits la mitad de la longitud del número.

Valor del texto Fuente: $01000010 = 42_{hex} = 66_{dec} =$ " B " $_{ASCII}$

Las dos funciones anteriores son de orden segundo, es decir , retornarán el valor inicial al cabo de dos iteraciones. Esto puede apreciarse más claramente con ayuda de las siguientes tablas de pares ordenados, en las cuales la abcisa corresponde al número de iteraciones, y la ordenada corresponde al valor de la función operándose en forma iterativa sobre su resultado anterior. (nótese que estas tablas corresponden únicamente al comportamiento de las funciones cuando operan el valor hexadecimal 42 (66_{dec}) como dato origen).

Función 1

Iteraciones	f_1(It.)
0	66
1	243
2	66
3	243
4	66
5	243
6	66
7	243

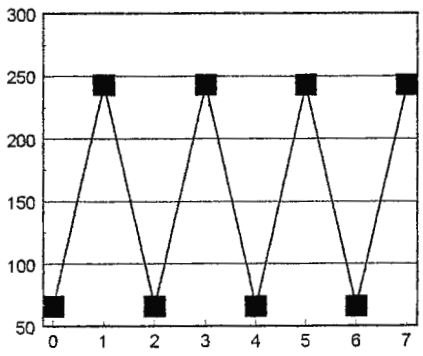
Tabla 15.1

Función 2

Iteraciones	f_2(It.)
0	66
1	36
2	66
3	36
4	66
5	36
6	66
7	36

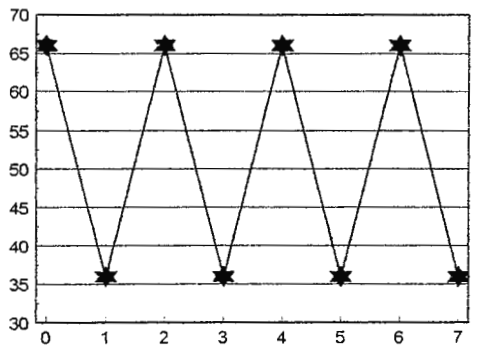
Tabla 15.2

Este comportamiento de las funciones en el dominio de las iteraciones puede ser comprendido de una manera visual con ayuda de los siguientes gráficos de los pares ordenados obtenidos en las tablas anteriores:



FUNCIÓN 1

Figura 15.16



FUNCIÓN 2

Figura 15.17

Se observa con claridad que ambas funciones son periódicas en el dominio iterativo, de manera que se puede esperar que su concatenación también lo sea. Para comprobar la afirmación anterior se desarrollará la concatenación hasta que se obtenga como resultado el dato origen que desencadenó la secuencia:

Primera Iteración:

$$f_1(f_2(x)) = \text{rot}\{[(10110001)\text{EXOR}(01000010)], 4\}$$
$$= \text{rot}\{[11110011], 4\}$$
$$= 00111111 = 3F_{hex} = \text{“ ? “}_{ASCII}$$

Segunda Iteración:

$$f_1(f_2(x)) = \text{rot}\{[(10110001)\text{EXOR}(00111111)], 4\}$$
$$= \text{rot}\{[10001110], 4\}$$
$$= 11101000 = E8_{hex} = \text{“ Þ “}_{ASCII}$$

Tercera Iteración:

$$f_1(f_2(x)) = \text{rot}\{[(10110001)\text{EXOR}(11101000)], 4\}$$
$$= \text{rot}\{[01011001], 4\}$$
$$= 10010101 = 95_{hex} = \text{“ ò “}_{ASCII}$$

Cuarta Iteración:

$$f_1(f_2(x)) = \text{rot}\{[(10110001)\text{EXOR}(10010101)], 4\}$$
$$= \text{rot}\{[00100100], 4\}$$
$$= 01000010 = 42_{hex} = 66_{dec} = \text{“ B “}_{ASCII}$$

Continuando el desarrollo del valor obtenido por las iteraciones se puede obtener la siguiente tabla de pares ordenados, para comprender mejor el comportamiento de la función con ayuda de su gráfica:

FUNCIÓN CONCATENACIÓN

Iteraciones	f_1(It.)
0	66
1	63
2	232
3	149
4	66
5	63
6	232
7	149

Tabla 15.3

Iteraciones	f_2(It.)
8	66
9	63
10	232
11	149
12	66
13	63
14	232
15	149

Tabla 15.4

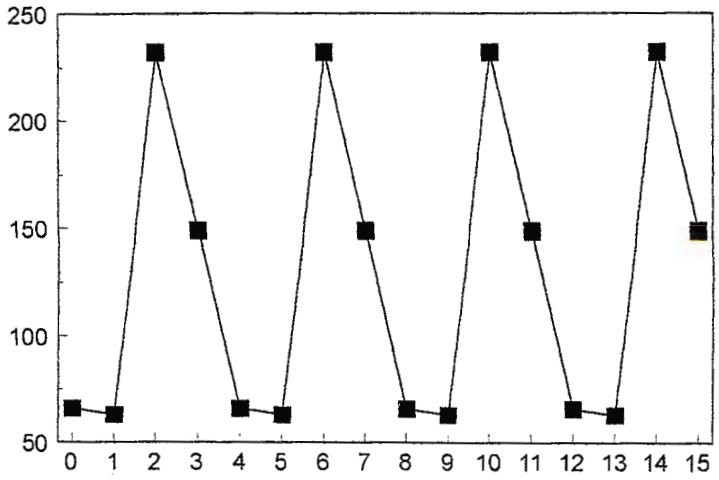


Figura 15.18

Como puede observarse, la función concatenación es también una función periódica en el dominio iterativo, lo cual demuestra que es también una función iterativa recursiva,

Podría pensarse que el período de la concatenación es igual a la suma del valor numérico del orden de las funciones que incluye, pero esto no es cierto, ya que este ciclo es único para cada concatenación, el lector puede intentar personalmente la concatenación de otras funciones, tal como el EXOR de 3 Sub-bloques susesivos con rotación de 1/6 de los bits, con la negación del total del bloque, con operación EXOR con un número clave fijo, ... Con múltiples desarrollos podrá demostrarse que el período de la concatenación depende únicamente de la naturaleza de las funciones de la concatenación.

15.3 FUNCIONES ITERATIVAS RECURSIVAS DE ORDEN MÚLTIPLE

Hasta el momento se ha trabajado bajo el supuesto de que las funciones iterativas recursivas tienen un orden fijo, pero en realidad el orden de una función depende de el valor origen que se asigne, ya que es este valor el que puede dar un período de concatenación que varíe el orden de la función.

Un ejemplo de este comportamiento se tiene en la función: $f(x) = \text{rot}(x, 4)$ (rotación de 4 bits en un número de 8), la cual si opera números hexadecimales con iguales cifras (00, 11, 88, dd ...) , se convierte en una función de primer orden, ya que no efectúa ningún cambio en el dato origen a lo largo de todas las iteraciones que pueda desarrollar.

Para ilustrar mejor este caso particular de las funciones iterativas recursivas se expone a continuación un ejemplo desarrollado con lógica digital binaria secuencial, en la cual se analiza el comportamiento de dos funciones, y su concatenación:

EJEMPLO:

Se tienen dos funciones secuenciales las cuales toman los siguientes valores:

F U N C I Ó N 1

valor anterior	valor siguiente
0	3
3	0
1	2
2	1

Tabla 15.5

F U N C I Ó N 2

valor anterior	valor siguiente
0	2
2	3
3	1
1	0

Tabla 15.6

Sus tablas de transición se pueden expresar:

F UNCIÓN 1

ESTA DO ANTE RIOR		ESTA DO SIGUIE NTE	
A	B	**A**	**B**
0	0	1	1
0	1	1	0
1	0	0	1
1	1	0	0

Tabla 15.7

F UNCIÓN 2

ESTA DO ANTE RIOR		ESTA DO SIGUIE NTE	
A	B	**A**	**B**
0	0	1	0
0	1	0	0
1	0	1	1
1	0	1	1

Tabla 15.8

En lógica secuencial sus ecuaciones de transición se podrían expresar:

función 1: $A = \text{NOT}(A)$ $B = \text{NOT}(B)$

función 2: $A = \text{NOT}(B)$ $B = A$

Su concatenación se puede llevar a cabo sustituyendo A y B de función 1 en las expresiones de función 2.

