ANÁLISIS COMPARATIVO A NIVEL TEÓRICO Y PRÁCTICO DE LOS ALGORITMOS CRIPTOGRÁFICOS ORIENTADO HACIA LA SOLUCIÓN DEL PROBLEMA EN LA SEGURIDAD DEL TRANSPORTE DE DOCUMENTOS DIGITALES.

CARLOS ANDRÉS FERNÁNDEZ ZAMBRANO

Trabajo Final presentado como requisito para optar por el título de Ingeniero en Electrónica y Telecomunicaciones

Director: Ing. Siler Amador Donado

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Popayán, Noviembre de 2007

ANÁLISIS COMPARATIVO A NIVEL TEÓRICO Y PRÁCTICO DE LOS ALGORITMOS CRIPTOGRÁFICOS ORIENTADO HACIA LA SOLUCIÓN DEL PROBLEMA EN LA SEGURIDAD DEL TRANSPORTE DE DOCUMENTOS DIGITALES.

CARLOS ANDRÉS FERNÁNDEZ ZAMBRANO

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Popayán, Noviembre de 2007

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ANEXOS

INTRODUCCIÓN

En un mundo cada vez más adentrado en la llamada “Era de la Información” o “Sociedad del Conocimiento”; el continuo crecimiento de las redes informáticas y empresariales afronta un problema con muy pocos precedentes, el problema de la seguridad informática, en donde se maneja información delicada y privada, y donde intrusos hacen de las suyas y violan la seguridad de dichos sistemas con el fin de causar daños en la integridad de la información y en muchos casos de las mismas empresas.

Aunque el tema de la seguridad puede carecer de importancia para muchas personas y organizaciones que consideran poco prioritario la inversión de capital en este aspecto, se observa que el auge de los fraudes electrónicos ejecutados por los denominados “piratas informáticos“ ha hecho que actualmente se vea la seguridad como un problema vital en el momento de diseñar un sistema informático para una organización.

Las redes informáticas están expuestas hoy a amenazas de seguridad que pueden clasificarse, atendiendo a la intencionalidad de las mismas en accidentales o intencionadas. Las amenazas accidentales son las que se producen sin necesidad de un intento premeditado, como por ejemplo, una avería en el sistema. Las amenazas intencionadas pueden variar desde el examen casual de la información de un computador hasta ataques sofisticados utilizando conocimientos especiales sobre el sistema. Si se analiza el daño ocasionado y no la intencionalidad, las amenazas a la seguridad se clasifican en pasivas y activas. Las amenazas pasivas son las que no conllevan ninguna modificación en la información que posee el sistema y por tanto no se modifica ni su operación ni su estado. Las amenazas activas suponen una alteración del sistema y un cambio en su estado de operación. Aunque obviamente las amenazas activas son mucho más perjudiciales que las pasivas, estas deben ser tenidas en cuenta pues en muchos casos pueden convertirse en activas con la intervención de un agente distinto. [1]

Por las razones anteriormente expuestas y en búsqueda de conocimientos y avances al respecto para su posterior aplicación en redes informáticas se propuso el presente proyecto “ANÁLISIS COMPARATIVO A NIVEL TEÓRICO Y PRÁCTICO DE LOS ALGORITMOS CRIPTOGRÁFICOS ORIENTADO HACIA LA SOLUCIÓN DEL PROBLEMA EN LA SEGURIDAD DEL TRANSPORTE DE DOCUMENTOS DIGITALES” como alternativa para comprender mejor el tema de la Seguridad Computacional en lo que respecta al uso de los algoritmos criptográficos, ayudar en el trámite de documentación en forma segura mejorando los tiempos de respuesta de los procesos, disminuyendo costos, aprovechando mejor los recursos computacionales y de redes y ayudando a disminuir el impacto ambiental por la consecuente disminución del papel.

CAPÍTULO 1

Marco Teórico

La seguridad es un tema que debe inquietar a cualquier organización que hoy día decida conectar su red a otras sobre Internet. Basta mirar las estadísticas para tomar conciencia del riesgo que se corre: el número de incidentes contra sistemas conectados casi se duplica cada año, según el Computer Emergency Response Team Coordination Center (CERT-CC). Y se debe tener en cuenta el vertiginoso crecimiento de Internet en los últimos años, que implica, por una parte, nuevas redes susceptibles de ser atacadas, y por otra, nuevos atacantes en potencia. [1]

Hoy día, atacar una red conectada a Internet que no haya sido protegida de un modo "especial", es relativamente fácil si se sabe cómo, y mucho más aún si se utilizan sistemas operativos antiguos que no han sido actualizados ni debidamente "parcheados". Es tan frecuente como erróneo creer que una filosofía de seguridad tradicional, basada en contraseñas y protección de archivos, es suficiente para protegerse en Internet; en la red es posible encontrar, listas de debilidades tanto de protocolos como de sistemas operativos, así como guías que señalan los pasos a seguir para explotar dichas debilidades. Incluso existen servidores de FTP anónimo con todo tipo de herramientas orientadas a tomar el control de cualquier máquina. [1]

Todas las líneas actuales de investigación en seguridad de redes comparten una idea: la concentración de la seguridad en un punto. Se obliga a que todo el tráfico entre la red que se pretende proteger y las redes externas pase por un mismo punto. Este punto se conoce con el nombre de cortafuego, y físicamente puede ser desde un simple host hasta un complejo conjunto de redes separadas por routers. El empleo de un cortafuego presenta enormes ventajas sobre los enfoques de seguridad en redes tradicionales (que requieren la seguridad individual de cada host conectado, y por tanto sólo pueden justificarse en entornos con un reducido número de máquinas), permitiendo concentrar todos los esfuerzos en el control de tráfico a su paso por el cortafuego. [1]

En los últimos años se han conocido muchos ataques que estos individuos han hecho contra compañías principalmente del área del e-commerce, como los que ocurrieron en enero del año 2000 y en septiembre del mismo año contra las compañías CD Universe y Western Union respectivamente, los cuales han consistido principalmente en la copia de las bases de datos en que se guardan los números de las tarjetas de crédito de los clientes; por obvias razones los ataques de este tipo son los más publicitados, debido a que son los que presentan las pérdidas de dinero más escandalosas, ya que en estos casos los bancos deben cambiar las tarjetas de crédito correspondientes a los números robados y las compañías involucradas pierden algo más importante que es la confianza de sus clientes, lo que a corto plazo se traduce en una disminución significativa de sus ventas [2].

Estas situaciones han sido muy difundidas por los medios, lo cual ha provocado un cierto temor general hacia el uso de tecnologías que permitan el transporte de la información en forma electrónica, las organizaciones han sido renuentes a evolucionar de su manejo de

documentación en forma impresa hacia la forma digital, ya que se piensa que de esta manera sería más vulnerable; esta hipótesis es válida hasta cierto punto, ya que la carencia de buenas políticas de seguridad y del soporte tecnológico adecuado es muy común debido a la falta de conocimiento sobre el tema, además si a esto se le suma que los mecanismos de autenticación, autorización, confidencialidad e integridad que usan los sistemas que implementan seguridad dependen en exclusivo del uso de algoritmos criptográficos que podrían ser vulnerados, se debe pensar detenidamente en que para poderse dar esta evolución primero se tienen que solucionar estos aspectos, ya que si en una organización personas inescrupulosas logran extraer documentación importante podrían causar serios daños por el posterior manejo incorrecto de la misma [2].

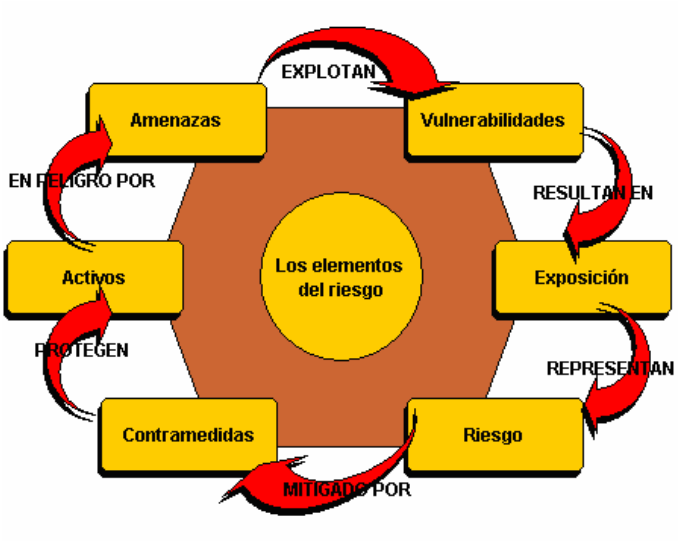

Figura No. 1 Los elementos de Riesgo [2]

El problema puede ser ilustrado mediante el gráfico mostrado en la figura 1, los activos que se deben proteger son los documentos digitales que se ven amenazados por personas inescrupulosas que buscan sacar ventaja de vulnerabilidades tales como: escasa o nula implementación de sistemas de seguridad o sistemas que implementan seguridad pero que han usado algoritmos criptográficos débiles que pueden ser descifrados usando herramientas criptoanalíticas, esto se traduce en que la información se exponga a riesgos que deben ser erradicados en lo posible usando contramedidas tales como el uso de algoritmos adecuados para proveer los requisitos mínimos de seguridad [2][3].

Además de tener algoritmos criptográficos fuertes, cabe destacar que un buen sistema de seguridad debe tener ayudas adicionales tanto software como hardware, en software se cuenta con la utilización de firewalls (en servidores o en la misma máquina), adicionalmente existen buenos antivirus y anti-spyware, también se puede contar con los honeypots y honeynets para engañar a los atacantes y distraer su objetivo a ciertos computadores aislados que miden estos ataques; en cuanto a hardware se refiere, existen soluciones implantadas en bancos y grandes organizaciones como el uso de escáneres de alta velocidad que convierten muy bien, los documentos físicos a digitales con reconocimiento de texto y además generan códigos de barra para marcar dichos documentos físicos; también se cuenta con escáneres de huellas digitales como refuerzo a las claves.

