UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE QUIMICA

ESTUDIO TECNOLOGICO Y EVALUACION DE SISTEMAS DE SEGURIDAD EN REDES DE COMPUTO

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO QUIMICO

P R E S E N T A :

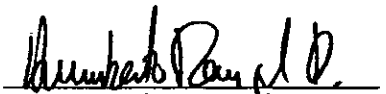
FRANCISCO JAVIER MONTES DE OCA JUAREZ

MEXICO, D.F. 2001

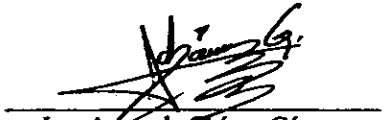
Jurado asignado:

Presidente	Prof. Jaime Noriega Bernechea
Vocal	Prof. Norma Gisela González Mariscal
Secretario	Prof. Humberto Rangel Dávalos
1er. Suplente	Prof. Marco Antonio Uresti Maldonado
2°. Suplente	Prof. Victor Manuel Vargas Chávez

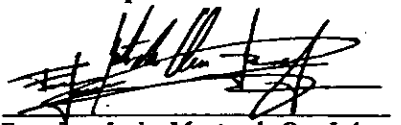
Sitio donde se desarrolló el tema: Instituto Mexicano del Petróleo.



I.Q. Humberto Rangel Dávalos
Asesor



Ing. Armando Chávez Gómez
Supervisor Técnico



Francisco Javier Montes de Oca Juárez
Sustentante

ESTE TRABAJO ESTÁ DEDICADO A TODAS LAS PERSONAS QUE HAN CONTRIBUIDO EN MI VIDA PARA LLEGAR A ESTE MOMENTO. GRACIAS DIOS POR PONERLOS AHÍ Y POR DEJARME LLEGAR.

GRACIAS:

A papá por todo, tú eres mi principal enseñanza.....

A mamá por tener fe en mi, no dejes de confiar.....

insisto, los amo, son lo mejor de esta vida.

A Güicho, no dejes de ser mi amigo antes que mi hermano.

A Lulú por acompañarme, tu amor es mi principal tesoro.

A tía Lupita, por preocuparse y estar ahí.

A Memo, Anibal, Braulio, Pepe y Héctor por su amistad y nobleza, hay que volver a jugar.

A Pilar y Mary por su sonrisa y amistad, a pesar de la distancia siguen pareciendo hermanas.

A los chacales, sin ustedes esta Facultad no sería tan especial.

A MI Universidad y a sus maestros, sin ellos no seriamos lo que somos ahora.

Y a todos que de un modo u otro han tenido o tienen que ver en este proyecto..... gracias. Y que sepais que del laberinto siempre se sale, pero primero hay que perderse.

CONTENIDO

La imaginación es más importante
que el conocimiento
-Albert Einstein

INTRODUCCIÓN

La comunicación de la información ha sido necesaria para el desarrollo de toda actividad humana desde la aparición del hombre. Sin embargo, dentro de este entorno, la comunicación de la información a veces se realiza en forma selectiva. Esto ocurre cuando la información tiene un carácter privado, restringido o secreto. Esta selectividad probablemente tiene sus orígenes en las formas más elementales de la sociedad, al aparecer intereses que defender o proteger.

La criptografía surge entonces como un medio de proteger la información y en un principio estuvo muy estrechamente vinculado a los círculos militares y diplomáticos, puesto que eran los únicos que en un principio tenían auténtica necesidad de ésta.

En la actualidad, la protección de la información ha cambiado de manera drástica, el vertiginoso desarrollo tecnológico de las comunicaciones electrónicas, unido al uso masivo y generalizado de las computadoras, hace posible la transmisión y almacenamiento de grandes volúmenes de información confidencial que es necesario proteger. Aunado a lo anterior, se ha dado un inusitado incremento en el uso de las redes públicas (Internet) y privadas. Es entonces, cuando las técnicas criptográficas pasan de ser una exigencia de minorías para convertirse en una necesidad real del hombre común, que ve en ésta amenaza de su información privada en una amenaza para su propia intimidad.

Debido a la necesidad de restringir el acceso indebido por parte de personas no autorizadas a información clasificada o personal, se han creado distintas formas de proteger esta información. La más importante, es mediante el uso de métodos criptográficos avanzados para la codificación de los datos. Muchos de estos métodos existen en el mercado y muchos más son creados a diario; es por esto, que la presente *tesis* pretende evaluar los diferentes métodos, de carácter público o comercial, que existen para proteger la información que se transmite en las redes públicas o privadas, las cuales no están protegidas y por ende, pueden ser fácilmente vulneradas.

El presente trabajo se divide en seis capítulos. En el primer capítulo, se presentan a manera de antecedentes, una breve semblanza histórica de la criptografía y su relevancia en la historia de la humanidad, así como un panorama general de la evaluación de tecnologías y sus principales características.

En el capítulo dos, se explican brevemente los principales algoritmos de codificación simétricos existentes.

En el capítulo tres, se explican brevemente los principales algoritmos de codificación asimétricos existentes.

En el capítulo cuatro, se explican brevemente las principales aplicaciones comerciales existentes.

En el capítulo cinco, se lleva a cabo la evaluación tecnológica mediante la evaluación de ciertos criterios, explicando la importancia de cada uno para la selección de un algoritmo seguro.

El capítulo seis, contiene el análisis de resultados y las conclusiones obtenidas a lo largo del desarrollo del presente trabajo.

Se incluye un pequeño glosario de términos utilizados durante el presente trabajo.

Finalmente, se anexa la bibliografía consultada.

De esta manera, los objetivos generales de este trabajo son: el de demostrar que las técnicas generales de evaluación tecnológica sean aplicables a cualquier campo y el de proporcionar una guía de selección rápida, objetiva y de fácil entendimiento, tanto para un usuario experto en el tema como para un principiante.

Las armas que fabriquemos,
otros las usarán.

-Shelly

CAPÍTULO I : ANTECEDENTES

I.1.- ANTECEDENTES HISTÓRICOS

Desde que el hombre desarrolló una forma de comunicación, se ha observado la importancia de transmitir información de diversa índole, ya sea de uno a otro individuo o de un grupo a otro. En algunas ocasiones, debido a la importancia de la misma, esta no debía llegar a oídos extraños, por lo cual era necesario transmitirla en secreto.

Conforme el hombre fue evolucionando, desarrolló la escritura y con ella la forma de transmitir la información en secreto, creando los métodos básicos de codificación de sustitución y transposición.

La sustitución[1] consiste en establecer una correspondencia entre las letras del alfabeto del mensaje original y otra secuencia de símbolos, sustituyendo uno a uno cada letra con el símbolo respectivo. Por ejemplo:

Clave: D
Interpretación: La primer letra será sustituida por la 5ª del abecedario, la 2ª por la 6ª y así sucesivamente.

Secuencia original:	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Símbolos correspondientes:	D E F G H I J K L N O P Q R S T U V X Y Z A B C
Mensaje original:	S I S T E M A D E S U S T I T U C I O N
Mensaje codificado:	V L V H P D G H V X V L W X F L R Q

Como se puede observar el número de claves son muy pocas, lo cual resulta en una relativa fácil decodificación para los criptoanalistas.

La transposición consiste en mezclar los símbolos del mensaje original para colocarlos en un orden distinto, de tal manera que resulte incomprensible su lectura.

Mensaje original:	S I S T E M A D E T R A N S P O S I C I O N
Codificación:	S E E N S O I M T S I N S A R P C T D A O I
Mensaje codificado:	S E E N S O I M T S I N S A R P C T D A O I

A este arte de esconder la información escrita se le dio el nombre de criptografía[2][3], cuyo significado proviene del griego criptos = oculto y logos = ciencia. El primer caso, históricamente reportado del uso de la criptografía se remonta al siglo V A.C. durante la guerra de Esparta y Atenas, cuando el General Lysandro recibe una tira de pergamino con símbolos sin aparente sentido, los cuales cobraban coherencia al ser envueltos en una vara de ciertas dimensiones.

En el siglo I A.C. dentro del imperio Romano, César escribía a sus cónsules y generales mediante un método que consistía en sustituir cada letra del mensaje por otra seleccionada de una forma fija, así por ejemplo, sustituía la primera letra del abecedario (A) por la quinta (E) y así sucesivamente.

En el siglo XIII, el inglés Roger Bacon relata observaciones hechas sobre un sistema de codificación denominado codificado bilitera, el cual consistía en sustituir cada pareja de letras de una palabra por una sola letra. Este método de codificación es considerado como el padre de los sistemas de codificación polialfabéticos, los cuales consisten en sustituir dos o más letras por una sola.

En el siglo XV, León Battista Alberti desarrolló un sistema de codificación de sustitución simple, cuya particularidad radica en que la codificación se realiza utilizando un sistema de dos anillos giratorios con letras grabadas tanto en el anillo interior como en el exterior. El codificado se efectúa enfrentando una casilla del anillo exterior con una del anillo interior. Esta pareja constituye la clave de codificación. El propio Alberti aconsejaba cambiar la clave cada cierto tiempo.

En 1480, G. Di Lavinde escribe acerca de un sistema de codificación consistente en sustituir palabras completas por un símbolo o letra en específico, o bien ciertas palabras por otras con un significado completamente distinto al original. Este sistema fue ampliamente utilizado durante los siglos XV y XVI en las cortes europeas, llegándose incluso a utilizar las notas musicales.

Uno de los más famosos criptoanalistas fue John Wallis, el cual a finales del siglo XVI y principios del XVII encontró la solución a varios mensajes codificados.

Durante el siglo XIX, se produce la consolidación de un método de codificación consistente en la alteración del orden de los símbolos de un mensaje, que previamente se ha subdividido en bloques de longitud fija.

El impulso decisivo a la criptografía, lo proporcionan las dos guerras mundiales, llegando al grado de que los Estados Unidos crearan su propio Departamento criptoanalítico denominado M18. Durante la primer guerra mundial, los alemanes desarrollaron un método

criptográfico denominado ADFGX, consistente en una matriz formada por estas cinco letras, en cuyo interior se encontraban 25 letras del abecedario, un mensaje en claro era dividido en bloques de 20 letras, las cuales eran sustituidas por parejas formadas por las letras ADFGX. Este método fue decodificado por los franceses.

Debido a los avances en el criptoanálisis, durante el siglo XX, se crearon máquinas mecánicas con complejos sistemas de codificación, dos de ellas fueron PURPLE y ENIGMA. Más tarde, con la invención de las computadoras, se observó la necesidad de construir métodos de codificación más complejos ya que estas podían decodificar casi cualquier método existente en cuestión de horas.

En la actualidad, debido al vertiginoso desarrollo tanto en la tecnología computacional como en las comunicaciones electrónicas, han dado como resultado la creación de diversos algoritmos criptográficos programables cuya principal finalidad es la de proteger las comunicaciones electrónicas. Los algoritmos actuales, a diferencia de sus antecesores, involucran complejas operaciones aritméticas, las cuales son repetidas en varias ocasiones, además de emplear claves de una extensa longitud (alrededor de 200 dígitos).

I.2.- GENERALIDADES SOBRE EVALUACIÓN TECNOLÓGICA.

Es por todos conocidos que para un problema determinado, muchas veces existen más de dos soluciones, y este caso no es la excepción.

En la búsqueda por encontrar formas más seguras de resguardar y transmitir la información, el hombre ha desarrollado diversos métodos y consecuentemente tecnologías criptográficas. Es ahora, cuando en un mundo con más recursos y necesidades informáticas donde estas tecnologías han encontrado una efectiva forma de comercialización debido a su gran demanda. Pero, ¿cuál es la tecnología que se debe elegir?, ¿qué método utilizar para nuestro caso en específico?. La respuesta a estas preguntas varían en base al tipo y

características del equipo de cómputo utilizado y a las necesidades de seguridad de cada persona.

Por tal motivo, antes de tomar una decisión, es conveniente obtener toda la información necesaria sobre las posibles alternativas de solución a nuestro problema, para después realizar una evaluación tecnológica y en base a ésta, tomar la decisión más adecuada.

Antes de continuar, es conveniente aclarar que la diferencia existente entre un análisis y una evaluación tecnológica radica en que mientras en el primero, solamente se estudia la tecnología utilizada, en la segunda, se emite una opinión y recomendaciones sobre su posible aplicación.

De ésta manera, el primer paso a realizar, es un análisis tecnológico de cada posible solución[4]. Siendo algunas de las principales especificaciones que se deben cumplir:

- **Análisis de tecnologías:** En este paso se lleva a cabo un análisis general del tipo de tecnología utilizada y sus principales características, tales como su actual vigencia, su distribución y su flexibilidad.

- **Análisis de equipo:** En este punto se involucra tanto la cantidad como el tipo de equipo empleado, así como su disponibilidad.

- **Análisis del proceso:** Es aquí donde se analizan los pasos involucrados en la tecnología analizada.

El primer paso, es el de clasificar criterios a evaluar para cada una de las características anteriores. Las alternativas que no satisfacen criterios esenciales son inmediatamente descartadas. El siguiente paso, es dar a cada uno de los criterios elegidos una jerarquía de prioridad, asignándose una puntuación, la cual tiene que ver con la manera en que estos se relacionan con la tecnología analizada, es decir, si le afecta directa o indirectamente, si un

criterio se encuentra intrínsecamente ligado a otro, etc. Finalmente, a cada tecnología se le asigna un valor o calificación de acuerdo a una escala establecida para este criterio.

Posteriormente, se suman todas las evaluaciones obtenidas de cada proceso, para con ello hacer una discriminación en base a la mayor puntuación, ya que esta no sólo determina cuáles alternativas son las mejores, sino el grado de superioridad.

Una vez que se han identificado las tecnologías más factibles, el paso siguiente es el de elegir la tecnología más acorde a nuestros requerimientos mediante un segundo análisis en el cual se deben considerar los siguientes puntos:

1. ***Identificación de los proveedores de tecnologías.*** En este paso se identifica sí la tecnología es o no de carácter público o privado, y en caso de presentarse la segunda opción se identifica al posible proveedor y la seriedad del mismo.

2. ***Análisis de diferencias básicas.*** Estas diferencias se pueden dar con respecto a los países de origen y el de aplicación. Algunas de éstas pueden ser:
   A) Calidad del producto.
   B) Servicios (disponibilidad).
   C) Marcos legales de aplicación.

3. ***Caracterización de la tecnología.*** Reiteradamente se presenta el error de que a toda tecnología se le debe dar el mismo tratamiento. Para caracterizar una tecnología es necesario tomar en cuenta los siguientes criterios:

A) En función al tipo de necesidades.
   - Tecnologías para pequeños y grandes requerimientos.
   - Tecnologías para demandas internas y/o externas.
   - Tecnologías compatibles.

B) En función al tipo de tecnología.
   - Tecnología de equipo. Referente a los equipos involucrados.

- Tecnología de producto. Referente a las especificaciones finales resultantes del proceso.
- Tecnología de proceso. Referentes a la forma de hacer las cosas.
- Tecnología de operación. Referentes a que tan amigable y flexible es a la tecnología.
- Tecnología de precio. Referentes al costo de licenciamiento.

I.3.- AMENAZAS EN LAS REDES DE COMPUTO

Dentro de las redes de computo, se han identificado dos grandes tipos de amenazas a la información, una es representada por los usuarios no autorizados a la misma, y la otra, por programas hostiles o no deseados que pueden comprometer la integridad de grandes volúmenes de información, los primeros son conocidos comúnmente con el nombre de crackers o hackers y los segundos como virus.

I.3.1.- INTRUSOS

Los intrusos también conocidos como hackers o crackers pueden comprometer la seguridad, integridad, autenticidad y confidencialidad de la información transmitida a través de una red. Estos, pueden ser clasificados en tres clases:

1. ***Enmascarados:*** Usuarios no autorizados que penetran dentro del sistema utilizando la identidad de un usuario autorizado.

2. ***Abusivos:*** Usuarios con acceso legítimo al sistema los cuales utilizan datos, programas o recursos para los cuales no tienen autorización o bien abusan de sus privilegios dentro de la red.

3. ***Usuario clandestino:*** Usuario que utiliza el control de supervisión del sistema y lo utiliza en su favor para introducirse dentro del mismo.

Generalmente, la procedencia del enmascarado es desde fuera del sistema, la del abusivo, es desde dentro del sistema y la del usuario clandestino se puede dar desde ambos sitios.

El nivel de ataque de uno de estos usuarios puede ser desde inofensivo como la exploración de la información contenida dentro de la red, hasta dañino si llega a comprometer la integridad de la información o del sistema.

I.3.2.- PROGRAMAS NO DESEABLES

Se denomina así a todo tipo de programa que lleva a cabo rutinas o funciones no ordenadas por el usuario, las cuales pueden llegar a comprometer a seguridad o integridad de la información que este almacene. Estos programas han sido clasificados de acuerdo a sus características tal y como se muestra en la figura 1.1.

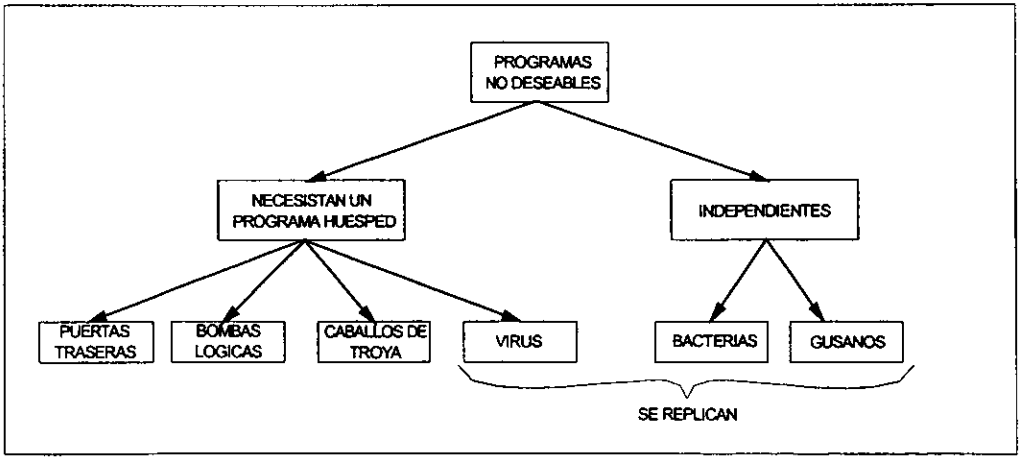


FIG. 1.1- CLASIFICACIÓN DE LOS PROGRAMAS NO DESEADOS.

Cada uno de las amenazas programables mostradas en la figura anterior, se definen a continuación:

A) **Puertas traseras:** Es un punto de entrada no registrado a un programa, el cual es utilizado para otorgar acceso al mismo sin la necesidad de los métodos normales de autentificación de acceso.

B) **Bombas lógicas:** Es un programa que ejecuta una serie de órdenes no autorizadas por el usuario cuando se cumplen una serie de condiciones de operación.

C) **Caballos de Troya:** Rutina secreta no registrada presente dentro de un programa útil. La ejecución del programa resulta en la ejecución de la rutina.

D) **Virus:** Código incluido dentro de un programa el cuál tiene como resultado la copia de sí mismo para ser insertado dentro de uno o más programas. Adicionalmente, a ésta propagación, este programa nocivo lleva a cabo la ejecución de ciertas rutinas no deseadas.

E) **Bacteria:** Programa que consume los recursos del sistema mediante la replicación de sí mismo.

F) **Gusanos:** Programa que puede replicarse a sí mismo y enviar copias de computadora a computadora a través de la red de trabajo. Una vez dentro, el gusano puede ser activado, replicarse y propagarse nuevamente. Adicionalmente a la propagación, el gusano lleva a cabo la ejecución de rutinas no deseadas.

La solución ideal para el caso de cada una de éstas amenazas es la prevención, esto es, no permitir que estos entren dentro del sistema mediante la utilización de controles de acceso y autentificación, o bien detectarlos en el primer momento para proceder después a su eliminación.

De todas las clases de los programas no deseables, los programas capaces de replicarse son los más temidos, debido a su alta capacidad de extender una infección dentro de una red de trabajo. La solución típica para eliminar este tipo de formas de ataque la proporcionan los programas denominados como "antivirus".

Estos programas buscan, eliminan y eventualmente protegen a un usuario de los programas no deseados, utilizando diferentes técnicas para encontrar a los mismos. Estas técnicas van desde el rastreo de la huella característica del virus hasta la utilización de una

serie de reglas heurísticas encaminadas a mostrar programas o archivos que pudiesen haber sido infectados.

A pesar de que el ataque mediante este tipo de programas es una forma de comprometer la seguridad dentro de las redes de cómputo, en el presente trabajo no se analizará ni se discutirá alguna forma de protección para este tipo de amenazas programables, debido a que la estructura de los antivirus no es comparable con la de los algoritmos criptográficos ni con las aplicaciones que los utilizan como herramienta, los cuales se pretende evaluar tecnológicamente a lo largo de este trabajo.

I.4.- GENERALIDADES SOBRE LOS SISTEMAS DE SEGURIDAD EN REDES DE COMPUTO

La extensión de la microinformática y de las redes de ámbito mundial que interconectan recursos informáticos de todo tipo, ha hecho que los peligros que sufre la información almacenada en los diversos sistemas crezcan considerablemente y se diversifiquen, y que las medidas adoptadas internamente en los Centros de Cómputo resulten insuficientes.

En los últimos meses, no sólo la prensa especializada en informática, sino todos los medios de difusión han hecho eco del futuro de las autopistas de la información, cuyo embrión está representado por la red Internet. A raíz de la interconexión del mundo empresarial a esta red, viaja por ella y se almacena información de todo tipo, que abarca desde noticias, documentos, normas y aplicaciones informáticas de libre distribución hasta complejas transacciones que requieren medidas de seguridad que garanticen la confidencialidad, la integridad y el origen de los datos.

La escucha electrónica, que permite la obtención y posible manipulación de información privada, y los sabotajes realizados tanto por atacantes externos como internos, están causando últimamente la pérdida de grandes cantidades de dinero.

Pero, ¿Cómo controlar el acceso indebido a aplicaciones y a la información almacenada?, ¿Cómo garantizar la integridad o la confidencialidad de la información que viaja a través de las redes?, ¿Cómo comprobar de una forma segura que el emisor y el receptor de una información son realmente quienes dicen ser? o ¿Cómo garantizar que el emisor no niegue haber enviado algo y el receptor no niegue haberlo recibido?.

El objetivo de la Criptografía es el de proporcionar comunicaciones seguras sobre canales inseguros, es decir, permitir que dos entidades, bien sean personas o bien aplicaciones, puedan enviarse mensajes por un canal que puede ser interceptado por una tercera entidad, de modo que sólo los destinatarios autorizados puedan leer los mensajes. Pero la Criptografía no es en sí la seguridad, sólo es la herramienta básica que utilizan mecanismos más complejos para proporcionar, además de confidencialidad, otros servicios de seguridad.

La Criptografía viene utilizándose desde la antigüedad para enviar mensajes bélicos y amorosos de forma confidencial.

I.4.1.- LOS SERVICIOS DE SEGURIDAD

El documento OSI referente a la seguridad en la información, presenta una Arquitectura de Seguridad. Según esta arquitectura, para proteger las comunicaciones de los usuarios en las redes, es necesario dotar a las mismas de los siguientes servicios de seguridad:

A) ***Autentificación de entidad par*** . Este servicio corrobora la fuente de una unidad de datos. La autentificación puede ser sólo de la entidad origen o de la entidad destino, o ambas entidades se pueden autentificar la una o la otra.

B) ***Control de acceso***. Este servicio se utiliza para evitar el uso no autorizado de recursos.

C) ***Confidencialidad de datos***. Este servicio proporciona protección contra la revelación deliberada o accidental de los datos en una comunicación.

D) ***Integridad de datos***. Este servicio garantiza que los datos recibidos por el receptor de una comunicación coinciden con los enviados por el emisor.

E) ***No repudio.*** Este servicio proporciona la prueba ante una tercera parte de que cada una de las entidades comunicantes han participado en una comunicación. Puede ser de dos tipos:

- *Con prueba de origen*. Cuando el destinatario tiene prueba del origen de los datos.
- *Con prueba de entrega*. Cuando el origen tiene prueba de la entrega íntegra de los datos al destinatario deseado.

Para proporcionar estos servicios de seguridad es necesario incorporar en los niveles apropiados del Modelo de Referencia OSI los siguientes mecanismos de seguridad:

CODIFICACIÓN

La codificación puede hacerse utilizando sistemas criptográficos simétricos o asimétricos y se puede aplicar extremo a extremo o individualmente a cada enlace del sistema de comunicaciones.

El mecanismo de codificación soporta el servicio de confidencialidad de datos al tiempo que actúa como complemento de otros mecanismos de seguridad.

FIRMA DIGITAL

Se puede definir la firma digital como el conjunto de datos que se añaden a una unidad de datos para protegerlos contra la falsificación, permitiendo al receptor probar la fuente y la integridad de los mismos. La firma digital supone la codificación, con una componente secreta del firmante, de la unidad de datos y la elaboración de un valor de control criptográfico.

La firma digital utiliza un esquema criptográfico asimétrico. La firma consiste en una cadena de datos que contiene el resultado de codificar con dicho algoritmo asimétrico aplicando la clave privada del firmante, una versión comprimida, mediante una función hash unidireccional y libre de colisiones, del texto a firmar.

Para verificar la firma, el receptor decodifica la firma con la clave pública del emisor, comprime con la función hash al texto original recibido y compara el resultado de la parte decodificada con la parte comprimida, sí ambas coinciden el emisor tiene garantía de que el texto no ha sido modificado. Como el emisor utiliza su clave secreta para codificar la parte comprimida del mensaje, puede probarse ante una tercera parte, que la firma sólo ha podido ser generada por el usuario que guarda la componente secreta.

El mecanismo de firma digital soporta los servicios de integridad de datos, autentificación de origen y no repudio con prueba de origen. Para proporcionar el servicio de no repudio con prueba de entrega es necesario forzar al receptor a enviar al emisor un recibo firmado digitalmente.

CONTROL DE ACCESO

Este mecanismo se utiliza para autentificar las capacidades de una entidad, con el fin de asegurar los derechos de acceso a los recursos que posee. El control de acceso se puede realizar en el origen o en un punto intermedio, y se encarga de asegurar sí el emisor está autorizado a comunicarse con el receptor y/o a usar los recursos de comunicación requeridos.

Sí una entidad intenta acceder a un recurso no autorizado, o intenta el acceso de forma impropia a un recurso autorizado, entonces la función de control de acceso rechazará el intento, al tiempo que puede informar del incidente, con el propósito de generar una alarma y/o registrarlo.

INTEGRIDAD DE DATOS.

Es necesario diferenciar entre la integridad de una unidad de datos y la integridad de una secuencia de unidades de datos ya que se utilizan distintos modelos de mecanismos de seguridad para proporcionar ambos servicios de integridad.

Para proporcionar la integridad de una unidad de datos la entidad emisora añade a la unidad de datos una cantidad que se calcula en función de los datos. Esta cantidad, probablemente codificada con técnicas simétricas o asimétricas, puede ser una información suplementaria compuesta por un código de control de bloque, o un valor de control criptográfico. La entidad receptora genera la misma cantidad a partir del texto original y la compara con la recibida para determinar sí los datos no se han modificado durante la transmisión.

Para proporcionar integridad a una secuencia de unidades de datos se requiere, adicionalmente, alguna forma de ordenación explícita, tal como la numeración de secuencia, un sello de tiempo o un encadenamiento criptográfico.

Intercambio de autentificación.

Existen dos grados en el mecanismo de autentificación:

- **Autentificación simple.** El emisor envía su nombre distintivo y una contraseña al receptor, el cual los comprueba.

- **Autentificación fuerte**. Utiliza las propiedades de los criptosistemas de clave pública. Cada usuario se identifica por un nombre distintivo y por su clave secreta. Cuando un segundo usuario desea comprobar la autenticidad de su interlocutor deberá comprobar que éste está en posesión de su clave secreta, para lo cual deberá obtener su clave pública.

Para que un usuario confíe en el procedimiento de autentificación, la clave pública de su interlocutor se tiene que obtener de una fuente de confianza, a la que se denomina Autoridad de Certificación. La Autoridad de Certificación utiliza un algoritmo de clave pública para certificar la clave pública de un usuario produciendo así un certificado.

Un certificado es un documento firmado por una Autoridad de Certificación, válido durante el período de tiempo indicado, que asocia una clave pública a un usuario.

El mecanismo de intercambio de autentificación se utiliza para soportar el servicio de autentificación de entidad par.

I.4.2.- SEGURIDAD EN REDES COMPLEJAS: EL CASO DE INTERNET

El fenómeno de la extensión de la Internet ha adquirido una velocidad tan rápida y unas proporciones, que el panorama actual y muchos de los efectos que se dan en su seno resultan sorprendentes y dificilmente imaginables hace sólo una década.

Inicialmente Internet nace como una serie de redes que promueven el intercambio de información entre investigadores que colaboran en proyectos conjuntos o comparten resultados usando los recursos de la red. En esta etapa inicial, la información circulaba libremente y no existia una preocupación por la privacidad de los datos ni por ninguna otra problemática de seguridad. Estaba totalmente desaconsejado usarla para el envío de documentos sensibles o clasificados que pudieran manejar los usuarios.

Los protocolos de Internet fueron diseñados de una forma deliberada para que fueran simples y sencillos. El poco esfuerzo necesario para su desarrollo y verificación jugó eficazmente a favor de su implantación generalizada, pero tanto las aplicaciones como los niveles de transporte carecían de mecanismos de seguridad que no tardaron en ser vulnerados.

Más recientemente, la conexión a Internet del mundo empresarial se ha producido a un ritmo vertiginoso muy superior a la difusión de ninguna otra tecnología anteriormente ideada. Ello ha significado que esta red de redes se haya convertido en "la red" por excelencia. Esto es, el medio más popular de interconexión de recursos informáticos y embrión de las anunciadas autopistas de la información.

Se ha incrementado la variedad y cantidad de usuarios que usan la red para fines tan diversos como el aprendizaje, la docencia, la investigación, la búsqueda de socios o mercados, la cooperación altruista, la práctica política o, simplemente, el juego. En medio de esta variedad han ido aumentando las acciones poco respetuosas con la privacidad y con la propiedad de recursos y sistemas. Hackers, crakers y demás familias han hecho aparición en el vocabulario ordinario de los usuarios y de los administradores de las redes.

La propia complejidad de la red implica una dificultad para la detección y corrección de los múltiples y variados problemas de seguridad que van apareciendo. Además de las técnicas y herramientas criptográficas antes citadas, es importante recalcar que una componente muy importante para la protección de los sistemas consiste en la atención y vigilancia continua y sistemática por parte de los gestores de la red.

A continuación en el siguiente capítulo se revisarán algunos de los más importantes algoritmos de codificación simétrica, así como una breve descripción de su funcionamiento.

La experiencia es madre de la ciencia,
pero de ambas es la paciencia.
- Anónimo

CAPÍTULO II :ALGORITMOS DE CODIFICACIÓN SIMÉTRICA.

II.1.- INTRODUCCIÓN

Estos algoritmos reciben la denominación de codificadores simétricos[5][6] debido a que tanto emisor como receptor deben guardar la misma secuencia de dígitos binarios, que se han etiquetado con el nombre de clave. Estos algoritmos son también llamados de clave secreta, debido a la necesidad de mantener en secreto la clave, ya que de lo contrario un tercero podría decodificar archivos en el caso de tener acceso a ésta.

Este tipo de algoritmos son los más utilizados hasta nuestros días y consisten en la codificación de archivos por parte de un emisor, mediante el uso de una clave, la cual será utilizada nuevamente por un receptor para poder decodificar el mensaje. Este sistema tiene el inconveniente de requerir un canal seguro para la transmisión de la clave, además de que es necesario contar con una clave por cada usuario con el cual se desee tener intercambio de textos codificados. Debido a lo anterior, sólo son utilizados para comunicaciones entre círculos cerrados de usuarios.

En el entorno computacional muchos de estos algoritmos no codifican archivos en un solo paso, por el contrario lo hacen en partes o bloques de texto de una longitud determinada, la cual dependerá del programa, por lo que también estos algoritmos entran en la clasificación

de codificadores en bloque. Todos los codificadores en bloque se componen de cuatro elementos principales:

1. Transformación inicial.
2. Una función criptográficamente débil iterada ***n*** veces.
3. Una transformación final.
4. Algoritmo de expansión de clave.