A continuación se muestran sus gráficos en los cuales se expresan cada uno de los valores que tienen en sus tablas de transición:

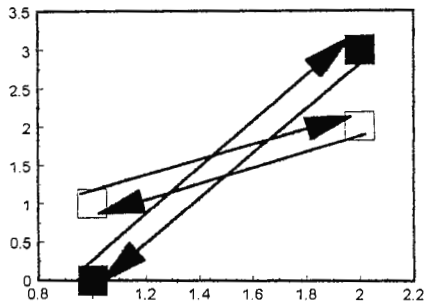


Figura 15.19

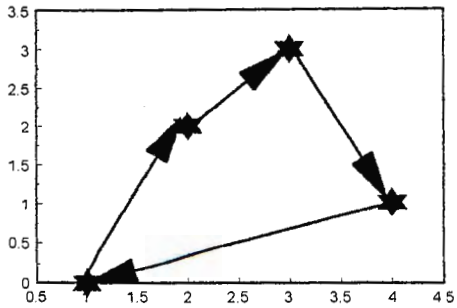


Figura 15.20

En el dominio iterativo estas mismas funciones pueden ser desarrolladas como lo muestran las siguientes tablas de pares ordenados:

F U N C I Ó N 1

ITERACIONES	valor siguiente	valor siguiente
0	0	1
1	3	2
2	0	1
3	3	2
4	0	1
5	3	2
6	0	1
7	3	2
8	0	1
9	3	2
10	0	1

Tabla 15.9

F U N C I Ó N 2

ITERACIONES	valor siguiente
0	0
1	2
2	3
3	1
4	0
5	2
6	3
7	1
8	0
9	2
10	3

Tabla 15.10

Estos datos pueden ser expresados en gráficos de la siguiente manera:

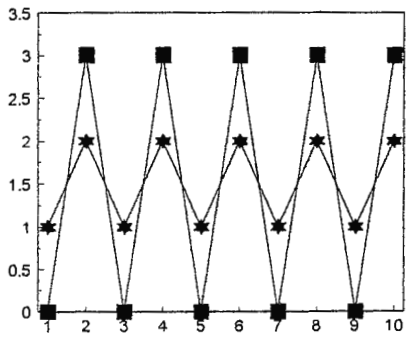


Figura 15.21

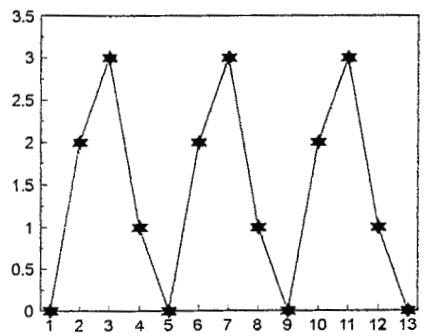


Figura 15.22

Como puede observarse, ambas funciones son periódicas en el dominio iterativo, es de esperar que su concatenación también lo sea.

La función concatenación puede obtenerse sustituyendo los valores de estado siguiente de la función 1 como estados actuales de la función 2, de ésta manera los valores de estado siguiente de la misma serán los valores que tomará la función concatenación como su estado siguiente cuando el estado actual es el que posee la función 1 como entrada. Esto puede ser desarrollado mediante las sustitución sucesiva de valores en las tablas de transición de las funciones, tal como se muestra a continuación:

	FUNCIÓN 1	FUNCIÓN 2
Valor Actual	valor siguiente	valor siguiente
0	3	1
1	2	3
2	0	2
3	1	0

Tabla 15.11

CONCATENACIÓN	
Valor Actual	valor siguiente
0	1
1	3
2	2
3	0

Tabla 15.12

Con la anterior información del comportamiento de la función concatenación, puede desarrollarse su conducta en el dominio de la frecuencia de la misma manera que se hizo con las funciones que la generaron. La siguiente tabla y el siguiente gráfico definen a la función de una manera bastante comprensible:

ITERACIONES	valor siguiente	valor siguiente
1	0	2
2	1	2
3	3	2
4	0	2
5	1	2
6	3	2
7	0	2
8	1	2
9	3	2
10	0	2
11	1	2

Tabla 15.13

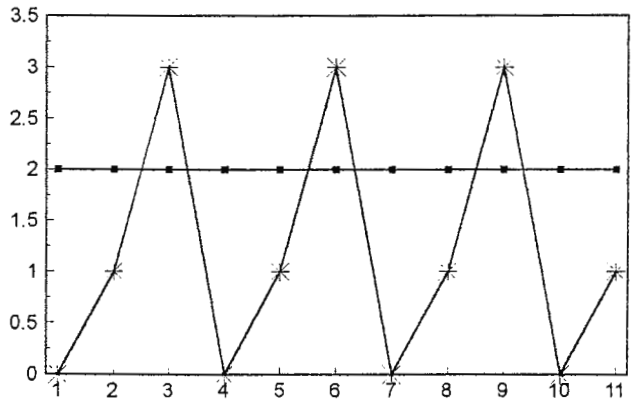


Figura 15.23

Para determinar el orden más adecuado para una función de orden múltiple, se debe seguir un procedimiento algebraico que no considere ningún dato origen en particular. En algunas ocasiones ésto no es posible ya que existen datos específicos para los valores de los estados actual y siguiente de la función, y éstos no pueden ser definidos por una ecuación algebraica en forma clara. En estos casos el orden de la función deberá ser acordado como el mínimo común múltiplo de los distintos órdenes que presenta. Con

esto se garantizará que sea cual fuere el dato origen, la función siempre lo retornará al cumplir el número de iteraciones que su orden indica.

Para terminar, basta recordar lo dicho al principio del presente capítulo: El objeto de esta exposición es simplemente ilustrar el método a seguir para la creación del algoritmo de encriptamiento original que se implementará en este proyecto de graduación. La Teoría de las funciones Iterativas Recursivas es súmamente útil para la creación de algoritmos simétricos de encriptamiento.

Todo algoritmo simétrico es siempre una función iterativa recursiva de orden 2 o al menos de orden par; la primera iteración, o la primera mitad de las iteraciones transforma el formato de la información expresada como números hexadecimales, y la segunda iteración o mitad de las iteraciones devuelve el valor origen que inició el proceso, tal como lo muestra la siguiente figura:

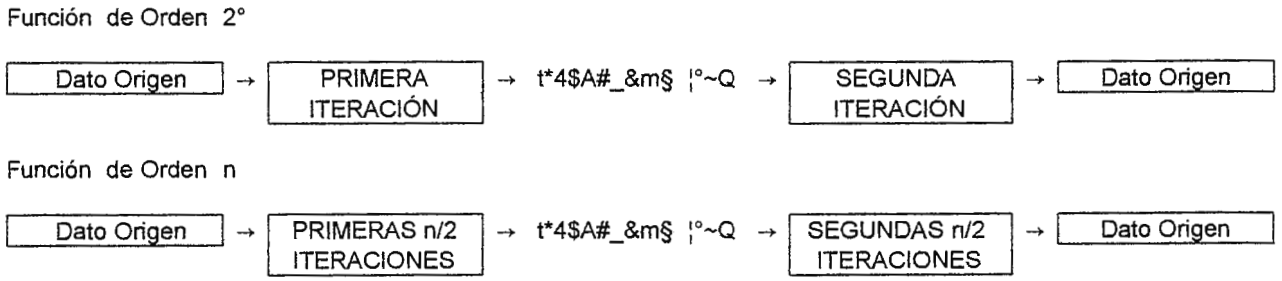


Figura 15.24

Se debe recordar que se trata de dos etapas completamente idénticas puesto que se utiliza una misma función durante todo el proceso (aunque ésta puede tener varias sub-etapas en cada iteración), y que el período en el cual desarrolla su ciclo debe ser un número par, puesto que de no serlo, las etapas de encriptamiento y desencriptamiento no serían iguales, y consiguientemente el algoritmo ya no sería simétrico.

No se debe de perder de vista que ésta función iterativa recursiva debe de generar un valor intermedio entre las etapas primera y segunda que sea lo más diferente posible del dato origen , puesto que de generar un valor con algún grado de semejanza o sencilla correlación ofrecería muy poca protección para el mensaje secreto.

Recuérdese que no se va a encriptar todo el mensaje como un sólo número hexadecimal (esto agotaría la memoria de cualquier dispositivo rápidamente), sino que se trabajará con bloques pequeños, cada uno de los cuales va a ser procesado por la función de encriptamiento. Esto abre la posibilidad a que cada bloque sea operado en la función de encriptamiento con parámetros modificadores individuales, ordenados secuencialmente para cada operación respectiva (a este parámetro modificador se le conoce como clave).

Por último, no se debe olvidar que para propósitos de encriptamiento las funciones iterativas recursivas deben trabajar en un rango de igual tamaño a su dominio, sin generar valores que excedan la capacidad de las variables de entrada o salida de la función. Recuérdese que en este campo no son útiles las aproximaciones, las variables en proceso deben ser enteras y sujetarse a los límites de su entorno de procedimientos.

Este capítulo no pretende ser un curso completo de la teoría de las funciones iterativas recursivas, se invita al lector a intentar el desarrollo de nuevas funciones para comprender mejor el funcionamiento de los algoritmos simétricos de encriptamiento, y poder crear uno a la medida de sus aspiraciones.

Todos los conceptos vertidos en este capítulo son importantes para la correcta comprensión del siguiente apartado. Es necesario el entendimiento exacto y completo de las funciones iterativas recursivas ya que basado en ellas se muestra el funcionamiento de la aplicación de encriptamiento que se desarrolla en este proyecto de graduación.

16.- DESARROLLO DE APLICACIÓN EN SOFTWARE.

Como reza el título del presente trabajo de graduación, se incluye dentro del documento una aplicación de software para encriptamiento. A continuación se expone detalladamente el funcionamiento de dicha aplicación. Se comienza con todas sus especificaciones, sus recursos, una exposición rápida para explicar lo que hace la aplicación, el pseudocódigo de la aplicación, el código fuente del programa redactado en lenguaje C, y una breve explicación para su uso.

La aplicación de software involucrará un algoritmo de encriptamiento en clave secreta, se ha diseñado el algoritmo para ser usado tanto para encriptamiento como para desencriptamiento, corriendo el mismo programa sin agregar ninguna opción a escoger más que la clave de encriptamiento. Para lograr este efecto se ha utilizado una función iterativa recursiva la cual necesita operarse en forma recurrente dos veces consecutivas para retornar el valor origen que se intervino en la primera operación.