En cuanto a la criptografía, a través de la Historia han existido muchos instrumentos para el cifrado de mensajes, uno de los más antiguos es la escítala del siglo V a. de C. usada por los lacedemonios y que era un sistema que consistía en una cinta que se enrollaba en un bastón y sobre el cual se escribía el mensaje en forma longitudinal, la clave de este sistema residía precisamente en el diámetro de aquel bastón, de forma que solamente el receptor autorizado tenía una copia exacta del mismo bastón, otro ejemplo es el cifrador de Polybios (Siglo II a. de C.) este sistema consistía en hacer corresponder a cada letra del alfabeto un par de letras que indicaban la fila y la columna en la cual aquella se encontraba, en un recuadro de 5 x 5 (25 caracteres), transmitiéndose por tanto en este caso el mensaje como un criptograma ; en el Siglo XX aparecieron las Máquinas de Rotores como: ENIGMA (patentada en 1923 por el Ingeniero Alemán Arthur Scherbius), esta máquina fue diseñada específicamente para facilitar las comunicaciones seguras, muy parecida a una máquina de escribir. Quien deseara codificar un mensaje sólo tenía que teclearlo y las letras correspondientes al mensaje cifrado se irían iluminando en un panel. El destinatario copiaba dichas letras en su propia máquina y el mensaje original aparecía de nuevo; fue utilizado por el ejercito alemán, pero sus debilidades fueron bien aprovechadas por el ejercito aliado. Además de la máquina alemana ENIGMA, existieron otros dispositivos criptográficos basados en rotores como la máquina SIGABA, empleada por el ejército norteamericano, la máquina de Hagelin y las máquinas japonesas empleadas en la II Guerra Mundial, que se denominaron PURPLE y RED. [4] (Ver ANEXO 4)

Con la llegada del computador y el auge de la biotecnología, las posibilidades se han ampliado de una manera sin precedentes, permitiendo combinar tanto software como hardware para la implementación de buenos sistemas de seguridad.

1.1 Análisis de Posibles Vulnerabilidades

Antes de analizar a profundidad los aspectos concernientes a los algoritmos, primero se deben evaluar algunos de sus riesgos y destacar aspectos muy importantes como el estándar empleado para soportar cada uno de ellos ya que para garantizar que un sistema de seguridad es seguro, no basta con saber que los diferentes algoritmos que usa son seguros por sí solos porque las fortalezas de estos algoritmos no sirven de nada si la plataforma que los sustenta es débil, por ejemplo, el PGP(Pretty Good Privacy – Protocolo de Seguridad Bastante Buena); a continuación se presentarán debilidades de esta plataforma:

1.2 Debilidad del Párrafo Secreto en PGP

Actualmente, el mejor ataque posible contra PGP consiste en atacar con diccionarios buscando el párrafo secreto. En este tipo de ataque un programa toma palabras de un diccionario y las junta de diferentes maneras intentando adivinar el párrafo secreto.

Por esto es fundamental escoger un buen párrafo, muchos de estos programas son muy sofisticados y toman ventaja de los dichos, refranes populares y reglas de la gramática para armar sus frases, las frases de una sola palabra, nombres propios (especialmente si son famosos), frases célebres, etc.; son usualmente crackeables por uno de estos programas [5].

Algunas características de un buen párrafo secreto recomendadas por lo expertos son:

1. Evitar el uso de idioma inglés, debido a ataques estadísticos de letras.

2. Mínimo 12 caracteres

3. Incluir letras, números, espacios y signos de puntuación

4. Mayúsculas y minúsculas

5. Frases que contengan palabras indígenas o poco conocidas

6. Evitar significados convencionalmente conocidos (frases o nombres famosos, referencias personales, etc.)

7. Evitar a toda costa las referencias personales a uno mismo (nombres propios de familiares, fechas de nacimiento, número de teléfono etc.)

Aunque también existen métodos para obtener párrafos secretos sin recurrir al criptoanálisis; por ejemplo la "ingeniería social" para averiguar las claves, captar la radiación electromagnética de los computadores (los ataques Tempest) o el chantaje [4].

Se han roto públicamente dos mensajes cifrados con RSA (el algoritmo usado para cifrar las claves públicas)

Primero la clave RSA-129. Los inventores del RSA publicaron un mensaje cifrado con una clave pública RSA de 129 dígitos (430 bits), y ofrecieron U.S. $100 al primero que lograse descifrar el mensaje. En 1994, un equipo internacional coordinado por Paul Leyland, Derek Atkins, Arjen Lenstra, y Michael Graff factorizaron con éxito esta clave pública y recobraron el texto plano, que decía:

THE MAGIC WORDS ARE SQUEAMISH OSSIFRAGE

Ellos encabezaron un enorme esfuerzo voluntario en el cual el trabajo fue distribuido vía e-mail, fax y correo a trabajadores en Internet cada uno de los cuales procesó su parte y devolvió los resultados. Se usaron alrededor de 1600 computadoras, con diverso poder de cálculo, desde maquinas de fax a supercomputadoras Cray. Ellos usaron el mejor algoritmo de factorización conocido a la fecha; desde entonces han aparecido otros mejores pero los resultados son aún instructivos sobre lo que costaría descifrar un mensaje cifrado con RSA [5].

Hay que tener en cuenta que la clave RSA-129 es aproximadamente igual de segura que una clave PGP de 426-bit. PGP recomendaba claves de 384-bit para un nivel de seguridad "casual"; las versiones modernas ofrecen un mínimo de 512 bits para sus claves. Aunque es de notar que todo este esfuerzo logró descubrir solo una clave, no un método general que permitiera romper otras claves [6].

Un año después, por primera vez una clave real de PGP fue rota. Se trató de una clave de 384-bits que fue crackeada por una entidad anónima llamada "Blacknet". Un equipo formado por Alec Muffett, Paul Leyland, Arjen Lenstra y Jim Gillogly consiguieron factorizar la clave en tres meses. Luego se usó para descifrar algunos mensajes de dominio público que se encontraban cifrados para tal efecto [6].

Lo más importante de este ataque es que fue realizado en secreto y no se divulgó hasta que se obtuvo la clave descifrada. La mayoría de los computadores se usaron durante el tiempo libre y la potencia computacional requerida estaba al alcance de una organización grande a media.

Estos ataques han consistido en determinar la clave secreta de alguien a partir de su clave pública, resolviendo el problema de la factorización de grandes números que es clave en el algoritmo RSA que se usa para cifrar las claves de sesión.

Los magazines populares de seguridad computacional describen a menudo los productos criptográficos en términos de algoritmos y longitudes de clave; y utilizan pocas palabras con las que comparan fácilmente un algoritmo de otro. Por ejemplo, declaraciones como “Claves de 128 bits significan seguridad fuerte, mientras que las claves de 40 bits son débiles” o “triple-DES es mucho más fuerte que el DES simple” o aún “ el RSA de 2,048 bits es mejor que el RSA de 1.024 bits.” Desafortunadamente, la criptografía no es tan simple: Claves más largas no garantizan más seguridad [5].

Un algoritmo criptográfico se puede comparar a una cerradura de una puerta delantera. La mayoría de estas tienen cuatro contactos de metal, que pueden estar en una de 10 posiciones. Una clave del metal fija los contactos en una configuración determinada. Si la clave los alinea todos correctamente, la cerradura se abre. Existen 10.000 claves posibles, y si un ladrón quiere intentar abrirla ensayando las diferentes combinaciones, tiene en el peor de los casos 10.000 posibilidades. Pero un bloqueo mejorado con 10 contactos (10 millones de claves posibles) probablemente no harán la casa más segura. Los ladrones no intentan cada clave posible (el equivalente de un ataque de la fuerza bruta), la mayoría no escogen forzar la cerradura (el equivalente de un ataque criptográfico); por el contrario, abren las ventanas, fuerzan las puertas, se disfrazan como policía, etc. [7]

La criptografía fuerte es muy poderosa cuando se hace bien, pero no es algo ideal. Si se enfoca solamente en los algoritmos criptográficos y no se hace caso de los demás aspectos de la seguridad es comparable a querer defender la casa sin construir una muralla alrededor de ella, pero poniendo un poste inmenso en la tierra y esperando que el ladrón no pase de este.

Hoy día se rompe el esquema de diseñar algoritmos seguros mediante el análisis y rompimiento de los sistemas criptográficos; más bien se trata de dedicar la mayor parte del trabajo a examinar productos reales y no dedicarse a encontrar lo errores dominantes sino a explotar errores en las fases de diseño, puesta en práctica e instalación; si un sistema criptográfico contiene buenos algoritmos de cifrado, algoritmos digitales de firma, funciones unidireccionales de hash y códigos de autentificación de mensaje confiables, todo en conjunto garantizará que el sistema es seguro siempre y cuando se usen correctamente, esto es evidente en sistemas actuales que no pueden controlar el tamaño de ciertos valores o reutilizan parámetros aleatorios que no deberían. Incluso los buenos ingenieros, las compañías bien conocidas, y un buen grupo de trabajo no son ninguna garantía de la puesta en práctica de un sistema robusto. Las debilidades criptográficas descubiertas en el acceso múltiple por división de códigos (CDMA) y en los algoritmos de cifrado para comunicaciones móviles GSM y en el Protocolo de Entunelamiento Punto a Punto de Microsoft (PPTP) ilustran eso. En PPTP, por ejemplo, sucedió que un algoritmo fuerte como el RC4 fue utilizado en un modo que negó casi totalmente la seguridad de este protocolo. [6]

Los generadores de número aleatorio son otro aspecto en el cual los sistemas criptográficos fallan a menudo. Los buenos generadores de números aleatorios son difíciles de diseñar porque su seguridad depende a menudo de los detalles de la dotación física y del software; el sistema criptográfico puede ser fuerte, pero si el generador de número aleatorio produce claves débiles, el sistema es mucho más fácil de romperse. [6]

Los tiempos de respuesta de los algoritmos son otro factor a destacar, no solo porque la puesta en práctica de algunos sea lenta precisamente por la cantidad de procesos tan complejos que maneja y por ende los recursos que consume; sino también porque mediante tarjetas inteligentes y haciendo cálculos estimativos, es posible saber por ejemplo que un retardo de un 0 dentro de un bit es más corto que el de un 1 y de esta forma estimando tales tiempos se pueden obtener los bits para su posterior desciframiento; lo anterior expuesto en un seminario de la Facultad de Matemáticas de la Universidad del Cauca por el Doctor Julio López, Profesor de la Universidad de Campinhas Brasil, en Julio de 2006 hablando específicamente de los algoritmos de curvas elípticas.