En la actualidad, la estructura básica de un codificador de bloque típico es un codificador de producto iterado, con transposición y simple sustitución como sus principales componentes. El intervalo típico de longitud de bloque es de 32 a 128 bits, normalmente en múltiplos de 8 bits. La clave es usualmente igual a la longitud del bloque, cuando es más pequeño, ésta es expandida artificialmente para acomodar la longitud completa del bloque.

Los sistemas de codificación en bloque operan sobre conjuntos de información muy reducidos, son adecuados para codificar pequeños mensajes como claves, identificaciones, firmas, contraseñas, etc. Pero en algunos casos, son totalmente inadecuados para la codificación de grandes cantidades de datos, tales como, textos formateados, listados, programas, tablas y formularios, esto debido a que la estructura del documento tiende a detectarse fácilmente.

II.2.- LUCIFER

LUCIFER[7] es un algoritmo de codificación del tipo Feistel, esto es, divide el bloque de texto a codificar en dos mitades y codifica cada una de ellas de forma alternada. Fue creado a mediados de los años 70's por la Corporación IBM con el fin de asegurar la confidencialidad de buena parte de su información técnica y más tarde se convirtió en la base para crear al DES.

Este algoritmo utiliza una clave de entrada de 128 bits para codificar bloques de texto de 128 bits. La parte central de este algoritmo consta de una función *f*, la cual se aplica una vez por iteración a la mitad derecha del bloque.

GENERACIÓN DE SUBCLAVES

Cada ciclo de codificación utiliza una subclave de 72 bits. La primera subclave utilizada dentro del primer ciclo iterativo, se forma a partir del primer byte de la clave principal repetido dos veces, seguida por los siguientes siete bytes de la misma clave. Para generar las siguientes subclaves de los siguientes ciclos, la clave principal se rota siete bytes a la izquierda.

LA FUNCIÓN *f*

Una vez dividido el texto principal en dos mitades, es aplicada la función OR exclusivo a la mitad derecha del bloque utilizando los últimos ocho bytes de la subclave correspondiente al ciclo de codificación. Una vez realizado lo anterior, se realiza una permutación de bits utilizando la subclave en turno y dos cajas S mostradas en la tabla 2.1, la primera caja llamada S-0 es para los bloques más significativos y la S-1 para los menos significativos.

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

TABLA 2.1.- ESTRUCTURA DE LAS CAJAS S.

Los bits resultantes de esta operación son intercambiados nuevamente utilizando la tabla 2.2, enumerando de 0 a 63, del bit más significativo al menos significativo.

10	21	52	56	27	1	47	38	18	29	60	0	35	9	55	46
26	37	4	8	43	17	63	54	34	45	12	16	51	25	7	62
42	53	20	24	59	33	15	6	50	61	28	32	3	41	23	14
58	5	36	40	11	49	31	22	2	13	44	48	19	57	39	30

TABLA 2.2.- TABLA DE PERMUTACIÓN.

Una descripción más gráfica sobre este algoritmo se muestra en la figura 2.1.

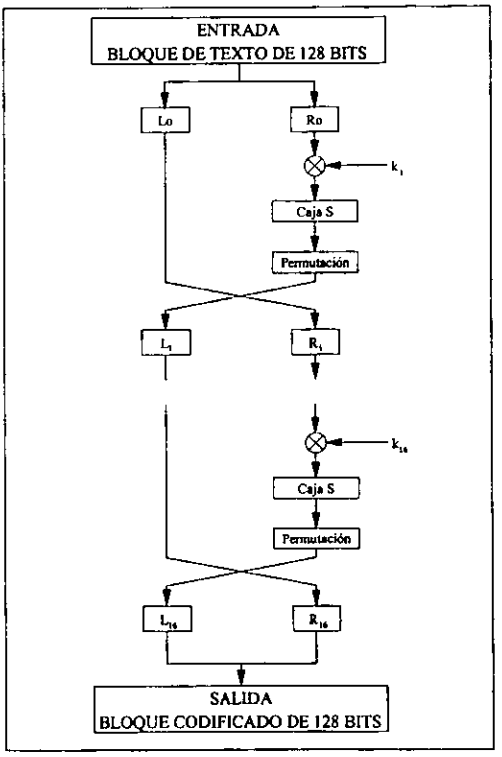


FIG. 2.1- DIAGRAMA DE BLOQUE DE LUCIFER.

A pesar de que LUCIFER cuenta con un tamaño de clave y un bloque de texto mayor al DES, es considerablemente más vulnerable a ataques por criptoanálisis diferencial, esto debido a la generación tan regular de sus subclaves. Por otra parte, es posible la utilización de este algoritmo de codificación sí lo ejecutamos junto con otro codificador, tal como el propio DES.

II.3.- DES (Data Encryption Standard)

En 1973 el NBS (National Bureau of Standards) de Estados Unidos organizó un concurso solicitando un algoritmo de codificación para protección de datos de computadoras para su transmisión y almacenaje. En 1974, IBM presentó una propuesta basada en un sistema de codificación propio denominado LUCIFER. En enero de 1977, el NBS publicó una variante del codificador LUCIFER, a este "nuevo" algoritmo se le dio la denominación de DES (Data Encryption Standard)[8].

El DES es un algoritmo de tipo Feistel que codifica bloques de texto de 64 bits con una clave original de 64 bits, de los cuales, el bit menos significativo de cada byte es utilizado como bit de paridad. En consecuencia, los ocho bits de paridad pueden ser eliminados al no aportar ninguna información a la clave, quedando ésta reducida a tan sólo 56 bits.

El bloque de texto original de 64 bits se enumeran del más al menos significativo de 1 a 64, se mezclan mediante una permutación inicial P1 definida por la siguiente tabla.

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

TABLA 2.3- PERMUTACIÓN INICIAL P1.

Una vez realizado lo anterior, los 64 bits permutados se dividen en dos sub-bloques (L_0, R_0), de 32 bits cada uno. En estas condiciones, la codificación del DES viene definida por las ecuaciones:

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

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

para i=1 a 16 y donde $\oplus$ es el OR exclusivo de los desarrollos binarios de los valores que toma como parámetros y $f(..)$ es una función de los 32 bits de R_{i-1} y de la subclave K_i. Con todo ello, al final de la iteración 16 se aplica la permutación inversa (tabla 2.4) de la inicial P1 a los 64 bits concatenados (L_{16},R_{16}), siendo la salida de ésta permutación la codificación del mensaje inicial.

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

TABLA 2.4- PERMUTACIÓN INVERSA $P1^{-1}$.

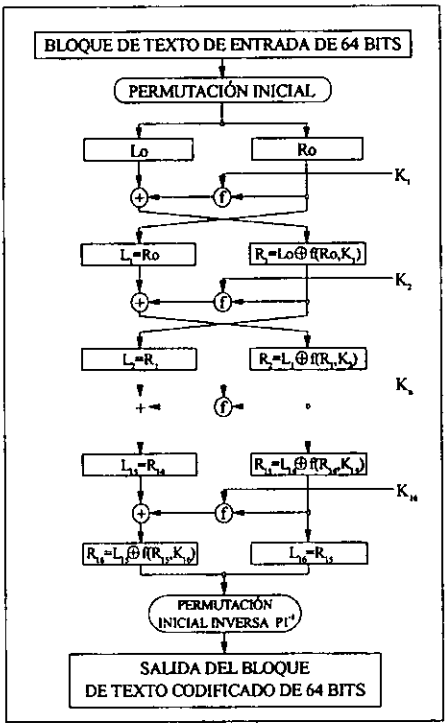


FIG. 2.2- DIAGRAMA DE BLOQUES DE CODIFICACIÓN DEL DES.

GENERACIÓN DE SUBCLAVES

La generación de las subclaves de codificado K_i para i=1..16 se lleva a cabo partiendo de la clave K de 64 bits numerados de 1 a 64 del más al menos significativo, a los cuales se les aplica una permutación PC1 que a su vez elimina los bits de paridad de cada byte de acuerdo con la tabla 2.5.

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

TABLA 2.5- TABLA DE PERMUTACIÓN Y SELECCIÓN PC1.

Los 56 bits de salida de PC1 se dividen en dos sub-bloques (C_0, D_0) de 28 bits cada uno. Con todo ello, la generación de las subclaves K_i para i=0..16 se lleva a cabo de acuerdo con:

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

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

donde LS es un desplazamiento circular a la izquierda de 1 ó 2 bits del entero binario que toma como argumento de acuerdo la tabla 2.6.

Vuelta de codificación N°	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
N° de bits desplazados a la izq.	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	1

TABLA 2.6- PATRÓN DE DESPLAZAMIENTO CIRCULAR A LA IZQUIERDA DE LS.

PC2 es una función de permutación y selección de los 56 bits (numerados del más al menos significativo de 1 a 56) resultantes de concatenar los valores (C_i, D_i) de 28 bits cada

uno. La salida PC2, es decir, la subclave K_i está constituida por los 48 bits obtenidos como se indica en la tabla 2.7.

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

TABLA 2.7- TABLA DE PERMUTACIÓN Y SELECCIÓN PC2.

Las operaciones LS y PC2 se repiten hasta obtener un total de 16 subclaves, así como se muestra en la figura 2.3.

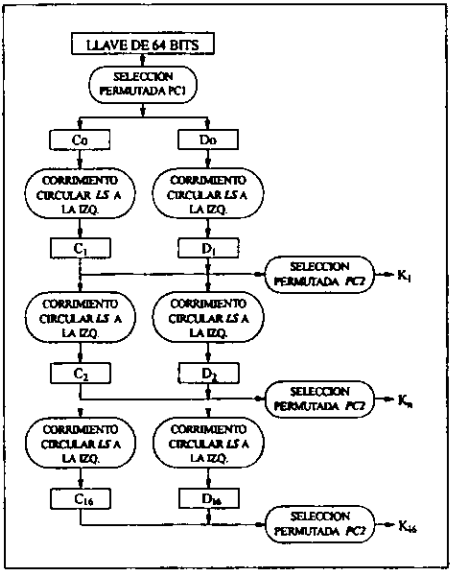


FIG. 2.3 - GENERACIÓN DE SUBCLAVES.

Para la decodificación se utiliza el mismo algoritmo pero con las subclaves en orden inverso, es decir, K_{16} en la primer iteración, K_{15} en la segunda y así sucesivamente. Debido a lo anterior, se tiene que para la codificación, las subclaves pueden calcularse simultáneamente

con la codificación parcial, mientras que para realizar la decodificación es necesario calcular de forma previa todas las subclaves.

LA FUNCIÓN $f(R_{i-1},K_i)$

La función f tiene como entrada los 32 bits de $\mathbf{R}_{i-1}$ y los 48 bits de la subclave $\mathbf{K}_i$. Inicialmente, los 32 bits de $\mathbf{R}_{i-1}$ atraviesan una función de expansión **E** (tabla 2.8) que los transforma en 48 bits, los cuales se suman con los 48 bits de la subclave $\mathbf{K}_i$ mediante una función OR exclusiva.

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

TABLA 2.8- FUNCIÓN DE EXPANSIÓN *E*.

Los 48 bits resultantes se subdividen en 8 bloques de 6 bits cada uno, los que servirán de entradas a ocho funciones no lineales denominadas cajas **S**. Cada una de estas cajas es una tabla de doble entrada con cuatro filas y dieciséis columnas.

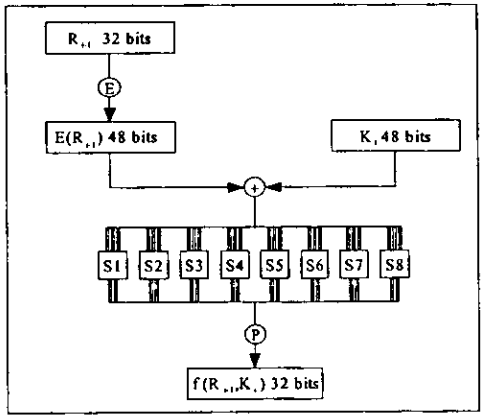
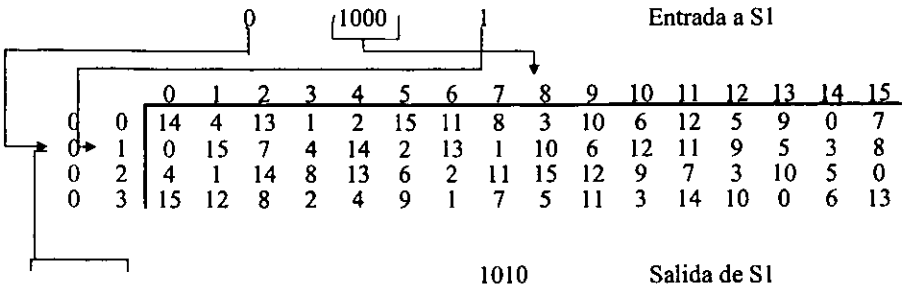


FIG. 2.4- DIAGRAMA DE BLOQUES DE LA FUNCIÓN *f*.

Los seis bits de entrada para cada caja S son utilizados para generar una dirección de la fila y una dirección de columna como se muestra en la fig. 2.5.



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

FIG. 2.5- EJEMPLO DEL PROCESAMIENTO DE BITS DENTRO DE UNA CAJA *S*.

Por su parte la permutación viene dada por la tabla 2.9.

16	7	20	21
29	12	26	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 2.9- PERMUTACIÓN *P*.

La robustez del DES está basada en la estructura de las cajas S, inclusive se ha demostrado que cualquier cambio en éstas debilita el método. Algunos de los principios conocidos de su diseño son los siguientes:

1. Las filas de cada caja son una permutación de los enteros 1 al 15.
2. Ninguna de las salidas de las cajas es una función lineal o afín de su entrada.
3. El cambio de un bit en la entrada a una caja, provoca al menos el cambio de dos bits de la salida de la misma.
4. Las cajas S se diseñaron de tal manera que cuando un bit de entrada se mantiene constante, minimizan la diferencia entre el número de ceros y unos a la salida.

5. Si se fija uno de los bits de la entrada a cualquiera de la caja y se observa un bit en posición fija en la salida, entonces en número de entradas para las que el bit de salida es 0 es aproximadamente igual al número de entradas para las que el bit de salida es 1.

IMPLEMENTACIÓN DEL DES

El algoritmo DES está regulado por las normas ISO 8372, ISO 8798 e ISO 10118. Las cuales se exigen que su implementación se lleve a cabo mediante un circuito integrado electrónico. Existe un chip DES, el cual es un producto estratégico norteamericano, por lo que no se permite su exportación sin un permiso especial del gobierno, además de que este está prohibida la comercialización de cualquier chip fabricado en el extranjero. Algunos fabricantes de este chip son:

- Cryptech. CRY 12C102: 22.5 Mbits/s con interfaz de 32 bits.
- Computer Elektronik Infosys. <<Super Crypt>> 100 Mb/s.
- Infosys DES Chip (Alemán). Las cajas S son cargadas en software, permitiéndose así versiones no estándar.
- Newbridge Microsystems. AM 9568 compatible con el chip DES 25 Mhz.
- Pijnenburg. PCC 100:20 Mbits/s.
- Semaphor. Com.. Roadrunner284, 1994 a 40 Mhz y 35.5 Mb/s.
- VISI Tech. Chip 6868, 1995 a 32 Mhz, 64 Mb/s.

En 1981, el ANSI adoptó el DES con el nombre de Data Encryption Algorithm (DEA). En esta versión, la norma no exige implementación con circuito integrado, por lo que puede ser utilizado en una versión en software sin que se afecte con esto, la velocidad de codificación.

MODOS DE UTILIZACIÓN DEL ALGORITMO DES.

Existen cuatro modos básicos de codificación DES[9], los cuales son descritos en las normas, además del codificado múltiple.

Electronic Code Book (ECB). Es el modo de codificación básica del DES. Consiste en dividir el mensaje a codificar en bloques de 64 bits (con un relleno adicional sí es necesario en el último bloque) llevándose a cabo la codificación bloque a bloque.

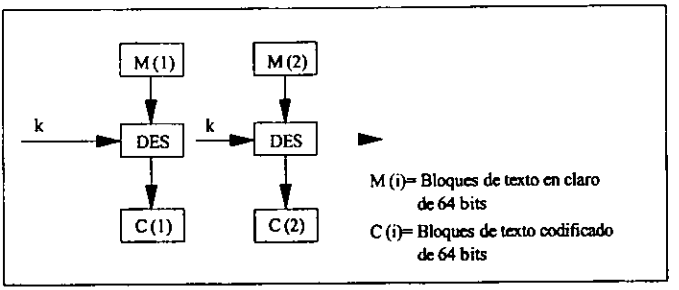


FIG. 2.6- DIAGRAMA DE BLOQUES DEL MODO ECB DEL DES.

El inconveniente de este procedimiento, es que los mensajes codificados con la misma estructura permiten a un eventual intruso obtener información sobre la variación de los datos y con esto eventualmente poder encontrar la clave utilizada.

Cipher Block Chaining (CBC). En este caso, la codificación se inicia con un vector de inicialización (Vd'I) de 64 bits que se suma mediante un OR exclusivo con el primer bloque del mensaje en claro. El resultado de esta suma, se codifica con el DES constituyéndose de ésta manera el primer bloque del mensaje codificado y así sucesivamente.

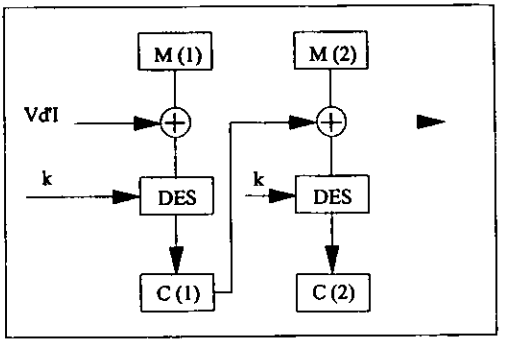


FIG. 2.7- DIAGRAMA DE BLOQUES DEL MODO CBC DEL DES.

Una de las principales características de este procedimiento, es que los mensajes codificados de un mismo mensaje en claro pueden ser distintos sin más que elegir en cada caso un vector de inicialización diferente. Sin embargo, este método tiene el inconveniente de requerir la transmisión de dicho vector de inicialización para poder llevar a cabo la decodificación.

Cipher Feed Back (CFB). Este modo de codificado es equivalente al anterior, con la diferencia de que en este caso, la codificación inicia con un vector de inicialización (Vd'I) de 64 bits cuya codificación DES se suma mediante un OR exclusivo con el primer bloque de mensaje codificado. Este se codifica a su vez con DES y el resultado se suma mediante otro OR exclusivo con el siguiente bloque de mensaje en claro, y así sucesivamente.

Al utilizar un vector de inicialización, ésta forma de codificación observa las mismas ventajas e inconvenientes del método anterior.

Output Feed Back (OFB). En este modo de codificación, se calcula una colección de valores pseudoaleatorios de 64 bits (tantos como bloques de mensajes a codificar), obtenidos mediante codificación consecutiva con DES de un vector de inicialización (Vd'I) de 64 bits elegido aleatoriamente. En estas condiciones, cada uno de estos valores pseudoaleatorios son sumados mediante un OR exclusivo con cada uno de los mensajes en claro. Los resultados de éstas sumas constituyen los respectivos bloques del mensaje codificado.

Uno de los inconvenientes de este método es que sí un eventual atacante conoce un bloque de texto en claro ($M(n)$) y su correspondiente codificación ($C(n)$), entonces sería capaz de conseguir que el receptor recuperara un mensaje falso.

Codificación múltiple. Es común suponer que un sistema de codificación sería más fuerte sí se codificara varias veces el mismo mensaje utilizando diferentes claves para cada caso. De ésta manera se desarrolló la codificación triple del DES.

En la codificación del triple DES se ha adoptado la siguiente nomenclatura: E para la codificación, D para la decodificación y un número que representa el número de claves utilizadas en el proceso. Así tenemos que los métodos más utilizados en la industria son el EDE-2 CBC y el CBC EDE-2.

SEGURIDAD DEL DES

Desde su creación en 1977, el DES es revisado cada cinco años por el NIST para asegurar su seguridad[10][11]. Los avances recientes en técnicas criptográficas y el avance tecnológico en la capacidad de las computadoras han hecho que en la actualidad deje de tener validez en su forma más simple, e incluso el NIST está buscando crear un método de seguridad más avanzado, denominado AES (Advanced Encryption Standard), a pesar de esto y mientras se da a conocer el nuevo algoritmo de codificación, el DES se seguirá utilizando en la modalidad del triple DES por algún tiempo más.

Ataques por fuerza bruta. Este tipo de ataque es también conocido con el nombre de búsqueda exhaustiva y se realiza probando cada una de las posibles claves existentes hasta dar con la correcta.

La decodificación de un mensaje utilizando ésta técnica requiere el análisis de 2^{56} claves posibles, siempre y cuando el mensaje sea codificado en el modo ECB, ya que es el más sencillo, lo cual tomaría más de 1000 años utilizando la computadora más moderna construida hasta la fecha.

Hasta hace unos años, algunos investigadores habían hecho diseños en el papel de máquinas capaces de decodificar mensajes codificados con el DES utilizando este método, pero de las cuales no se conocía su posible existencia. Como muestra de estas máquinas tenemos la diseñada por M. Weiner en 1993, consistente en 5760 chips capaces de realizar 16 codificaciones simultáneas, con lo cual encontraría claves en menos de dos días de trabajo ininterrumpido, ésta máquina tendría un costo de 100,000 dólares.

A pesar de esto, hasta 1997 no se tenía conocimiento de que alguien hubiese probado haber decodificado un mensaje codificado con DES, pero en enero de este año la Compañía RSA lanzó un reto en su conferencia anual, consistente en decodificar un mensaje codificado con DES. El premio a este reto, eran 10,000 dólares. El 17 de junio del mismo año, fue decodificado el mensaje utilizando aproximadamente 75,000 computadoras conectadas por internet, habiéndose explorado tan solo el 25% de las posibilidades, mediante una exploración por bloques de claves. Debido a lo anterior se puede considerar que esto fue tan sólo suerte.

Ataques con texto en claro conocido. Este tipo de ataque, supone que el criptoanalista tiene acceso a parejas arbitrarias de texto en claro y su correspondiente texto codificado, utilizando dichas parejas para encontrar la clave. Sin embargo, no se conoce ningún método que permita romper el DES con menos de 2^{64} de estas parejas.

Ataques con texto en claro conocido. En este caso el criptoanalista tiene la posibilidad de elegir los textos en claro y obtener sus correspondientes codificaciones, utilizando dichas parejas para efectuar la búsqueda de la clave. En estas condiciones, se supone que el criptoanalista tiene más información que en el caso anterior al haber sido capaz de elegir los textos en claro. Sin embargo, hasta 1991 no se conocía ningún método que requiera menos de 2^{52} parejas de mensajes para poder romper el DES.

Criptoanálisis diferencial. En 1991, E. Biham y A. Shamir introducen este tipo de ataque, el cual se había mantenido en secreto por IBM. Este ataque está basado en comparaciones del OR exclusivo de dos textos en claro escogidos con el correspondiente OR exclusivo de sus correspondientes codificados. Este método permite romper el DES utilizando 2^{47} parejas de texto en claro conocidos, siendo aproximadamente 2^{36} las parejas útiles para el criptoanálisis, con un tiempo de cálculo equivalente a 2^{37} codificados. A pesar de lo anterior, este método supone aún la realización de un gran número de operaciones.

Uno de los principales problemas a los que se enfrenta este método es la necesidad de obtener el número de mensajes codificados y en claro seleccionados, por el criptoanalista, lo cual en la práctica resulta bastante difícil. Sin embargo, los autores indican que con una versión avanzada del método y siempre que se disponga de una memoria suficiente (10^{10} Mb), las 2^{47} parejas necesarias pueden obtenerse, pudiendo romper el sistema en tiempo real incluso sí la clave es cambiada frecuentemente.

Criptoanálisis lineal. En 1994, M. Matsui presentó un nuevo procedimiento de criptoanálisis basado en la idea de obtener un modelo matemático lineal que represente con una cierta probabilidad la relación existente entre algunos de los bits del mensaje en claro, del codificado y de la clave en un sistema DES. Este método, requiere la utilización de 2^{47} bloques de texto para poder obtener un bit de la clave del esquema básico del DES (codificado con ECB).

II.4.- GOST

Este procedimiento surgido en la desaparecida Unión Soviética fue publicado parcialmente en la conferencia EUROCRYPT'95, este método[11] de codificación es equivalente al DES, codifica bloques de 64 bytes con una clave de 256 bytes utilizando 32 iteraciones de codificación. El método dispone a su vez de un equivalente a las cajas *S* del DES, con la diferencia de que en el GOST, las cajas tienen cuatro bytes de entrada y cuatro de salida y son secretas. Las subclaves son de 32 bytes que se aplican sucesivamente y en orden a las 24 primeras iteraciones y en orden inverso en las ocho iteraciones restantes. El método GOST, no utiliza tablas de permutaciones sino simples rotaciones. El efecto de difusión es menor que en el caso del DES, por lo que se necesitan ocho iteraciones para que un cambio de un byte afecte a los restantes. Puesto que la caja *S* del GOST es secreta, existen muchas dificultades para hacer su criptoanálisis, pero aparentemente, el tamaño de la clave y el número de iteraciones parecen generar un cierto nivel de seguridad.

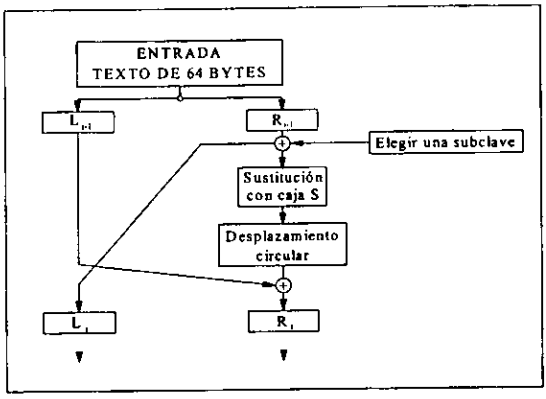


FIG. 2.8- DIAGRAMA DE BLOQUES DE UNA ITERACIÓN DE GOST.

En la actualidad existen una serie de variantes de este método de codificación, que para evitar posibles ataques utilizando criptoanálisis diferencial definen las cajas ***S*** mediante funciones especiales. A su vez, estos métodos alternativos, no solo utilizan la operación de OR exclusivo sino también otras operaciones modulares en diferentes cuerpos algebraicos finitos.

II.5.- SKIPJACK

SKIPJACK[12] es uno de los algoritmos más controvertidos de los últimos años. Fue creado por la NSA (National Security Agency) de los Estados Unidos en 1985 y evaluado desde su creación hasta 1990 cuando se dio a conocer públicamente. Debido a que está considerado como un secreto de alto nivel se conoce poco sobre su funcionamiento.

Este algoritmo codifica bloques de información de 64 bits por medio de 32 rondas tipo Feistel que utilizan una clave de 80 bits. Lo interesante de este algoritmo, es que incorpora el uso de una segunda clave, capaz de decodificar cualquier texto codificado por un usuario, ésta segunda clave es presentada por el usuario ante el gobierno de los EUA, y el gobierno la utilizaría cuando fuese conveniente.

IMPLEMENTACIÓN DE SKIPJACK

La implementación del algoritmo es por medio de hardware dentro de los chips Clipper y Capstone.

Clipper.- Este chip también es conocido como MYK-78K, de diseño antimanipulaciones. VLSI Technologies es la única empresa autorizada por el gobierno de los EUA para fabricarlo y Mykotronx es quien lo programa. Este chip permite utilizar el algoritmo de manera convencional, esto es, codifica y decodifica mensajes con una misma clave.

Capstone.- También conocido con el nombre de MYK-80, es también diseñado por VLSI Technologies, pero a diferencia del chip Clipper, Capstone pretende ser una aplicación comercial de SKIPJACK ya que integra diversas funciones como firma digital, un algoritmo Hash y un generador de números aleatorios. La primer aplicación de este chip fue la tarjeta Fortezza.

El punto de controversia sobre el cual, gira el algoritmo es que integra la utilización de una clave capaz de decodificar todo mensaje codificado con este algoritmo. Esta clave se deposita en dos partes en dos fideicomisos distintos además de que el usuario debe de proporcionar el número de serie del chip.

El gobierno de los EUA puede reclamar en cualquier momento a los fideicomisos las dos partes de la clave, esto siempre y cuando el gobierno tenga una razón justificada (tal como, que el usuario fuese un posible terrorista) y que se pueda demostrar de forma legal. Este procedimiento es comparable a la expedición de una orden de cateo por parte del poder judicial, sólo que en este caso se le puede llamar cateo informático.

Además de lo anterior, se sabe que cada vez que se establece una comunicación, se selecciona una clave de sesión de 80 bits con la que se codifica la información; pero al mismo tiempo, también se transmite un campo de defensa de la ley, esto es, un archivo anexo al

mensaje que contiene los datos del chip y la clave de sesión, con los cuales se puede saber quien es el dueño del chip y se puede tener una forma certera de demostrar que el mensaje que el gobierno pudiese haber recuperado es el mismo que el usuario ha enviado.

SEGURIDAD DE SKIPJACK

Para muchos criptoanalistas profesionales, este algoritmo parece ser algo similar a una torre de naipes, debido a que ha sido extremadamente fácil, en ocasiones, decodificar mensajes codificados con una versión simplificada o modificada de este algoritmo, pero no ha sido posible hacerlo con un mensaje codificado con una versión completa de SKIPJACK. Incluso, la propia NSA ha asegurado que pasarán cerca de 35 años antes de que la dificultad de romper SKIPJACK mediante un ataque con fuerza bruta sea equivalente al esfuerzo requerido en la actualidad para romper el DES.

Por otra parte, debido a la clasificación del algoritmo SKIPJACK como un secreto de alto nivel, hace suponer que la NSA trata de esconder alguna posible debilidad existente en su programación, aunque la misma NSA asegura que esto lo ha hecho para no poner en riesgo todo tipo de documento clasificado y codificado con este algoritmo.

En años recientes, el SKIPJACK fue desclasificado por el gobierno de los EUA. Las razones para tomar esta medida, fueron de índole tecnológico.

LA DESCLASIFICACIÓN DE SKIPJACK

DMS (Defense Messaging System) es un sistema clasificado de mensajes por computadora; similar al e-mail, que utiliza tarjetas Fortezza PCMIA como medida de seguridad. Como algunos de los algoritmos de Fortezza estaban clasificados, las tarjetas con Fortezza tenían todos los controles físicos y características de resistencia a la manipulación que se necesitaban para proteger dichos algoritmos.

Dentro de los protocolos DMS no hay manera de tener varios codificadores distintos. S/MIME por ejemplo, define múltiples algoritmos, hay una variable en el mensaje S/MIME que dice al receptor qué algoritmos fueron utilizados para codificar ese mensaje en particular.

El problema surgió porque la NSA no era capaz de instalar tarjetas Fortezza y lectores lo suficientemente rápido, esto quizá, debido a su elevado costo. En cualquier caso, un usuario que deseara participar dentro de DMS se encontraba ante la falta del hardware necesario. Sí la NSA hubiera podido establecer un conjunto de algoritmos alternativos en DMS, entonces hubieran podido distribuir una versión sólo de software con algoritmos no clasificados: triple-DES, Diffie-Hellman, etc. Cada terminal sabría sí estaba comunicándose con un DMS con Fortezza habilitada o con un DMS sólo software y el problema desaparecería. Pero el DMS no podía permitir esto, o se usa SKIPJACK/KEA o nada. Así que o eliminaban el sistema de codificación o pasaban los algoritmos clasificados a software. En este caso se eligió la primer opción.

II.6.- EL ALGORITMO IDEA™ (INTERNATIONAL DATA ENCRIPTION ALGORITHM)

El PEES (Proposed European Encryption Standard) es un codificador de bloques de 64 bits compuesto de una clave de 128 bits. Fue creado en Suiza por Xuejia Lai y James Massey y propuesto en la *Eurocrypt'90*[13][14], un año más tarde este algoritmo ya modificado fue presentado en *Eurocrypt'91* como IDEA™ (International Data Encryption Algorithm).