16.1 PARÁMETROS, ESPECIFICACIONES, TIPOS Y MODOS DE ENCRIPTAMIENTO UTILIZADOS EN LA APLICACIÓN.

- La aplicación realiza encriptamiento punto a punto. Se encripta la información desde el punto origen, señalando el archivo a proteger y se le sustituye con el archivo encriptado (sin cambiar el nombre del archivo), no se interviene de manera alguna en los protocolos de transferencia de datos.
- La aplicación hace uso de la clave secreta para generar el algoritmo de encriptamiento.
- Con el fin de hacer difícil el desencriptamiento por medio de fuerza bruta, la clave de encriptamiento tiene una longitud de 128 bits.
- El encriptamiento se lleva a cabo encriptando bloques de texto fuente con una longitud de 128 bits.
- El documento de texto fuente es obligado a tener una longitud de n * 128 bits (n un número entero), para esto se genera un relleno de bits con ceros en el último bloque incompleto, y se genera un bloque con 120

bits en estado cero, y en los últimos 8 bits se anota al final el número de bits empleados para la justificación del bloque anterior.

- Con el fin de ocupar un sólo algoritmo para encriptar y desencriptar, se utiliza una variación del modo de encriptamiento de retroalimentación de salida de clave, ya que con este modo se genera una secuencia de claves de encriptamiento con un elemento correspondiente para cada bloque de texto, y esta secuencia que se genera, encripta en su primera operación en función EXOR con el texto fuente, y desencripta en su segunda operación en función EXOR con el texto cifrado.
- El bloque de texto no sólo es encriptado en función EXOR con la secuencia de la clave, también es fraccionado, rotado y recombinado en función EXOR entre sus fracciones, antes de realizar la operación EXOR con la clave .
- Con el fin que el texto encriptado no presente en alguna manera semejanzas con el texto fuente, la clave es transpuesta, antes de cada combinación con cada bloque, en todos sus bits de acuerdo a lo indicado por una matriz de transposición, y además es rotada en un bit hacia la derecha. con esto se asegura que el ciclo de la secuencia de la clave de encriptamiento se repita hasta dentro de $(128)^2$ ciclos, y para hacer que este ciclo no se repita tan rápido, además se le suma un uno, con lo cual el período del ciclo anterior se multiplica $(2)^{128}$ veces (que es el número de veces que se tendría que sumar un uno para obtener de nuevo el mismo valor de la clave).
- El texto encriptado no presentará ningún dato acerca del archivo que encriptó. Ni vector de inicialización, ni información de relleno, ni siquiera qué algoritmo lo encriptó.
- El programa utilizará un archivo temporal para escribir cada bloque encriptado a continuación del anterior. Terminado el encriptamiento de todos los bloques guardará el contenido del archivo temporal en el archivo original, y anulará todos los datos del original.
- El número máximo de bloques a encriptar estará limitado por lo que pueda guardar en uno de sus medios de almacenamiento la máquina que esté desarrollando la aplicación. El programa de encriptamiento sólo cargará y guardará bloques de 128 bits en forma sucesiva hasta terminar de encriptar el archivo señalado.

16.2 ALGORITMO DE ENCRIPTAMIENTO A UTILIZAR.

Las funciones matemáticas presentadas en el capítulo anterior constituyen una herramienta indispensable para la creación del algoritmo de encriptamiento. Como se especificó en el apartado anterior,

se tiene el objetivo de crear un algoritmo en clave secreta el cual pueda ser usado tanto para encriptar como para desencriptar sin ordenar bajo ningún comando algún cambio en su funcionamiento.

La característica anterior en los algoritmos de clave secreta es casi una norma de funcionamiento para la mayoría de los métodos de encriptamiento simétricos. Casi todas las aplicaciones de encriptamiento consisten de un algoritmo que cifra la información en su primera operación al texto, y la descifra al momento de desarrollar una segunda operación. En términos matemáticos se puede explicar este tipo de algoritmos como una función iterativa recursiva de segundo grado, y con un alto grado de complejidad.

Cumplir el anterior requisito no es la característica más importante de la clave secreta. Es necesario también desarrollar un alto grado de confusión y difusión en el formato de la información encriptada para que dicha representación tenga en apariencia un orden completamente aleatorio que no presente el más mínimo indicio del contenido del texto fuente, ni tampoco de un ciclo que indique el avance de la función iterativa recursiva durante su primera iteración.

Para Cumplir con esta tradición de la mayoría de los algoritmos simétricos, la aplicación a desarrollar en este proyecto hará uso de varias funciones iterativas recursivas en forma sucesiva. Esto cambiará el formato de la información al mismo tiempo que creará en un método de desencriptamiento a la vez que desarrolla el de encriptamiento,

Se utilizarán cuatro funciones iterativas recursivas, las cuales son:

- ROTACIÓN DE LOS MEDIOS BLOQUES DE 64 BITS, 32 BITS HACIA LA DERECHA. Esto quiere decir, que los dos cuartos de bloques de los cuales está compuesto cada medio bloque serán intercambiados en sus posiciones, de manera que el primer cuarto de bloque ocupará la posición del segundo, y el segundo la posición del primero. Gráficamente esto puede ser detallado mediante la siguiente figura:

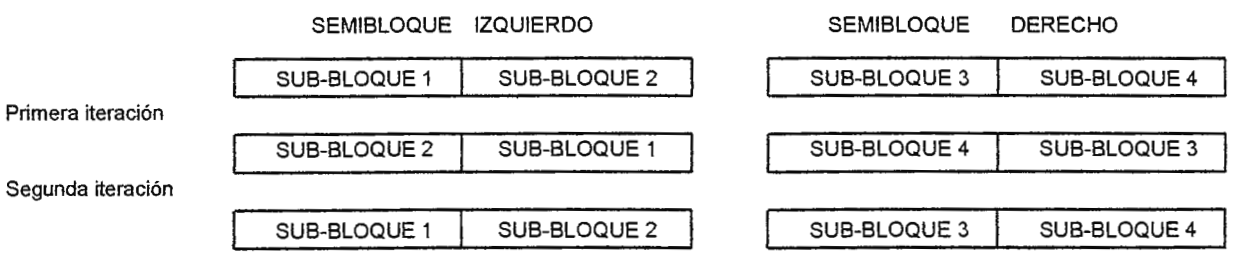


Figura 16.1

➢ OPERACIÓN EN FUNCIÓN EXOR DE MEDIOS BLOQUES SUCESIVOS. Esto es, cada medio bloque se opera en EXOR con el bloque adyacente a la izquierda. El primer medio bloque no sufre ninguna variación. Gráficamente esto puede ser detallado mediante la siguiente figura:

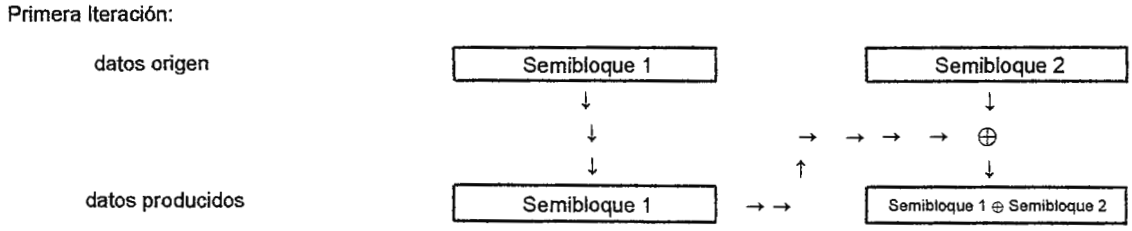


Figura 16.2

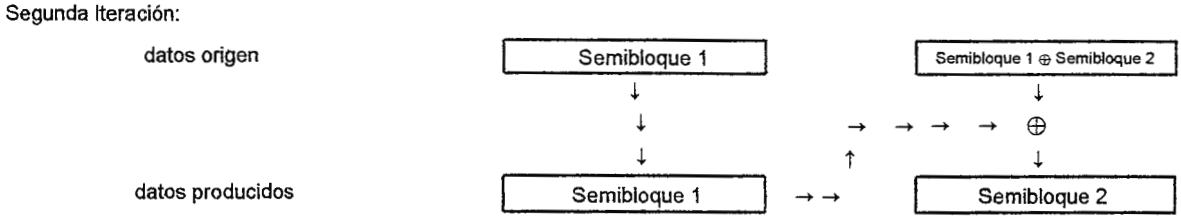


Figura 16.3

➢ ROTACIÓN DEL BLOQUE 64 BITS HACIA LA DERECHA. Esto quiere decir, que los dos semibloques de los cuales está compuesto cada bloque serán intercambiados en sus posiciones, de manera que el primer semibloque ocupará la posición del segundo, y el segundo la posición del primero. Gráficamente esto puede ser detallado mediante la siguiente figura:

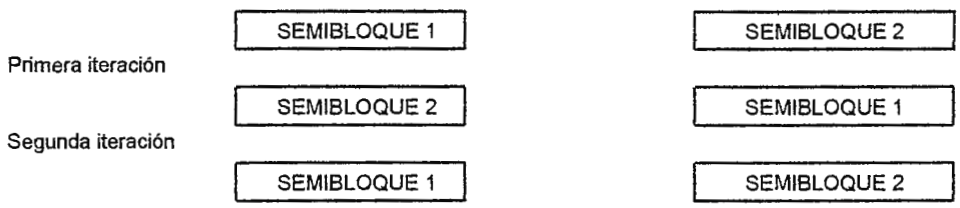


Figura 16.4

➢ OPERACIÓN EN FUNCIÓN EXOR CON OTRO NÚMERO BINARIO. El bloque entero de 128 bits será operado completo en función EXOR con un número clave K de igual longitud el cual al ser operado bit a bit devolverá un valor encriptado en la primera iteración, y reproducirá el bloque íntegro en la

segunda. Esta función puede ser realizada n veces, y para efectos de manipulación algebraica considerar a los n números como uno sólo K producto de la operación EXOR de todos sus componentes.