Otros ataques se enfocan en las interacciones entre protocolos de cifrado seguros individualmente; dado un protocolo seguro, a veces es posible construir otro protocolo seguro que rompa el primer si ambos utilizan las mismas claves en el mismo dispositivo. Dado la proliferación de los estándares diversos de la seguridad usando la misma infraestructura, esta clase de incidente de la interacción es potencialmente muy peligroso. [6]

En cuanto a los ataques hacia un algoritmo en la puesta en práctica, se tienen algunos sistemas que no se aseguran de que el texto plano sea destruido después de que se haya cifrado. Otros sistemas utilizan archivos temporales para proteger contra pérdida los datos durante un fallo del sistema o para utilizar memoria virtual para aumentar la memoria disponible. Estas características pueden dejar accidentalmente el texto plano deambulando en el disco duro. [6].

Otro tipo de vulnerabilidades a destacar son las relacionadas con el lenguaje o soporte de programación a utilizar, por ejemplo las concernientes a las aplicaciones web, principalmente a las escritas en PHP, sin duda alguna las más populares son los ataques XSS (Cross-Site Scripting mediante el cual se pueden enviar cadenas con comandos ocultos en vínculos que se almacenan y reflejan en las instrucciones PHP) y CSRF (Cross-Site Request Forgery el cual es una clase de ataque que afecta a las aplicaciones web con una estructura de invocación predicable usando cookies, autentificación de navegador o certificados de cliente). Sin embargo, en el caso de detectar este tipo de vulnerabilidad en la aplicación, los programadores no les prestan mucha atención, ya que las consideran como inofensivas, además de que a veces no se interesan por proteger las aplicaciones que crean desde el principio [8]..

El objetivo de ambos ataques es el mismo: aprovechan cierta vulnerabilidad, el agresor puede colocar en la página cualquier código, el cual posteriormente puede servir para la ejecución de operaciones no planificadas por el creador del sitio Web, por ejemplo, capturar archivos sin que el usuario se percate.

La diferencia entre XSS y CSRF radica en la manera de suministrar el código empleado para el ataque. XSS aprovecha la posibilidad de introducir cualquier código en un campo de texto no verificado, por ejemplo, anunciado en un formulario con el método POST, en

cambio durante los ataques tipo CSRF para la descarga y ejecución del código (empleado por el intruso) no se utiliza la falta de validación, sino funciones determinadas de los navegadores Web. [8]

El tipo de ataque CSRF más popular se basa en el uso del marcador HTML <img>, el cual sirve para la visualización de gráficos. En vez del marcador con la URL del archivo gráfico, el agresor pone un tag que lleva a un código JavaScript que es ejecutado en el navegador de la víctima. Esto permite llevar a cabo varias operaciones en el marco de la sesión del usuario, el cual con frecuencia no es consciente de que precisamente está siendo atacado. [8]

Lo importante de estos ataques es el modo de presentar a la víctima la página Web modificada. En la mayoría de los casos la agresión se basa en la adición o modificación del contenido de la página mediante el cambio de la dirección URL a petición de GET y obligar al usuario a que haga clic en el enlace de esta dirección. Esta última etapa del ataque requiere un poco de técnica social y de aquí precisamente nace la idea de los ataques (aparentemente “inocentes”) XSS, después de todo la efectividad de éstos requiere la cooperación del usuario (incluso si ésta es inconsciente) [8].

No obstante, este es sólo uno de los posibles métodos para suministrar los datos que atacan. La información imprescindible para llevar a cabo el ataque XSS también se puede almacenar permanentemente, si el agresor logra guardar datos maliciosos en el sitio Web atacado, éstos se podrán ejecutar en una cantidad innumerable de usuarios que visitan el servicio sin la intervención de ellos. Esto significa que no son necesarios trucos de técnicas sociales, dado que cada usuario que visite el sitio estará expuesto al ataque [8].

Para establecer un buen sistema de seguridad se puede comenzar por evaluar su desempeño y eficiencia, actualmente se cree que los algoritmos de curva elíptica pueden superar las vulnerabilidades de seguridad existentes, a continuación se mostrará más detalladamente sus principales aspectos para determinar su aplicabilidad y se comenzará haciendo una breve comparación entre los tres principales tipos de criptosistemas utilizados.

Los algoritmos criptográficos buscan brindar un alto nivel de seguridad en como mínimo los requisitos de: disponibilidad, integridad, autenticación, aceptación y confidencialidad; con el fin de asegurar que la información no pueda ser alterada ni leída por personas indeseadas, además que siempre esté disponible y nadie pueda negar lo que hizo con la misma. Por todo lo anterior se generó la inquietud por llevar a cabo un estudio en profundidad a los algoritmos criptográficos con el fin de dar conclusiones y recomendaciones acerca de su manejo adecuado como solución al problema de la seguridad en el tránsito de documentos digitales a través de una red informática.

Posteriormente se analizarán los algoritmos más usados para evaluar sus fortalezas y debilidades (incluyendo aspectos como el punto de vista del estándar empleado), esto con el fin de escoger los más idóneos para los diferentes niveles de seguridad que se pueden tener en el manejo de documentación digital según el tipo de información a transportar por una Intranet.

Capítulo 2

Estado del Arte de los Algoritmos Criptográficos Reconocidos y de Algunos Sistemas de Gestión Documental y de Transferencia de Archivos

2.1 Análisis del Estado del Arte de los Algoritmos Criptográficos más representativos

Actualmente, se experimenta en el mundo, una serie de problemas que afectan gravemente los intereses de cualquier organización y son básicamente: los relacionados a la protección de los dispositivos de almacenamiento y los relacionados a la intervención de los medios de transmisión utilizados, siendo estos los más incidentes en el tema a desarrollar porque la información se ha convertido en un patrimonio fundamental de las empresas, ya que estas pueden por ejemplo efectuar transacciones o manejar información privada; y esto lo hacen en entornos multiusuario en los cuales se manejan esquemas de protección de clave pública con los protocolos de inicio de conexión y palabra clave de acceso al sistema, acompañados de la firma digital y la certificación digital que garantizan la autenticidad del proceso.

A través de la historia ha habido distintas maneras de ocultar información y en muchos casos la evolución de estos medios, tanto hardware como software, han sido incentivados en tiempos de guerra, lo que impulsó su evolución y posterior aplicación en procesos comerciales y corporativos.

A través del presente trabajo se tratará lo concerniente a los algoritmos criptográficos probados y utilizados desde la década del setenta hasta nuestros días, ya que la constante mejora de los algoritmos criptográficos tanto simétricos, asimétricos y de hash y la gran variedad de estos, dificultan en gran manera poder realizar un análisis completo según su nivel de seguridad; razón por la cual se han seleccionado unos cuantos para realizar una clasificación de los mismos para su posterior uso en un piloto de pruebas.

2.1.1 Algoritmos Simétricos

Los algoritmos simétricos en general, trabajan en forma similar; de todos el más difundido es el DES (Data Encryption Standar) el cual cifra y descifra textos de 64 bits utilizando la misma clave, que también es de 64 bits (de los cuales 56 son usados directamente por el algoritmo y 8 son usados para detección de errores); utiliza una función compleja que puede definirse en términos de una función f denominada función de cifrado y una función KS denominada planificación de clave. [9]

Es de anotar que el DES constituye el estándar de criptografía de NIST (Instituto Nacional de Estándares y Tecnología) hasta nuestros días, y su versión implementada con hardware entró a formar parte de los estándares de la Organización Internacional de Estándares (ISO) con el nombre de DEA (Algoritmo de Cifrado de Datos); como ventajas de este algoritmo se tiene que es el de uso más extendido en el mundo, es el más barato, el más probado y en veinte años no había sido roto con un sistema de criptoanálisis normal, aunque entre sus inconvenientes se tiene que no puede ser utilizado fuera de Estados Unidos, sin un permiso del Departamento de Estado de ese país, su clave es corta y existe un nuevo sistema de criptoanálisis, el diferencial, capaz de romper el DES

teniendo suficientes muestras; y esto fue hecho por Matsui en 1994 el cual permitió romper el DES con ataques a texto claro conocido, usando 243 pares, en 40 días de trabajo utilizando ataques diferencial y lineal; además se cuenta con otro referente el descrito por DES Challenge III y DESCracker junto con otros 100.000 computadores a través de Internet, rompieron DES en 22 horas en 1999. [10] [11]

Con el fin de hacer mejoras como en la longitud de la clave, y para seguir utilizando el embalaje del DES apareció el TripleDES, que utiliza una clave de 128 bits (16 de paridad y 112 de clave) y además es compatible con el DES; el principio de funcionamiento consiste en aplicar 64 bits a los dos DES contenidos y los otros 64 bits al DES inverso o ANTIDES que se realiza entre los otros dos. [10]

Otro algoritmo simétrico a considerar es el IDEA (Algoritmo de Cifrado de Datos Internacional) inventado en 1990 por Lai y Massey del Instituto Federal de tecnología suizo. En 1992 se publicó la segunda versión resistente a ataques de criptología diferencial, este algoritmo está libre de restricciones y permisos nacionales y es de libre distribución por Internet; esto lo ha popularizado y actualmente se lo utiliza en sistemas UNIX, en aplicaciones como PGP, etc. El algoritmo trabaja con bloques de texto de 64 bits y una clave de 128 bits. Siempre opera con números de 16 bits utilizando operaciones como XOR, suma de enteros o multiplicación de enteros, el algoritmo de descifrado es similar. Hasta ahora nunca ha sido roto aunque no tiene la confiabilidad que presenta el DES.[10]