En IDEA™, tanto los datos en claro como los datos codificados, están compuestos por bloques de 64 bits, mientras que la clave consta de 128 bits. El método de expansión de la clave se inicia dividiendo la palabra clave de 128 bits introducida por el usuario, en 8 subclaves de 16 bits que constituye los 8 primeros sub-bloques de claves. A continuación se rota la clave 25 bits hacia la izquierda y se obtienen los siguientes 8 sub-bloques de claves, que

serán divididas para formar el siguiente grupo de sub-bloques de claves, esta operación se repite sucesivamente hasta formar un total de 52 sub-bloques de claves.

El algoritmo consiste en 8 iteraciones de codificación idénticas, seguidas de una transformación de salida. La codificación se basa en el concepto de mezclar operaciones aritméticas de 3 grupos algebraicos diferentes sobre pares de sub-bloques diferentes de 16 bits, tomando en cuenta:

1. Multiplicación modular con módulos de $Z_{2^{16}-1}$ con el símbolo "⊗".
2. Grupo aditivo en $Z_{2^{16}}$ con el símbolo "+".
3. Grupo aditivo en Z_2 de las 16- uplas, bit a bit con el símbolo "⊕".

Los bloques de texto de 64 bits son divididos en 4 segmentos de 16 bits, los cuales nombraremos como X1, X2, X3 y X4. Los sub-bloques de claves serán k1, k2, k3.....k52.

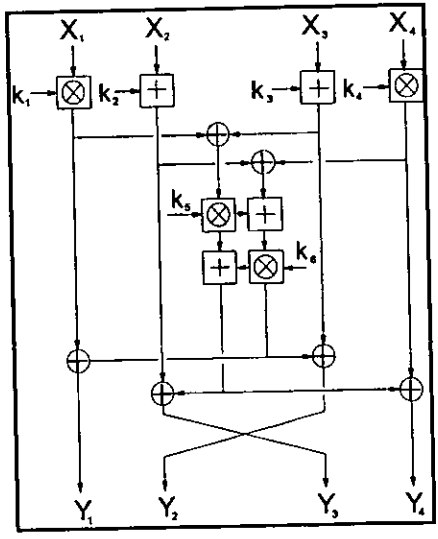


FIG. 2.9- DISEÑO DE LA PRIMER ITERACIÓN DE IDEA™.

X1 x k1 α Y1	Y6 + Y7 α Y8
X2 + k2 α Y2	Y8 x k6 α Y9
X3 + k3 α Y3	Y7 + Y9 α Y10
X4 x k4 α Y4	Y1 XOR Y9 α Y11
Y1 XOR Y3 α Y5	Y3 XOR Y9 α Y12
Y2 XOR Y4 α Y6	Y2 XOR Y10 α Y13
Y5 x k5 α Y7	Y4 XOR Y10 α Y14

Después de este proceso los bloques de salida Y12 y Y13 son intercambiados, de modo que Y11, Y13, Y12 y Y14 son usados en ese orden como entrada para la siguiente iteración junto con las siguientes subclaves k7 a k12. Al final de las 8 iteraciones de codificación tendremos 4 bloques de salida que llamaremos e1, e2, e3 y e4. Estos cuatro bloques entran al ya mencionado último paso, la transformación de salida.

e1 x k49 α c1

e2 + k50 α c2

e3 + k51 α c3

e4 x k52 α c4

Los últimos cuatro bloques de salida son reacomodados para formar bloques de 64 bits de texto codificado.

Todas las operaciones utilizadas en IDEA™ son invertibles en sí mismas, pero incompatibles entre sí, en el sentido de que no gozan de la ley distributiva ni asociativa, no forman un grupo y la sucesión de ellas no puede dar lugar a duda a una cancelación de operaciones.

Como se observa, en cada iteración de codificación, el bloque de datos de entrada de **X** es dividido en cuatro sub-bloques de 16 bits X_1, X_2, X_3 y X_4. Cada iteración emplea una subclave diferente.

Este algoritmo presenta las siguientes diferencias con el DES que lo hacen más atractivo:

- ✓ El tamaño de la clave es mucho más grande, por lo cual es más difícil de descifrar ($3.4 \cdot 10^{38}$ combinaciones).
- ✓ Todas las operaciones de codificación son algebraicas, sin necesidad de utilizar cajas ***S***.
- ✓ No hay operaciones a nivel de bit, lo cual facilita su programación en alto nivel.
- ✓ Es más eficiente que los algoritmos de tipo Feistel, porque a cada iteración se modifican todos los bits del bloque y no solamente la mitad.
- ✓ Se pueden utilizar todos los tipos de operación definidos para el DES.

El proceso de decodificación es esencialmente el mismo que el de codificación, con la diferencia que los sub-bloques de clave de decodificación son distintos, calculándose como los inversos de los sub-bloques de codificación y también en orden inverso.

En la actualidad la compañía Ascom Systec Ltd. tiene los derechos de patente sobre el algoritmo IDEA™ por lo cual es una marca registrada, patentándolo tanto en Suiza, toda Europa y Estados Unidos. Actualmente se comercializa el algoritmo a través del chip IDEACrypt, el cual asegura tener un alto nivel de seguridad basado en el hecho de que almacena las claves dentro del mismo y según la compañía es prácticamente imposible que alguien pueda leer las claves desde fuera del chip.

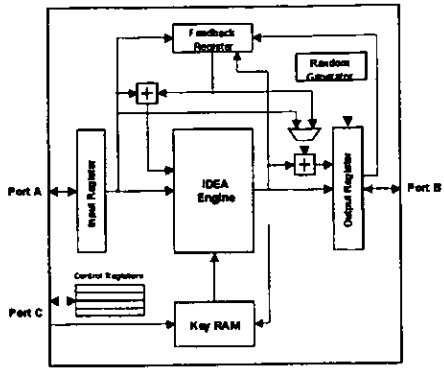


FIG. 2.10- DIAGRAMA DE INSTALACIÓN DEL EQUIPO ascom IDEA™.

Según la compañía Ascom Ltd. el chip IDEACrypt es extremadamente flexible en consideración a sus interfaces[15][16], es decir, contiene dos puertos de datos que son optimizados según su desempeño y pueden ser configurados con respecto a su tamaño (8, 16 ó 32 bits). La configuración de los puertos de datos son controladas por el puerto de control de registro. Por razones de seguridad, un puerto de control de 8 bits se proporciona de manera separada para control y manejo de claves. IDEACrypt puede ser configurado directamente con una interface con las siguientes familias de procesadores: Intel i960, Intel 80xx, Motorola 683xx, MPC 860; así como con las tarjetas controladoras PCI y PCMCIA.

IDEACrypt es comúnmente utilizada para operaciones de comercio electrónico, codificación de archivos, codificación de información almacenada en disco duro, aplicaciones financieras y para proporcionar seguridad en comunicaciones a alta velocidad.

Actualmente el equipo proporcionado por la compañía Ascom está basado en el algoritmo IDEA, es utilizado en muchas compañías en todo el mundo, como son:

Banco DBS, Singapur	Union Bank de Suiza
Corporación Bancaria Suiza	Audi AG, Alemania
BMW Rolls Royce GmbH, Alemania	Mitsubishi Electric, Japón
Motorola, USA	Nissan Electric Co., Japón

Siemens, Alemania	Volkswagen AG, Alemania
Munich Airport, Alemania	Cray Communications, Dinamarca
FTP Software, USA	Highware Inc., Belgica
IBM, USA	Lange Electronic, Alemania
Dirección General Impositiva, Argentina	Marx Datentechnik, Alemania
Netscape Communications, USA	Network Systems Corp., USA
Deutsche Telekom, Alemania	Utimaco Safeware, Alemania
Federal Defense Department, Suiza	Reuters Asia, Hong Kong

SEGURIDAD DE IDEA

En primer lugar, el ataque por fuerza bruta de IDEA resulta impracticable, ya que es necesario probar 10^{38} claves, cantidad imposible de manejar con los medios informáticos actuales.

Los autores de IDEA han demostrado que es inmune al criptoanálisis diferencial a partir de la cuarta iteración.

En cuanto a la distributibidad de las operaciones involucradas en IDEA, Meier W. afirma haber encontrado que las operaciones $\oplus$ y + satisfacen a una ley distributiva parcial, basándose en la aritmética módulo $2^{16}+1$, la cual ha sido explotada para realizar un criptoanálisis de las dos primeras iteraciones de IDEA, aunque no resulta útil para el criptoanálisis de IDEA completo con ocho iteraciones.

El mecanismo de expansión permite una vía de ataque al algoritmo, pues si el criptoanalista ha llegado a deducir algunos de los sub-bloques de la clave expandida, puede llegar a obtener la clave del usuario en su totalidad.

II.7.- BLOWFISH

Blowfish[17][18][19] es un algoritmo de tipo Feistel. Fue diseñado en Inglaterra por Bruce Schneier en 1993. Este algoritmo codifica bloques de texto de 64 bits por medio de 16 iteraciones del tipo Feistel utilizando una clave de longitud variable, la cual puede ir desde los 32 bits hasta los 448 bits.

Este algoritmo es significativamente más rápido que el DES o IDEA, cuando es implementado en microprocesadores de 32 bits con grandes cachés de datos tales como los procesadores Pentium y Power PC.

El algoritmo consiste en dos secciones: una sección de expansión de claves y una sección de codificación de datos. En la primer sección, se convierte una clave de hasta 448 bits en varios arreglos de subclaves totalizando 4168 bytes.

El bloque de texto original es dividido en dos mitades de 32 bits cada una. La mitad izquierda es sumada mediante una operación OR exclusivo a la subclave en turno, mientras que a la mitad derecha se le aplica un OR exclusivo con la función f de la mitad izquierda. Una vez realizado lo anterior, la mitad izquierda toma el lugar de la derecha, lo mismo sucede con esta última. Esta operación se repite hasta la penúltima iteración.

En la última iteración solo se aplica un OR exclusivo a cada mitad, la mitad derecha con la subclave $\mathbf{K}_{18}$ y a la izquierda con la subclave $\mathbf{K}_{17}$.

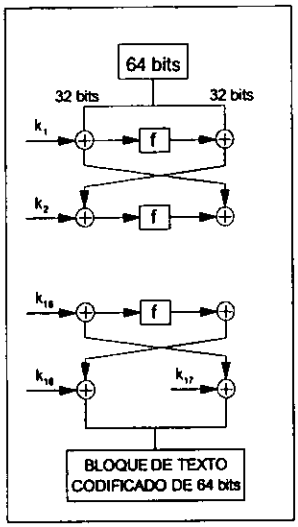


FIG. 2.11- DIAGRAMA DE BLOQUES DEL BLOWFISH.

FUNCIÓN *f*

La función *f* consiste en dividir la mitad izquierda en cuatro bloques de 8 bits cada uno (**a, b, c, d**), los cuales son introducidos en cuatro cajas **S**. Cada caja S consta de 256 entradas cada una. Después de haber pasado por las cajas **S**, cada bloque de 8 bits se convierte en bloques de 32 bits cada uno (**A, B, C, D**). El primer bloque resultante (**A**) es adicionado al segundo bloque (**B**) mediante una suma módulo 32, el resultado es adicionado mediante un OR exclusivo al tercer bloque resultante (**C**) y este último resultado es adicionado al cuarto bloque resultante (**D**) mediante una suma módulo 32.

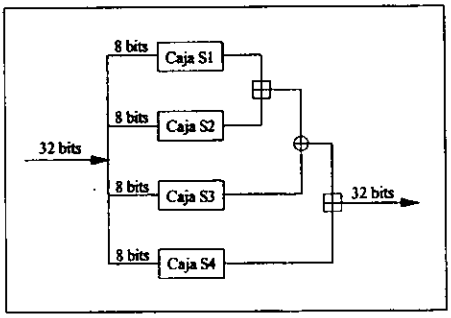


FIG. 2.12- LA FUNCIÓN *f*.

GENERACIÓN DE SUBCLAVES

El Blowfish utiliza un gran número de subclaves. Esas claves deben ser precalculadas antes de cualquier operación de codificación o decodificación para lo cual se requiere de 521 complejas iteraciones, las cuales mezclan operaciones de OR exclusivo y substituciones, además de una fase de inicialización, de esta manera se generan 18 subclaves de 32 bits cada una.

SEGURIDAD DEL BLOWFISH

Este algoritmo ha sido ampliamente analizado y hasta la fecha no ha sido posible decodificar algún mensaje codificado con BLOWFISH. Debido a lo anterior BLOWFISH está ganando cierto prestigio como un magnífico criptosistema.

La revista Dr. Dobbs Journal lanzó un reto para decodificar un mensaje codificado con Blowfish. El ganador recibiría 1,000,000 dólares si lograba la decodificación antes de abril de 1995. Algunos criptoanalistas, tales como John Kelsey pudieron romper hasta la tercera ronda del algoritmo, otros como Serge Vaudenay analizaron versiones simplificadas de Blowfish logrando un criptoanálisis aceptable mediante la técnica de texto elegido hasta una octava ronda de codificación, pero a pesar de estos esfuerzos aun no se ha dado una solución satisfactoria.

Este algoritmo no esta patentado por lo cual es cien por ciento público y no existen restricciones para su uso o comercialización en el Reino Unido o en el extranjero. La licencia para el uso de este algoritmo es gratuita.

II.8.- CAST 128

El CAST-128[20] pertenece a la clase de algoritmos de codificación conocidos como codificadores de Feistel, casi todas las operaciones son similares al algoritmo DES. El algoritmo de codificación está dado por los siguientes cuatro pasos:

1- Se calculan 16 pares de subclaves a partir de la clave principal.
2- Se divide el bloque de texto en dos mitades de 32 bits cada una (L_0=1..32 y R_0=33..64).
3- Se llevan a cabo 16 iteraciones en las cuales se calcula L_i y R_i de la siguiente forma:

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

$$R_i=L_{i-1} \oplus f(R_{i-1}, k1_i, k2_i)$$

Existen tres tipos de funciones *f* las cuales cambian dependiendo del número de iteración de codificación.

4- Se intercambian los bloques finales (L_{16} y R_{16}) y se concatenan para formar el bloque de texto codificado.

El algoritmo de decodificación es el mismo que el utilizado para la codificación, la única diferencia es que las iteraciones y los pares de subclaves son utilizados en orden inverso al calculado en la codificación.

El CAST utiliza un par de subclaves por iteración: 32 bits de $k1_i$, son utilizados como claves de enmascaramiento, mientras que los 5 bits de $k2_i$ son utilizados como clave de rotación.

FUNCIONES *f*

Como se mencionó anteriormente, el CAST-128 utiliza 3 diferentes funciones *f* definidas por las siguientes ecuaciones:

Función $f1$: $I = ((k1_i + R_{i-1}) <<< k2_i)$

$f = ((S1[Ia] \oplus S2[Ib]) - S3[Ic]) + S4[Id]$

Función $f2$: $I = ((k1_i \oplus R_{i-1}) <<< k2_i)$

$f = ((S1[Ia] - S2[Ib]) + S3[Ic]) \oplus S4[Id]$

Función $f3$: $I = ((k1_i - R_{i-1}) <<< k2_i)$

$f = ((S1[Ia] + S2[Ib]) \oplus S3[Ic]) - S4[Id]$

Donde $\oplus$ representa la operación OR exclusivo, + y - son suma y resta de bits módulo 32 respectivamente; Ia, Ib, Ic e Id son de los bits más significativos a los bits menos significativos de I, <<< representa una rotación circular de bits hacia la izquierda.

Cada una de las funciones f arriba mencionadas se utilizan alternadamente según el número de ronda de codificación que se este llevando a cabo(tabla 2.10).

	Número de ronda de codificación
Función f1	1, 4, 7, 10, 13 y 16
Función f2	2, 5, 8, 11 y 14
Función f3	3, 6, 9, 12 y 15

TABLA 2.10.- RELACIÓN ENTRE EL NÚMERO DE RONDA Y LA FUNCIÓN *f* APLICADA.

CAJAS S

CAST-128 utiliza ocho cajas de substitución: S1, S2, S3 y S4 son utilizadas como cajas de función en cada ronda. Las cajas S5, S6, S7 y S8 son cajas utilizadas en la generación de las subclaves.

GENERACIÓN DE SUBCLAVES

Al principio la clave de 128 bits es dividida en cuatro bloques de 32 bits cada uno llamados **K1, K2, K3** y **K4** los cuales a su vez constan de cuatro bytes (p. ej. $\mathbf{K1_A}$, $\mathbf{K1_B}$, $\mathbf{K1_C}$ y $\mathbf{K1_D}$) con ocho bits cada uno. Dentro del algoritmo se utilizan bloques temporales (**T1, T2, T3** y **T4**) los cuales a su vez tienen cuatro bytes con la misma notación que los provenientes de la clave principal.

Para la generación de las primeras dieciséis subclaves se utilizan las siguientes ecuaciones:

$\mathbf{T1} = \mathbf{K1} \oplus S5[K4_B] \oplus S6[\mathbf{K4_D}] \oplus S7[K4_A] \oplus S8[\mathbf{K4_C}] \oplus S7[\mathbf{K3_A}]$

$\mathbf{T2} = \mathbf{K3} \oplus S5[\mathbf{T1_A}] \oplus S6[\mathbf{T1_C}] \oplus S7[\mathbf{T1_B}] \oplus S8[\mathbf{T1_D}] \oplus S8[\mathbf{K3_C}]$

$\mathbf{T3} = \mathbf{K4} \oplus S5[\mathbf{T2_D}] \oplus S6[\mathbf{T2_C}] \oplus S7[\mathbf{T2_B}] \oplus S8[\mathbf{T2_A}] \oplus S5[\mathbf{K3_B}]$

$\mathbf{T4} = \mathbf{K2} \oplus S5[\mathbf{T3_C}] \oplus S6[\mathbf{T3_B}] \oplus S7[\mathbf{T3_D}] \oplus S8[\mathbf{T3_A}] \oplus S6[\mathbf{K3_D}]$

$k1 = S5[\mathbf{T3_A}] \oplus S6[\mathbf{T3_B}] \oplus S7[\mathbf{T2_D}] \oplus S8[\mathbf{T2_C}] \oplus S5[\mathbf{T1_C}]$

$k2 = S5[\mathbf{T3_C}] \oplus S6[\mathbf{T3_D}] \oplus S7[\mathbf{T2_B}] \oplus S8[\mathbf{T2_A}] \oplus S6[\mathbf{T2_C}]$

$k3 = S5[\mathbf{T4_A}] \oplus S6[\mathbf{T4_B}] \oplus S7[\mathbf{T1_D}] \oplus S8[\mathbf{T1_C}] \oplus S7[\mathbf{T3_B}]$

$k4 = S5[\mathbf{T4_C}] \oplus S6[\mathbf{T4_D}] \oplus S7[\mathbf{T1_B}] \oplus S8[\mathbf{T1_A}] \oplus S8[\mathbf{T4_A}]$

$\mathbf{K1} = \mathbf{T3} \oplus S5[\mathbf{T2_B}] \oplus S6[\mathbf{T2_D}] \oplus S7[\mathbf{T2_A}] \oplus S8[\mathbf{T2_C}] \oplus S7[\mathbf{T1_A}]$

$\mathbf{K2} = \mathbf{T1} \oplus S5[\mathbf{K1_A}] \oplus S6[\mathbf{K1_C}] \oplus S7[\mathbf{K1_B}] \oplus S8[\mathbf{K1_D}] \oplus S8[\mathbf{T1_C}]$

$\mathbf{K3} = \mathbf{T2} \oplus S5[\mathbf{K2_D}] \oplus S6[\mathbf{K2_C}] \oplus S7[\mathbf{K2_B}] \oplus S8[\mathbf{K2_A}] \oplus S5[\mathbf{T1_B}]$

$\mathbf{K4} = \mathbf{T4} \oplus S5[\mathbf{K3_C}] \oplus S6[\mathbf{K3_B}] \oplus S7[\mathbf{K3_D}] \oplus S8[\mathbf{K3_A}] \oplus S6[\mathbf{T1_D}]$

$k5 = S5[\mathbf{K1_D}] \oplus S6[\mathbf{K1_C}] \oplus S7[K4_A] \oplus S8[K4_B] \oplus S5[\mathbf{K3_A}]$

$k6 = S5[\mathbf{K1_B}] \oplus S6[\mathbf{K1_A}] \oplus S7[\mathbf{K4_C}] \oplus S8[\mathbf{K4_D}] \oplus S6[K4_B]$

$k7 = S5[\mathbf{K2_D}] \oplus S6[\mathbf{K2_C}] \oplus S7[\mathbf{K3_A}] \oplus S8[\mathbf{K3_B}] \oplus S7[\mathbf{K1_D}]$

$k8 = S5[K2_B] \oplus S6[K2_A] \oplus S7[K3_C] \oplus S8[K3_D] \oplus S8[K2_D]$

$T1 = K1 \oplus S5[K4_B] \oplus S6[K4_D] \oplus S7[K4_A] \oplus S8[K4_C] \oplus S7[K3_A]$

$T2 = K3 \oplus S5[T1_A] \oplus S6[T1_C] \oplus S7[T1_B] \oplus S8[T1_D] \oplus S8[K3_C]$

$T3 = K4 \oplus S5[T2_D] \oplus S6[T2_C] \oplus S7[T2_B] \oplus S8[T2_A] \oplus S5[K3_B]$

$T4 = K2 \oplus S5[T3_C] \oplus S6[T3_B] \oplus S7[T3_D] \oplus S8[T3_A] \oplus S6[K3_D]$

$k9 = S5[T1_D] \oplus S6[T1_C] \oplus S7[T4_A] \oplus S8[T4_B] \oplus S5[T3_B]$

$k10 = S5[T1_B] \oplus S6[T1_A] \oplus S7[T4_C] \oplus S8[T4_D] \oplus S6[T4_A]$

$k11 = S5[T2_D] \oplus S6[T2_C] \oplus S7[T3_A] \oplus S8[T3_B] \oplus S7[T1_C]$

$k12 = S5[T2_B] \oplus S6[T2_A] \oplus S7[T3_C] \oplus S8[T3_D] \oplus S8[T2_C]$

$K1 = T3 \oplus S5[T2_B] \oplus S6[T2_D] \oplus S7[T2_A] \oplus S8[T2_C] \oplus S7[T1_A]$

$K2 = T1 \oplus S5[K1_A] \oplus S6[K1_C] \oplus S7[K1_B] \oplus S8[K1_D] \oplus S8[T1_C]$

$K3 = T2 \oplus S5[K2_D] \oplus S6[K2_C] \oplus S7[K2_B] \oplus S8[K2_A] \oplus S5[T1_B]$

$K4 = T4 \oplus S5[K3_C] \oplus S6[K3_B] \oplus S7[K3_D] \oplus S8[K3_A] \oplus S6[T1_D]$

$k13 = S5[K3_A] \oplus S6[K3_B] \oplus S7[K2_D] \oplus S8[K2_C] \oplus S5[K1_D]$

$k14 = S5[K3_C] \oplus S6[K3_D] \oplus S7[K2_B] \oplus S8[K2_A] \oplus S6[K2_D]$

$k15 = S5[K4_A] \oplus S6[K4_B] \oplus S7[K1_D] \oplus S8[K1_C] \oplus S7[K3_A]$

$k16 = S5[K4_C] \oplus S6[K4_D] \oplus S7[K1_B] \oplus S8[K1_A] \oplus S8[K4_B]$

Para el cálculo de las dieciséis subclaves restantes se siguen las mismas operaciones.

TAMAÑO VARIABLE DE CLAVE

Como se mencionó anteriormente este algoritmo permite la utilización de distintos tamaños de claves, desde 40 hasta 128 bits en incrementos de 8 bits (40, 48, 56,..., 112, 120 y 128 bits) para las cuales se tienen que cumplir las siguientes condiciones durante su operación.

1. Para claves de tamaño menor o igual a 80 bits el algoritmo se utilizará con sólo 12 iteraciones en lugar de 16.
2. Para claves cuyo tamaño de clave sea superior a los 80 bits se utilizarán las 16 iteraciones.
3. Para claves cuyo tamaño sea menor a 128 bits, la clave será rellenada con ceros en las posiciones de los bits más significativos o menos significativos en un paso de iniciación o ajuste del tamaño de la clave, esto es porque el CAST asume que el tamaño de la clave de entrada es de 128 bits.

Cabe remarcar que el CAST 128 es capaz de soportar todos los tamaños de clave antes descritos pero los tamaños de 40, 64, 80 y 128 bits son los más comúnmente utilizados.

SEGURIDAD DEL ALGORITMO

Debido a que el algoritmo utiliza tres diferentes funciones *f*, cualquier ataque por medio de criptoanálisis diferencial o lineal se vuelve impracticable. Por otra parte, aún no se conoce a nadie que demuestre haber decodificado algún texto codificado con este algoritmo.

La utilización de este algoritmo no requiere del pago de licencias ante algún autor, esto es válido, tanto para uso comercial como privado.

II.9.- SAFER-64 (Secure And Fast Encryption Routine)

Este método fue diseñado por J. Massey e incorporado en productos de la Compañía Norteamericana Cylink Corp. Safer-64[22] recibe su nombre porque codifica bloques de texto de 64 bits. El Gobierno de Singapur va a incorporar en algunas de sus aplicaciones una modificación del mismo que utiliza una clave de 128 bits. El método SAFER comenzó a aplicarse con 6 iteraciones de codificado siendo inmune a los ataques basados en el criptoanálisis diferencial. Sin embargo, en este tipo de implementaciones se encontraron claves

débiles con efectos nocivos que debilitaban el método. En consecuencia, se recomienda que éste se aplique con una implementación de 8 iteraciones. Cada iteración es una mezcla de las combinaciones de OR exclusivo, adiciones, potencias y logaritmos discretos que dan el cuerpo finito Z_{257} en el que 45 es un elemento primitivo. La introducción de operaciones no lineales en cuerpos finitos se refleja igualmente en el diseño de otros codificadores en bloque posteriores.

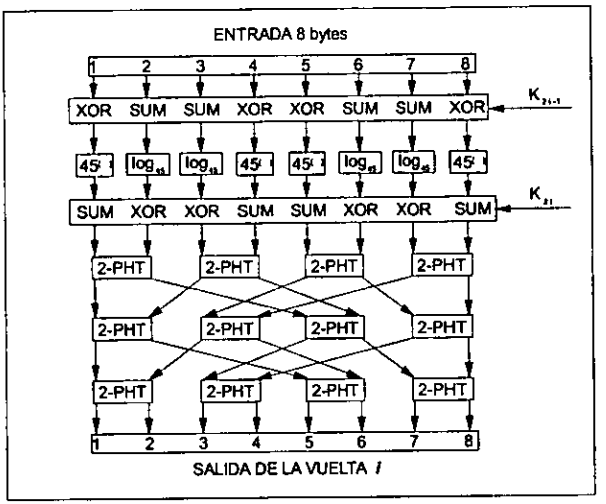


FIG. 2.13- DIAGRAMA DE BLOQUES DEL MÉTODO SAFER-64.

El texto a codificar se divide en bloques de 64 bits y cada uno de ellos a su vez en 8 bits. Estos se suman (alternativamente mediante un OR exclusivo y una suma normal) con los bits correspondientes a una de las subclaves de cada iteración. Los resultados se transforman igualmente en forma alternativa mediante potencias o logaritmos discretos en Z_{257} y nuevamente mediante sumas u OR exclusivos con los bits de otra de las subclaves de cada iteración. A continuación, los resultados de las operaciones anteriores pasan por tres niveles de operaciones lineales denominadas PTH o PHT (Pseudo Transformaciones de Hadamard), definidas mediante las ecuaciones:

$$b_1 \equiv (2a_1 + a_2) \pmod{256}$$

$$b_2 \equiv (a_1 + a_2) \pmod{256}$$

donde a_1 y a_2 son las entradas a la PTH y, b_1 y b_2 las salidas. Estas operaciones se utilizan básicamente para aumentar la difusión de bits, por lo que se refiere a las subclaves, éstas se generan mediante una rotación de la clave de codificación K, de acuerdo a la ecuación:

$$K_{i+1} = (K <<< 3i) + ci$$

donde el símbolo '$x<<<y$' representa una rotación circular de x de y bits hacia la izquierda y "ci" es una constante de rotación. El método SAFER es inmune al criptoanálisis diferencial. Sin embargo, la existencia de claves débiles, es decir, claves que transforman distintos textos en claro en el mismo codificado, sugieren la utilización de este método con cierta precaución.

II.10.- AKELARRE

Akelarre[24] fue creado en 1996, es un codificador de datos de tipo Feistel que trabaja en bloques de texto por separado, la principal característica que presenta, es la posibilidad de seleccionar la longitud de la clave, el tamaño del bloque a codificar y el número de iteraciones a utilizar, lo cual le presenta al usuario una amplia flexibilidad operativa al permitirle la posibilidad de codificar información con distintos niveles de seguridad con la misma clave variando únicamente el número de iteraciones de codificación o el tamaño del bloque para así optimizar al máximo el tiempo de codificación.

Este algoritmo cuenta con tres bloques bien definidos, en primer lugar, se lleva a cabo una transformación inicial, en segundo lugar, se encuentran los ciclos de codificación y en tercer lugar, una transformación de salida. Los datos se someten a un proceso de reorganización de forma previa a su entrada en el algoritmo, este proceso consiste en una división en bloques **X** de una longitud de **4w** bits. Una vez que han sido introducidos dentro del algoritmo, estos bloques son a su vez divididos en 4 sub-bloques de w bits cada uno ($X1$, $X2$, $X3$, $X4$). Estos cuatro sub-bloques constituyen la entrada a la transformación inicial tras la cual se llevan a cabo v iteraciones de codificación, el cual como ya se mencionó, puede ser

configurado por el usuario. En cada una de las iteraciones, los cuatro sub-bloques se suman y rotan unos con otros y con los *n* sub-bloques de tamaño *w* procedentes de la clave. Después de la última iteración, se lleva a cabo una transformación de salida y se enlazan de nuevo los cuatro sub-bloques codificados resultantes (*Y1*, *Y2*, *Y3*, *Y4*) para conformar el bloque codificado Y de salida.

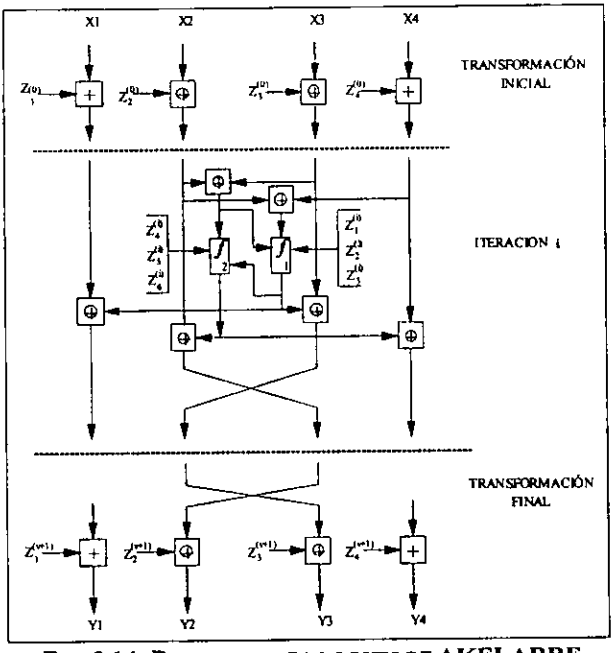


FIG. 2.14- DIAGRAMA DE BLOQUES DE AKELARRE.

GENERACIÓN DE SUBCLAVES.

El algoritmo para la generación de las subclaves de codificación consiste en elevar al cuadrado una subclave de *w* bits tomada de la clave introducida por el usuario. A continuación, se le suma una constante *C*. Los bits de la suma resultante se utilizan para generar un nuevo bloque de *2w* bits, del cual los *w/2* bits menos significativos y los *w/2* bits más significativos serán utilizados para generar la subclave. Este proceso se repite para generar la totalidad de las subclaves requeridas.

SEGURIDAD DEL ALGORITMO

Según N. Ferguson y B. Schneier, Akelarre puede ser atacado mediante el sistema de texto conocido utilizando 100 parejas de texto original y codificadas. Debido a lo anterior, ambos hacen sugerencias para hacer más fuerte al algoritmo, sin embargo concluyen que este algoritmo es poco seguro y por tanto sería mejor explorar otras alternativas.

II.11.- FEAL (Fast Encryption Algorithm)

Feal[25] fue creado por los japoneses en años recientes, este algoritmo codifica bloques de texto de 64 bits utilizando una clave del mismo tamaño la cual servirá para generar 16 subclaves de 16 bits cada una.