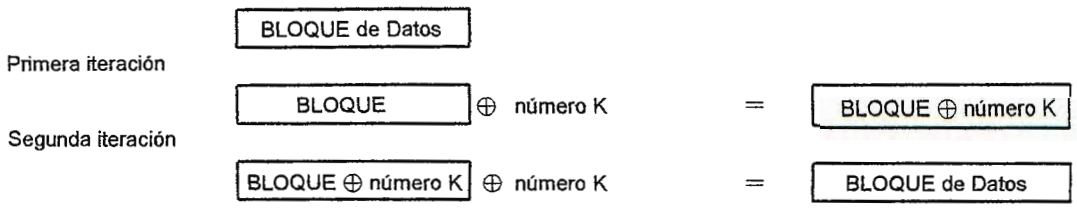


Figura 16.5

Como se explicó en el capítulo anterior, la combinación sucesiva de varias funciones iterativas recursivas siempre da como resultado una función de la misma naturaleza, pero la cual no necesariamente deberá tener el mismo orden de las que la generaron. Para averiguar el orden de la función que se ha producido como resultado se debe realizar un procedimiento de sustitución algebraica en cada paso de la función, siguiendo este procedimiento en tantas iteraciones como fueran necesarias para reconstruir el bloque origen.

A continuación se muestra este procedimiento algebraico realizado en la sucesión de funciones que se escogieron para realizar el algoritmo de encriptamiento. Se ha querido representar el bloque origen de texto fuente como compuesto de cuatro sub-bloques de 32 bits denominados como SB1, SB2, SB3 y SB4, así mismo se ha representado a los números claves para operar en EXOR como un número K de 128 bits compuesto por K1, K2, K3 y K4.

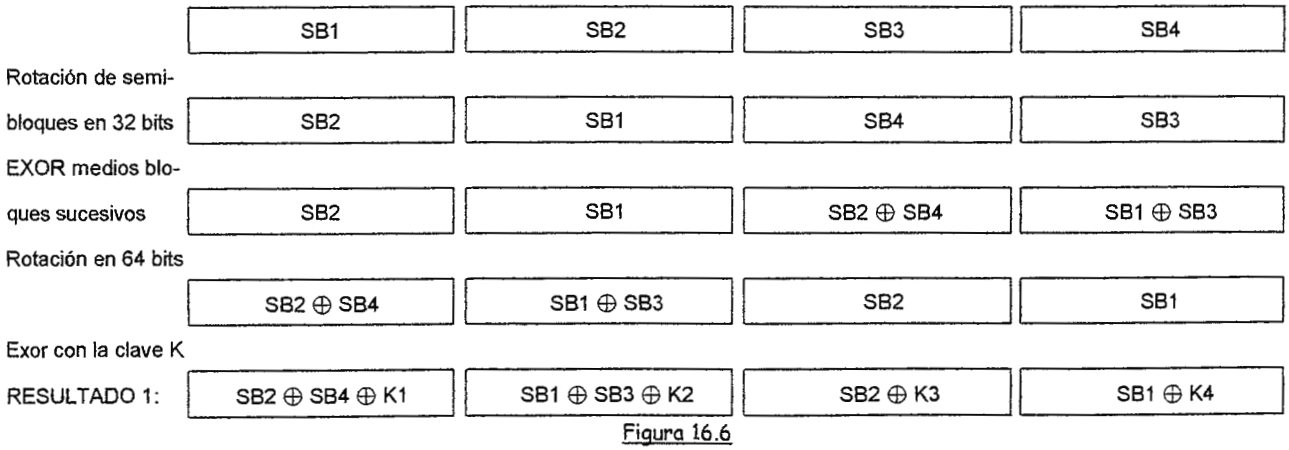


Figura 16.6

Rotación de semi-bloques en 32 bits	SB1 ⊕ SB3 ⊕ K2	SB2 ⊕ SB4 ⊕ K1	SB1 ⊕ K4	SB2 ⊕ K3
EXOR medios blo-ques sucesivos	SB1 ⊕ SB3 ⊕ K2	SB2 ⊕ SB4 ⊕ K1	SB3 ⊕ K2 ⊕ K4	SB4 ⊕ K1 ⊕ K3
Rotación en 64 bits	SB3 ⊕ K2 ⊕ K4	SB4 ⊕ K1 ⊕ K3	SB1 ⊕ SB3 ⊕ K2	SB2 ⊕ SB4 ⊕ K1
Exor con la clave K				
RESULTADO 2:	SB3 ⊕ K1 ⊕ K2 ⊕ K4	SB4 ⊕ K1 ⊕ K2 ⊕ K3	SB1 ⊕ SB3 ⊕ K2 ⊕ K3	SB2 ⊕ SB4 ⊕ K1 ⊕ K4

Figura 16.7

Rotación de semi-bloques en 32 bits	SB4 ⊕ K1 ⊕ K2 ⊕ K3	SB3 ⊕ K1 ⊕ K2 ⊕ K4	SB2 ⊕ SB4 ⊕ K1 ⊕ K4	SB1 ⊕ SB3 ⊕ K2 ⊕ K3
EXOR medios blo-ques sucesivos	SB4 ⊕ K1 ⊕ K2 ⊕ K3	SB3 ⊕ K1 ⊕ K2 ⊕ K4	SB2 ⊕ K2 ⊕ K3 ⊕ K4	SB1 ⊕ K1 ⊕ K3 ⊕ K4
Rotación en 64 bits	SB2 ⊕ K2 ⊕ K3 ⊕ K4	SB1 ⊕ K1 ⊕ K3 ⊕ K4	SB4 ⊕ K1 ⊕ K2 ⊕ K3	SB3 ⊕ K1 ⊕ K2 ⊕ K4
Exor con la clave K				
RESULTADO 3:	SB2⊕ K1⊕ K2⊕ K3⊕ K4	SB1⊕ K1⊕ K2⊕ K3⊕ K4	SB4 ⊕ K1 ⊕ K2	SB3 ⊕ K1 ⊕ K2

Figura 16.8

Rotación de semi-bloques en 32 bits	SB1⊕ K1⊕ K2⊕ K3⊕ K4	SB2⊕ K1⊕ K2⊕ K3⊕ K4	SB3 ⊕ K1 ⊕ K2	SB4 ⊕ K1 ⊕ K2
EXOR medios blo-ques sucesivos	SB1⊕ K1⊕ K2⊕ K3⊕ K4	SB2⊕ K1⊕ K2⊕ K3⊕ K4	SB1⊕ SB3⊕ K3⊕ K4	SB2⊕ SB4⊕ K3⊕ K4
Rotación en 64 bits	SB1⊕ SB3⊕ K3⊕ K4	SB2⊕ SB4⊕ K3⊕ K4	SB1⊕ K1⊕ K2⊕ K3⊕ K4	SB2⊕ K1⊕ K2⊕ K3⊕ K4
Exor con la clave K				
RESULTADO 4:	SB1⊕SB3⊕ K1⊕K3⊕ K4	SB2⊕ SB4⊕K2⊕K3⊕ K4	SB1⊕ K1⊕ K2⊕ K4	SB2⊕ K1⊕ K2⊕ K3

Figura 16.9

Rotación de semi-bloques en 32 bits	SB2⊕ SB4⊕K2⊕K3⊕ K4	SB1⊕SB3⊕ K1⊕K3⊕ K4	SB2⊕ K1⊕ K2⊕ K3	SB1⊕ K1⊕ K2⊕ K4
EXOR medios blo-ques sucesivos	SB2⊕ SB4⊕K2⊕K3⊕ K4	SB1⊕SB3⊕ K1⊕K3⊕ K4	SB4 ⊕ K1⊕ K4	SB3⊕ K2 ⊕ K3
Rotación en 64 bits	SB4 ⊕ K1⊕ K4	SB3⊕ K2 ⊕ K3	SB2⊕ SB4⊕K2⊕K3⊕ K4	SB1⊕SB3⊕ K1⊕K3⊕ K4
Exor con la clave K				
RESULTADO 5:	SB4 ⊕ K4	SB3 ⊕ K3	SB2⊕ SB4⊕K2⊕ K4	SB1⊕SB3⊕ K1⊕K3