Continuando con los algoritmos más conocidos, está el RC5, inventado por Rivest que proviene del RC4 y es propiedad de RSA Data Security Inc. Es utilizado por Nestcape en su sistema de seguridad SSL. Como la mayoría de los algoritmos permite diferentes longitudes de clave, fuera de los EEUU solo se puede exportar, de acuerdo a las últimas normas ITAR (Normas que regulan el tráfico de armamento a nivel internacional en los Estados Unidos), la versión con clave de 40 bits. Funciona como un generador de números aleatorios que se suman al texto mediante una XOR. Permite la configuración de muchos parámetros: número de iteraciones, longitud de clave y tamaño de bloque, esto permite adaptarse a las necesidades de velocidad/seguridad de cada aplicación. [10]

El nuevo estándar de Seguridad de el NIST, AES (Estándar de Cifrado Avanzado) nace de las necesidades que no alcanza a cubrir el DES (porque las aplicaciones eran cada vez más versátiles) tales como la clave corta, clave fija y limitaciones legales, entre otras; razón por la cual NIST dejo de usarlo en 1997 y en consecuencia se convocó un concurso para buscar un nuevo estándar, el cual, se llamará AES y su algoritmo AEA (Algoritmo de Cifrado Avanzado). Hasta junio de 1998 se receptaron los candidatos y luego de la primera ronda se obtuvo los 15 semifinalistas (tabla semifinalistas), pasaron a la segunda ronda 5 finalistas: MARS, RC6, Rijndael, Serpent y Twofish. Las bases del concurso eran: permitir claves simétricas mayores y de longitud variable, utilizar bloques de cifrado de gran tamaño, proveer una gran velocidad de cifrado y dar la mayor flexibilidad y soporte multiplataforma; finalmente el 2 de octubre de 2000 se anunció el algoritmo ganador: El RIJNDAEL con 86 votos, propuesto por los belgas Vincent Rijmen y Joan Daemen. Los motivos para su escogencia fueron su fácil diseño, su versatilidad al ser implementado en diferentes dispositivos, ser inmune a los ataques conocidos hasta la fecha, soportar bloques de datos de 128 bits y claves de 128, 192, y 256 bits y por darle “performance” al TDES. [10] [12]

En el estado actual de AES proporcionado por el NIST, en cuanto a la seguridad del AES se registra en el siguiente reporte dado en la conferencia AES4, donde se afirmaron los siguientes puntos [13]

1. AES maneja muy bien longitudes de clave de 80, 112, 128, 192, y 256 bits.

2. Usar claves de 80 bits será seguro hasta el año 2010, 112 hasta el año 2020, y 128 posteriormente. Aunque esta seguridad puede reducirse por el modo de operación de los algoritmos en consideración.

3. Para 80 bits de seguridad AES, es equivalente a 1024 bits de RSA, y 163 de ECDSA.

4. Se recomienda estandarizar los niveles de seguridad de todos los algoritmos en una aplicación, por ejemplo al usar AES 80, RSA 1024/ECDSA-160, y un MAC de 160 bits.

5. Simple DES queda totalmente descontinuado, TDES con dos claves podrá ser soportado hasta el año 2010, TDES con 3 claves hasta 2020, de ahí se espera sólo usar AES con 128.

6. Se habla de los ataques llamados algebraicos que actualmente no son operables, sin embargo, se presume pueden ser una línea de criptoanálisis eficiente en el futuro, aunque hay más preguntas que respuestas respecto a este tipo de ataques [15] [16]

7. Como un punto complementario del estado actual de AES cabe mencionar su utilización en algunas aplicaciones como: TLS, HMAC, IPsec, IKE, S/MIME, SIP, etc. o RFIDs

En cuanto al estándar PGP (Pretty Good Privacy - Privacidad Bastante Buena) que por defecto es usado para el cifrado de e-mails, discos duros, comunicaciones de voz, entre otras otras aplicaciones; Javier Requejo y Cristian Alfredo Boaglio, (expertos en PGP consultados) se recomienda usar los siguientes algoritmos:

En bloque o simétricos:

1. CAST: clave de 128 bits, rápido, bien diseñado, su desventaja: es relativamente nuevo y no ha sido sometido a ataque durante mucho tiempo. Tiene patentes pendientes pero existe el compromiso de sus creadores de dejar una versión libre en cualquier caso. [5]

2. Triple-DES: clave de 168 bits, el algoritmo DES, publicado por IBM en los 70s aplicado tres veces lo hace un sistema muy robusto, tal como los otros dos algoritmos no se sabe de nadie que haya logrado romperlo a la fecha. No está protegido por patentes, es el más lento de los tres [5]

3. IDEA: clave de 128 bits. el algoritmo fue publicado en 1990 y ha soportado toda clase de ataques desde esa fecha, probablemente obtenga patente lo que podría

ser un problema para su uso futuro. (Para mayor información en cuanto a los Algoritmos Criptográficos Ver Anexo 2) [5]

Para generación de firmas digitales se usa el siguiente algoritmo:

4. MD5: genera un hash de 128 bits y cumple con las propiedades de las funciones de hash seguras: 1. Es computacionalmente infactible encontrar dos mensajes indistintamente que tengan el mismo hash y 2. Dada una función hash es computacionalmente infactible encontrar un mensaje cualquiera que lo produzca. Se ha calculado que con este algoritmo la dificultad de conseguir dos mensajes con el mismo hash es de 2^{64} intentos. [6]

5. SHA: genera un hash de 160 bits, ha sido publicado sin limitaciones y ha sido extensamente revisado por la mayoría de los mejores criptógrafos del mundo que se especializan en funciones de fragmentos mezclados, y la opinión unánime es que SHA está extremadamente bien diseñado, dejando a un lado el MD5 que según criptoanalistas posee serias debilidades y que ya es posible quebrar su integridad de forma rápida.[6]

Como Algoritmo Asimétrico:

6. RSA : La clave pública es el par (N, E). La clave privada es el número D. En principio, no se conocen métodos por los que sea fácil calcular D, P o Q, dados solamente (N, E), y dado que P y Q son números de 1024 bits, no existe todavía una computadora lo suficientemente potente para sacar todos los factores de N y probar combinándolos. Sin embargo, hay que tener en cuenta que no está probado del todo que no exista ese método, o que no pueda conseguirse dentro de unos años. Tampoco está probado que la única manera de romper el algoritmo RSA sea sacando los factores de N. [6]

2.1.2 Algoritmos Asimétricos

En 1976 nace el concepto de “Algoritmos de Clave Pública” (Ver Anexo 2), el cual fue publicado en “New Directions in Cryptography” IEEE Trans on Info Theory, cuyos autores son W. Diffie y M. Hellman; los cuales proponen un sistema de comunicación privada que emplea un directorio de claves públicas, de tal modo que cada usuario fija un procedimiento E para que sea usado por otros usuarios cuando cifren mensajes que vayan dirigidos a él, mientras que guarda en secreto su propio procedimiento de descifrado D. La principal diferencia respecto de otros sistemas es la característica de asimetría; es decir, la clave para cifrar y descifrar son distintas y es prácticamente imposible identificar la una partiendo de la otra. También las capacidades de cifrar y descifrar están separadas, con lo que se puede proteger la información sin guardar en secreto la clave de cifrado, ya que esta no será necesaria para descifrar. Como ejemplo de este tipo de algoritmo se tiene al RSA (Rivest, Shamir y Adleman) como el más representativo. [9] [10]

En agosto de 1999 la propia compañía consiguió romper un RSA con clave de 512 bits en algo mas de 5 meses con el trabajo conjunto de más de 290 PC’s, por ello es aconsejable usar claves de 1024 bits o más; razón que resulta paradójica ya que se encuentra fuera de los límites legales. [10]

El Algoritmo de Diffie-Hellman, publicado en 1976 fue el primer algoritmo asimétrico desarrollado, usado para ilustrar la nueva criptografía de clave pública, la seguridad del algoritmo depende de la dificultad del cálculo de un logaritmo discreto. Junto con el RSA estos algoritmos son utilizados indistintamente por el PGP para la creación de pares de claves (pública y privada). [10]

Además de los algoritmos simétricos y asimétricos, también se ha avanzado en el desarrollo de nuevos sistemas criptográficos con algoritmos diseñados para cubrir otros aspectos del embalaje de la infraestructura de seguridad utilizada, tal es el caso del estándar DSS (Estándar de firma digital), el cual es un sistema de firma digital adoptado como estándar por la NIST, utiliza la función hash SHA y el algoritmo asimétrico DSA (Algoritmo de firma digital) que únicamente se puede utilizar con firma digital. Utiliza más parámetros que el RSA y así consigue un grado mayor de seguridad (KG, KU, KP, k, s y r). [10]

2.1.3 Algoritmos Hash

En muchas situaciones se requiere conocer la validez de un mensaje, es decir, hay que verificar que no ha sido modificado en el transcurso del mismo y que llega tal y como fue escrito originalmente, para ello se utiliza una función de dispersión o compendio de mensaje (message digest o función HASH), que tiene tres propiedades importantes:

- Dado un texto P, es fácil calcular su compendio de mensaje MD(P).
- Dado un compendio de mensaje MD (P), es imposible encontrar P.
- Nadie puede generar dos mensajes que tengan el mismo compendio de mensaje.

Para cumplir con el tercer criterio, la dispersión debe ser de cuando menos de 128 bits de longitud, y preferentemente mayor.

Los compendios de mensaje funcionan tanto en clave privada como en clave pública, siendo los de mayor uso el MD5 y el SHA.

Hasta hace poco, tanto el MD5 como el SHA se habían mostrado inviolables, pues aún con ataques sofisticados ideados (como el ataque de cumpleaños), se tardarían más de 500 años en calcular los compendios de dos cartas con 64 variantes cada una, e incluso entonces no se garantizaba una equivalencia [4].; pero Microsoft recomendó una política de desarrollo que prohíbe a sus programadores el uso de DES, MD4, MD5 y algunas implementaciones de SHA, hasta el punto de que, en caso de utilizarlos, serán marcados por los sistemas automáticos de escaneo de código para ser sustituidos de inmediato, este descarte se realizó por considerarlos "poco seguros"; además esta política se quiere extender también al código antiguo. [17]

Como sustitutos más fiables, Microsoft recomienda el uso SHA-256 y AES.