Como todo algoritmo de Feistel, Feal contiene una sección de generación de claves y un algoritmo principal el cual se repite n veces para poder llevar a cabo una codificación completa.

A la entrada del algoritmo, se lleva a cabo una operación OR exclusivo entre los 64 bits del texto original y 4 subclaves ($\mathbf{K_8}$, $\mathbf{K_9}$, $\mathbf{K_{10}}$ y $\mathbf{K_{11}}$). Después de este paso, el bloque resultante se divide en 2 mitades L_0 y R_0, a la mitad derecha (R_0) se le aplica un OR exclusivo con los bits de la mitad izquierda (L_0), este bloque resultante es introducido a una función f a la cual también entra una de las subclaves ($\mathbf{K_0}$), al valor resultante se le aplica un OR exclusivo con el L_0 y una vez llevado a cabo esto, los sub-bloques se intercambian(Fig. 2.15).

La función f de codificación no contiene cajas S, pero en cambio existen dos funciones con similar denominación $\mathbf{S_0}$ y $\mathbf{S_1}$ (Fig. 2.16).

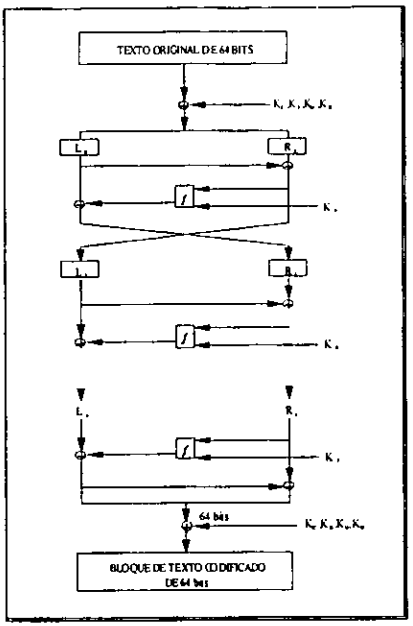


FIG. 2.15- DIAGRAMA DE BLOQUES DEL ALGORITMO FEAL (8 ITERACIONES).

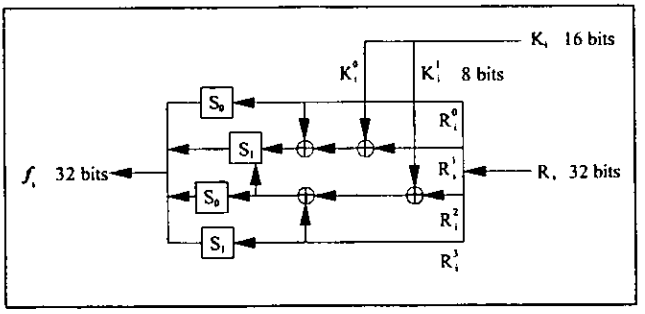


FIG. 2.16- DIAGRAMA DE BLOQUES DE LA FUNCIÓN *f*.

GENERACIÓN DE SUBCLAVES

La clave original de 64 bits es sometida a un proceso parecido al de codificación. En cada una de las iteraciones son generadas dos subclaves por medio de operaciones OR exclusivo y la aplicación de las operaciones S_0 y S_1, las cuales se incluyen dentro de una función f_k parecida a la función *f* del algoritmo de codificación.

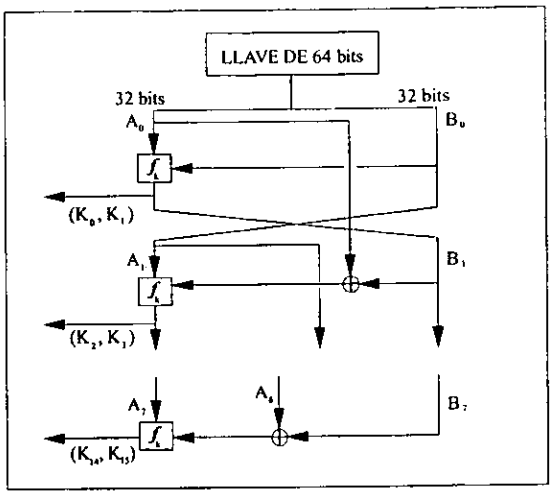


FIG. 2.17- DIAGRAMA DE BLOQUES DEL ALGORITMO GENERADOR DE SUBCLAVES.

La estructura de la función f_k se muestra en la figura 2.18.

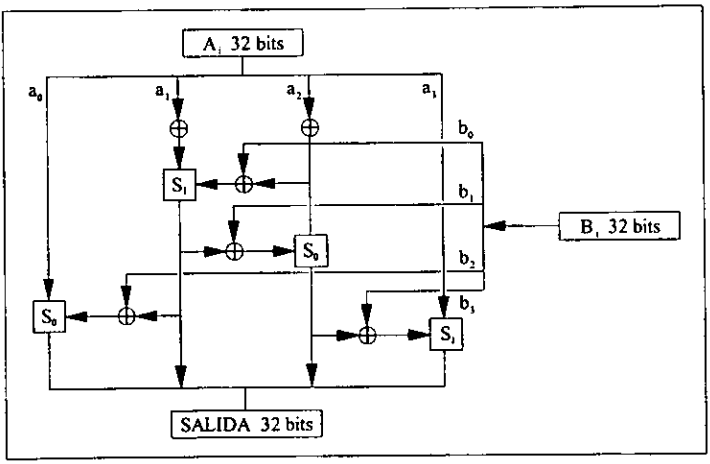


Fig. 2.18- DIAGRAMA DE BLOQUES DE LA FUNCIÓN f_k.

Siendo S_0 y S_1 las funciones definidas por las siguientes ecuaciones:

$$S0 = \left[(A + B)(\text{mod}\,256)\right] \text{ rotar 2 bits a la izquierda}$$

$$S1 = \left[(A + B + 1)(\text{mod}\,256)\right] \text{ rotar 2 bits a la izquierda}$$

SEGURIDAD DEL ALGORITMO

Este procedimiento parecería ser muy seguro debido al tamaño de la clave y a lo complejo de su estructura de codificación, pero no es así. El procedimiento fue presentado con 4 iteraciones inicialmente e inmediatamente fue criptoanalizado y por tanto decodificado. Igualmente sucedió con las implementaciones de 8 y 16 iteraciones. Debido a lo anterior, en la actualidad no se recomienda su uso con menos de 32 iteraciones y para documentos cuya confidencialidad no sea muy alta.

II.12.- LOKI97

Este algoritmo de origen australiano fue diseñado en su primera versión (LOKI89) en 1989, y dado a conocer en la convención Asiacrypt'90. La versión 97 de este algoritmo fue diseñada por los doctores Lawrie Brown, Josef Pieprzyk y Jeniffer Seberry.

LOKI 97[26] es un codificador en bloque tipo Feistel cuyo tamaño de bloque de texto es de 128 bits. Consta de 16 iteraciones, en cada una de éstas se adicionan 2 subclaves de 64 bits a la mitad derecha del bloque (R_i), una antes de la función *f* y una después, así mismo, se utiliza una tercer subclave dentro de la función *f*. A la salida de dicha función *f*, las mitades se invierten tomando así la mitad del lado derecho la posición izquierda y viceversa.

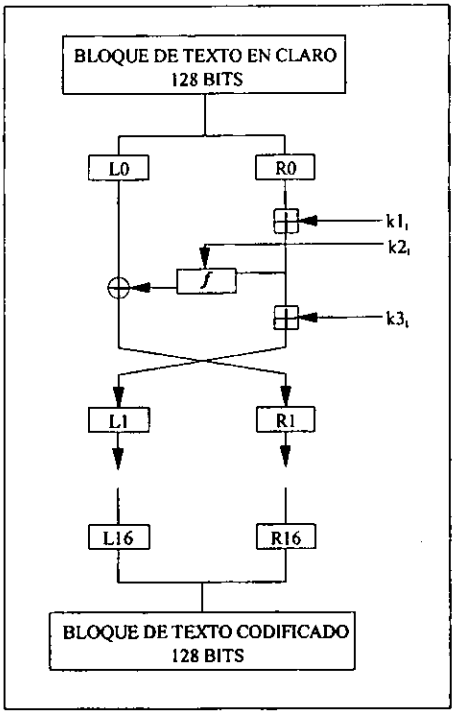


FIG. 2.19- DIAGRAMA DE BLOQUES DE LOKI97.

LA FUNCIÓN *f*

A diferencia de la función *f* del DES, la cual incluye un paso en el cual se aplica un OR exclusivo entre la subclave de 48 bits y la mitad derecha del bloque después de la función de permutación, en LOKI 97, este paso se lleva a cabo fuera de la función *f*.

Una vez que la mitad derecha del bloque ha sido sumada a la primera subclave de la iteración ($\mathbf{k1_i}$), el valor es introducido dentro de la función *f*. La segunda subclave de 64 bits de cada iteración ($\mathbf{k2_i}$) es fragmentada en dos mitades ($\mathbf{k2_i^1}$ y $\mathbf{k2_i^2}$) de 32 bits. Los bits de $\mathbf{k2_i^1}$ son utilizados dentro del primer paso de la función *f* el cual consiste en la substitución del bit menos significativo de cada byte de $\mathbf{R_i}$ por el más significativo de cada byte de $\mathbf{k2_i^1}$.

El resultado del paso anterior, se somete a una función de expansión la cual consiste en adicionar 3 ó 5 bits menos significativos (según sea el caso) procedentes del byte precedente, esto es, sí el resultado del paso anterior contenía 64 bits ó 8 bytes llamados A, B, C, D, E, F, G y H; al byte B se le adicionarán los 3 bits menos significativos de A, al byte D los 5 bits menos significativos de C y así sucesivamente hasta que cada byte aumente su tamaño de acuerdo a la tabla 2.11.

Número de bits antes de la expansión	8	8	8	8	8	8	8	8
Byte correspondiente	a	b	c	d	e	f	g	h
Número de bits después de la expansión	13	11	13	11	11	13	11	13

TABLA 2.11.- RELACIÓN DE NÚMERO DE BITS A EXPANDIR PARA LA FUNCIÓN *f*.

Después de la expansión, el resultado es introducido a una serie de cajas S. LOKI contiene dos distintas cajas S; la caja S1 contiene 8192 valores que se encuentran entre 0 y 255, mientras que S2 contiene 2048 valores que se encuentran entre el mismo intervalo que S1. Las cajas S son utilizadas en el siguiente orden:

S1, S2, S1, S2, S2, S1, S2, S1

A la salida de las cajas S se obtendrá como resultado un bloque de 64 bits. El resultado obtenido de la salida de las cajas S, es entonces introducido dentro de una función de permutación la cual está dada por la tabla 2.12.

7	15	23	31	39	47	55	63
5	13	21	29	37	45	53	61
3	11	20	27	35	43	51	59
1	9	18	25	33	41	49	57

6	14	22	30	38	46	54	62
4	12	20	28	36	44	52	60
2	10	18	26	34	42	50	58
0	8	16	24	32	40	48	56

TABLA 2.12.- PERMUTACIÓN DE BITS A LA SALIDA DE LA CAJA S.

El resultado de la permutación es expandido nuevamente utilizando los 32 bits restantes de la mitad $k2_i^2$ adicionando 3 ó 5 bits a cada byte según sea el caso, ésta vez en el siguiente orden:

Número de bits antes de la expansión	8	8	8	8	8	8	8	8
Byte correspondiente	a	b	c	d	e	f	g	h
Número de bits después de la expansión	11	13	11	13	13	11	13	11

TABLA 2.13.- RELACIÓN DE BITS EN LA SEGUNDA EXPANSIÓN DE LA FUNCIÓN *f*.

Finalmente, los 96 bits resultantes son nuevamente introducidos dentro de las cajas S, pero ésta vez seguirán el siguiente orden:

S2, S2, S1, S1, S2, S2, S1, S1

Formando de esta manera el bloque de 64 bits que será la salida de la función *f*. Una mejor explicación se del procedimiento completo se muestra en la figura 2.20.

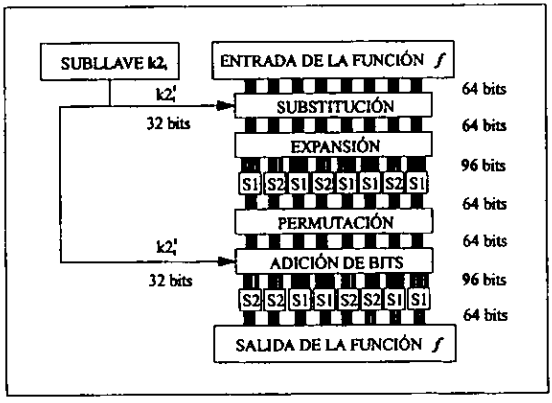


FIG. 2.20- DIAGRAMA DE BLOQUES DE LA FUNCIÓN *f* DE LOKI97.

GENERACIÓN DE SUBCLAVES.

Las subclaves son generadas a partir de una versión modificada de la función *f* del algoritmo. La entrada de la subclave se descompone en 4 subclaves de 64 bits llamadas **K1**, **K2**, **K3** y **K4**. Cuando se utiliza una clave de 256 bits, ésta es descompuesta en 4 bloques, el primer bloque se convierte en **K4**, el segundo en **K3** y así sucesivamente. Como en un principio se mencionó, este algoritmo tiene la flexibilidad de poder manejar distintos tamaños de clave, así en el caso de que la clave introducida sea de un tamaño de 192 ó 128 bits, el primer paso a realizar será su expansión hasta los 256 bits necesarios, acto seguido se fragmentará en los cuatro bloques ***K***, mencionados con anterioridad.

Para poder generar las subclaves se utiliza una función *f* cuya primer entrada se calcula a partir de la suma de **K1**, **K3** y una constante hexadecimal (9E3779B97F4A7C13) multiplicada por el número de iteración de generación de la subclave (1 a 48), el segundo valor introducido a la función *f* es el formado a partir de **K2**. **K4** es adicionada mediante un OR exclusivo al final de la función *f*, el resultado producido es la subclave a utilizar en el proceso.

Una vez que se ha calculado la primera subclave, se vuelve al ciclo iterativo pero ésta vez haciendo un ligero cambio, **K1** tomará el lugar de **K2**, **K2** el de **K3**, **K3** el de **K4** y **K4** el de **K1**.

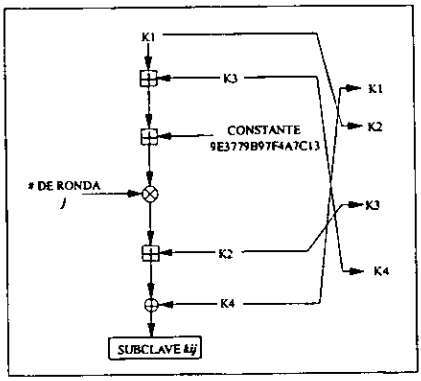


FIG. 2.21- DIAGRAMA DE BLOQUES DEL GENERADOR DE SUBCLAVES.

SEGURIDAD DEL ALGORITMO.

Debido a la reciente creación de este algoritmo, se puede decir que su criptoanálisis aún continua, por lo cual aún no se tienen resultados certeros sobre su nivel real de seguridad. El único precedente que se tiene es que en su versión '89 resistió el criptoanálisis diferencial, sin embargo, se desconoce la debilidad o el inconveniente que motivó la creación de ésta nueva versión.

II.13.- RIJNDAEL

Creado en 1997 por Joan Daemen y Vincent Rijmen en Bélgica, Rijndael[27][28][29] fue propuesto al NIST en 1998 para sustituir al actual DES.

Rijndael es un algoritmo que presenta cierta flexibilidad al permitirle al usuario la elección del tamaño de bloque de texto a codificar, ya sea de 16, 24 ó 32 bytes, esta posibilidad de elección repercute directamente en el tamaño de clave ya que debe de ser del mismo tamaño del bloque. El número de ciclos de codificación también varía en función del tamaño de bloque de texto, así tenemos que: para 16 bytes se utilizan 9 ciclos, para 24 bytes 11 ciclos y para 32 bytes 13 ciclos.

Cada ciclo iterativo se compone de cuatro pasos:

a) Substitución.
b) Intercambio de fila.
c) Intercambio de columna.
d) Adición de la clave.

Además de estos pasos, se tiene que tener en consideración el paso de generación de subclaves el cuál se lleva a cabo al inicio de la codificación.

A diferencia de los demás algoritmos simétricos que suelen ser del tipo Feistel, Rijndael no divide el bloque de texto en sub-bloques, en lugar de esto codifica el bloque de texto completo byte a byte de forma simultánea. El inicio de Rijndael consiste en el arreglo matricial del bloque de texto y de la clave, byte a byte de arriba abajo y de izquierda a derecha en un arreglo matricial de cuatro filas y ***n*** columnas (Fig. 2.22).

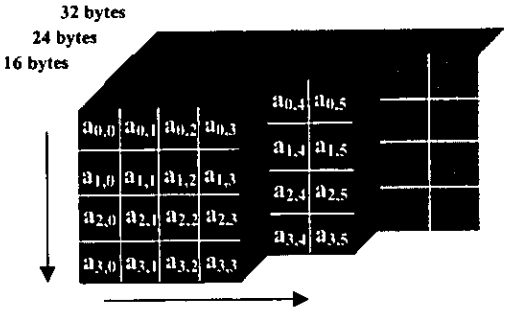


FIG. 2.22.- ARREGLO MATRICIAL DEL TEXTO EN CLARO Y DE LA CLAVE.

La explicación completa de cada paso del ciclo se da a continuación.

SUBSTITUCIÓN.

El primero de los cuatro pasos, consiste en remplazar cada uno de los bytes del texto en claro por su correspondiente byte dentro de una caja-S compuesta de 256 elementos.

Este paso garantiza que no exista una linealidad entre el bloque de texto y el codificado, lo cuál hace difícil la decodificación del mismo.

INTERCAMBIO DE FILA.

Dentro de este paso, como su nombre lo indica, consiste en intercambiar de fila los bytes dentro de la matriz, así se puede observar en las siguientes tablas los cambios para 16, 24 y 32 bytes, respectivamente:

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

TABLA 2.14.-INTERCAMBIO DE FILA PARA 16 BYTES.

4 8 12 16 20 24	1 5 9 13 17 21
2 6 10 14 18 22	6 10 14 18 22 2
3 7 11 15 19 23	11 15 19 23 3 7
4 8 12 16 20 24	16 20 24 4 8 12

TABLA 2.15.- INTERCAMBIO DE FILA PARA 24 BYTES.

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

TABLA 2.16.- INTERCAMBIO DE FILA PARA 32 BYTES.

Este paso asegura que exista una amplia difusión o mezcla de bytes a través de los diferentes ciclos del algoritmo.

INTERCAMBIO DE COLUMNA.

En este paso, cada columna de la matriz resultante del paso anterior se multiplica por la matriz de la figura 2.23, esto garantiza una alta difusión de las columnas durante la codificación.

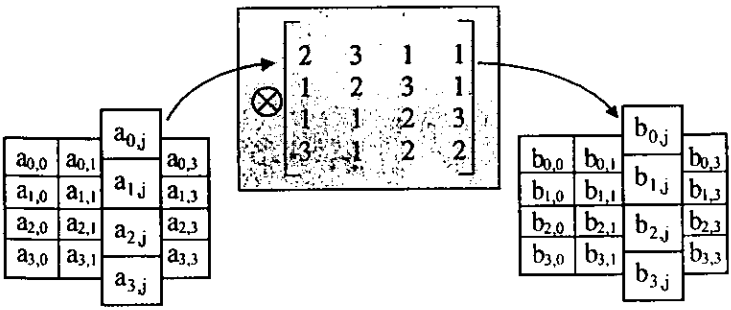


FIG. 2.23.- INTERCAMBIO DE COLUMNA DE RIJNDAEL.

ADICIÓN DE CLAVE.

El último paso, es la adición de la subclave correspondiente al número de iteración que se esté llevando a cabo en ese momento mediante un OR exclusivo.

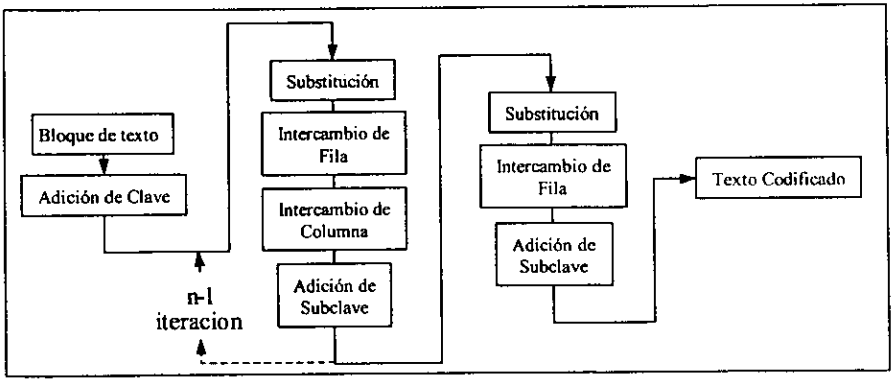


FIG. 2.24.- DIAGRAMA DE BLOQUES DE RIJNDAEL.

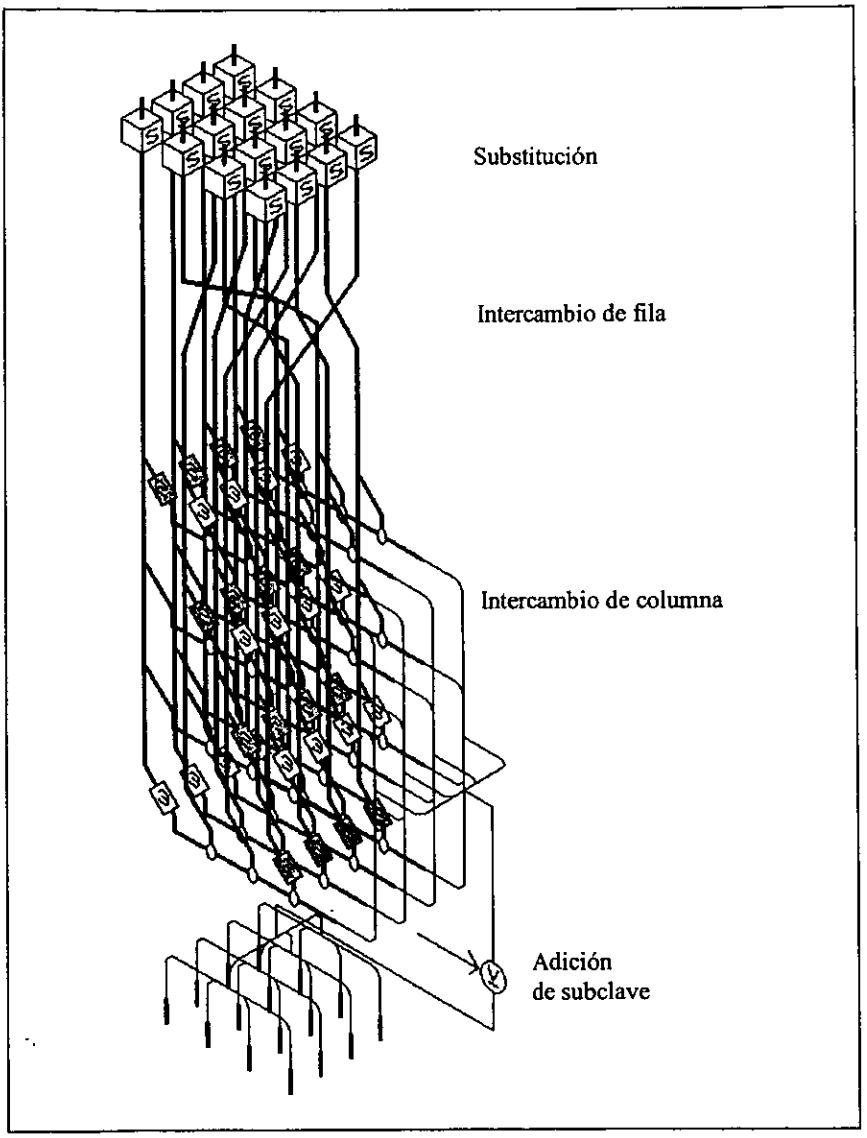


FIG. 2.25.- REPRESENTACIÓN TRIDIMENSIONAL DE UNA ITERACIÓN DE RIJNDAEL.

GENERACIÓN DE SUBCLAVES

La generación de subclaves de codificación K_i para i=0..n+2 y n=9, 11 o 14, se lleva a cabo partiendo de la clave original K, ordenando los bytes que la constituyen de la misma forma que el texto en claro (Fig. 2.22).

La clave original se subdivide en cuatro ***W0, W1, W2*** y ***W3*** (Tabla 2.17) y estas servirán para generar las siguientes subclaves.

$$K_0 = W0 + W1 + W2 + W3$$

$$K_1 = W4 + W5 + W6 + W7 \ldots.$$ donde:

$$W_{(i)} = W_{(i-4)} \text{ Xor } W_{(i-1)}$$

para i = múltiplo de 4

$$W_{(i)} = W_{(i-4)} \text{ Xor (Substitución en caja } S \text{ de } W_{(i-1)} \text{ XOR } (\mathbf{RC}(k),00,00,00)$$

Y donde $\mathbf{RC}(k) = i/4$

W0	*W1*	*W2*	*W3*
00	04	08	0C
01	05	09	0D
02	06	0A	0E
03	07	0B	0F

TABLA 2.17.- DIVISIÓN DE LA CLAVE ORIGINAL EN SUBLOQUES *W*.

SEGURIDAD DE RIJNDAEL

Rijndael a pesar de su relativa juventud, ha sido probado con éxito contra diversos ataques por lo cuál ha sido seleccionado para servir como base del nuevo estándar de codificación de EUA. Por tal motivo se puede decir que este algoritmo es muy seguro.

Las características que presenta este algoritmo con respecto a los otros que sobrevivieron hasta la última fase de la selección para el nuevo AES son: seguridad, rapidez en la iteración, flexibilidad y sencillez de programación.

II.14.-AES (Advanced Encryption Standard)

A mediados de la década de los 90's, el Instituto Nacional de Estándares y Tecnología de los E.U.A. (NIST) comenzó a observar que el Algoritmo Estándar de Codificación (DES) propuesto por el propio Instituto en 1977 comenzaba a volverse altamente inseguro, esto debido a que los avances tecnológicos permitían decodificar un mensaje codificado con este algoritmo probando todas las 72,057,594,037,927,776 posibles claves en un tiempo razonablemente corto[30][31][32].

En la primavera de 1997, el NIST lanzó la convocatoria para la creación de un nuevo sistema que permitiera sustituir al DES, el Instituto estableció una primer serie de requisitos que debían ser cubiertos por los concursantes, entre ellos estaban que el algoritmo seleccionado debería codificar bloques de texto y utilizara una longitud de clave variable de 128, 196 y 256 bits, para el verano del mismo año se decidió modificar dicho requisito debido a dos razones: la primera, es que entre más grande sea el tamaño de bloque de texto mayor será el tiempo en el que algún algoritmo lo pueda codificar y la segunda, es que ninguno de los participantes pudo cubrir dicho requisito, por tal motivo el NIST decidió restringir tanto el bloque de texto como el tamaño de clave a 128 bits. En agosto de 1998 se presentó a la opinión pública una lista consistente de quince algoritmos los cuales cubrían los requisitos de seguridad que el NIST planteaba, estos son:

NOMBRE	AUTORES	PAÍS	NÚMERO DE ITERACIONES
CAST-256	Carlisle Adams	E.U.A	48
CRYPTON	C.H.Lim		12
DEAL	Knudsen, Outerbridge	E.U.A.	6,8
DFC	Vaudenay et al	Francia	8
E2	Aoki, Kanda, Matsumoto, Moriai, Ohta, Ookubo, Takashima, Ueda	Japón	12

NOMBRE	AUTORES	PAÍS	NÚMERO DE ITERACIONES
FROG	Georgoudis, Leroux, Chaves	España	8
Hasty Pudding	R. Schroeppel	E.U.A.	8
LOKI97	Brown, Pieprzyk	Australia	16
MARS	IBM	E.U.A.	32
Magenta	Deutsche Telekomm	Alemania	6,8
RC6	Rivest, Robshaw, Sidney, Yin	E.U.A.	20
RIJNDAEL	Rijmen, Daemen	Bélgica	10,12,14
SAFER+	Massey, Khachatrian, Kuregian	E.U.A.	8,12,16
SERPENT	Anderson, Biham, Knudsen	Isrrael	32
TWOFISH	Schneier, Kelsey, Whiting, Wagner, Hall, Ferguson	Inglaterra	16

En agosto de 1999, se anunció a los cinco finalistas, estos fueron: MARS, RC6, RIJNDALE, SERPENT y TWOFISH. En Abril del 2000, durante la tercera conferencia de candidatos para el AES en Nueva York, se reunieron diversos grupos con la finalidad de presentar resultados sobre la seguridad de estos algoritmos, de ahí se obtuvieron resultados decisivos para la selección del algoritmo ganador. Aquí se presentaron una serie de ataques a versiones incompletas de todos estos algoritmos. El 2 de octubre del 2000, el NIST estableció que el algoritmo denominado RIJNDALE era el seleccionado para servir como base del nuevo estándar en seguridad. A partir de esa fecha el NIST establecerá los protocolos estándar de uso, así como una versión modificada de RIJNDALE el cual será el nuevo AES.

En el siguiente capítulo se describe el funcionamiento de los algoritmos asimétricos así mismo se hacen algunas referencias sobre la seguridad de los mismos.

Ser hombre es saber decir "me equivoqué"
y proponerse no repetir el mismo error.
-Roger Patrón

CAPÍTULO III: ALGORITMOS DE CODIFICACIÓN ASIMÉTRICA

III.1.- INTRODUCCIÓN

En 1975, dos ingenieros electrónicos de la Universidad de Stanford, Whitfield Diffie y Martin Hellman[33][34][35], sugieren la utilización de problemas computacionalmente irresolubles para el diseño de criptosistemas seguros. La idea consiste básicamente en encontrar un sistema de codificación, cuya implementación sea fácil desde el punto de vista computacional, el cual deberá contar con las siguientes características:

1. Cualquier usuario debe ser capaz de calcular sus propias claves (privada y pública).
2. El usuario emisor puede codificar su mensaje con la clave pública del receptor.
3. El usuario receptor puede decodificar el mensaje codificado con su clave privada.
4. El criptoanalista que intente averiguar la clave privada mediante la clave pública se encontrará con un problema matemáticamente intratable.
5. El criptoanalista que intente decodificar un mensaje codificado teniendo la clave pública se encontrará con un problema intratable.

Como se puede observar, para este tipo de algoritmos no es necesaria la utilización de un canal seguro para la transmisión de la clave pública, incluso, esta puede ser transferida junto con un mensaje codificado o bien, esta puede ser colocada en un directorio parecido a la

Sección Amarilla. Debido a éstas características, este tipo de tecnologías criptográficas permiten el intercambio de mensajes codificados entre 2 usuarios que nunca se han visto o conocido en persona, además de que no es necesario tener una clave por pareja de usuarios.

III.2.- RSA

En 1977, tres investigadores del MIT, Ron Rivest, Adi Shamir y Len Adleman[36][37][38] desarrollan un algoritmo para la codificación de datos al cual denominan como, RSA por las siglas de sus apellidos.

El algoritmo que ellos propusieron es el siguiente:

1. Se eligen 2 números primos p y q.
2. Se calcula $n=p \cdot q$
3. Se selecciona un número d el cual no tendrá divisores en común con $(p-1)\cdot(q-1)$ diferente de 1, se recomienda que d sea mayor que p y q.
4. Se calcula un entero, e tal que $1 \leq e \leq (p-1)\cdot(q-1)$ mediante la siguiente fórmula:

$$e \cdot d \equiv 1 (\text{mod } [(p-1)\cdot(q-1)])$$

Los usuarios publicarán entonces la pareja de (e,n) la cual constituirá su clave pública, mientras que d deberá permanecer en secreto y junto con n formarán la clave privada de cada usuario. También, deberán permanecer en secreto p y q que se utilizarán sólo para la obtención de claves y no volverán a ser utilizados durante el funcionamiento del sistema.