Figura 16.10

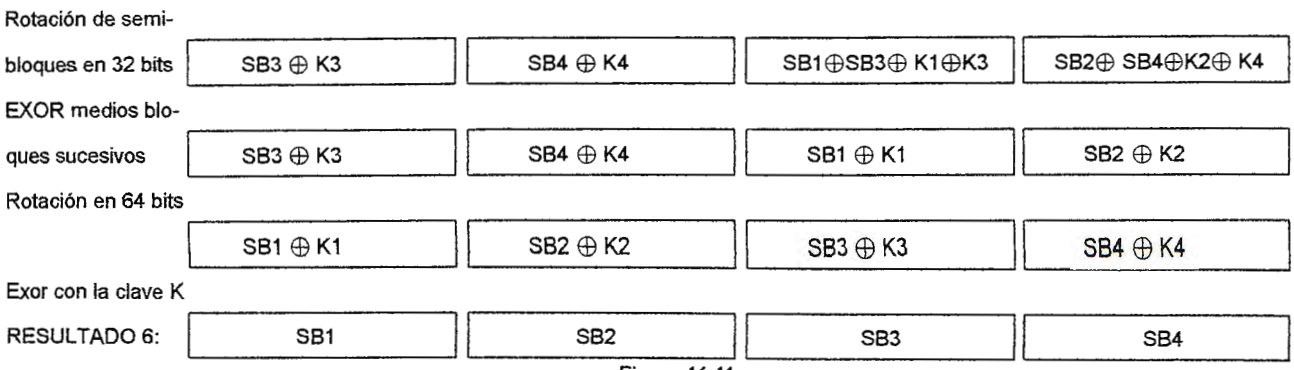


Figura 16.11

Cabe destacar que esta combinación de funciones fue escogida de entre muchas otras por tener como resultado una función iterativa recursiva de orden par. Obsérvese que al ser de sexto orden cabe la posibilidad de que tanto el algoritmo de encriptamiento como el de desencriptamiento posean tres iteraciones idénticas de esta combinación. De manera que la primera vez que la función definida como las tres iteraciones sea desarrollada en el texto fuente, se genere un texto cifrado, y la segunda vez que esta misma función opere el texto producido en la iteración anterior se recupere el texto original.

El número binario K que representa la clave de encriptamiento no es un número fijo, si lo fuera los caracteres repetidos producirían un texto cifrado con códigos repetidos. Este número evoluciona en cada iteración de acuerdo al siguiente procedimiento, el cual se produce antes de ser operado en función EXOR con el bloque transformado:

1. A la clave se le suma uno.
2. La clave es rotada 1 bit hacia la derecha.
3. La clave sufre la transposición de todos sus bits. Esto se desarrolla considerando a la clave como formada de 16 bytes, cada uno de los bits de cada byte tomará el estado en el cual se encuentra el bit que se le ordena a continuación:
    - El bit 7 tomará su estado de la posición 1 del byte situado dos bytes atrás.
    - El bit 6 tomará su estado de la posición 0 del byte actual.
    - El bit 5 tomará su estado de la posición 2 del byte situado cuatro bytes atrás.
    - El bit 4 tomará su estado de la posición 6 del byte anterior.
    - El bit 3 tomará su estado de la posición 4 del byte situado tres bytes atrás.
    - El bit 2 tomará su estado de la posición 7 del byte situado ocho bytes atrás.

- El bit 1 tomará su estado de la posición 6 del byte situado cinco bytes atrás.
- El bit 0 tomará su estado de la posición 5 del byte situado seis bytes atrás.

La transposición de cada bit no ha sido completamente al azar. Se ha tenido especial cuidado de que ningún bit sea tomado de una posición con igual jerarquía dentro de la cifra hexadecimal que está representando, esto obliga a que todas las cifras cambien (siempre que los bits origen y destino de una determinada cifra no sean correspondientemente iguales).

Todo este procedimiento puede ser comprendido fácilmente observando con atención el diagrama mostrado en la siguiente figura:

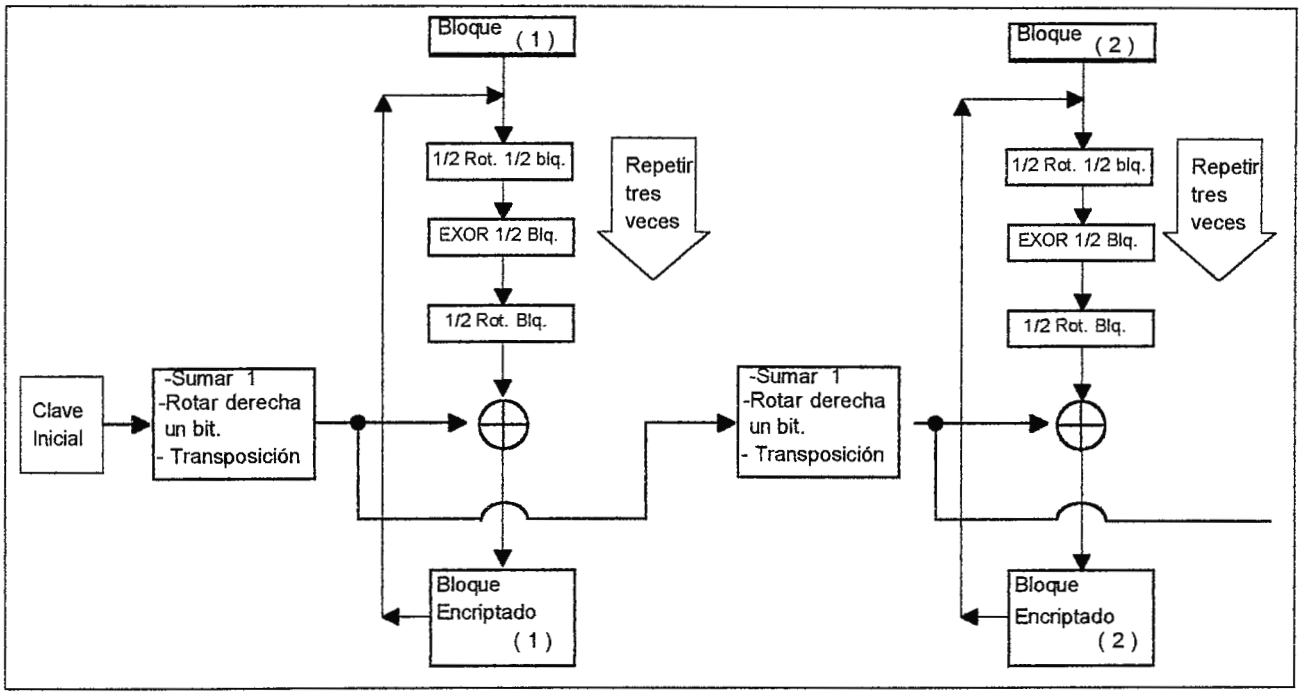


Figura 16.12

16.2 ALGORITMO DE LA APLICACIÓN EN PSEUDOCÓDIGO.

A continuación se presenta una versión castellana de lo que expresa el código fuente en lenguaje C ofrecido en el siguiente apartado. Se ha querido exponer este programa en lenguaje coloquial para la mejor comprensión del mecanismo de operación del algoritmo en lenguaje C, ya que los códigos de programación no siempre son fácilmente legibles, y además casi siempre presentan algún comando que no es conocido dentro de las versiones manejadas por el lector. Obsérvese que no sólo se expone el procedimiento del

manejo de bits del archivo a encriptar, sino también el manejo del archivo, el cual no necesariamente debe tener un tamaño que sea múltiplo de 128 bits, y por tanto deberá ser ajustado en su longitud con un relleno de bits para completar bloques exactos de 128 bits, encriptado, y luego en el proceso inverso se reconocerán los bits de relleno[40] para ser retirados y presentar el archivo con toda la integridad que tenía al principio de todo el proceso.

1. Presentación de cuadro de diálogo
2. Petición del archivo a encriptar por su nombre.
3. Abra el archivo a encriptar y cree el archivo temporal para escribir
4. Petición de la clave de encriptamiento, guardarla como {clave}
5. Guarde {clave } en 16 bytes conocidos como { clave[0] } hasta { clave[15] }
6. Contar el número de bits o Bytes totales y guarde en la variable {longitud_del_archivo}.
7. // ********** INICIO DE LAZO PRINCIPAL DE ENCRIPTAMIENTO ********** //
8. Cargue 16 bytes del archivo en las variables { bloque[0] } a { bloque[15] }
9. Si no le alcanzan a salir los 16 bytes del archivo, entonces rellene las variables { bloque[x] } que no usó con ceros, marque que ha llegado al fin del archivo y anote el número de datos encontrados en este bloque en la variable { datos_válidos }
10. Asígnese {clave} = {clave} + 0x0000000000000001
11. Rotar a la derecha un bit la variable {clave}
12. Opérese la función TRANSPOSICIÓN ({clave}) y guárdese el resultado en la misma variable {clave}
13. // ** INICIO DEL LAZO DE LAS ITERACIONES DE LA FUNCIÓN DE ENCRIPTAMIENTO *** //
14. Realice el siguiente lazo para {iteración} =0 hasta 2
15. Intercambie valores de { bloque[0] } y { bloque[1] }.
16. Intercambie valores de { bloque[2] } y { bloque[3] }.
17. Asígnese { bloque[2] }={ bloque[2] } EXOR { bloque[0] }
18. Asígnese { bloque[3] }={ bloque[1] } EXOR { bloque [1] }
19. Intercambie valores de { bloque [0] } y { bloque[2] }.
20. Intercambie valores de { bloque[1] } y { bloque[3] }.

[40]Este relleno consiste de ceros para completar 128 bits, el siguiente bloque de relleno tiene 120 ceros, y los últimos 8 bits informan cuantos ceros de relleno se han puesto en el bloque anterior. En el desencriptamiento, al reconocer el programa un bloque final con sus primeros 120 bits ceros, revisará si el número especificado en los últimos 8 bits de este corresponde bits en cero del bloque anterior, si esto es cierto anulará todo este relleno y presentará el archivo original sin agregar ni quitar ningún bit.