Al conocer esta nueva política de desarrollo el Dr. Bruce Schneier afirmó que Microsoft debería haber desechado MD4, MD5 y DES hace años. Aunque se estima que el SHA es uno de los más usados.

2.1.4 Algoritmos de Curva Elíptica

Criptosistemas de Clave Compartida: Inconvenientes

- Distribución de claves
- Cada Usuario tiene que gestionar una gran cantidad de claves (depende del número de usuarios del sistema)
- Imposibilidad de firmar mensajes

Criptosistemas de Clave Pública

- La seguridad reside en problemas matemáticos subyacentes, computacionalmente difíciles como el problema de la factorización de los números aleatorios y el problema del cálculo del Logaritmo discreto (Ej: EL Gamal).

Criptosistemas basados en Curvas Elípticas (CCE): Disminuyen el tamaño de las claves garantizando la misma seguridad, esto se refleja en el mejor aprovechamiento de los recursos disponibles ya que al realizar las operaciones de suma, producto, exponenciación, entre otras, se observa que la suma se realiza sin costo considerable de cómputo, ya que la suma puede ser reemplazada por operaciones estándar en cualquier lenguaje de computación (como la operación xor), respecto a la exponenciación existen métodos que ayudan a realizarla a un costo mínimo o incluso sin costo, que es el caso cuando se usa una base normal; sin embargo la operación de mayor preocupación es el producto ya que de éste depende en su mayoría las operaciones que se realizan en un sistema criptográfico, por lo tanto es irreversible buscar la mejor forma de implementarla [18].

Se debe tener en cuenta que para la criptografía y especialmente para los CCE es muy importante conocer el orden del grupo, es decir, el número de parejas (x, y) que satisface la ecuación que define la curva, llamados puntos racionales. Será fundamental conocer este número, ya que de esto depende en gran parte si se usala curva elíptica o no, más concretamente este número debe de tener un factor primo de al menos 48 dígitos para que el sistema criptográfico se considere seguro (Ver Anexo 2) [18].

Hoy día algunas empresas han establecido sus propios niveles de seguridad a partir de criterios propietarios, como por ejemplo el caso de PentaWare, el cual admite 7 algoritmos de codificación fuerte PGP (este no es un algoritmo como tal, más bien es un protocolo que ayuda a codificar y decodificar datos, utilizando el sistema de codificación de llave pública RSA), AES, DES, Triple DES, Blowfish, Serpent, Mars y Twofish para comprimir, ver y convertir archivos.[19]

2.2 Análisis de algunos sistemas de gestión documental y de transferencia de archivos.

Como se ha visto hasta el momento la información es un factor determinante que debe protegerse, y la prueba más palpable de ello es lo concerniente a las comunicaciones militares, además esto ha generado a su vez avances en el desarrollo tecnológico, ya que la combinación conjunta de software y hardware, han dado a la criptografía herramientas para que esta sea utilizada como medio, para construir soluciones en seguridad electrónica que cumplan los siguientes aspectos: Identificación, Autenticación, Autorización, Integridad Confidencialidad y Aceptación (Ver Anexo 1).

Teniendo presente lo anterior, se dará una visión de los sistemas existentes a nivel local, nacional e internacional que proveen seguridad mediante el cifrado de datos, transferencia tanto de archivos, como de texto plano y de mensajes en claro, sin detallar las características confidenciales de los mismos.

2.2.1 Sistema GDOC

Este sistema de Gestión Documental fue desarrollado bajo la dirección de la Ingeniera de Sistemas Clara Inés Uribe y se puede adaptar a las necesidades de la Universidad del Cauca; este sistema cumple con las leyes de archivística, maneja adecuadamente las tablas de retención documental y tiene operaciones básicas tales como: la radicación de documentos que contiene los datos generales, junto con una breve descripción de los mismos, permite adjuntar archivos: descripción del material adjunto (por ejemplo: cd's, revistas, etc.) y genera alarmas para dar avisos oportunos y así cumplir con la entrega precisa de los documentos, esto lo hace mediante el adecuado manejo de los tiempos de respuesta; lo anterior se presenta en la Tabla No. 1 (Cortesía de la Ingeniera Clara Inés Uribe).

Según estudios previos en algunas dependencias como la ventanilla única hay software hecho según las necesidades propias para manejo de archivos, pero este no maneja tablas de retención documental; además se cuenta con el inconveniente de que la Universidad del Cauca no tiene un manual de funciones y procedimientos bien definido, razón por la que este sistema representa un modelo a tener en cuenta ya que la Universidad del Cauca es autónoma en cuanto a procesos y decisiones internas se refiere y además brindaría agilidad a los trámites de documentos internos junto con el orden adecuado debido a que se han presentado problemas de pérdida de los mismos durante su transporte.

2.2.1.1 Características:

Maneja la siguiente clasificación de Documentos:

CLASE	TIPO	SERIE	SUB-SERIE
OFICIOS DE ENTRADA Y DE SALIDA	ACTA	Comité Dirección	
	INFORMES	Ministerio del ambiente Ministerio de Hacienda Contraloría Consejo Directivo Procuraduría Asamblea Función Publica Dirección General Interinstitucional Técnicos Interventoria Visita Imprenta Nacional Asocar Fondo Nacional del Ahorro	
	PLANES	Acción Trianual Operativo Anual Gestión ambiental regional Mejoramiento	
	PROYECTOS	Bosques Recurso Hídrico Biodiversidad Procesos Productivos Fortalecimiento Institucional	
	AUTO	De Inicio De Prueba De Cargp	
	RESOLUCIONES	Licencias Permisos Concesiones Sanciones Autorización Donación Otro	
	NOTIFICACIONES	De Auto de inicio De Auto de Prueba De auto de Cargos De Resoluciones	
	PLIEGO CARGOS		
	EDICTOS		
	PODERES		
	SENTENCIAS		
MEMORANDOS			
PROYECTOS			

Tabla No.1 Clasificación de los Documentos del GDOC

2.2.2 Sistema Adapting Document

Es un sistema de gestión documental de tecnología abierta configurable dentro de una Intranet/Extranet. Con un repositorio de documentos seguro y centralizado, el sistema es administrable desde navegador, y dispone de herramientas para la búsqueda y distribución controlada de documentos, además de posibilitar la publicación en Intranet o Extranet de documentos en cualquier formato (texto o multimedia), incluyendo documentación académica, comercial, de laboratorio, de calidad, etc. Se trata de un

repositorio virtual accesible por medio de un navegador, en cualquier lugar y a cualquier hora, pero protegido contra accesos no autorizados. Los documentos se personalizan a perfiles profesionales o a usuarios concretos, a los que es posible solicitar confirmación de lectura mediante firma digital (opcional). Cada documento dispone de una ficha técnica de descripción. El usuario, después de leerla, puede descargarse el documento por Red o no. [10]

Existe además un buscador inteligente de documentos que es capaz de buscar por ocurrencias de texto o por palabras clave, con posibilidad de clasificación por carpetas y por tipo de documento. Igualmente, se puede implementar un workflow de propuesta/ emisión/ revisión/ aprobación/ publicación definitiva de documentos, con la opción de impedir la modificación simultánea del mismo documento y cumpliendo con las últimas normas internacionales de calidad (ISO 9000, 14001, 17025). El administrador puede gestionar la base de documentos desde un panel de administración (vía Web), subiendo o borrando archivos en el directorio virtual. Asimismo, dispone de una base de datos de usuarios autorizados, provista de estadísticas y de control de accesos. Este sistema realiza las siguientes fases:

Fase 1: Recogida y clasificación de la documentación

El personal que maneja la documentación deberá depositar de forma ordenada sobre la bandeja del nuevo escáner la documentación a digitalizar. A partir de aquí, el proceso es totalmente automático y no requiere de intervención humana.

Fase 2: Digitalización y caracterización de los documentos

El escáner realizará de forma autónoma la digitalización de los documentos. A continuación, el software OCR asociado al escáner reconocerá los documentos digitalizados y los clasificará en función de la información relevante que se haya estipulado con anterioridad (por ejemplo, código del documento), de forma que éstos quedarán accesibles en formato electrónico dentro de la carpeta apropiada. [10]

Los documentos digitalizados y caracterizados disponen a partir de ahora de dos apartados:

Ficha de metadatos. Permite incluir las características del documento, su ubicación, destinatario, estado de tramitación, palabras clave, etc.

Imagen del documento. Permite la lectura en monitor o impresión del documento con una calidad suficiente y un tamaño de fichero reducido (aprox. 50 KB). El formato de esta imagen puede ser TIFF o PDF.

Fase 3: Archivo de los documentos digitalizados

Como último paso del proceso automático, el software guardará toda la información anteriormente citada a una base de datos que será accesible mediante la aplicación Web Adapting Document, de forma que únicamente será necesario disponer de un navegador tipo Explorer para poder acceder a la documentación digitalizada.