Antes de codificar un mensaje utilizando este algoritmo, es necesario hacer que cada uno de los caracteres del mensaje original tome un valor numérico, tal como el proporcionado por el código ASCII y después dividir la secuencia de números resultantes en bloques, de modo, tal que n sea distinto del valor máximo de cada bloque, por ejemplo, si cada bloque está compuesto por un caracter ASCII, n deberá ser mayor a 256, si está compuesto por 2, deberá ser distinto a 256^2 y así sucesivamente. Cuando el valor de n es igual al tamaño del bloque de

texto, se puede dar el caso de que la decodificación del texto no sea único por tal razón esto debe evitarse.

Cada bloque se codificará transformándolo en un nuevo bloque numérico de acuerdo a:

$$C_i = M_i^{\ e} \pmod{n}$$

Donde M_i es cada uno de los caracteres del texto normal y C_i es el caracter ya codificado.

Para decodificar un mensaje, se utilizará la clave privada (d,n) en la siguiente ecuación:

$$M_i = C_i^{\ e} \pmod{n}$$

La seguridad del algoritmo dependerá en gran medida del tamaño de la clave que se elija, por ésta razón se recomienda que se seleccionen claves de más de 100 dígitos.

ATAQUES A RSA

Factorización

Para el uso de RSA, es necesario conocer tanto los códigos de codificación (la pareja de e y n) como los de decodificación (d y n), pero dado que e y n son públicos ahora solo queda el problema de encontrar d, y para realizar esto, es necesario factorizar n ya que así obtendremos los valores de p y q, y con ellos d.

Para resolver el problema de la factorización de n, existen en la actualidad diversos algoritmos tales como el de fracciones continuas de *Morrison - Brillart*, que es uno de los más rápidos. Sin embargo, los algoritmos que resultan más rápidos para la factorización de enteros grandes son el de factorización con curvas elípticas de *Hendrik Lestra* y al de factorización con filtro cuadrático de *Carl Pomerance*. Ambos algoritmos, convierten el problema de la factorización de un entero en el problema de encontrar soluciones no triviales de la acuación $x^n \equiv y^n \pmod{n}$. Si se supone que ni $(x + y)$ son múltiplos de n enteros, se deduce que el máximo

común divisor (m.c.d.) (x+y,n) o el m.c.d. (x-y,n) es con seguridad un factor no trivial de n, por lo que se resuelve el problema de la factorización.

Por ejemplo: sí n= 97343, entonces la ecuación $x^2 \equiv y^2$ (mod 97343) es fácilmente resoluble por ser los factores de n dos números enteros primos muy cercanos, como: $312^2 = 1$(mod 97343) y ni 313 ni 311 son múltiplos de 97343, se concluye que 313 y 311 son los factores de n.

Sin embargo, aún con los actuales métodos para factorizar grandes números primos, este tipo de ataque es difícil, ya que depende mucho del tamaño de clave utilizada y de la velocidad del hardware utilizado, así por ejemplo, en 1977, Ron Rivest estimaba que se necesitarían 40 cuadrillones de años para factorizar una clave de 125 dígitos, en 1994, una clave de 125 dígitos fue factorizada en 8 meses utilizando varias computadoras a través de Internet, en 1995 la llave Blacknet de 116 dígitos fue factorizada en 3 meses utilizando varias Workstations.

Ataque de Blakley y Borosh

El sistema RSA tiene una característica muy peculiar advertida por Blakley y Borosh, y que no siempre esconde el mensaje.

Si tenemos que e=17, n=35 y los mensajes a codificar son M_1= 6 y M_2=7, entonces se obtiene que $6^{17} \equiv 6$ (mod 35) y $7^{17} \equiv 7$ (mod 35). Una situación más peligrosa para el sistema se observa con este ejemplo, cuando los valores de p=97, q=109 y e=865, ya que el criptosistema resultante no esconde ningún mensaje, pues $M^{865} \equiv M$ (mod 97·109).

Lo anterior, ocurre siempre que e-1 es múltiplo de p-1 y q-1, pues en ese caso $M^e \equiv M$(mod p·q). Además, se tiene que para cualquier elección de n=p·q siempre existen al menos 9 mensajes M que no se codifican en realidad, ya que verifican la ecuación $M^e \equiv M$(mod n).

Ataque a texto codificado

Supongamos que tenemos a un usuario interceptando las transmisiones de archivos codificados hechas a través de un canal inseguro, este usuario colecta los mensajes **C** y conoce la clave pública del emisor (*e*,*n*), y con estos datos pretende decodificar el mensaje, por lo que primero selecciona un número aleatorio $r<n$, y con este realiza las siguientes operaciones:

$x=r^e \bmod n$ (Codifica *r* con la llave pública del "usuario objetivo").

$y=xc \bmod n$ (Multiplica el texto codificado del "objetivo" con el temporal *x*).

$t=r^{-1} \bmod n$ (El inverso multiplicativo de *r* mod *n*).

El usuario escucha, tiene en cuenta que:

Si $x=r^e \bmod n$, entonces $r=x^d \bmod n$

El escucha hará que el usuario objetivo firme y con su clave privada la cual realmente decodifica *y* y envía $u=y^d \bmod n$ al usuario escucha, el cual no le queda más que calcular:

$t \cdot u \bmod n = (r^{-1})(y^d) \bmod n = (r^{-1})(x^d)(c^d) \bmod n = (c^d) \bmod n$

COMENTARIOS

La seguridad de este algoritmo se basa principalmente en el tamaño de la clave que el usuario seleccione, y este tamaño de clave lo deberá seleccionar en función del nivel de seguridad que le desee dar a la información que transmite, tomando en cuenta el tipo de posibles usuarios que pudieran desear ésta información y los recursos con los cuales estos contarán. Por lo cual, se recomienda que la clave seleccionada sea mayor a 1024 bits, ya que

será bastante difícil factorizar un número tan grande en un tiempo razonablemente corto mediante los algoritmos actuales, o bien, cambiar frecuentemente de clave.

Para evitar el ataque de Blakley y Borosh, es conveniente elegir como claves privadas números primos de la forma $p= 2p'+1$ donde p' es un primo impar.

Para poder evitar el ataque al texto codificado, se recomienda no firmar cualquier documento, en tal caso, se recomienda firmar un mensaje producido por un algoritmo Hash unidireccional, tal como el MD4 o MD5.

III.3.- SISTEMA DE MERKLE-HELLMAN

En 1978, Merkle y Hellman[39] desarrollaron un sistema de clave pública basado en el problema de Knapsack, también conocido como el problema de la Mochila Tramposa, el cual se menciona a continuación: "*Dada una mochila de volumen N y un número de objetos de volúmenes e_i= 1,...,n que se desean introducir dentro de la mochila, el objetivo consiste en llenar completamente la mochila encontrando un subconjunto $I \subseteq \{1,....,n\}$ tal que* $\sum_{i \in I} e_i = N$ *si tal conjunto existe*".

A continuación se presenta una descripción del algoritmo de este sistema de codificación, considerando en primera instancia que el mensaje a codificar ha sido transformado a binario y se ha dividido en bloques de tamaño n, cada bloque se denota como M.

1. El emisor escoge una serie de números en orden creciente $\{e_1,...,e_n\}$, un entero N mayor que $\sum_{i=1}^{n} e_i$ y un entero grande w primo tal que $0<w<N$.

2. El emisor calcula w^{-1} (mod N), y el conjunto $\{a_i\}$ de n elementos definida mediante $a_i \equiv w \cdot e_i$ (mod N). De los cuales mantiene e_i, N, w y w^{-1} en secreto y publica el conjunto de a_i. La clave de codificación del emisor es $K_A=\{a_i,...,a_n\}$, mientras que, la de decodificación es $k_A=(w,N)$.

3. Todo aquel que quiera enviar un bloque de mensaje $M=(M_1, M_2,....,M_n)$ al emisor codificándolo, tiene que calcular $C = \sum_{i=1}^{n} M_i a_i$ y transmitir este entero al receptor.

4. Para decodificar el mensaje, el emisor tiene que calcular el menor residuo positivo de w^{-1} C(mod N) que, como $w^{-1}C \equiv \sum_{i=1}^{n} M_i w^{-1} a_i \equiv \sum_{i=1}^{n} M_i e_i (\text{mod } N)$, corresponde a $V = \sum_{i=1}^{n} M_i e_i$. Entonces revisará uno a uno los valores de e_i comenzando con el mayor hasta encontrar el primero que sea mayor o igual a N. Este e_{i0} se incluye en el subconjunto solución **I** haciendo que el valor e_{i0}=1. Se reemplaza N por N- e_{i0} y se continúa la búsqueda en descenso desde e_{i0} hasta encontrar el mayor valor que sea menor o igual a N- e_{i0}. Continuando con este proceso, iterativamante se obtiene un subconjunto de $\{ e_i\}$ que suma N.

Cabe mencionar que una simplificación de este algoritmo supone que todos los números que constituyen la clave privada sean elegidos en estricto orden ascendente, a este caso se le conoce con el nombre de *mochila fácil.*

Este algoritmo presenta la ventaja de ser muy rápida su codificación y decodificación en comparación con otros algoritmos asimétricos, como muestra de esto, se sabe que es 100 veces más rápido que el sistema RSA.

EJEMPLO DE CODIFICACIÓN CON MERKLE-HELLMAN

Se consideran:

clave secreta (2, 3, 7, 15, 31)

N=61

w=17

$w^{-1}=18\equiv 17^{-1}(\text{mod } 61)$

Calculando:

$a_1=17\cdot 2=34(\text{mod } 61)$

$a_2=17\cdot 3=51(\text{mod } 61)$

$a_3=17\cdot 7=58(\text{mod } 61)$

$a_4=17\cdot 15=11(\text{mod } 61)$

$a_5=17\cdot 31=39(\text{mod } 61)$

Tenemos:

como clave pública (34, 51, 58, 11, 39)

Para codificar el mensaje "HAY" enumeramos del 1 al 26 las letras del abecedario y su número correspondiente lo transformamos a binario:

H=8 $\Rightarrow$ (01000) $\rightarrow$ $0\cdot 34+1\cdot 51+0\cdot 58+0\cdot 11+0\cdot 39=51$

A=1 $\Rightarrow$ (00001) $\rightarrow$ $0\cdot 34+051+0\cdot 58+0\cdot 11+1\cdot 39=39$

Y=25 $\Rightarrow$ (11001) $\rightarrow$ $1\cdot 34+1\cdot 51+0\cdot 58+0\cdot 11+1\cdot 39=124$

Para decodificar el mensaje (51, 39, 124) primero se multiplica el mensaje por 18 en módulo 61, obteniéndose (3, 31, 36), y luego se resuelve el problema de la mochila fácil con la clave secreta (2, 3, 7, 15, 31) para cada uno de los 3 casos, recuperándose así el mensaje: (01000), (00001) y (11001).

SEGURIDAD DEL ALGORITMO

En 1982, Shamir diseñó un algoritmo capaz de romper este sistema. Este ataque procede de analizar los números que forman la clave de codificación para intentar encontrar un par de números enteros N y Z tales que el conjunto $\{Z \cdot a_i \pmod{N}\}$ constituya una mochila fácil, cuya suma sea menor que N. Para ello, es necesario suponer que al orden creciente de los coeficientes a_i corresponde al orden creciente de los coeficientes de la mochila fácil original, y hay que calcular una serie de intervalos cada vez más pequeños, donde se sabe que están cada uno de los números a_i.

Este procedimiento es válido siempre que la longitud de los números de la mochila fácil no sea mayor de 100 bits, pues en ese caso la consecución de los intervalos y su posterior reducción se hace infactible.

En 1983, Adelman, Lagarias y Odlyzko crearon algoritmos de ataque destinados de forma específica a decodificar mensajes que hubiesen sido codificados con este algoritmo, con lo cual pusieron en evidencia sus puntos débiles.

En la actualidad a pesar de no considerarse este sistema como muy seguro, existen investigaciones para crear nuevos y más resistentes algoritmos basados en el problema de la mochila, esto se debe principalmente a que las velocidades de codificación y decodificación son muy altas en comparación con otras opciones.

III.4.- SISTEMA DE McELICE

McElice[39] introdujo en 1978 un criptosistema de clave pública basado en la teoría de codificación algebraica, utilizando, entre otros, el concepto del código de Goppa el cual se emplea como un código corrector de errores altamente eficiente.

El algoritmo se describe de la siguiente forma:

1. El receptor parte de un código de Goppa fácilmente resoluble con una matriz generadora de orden k x n, denotada por G, la cual cuenta con una capacidad de corrección de errores t. Esta matriz se calcula a partir de una matriz de chequeo de paridad $H = \left(\frac{A}{I_d}\right)$, según la relación $G = \left(\frac{I_d}{-A^T}\right)$.

2. La matriz G se transforma en G' mediante la expresión $G'=S \cdot G \cdot P$, donde S es una matriz aleatoria inversible de orden k x k, y P es una matriz de permutaciones de orden n x n. La matriz G', es una matriz generadora para un código lineal tal que no existe un algoritmo suficientemente rápido como para corregir los errores con este código. Este código tiene la misma información y mínima distancia que el código generado por G.

3. G' es la clave pública de este sistema. El emisor puede codificar un mensaje M de k bits mediante la operación $C=M \cdot G' + e$, donde e es un vector corrector de errores de n bits de peso t escogido por el remitente y "+" es la operación suma módulo 2.

4. Para la decodificación del mensaje, el receptor conoce que $C=M \cdot G'+e=M \cdot S \cdot G \cdot P+e$, por lo cual, calcula $C \cdot P^{-1}=(M \cdot S)G+e \cdot P^{-1}$ y utiliza el algoritmo de decodificación para el código original que elimina el vector de errores $e \cdot P^{-1}$ y recupera así el vector $M \cdot S$. El mensaje del remitente se encuentra fácilmente mediante $M=(M \cdot S)S^{-1}$.

SEGURIDAD DEL ALGORITMO

La seguridad de este algoritmo radica en la dificultad que implica el determinar el vector de errores e, a partir del producto de las matrices $G'=S \cdot G \cdot P$. Sin embargo, se han desarrollado algunas propuestas para poder resolver ese pequeño inconveniente.

En 1991, Korzhik propuso en el Congreso Eurocript celebrado ese año, un ataque basado en un algoritmo iterativo de optimización. Este algoritmo garantiza la corrección de errores con peso máximo (d-1)/2 para un código lineal arbitrario con distancia mínima d. El criptoanalista puede usar este algoritmo sobre el criptograma C para obtener la palabra código $M \cdot G'$, de donde simplemente recupera el mensaje M invirtiendo k columnas cualesquiera de G' que contenga una matriz no singular.

Cabe mencionar que aunque este sistema no ha sido totalmente acogido, continúa investigándose el posible uso de la teoría de la codificación para futuros desarrollos de criptosistemas de clave pública.

III.5.- EL SISTEMA DE RABIN

En 1979, Rabin[40] propuso un sistema de codificación asimétrica de clave pública. Este sistema consiste de dos números primos grandes p y q (100 o más dígitos), tales que, $p \equiv q \equiv 3 \pmod 4$, así como su producto $N = p \cdot q$. El número N es la clave pública de este sistema, mientras que p y q constituyen la clave secreta. En estas condiciones, la codificación de un mensaje original M se lleva a cabo mediante la siguiente congruencia:

$$C \equiv M^2 \pmod N$$

Donde C, es el mensaje codificado. En estas condiciones, para recuperar el mensaje original M hay que calcular la raíz cuadrada del mensaje codificado C (mod N), lo cual se presenta mediante la siguiente ecuación:

$$M \equiv \sqrt{C} \pmod N$$

El principal inconveniente de este sistema de codificación, reside en que el valor resultante de esta raíz cuadrada no es único. Esto se debe a que la función utilizada para

realizar la codificación (cuadrado modular) no es inyectiva. Así, entonces tenemos que todo entero relativamente primo con N, tiene 4 posibles raíces cuadradas (mod N). Por tanto, en el momento que se desee decodificar C, se deberá seleccionar una de estas 4 posibilidades, como posible valor del mensaje M. Consecuentemente, la recuperación del mensaje M requiere la aplicación de un procedimiento de raíces cuadradas (mod N). Para el caso que nos ocupa, es decir, cuando se trabaja con valores de N demasiado grandes, los posibles algoritmos aplicables son sólo viables en el caso de conocer p y q, y aún así, son de una complejidad computacional elevada. Sin embargo, si el valor de los factores primos de N se desconocen, el cálculo de las raíces cuadradas se convierte en un problema intratable.

Para cuando el problema se convierte en un caso particular, esto es, cuando se conocen p, q y N, existe un algoritmo eficiente para calcular las raíces cuadradas (mod N):

$$\sqrt{C} \equiv \left\{ \pm A_p \left[C_p^{d_p} (\text{mod } p) \right] \pm A_q \left[C_q^{d_q} (\text{mod } q) \right] \right\} (\text{mod } N)$$

donde los valores de las variables vienen dadas por:

$$C_p \equiv C (\text{mod } p)$$

$$C_q \equiv C (\text{mod } q)$$

$$A_p \equiv q \left[q^{-1} (\text{mod } p) \right] \equiv q^{p-1} (\text{mod } N)$$

$$A_q \equiv p \left[p^{-1} (\text{mod } q) \right] \equiv p^{q-1} (\text{mod } N)$$

$$d_p = \frac{p+1}{4}$$

$$d_q = \frac{q+1}{4}$$

EJEMPLO DE LA CODIFICACIÓN CON EL SISTEMA DE RABIN

Para poder ilustrar un ejemplo de la aplicación de este sistema de codificación, asignaremos los siguientes valores a las variables p, q, N y M.

$p = 67$

$q = 59$

$N = 67 \cdot 59 = 3953$

$M = 247$

Para codificar M, tenemos que:

$$C \equiv M^2 \pmod{N} \quad \therefore \quad C \equiv 247^2 \equiv 1714 \pmod{3953}$$

Para recuperar el mensaje original M, el receptor debe calcular las raíces cuadradas de la codificación (mod 3953). De esta manera, el receptor calcula las siguientes congruencias:

$$\sqrt{C} \equiv \sqrt{1714} \equiv \sqrt{39} \equiv \pm 39^{17} \equiv \pm 21 \equiv \{21,46\} \pmod{67}$$

$$\sqrt{C} \equiv \sqrt{1714} \equiv \sqrt{3} \equiv \pm 3^{15} \equiv \pm 48 \equiv \{48,11\} \pmod{59}$$

$$\sqrt{C} \equiv \sqrt{1714} \equiv \pm 21 \cdot 59\left[59^{-1} \pmod{67}\right] \pm 48 \cdot 67\left[67^{-1} \pmod{59}\right] \equiv$$

$$[(247,3706),(1051,2902)] \pmod{3953}$$

Evidentemente una de las 4 raíces recuperadas corresponde al mensaje original. Sin embargo, el sistema de codificación de Rabin no da suficiente información al receptor para elegir cual de las 4 raíces cuadradas recuperadas corresponde con el mensaje original que el emisor codificó.

SEGURIDAD DEL ALGORITMO

Como se puede observar, este algoritmo es bastante seguro y que en caso de no conocerse los valores de p, q y N, la decodificación de un mensaje se vuelve prácticamente imposible.

El inconveniente que presenta este algoritmo radica en que incluso conociendo los valores de p, q y N, existen 4 raíces elegibles, de las cuales, el usuario debe seleccionar una. El problema parece sencillo cuando el mensaje está escrito en un idioma común tanto para el emisor como para el receptor, pero se convierte en complejo cuando el idioma no es el mismo. Debido a lo anterior, el uso de este sistema no es muy popular, sin embargo, ha servido de base para el desarrollo de nuevos y mejores criptosistemas de clave pública.

III.6.- SISTEMA DE WILLIAMS

En 1980, H. C. Williams[40] modificó el sistema de Rabin con el fin de eliminar el inconveniente de las múltiples raíces (mod N). Los parámetros del nuevo sistema son p y q ($\cong$100 dígitos), tales que, $p \equiv q \equiv 3 \pmod 4$, así como $N=p \cdot q$, además de las variables ya conocidas, tenemos un parámetro adicional S. Los enteros N y S son la clave pública, mientras que la clave secreta es "d", la cual es función de p y q, que también son secretos. Así tenemos que d se calcula de la siguiente manera:

$$d = \frac{\phi(N)+4}{8} = \frac{(p-1)(q-1)+4}{8}$$

En estas condiciones, la codificación mediante el procedimiento de Williams de un mensaje original M tal que $1<M<N$ es el triplete (C, b_1, b_2), donde:

$$(M / N) = (-1)^{b_1} \text{ con } b_1 \in \{0,1\}$$

$$M' \equiv MS^{b_1} \pmod N$$

$$C \equiv M'^2 \pmod N$$

$$b_2 \equiv M' \pmod 2$$

La recuperación del mensaje original M a partir del mensaje codificado (C, b_1, b_2) y de la clave secreta d se lleva a cabo calculando:

$$M'' \equiv C^d \pmod{N}$$

$$b_3 = b_2 \oplus \left[M''(\bmod 2)\right]$$

$$M \equiv \frac{(-1)^{b_3} \cdot M''}{S^{b_1}} \pmod{N}$$

Como se puede observar, son precisamente los bits (b_1, b_2) los que permiten decidir sin ambigüedad cual de las 4 posibles raices cuadradas del mensaje codificado C constituyen el mensaje original M. Este sistema de codificación de clave pública fue posteriormente redefinido por el propio Williams. En su nueva versión, la codificación del mensaje se realiza elevando M' a la tercera potencia, en lugar de al cuadrado, de igual manera p y q de N deben ser tales que, $p \equiv q \equiv 1 \pmod 3$, lo cual evita una posible coincidencia de las claves de codificación y decodificación.

SEGURIDAD DEL ALGORITMO

Williams demostró que la seguridad de este algoritmo es prácticamente la misma que la del sistema RSA, ya que radica en la dificultad que supone la factorización de la clave pública N. Sin embargo, también se demostró que dicho algoritmo era completamente inseguro ante los ataques basados en texto codificado escogido.

III.7.- EL GAMAL

En 1985, T. Elgamal[41] propuso un esquema de codificación con clave pública basado en la exponenciación discreta sobre un grupo multiplicativo de un cuerpo finito Z_p, siendo p un número primo. El algoritmo que desarrolló es el siguiente:

GESTIÓN DE CLAVES

- Se selecciona un número primo p suficientemente grande (entre 200 y 600 dígitos).
- Se selecciona un número entero g, tal que, $1<g<p$.
- Se selecciona un número aleatorio a, tal que, $1<a<p-1$, el cuál será la clave secreta del usuario.
- Se calcula la clave pública del usuario con: $y = g^x \pmod p$.

CODIFICACIÓN DE UN MENSAJE M

- Se selecciona un número primo k tal que $1<k<p-1$ que será la clave de sesión.
- Se calcula: $r \equiv g^k \pmod p$.
- Se calcula: $s \equiv M y^k \pmod p$.
- Se envía el mensaje codificado (r, s).
- Para decodificar el mensaje, hay que calcular M con la resolución de la siguiente congruencia.

$$M \equiv \frac{s}{r^x} \pmod p$$

SEGURIDAD

Es obvio que para este algoritmo, el ataque por fuerza bruta es impracticable ya que habría de probar cerca de 2^{200} a 2^{600} claves distintas. Por otra parte, se ha hecho un estudio criptográfico a este algoritmo y se ha determinado que el ataque más factible sería el cálculo de la clave secreta a partir de los datos conocidos de (p, y) a partir de la siguiente ecuación:

$$a \equiv \log_g y \pmod p$$

Lo cual convierte la solución en un problema de algoritmo discreto y cuyo algoritmo de solución es aún más complejo que la solución a partir de fuerza bruta.

COMENTARIOS

El principal problema de la implementación de este criptosistema es que el mensaje codificado es del doble de largo que el mensaje original, lo cual duplica el tiempo de transferencia de la información.

En el siguiente capítulo, se analiza la interacción de los diferentes métodos de codificación de datos para la generación de aplicaciones que nos ofrecen ventajas de uno u otro método, dependiendo del servicio que se busque.

Nuestro defecto es aprender más por la escuela que por la vida.

-Séneca

CAPÍTULO IV: APLICACIONES

IV.1.- INTRODUCCIÓN

Los diversos métodos expuestos en los capítulos anteriores, son utilizados sin la necesidad de una herramienta o aplicación paralela, sin embargo, en los últimos años, han surgido algunos programas que integran diversas características, tales como mezclar algoritmos simétricos y asimétricos, o bien incorporar algunas herramientas que puedan hacer más eficiente y rápida la transferencia de un mensaje codificado.

Por otra parte, debido al creciente comercio electrónico, se han buscado crear métodos de certificación que permitan corroborar la identidad del cliente o del usuario al otro extremo de la red, los cuales no son propiamente sistemas de codificación, sin embargo nos permiten tener cierto nivel de seguridad en la transferencia de información.

Debido a lo anterior, se ha decidido clasificar a estos sistemas o herramientas auxiliares como aplicaciones de los sistemas criptográficos básicos existentes.

IV.2.- FIRMAS DIGITALES

Una firma se entiende como un símbolo escrito que cada persona diseña con la finalidad de autentificar su identidad en cualquier tipo de documento impreso, para la cual, según el grado de dificultad con que sea diseñada, será la facilidad con la que un tercero pueda duplicarla o imitarla.

A pesar de la relativa facilidad y bajo costo que implica la creación, la autentificación y el uso de una firma manual, ésta se vuelve inútil como medio de identificación en todo tipo de documentos electrónicos, ya que hoy en día, muchas transacciones comerciales se llevan a cabo diariamente a través del internet, llevando con esto, a influir en varios sectores de negocios, aunado a esto, cada día más compañías han lanzado nuevos servicios para explotar las oportunidades del comercio digital.

Por tales razones, cada día más negocios y clientes cuyas transacciones son llevadas a cabo a través de internet buscan un medio que les proporcione ciertas características de seguridad y confianza al momento de hacer pedidos y transmitir sus claves bancarias con el riesgo que esto implica de una posible intercepción o modificación por parte de un tercero.

En busca de esto, se han creado las denominadas "Firmas Digitales" [42][43], las cuales tienen la finalidad de confirmar la identidad de la persona que envía el mensaje, así como la autenticidad y proteger la integridad de los documentos electrónicos.

Una firma digital se clasifica como:

- IMPLÍCITA, sí está contenida en el propio mensaje.
- EXPLÍCITA, sí son añadidas como una marca inseparable del mensaje.
- PRIVADAS, sí permiten identificar al remitente ante cualquier persona a partir de información públicamente disponible.

- REVOCABLE, sí el remitente puede, posteriormente, negar que la firma en cuestión le pertenece.
- IRREVOCABLE, sí el receptor puede probar que el remitente escribió el mensaje.

En comparación a las firmas escritas, las firmas digitales no solo identifican al remitente de un mensaje electrónico, también puede asegurar la autenticidad del mismo (parte crítica del comercio digital). Por estas razones, todo diseño de algoritmo de las Firmas Digitales debe contar con las siguientes características:

- Ser única, pudiéndola generar solamente el usuario legítimo.
- Infalsificable, el intento de falsificación debe llevar asociada la resolución de un problema numérico intratable.
- Fácil de autentificar, pudiendo cualquier receptor establecer su autenticidad aún después de mucho tiempo.
- Irrevocable, el autor de una firma no puede negar su autoría.
- Barata y fácil de generar.

Las firmas digitales usan el típico sistema de la llave pública, ambos son conceptos que proveen el mismo nivel de seguridad y que no necesitan de un canal seguro para distribuir una contraseña a otro usuario. Por tal razón son ideales para comunicaciones en red abierta.

El mecanismo de firma digital soporta los servicios de integridad de datos, autentificación de datos y no repudio con prueba de origen, esto es, garantiza que los datos recibidos por el receptor de una transmisión sean los mismos que los enviados por el emisor, autentifica el origen de la transmisión y es capaz de probar esto. Para el caso del servicio de no repudio con prueba de entrega, es necesario forzar al receptor enviar un recibo firmado digitalmente al emisor.

En el sistema de codificación de llave pública, cada usuario genera un par de llaves conocidas como la ***llave pública*** y la ***llave privada***. Sólo la llave pública puede decodificar un mensaje codificado por medio de su respectiva llave privada.

En cuanto a la longitud de la clave que se debe de seleccionar para que una firma digital sea segura, se recomienda que sea de 768 bits para actividades personales, 1024 para Corporaciones y 2048 para actividades de alto riesgo.

IV.2.1.- FIRMA DIGITAL DEL CRIPTOSISTEMA RSA

Para enviar la firma digital[44] de un mensaje determinado utilizando el criptosistema RSA, el emisor utilizará su clave pública y su clave privada de la siguiente manera:

1. Calcula su rúbrica codificando el mensaje mediante su clave privada utilizando el codificador RSA.

2. Determina su firma digital codificando por medio de RSA y la clave pública del receptor la rúbrica anterior.

El mensaje firmado que el emisor envía es la pareja formada por el criptograma correspondiente al mensaje completo y la firma digital. Es claro que el remitente es el único que puede determinar la rúbrica anterior, porque es el único que conoce su respectiva clave privada.

Para que el destinatario del mensaje pueda verificar que la firma corresponde al supuesto emisor, sólo tiene que comprobar que:

1.- El mensaje contenido en la firma digital sea abierto por medio de su clave privada.

2.-La rúbrica hecha por el emisor sea abierta por medio de la clave pública del mismo, mostrando al final de este paso, el mensaje completo.

Al momento de llevar a cabo la firma digital de un mensaje, se deberá tener en cuenta que para elaborar la rúbrica del mensaje, éste debe estar dentro del intervalo de codificación de la clave privada y pública, tanto del emisor como del receptor, lo mismo se debe cuidar en el caso del paso de rúbrica a firma. En caso de que esto no suceda, la rúbrica deberá ser trozada en bloques y codificar cada uno de los mismos.

Este análisis puede ser fácil, sí el mensaje se ha de enviar a una sola persona, pero sí se trabaja en red y un usuario desea comunicarse con diferentes usuarios de la red, sería conveniente determinar un proceso que permitiera no tener que llevar a cabo el análisis anterior para cada uno de los mensajes que alguien desee enviar. Para evitar este inconveniente, Rivest, Shamir y Adleman propusieron un protocolo orientado a una red de comunicaciones.

1. Se elige un umbral h.

2. Cada usuario publica dos pares de claves públicas (a_1,a_2) y (b_1,b_2). La primera de ellas servirá para ser utilizado en el codificado de mensajes, y el otro par, para ser utilizado en la verificación de la firma.

3. Los módulos de cada uno de los usuarios deberán verificar la siguiente condición: $b_1<h<a_1$.

Con las condiciones anteriores, para que un usuario envíe un mensaje firmado a otro usuario, bastará solamente con que los bloques del mensaje a enviar verifiquen la siguiente condición: $0<\{b_1\}$. Una vez dividido el mensaje en bloques que cumpla esta condición, se firmará el mensaje de la manera tradicional.

Actualmente el sistema de codificación RSA, es el más difundido a nivel mundial en parte debido a la gran publicidad y agresivo mercadeo realizado por RSA Data Security. Una de las más importantes compañías en el área de protección de datos, la cual ha comercializado varias versiones de su sistema de seguridad tanto para sistema Windows como para Macintosh.

SEGURIDAD DE LA FIRMA DIGITAL RSA

El ataque al protocolo de firma digital con el criptosistema RSA es el mismo que el que se debería llevar a cabo para romper el propio criptosistema, dado que en ambos casos las operaciones que se realizan son las mismas.

IV.2.2.- FIRMA DIGITAL DEL CRIPTOSISTEMA EL GAMAL

Para que una persona desee enviar un documento firmado por medio del criptosistema El Gamal[44] requiere de efectuar los siguientes pasos:

1. Generar un número aleatorio h, tal que mcd $(h,\phi(n))= 1$.
2. Calcular el elemento $r \equiv \alpha^h \pmod{n}$.
3. Resolver la congruencia: $m \equiv a \cdot r + h \cdot s \pmod{\phi(n)}$.

Una vez efectuado lo anterior, el emisor tendrá la pareja (r,s) como firma digital para el documento deseado.

Para poder corroborar la autenticidad de la firma digital del emisor, el receptor deberá hacer las siguientes operaciones.

1. Calcular $r^s \equiv (\alpha^h)^s \pmod{n}$ y $(\alpha^a)^r \pmod{n}$.
2. Calcular $(\alpha^a)^r (\alpha^h)^s \pmod{n}$ y comprobar que es igual a $\alpha^m \pmod{n}$.