21. realice el siguiente lazo para {i} = 0 hasta 15
22. Asígnese { bloque[i] } = { bloque[i] } EXOR { clave[i] }
23. Regrese al lazo {i}
24. Regrese al lazo {iteración}
25. // ****** FIN DE LAS TRES ITERACIONES DE LA FUNCIÓN DE ENCRIPTAMIENTO ******//
26.
27. // ********** DEPOSITANDO EL RESULTADO EN ARCHIVO TEMPORAL **********//
28. Si no ha llegado al fin del archivo recoja un byte
29. Si los bytes cargados son menores que {longitud_de_archivo} y además no ha llegado al fin del archivo entonces realice los siguientes 2 pasos:
    a. Si existe bloque anterior guarde { bloque_ant [0] } hasta { bloque_ant [15] } en temporal
    b. copie las variables { bloque [0] } hasta { bloque [15] } en { bloque_ant [0] } hasta { bloque_ant [15] }
30. Si los bytes cargados son mayores que {longitud_de_archivo} y además ha llegado al fin del archivo entonces realice los siguientes pasos:
    a. Si existe bloque anterior guarde { bloque_ant [0] } hasta { bloque_ant [15] } en temporal
    b. copie las variables { bloque [0] } hasta { bloque [15] } en { bloque_ant [0] } hasta { bloque_ant [15] }
    c. Guarde { bloque_ant [0] } hasta { bloque_ant [15] } en temporal
    d. Rellene con ceros las variables { bloque [0] } hasta { bloque [14] } y escriba en { bloque [15] } el valor de la variable { datos_válidos}
    e. Realice el siguiente lazo para {iteración} =0 hasta 2
    f. Intercambie valores de { bloque[0] } y { bloque[1] }.
    g. Intercambie valores de { bloque[2] } y { bloque[3] }.
    h. Asígnese { bloque[2] }={ bloque[2] } EXOR { bloque[0] }
    i. Asígnese { bloque[3] }={ bloque[1] } EXOR { bloque [1] }
    j. Intercambie valores de { bloque [0] } y { bloque[2] }.
    k. Intercambie valores de { bloque[1] } y { bloque[3] }.
    l. realice el siguiente lazo para {i} = 0 hasta 15
    m. Asígnese { bloque[i] } = { bloque[i] } EXOR { clave[i] }
    n. Regrese al lazo {i}
    o. Regrese al lazo {iteración}
    p. copie las variables { bloque [0] } hasta { bloque [15] } en { bloque_ant [0] } hasta { bloque_ant [15] }

q. Guarde { bloque_ant [0] } hasta { bloque_ant [15] } en temporal

31. Si los bytes cargados son mayores que {longitud_de_archivo} y además ha llegado al fin del archivo con el último byte leído entonces realice los siguientes pasos:

a. Si los valores de las variables { bloque [0] } hasta { bloque [14] } son ceros, y además los valores desde {bloque_ant[{ bloque [15] }] hasta { bloque_ant[15] } son ceros entonces guarde { bloque_ant [0] } hasta { bloque_ant [bloque [15]] } en temporal

b. De lo contrario:

i. Guarde { bloque_ant [0] } hasta { bloque_ant [15] } en temporal

ii. copie las variables { bloque [0] } hasta { bloque [15] } en { bloque_ant [0] } hasta { bloque_ant [15] }

iii. Guarde { bloque_ant [0] } hasta { bloque_ant [15] } en temporal

32. Si todavía los bytes cargados son menores que {longitud_de_archivo} entonces regrese hasta el *** *INICIO DEL LAZO PRINCIPAL DE ENCRIPTAMIENTO* ***

33. // ********** FIN DE LAZO PRINCIPAL DE ENCRIPTAMIENTO ********** //

34. Cierre el archivo Temporal.

35. borre el archivo original de donde sacó el texto que acaba de encriptar

36. cambie el nombre del archivo Temporal por el nombre del archivo original de donde sacó el texto que acaba de encriptar

37. fin.

38. //********** RUTINA DE TRANSPOSICIÓN **********//

39. Cada uno de los bits de cada byte de las variables { clave [x] }tomará el estado en el cual se encruentra el bit que se le ordena a continuación:

- El bit 7 tomará su estado de la posición 1 del byte situado dos bytes atrás.
- El bit 6 tomará su estado de la posición 0 del byte actual.
- El bit 5 tomará su estado de la posición 2 del byte situado cuatro bytes atrás.
- El bit 4 tomará su estado de la posición 6 del byte anterior.
- El bit 3 tomará su estado de la posición 4 del byte situado tres bytes atrás.
- El bit 2 tomará su estado de la posición 7 del byte situado ocho bytes atrás.
- El bit 1 tomará su estado de la posición 6 del byte situado cinco bytes atrás.
- El bit 0 tomará su estado de la posición 5 del byte situado seis bytes atrás.

40. // ** Nota: todo lo anterior se hará separando el valor de cada bit mediante la operación lógica AND con los números hexadecimales 01, 02, 04, 08, 10, 20, 40 y 80 y recombinando cada bit dentro del nuevo byte con la operación lógica OR sucesiva del valor de cada bit en AND con su respectivo valor 01, 02, 04, 08, 10, 20, 40 o 80 que representa su nueva posición dentro de su nuevo byte.**//