Esta documentación se archiva de forma estructurada, por lo que su acceso es posible mediante búsquedas inteligentes para todos los integrantes de la red local, por ejemplo, seleccionando fechas, clientes o números de apunte, etc.[10]

2.2.3 Sistema Datotek

Es un sistema de transferencia de mensajes cifrados que existe en una institución militar de Ecuador, desde mediados de los años 80. Esta familia de máquinas diseñadas especialmente para este propósito, permite enviar mensajes cifrados por teléfono o vía radio. De acuerdo al modelo estas máquinas pueden almacenar un limitado número de mensajes para su transmisión. La velocidad de transferencia máxima para teléfono es de 1200 baudios. Además de las máquinas de cifrado, tiene un generador de claves propietario, estas claves son repartidas por distintas vías a los usuarios del sistema de todo el país por tierra y mar. En cuanto al cifrado, este se compone de dos pasos: primero, en código interno se cambia todos los caracteres ingresados en texto plano a caracteres comprendidos entre la A y la Z, luego esta cadena de caracteres se codifica con claves de 7 bits usando un método especial de ruedas; esto permite, considerando un alfabeto de 26 caracteres, que se obtengan hasta 5 rondas de cifrado. La seguridad del sistema reside no en el “secreto” del algoritmo sino en su diseño complejo desde el punto de vista matemático y en el secreto y el cambio frecuente de las claves que inicializan el algoritmo. [10]

2.2.4 Sistema Cóndor

La ESDESU (Escuela de Superficie. Anteriormente llamada CentroTáctico Naval, es la encargada del mantenimiento del sistema de Comando y Control de las unidades navales en Ecuador) en el año de 1994 conformó un grupo de trabajo que desarrolló un sistema criptográfico denominado “Sistema Cóndor” el cual suministraba codificación segura para archivos. Este sistema tiene tres limitantes importantes:

No brinda un entorno amigable al usuario
El grupo de los desarrolladores no se encuentra ya dentro de la Institución
No existe documentación que permita mantener o comprender el sistema al 100%

El sistema fue desarrollado bajo C++, y es una fusión de los algoritmos DES y un sistema de ruedas similar al usado por la Datotek. [10]

2.2.5 Sistema Siam IRS

Siam IRS es una gama de software diseñada para la Creación, Distribución, Almacenamiento, Edición, Mantenimiento y Gestión de todo tipo de información, documentos e imágenes a través de tecnología Internet. La solución Siam IRS (Soluciones Inteligentes para Aplicaciones Multimedia) consiste en aplicaciones de Gestión de Información y Documentos que el usuario gestiona a través de Internet, por lo que no necesita tener conocimientos especiales; además integran las actividades de información en una sola aplicación relacionando todas las informaciones correspondientes a la gestión del negocio, como son: la gestión de Bases de Datos documentales, el Archivo Electrónico, el seguimiento de tareas, la distribución y de documentos y su firma, la recuperación de informaciones y documentos de forma exhaustiva y pertinente, mediante motores de búsqueda con algoritmos propios. [20]

2.2.6 Sistema neoDOC

Con este sistema es posible compartir documentos e información en tiempo real entre varias sedes o incluso con clientes, garantizando la seguridad y la privacidad, y utilizando siempre un entorno web, lo que permite trabajar vía intranet/extranet desde cualquier equipo, sin necesidad de ninguna instalación cliente y con cualquier sistema operativo o arquitectura.[21]

En la mayoría de aplicaciones de gestión documental no se consigue fácilmente información acerca de la manera como implementan la seguridad, ya que esta la consideran como información privada, aunque el sistema Siam IRS es de los pocos que admiten la utilización de algoritmos propios, lo cual es un riesgo porque el proceso para considerar un algoritmo como seguro depende de la publicación tanto de sus procesos matemáticos, como de las pruebas realizadas, lo que le da la información necesaria a los criptoanalístas para que ellos hagan su propia investigación y determinen qué tan seguro puede llegar a ser; además otro aspecto importante a tener en cuenta es que los trámites de documentos digitales están sujetas a las leyes de cada país y por lo general las soluciones ofrecidas por las distintas empresas tienen un carácter estándar, razón por la cual no se adaptan totalmente a las necesidades de las diferentes empresas u organizaciones en donde van a ser implementadas.

Capítulo 3

Criterios de Escogencia de los Algoritmos Criptográficos

Antes de estudiar los niveles de seguridad necesarios para la implementación eficaz de un buen sistema de manejo de información, conviene elegir un buen algoritmo, donde “bueno” significa que utilice pocos recursos, siendo usualmente los más importantes el tiempo que lleve ejecutarse y la cantidad de espacio en memoria que requiera. Es engañoso pensar que todos los algoritmos son "más o menos iguales" y convertir un mal algoritmo en un producto eficaz; es así asimismo engañoso confiar en la creciente potencia de las máquinas y el abaratamiento de las mismas como remedio de todos los problemas que puedan aparecer; es por eso que antes de realizar el análisis comparativo de los diferentes algoritmos es necesario establecer que los criterios a utilizar se extractaron de características comunes y consideradas como relevantes para efectuar un estudio estimativo y adaptado a las necesidades de un sistema de documentación digital; estos criterios a detallar se han tomado como base para la comparación entre los diferentes algoritmos y su escogencia se realizó de modo subjetivo.

3.1 Eficiencia en el uso de los Recursos Computacionales

La eficiencia es de las propiedades más importantes que debe tener un algoritmo criptográfico; porque además de depender del diseño del algoritmo y de su implementación depende de la plataforma donde se ejecute y de los recursos físicos disponibles, entre otras cosas. Obviamente la mejor eficiencia alcanzada por cualquier algoritmo se obtiene con la implementación de hardware dedicado. Se tiene entonces que el campo de la eficiencia esta dividida en dos campos el de software y el de hardware.

Lo anterior se debe considerar debido a sus implicaciones en el aprovechamiento de los recursos de cualquier empresa u organización, ya que generalmente se observa que los equipos cliente asignados a los empleados tienen apenas la cantidad suficiente de recursos para cumplir con las tareas asignadas, razón por la cual no se pueden cargar de tareas o procesos complejos que hagan que las demás tareas sean ineficientes.

La eficiencia de un algoritmo ya sea software o hardware contiene dos ingredientes fundamentales: espacio y tiempo.

- La eficiencia en espacio es una medida de la cantidad de memoria requerida por un programa.

- La eficiencia en tiempo se mide en términos de la cantidad de tiempo de ejecución del programa.

Ambas dependen del tipo de computador, el tipo de compilador y de si se trabaja con máquinas de un solo procesador o de varios.

Para la eficiencia en tiempo generalmente se utiliza como indicador para este criterio el concepto de productividad.

La Productividad es el número de bits procesados por segundo (que es la unidad de tiempo estándar) a medida que aumente el número de datos o información a procesar.

Por otra parte, para los equipos servidor (que cuentan con más recursos que un equipo cliente promedio) generalmente se utiliza el concepto de Rendimiento que es el porcentaje entre el número de usuarios que se pueden atender simultáneamente y los tiempos de respuesta de cada algoritmo.

Lo anteriormente expuesto se hace en tiempo real y constituye una estrategia empírica para medir con datos reales esta eficiencia; con máquinas comunes y para efectos de análisis teóricos se utiliza comúnmente La Máquina de Turing desarrollada en 1936 por Alan Turing (la cual se cubre en los cursos denominados Teoría de la Computación o Teoría de Automátas), estableciendo que cualquier algoritmo puede ser representado por ella.[22]

Turing mostró también que existen problemas matemáticos bien definidos para los cuales no hay un algoritmo. Hay muchos ejemplos de problemas para los cuales no existe un algoritmo. Un problema de este tipo es el llamado problema de paro (*halting problem*) que se basa en el siguiente desarrollo: Dado un programa de computadora con sus entradas, ¿parará éste alguna vez?; Turing probó que no hay un algoritmo que pueda resolver correctamente todas las instancias de este problema.[22]

En cuanto a lo anterior, en el presente trabajo no se va a tomar como referencia la máquina de Turing ya que esta representa un modelo matemático e ideal que explica y permite demostrar eficiencias en los algoritmos pero que está más allá de los conocimientos de un Ingeniero Electrónico y de matemáticos no especializados en el campo de la matemática computacional según lo comentó el profesor Mauricio Maca, profesor del Departamento de Matemáticas de la Universidad del Cauca; razón por la cual se harán los respectivos análisis en los casos en los que puedan ser realizados y no se pueda conseguir reportes de mediciones al respecto.

Un algoritmo es eficiente cuando logra llegar a sus objetivos planteados utilizando la menor cantidad de recursos posibles, es decir, minimizando el uso memoria, de pasos y de esfuerzo humano. Está claro que para cada algoritmo la cantidad de recursos (tiempo, memoria) utilizados depende fuertemente de los datos de entrada (tales como: la cantidad de dígitos para un número, la cantidad de elementos para un arreglo, la cantidad de caracteres de una cadena, en problemas de ordenación es el número de elementos a ordenar, en matrices puede ser el número de filas, columnas o elementos totales). En general, la cantidad de recursos crece a medida que crece el tamaño de la entrada, que es la variable en función de la cual se miden los recursos (tiempo, memoria).

3.1.1 Tipos de análisis:

3.1.1.1 Peor caso

Indica el mayor tiempo obtenido, teniendo en consideración todas las entradas posibles.

3.1.1.2 Mejor caso

Indica el menor tiempo obtenido, teniendo en consideración todas las entradas posibles.

3.1.1.3 Caso Medio

Indica el tiempo medio obtenido, considerando todas las entradas posibles.

En el mejor de los casos por lo general se asumen condiciones ideales y pocas veces se presenta de manera completa y no muestra de forma concreta el rendimiento del algoritmo.

Con el peor de los casos, se obtiene la forma como se desempeñará el algoritmo.

Si un algoritmo se ejecuta muchas veces en muchos tipos de entrada, es conveniente estudiar el comportamiento promedio o típico; desafortunadamente, esto supone que con antelación se sabe cómo están distribuidos los datos.

El caso promedio es la medida más realista del desempeño, pero es más difícil de calcular pues establece que todas las entradas son igualmente probables, lo cual puede ser cierto o no. Se analizará por lo tanto específicamente con el peor caso.[23]

3.2 Longitud de las Claves

En este criterio se determinará según estudios realizados por los criptoanalístas, los niveles de seguridad que pueden proporcionar los diferentes algoritmos criptográficos según las longitudes de clave estándar aceptadas para cada uno de ellos, razón por la cual se tendrán presentes estas tres razones:

1. Puesto que una clave más larga requiere más tiempo y esfuerzo para atacar, esta ofrece un menor riesgo, ya que las longitudes más largas de claves proporcionan una mayor seguridad al disminuir la posibilidad de ataques de búsqueda de claves al aumentar el número posible de combinaciones de éstas en el mismo algoritmo.

2. Distintos algoritmos con la misma longitud de clave (por ejemplo 128) no necesariamente proporcionan la misma seguridad.

3. Permite medir el rendimiento cuando el algoritmo ofrece la máxima seguridad, ya que para cada tamaño de la clave existe su equivalente en el tamaño de bloque a procesar, y estos resultados se asignarán a algunos tipos de documentos representativos, según la importancia de los mismos. [23]

3.3 Resistencia a Ataques (Pruebas Criptoanalíticas Suficientes)

En este criterio se desarrollarán o documentarán las pruebas que midan los niveles de seguridad de los diferentes algoritmos, tomando en cuenta la información suministrada por los criptoanálistas.