SEGURIDAD DE LA FIRMA DIGITAL EL GAMAL

Para conseguir la falsificación de la firma del emisor en un mensaje determinado, un tercero, tendría que resolver la ecuación $\alpha^m = (\alpha^a)^r \cdot r^s$ con las incógnitas r y s. Sí fijara "r" y tratara de resolver la ecuación con "s" como incógnita, se encontraría con un problema de

logaritmos discretos; por otra parte, si fijara "*s*" e intentara resolver la ecuación para "*r*", se encontraría ante una congruencia exponencial mixta, para lo que no hay algoritmo conocido. Este problema se conoce como el *problema de la firma digital El Gamal.*

OBSERVACIONES: Las firmas digitales calculadas con este criptosistema son más largas que las realizadas con el criptosistema RSA. Por otra parte, este criptosistema es más lento para codificar y verificar la firma digital que el RSA.

IV.2.3.- FIRMA DIGITAL ESTÁNDAR DEL NIST

En agosto de 1991, el Instituto Nacional de Estándares y Tecnología de los Estados Unidos (National Institute of Standard and Technology, NIST) propuso un estándar para la firma digital (Digital Signature Standard, DSS)[44] y su algoritmo correspondiente: DSA (Digital Signature Algorithm), solicitando comentarios públicos para la adopción del estándar propuesto. El objetivo era el de proporcionar a las oficinas gubernamentales de los Estados Unidos una forma de estándar para firmar sus comunicaciones cuando esto fuera necesario.

El DSA propuesto, es una variante de la firma digital de El Gamal, y el protocolo correspondiente es el siguiente:

⇒ Cada usuario elige los siguientes parámetros:

1. p: un número primo con $2^{511}<p<2^{512}$.
2. q: un divisor primo de p-1, con $2^{159}<q<2^{160}$.
3. g: un generador del único subgrupo cíclico de Z^*_p de orden q.
4. Su clave privada, x, un entero con $0<x<q$.
5. Su clave pública, y, $y=g^x \pmod p$.

⇒ Una vez teniendo los parámetros anteriores, la firma se elabora como sigue:

Sea $H : \mathrm{M} \rightarrow \mathrm{Z}$ una función hash unidireccional, y supongamos que el mensaje a firmar es m.

1. Se selecciona un entero aleatorio **k** para cada mensaje, con $0<\mathbf{k}<\mathbf{q}$.
2. Se calcula el valor de $\mathbf{r}= (\mathbf{g}^{\mathbf{k}} \pmod{\mathbf{p}}) \pmod{\mathbf{q}}$.
3. Se resuelve la congruencia: $H(m) \equiv -x \cdot r + k \cdot s \pmod{q}$ para s.

La firma digital para el mensaje m es la pareja (r,s).

$\Rightarrow$ Para verificar la firma se procede como sigue:

1. Se calcula el valor de $w \equiv s^{-1} \pmod{q}$
2. Se calculan $u_1 \equiv H(m) \cdot w \pmod{q}$ y $u_2 \equiv r \cdot w \pmod{q}$.
3. Se calcula $v \equiv (g^{u1} y^{u2} \pmod{p}) \pmod{q}$.
4. Se comprueba que $v=r$.

Para ver como funciona la verificación, basta notar que la congruencia anterior se tiene: $w \cdot H(m) + x \cdot r \cdot w \equiv k \pmod{q}$, donde $w \equiv s^{-1} \pmod{q}$, o $u_1 + x \cdot u_2 \equiv k \pmod{k}$. Finalmente, elevando g a $u_1 + x \cdot u_2$, se tiene:

$$r = (g^{u1} y^{u2} \pmod{p}) \pmod{q}$$

Al igual que la firma digital El Gamal, la firma digital del NIST se puede extender a un grupo cíclico de orden q.

La elección de los parámetros de la firma digital DSS se llevan a cabo como sigue:

1. En primer lugar se genera un número primo q repitiendo la elección aleatoria de un número impar q con $2^{159} < q < 2^{160}$ y comprobando su "primalidad" hasta que q sea primo.

2. A continuación se genera el primo p repitiendo la elección aleatoria de un número entero n con $\frac{2^{511}-1}{2q} < n < \frac{2^{512}-1}{2q}$ hasta que $p = 2n \cdot q+1$ sea primo.

3. Posteriormente, se genera un elemento g de orden q del grupo Z^*_p repitiendo la elección aleatoria de un número entero h con $1<h<p-1$ y computando $g=h^{(p-1)/q}$ hasta que $g\neq 1$.

Las ventajas mencionadas por el NIST para su propuesta de firma digital estándar son las siguientes:

1. La seguridad del DSS propuesto está basada en la dificultad del problema del logaritmo discreto en el subgrupo cíclico de orden q de Z^*_p generado por g. Sin embargo, como el mejor algoritmo conocido para este problema requiere calcular logaritmos en Z^*_p, se puede decir, que su seguridad está basada en la dificultad del problema del logaritmo discreto Z^*_p.

2. La ventaja de trabajar en un subgrupo de Z^*_p es que los tamaños de las firmas son menores. Por ejemplo, si $p\approx 2^{512}$, entonces la firma digital El Gamal (en el grupo Z^*_p) tiene 1024 bits, mientras que con la propuesta del NIST, la firma sólo tiene 320 bits.

3. El DSS no puede ser utilizado, de forma obvia, para codificación. (lo que para algunos supone más un inconveniente que una ventaja).

El DSS, junto con su algoritmo, el DSA, es sólo una parte de un proyecto más amplio propuesto por el NIST y la Agencia Nacional de Seguridad del gobierno de los Estados Unidos (National Segurity Agency, NSA), cuyo nombre es ***Capstone***. Capstone es un proyecto para desarrollar un estándar para la criptografía de clave pública para el gobierno estadounidense, y consta de cuatro componentes.

Todas las partes de Capstone tienen una seguridad de 80 bits, y todas las claves implicadas tienen la misma seguridad. Las primeras implementaciones de este proyecto se han

cristalizado en la creación de un chip llamado ***Clipper***. Este chip pemite codificar mensajes, mediante el algoritmo ***Skipjack***. También Clipper ha sido propuesto como estándar del gobierno estadounidense, de modo que todas las comunicaciones internas gubernamentales se harían con este chip, así como cualquier comunicación de quien desee hacer negocios con el gobierno.

IV.2.4.- FUNCIONES HASH

Una función Hash se puede definir como una función capaz de crear un mensaje resumido (message digest) de un documento completo, este mensaje será la huella digital que caracterizará al mensaje original y cualquier cambio hecho al mismo provocará que la función Hash produzca un mensaje distinto al decodificar.

Para que un usuario envíe un documento firmado, primero le alplica al documento completo una función Hash, produciéndose así un mensaje resumido. Una vez hecho lo anterior, el emisor codificará el resumen del mensaje por medio de su llave privada. El mensaje codificado formará la firma digital del mensaje.

Una vez recibido el mensaje, el software del receptor hará dos operaciones por separado: una verificar la identidad del emisor y la segunda, determinar si el mensaje ha sido alterado durante el tránsito. Para verificar la identidad del emisor el sistema tomará la firma digital y la llave pública del emisor para decodificar la firma digital, la cual producirá un mensaje resumido. Si esto ocurre de manera satisfactoria, el receptor tendrá la certeza de que la persona que envió el mensaje es quien dice ser.

Para asegurarse que el mensaje no ha sido modificado, el receptor le aplicará al mensaje la misma función Hash que el emisor utilizó. Comparando ambos mensajes resumidos. el receptor se asegurará que un tercero no ha alterado el contenido del mensaje.

Como el emisor utiliza su clave secreta para codificar la parte comprimida del mensaje, es posible probar ante una tercera persona que la firma solo ha podido ser generada por el usuario que guarda la componente secreta.

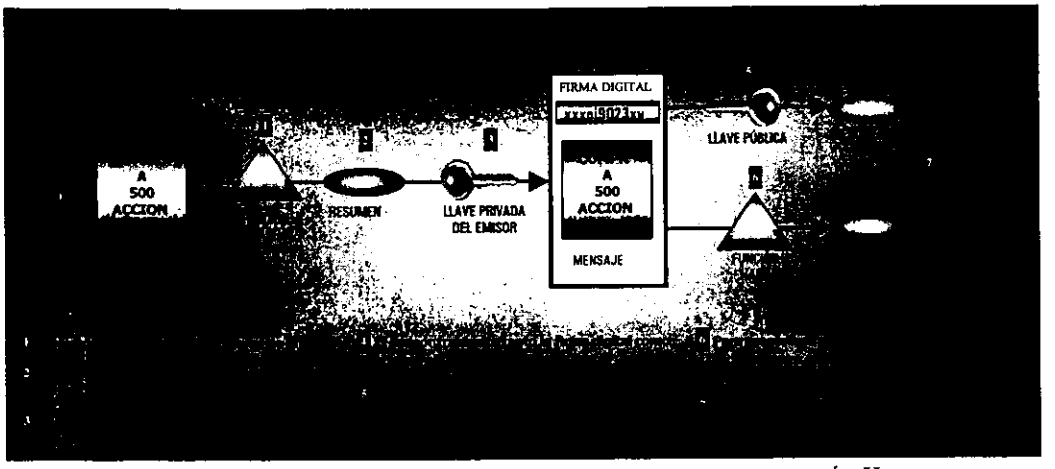


FIG. 4.1.- DIAGRAMA DE BLOQUES DE LA FIRMA DIGITAL CON FUNCIÓN HASH.

La elección adecuada de una función Hash para firmar un mensaje de forma digital deberá tomar en cuenta que la longitud del resumen sea lo suficientemente larga como para evitar que una investigación exhaustiva por parte de un tercero obtenga la información a corto plazo. Por ejemplo, si una función Hash proporciona un resumen de 100 bits, la investigación exhaustiva debería de llevar por lo menos 2^{100} intentos para conseguir igualar un valor dado, y aproximadamente 2^{50} intentos, de media, para poder producir dos entradas con el mismo resumen.

Las funciones Hash más utilizadas con propósitos criptográficos son las funciones MD2, MD4 y MD5 (Message Digest, MD), propuestas por Rivest. Estas funciones producen resúmenes de 128 bits y el único ataque que se conoce contra ellas es la investigación exhaustiva.

IV.3.- CERTIFICADOS DE AUTENTIFICACIÓN

A pesar del uso de una firma digital o de un sistema de llave pública por parte de 2 usuarios, esto no dará una total certeza de que la persona con quien se está en contacto sea quien dice ser, ya que una persona determinada puede generar 2 llaves (pública y privada) con el nombre de otra. Por tal motivo y para asegurar una mayor confianza sobre todo en contactos entre 2 usuarios ocasionales o contactos iniciales, se han creado las *Autoridades de Certificación* (Certificate Authority, CA's)[45][46].

Las Autoridades de Certificación proporcionan un documento firmado, este será un *certificado* que asociará una clave pública a un usuario determinado y que será válido durante un periodo de tiempo indicado por la misma CA.

Para poder adquirir un certificado digital, una persona se presentará ante una CA con una copia de su llave pública y una serie de requisitos solicitados por la CA, los cuales tienen el propósito de asegurar con la mayor certeza la identidad del solicitante.

Una vez obtenido el certificado, un usuario que reciba un documento enviado por otro, y que tenga un certificado, recibirá el documento codificado con un sistema de llave pública o una firma digital, como lo recibía de forma habitual, además recibirá un mensaje adicional (el certificado) que servirá para que el software del receptor identifique a la Autoridad de Certificación que lo expidió y se contacte con ella para pedir información acerca del dueño del certificado, o bien si el certificado no ha expirado o ha sido revocado debido a un uso indebido o pérdida de la clave por parte del dueño.

Los certificados fueron inventados para garantizar la identidad del dueño de una llave pública, pero en la actualidad pueden hacer mucho más. Esto es debido a que cada certificado contiene un amplio abanico de posibilidades, algunas de estas ya vienen predefinidas, pero pueden ser añadidas tantas como se deseen. Por ejemplo una Corporación quizá desee crear

fuertes estructuras de certificación que puedan proporcionar información acerca de los privilegios proporcionados a su dueño. Esto es, un certificado puede establecer una limitación en el tamaño de un contrato que un empleado pueda garantizar. El tesorero de la Corporación, por ejemplo, puede tener un certificado que contenga una firma digital aplicable a cualquier contrato no importando que tan grande sea este.

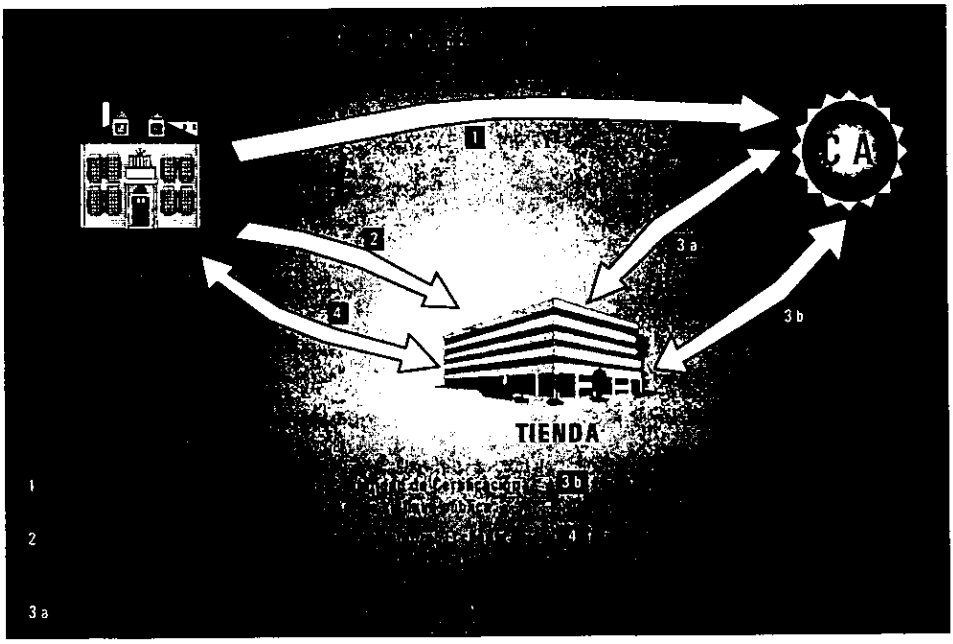


FIG. 4.2.- UTILIZACIÓN DE CERTIFICADOS PARA FIRMAS DIGITALES.

La codificación asimétrica y las firmas digitales han sido usadas por varios años a nivel mundial en grupos cerrados de usuarios, tales como Instituciones Financieras u Organizaciones Médicas. Las Autoridades de Certificación, como Belsign en Bélgica, COST en Suecia y Deutsche Telekom's Telsec en Alemania, han ofrecido estos servicios por años, basados en el estándar X.509. Estos servicios han sido proporcionados a grupos corporativos que ocasionalmente operan a nivel nacional.

Algunos de los datos más importantes de este formato son los siguientes:

Versión: 1, 2 o 3
Número de Serie: 0000000000000000
Emisor del Certificado: VeriMex
Identificador del Algoritmo usado en la firma: RSA, DSA o CE
Periodo de Validez: De Enero 2000 a Dic 2000
Sujeto: Jesús Angel
Información de la clave pública del sujeto: la clave, longitud, y demás parámetros
Algunos datos opcionales, extensiones que permite la v3
Firma de la Autoridad Certificadora

Un certificado digital entonces se reduce a un archivo de uno o dos Kbytes de tamaño, que autentifica a un usuario de la red (Fig. 4.3).

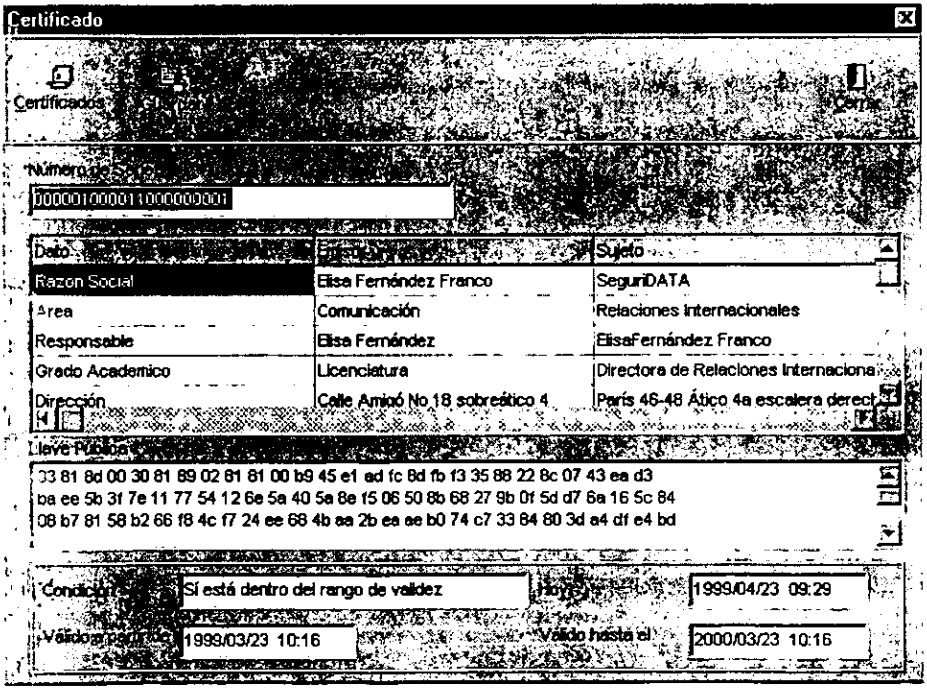


FIG. 4.3.- APARIENCIA DE UN CERTIFICADO EN PANTALLA.

IV.3.1.- EL CASO DE VeriSign

Una de las Autoridades de Certificación más popular en el mundo es la compañía norteamericana VeriSign[47], esta compañía proporciona servicios de certificación para firmas digitales con varios niveles de seguridad, de acuerdo a las necesidades de seguridad que un cliente solicite.

El certificado VeriSign clase 1, solamente confirma que el nombre de la persona que firmó digitalmente y el nombre del dueño del certificado sean los mismos, además de proporcionar un seguro en el caso de que el usuario sea afectado por un uso indebido de su certificado. Esta clase de certificado no es capaz de garantizar que alguien sea una persona real, ya que se solicita llenando una forma publicada en la página de VeriSign en Internet.

El certificado VeriSign clase 2, este certificado trata de corroborar con el respaldo de la información proporcionada por otras fuentes, la identidad y presencia física del usuario, ya que el certificado se proporciona una vez que la compañía ha corroborado que sus datos sean consistentes con los contenidos en algunas bases de datos tales como, bancos, compañía de agua, luz o teléfono, además de proporcionar un seguro similar al de la clase 1. Sin embargo, aún no está muy claro si VeriSign o alguna otra compañía tiene las herramientas suficientes para impedir el robo de la identidad por alguien que conozca suficientes datos sobre la vida de una persona, aunado a esto, existen dudas acerca de la información contenida en las bases de datos con las cuales se compara la información proporcionada por el cliente ya que existen experiencias de la creación de identidades en el uso y expedición de tarjetas de crédito.

El certificado VeriSign clase 3, trata de borrar las dudas sobre la existencia física y la posible suplantación de identidades existentes en las otras 2 clases de certificado al solicitar al cliente que se presente en una notaría pública con suficientes pruebas de su identidad, con el fin de que el notario proporcione un documento firmado que certifique tanto la presencia física del cliente en cuestión ante el notario como la congruencia de este con los datos proporcionados, una vez

realizado esto, el solicitante deberá enviar este documento expedido por el notario a VeriSign con los datos de la notaria en la cual se expidió este documento.

A pesar de la amplia variedad de certificados expedidos por VeriSign, la compañía se enfrenta ante el problema legal que implica la certificación de una firma digital, debido principalmente a que la legislación de EUA no tiene en la actualidad una ley que regule esta actividad a nivel Nacional, por otra parte existen leyes a nivel estatal por parte de algunos Estados pero muchas de éstas, difieren entre sí.

IV.3.2.- IMPLICACIONES LEGALES

Uno de los principales usos que una firma escrita o digital tienen en la actualidad es en la firma de contratos comerciales, y en vista que el comercio digital[48][49][50][51] ha crecido de una manera sorprendente en los últimos años debido al uso de Internet, uno de los aspectos decisivos para afianzar el comercio electrónico está constituido por el entorno jurídico, es decir, las leyes que sirvan de soporte para las transacciones, e introduzcan el concepto de seguridad jurídica en el mercado digital.

Existe una opinión generalizada de que, si ya es complicado, en la vida física, demostrar la existencia de una deuda que no se ha formalizado en un título ejecutivo, la dificultad probatoria será mayor en un ámbito virtual en la que el consentimiento se transmite en forma de bits.

La firma digital pretende ser el instrumento que permitirá, entre otras cosas, determinar de forma confiable si las partes que intervienen en una transacción son realmente las que dicen ser, y si el contenido del contrato ha sido alterado o no posteriormente.

A pesar de que la firma digital pretende tener las mismas características que la firma manual, en la actualidad esta idea no es respaldada por la mayoría de los modelos jurídicos mundiales, es más, si solo restringimos este hecho a países desarrollados o de primer mundo,

encontraremos que algunos aún están en un proceso de discusión sobre la aceptación de una firma digital como prueba escrita de que una persona pudiera haber redactado un documento electrónico, por lo que aún faltará por determinar la legalidad y los estándares que deben cumplir tanto las firmas digitales como las Autoridades de Certificación.

Varios países europeos han adoptado las firmas digitales. Pero las estructuras no son compatibles y aún no son internacionalmente aceptadas, es más, legalmente no son aceptadas como prueba ante una corte en algunos estados de Estados Unidos como en algunos países europeos, por lo que en el caso de que el firmante de un documento no desee reconocer su firma digital no existe una autoridad a la que se pueda recurrir.

En Estados Unidos, el primer estado que redactó una ley que regula los aspectos jurídicos de la firma digital como instrumento probatorio se aprobó en 1997, en Utah. Posteriormente surgieron proyectos legislativos en Georgia, California y Washington. Desafortunadamente, las leyes tienen pequeñas diferencias de estado a estado. Eventualmente, se espera que el gobierno de los EUA publique una ley general que permita minimizar estas diferencias. En la actualidad, la Asociación Americana de Barras tiene un comité el cual proporciona recomendaciones legales sobre el uso de las firmas digitales en el comercio electrónico.

El caso europeo es más complejo aún ya que existen 2 distintos tipos de sistemas legales en los países de Europa. En la mayoría de los países europeos incluyendo a Alemania, Holanda y el Reino Unido, el sistema legal acepta todo tipo de evidencia, esto es conocido como sistema de libre evidencia. Sin embargo, en otros países como Francia y Bélgica la ley define el tipo de evidencia que las cortes puedan aceptar, en estos países, las firmas digitales no son aceptadas como una firma válida en el sentido legal. Incluso, dentro de Europa, es excepcional el caso de Francia donde el simple uso de la criptografía está prohibida a no ser que se disponga de una licencia. En sentido contrario, algunos países nórdicos no tienen ningún tipo de limitación respecto a la exportación. Hay que tener en cuenta que este tipo de limitaciones tienen muy poca efectividad ante la dificultad existente para vigilar la exportación de software. Por otra parte,

aunque se prohiba la exportación de los sistemas seguros, los algoritmos criptográficos y protocolos para la consecución de servicios de seguridad son frecuentemente públicos, con lo que los sistemas se pueden volver a desarrollar en el país de destino.

En 1996, Alemania se convirtió en el primer país europeo en publicar una ley referente a las firmas digitales. La ley alemana está divida en 2 partes, un texto principal y un reglamento que desarrolla aspectos concretos de la ley, como el procedimiento de concesión, transferencia y revocación de una licencia de entidad certificadora, así como los deberes de los certificadores, el periodo de validez de los certificados, los métodos de control de los certificados, los requisitos de los componentes técnicos y el procedimiento de examen de los mismos.

Según la ley alemana, la emisión de certificados y la creación de claves privadas para firmas digitales acostumbra depender de una pluralidad de entidades que están jerarquizadas de una manera que las de nivel inferior obtienen su capacidad de certificación de otras entidades de nivel superior. Finalmente, en la cúspide de la pirámide suele hallarse una autoridad certificadora, que puede pertenecer al Estado, y que en el proyecto alemán coincide con el organismo que controla las telecomunicaciones.

A pesar de la excelente respuesta por parte del sistema jurídico alemán ante el creciente uso de firmas digitales y certificados expedidos por las autoridades de certificación, la ley alemana ha recibido muchas críticas por parte de usuarios y expertos ya que la consideran altamente centralizada e inflexible. Otro argumento en contra es que la ley alemana trata diferente a las firmas escritas y a las digitales, según algunas voces, sólo regula la infraestructura de las firmas. Aunado a esto, la ley alemana no contempla la verificación de datos de un solicitante por parte de una Autoridad de Certificación con otra CA, lo cual sí contempla la ley en Bélgica.

Debido a estas diferencias que existen, la Unión Europea ha lanzado un comunicado con fecha 8 de octubre de 1997, la cual persigue el fin de sensibilizar a los Estados miembros sobre el creciente uso de Internet como plataforma de comunicación y de comercio, así como de la

necesidad de establecer un marco uniforme en materia de codificado de la información y firma digital. El comunicado recuerda que los mensajes en Internet pueden ser interceptados y manipulados, y esta circunstancia puede impedir que se conceda validez a los documentos enviados a través de la red.

Las tecnologías de codificado pueden resolver este problema, ya que constituyen una herramienta esencial para garantizar la seguridad y la fiabilidad de las comunicaciones y transacciones electrónicas.

Dos aplicaciones importantes de estas tecnologías son las firmas digitales y el codificado de mensajes. Varios estados miembros han anunciado su intención de promulgar leyes específicas relativas a la codificación, y algunos ya lo han hecho. Pero la divergencia legal y técnica de estas regulaciones podría constituir un serio obstáculo para el mercado interior e impedir el desarrollo de nuevas actividades económicas relacionadas con el comercio electrónico.

Los objetivos del comunicado de la CE son:

- El desarrollo de un marco legal que asegure el funcionamiento de los productos y servicios de codificado en el Mercado Interior.

- Establecer un marco europeo para las firmas digitales.

Este comunicado anuncia la intención de la comisión de proponer una legislación que cubra estos 2 objetivos durante el primer semestre de 1998.

El uso de firmas digitales exige el ajuste y la armonización de diversas áreas.

Actualmente, la mayor parte de los problemas se centran en los siguientes puntos:

- Ausencia de requisitos uniformes para las Autoridades de Certificación.

- Ausencia de requisitos uniformes para los productos de firma digital.
- Ausencia de normas uniformes en materia de responsabilidad.
- Ausencia de normas uniformes respecto al reconocimiento legal de las firmas digitales y su eficacia probatoria.

La evidente naturaleza transfronteriza de las firmas digitales exige el reconocimiento mutuo de los requisitos legales establecidos en esta materia por cada Estado, con el fin de evitar la fragmentación del comercio electrónico en el Mercado Interior.

La Comisión propone la siguiente estrategia:

- Establecer un marco comunitario para las firmas digitales con el fin de que la regulación de cada Estado no genere barreras internas para el comercio electrónico.

- Determinar unos requisitos comunes para las Autoridades de Certificación en Europa.

- El sistema jurídico de cada Estado debe reconocer y tratar las firmas digitales de manera idéntica a las firmas convencionales.

- La interoperabilidad entre diferentes sistemas de codificado y firmas digitales es absolutamente necesaria.

Es evidente que la eficacia de las leyes que se publiquen radica en su uniformidad, ya que si su contenido difiere en cada estado o país, será difícil su aplicación a un entorno global como Internet.

IV.4.- PROTOCOLOS DE SEGURIDAD.

Un protocolo de segurida[52] es la parte visible de una aplicación, es el conjunto de programas y actividades programadas que cumplen con un objetivo específico y que usan esquemas de seguridad criptográfica.

Los protocolos de seguridad procura resolver algunos de los problemas de la seguridad como la integridad, la confidencialidad, la autenticación y el no rechazo, mediante sus diferentes características.

Las características de los protocolos se derivan de las múltiples posibilidades con que se puede romper un sistema, es decir, robar información, cambiar información, leer información no autorizada, y todo lo que se considere no autorizado por los usuarios de una comunicación por red, por ejemplo, sobre la seguridad por Internet se deben de considerar las siguientes tres partes: seguridad en el browser (Netscape o Explorer), la seguridad en el Web server (el servidor al cual nos conectamos) y la seguridad de la conexión.

IV.4.1.-SSL (SECURE SOCKETS LAYER)

SSL es el protocolo de comunicación segura más conocido y usado actualmente[52], SSL actúa en la capa de comunicación y es como un túnel que protege a toda la información enviada y recibida. El SSL es usado en gran cantidad de aplicaciones que requieren proteger la comunicación.

Con SSL se pueden usar diferentes algoritmos para las diferentes aplicaciones, por ejemplo, usa DES, TDES, RC2, RC4, MD5, SHA-1, DH y RSA, cuando una comunicación está bajo SSL la información que se codifica es:

- El URL del documento requerido.
- El contenido del documento requerido.

- El contenido de cualquier forma requerida.
- Los archivos de identificación usuario-página web (cookies) enviados del browser al servidor.
- Los "cookies" enviados del servidor al browser.
- El contenido de las cabeceras de los http.

El procedimiento que se lleva a cabo para establecer una comunicación segura con SSL es el siguiente:

1. EL cliente (browser) envía un mensaje de saludo al Server "ClientHello".
2. El servidor responde con un mensaje "ServerHello".
3. El servidor envía su certificado.
4. El servidor solicita el certificado del cliente.
5. El cliente envía su certificado: sí es válido continúa la comunicación si no, "para" o "sigue" la comunicación sin certificado del cliente.
6. El cliente envía un mensaje "ClientKeyExchange" solicitando un intercambio de claves simétricas sí es el caso.
7. El cliente envía un mensaje "CertificateVerify" sí se ha verificado el certificado del servidor, en caso de que el cliente este en estado de autentificación.
8. Ambos cliente y servidor envían un mensaje "ChangeCipherSpec" que significa el comienzo de la comunicación segura.
9. Al término de la comunicación ambos envían el mensaje "finished" con lo que termina la comunicación segura, este mensaje consiste en un intercambio de un fragmento de información codificada (Hash) de toda la conversación, de manera que ambos están seguros que los mensajes fueron recibidos intactos (íntegros).

La versión más actual de SSL es la v3, existen otro protocolo parecido a SSL sólo que es desarrollado por IETF que se denomina TLS (Transport Layer Security Protocol) y difiere en que usa un conjunto un poco más amplio de algoritmos criptográficos. Por otra parte existe

también SSL plus, un protocolo que extiende las capacidades de SSL y tiene por mayor característica que es interoperable con RSA, DSA/DH y CE (Criptografía Elíptica).