16.3 ALGORITMO DE LA APLICACIÓN EN CÓDIGO FUENTE EN LENGUAJE C.

El programa listado a continuación está hecho para ser desarrollado por el compilador de lenguaje C: Microsoft Visual C ++ versión 1.00 (1993), y dentro de este sistema para ser compilado como el código fuente " *********.cpp " en un proyecto del tipo Quick Win Aplication (. EXE) . El archivo ejecutable que produce el código fuente está hecho para ser ejecutado en el sistema operativo Windows 95 , Windows 98, o Windows NT. Las instrucciones para el uso de este ejecutable están especificadas en el siguiente apartado.

```
/* ***********************************************
          Código fuente de RAPICRIP
                24 de abril de 1999
          José Armando Hernández Saavedra
*********************************************** */

#include <iostream.h>
#include <iomanip.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <io.h>
#include <fcntl.h>
#include <sys\stat.h>
#include <sys\types.h>

// ***********************************************
// Definición de variables globales
// ***********************************************

char bloque[16];
char bloque_ant[16];
char clave[16];
char ENTRADA_CLAVE[16];
int   clave_hex[16];
char* caracter;
int  incompleto;
int  bloque_anterior = 0;
int  datos_validos = 0;
int  archivo;
int  temporal;
char nombre_archivo[80];
char nombre_temporal[80];
int  longitud_direccion=0;
char temp[12]={'t','e','m','p','o','r','a','l','.','t','m','p'};
char bits_entrada[16][8];
char bits_salida[16][8];
int  bytes[]= {10, 11, 8, 13, 15, 12, 0, 14};
int  bits[] = { 5,  3, 7,  4,  6,  2, 0,  1};
//            {lsb.....................msb}
char  hexadec[16]={'0','1','2','3','4','5','6','7','8','9','a','b','c','d','e','f'};
char  HEXA[16]={'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};
char obtener_bit(char, int);
unsigned long bytes_cargados = 0;
unsigned long longitud_del_archivo=0;
int  lectura;
char  forma_clave;
char* clave_hexadecimal;
char CLAVE_HEX[32];
```

```
// ***********************************************
// Prototipo de funciones
// ***********************************************

void apertura();
void encriptar();
void cierre();
void leer_datos();
void guardar_datos(int);
void lazo_encriptamiento();
void verificacion_almacenamiento();
void copiar_en_anterior();
void cambio_clave();
void intercambiar(char &, char &);
void clave_hdec();
void clave_caracteres();
void ruta_temporal();

// ***********************************************
// Programa principal
// ***********************************************

main()
{
        apertura();
        encriptar();
        cierre();
        cout << " \n \n \n \n \n \n \n \n \n \n ";
        cout << " \n \n \n \n \n \n \n \n \n \n ";
        cout << " \n \n \n \n \n \n \n \n \n \n ";
        cout << " \n \n \n \n \n \n \n \n \n \n ";
        cout << "\t      E L   A R C H I V O \n \n                    ";
        cout << nombre_archivo;
        cout << "\n \n \t H A   S I D O   P R O C E S A D O";
        cout << " \n \n \n \n";
        return 0;
}

// ***********************************************
// Definición de las funciones
// ***********************************************
void apertura()
{
        cout << "\n   Introduzca el nombre del archivo a encriptar\n";
        cout <<   "       o desencriptar con su ruta de acceso\n \n         ";
        cin >> nombre_archivo;
    cout << " \n \n";
    ruta_temporal();
//************************ Longitud del archivo
    FILE *fp;
    int fhandle;
    fp = fopen (nombre_archivo, "r");
    if (!fp)
    {
              cerr << "Error al abrir el archivo: " << nombre_archivo << endl;
              exit(-1);
    }
    fhandle=fileno(fp);
    longitud_del_archivo = filelength(fhandle);
    fclose(fp);

//********************************** Aperturas
    archivo  = _open( nombre_archivo, _O_BINARY | _O_RDONLY );
    if (!archivo)
    {
              cerr << "Error al abrir el archivo: " << nombre_archivo << endl;
              exit(-1);
     }
     temporal = _open( nombre_temporal , _O_BINARY | _O_RDWR |  _O_CREAT , _S_IWRITE | _S_IREAD);
     if (!temporal)
        {
              cerr << "Error al abrir el archivo temporal\n\n" << endl;
```

```
                exit(-1);
        }

//********************* Adquisisión de la clave
     cout << "\n   ¿Escribirá la clave con caracteres \n ";
     cout << "  o con cifras hexadecimales ?  ( c / h )  \n";
     cout << "\n              ";
     cin  >> forma_clave;

     if (forma_clave == 'h' | forma_clave == 'H' )
     {
                clave_hdec();
      }
      else
      {
       clave_caracteres();
       }
}

//************************************************
void ruta_temporal()
{
        int i=79;
        int ruta=0;
        while (!ruta && i>=0 )
        {
                ruta = (int) ( nombre_archivo[i] ==  '\\'  |  nombre_archivo[i] == ':'  |
nombre_archivo[i] == '/'  );     //188
                i= i-1;
        }
        if (ruta)
        {
                longitud_direccion = i+1 ;
                int j;
                for ( j=0 ; j<=longitud_direccion ; j++ )
                {
                        nombre_temporal[j] = nombre_archivo[j] ;
                }
                int k;
                for (k=longitud_direccion+1 ; k<=longitud_direccion+12 ; k++ )
                {
                        nombre_temporal[k] = temp[k-longitud_direccion-1];
                }
        }
    else
    {
       int k;
                for (k=0 ; k<12 ; k++ )
                {
                        nombre_temporal[k] = temp[k];
                }
    }

}


//************************************************
void clave_hdec()

{
        cout << "Introduzca la clave con 32 cifras hexadecimales  \n";
        cin >> CLAVE_HEX ;
        int i=0;
        int j;
        int y=0;
        int z=0;
        while (CLAVE_HEX[y]!='\0' && CLAVE_HEX[z]!='\0'  && i<16)
        {
                y=i*2;
                z=i*2+1;
                for (j=0 ; j<16 ; j++)

                {
```

```
                        if ( ( CLAVE_HEX[y] == hexadec[j] ) | ( CLAVE_HEX[y] == HEXA[j] ) )
        //139
                                clave_hex[i] = j*16 ;
                        if ( ( CLAVE_HEX[z] == hexadec[j] ) | ( CLAVE_HEX[z] == HEXA[j] ) )
                                clave_hex[i] += j ;
                }
                clave[i] = (char) clave_hex[i];
                i++;
        }
        for (j=i; j<16; j++)
        {
                clave[j] = 16*(j-1)+j-i;
        }
}
//************************************************
void clave_caracteres ()
{
        cout << "Introduzca la clave con 16 caracteres: \n";
        cin >> ENTRADA_CLAVE;
        int i = 0;
        while (ENTRADA_CLAVE[i]!='\0' && i<16)
        {
                clave[i]=ENTRADA_CLAVE[i];
                i++;
        }
        int j;
        for (j=i; j<16; j++)
        {
                clave[j] = 16*(j-1)+j-i;
        }
}

//************************************************
void encriptar()
{
        while ( bytes_cargados <= longitud_del_archivo )
        {
                leer_datos();
                lazo_encriptamiento();
                verificacion_almacenamiento();
        }
}

// ***********************************************
void cierre()
{
        _close(archivo);
        _close(temporal);
    remove(nombre_archivo);
    rename(nombre_temporal, nombre_archivo);
}

// ***********************************************
void leer_datos()
{
        int i=0, j=0 ;
        if ( !bloque_anterior )
        {

                lectura = _read ( archivo , caracter , 1 );
                bytes_cargados++;
        }
        if ( bytes_cargados<=longitud_del_archivo )
        {
                while (bytes_cargados<=longitud_del_archivo && i<16)
                {
                        bloque[i] = caracter[0];
                        i++;
                        if (i<16)
                        {
                                lectura = _read ( archivo , caracter , 1 );
                                bytes_cargados++;
                }
```

```
                }

                if (i!=16)
                {
                        incompleto = 1;
                        datos_validos = i;
                        for (j=i ; j<16; j++)
                        {
                        bloque[j]=0;
                        }
                }
        }
}

// ***********************************************
void lazo_encriptamiento()
{
        int i, j;
        cambio_clave();
        for ( j=0; j<3; j++)
        {
                for ( i=0; i<4; i++)
                {
                        intercambiar( bloque[i], bloque[i+4]);
                        intercambiar( bloque[i+8], bloque[i+12]);
                }

                for ( i=0; i<8; i++)
                {
                        bloque[i+8]^=bloque[i];
                }
                for ( i=0; i<8; i++)
                {
                        intercambiar( bloque[i], bloque[i+8]);
                }
                for ( i=0; i<16; i++)
                        bloque[i]^=clave[i];
        }
}

// ***********************************************
void verificacion_almacenamiento()
{
         if ( !incompleto )
        {
                lectura = _read ( archivo , caracter , 1 );
                bytes_cargados++;
        }
        if (bytes_cargados<=longitud_del_archivo && !incompleto)
        {
                if (bloque_anterior)
                        guardar_datos(16);
                copiar_en_anterior();
                bloque_anterior=1;
        }
        if (bytes_cargados>longitud_del_archivo && incompleto)
        {
                if ( bloque_anterior )
                        guardar_datos(16);
                copiar_en_anterior();
                guardar_datos(16);
                int i=0;
                for ( i=0 ; i<15 ; i++ )
                        bloque[i]=0;
                bloque[15] = datos_validos;
                lazo_encriptamiento();
                copiar_en_anterior();
                guardar_datos(16);
        }
        if (bytes_cargados>longitud_del_archivo && !incompleto)
        {
                char b1=0 , b2=0;
                int i=0;
```

```
                if (bloque_anterior)
                {
                        for (i=0 ; i<15 ;i++ )
                                b1|=bloque[i];
                        if ( !b1)
                        {
                                int i;
                                for (i=bloque[15] ; i<16 ; i++ )
                                        b2 |= bloque_ant[i];
                        }
                        if (!b1 && !b2)
                                guardar_datos( bloque[15] );
                        else
                        {
                                guardar_datos(16);
                                copiar_en_anterior();
                                guardar_datos(16);
                        }
                }
                else
                {
                        copiar_en_anterior();
                        guardar_datos(16);
                        bloque_anterior=1;
                }
        }
}

// ***********************************************
void cambio_clave()
{
        int i;
        int j;
        char primer_bit, ultimo_bit;

//************************ Sumar uno a la clave
        clave[15]++;
        for (i=14;i>=0;i--)
                if (clave[i+1] ==0  && clave[i] ==0)
                {
                        clave[i]++;
                }

//********************************** Rotación
        primer_bit=( (clave[0]&0x80)? 0x01: 0x00);
        for (i=15;i>=0;i--)
        {
                ultimo_bit=( (clave[i]&0x80)? 0x01: 0x00);
                clave[i]<<=1;
                clave[i]|=primer_bit;
                primer_bit=ultimo_bit;
        }

//****************************** Transposición
        for (i=0;i<16;i++)
                for (j=0;j<8;j++)
                        bits_entrada[i][j] = obtener_bit(clave[i],j);

        for (i=0;i<8;i++)
                for (j=0;j<16;j++)
                        bits_salida[j][i]=bits_entrada[(j+bytes[i])%16][bits[i]];

        for (i=0; i < 16; i++)
        {
                clave[i]=0;
                for (j=7; j>=0; j--)
                        clave[i] += (char)(( bits_salida[i][j] )*pow(2,j));
        }

}

// ***********************************************
void intercambiar(char &x, char &y)
```

```
{
        x^=y;
        y^=x;
        x^=y;
}

// ***********************************************
char obtener_bit(char dato, int l)
{
        return ((dato & (char) pow(2,l))? 1 : 0);
}

// ***********************************************
void copiar_en_anterior()
{
        int i=0;
        for ( i=0 ; i<16 ; i++)
                bloque_ant[i]=bloque[i];
}

// ***********************************************
void guardar_datos(int z)
{
        int i;
        unsigned byte_escrito;
        char* dato;
        for ( i=0; i<z; i++)
        {
                dato = &bloque_ant[i];
                byte_escrito = _write( temporal, dato, sizeof( char) );
    }
}
```

16.4 INSTRUCCIONES PARA EL USO DE LA APLICACIÓN.

La aplicación está diseñada para encriptar la primera vez que opera un archivo, y desencriptar la segunda vez que opera el mismo archivo. Simplemente respóndase a todas las preguntas que hace el programa: escriba el nombre del archivo a encriptar con todo y su ruta de acceso y la clave en su debido momento. La clave puede ser digitada como 16 caracteres (en cuyo caso se asumirán los códigos ASCII de dichos caracteres), o puede ser digitada por 32 cifras hexadecimales comprendidos entre 0 y f. [41]

Si no se ocupan 16 bytes con la clave que se digitó no existe ningún motivo de alarma, el resto de los bytes serán completados con valores que son función de lo poco que ya se ha digitado, aún la clave nula (ningún caracter) desarrolla un número que es el resultado de dicha función con un valor de entrada de cero.

El archivo que se produce como salida no tiene ninguna marca que indique si está o no encriptado, esto hace mucho más difícil para el cripto-analista saber qué método de encriptamiento se ha usado. El programa sólo reconocerá que ha desencriptado un archivo cifrado cuando reconozca el relleno de ceros y lo

[41] Si se escribe la clave con caracteres será muy difícil ocupar todos los valores entre 0 y f del primer caracter del código ASCII ya que éste no asume los valores entre 0 y 255, En cambio si se emplea cifras Hexadecimales se asegura utilizar aún aquellos números que no tienen una representación en el código ASCII.

retire, pero no informará al usuario que ha reconocido un archivo desencriptado. Debe ser el usuario quien recuerde que el archivo que está operando lo está desarrollando por primera o por segunda vez.