3.4 Nivel de Complejidad

Para desarrollar de la mejor manera este criterio, es necesario tener en cuenta el concepto de la teoría de la complejidad, mediante la cual se trata de conseguir una función de complejidad, la cual mide el grado de resistencia a ataques desde el punto de vista matemático, esta puede ser de recurrencia o de recursividad; las funciones de recurrencia se dan en los casos en los que el valor final deseado depende del valor

anterior, y las funciones de recursividad se desarrollan cuando el algoritmo se basa en procesos iterativos. Aunque hay que tener en cuenta que la matemática aplicada en los algoritmos criptográficos simétricos no se basa en problemas matemáticos difíciles de solucionar sino más bien en la matemática discreta y este tipo de algoritmos es más difícil de analizar debido a que aunque el número de operaciones elementales y de cifrado es constante.

En general, se dice que el orden de complejidad de un problema es el del mejor algoritmo que se conozca para resolverlo ya que en un sistema criptográfico, cuanto más complejos son los algoritmos, más difícil es vulnerar el sistema. [25]

En el análisis de algoritmos se considera usualmente el peor caso, ya que involucra mayor cantidad de procesos y da un mejor estimativo del tiempo y el espacio empleado para su implementación; aunque si bien a veces conviene analizar igualmente el caso mejor y hacer alguna estimación sobre un caso promedio. Para independizarse de factores tales como el lenguaje de programación, la habilidad del codificador, los recursos computacionales, entre otros; se suele trabajar con un cálculo asintótico que indica como se comporta el algoritmo para datos muy grandes y salvo algún coeficiente multiplicativo. Para problemas pequeños es cierto que casi todos los algoritmos son "más o menos iguales", primando otros aspectos como esfuerzo de codificación, legibilidad, etc. [25]

Este criterio se seleccionó según la versión actualizada de los criterios de Shannon, los cuales son:

1. La cantidad de seguridad deseada, determina la cantidad de trabajo y tiempo de cálculo necesario para vulnerar el mensaje cifrado.

2. Las claves utilizadas deben ser de fácil construcción, lo más cortas posibles, fáciles de alimentar, modificar y consecuentemente que ocupen poca memoria.

3. Las operaciones de cifrado y descifrado, conocida la clave, deben implicar la menor cantidad de cálculo posible.

4. Las claves y el sistema de cifrado deben ser tales que destruyan los parámetros estadísticos del lenguaje, o bien su estructura natural.

5. Los errores de transmisión en los criptogramas, no deben originar ambigüedades o pérdida del sentido en la información original, haciéndola inútil.

6. El análisis de un criptograma tratando de vulnerarlo debe necesitar una cantidad de cálculo tal, que sea considerado como un problema intratable, incluso con un computador como apoyo. [17]

Existen dos medidas básicas en la complejidad computacional, el tiempo (T) y el espacio (E), que son expresadas en función de (l), siendo l la longitud de los datos de entrada al algoritmo. Una función f(l) para expresar complejidad se representa como O(g(l)) para significar “del orden de”.

3.4.1 La O Mayúscula

La notación O se utiliza para comparar funciones. Es bastante útil cuando se quiere analizar la complejidad de un algoritmo, ya que mide la cantidad de tiempo que le toma a un computador (por lo general se toma como referencia la máquina de Turing) ejecutar todos los pasos del algoritmo.

Definición: Sean f y g funciones con dominio en **R** ≤ 0 o **N** es imagen en **R**. Si existen constantes C y k tales que:

$\forall$ x > k, | f (x) | ≤ C | g (x) |

es decir, que para x > k, f es menor o igual a un múltiplo de g, se tiene que:

f (x) = O (g (x))

La definición formal es:

f (x) = O (g (x)) □$\exists$ k , N | $\forall$ x > N, | f (x) | ≤ k| g (x) |

Lo que significa que una función es siempre menor que otra función si no tenemos en cuenta los factores constantes (que son representados por la letra k) y si no tenemos en cuenta los valores pequeños (representados por la letra N); ya que para entradas pequeñas, el tiempo que tarda el programa no es significativo y casi siempre el algoritmo será suficientemente rápido para realizar las operaciones establecidas. [27]

Dada la función f(n), se tienen las siguientes definiciones:

Límite superior asintótico: f(n) = O(g(n)) si existe una constante positiva c y un número entero positivo n_0 tales que 0 β f(n) β cg(n) $\forall$ n ρ n_0.

Límite inferior asintótico: f(n) = Ω(g(n)) si existe una constante positiva c y un número entero positivo n_0 tales que 0 β cg(n) β f(n) $\forall$n ρ n_0.

Límite exacto asintótico: f(n) = Θ(g(n)) si existen dos constantes positivas c1; c2 y un número entero positivo n_0 tales que c1g(n) βf(n) βc2g(n) $\forall$ n ρ n_0.

Notación o: f(n) = o(g(n)) si para cualquier constante positiva c existe un número entero positivo n_0 > 0 tal que 0 β f(n) β cg(n) $\forall$ n ρ n_0.

Intuitivamente, f(n) = O(g(n)) significa que f(n) crece asintóticamente no más rápido que g(n) multiplicada por una constante. Análogamente f(n) = (g(n)) quiere decir que f(n) crece asintóticamente al menos tan rápido como g(n) multiplicada por una constante.

Propiedades de la notación:

a) f(n) = O(g(n)) $\Leftrightarrow$ g(n) = Ω (f(n)).
b) f(n) = Θ(g(n)) $\Leftrightarrow$ f(n) = O(g(n)) $\wedge$f(n) = Ω (g(n)).
c) Si f(n) = O(h(n)) $\wedge$ g(n) = O(h(n)), entonces (f + g)(n) = O(h(n)).
d) Si f(n) = O(h(n)) $\wedge$g(n) = O(l(n)), entonces (f - g)(n) = O(h(n)l(n)).

e) f(n) = O(f(n)).
f) Si f(n) = O(g(n)) $\wedge$g(n) = O(h(n)), entonces f(n) = O(h(n)).

Para algunas funciones de uso común, se puede definir directamente su orden de complejidad:

Funciones polinomiales: Si f(n) es un polinomio de grado k, y su coeficiente de mayor grado es positivo, entonces $f(n) = \Theta(n^k)$.

Funciones logarítmicas: Para cualquier constante c > 0, $\log_c(n) = \Theta(\ln(n))$.

Factoriales: $n! = \Omega(2^n)$.

Logaritmo de un factorial: $\ln(n!) = \Theta\ (n \ln(n))$.

Un concepto a considerar es el de operación elemental, la cual es aquella que se ejecuta siempre en tiempo constante. Lo anterior se toma en función de las características concretas del computador que se esté manejando, ya que habrá operaciones que podrán considerarse elementales o no. Por ejemplo, en una computadora que pueda operar únicamente con números de 16 bits, no podrá considerarse elemental una operación con números de 32 bits.

En general, el tamaño de la entrada a un algoritmo se mide en bits, y se consideran en principio elementales únicamente las operaciones a nivel de bit. Sean a y b dos números enteros positivos, ambos menores o iguales que n. Se necesitan aproximadamente $\log_2(n)$ bits (en este caso, $\log_2(n)$ es el tamaño de la entrada) para representarlos. Según este criterio, las operaciones aritméticas, llevadas a cabo mediante los algoritmos tradicionales, presentan los siguientes órdenes de complejidad:

Suma (a + b): $O(\log_2(a) + \log_2(b)) = O(\log_2(n))$

Multiplicación (a · b): $O(\log_2(a) \cdot \log_2(b)) = O((\log_2(n))^2)$

El orden de complejidad de un logaritmo es independiente de la base, por lo que la capacidad de realizar en tiempo constante operaciones aritméticas con números de más bits únicamente introducirá factor de proporcionalidad (por ejemplo $\log_a(x) = \log_b(x) \cdot \log_a(b)$) Dicho factor no afectará al orden de complejidad obtenido, por lo que se puede considerar que estas operaciones se efectúan en grupos de bits de tamaño arbitrario. La medida con una magnitud de las necesidades de tiempo y espacio de un algoritmo por el orden de complejidad, tiene la gran ventaja de ser independiente del sistema ordenador en que se implante. [27]

Un algoritmo es de complejidad polinomial, si el número de operaciones que necesita efectuar es del tipo: T = O(t) siendo t una constante que según la potencia con valores de 0, 1, 2, 3 dará el nombre de constante lineal, cuadrática, cúbica, etc.

Si el algoritmo es exponencial, entonces T = O(th(l)) siendo t una constante y h(l) una función polinómica de l. [26]

3.4.2 Órdenes de Complejidad

Se dice que O(f(n)) define el "orden de complejidad". A nivel general se tienen los siguientes ordenes (se toma f(n) como una función representativa de cada orden):

O-Notación	Tiempo de ejecución	Clase
$O(1)$	Constante	Algunos algoritmos de búsqueda en tabla
$O(\log n)$	Logaritmico	Algoritmo de búsqueda (Búsqueda Binaria)
$O(n)$	Lineal	Búsqueda secuencial , Búsqueda en texto.
$O(n \cdot \log n)$		„sclaues Sortieren“, z.B. Quicksort
$O(n^2)$	Cuadrático	Algunos algoritmos de ordenamiento: Bubble-Sort, QuickSort(caso peor).
$O(n^k)$, si $k \geq 0$		Multiplicación de Matrices
$O(2^n)$	Exponencial	Muchos problemas de optimización, por lo general en “fuerza bruta”
$O(n!)$		Todas las Permutaciones

Tabla No. 2 Órdenes de Complejidad [27]

El orden de complejidad se expresa generalmente en términos de la cantidad de datos procesados por el Algoritmo, denominada n, que puede ser el tamaño dado o estimado.