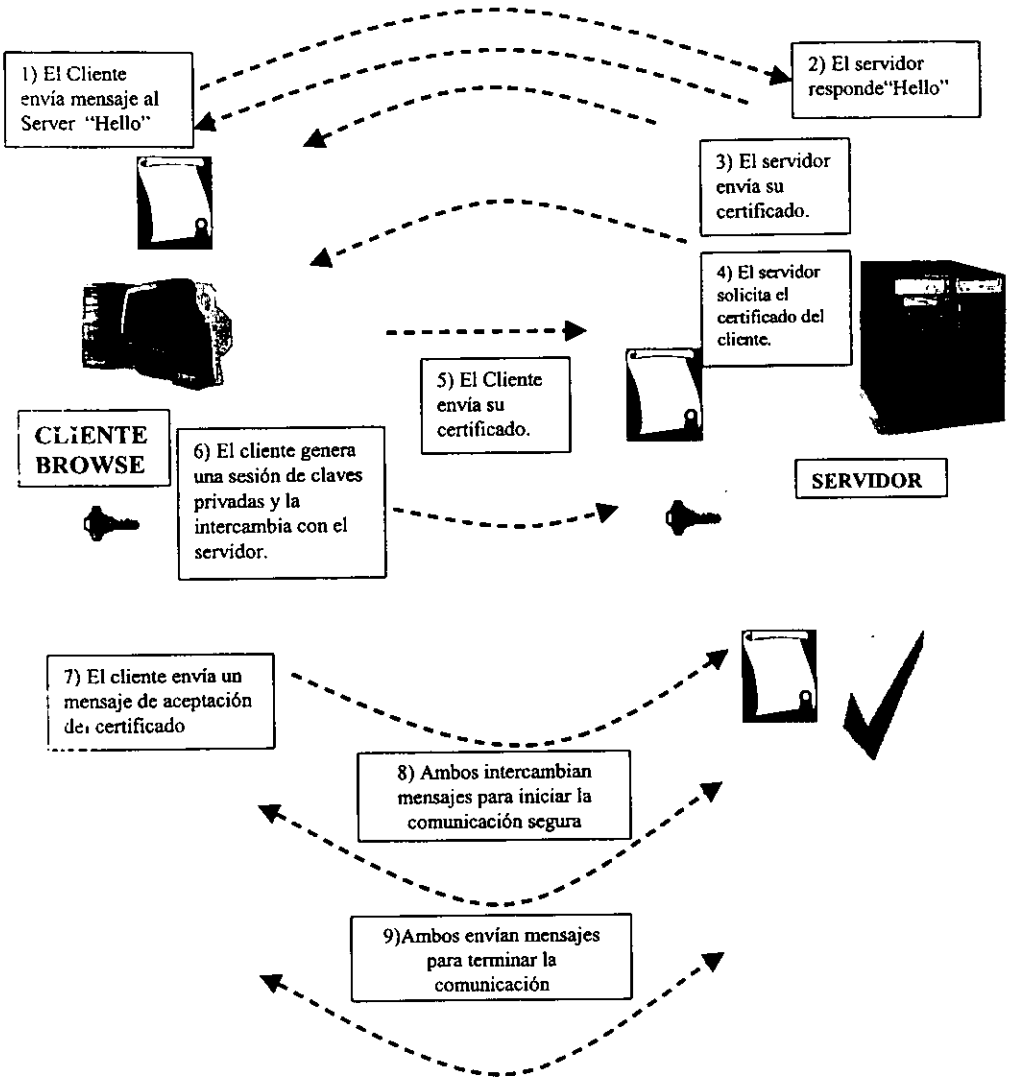


FIG. 4.4.- DESCRIPCIÓN DEL PROTOCOLO SSL

IV.4.2.- SET (SECURE ELECTRONIC TRANSACTION) 

El protocolo SET[52] está especialmente diseñado para asegurar las transacciones por Internet que se pagan con tarjeta de crédito. Esto es debido a que una gran cantidad de transacciones de compra por Internet son efectuadas con tarjeta de crédito, por otro lado SSL deja descubierto alguna información sensible cuando se usa para lo mismo. La principal característica de SET, es que cubre estos huecos en la seguridad que deja SSL.

Por ejemplo, con el SSL sólo se protege el número de tarjeta cuando se envía del cliente al comerciante, sin embargo, no hace nada para la validación del número de tarjeta, para verificar sí el cliente esta autorizado a usar ese número de tarjeta, para ver la autorización de la transacción del banco del comerciante, etc. Además que el comerciante puede fácilmente guardar el número de tarjeta del cliente. En fin todas estas debilidades son cubiertas por SET, éste permite dar seguridad tanto al cliente, al comerciante como al banco emisor de la tarjeta y al banco del comerciante.

El proceso de SET es el siguiente:

1. El cliente inicializa la compra: consiste en que el cliente usa el browser para seleccionar los productos a comprar y llena la forma de orden correspondiente. SET comienza cuando el cliente hace clic en "pagar" y se envía un mensaje de iniciar SET.

2. El cliente usando SET envía la orden y la información de pago al comerciante: el software SET del cliente crea dos mensajes uno conteniendo la información de la orden de compra, el total de la compra y el número de orden. El segundo mensaje contiene la información de pago, es decir, el número de la tarjeta de crédito del cliente y la información del banco emisor de la tarjeta. El primer mensaje es codificado usando un sistema simétrico y es empaquetado en un sobre digital que se codifica usando la clave pública del comerciante. El segundo mensaje, también es codificado pero usando la

clave pública del banco (esto previene que el comerciante tenga acceso a los números de tarjetas de los clientes). Finalmente el cliente firma ambos mensajes.

3. El comerciante pasa la información de pago al banco: el software SET del comerciante genera un requerimiento de autorización, éste es empaquetado (con un hash) y firmado por el comerciante para probar su identidad al banco del comerciante, además de ser codificado con un sistema simétrico y guardado en un sobre digital que es codificado con la clave pública del banco.

4. El banco verifica la validez del requerimiento: el banco decodifica el sobre digital y verifica la identidad del comerciante, en el caso de aceptarla decodifica la información de pago del cliente y verifica su identidad. En tal caso, genera un requerimiento de autorización, lo firma y envía al banco que generó la tarjeta del cliente.

5. El emisor de la tarjeta autoriza la transacción: el banco del cliente (emisor de la tarjeta) confirma la identidad del cliente, decodifica la información recibida y verifica la cuenta del cliente en caso de que no exista problemas, aprueba el requerimiento de autorización, lo firma y lo regresa al banco del comerciante.

6. El banco del comerciante autoriza la transacción: una vez recibida la autorización del banco emisor, el banco del comerciante autoriza la transacción la firma y la envía al servidor del comerciante.

7. El servidor del comerciante complementa la transacción: el servidor del comerciante da a conocer que la transacción que la tarjeta fue aprobada y muestra al cliente la conformidad de pago, y procesa la orden que pide el cliente, terminado la compra cuando se le son enviados los bienes que compró el cliente.

8. El comerciante captura la transacción: en la fase final de SET el comerciante envía un mensaje de "captura" a su banco, esto confirma la compra y genera el cargo a la cuenta del cliente, así como acreditar el monto a la cuenta del comerciante.

9. El generador de la tarjeta envía el aviso de crédito al cliente: el cargo de SET aparece en estado de cuenta del cliente que se le envía mensualmente.

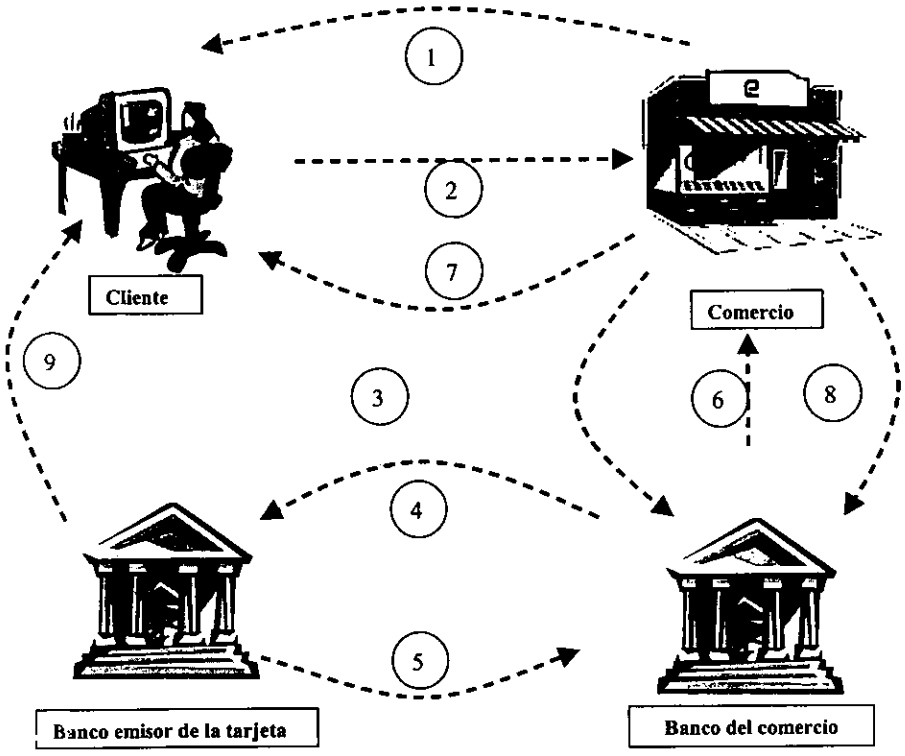


FIG. 4.5.- DESCRIPCIÓN DEL PROTOCOLO SET.

IV.4.3.- PROTOCOLO X.509

El protocolo X.509[52] [53] es el sistema de certificados de clave pública más utilizado. Su origen es el directorio X.500, inventado por la UIT para dar servicio al correo electrónico X.400. Actualmente se utiliza en los protocolos seguros y en los sistemas de correo Internet más conocidos, excepto el PGP. Permite trabajar con CA y anidar certificados para crear estructuras jerárquicas.

En la figura 4.7 se puede ver el formato de los certificados X.509.

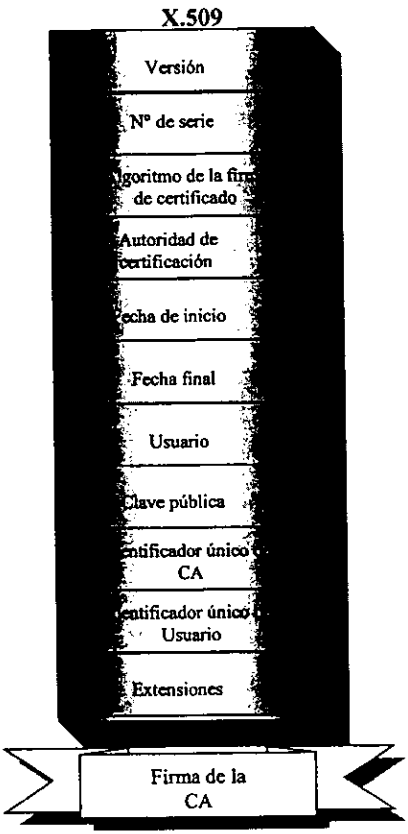


FIG. 4.6.- DESCRIPCIÓN DEL CONTENIDO DEL PROTOCOLO X.509

- **Versión**. La versión de protocolo X.509.
- **Número de serie** (SerialNumber). Identificador único del certificado, asignado por el CA.
- **Algoritmo de la firma del certificado** (Signature). X.509 permite utilizar diferentes algoritmos para firmar el certificado, este campo lleva el identificador del algoritmo.
- **Autoridad de certificación** (Issuer). Nombre de la CA.
- **Fechas de inicio y final** (Validity). El certificado sólo tiene validez entre estas dos fechas. Es conveniente no permitir un período de validez largo y así obligar a renovar certificados y claves muy frecuentes.
- **Usuario** (Subject). Nombre del usuario.
- **Clave pública** (SubjectPublicInfo). La clave pública del usuario, permite múltiples longitudes.
- **Identificador único de la CA** (IssuerUniqueID). Las CA tienen un número de identificación único en el mundo.
- **Identificador único del usuario** (SubjectUniqueID). Los usuarios tienen un identificador único en la CA para todos sus certificados.
- **Extensiones**. Posibles extensiones de la información.
- **Firma del CA**. El CA firma con su clave privada todos los campos anteriores.

IV.4.4.- Intimidad Bastante Buena PGP® (Pretty Good Privacy)

En 1991, un profesor del MIT llamado Philip R. Zimmermann crea la versión 1.0 de PGP[54] [55] [56] [57] [58] [59] como una forma de rebelarse contra una propuesta del Senado estadounidense que proponía tener una puerta trasera en todos los sistemas de codificación de datos, la cual podría ser utilizada por el gobierno, con el fin de enterarse de posibles filtraciones de información al exterior de EUA que pudieran ser un riesgo en la seguridad nacional

(espionaje, contrabando, narcotráfico). Esta propuesta no fue aceptada pero a pesar de ello, el PGP ya había sido comercializado dentro y fuera de EUA.

La versión 1.0 de PGP integra en su estructura varios algoritmos independientes:

⇒ Utiliza el algoritmo RSA combinado con un codificador de clave simétrica que Zimmermann diseñó llamado Bass-O-Matic, con el fin de gestionar claves públicas y codificar los mensajes.

⇒ Agrega un algoritmo para hacer los resúmenes de textos (función hash) el MD4.

⇒ Un transportador de 7 bits, *uuecode,* y un programa de compresión , el LZHuf.

La versión 2.0 de PGP y posteriores tienen diferentes componentes ya que se descubrieron errores en los programas que pertenecían a la versión 1.0 por lo que:

⇒ Bass-O-Matic se remplazó por IDEA ya que resultó ser débil para ataques criptográficos.

⇒ MD4 se reemplaza por MD5 ya que su creador Ron Rivest detectó una debilidad en MD4. Esta debilidad provocó que recientemente se lograra romper este algoritmo en tan solo 8 días.

⇒ *uuecode* se reemplaza por una armadura, radix-64-ascii.

⇒ LZHuf resultó ser muy lento por lo que se reemplazó por un algoritmo ZIP.

En la actualidad el PGP es una de las herramientas más populares utilizadas para la transmisión de datos por medio de canales inseguros, esto se debe principalmente a:

✓ El PGP se puede conseguir en varias direcciones de internet de una manera relativamente fácil.

✓ El PGP es cien por ciento un software de aplicación, por lo que no requiere de hardware alguno para su uso.

✓ El programa corre en varias plataformas (UNIX, Windows, MS-2, Mac, etc), y es independiente del sistema operativo.

✓ Ofrece confianza al usuario ya que se basa en algoritmos muy confiables.

PGP proporciona al mismo tiempo los servicios de:

1. **Confidencialidad.** Permite a un usuario, mediante codificación, garantizar que solamente el destinatario podrá leer el mensaje.

2. **Autentificación.** Permite a un usuario firmar un documento antes de enviarlo, lo cual permite:
   •Tener certeza de que el documento no ha sido modificado puesto que ha sido firmado. Si se alterara el mensaje la firma no seria valida.
   •Verificar que el mensaje ha sido firmado por una determinada persona.

3. **Integridad.** La firma antes mencionada tiene la particularidad de que depende no sólo de la identidad del remitente sino también del contenido del mensaje, por lo que si este es alterado, la firma ya no es válida.

IV.4.4.1.- FUNCIONAMIENTO DE PGP

PGP utiliza un generador criptográfico fuerte de números pseudoaleatorios, para generar las claves temporales de las sesiones convencionales. El archivo semilla que necesita se llama "randseed.bin". Puede estar en cualquier directorio, indicado por la variable PGPPATH. Si no

existe ese archivo, se crea automáticamente y se rellena de números verdaderamente aleatorios obtenidos midiendo tiempos entre pulsaciones de tecla.

El generador vuelve a construir el archivo cada vez que se utiliza, mezclando nuevos datos de claves derivados parcialmente con la hora y otras fuentes verdaderamente aleatorias. Ese archivo contiene material aleatorio para semillas y datos aleatorios de claves, para poner a punto el motor de codificado convencional como generador de números aleatorios.

PGP comprime el texto normal antes de codificarlo. Después es demasiado tarde para comprimir; los datos codificados son incomprimibles. La compresión ahorra tiempo de transmisión por modem y espacio en el disco, y, lo que es más importante, refuerza la seguridad criptográfica. La mayoría de las técnicas criptoanalíticas explotan las redundancias del texto normal para romper el codificado. La compresión de datos reduce la redundancia en el texto, mejorando la resistencia al criptoanálisis. Lleva más tiempo comprimir el texto normal, pero merece la pena desde el punto de vista de la seguridad.

La versión 2.3 de PGP utiliza las rutinas gratuitas de compresión ZIP escritas por Jean-loup Gailly, Mark Adler y Richard B. Wales. Este programa ZIP utiliza algoritmos de compresión funcionalmente equivalentes a los que utiliza el nuevo PKZIP 2.0.

Para codificar mensajes PGP emplea un esquema híbrido de clave pública y convencional, el algoritmo de codificado convencional de clave única por bloques es IDEA™, y RSA es el algoritmo de clave pública.

Por último, para elaborar una firma digital, PGP codifica con la clave secreta del usuario. Sin embargo, el PGP no codifica el mensaje completo con esa clave ya que llevaría demasiado tiempo. En lugar de eso, el PGP codifica un "resumen del mensaje", el cual es realizado utilizando el programa MD5, proporcionado al dominio público por RSA Data Security, Inc.

IV.4.4.2.- SEGURIDAD DE PGP

Aunque los algoritmos con los cuales el PGP está hecho, goza en la actualidad de gran reputación a nivel mundial, en la actualidad mucha literatura ha sido escrita por varios autores acerca de las maneras con los cuales se puede atacar al PGP, gran parte de esta se encuentra disponible en Intenet, incluso el propio Philip R. Zimmermann ha dedicado un capítulo completo para tratar este tema dentro del volumen dos de la "Guía para el usuario de PGP". Cabe mencionar que hasta el momento no existe ningún ataque de tipo criptográfico contra el PGP y muchos de los ataques propuestos se limitan a fallos físicos, búsqueda de errores en la aplicación e incluso Tempest.

ATAQUES POR FUERZA BRUTA

Los ataques por fuerza bruta son prácticamente inútiles ya que el PGP utiliza el algoritmo IDEA y para atacarlo se necesitarían 10^{38} intentos. A pesar de esto, existe en el mercado un programa llamado "PGPCrack", es una aplicación de fuerza bruta ampliamente diseminada, diseñada para reventar archivos codificados convencionalmente mediante PGP y que ataca la contraseña de la clave secreta. Depende de un archivo separado de diccionario, y prueba cada palabra como contraseña potencial. En un archivo protegido por PGP mediante codificado convencional, el programa comprueba más de 15,000 palabras por segundo en un Pentium a 100 Mhz. Este programa se aprovecha principalmente de los usuarios que utilizan claves sencillas para codificar sus datos, por lo que de utilizar llaves complicadas PGPCrack no podrá hallar la clave utilizada en un tiempo razonable (hasta $2.11*10^{26}$ años). También se recomienda utilizar 2 juegos de claves, una para codificar archivos que requieran de una débil seguridad y otra para archivos que contengan datos de suma importancia.

CRIPTOANÁLISIS

En la actualidad no existe manera de romper el algoritmo IDEA, y hasta la fecha ha soportado un profundo análisis de seguridad, así como una revisión de varios criptoanalistas

clasificados entre los mejores del mundo no clasificados (Hackers). Parece tener ciertas ventajas de diseño respecto a DES en cuanto a resistencia al criptoanálisis diferencial que se ha empleado para romper el DES.

Por otra parte, si este algoritmo tuviese algunas debilidades sutiles aún no descubiertas, el PGP comprime el texto normal antes de codificar, lo que debería reducirlas bastante. El trabajo computacional que hace falta para romperlo es seguramente mucho más caro que el valor del mensaje.

CABALLOS DE TROYA

El PGP puede ser alterado y producir una versión débil la cual puede ser distribuida con relativa facilidad, por lo cual se recomienda que el PGP se adquiera en sitios autorizados, tales como la página de PGP - MIT.

VIRUS

Se presume que existen ciertos virus o programas destinados a destruir la seguridad del PGP alterando las rutinas del MD5, IDEA o RSA e incluso diseñados para capturar claves y retransmitirlas a intervalos variables por la red. Se recomienda tener un paquete antivirus de reciente creación y aplicaciones que permitan detectar transferencias de información no autorizadas en la red.

IV.4.4.3.- CONSIDERACIONES LEGALES

Las consideraciones legales en el caso específico de PGP fueron las siguientes: en un principio tuvo problemas para su comercialización tanto dentro como fuera de EUA, ya que los creadores de RSA argumentaban que se había violado su patente al utilizar su algoritmo sin su permiso. En la actualidad existen 2 versiones comerciales del PGP, la versión 2.6 para uso

dentro de EUA y la 2.6i para su uso en el extranjero, la diferencia entre pgp2.6x y pgp2.6ix es que pgp26ix usa la librería de enteros grandes MPILIB de Zimermann, que es más rápida que el RSAREF de RSA Data Security Inc. y ambas versiones son cien por ciento legales ya que están dentro de la licencia de RSA Data Security Inc.

La razón de la creación de la versión internacional es que RSAREF es un paquete de software bajo copyright de RSA, y RSA no está autorizada para exportarlo debido a que la International Traffic in Arms Regulations (ITAR) lo prohibe.

Para el uso comercial del PGP, es necesario contar con una licencia de Ascom Tech o bien adquirir una versión con licencia de Via Crypt ya que PGP utiliza el algoritmo IDEA™. Sin embargo, si se desea utilizar PGP para usos particulares no comerciales Ascom-Tech ha dado permiso para que IDEA sea utilizado en todas partes sin necesidad de una licencia.

IV.5.-DINERO ELECTRÓNICO

Una aplicación más, que puede ser realidad gracias a la criptografía es la conocida como dinero electrónico[60], en términos sencillos el dinero electrónico es otra representación de lo que conocemos como dinero o valor, por ejemplo tenemos dinero en billetes emitidos por algún país, podemos tener cheques pagaderos en un banco, bonos, pagarés pagaderos en algún plazo, en fin. El dinero electrónico es físicamente un número que se genera aleatoriamente, se le asigna un valor, se codifica y firma y se envía al banco, ahí el banco valida el número y certifica el valor, y lo regresa al usuario firmado por el banco, entonces el usuario puede efectuar alguna transacción con ese billete electrónico.

Las principales propiedades del dinero electrónico son las siguientes:

1. **Independencia**: la seguridad del dinero digital no debe depender del lugar físico donde se encuentre, por ejemplo, en el disco duro de una computadora personal.

2. **Seguridad**: el dinero digital (el número) no debe de ser usado en dos diferentes transacciones.

3. **Privacidad**: el dinero electrónico debe de proteger la privacidad de su usuario, de esta forma cuando se haga una transacción debe de poder cambiarse el número a otro usuario sin que el banco sepa que dueños tuvo antes.

4. **Pagos fuera de línea**: el dinero electrónico no debe de depender de la conexión de la red, así un usuario puede transferir dinero electrónico que tenga en una "smart card" a una computadora, el dinero digital debe ser independiente al medio de transporte que use.

5. **Transferibilidad**: el dinero electrónico debe de ser transferible, cuando un usuario transfiere dinero electrónico a otro usuario debe de borrarse la identidad del primero.

6. **Divisibilidad**: el dinero electrónico debe de poder dividirse en valores fraccionarios según sea el uso que se de, por ejemplo en valor de 500, 100 y 50.

La serie de pasos que puede seguir una transacción que se realiza con dinero electrónico en un escenario simple es la siguiente:

Supóngase que el usuario **"A"** quiere mandar un cheque a **"B"**, usando ahora dinero electrónico.

1. **"A"** genera un número aleatorio grande **"N"** de digamos 100 dígitos y le da un valor digamos 1000 pesos.
2. **"A"** codifica este número junto a su valor con su clave secreta asimétrica.
3. **"A"** firma este número y lo transmite a su banco.

4. El banco de "**A**" usa, la clave pública de "**A**" para decodificar el número y verificar la firma, así recibe la orden y sabe que es de "**A**". El banco borra la firma de "**A**" del documento electrónico.
5. El banco revisa que "**A**" tenga en sus cuentas la cantidad pedida 1000 pesos y retira esta cantidad de alguna de sus cuentas.
6. El banco firma el número que mandó "**A**", con el valor asignado de 1000 pesos.
7. El banco regresa el número que ya es dinero a, "**A**".
8. "**A**" envía este dinero a "**B**".
9. "**B**" verifica la firma del banco de "**A**", que esta en "**N**".
10. "**B**" envía "**N**" a su banco.
11. El banco de "**B**" vuelve a verificar la firma del banco de "**A**" en "**N**".
12. El banco de "**B**" verifica que "**N**" no este en la lista de números "ya usados".
13. El banco de "**B**" acredita la cantidad de 1000 pesos a la cuenta de "**B**".
14. El banco de "**B**" pone a "**N**" en la lista de números "ya usados".
15. Finalmente, el banco de "**B**" envía un recibo firmado donde establece que tiene 1000 pesos más en su cuenta.

En el mundo comercial existen varias empresas privadas que proveen el servicio de dinero electrónico en diferentes modalidades, entre ellas están: CheckFree, CyberCash, DigiCash, First Virtual, Open Market, NetBill y Netscape.

IV.6.- COMERCIO ELECTRÓNICO

Hoy en día, gran parte de la actividad comercial ha podido transformarse gracias a redes de conexión por computadoras como Internet, ésta transformación facilita hacer transacciones en cualquier momento, de cualquier lugar del mundo. Todo lo que está alrededor de ésta nueva forma de hacer negocios es lo que se ha llamado comercio electrónico[60], sin duda la gran variedad de actividades que giraban alrededor del quehacer comercial se ha tenido que juntar

con las nuevas técnicas cibernéticas. Así hoy, tanto un comerciante, un banquero, un abogado o un matemático puede hablar de comercio electrónico enfocándose a la parte que le corresponde.

Existen diferentes niveles de hacer comercio electrónico, y su clasificación aún está por formarse, sin embargo, la parte más visible es la que cualquier usuario en una computadora personal puede ver, esto es, hacer comercio electrónico se convierte en comprar o vender usando una conexión por Internet en lugar de ir a la tienda. La forma de hacer esto es muy similar a lo que tradicionalmente se hace, por ejemplo: en la tienda uno entra al establecimiento, de forma electrónica se prende la computadora y una vez conectado a Internet entra a la página del negocio, enseguida un comprador revisa los productos que posiblemente compre y los coloca en un carrito, de la misma forma en la computadora se navega por la página del negocio y con el browser se revisa los productos que éste vende, al escoger estos se colocan en un carrito virtual, que no es nada más que un archivo del usuario. Una vez elegido bien los productos de compra se pasa a la caja, donde se elige un sistema de pago y se facturan los productos al comprador. De forma similar, en la computadora se pueden borrar productos que no se quieren comprar o añadir nuevos, una vez elegidos éstos, se procede a una parte de la página que toma los datos y solicita el método de pago, generalmente se lleva a cabo con tarjeta de crédito.

En la parte tradicional de comprar, al pagar en la caja termina el proceso, en la parte por computadora aún tiene que esperarse que sean enviados los productos. A pesar de esto las ventajas que ofrece el comercio electrónico son magníficas, ya que es posible comprar en un relativo corto tiempo una gran cantidad de productos sin necesidad de moverse de lugar, es decir, al mismo tiempo se puede comprar una computadora, un libro, un regalo, una pizza, hacer una transacción bancaria etc., de la forma tradicional se llevaría al menos un día completo y eso si los negocios están en la misma ciudad, si no, el ahorro de tiempo que representa comprar por Internet es incalculable.

Al efectuar una operación comercial por Internet se presentan nuevos problemas, por ejemplo, cómo saber que la tienda virtual existe verdaderamente, una vez hecho el pedido cómo saber que no se cambia la información, cuando se envía el número de tarjeta de crédito, cómo

saber si este permanecerá privado, en fin, para el comerciante también se presentan problemas similares, cómo saber que el cliente es honesto y no envía información falsa, etc. Todos estos problemas pueden ser resueltos de manera satisfactoria sí se implementan protocolos de comunicación segura usando criptografía.

En el siguiente capítulo se lleva a cabo la evaluación tecnológica de los algoritmos de codificación presentados en los capítulos anteriores, así mismo se establecen una serie de criterios a evaluar y la importancia de los mismos para la selección de un sistema seguro.

Saber leer es saber andar.
Saber escribir es saber ascender.
-José Martí

CAPITULO V: EVALUACIÓN TECNOLÓGICA

V.1.- INTRODUCCIÓN

Como se mencionó con anterioridad, la evaluación tecnológica requiere tanto del conocimiento del proceso como de la tecnología, para así, poder hacer una caracterización y clasificación de las diversas tecnologías disponibles. Hasta este momento, se han presentado las características inherentes para algunos de los desarrollos informáticos existentes en el mercado para la protección de la información por medio de un canal inseguro. Ahora se evaluarán todas y cada una de las tecnologías descritas en los capítulos anteriores, con el fin de presentar al final un conjunto de las mejores y más viables opciones.

Las características a tomarse en cuenta son:

- Antigüedad del algoritmo.
- Tamaño de bloque codificado.
- Tamaño de clave.
- Número de ciclos iterativos.
- Tipo de implementación.
- Nivel de seguridad.
- Distribución o aplicación entre usuarios.

Es posible que existan algunas otras características que pudiesen ser tomadas en cuenta, pero cabe recordar que una de las finalidades de esta tesis es proporcionar una selección rápida, objetiva y de fácil entendimiento tanto para un usuario experto en el tema como para un principiante.

V.2.- ANTIGÜEDAD DEL ALGORITMO

Esta característica es importante, en el sentido de que si utilizamos un algoritmo cuya creación no es muy reciente es de esperarse que pudiesen haberse desarrollado algún método exitoso para su decodificación, por lo cual se pone en duda su nivel de protección a menos de que pudiese ser avalado por la experiencia de diversos usuarios. Por otra parte, la combinación de antigüedad, seguridad y vigencia nos puede servir como parámetro para determinar la distribución del algoritmo dentro del círculo de usuarios asiduos a la utilización de estos sistemas, por lo que, en el caso de querer transmitir información con usuarios desconocidos (ej. comercio electrónico) puede ser muy útil.

V.3.- TAMAÑO DE BLOQUE CODIFICADO

Dentro de las características más importantes que un algoritmo criptográfico puede contener, se encuentra la velocidad de codificación, esto debido a que los procesos iterativos o de cálculo que se requieren para que un algoritmo pueda codificar un mensaje, son en ocasiones algo extensos.

Una de las características de las cuales depende que un algoritmo sea más rápido que otro es determinada por el tamaño de bloque codificado, esto se explica de la siguiente forma. Cuando se desea codificar un texto cuyo tamaño es relativamente extenso, un algoritmo que utilice un bloque de codificación relativamente pequeño tardará más tiempo en llevar a cabo todo el proceso de codificación en comparación con uno que utilice un tamaño de bloque mayor extensión.

V.4.- TAMAÑO DE CLAVE

Esta es sin duda una de las características primordiales para que un algoritmo sea catalogado como seguro o no, debido a que la clave de codificación es considerada como la puerta principal y el cerrojo más importante del cual depende la protección proporcionada a un mensaje.

Muchos de los algoritmos que se han desarrollado cuentan con una longitud de clave fija, tal como el DES, el Skipjack, Feal o IDEA, entre otros; para los cuales va desde los 64 a los 128 bits dependiendo de cada algoritmo. Sin embargo, en los últimos años, el desarrollo de dichos instrumentos de seguridad se ha enfocado hacia el desarrollo de tecnologías que utilizan un tamaño de clave variable, como ejemplo de esto, tenemos el algoritmo simétrico Akelarre cuyo tamaño de clave, bloque de texto y ciclos iterativos es variable, dando oportunidad a seleccionar la seguridad con la cual se protegerá un mensaje, así mismo se controla de manera indirecta la velocidad de codificación (a más ciclos iterativos menor velocidad).

En algunos sistemas de codificación se utiliza la multiplicación de grandes números primos para la selección de claves, esto se hace bajo la premisa de la dificultad que implica la factorización de un número muy grande. Muestra de lo anterior, son los algoritmos asimétricos los cuales utilizan números primos de al menos 100 bits.

Hasta hace algunos años, se creía que se necesitarían algunos cientos de años para poder factorizar números grandes con los sistemas y tecnologías existentes, o bien en encontrar algún método o sistema capaz de factorizar este tipo de números; en 1994, se logró la factorización de un número de 129 dígitos en un lapso de 8 meses utilizando el tiempo muerto de 1,600 computadoras conectadas a internet por todo el mundo. Esto nos demuestra lo arriesgado que es hacer predicciones en cuanto a la seguridad que puede o no representar el tamaño de una clave.

A pesar de lo anterior, día con día se hacen predicciones acerca del tamaño de clave recomendado, esto se debe principalmente a que cuando se utiliza un criptosistema, es con el fin de asegurar la información durante un periodo de tiempo. Dicho periodo de tiempo puede variar de unos minutos al transferir información a través de un canal público, a varios meses o incluso años al guardar la clave de un banco para un sistema de dinero digital o la firma digital de un notario en un documento electrónico. Por tal motivo, una clave de 1024 bits puede ser lo bastante grande como para firmar algo que pudiera ser verificado dentro de un mes o una semana, o incluso dentro de algunos cuantos años, pero puede no ser muy seguro para firmar algo que pudiera ser verificado dentro de algunos lustros.

Por tal razón, dentro de las diversas propuestas de tamaño de clave segura hechas por algunos expertos, se encuentra la de Bruce Schneier en donde propone diversas longitudes de claves dependiendo de contra quién se desee proteger la información: contra un individuo curioso y perseverante (I), contra una corporación (C) o una contra un gobierno (G).

AÑO	CONTRA I	CONTRA C	CONTRA G
1995	768	1280	1536
2000	124	1280	1536
2005	1280	1536	2048
2010	1280	1536	2048
2015	1536	2048	2048

TABLA 5.1- TAMAÑO DE CLAVE RECOMENDADO EN BITS.