La aplicación es de encriptamiento en clave secreta, por lo tanto debe ser el usuario quien recuerde qué clave usó la primera vez , de no usar la misma clave la segunda vez, no se conseguirá ningún desencriptamiento.

Si se desea un nivel superior de encriptamiento cífrese con múltiples claves varias veces consecutivas, hágase este procedimiento siguiendo cuidadosamente los siguientes pasos:

1. Opérese el documento con la clave 1.
2. Opérese el documento obtenido en la etapa anterior con la clave 2.
3. Opérese el documento obtenido en la etapa anterior con la clave 3.
4. .....

N. Opérese el documento obtenido en la etapa anterior con la clave n.

Cuando se desee desencriptar el archivo desarrollese el orden inverso:

1. Opérese el documento con la clave n.
2. Opérese el documento obtenido en la etapa anterior con la clave n-1.
3. Opérese el documento obtenido en la etapa anterior con la clave n-2.
4. .....

N. Opérese el documento obtenido en la etapa anterior con la clave 1.

Este modo de operar el encriptamiento multiplica por 2^{128} la dificultad para desencriptar por fuerza bruta el texto cifrado. Debe recordarse que cada clave debe ser distinta a todas las demás en por lo menos un byte, esto permitirá que con cada operación el texto cifrado se torne más y más confuso. No pueden usarse dos claves iguales en forma consecutiva, puesto que el desarrollo del algoritmo dos veces consecutivas usando la misma clave no producirá ningún cambio.

En la práctica comercial se puede utilizar un sólo algoritmo desarrollado tres veces consecutivas utilizando sólamente dos claves: el primer desarrollo utiliza la primera clave, el segundo desarrollo la segunda, y el tercero nuevamente la primera [42].

[42] Este procedimiento es utilizado en el algoritmo triple DES también conocido como TDES y 3DES, hasta el momento se considera prácticamente imposible de ser criptoanalizado [8].

17.- RECOMENDACIONES.

Tomando en cuenta todos los conceptos expuestos en la redacción de este proyecto es posible resumir los datos más relevantes del contenido en unas pocas afirmaciones que expresen rápidamente la forma óptima de practicar el encriptamiento de archivos digitales:

- Lo más importante al aplicar un sistema de encriptamiento es mantener el más absoluto sigilo de la información relacionada con el sistema. Pues de poco serviría implementar un algoritmo sumamente seguro si se descuida la seguridad en el manejo de la clave. Así mismo, todo sistema de seguridad es inefectivo si se tiene poco cuidado de resguardar los canales de acceso al sistema.
- Todo sistema de seguridad para la información constituye una barrera de protección para los datos, pero si verdaderamente se va a hacer uso de dicha información, se hace necesario el uso de una compuerta de acceso lo suficientemente fuerte para soportar cualquier tipo de ataque. En el caso del encriptamiento la llave de esta compuerta es siempre la clave, y por lo tanto, ésta debe de ser cuidadosamente manejada, transferida, cambiada y difundida. Se deben crear métodos de seguridad para el manejo de la clave en una organización confidencial o en una relación sigilosa de transferencia de datos, puesto que cualquier descuido puede dar acceso a la llave al delincuente.
- El éxito del encriptamiento es siempre inversamente proporcional al conocimiento global que tiene el delincuente digital del sistema. Entre más información tenga el delincuente digital menos seguro es el sistema. Es importante guardar celosamente todos los datos referentes a la seguridad, pues aunque uno sólo no proporcione el medio total para desencriptar, es una pista para la violación de todo sistema.
- Aunque parezca poco confiable, es siempre una garantía de seguridad el hecho que el fabricante de sistemas de seguridad proporcione toda la información acerca de sus productos. Pues si esta información no estuviera al alcance del comprador: ¿ Cómo puede saber el comprador que el sistema que está comprando no es algo tan simple como un alfabeto sustituto para el código ASCII ? , ¿Qué garantía se puede tener de un producto que no ofrece ninguna especificación de lo que hace, ni da ninguna información de cómo lo hace?

- La seguridad de los métodos de encriptamiento contemporáneos radica más que todo en su clave de encriptamiento, debido a que, si bien son conocidos ampliamente por el público general, ésta es sumamente difícil de ser encontrada entre todas las posibles combinaciones que puede tener. Esto fácilmente obliga a la deducción: entre más cantidad de combinaciones tenga una clave, mayor seguridad va a prestar el sistema de encriptamiento, entre más grande sea el tamaño de la clave en un sistema, mayor garantía se tendrá en la dificultad de ser encontrada.
- Aunque los algoritmos Standard para el encriptamiento ofrecen muy buenas medidas de seguridad, son siempre seguros los algoritmos exóticos de encriptamiento, ya que su uso incrementa en buena medida el grado de desinformación que el delincuente pueda tener del sistema. No obstante debe tenerse en cuenta que el algoritmo a utilizar debe ofrecer siempre un alto grado de confusión y difusión a la información encriptada.
- Los sistemas de autenticación digital, tales como las firmas digitales no son de ninguna utilidad si no se goza de los servicios de una entidad que autentique, esto significa, que para autenticar cualquier documento, es necesario que la autenticación sea reconocida por una entidad confiable que reconozca el origen del documento.
- No se puede autenticar ningún documento simplemente por el remitente. Toda autenticación es siempre la verificación de un secreto compartido, ya sea porque el secreto se comparte entre remitente y destinatario, o bien entre el remitente y un tercero certificador, y entre el mismo tercero y el destinatario.
- No sólo el encriptamiento de la información en sí puede proporcionar seguridad a los datos confidenciales, también el encriptamiento de la información protocolar necesaria para efectuar la transferencia de datos es garantía de seguridad, ya que si no se tiene la información necesaria para reconocer el paquete, este no puede ser tomado ni accedido por el delincuente digital.
- Si algo malo puede suceder (si existe la posibilidad), lo más seguro es que suceda algún día. Si todavía no ha sucedido, lo mejor será cambiar el sistema antes que la desgracia suceda. El encriptamiento nunca es una labor del todo terminada, debido a su misión de brindar seguridad para la información, es obligado a estar en continuo desarrollo para no dejarse alcanzar nunca por su contraparte: la inteligencia delincuencial. Cualquier aporte es siempre una gran contribución para este desarrollo.

BIBLIOGRAFIA

[1] Dvorak , John C., y Anis, Nick, "Dvorak´s Guide to DOS and PC performance", Osborne McGraw-Hill, Berkeley CA, 1991. pp 315- 327.

[2] Rodriguez G., Jorge E., "Introducción a las redes de área local", Mc. Graw-Hill-Interamericana editores S.A., México, 1996

[3] Groza, Calin, "BTB Cryptography", Página web en: http://ei.cs.vt.edu/~www.btb/hardcopy/book/chap12/crypto.html

[4] Chapman, Brent y Zwicky, Elizabeth D., "Construya firewalls para internet", Mc Grall Hill Interamericana editores, México, 1997, ISBN 970-10-1592-4

[5] Frazier, R.E., "Data Encription Tecniques", Pagina web en: http://www.catalog.com/sft/encrypt.html

[6] Smith, Richard E., "Internet Criptography", Adison-Wesley Pub. Co.,Reading, Massachusetts, 1997, ISBN 0-201-92480-3

[7] Ellison, Carl, "Criptography Timeline", Página web en: http://www.clarknet/pub/cme/html/timeline.html

[8] Schneier, Bruce ,"Applied Cryptography", John Wiley and sons Inc., New York , 1996, ISBN 0-471-12845-7

[9] "Criptographyc Algorithms", Página web en: http://www.cs.hut.fi/ssh/crypto/algorithms.html

[10] Autores varios de Crypt Cabal Society, "Cryptography FAQ", Página Web en: http//www.cis.ohio.edu/hypertext./faq/usenet/cryptography-faq/part01
http//www.cis.ohio.edu/hypertext./faq/usenet/cryptography-faq/part02
http//www.cis.ohio.edu/hypertext./faq/usenet/cryptography-faq/part03
http//www.cis.ohio.edu/hypertext./faq/usenet/cryptography-faq/part04
http//www.cis.ohio.edu/hypertext./faq/usenet/cryptography-faq/part05
http//www.cis.ohio.edu/hypertext./faq/usenet/cryptography-faq/part06
http//www.cis.ohio.edu/hypertext./faq/usenet/cryptography-faq/part07
http//www.cis.ohio.edu/hypertext./faq/usenet/cryptography-faq/part08
http//www.cis.ohio.edu/hypertext./faq/usenet/cryptography-faq/part09
http//www.cis.ohio.edu/hypertext./faq/usenet/cryptography-faq/part10

[11] "Introduction to Cryptography", Página web en: http://www.cs.hut.fi/ssh/crypto/intro.html

[12] "The neophite guide to hacking" , Pagina web en: http://www.wpe.com/~russel/neophite.txt

[13] "The ultimate begginers guide to hack", Pagina web en: http://www.wpe.com/~russel/newbie.txt

[14] Chapman , Brent y Zwicky , Elizabeth D. , " Construya Firewalls para internet" , Mc Graw Hill Interamericana editores, México, 1997
ISBN 970-10-1592-4