Para fines prácticos, de lo anterior de concluye lo siguiente:

La longitud de clave seleccionada dependerá tanto de la clase de oponente contra el cual se desee proteger la información como del tiempo que se desee guardar ésta, aunado al tiempo que esta esté expuesta a un oponente.

La utilización de una clave de gran longitud disminuye la velocidad de codificación de un texto.

V.5.- NÚMERO DE CICLOS ITERATIVOS.

Dentro de las diversas características que un sistema criptográfico comercial ofrece, se encuentra la del número de ciclos iterativos; ésta se puede traducir como el número de veces en la cual se calculan los símbolos sustitutos del mensaje original, por lo tanto, a un mayor número de ciclos iterativos, más ilegible será la información original y por tanto será más difícil su decodificación sin el uso de una clave. Pero como es de esperarse, entre más ciclos iterativos se requieran para codificar un mensaje, más tiempo se requerirá para llevar a cabo toda la operación.

Ahora bien, para todos los algoritmos presentados dentro de esta tesis, se puede aplicar una clasificación la cual se basa en esta característica:

1. Algoritmos de un sólo ciclo.
2. Algoritmos iterativos:
   - Algoritmos de ciclos iterativos fijos
   - Algoritmos de ciclos iterativos variables

Dentro del primer grupo se incluyen todos los algoritmos asimétricos, los cuales se basan en la aplicación de una ecuación con 3 entradas o claves para la codificación de un mensaje. De estas 3 claves, una de ellas es común, tanto para el emisor como para el receptor, las otras 2 restantes, se distribuyen de la siguiente manera, una para el emisor y la otra para el receptor.

Dentro de la segunda categoría, se encuentran todos los algoritmos del tipo simétrico ya que estos utilizan una serie de cálculos repetitivos para la codificación de un mensaje. Ahora bien, esta clasificación se ha seccionado en 2 subcategorías, ya que dentro de ésta clase de algoritmos, se encuentran los que tienen un número de iteraciones ya definidas, tal es el caso, del DES con 16 ciclos o el IDEA con 8 ciclos, pero por otra parte, ha surgido una nueva alternativa y esta es el AKELARRE el cual cuenta con un número de ciclos iterativos variables los cuales pueden ser definidos por el usuario.

V.6.- TIPO DE IMPLEMENTACIÓN

La implementación de los algoritmos criptográficos comerciales se puede dividir en 2 clases: software y hardware. En el primer caso, se tiene que el programa encargado de llevar a cabo el proceso de codificación se encontrará contenido en un archivo ejecutable, el cual puede ser transmitido por un usuario a otro por medio de una transmisión vía módem o por el intercambio de algún medio de almacenamiento temporal (memoria EPROM).

En el caso de la implementación tipo Hardware, nos referimos a que la distribución y almacenamiento del programa se lleva a cabo desde un medio físico implementado dentro del propio CPU, tal es el caso de una tarjeta especial o una interfase interna o externa, las cuales pueden contener tanto el programa como la memoria necesaria para que se pueda ejecutar.

La diferencia entre una y otra implementación, van más allá de una simple tarjeta que se pueda adquirir en una tienda electrónica o una adquisición electrónica pirata obtenida por medio de internet, para el caso de una implementación hardware, es casi seguro el pago de una licencia inherente al uso del programa, pero además de esto, se tiene la certeza de que el programa es original y no un posible troyano o una versión incompleta del programa que se desea, además de lo anterior, una versión hardware casi por lo general incluye los recursos que el programa necesitará tal como la cantidad de memoria necesaria para poder ejecutar el programa, además de incluir candados electrónicos de seguridad inviolables, los cuales aseguran (según sus creadores) que ningún hacker, cracker u otro familiar suyo pueda extraer la clave de codificación del sistema interno de la computadora del usuario, lo cual no es asegurado al cien por ciento por una implementación vía software (aunque ciertamente no ha habido nadie que haya probado, hasta la fecha, haber realizado una decodificación utilizando este sistema pero lo cual no asegura que esto sea imposible).

Ahora bien, en la comparación del desempeño de una implementación hardware contra una de software, por lo general, una versión hardware resulta ser mucho más rápida y esto se

debe en gran medida a lo que anteriormente se mencionó, la inclusión de los recursos para poder ejecutar el programa dentro de la tarjeta.

V.7.- NIVEL DE SEGURIDAD

Este es uno de los aspectos más importantes y al mismo tiempo delicados durante la selección de un paquete destinado a mantener la seguridad de un mensaje durante la transmisión a través de un canal seguro.

La calidad de seguridad con la cual se etiqueta a un paquete puede ser obtenida ya sea después de un extenso, concienzudo y complejo análisis matemático practicado tanto a la clave del algoritmo como al mismo algoritmo de codificación. También puede ser calificado en base a los intentos que varios expertos alrededor del mundo han hecho al algoritmo para tratar de quebrantar su seguridad.

En vista que este trabajo pretende ser una guía rápida y entendible para la selección de un algoritmo seguro, se ha desechado la primer opción.

V.8.- DISTRIBUCIÓN O APLICACIÓN ENTRE LOS USUARIOS

Para algunos usuarios, esta es una característica importante que todo algoritmo debe tener, tal es el caso de los asiduos al comercio electrónico, al cual en nuestros días se ha vuelto muy difundido gracias al internet. Este es un caso muy claro de la necesidad de emplear un algoritmo para codificar y esconder cierto tipo de información, tal como el número de tarjeta de crédito de un usuario o incluso el pedido solicitado, de cualquier escucha mal intencionado que quisiera dañar ya sea al cliente o al vendedor, ya sea de forma intencional o simplemente por diversión.

En este caso como en otros más, el empleo de un canal inseguro para transmitir información clasificada o selecta, implica que tanto el emisor como el receptor, deberán poseer

el mismo tipo de codificador, en ciertos casos, incluso es necesario que posean no solo el mismo programa sino la misma versión y plataforma, en este caso, estos algoritmos se vuelven algo inflexibles.

Ahora bien, como en muchos casos, si todos tienen el conocimiento sobre algo, esto implica que esta información ya no es selecta, este caso es aplicable a los algoritmos informáticos, esto es, al estar más distribuido el algoritmo implica que más personas lo conocen, por tanto es probable que algunas de ellas hayan intentado hacer un algoritmo para poder romper la clave o simplemente saltársela y decodificar automáticamente el mensaje codificado haciendo un análisis concienzudo del algoritmo.

De todo lo anterior podemos resumir que; para casos de distribución entre una gran cantidad de usuarios, incluyendo contactos de primera vez, es importante que el algoritmo que se vaya a emplear esté ampliamente difundido entre los usuarios o bien, sea relativamente fácil de obtener, lo cual no deberá dejar de lado la prioridad de la seguridad. Para el caso de contactos frecuentes con ciertos usuarios es conveniente utilizar en ocasiones algoritmos poco conocidos ya que en caso de que el mensaje quisiera ser decodificado por un tercero, implicaría dos problemas: el primero, sería determinar la llave del algoritmo usado, y el segundo determinar el algoritmo utilizado.

Cabe señalar que la protección que se de a un mensaje dependerá de la importancia que este tenga y de los posibles recursos, que en su caso, se podrían emplear para decodificarlo.

V.9.- RESULTADOS

Como resultado de la evaluación tecnológica de los algoritmos analizados en la presente tesis con respecto a los parámetros seleccionados para su comparación en este capítulo tenemos que:

TABLA COMPARATIVA DE CARACTERÍSTICAS DE LOS DIVERSOS ALGORITMOS CRIPTOGRÁFICOS

ALGORITMO	AÑO	TAMAÑO DE BLOQUE	TAMAÑO DE CLAVE	Nº DE CICLOS ITERATIVOS	IMPLEMENTACIÓN		ALGORITMO PÚBLICO	ACTUAL VIGENCIA	SEGURIDAD	APLICACIÓN ENTRE USUARIOS	APLICACIÓN
					SOFTWARE	HARDWARE					
LUCIFER	1970's	128 bits	72 bits	16	X		SI	NO	*	+	S/A
DES	1977	64 bits	64 bits	16	X	X	NO	SI	****	+++++	G. P. Pu.
GOST	1989	64 bits	256 bits	32	X		NO	SI	*****	++++	G. P. Pu.
LOKI	1997	64 bits	128 - 192	16	X		?	SI	*****	+++	?
SKIPJACK	1990	64 bits	80 bits	32		X	NO	SI	*****	+	S/A
IDEA	1991	64 bits	128 bits	8	X	X	NO	NO	*****	+++++	G. P. Pu.
BLOWFISH	1993	64 bits	32 - 448	16	X		SI	SI	*****	++++	Pu.
CAST-128	1996	64 bits	40 - 128	16	X		SI	SI	*****	+++	Pu.
SAFER-64	1996	64 bits	64 bits	8	X		SI	SI	****	++++	G. P.
AKELARRE	1996	variable	variable	variable	X		?	SI	***	++	Pu.
FEAL	1990's	64 bits	64 bits	32	X		SI	SI	**	++	Pu.
RIJNDAEL	1998	128, 192 o 256	128, 192 o 256	9,11 o 13	X		SI	SI	*****	+++	G.
RSA	1977	variable	≤ 100 bits	1	X		NO	SI	*****	+++++	P. Pu.
MERKLE HELLMAN	1978	variable	≤ 100 bits	1	X		SI	NO	**	+	S/A
McELICE	1978	variable	≤ 100 bits	1	X		SI	NO	**	+++	S/A
RABIN	1979	variable	≤ 100 bits	1	X		SI	NO	*****	+	S/A
WILLIAMS	1980	variable	≈ 100 bits	1	X		SI	NO	***	++	S/A
EL GAMAL	1985	variable	≤ 100 bits	1	X		SI	SI	*****	+++	?

* Muy bajo (Sensible a criptoanálisis diferencial, muy estudiado).
** Bajo (Se han creado algoritmos eficientes para atacar a estos sistemas o bien se duda seriamente sobre su seguridad).
*** Regular (Algoritmo con debilidades estructurales que lo hacen vulnerable).
**** Alto (Débiles en ciertas condiciones de operación).
***** Muy alto (Son resistentes a todo tipo de ataques o bien se pueden quebrantar con mucha dificultad y varios recursos).

? Se desconoce si se aplica en la actualidad por algún grupo de usuarios.
S/A Sin aplicaciones actuales.
P. De aplicación privada o corporativa para lo cual es necesario el uso de licencias.
Pu. Pública
G. Gubernamental.

\+ Muy poco difundida entre los usuarios debido a su desuso, antiguedad o a su relativa inseguridad o algún defecto técnico.
++ Poca difusión, debido al desconocimiento popular acrecentado por su inseguridad.
+++ Regular difusión debido al desconocimiento popular
++++ Buena difusión entre ciertos grupos de usuarios.
+++++ Excelente difusión ganada por su resistencia a todo tipo de ataques.

Si hay rectitud en el corazón,
habrá belleza en el carácter.
Si hay belleza en el carácter,
habrá armonía en el hogar y orden en la nación.
Cuando hay orden en la nación,
habrá paz en el mundo.

-Proverbio Chino

CAPITULO VI: ANALISIS DE RESULTADOS Y CONCLUSIONES

VI.1.- ANALISIS DE RESULTADOS

De los resultados obtenidos en la presente tesis, se puede observar que dentro de las dos grandes clasificaciones de algoritmos existentes, los que han sido creados en mayor número son los del tipo simétrico, esto es debido en parte, a la sencillez que implica la utilización de ciclos repetitivos ya determinados, en comparación a los algoritmos asimétricos ya que estos últimos implican el uso de ecuaciones matemáticas complejas.

En una comparación entre la seguridad proporcionada por los algoritmos simétricos y asimétricos, podemos observar que el porcentaje de algoritmos asimétricos que han demostrado ser seguros es mayor al de los algoritmos simétricos analizados. Cabe recalcar que, para el presente estudio se buscaron no sólo los algoritmos más conocidos, si no también se buscaron todos aquellos algoritmos de los cuales se tuviese suficiente información tanto sobre su funcionamiento como sobre ataques sufridos al mismo, esto con el fin de tener suficientes datos con los cuales poder evaluar su seguridad, por lo cual se puede aseverar que el número de algoritmos presentados, no tienen una tendencia forzada hacia la presentación de más algoritmos simétricos.

Por otra parte, se puede observar que los algoritmos simétricos utilizan en general un tamaño de bloque establecido para la codificación de datos, esto es por la estructura de este tipo de algoritmos, ya que en ésta, se establece el número bytes involucrados en la codificación en el inicio del algoritmo, por tal motivo estos algoritmos tienen una estructura rígida en la codificación. En comparación para los algoritmos asimétricos el tamaño del bloque de codificación puede ser variado por el usuario mediante la modificación o bien la definición de ciertos parámetros iniciales, aunque también se puede observar que ésta tendencia tiende a cambiar debido a las necesidades actuales.

Podemos observar, que la implementación de la mayoría de los algoritmos es por medio de software, esto no es sorprendente debido a que la transmisión y comercialización vía electrónica de un paquete de este tipo siempre será más sencilla que la de un hardware.

La determinación de un algoritmo como público o privado está en función a su libre utilización por cualquier tipo de usuarios sin el previo permiso o licencia expedida por la compañía distribuidora del algoritmo, en este campo se observa que la mayoría de los algoritmos son de libre utilización, sin embargo, existe un grupo de algoritmos que requiere de licencia para su utilización, en este sentido, esto se debe a que este grupo de algoritmos gozan de un prestigio en cuanto al nivel de protección que proporcionan, aunado a que poseen un grupo de clientes cautivos, en comparación, el resto de algoritmos han buscado captar un mayor número de usuarios, además de demostrar su efectividad antes de poner precio a su utilización.

VI.2.- CONCLUSIONES

De la presente tesis se puede concluir que existe tan sólo un grupo pequeño de algoritmos que han demostrado ser seguros en la práctica, así mismo, debido a que la tecnología día a día avanza en una dirección de complejidad y aumento en velocidad de tiempo real de procesamiento de datos, es muy difícil predecir cuanto tiempo más podrán

durar estos algoritmos en este estatus, lo cierto es que para cada necesidad existe un producto en el mercado.

Por otra parte, podemos argumentar que el desarrollo de un algoritmo asimétrico es más complicado que el desarrollo de un algoritmo simétrico, esto se debe a que es más sencillo el desarrollo de un programa utilizando ciclos iterativos, cajas de permutación o intercambio o la eventual utilización de comandos AND, OR o NOT y que implique una codificación de una entrada y una salida al desarrollo de una ecuación capaz de utilizar dos números completamente diferentes para codificar o decodificar un grupo de datos de forma segura y eficiente.

También se evaluó la seguridad de los diversos algoritmos haciendo un análisis de las experiencias obtenidas en campo, en los algoritmos simétricos los que mayor seguridad ofrecen en la actualidad son el DES, GOST, IDEA, BLOWFISH, LOK, CAST-128 y RIJNDAEL, podemos decir que el DES aún es vigente, ya que aún no se publican las condiciones de operación del nuevo estándar de seguridad AES, por lo cuál se podrá seguir utilizando el DES en sus implementaciones compuestas, tales como el triple DES, ECB, CBC, CFB y OFB ya que en su versión sencilla se ha vuelto vulnerable. En cuanto a los algoritmos de tipo asimétrico los que han demostrado su seguridad son el RSA, el algoritmo de EL GAMAL y el de RABIN. No obstante, la seguridad y gama de opciones proporcionadas por programas de aplicación, tales como PGP aunado a servicios de verificación de autenticidad de firmas digitales o protocolos de transmisión de datos, aumenta la seguridad y certidumbre en la transmisión de información.

En la actualidad muchos de estos algoritmos son utilizados dentro de páginas comerciales en internet, para las cuales se solicita desde un número de tarjeta de crédito hasta la simple clave de acceso de un correo electrónico y por lo que podemos observar la utilización de estos aumentará con el paso del tiempo, por lo cuál es importante conocer los sistemas que protegen nuestra información y que se encuentran disfrazados detrás de un programa de aplicación.

La seguridad de un algoritmo se puede determinar mediante un estudio complejo de la estructura del mismo, pero por otra parte si se realiza un estudio concienzudo mediante la recopilación y análisis de la experiencia obtenida en la implementación de las diversas opciones en seguridad que existen puede dar un resultado satisfactorio, este es el objetivo principal de la presente tesis y de lo cual se puede observar que se ha obtenido un magnifico resultado.

GLOSARIO

Aplicación: Programa que lleva a cabo una función determinada.

Autentificación: Verificación de la identidad de una persona o de un proceso para acceder a un recurso o para poder realizar determinada actividad.

Bacteria: Programa que consume los recursos del sistema mediante la replicación de sí mismo.

Bit: Unidad mínima de información digital que puede ser tratada por una computadora. Proviene de la contracción de la expresión *binary digit* (digito binario).

Bombas lógicas: Es un programa que ejecuta una serie de órdenes no autorizadas por el usuario cuando se cumplen una serie de condiciones de operación.

Browser (navegador, visor, visualizador): Aplicación para visualizar documentos WWW y navegar en el espacio de Internet.

Bucaneros: Término utilizado para denominar a piratas informáticos que comercian con programas informáticos crackeados. Un bucanero es simplemente un comerciante.

Bug (error, insecto): Término aplicado a los errores descubiertos al ejecutar un programa informático. Este término fue utilizado por primera vez en 1945 por Grace Murria Hooper, una de las pioneras de la programación moderna, al descubrir como un insecto (*bug*) había dañado un circuito de la computadora Mark.

Byte: Conjunto significativo de ocho bites que representan un carácter.

Caballos de Troya: Rutina secreta no registrada presente dentro de un programa útil. La ejecución del programa resulta en la ejecución de la rutina.

Chip: Circuito integrado en un soporte de silicio, formado por transistores y otros elementos electrónicos miniaturizados. Son uno de los elementos esenciales de una computadora.

Ciberespacio: Término utilizado por William Gibson en su novela de ciencia ficción "Neuromancer" para describir el "mundo" de las computadoras y la sociedad creada en torno a ellos.

Cibernauta: Persona que navega por la red.

Clave: Código de signos empleados para codificar un mensaje por medio de algún algoritmo criptoráfico.

Codificación asimétrica: Sistema de codificación en el cuál se tienen dos claves, una privada y una pública. Una de ambas claves se puede seleccionar para codificar un mensaje y la otra deberá de decodificarlo.

Codificación simétrica: Sistema de codificación mediante el cuál se tiene una sola clave para poder codificar y decodificar un mensaje. En este caso la clave debe ser secreta.

Codificación: Es el tratamiento de un conjunto de datos, contenidos o no en un paquete, a fin de impedir que nadie excepto el destinatario de los mismos puedan leerlos.

Cookie: Conjunto de caracteres que se almacenan en el disco duro o en la memoria temporal de la computadora cuando un usuario accede a ciertas páginas de los sitios web y que sirven para que el servidor accedido pueda conocer las preferencias del usuario.

Copyhackers: Personas con alto conocimiento en tecnología las cuales se especializan en el crackeo de Hardware.

Crackers: Término utilizado para denominar a las personas que se dedican a romper sistemas de seguridad dentro de redes de cómputo y software. Un cracker conoce perfectamente las dos caras de la tecnología, esto es, la parte física de la electrónica y la parte de programación.

Criptografía: Término proveniente del griego *criptos,* oculto y *grafos,* escritura y significa el arte de escribir la información con ayuda de una clave secreta.

Criptología: Del griego *criptos,* oculto y *logos,* estudio y es la parte de la criptografía encargada de descifrar criptogramas cuando se ignora la clave.

Feistel, algoritmos tipo: Algoritmos en los cuáles se divide el bloque de texto original en dos mitades y se comienza a trabajar en una de ellas para poder codificar todo el bloque.

Firewall (cortafuegos): Sistema que se coloca entre una red local e Internet. La regla básica es asegurar que todas las comunicaciones entre dicha red e internet se realicen conforme a las políticas de seguridad de la organización que lo instala. Además, estos sistemas suelen incorporar elementos de privacidad, autentificación, etc..

Firma digital: Información codificada que identifica al autor de un documento electrónico y autentifica quien dice ser.

Gusanos: Programa que puede replicarse a sí mismo y enviar copias de computadora a computadora a través de la red de trabajo. Una vez dentro, el gusano puede ser activado, replicarse y propagarse nuevamente. Adicionalmente a la propagación, el gusano lleva a cabo la ejecución de rutinas no deseadas.

Hackers: Término empleado para denominar a personas expertas en sistemas de computación avanzados. Dominan la programación y la electrónica con el fin de comprender el funcionamiento de sistemas complejos tales como los de comunicación móvil. Detectan fallas en programas y sistemas de computo y en ocasiones advierten al creador de los mismos para que pueda protegerse. Extraen información de la red más por diversión o curiosidad, que por comercio o por daño.

Hash, función: Función mediante la cuál se puede crear un mensaje resumido de un documento completo.

Host: Computadora que mediante la utilización de los protocolos TCP/IP, permite a los usuarios comunicarse con otros sistemas anfitriones de la red.

IP, dirección: Dirección de 32 bits definida por el protocolo de internet en STD 5, RFC791. Se representa usualmente mediante una notación decimal separada por puntos. Un ejemplo de esta es 193.127.88.345

Keyword (palabra clave): Conjunto de caracteres que puede utilizarse para buscar información en un buscador o en un sitio web.

Pasword (contraseña): Conjunto de caracteres que permiten a un usuario el acceso a un determinado recurso o a la utilización de un servicio dado.

Puertas traseras: Es un punto de entrada no registrado a un programa, el cual es utilizado para otorgar acceso al mismo sin la necesidad de los métodos normales de autentificación de acceso.

Virus: Código incluido dentro de un programa el cuál tiene como resultado la copia de sí mismo para ser insertado dentro de uno o más programas. Adicionalmente, a ésta propagación, este programa nocivo lleva a cabo la ejecución de ciertas rutinas no deseadas.

BIBLIOGRAFÍA

[1] Fuster Sábater, A. Técnicas criptográficas de protección de datos, Madrid España, Ra-ma, 1997. pp. 11-29.

[2] Pons Martorell, M. Criptología, Barcelona España, Escuela Universitaria Politécnica de Mataró, Departamento de telecomunicaciones, 2000. pp. 11–15.

[3] Angel, J.J Criptografía para principiantes, España, Kriptopolis, 2001. pp. 5-8.

[4] Cassaigne, R. Notas del curso de Planeación tecnológica. Facultad de Química, México, 1998.

[5] Pastor Franco, J. y Sarasa López, M.A. Criptografía digital: Fundamentos y aplicaciones, Zaragoza España, 1ª Prensas Universitarias de Zaragoza, 1998. pp. 78-108.

[6] Schneier, B. Applied Cryptography, John Wiley & Sons, Inc. 1996. pp. 45-60.

[7] Savard, J. "LUCIFER: The first block cipher", en Notas del taller de criptografía de la Universidad de Granada, España, 1998.
http://www.ugr.es/~aquiran/cripto/protocol/lucifer.htm

[8] "Specifications for the Data Encryption Standard", en Federal Information Processing Standards, EUA, 15 de Enero de 1977, Pub. 46. pp. 64-74.

[9] Matsui, M. Linear cryptanalysis method for DES cipher, Advances in Cryptology EUROCRYPT'93, LNCS 765. pp. 386-397, 1994

[10] Rogaway P., The Security of DESX, CryptoBytes Vol 2, No 2. EUA,1996. pp. 8-10.

[11] Pastor Franco, J. y Sarasa López, M.A. Criptografía digital: Fundamentos y aplicaciones, Zaragoza España, 1ª Prensas Universitarias de Zaragoza, 1998. pp. 110-111.

[12] Suárez Martínez, J. "Skipjack: Toda la verdad" en Tribuna de Kriptópolis, España, Septiembre 1998. http://www.kriptopolis.com/trib004.html

[13] Ascom Systec Ltd. Página informativa de ascom, Suiza, 1999. http://www.ascom.ch

[14] Fuster Sábater, A. Técnicas criptográficas de protección de datos, Madrid España, Ra-ma, 1997. pp. 65-68.

[15] Mazzara, P. "The IDEA algorithm" en Instututo de Ingeniería Eléctrica, Montevideo Uruguay, 1998. http://www.iie.edu/~mazzara/pgp/idea.htm

[16] Quirantes Sierra, A. "IDEA" en Notas del taller de criptografía de la Universidad de Granada,Granada España, 7 de noviembre de 1997. http://www.ugr.es/~aquiran/cripto/protocol/idea.htm

[17] Schneier, B."The Blowfish Encryption Algorithm" en Página de Counterpane Systems, Grán Bretaña, 12 de Noviembre de 1998. http://www.counterpane.com/blowfish.html

[18] Schneier, B."The Blowfish Encryption Algorithm - One Year Later" en Dr. Dobb's Journal, Grán Bretaña, Septiembre de 1995. http://www.counterpane.com/blowfish.html

[19] Schneier, B. "Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)" en Notas del taller de criptografía de la Universidad de Granada, España, 9 de Agosto de 1997. http://www.ugr.es/~aquiran/cripto/protocol/blowfish.txt

[20] Adams, C. "The CAST-128 Encryption Algorithm" en Notas del taller de criptografía de la Universidad de Granada, España, Mayo de 1997, http://www.ugr.es/~aquiran/cripto/protocol/cast.txt

[21] Pastor Franco, José y Sarasa López, Miguel A. Criptografía digital: Fundamentos y aplicaciones, Zaragoza España, 1ª Prensas Universitarias de Zaragoza, 1998. pp. 111-113.

[22] Savard, J.J. A Criptographic Compendium, E.U.A., Agosto 2001, http://www.cylink.com/internet/library.nsf/pages/SAFER/.

[23] Pastor Franco, J. y Sarasa López, M.A. Criptografía digital: Fundamentos y aplicaciones, Zaragoza España, 1ª Prensas Universitarias de Zaragoza, 1998. pp. 117-121.

[24] Ferguson, N. y Schneier B. "Cryptanalysis of Akelarre" en Página de Counterpane Systems, Grán Bretaña, Agosto de 1997, http://www.counterpane.com/akelarre.html

[25] Pastor Franco, J. y Sarasa López, M.A. Criptografía digital: Fundamentos y aplicaciones, Zaragoza España, 1ª Prensas Universitarias de Zaragoza, 1998. pp. 121-123.

[26] Savard, John. "LOKI97" en Notas del taller de criptografía de la Universidad de Granada, España, 1998. http://www.ugr.es/~aquiran/cripto/protocol/loki97.htm

[27] Lucena, M.J. "DES ha muerto... ¡larga vida a Rijndael" en Kriptopolis, España, 18 octubre 2000. http://www.kriptopolis.com/luc/20001018.html

[28] Savard, J.J. A Criptographic Compendium, E.U.A., Agosto 2001, http://home.ecn.ab.ca/~jsavard/crypto/co040801htm

[29] Daemen, J., Rijmen, V. The Rijndael Block Cipher, EUA, 3 Septiembre 1999.

[30] Knudsen, L.R, Rijmen, V. The Block Cipher Lounge-AES, EUA, Agosto 2001, http://www.ii.uib.no/~larsr/aes.html

[31] Gómez, J.M., "AES práctico" en Kriptopolis, España, 18 octubre 2000, http://kriptopolis.com/jmg/20001018.html

[32] Schneier, B. "Novedades sobre AES" en Kriptopolis, España, 9 mayo 2001. http://www.kriptopolis.com/criptograma/0024_1.html

[33] Caballero Gil, P. Técnicas criptográficas, Madrid España, Ra-ma, 1996. pp. 57-68.

[34] Fúster Sábater, A. De la Guia Martínez, Dolores. Técnicas Criptográficas de protección de datos, Madrid España, Ra-ma, 1997. pp. 122-125.

[35] Nombela, J. Seguridad Informática, Madrid España, Paraninfo,1997. pp. 130-140.

[36] Goldberg, Jeffrey. "The Feasibility of Breaking PGP" en Cranfield Computer Centre, Inglaterra, Febrero de 1996. http://www.cranfielkl.ac.uk/docs/email/pgp/pgp-attack-faq.txt

[37] Ylönen, Tatu. "Cryptographic Algorithms" en SSH Comunications Security, E.U.A, 1998. http://www.cs.hut.fi/ssh/crypto/algorithms.html#RSA

[38] Osoria. "RSA en PGP" en Página de usuarios de INTERCOM, España, 1996. http://usuarios.intercom.es/eugeni/pgp/foro1.htm

[39] Caballero Gil, P. Técnicas criptográficas, Madrid España, Ra-ma, 1996. pp. 73-85.

[40] Pastor Franco, J. y Sarasa López, M. A. Criptografía digital: Fundamentos y aplicaciones, Zaragoza España, 1ª Prensas Universitarias de Zaragoza, 1998. p.p. 195-196.

[41] Caballero Gil, P. Técnicas criptográficas, Madrid España, Ra-ma, 1996. p.p. 71-73.

[42] Caballero Gil, P. Técnicas criptográficas, Madrid España, Ra-ma, 1996.p.p. 106-108.

[43] Fúster Sábater, A. De la Guia Martínez, Dolores. Técnicas Criptográficas de protección de datos, Madrid España, Ra-ma, 1997. p.p. 141-152.

[44] Nombela, J. Seguridad Informática, Madrid España, Paraninfo,1997. p.p. 130-143.

[45] López, Lourdes y Portillo, Eloy. "Seguridad en redes telemáticas (1ª Parte)", RedIRIS Versión electrónica, Madrid, España, Junio 1998.
http://www.rediris.es/rediris/boletin/31/enfoque1.html

[46] Wayner, P. "Who goes there?", en Byte, EUA, Junio 1997, vol. 22, núm. 6, p.p. 70-80.

[47] VeriSign, "Información de servicios de certificación" Página informativa de VeriSign, Octubre 1998, http://www.verisign.com/

[48] Rainer Mauth. "Digital Signatures to Power E-Commerce", en Byte sección internacional. EUA, Enero 1998, vol. 23, núm. 1, pp. 5-10.

[49] Rivas, X. Asertel, "Las primeras experiencias legislativas" en Area de Servicios Telemáticos asertel, Enero 1997, http://www.asertel.es/cs/ 06041001.htm.

[50] Rivas, X. Asertel, "Ley alemana sobre firma digital" en Area de Servicios Telemáticos asertel, Octubre 1996, http://www.asertel.es/cs/ 06041005.htm

[51] Rivas, X. Asertel, "Comunicación de la CE sobre firma digital" en Area de Servicios Telemáticos asertel, Octubre 1997, http://www.asertel.es/cs/ 06041006.htm

[52] Angel, J.J. Criptografía para principiantes, España, Escuela Universitaria Politécnica de Mataró, Departamento de telecomunicaciones, 2000. pp.37-43

[53] Pons Martorell, M. Criptología, Barcelona España, Escuela Universitaria Politécnica de Mataró, Departamento de telecomunicaciones, 2000.pp. 38–40.

[54] Back, Adam. "Historia de PGP" en Notas del taller de criptografía de la Universidad de Granada, España. http://www.ugr.es/~aquiran/cripto/expedien002.htm

[55] Goldberg, Jeffrey. "The Feasibility of Breaking PGP" en Página de MIT para la distribución de PGP, EUA, Febrero de 1996.
http://www.cranfielkl.ac.uk/docs/email/pgp/pgp-attack-faq.txt

[56] McNamara, Joel. "Practical Attacks on PGP" en Notas del taller de criptografía de la Universidad de Granada, España, 9 de Agosto de 1997.
http://www.ugr.es/~aquiran/cripto/expedien004.htm

[57] Sanz de las Heras, Jesus y Martínez, Ruben. "Seguridad en Correo Electronico:PGP" en RedIRIS Versión electrónica, Madrid, España, Noviembre 1995.
http://www.rediris.es/rediris/mail/coord/seguridad/seguridad.html.

[58] Schiller, Jeffrey I.. "Pretty Good Privacy" en Página del MIT para la distribución de PGP, EUA, 3 de Diciembre de 1997. http://web.mit.edu/network/pgp.html

[59] Zimmermann, Phil. "PGP user's guide" en MIT Distribution Center for PGP, U.S.A. 2001. http://web.mit.edu/network/pgp.html

[60] Angel, J.J. Criptografía para principiantes, España, Escuela Universitaria Politécnica de Mataró, Departamento de telecomunicaciones, 2000. pp.35-37.