Un ejemplo de algunas de las funciones más comunes en análisis de algoritmos son:

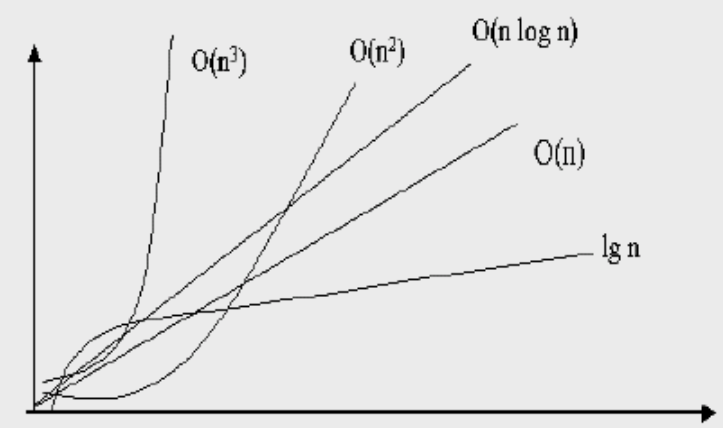

Figura No. 2 Algunas Funciones O [27]

Es de destacar la mutua interacción y el parecido que tienen los criterios de eficiencia en los recursos computacionales y el del nivel de complejidad; podrían llegar a confundirse debido a que utilizan las mismas herramientas matemáticas, pero la razón principal para su separación radica en que el primer criterio busca analizar los algoritmos criptográficos desde el punto de vista del hardware asociado como medio para establecer su eficiencia y/o vulnerabilidad y sin necesidad de tener los conocimientos matemáticos suficientes (por

lo que se realizan las pruebas según las características comunes de las máquinas del piloto de pruebas); mientras que el segundo criterio busca establecer que tanta dificultad (a nivel matemático) le representa a un posible atacante que conozca el algoritmo, el poder encontrar medios que le permitan violar la seguridad de este, teniendo como principal fuente de mediciones el tamaño de los datos de entrada y por ende es más de carácter teórico. [27]

3.4.3 Notación Asintótica

El interés principal del análisis de algoritmos radica en saber cómo crece el tiempo de ejecución, cuando el tamaño de la entrada crece. Esto es la eficiencia asintótica del algoritmo. Se denomina “asintótica” porque analiza el comportamiento de las funciones en el límite, es decir, su tasa de crecimiento.

La notación asintótica se describe por medio de una función cuyo dominio son los números naturales (N) estimado a partir de tiempo de ejecución o de espacio de memoria de algoritmos en base a la longitud de la entrada. Se consideran las funciones asintóticamente no negativas. [27]

La notación asintótica captura el comportamiento de la función para valores grandes de N.

3.4.4 Complejidad Computacional

Se define la complejidad computacional como el tiempo que tarda un computador en realizar una tarea (tiempo que tarda en procesar un algoritmo). Este concepto se tendrá en cuenta en los algoritmos en los que pueda ser calculado.

En criptografía es importante desde dos puntos de vista:

1. Interesa que los algoritmos de cifrado/descifrado y elección de claves requieran poco tiempo de cálculo.

2. Cualquier algoritmo de cifrado, suponiendo no conocida la clave, requiera un alto coste computacional.

El coste computacional de un algoritmo depende de:

1. El propio algoritmo, número de operaciones a realizar y “que” operaciones tiene que realizar.

2. El computador sobre el que se esta ejecutando el algoritmo.

Se define como coste computacional a una función que depende del tamaño de la entrada. Para un tamaño n entenderemos por coste computacional el peor de todos los casos posibles.

Los computadores trabajan con bits, bytes y por esta razón, para calcular el coste computacional de una operación se calcula el número de operaciones, a nivel de bits, que son necesarias para realizarla.

Sea $n \in \zeta$ y sea $a_r a_{r-1} \ldots a_1 a_0$ su expresión en código binario, en donde:

$$a_r a_{r-1} \ldots a_1 a_0 = \sum^{r}_{i=0} a_i \cdot 2^i = n \text{ , donde los } a_i\ 2 \in \{ 0,1 \}$$

$$r = [\log_2 n]+1 = [\log n/\log 2]+1$$

Con los algoritmos clásicos de suma y multiplicación se tiene que para sumar dos números de k bits se necesita realizar $2 \cdot k$ operaciones, y se dice que el coste computacional de ese algoritmo es $O(2 \cdot k)$ y para multiplicar dos enteros de m y n bits se necesitan realizar $m \cdot n$ operaciones, y su coste computacional es $O(m \cdot n)$. [26]

En la mayoría de los casos carece de interés calcular el tiempo de ejecución concreto de un algoritmo en una computadora, e incluso algunas veces simplemente resulta imposible. En su lugar se emplea una notación de tipo asintótico, que permitirá acotar dicha magnitud. Normalmente se considera el tiempo de ejecución del algoritmo como una función f(n) del tamaño n de la entrada. Por lo tanto f debe estar definida para los números naturales y devolver valores en R+. [17] [29]

Por lo anterior, es necesario conocer las principales clases de problema matemático que se utilizan en los algoritmos para evitar ser descifrados:

3.4.5 Problema Clase P

Los algoritmos de complejidad polinómica se dice que son tratables en el sentido de que suelen ser abordables en la práctica. Un algoritmo se dice que tiene tiempo de ejecución polinomial (no es lo mismo que lineal aunque tiene un comportamiento similar) si éste depende polinómicamente del tamaño de la entrada. Los problemas para los cuales se conocen algoritmos con esta complejidad se dice que son de clase P. Aquellos problemas para los que la mejor solución que se conoce es de complejidad superior a la polinómica, se dice que son problemas intratables. [27]

3.4.6 Problema Clase NP

Estos se caracterizan por tener la particularidad de que es fácil calcular f(x), pero es muy difícil calcular $f^{-1}(x)$ salvo que se conozca un secreto o trampa. Un algoritmo se dice que tiene tiempo de ejecución polinomial no determinista si éste depende exponencialmente del tamaño de la entrada. Algunos de estos problemas intratables pueden caracterizarse por el curioso hecho de que puede aplicarse un algoritmo polinómico para comprobar si una posible solución es válida o no. Esta característica lleva a un método de resolución no determinista consistente en aplicar técnicas heurísticas (de indagación y descubrimiento) para obtener soluciones hipotéticas que se van desestimando (o aceptando) a ritmo polinómico. Los problemas de esta clase se denominan NP (la N de no-deterministas y la P de polinómicos). [27]

3.4.7 Problemas Clase NP completos

Se conoce una amplia variedad de problemas de tipo NP, de los cuales destacan algunos de ellos de extrema complejidad. Estos problemas se caracterizan por ser similares, ya que si se descubriera una solución P para alguno de ellos, esta solución sería fácilmente aplicable a todos ellos; y si se descubriera una solución para los problemas NP-completos, esta sería aplicable a todos los problemas NP y, por tanto, ya no habrían problemas de este tipo. En síntesis lo anterior significa que no hay un algoritmo determinístico de complejidad polinomial que lo resuelva, y los algoritmos que lo podrían resolver son de complejidad exponencial y requerirían muchos años (o siglos incluso) para el cálculo de entradas moderadamente grandes, con la tecnología actual. [27]

Aunque si bien es cierto, la eficiencia de un determinado algoritmo depende de la maquina, y de otros factores externos al propio diseño. Para comparar dos algoritmos sin tener en cuenta estos factores externos se usa la complejidad. Esta es una media informativa del tiempo de ejecución de un algoritmo, y depende de varios factores:

- Los datos de entrada del programa. Dentro de ellos, lo más importante es la cantidad, su disposición, etc.
- La calidad del código generado por el compilador utilizado para crear el programa.
- La naturaleza y rapidez de las instrucciones empleados por la máquina y la propia máquina.
- La propia complejidad del algoritmo base del programa.

El hecho de que el tiempo de ejecución dependa de la entrada, indica que el tiempo de ejecución del programa debe definirse como una función de la entrada. En general la longitud de la entrada es una medida apropiada de tamaño, y es por lo general la más utilizada.

Se acostumbra, pues, a denominar T(n) al tiempo de ejecución de un algoritmo en función de n datos de entrada.

Hay programas que tienen características de tiempo de ejecución similares a esta: $T(n)=Cn^2$, donde C es una constante que engloba las características de la máquina entre otros factores.

Las unidades de T(n) se dejan sin especificar, pero se puede considerar a T(n) como el número de instrucciones ejecutadas en un computador idealizado, y es lo que por lo general se entiende por complejidad.
Para muchos programas, el tiempo de ejecución es en realidad una función de la entrada especifica, y no sólo del tamaño de ella, pero siempre se define T(n) como el tiempo de ejecución del peor caso, es decir, el máximo valor del tiempo de ejecución para las entradas de tamaño n. [27]

En síntesis, se tiene que:

- Un algoritmo es de orden polinomial si T(n) crece más despacio (a medida que aumenta n), que un polinomio de grado n; y los algoritmos de este tipo se pueden ejecutar en un computador.

- En caso contrario el algoritmo es exponencial, y estos no son ejecutables en un computador.

3.4.8 Algoritmos Probabilísticos

Los problemas que se han tratado hasta el momento miden la complejidad de algoritmos de tipo determinístico, es decir, que siempre siguen el mismo camino de ejecución y que siempre llegan —si lo hacen— a la misma solución.

Sin embargo, existen problemas para los cuales puede ser más interesante emplear algoritmos de tipo no determinístico, también llamados probabilísticos o aleatorizados.

Este tipo de algoritmos maneja algún tipo de parámetro aleatorio, lo cual hace que dos ejecuciones diferentes con los mismos datos de entrada no tengan por que ser idénticas. En algunos casos, los métodos de este tipo permiten obtener soluciones en una cantidad de tiempo considerablemente inferior a la necesaria, que si se emplean algoritmos determinísticos. Se pueden clasificar los algoritmos no determinísticos según la probabilidad con la que devuelvan la solución correcta.

Sea A un algoritmo aleatorizado para el problema de decisión L, y sea I una instancia arbitraria de L. Sea P1 la probabilidad de que A devuelva cierto cuando I es cierto, y P2 la probabilidad de que A devuelva cierto cuando I es falso. [26]

- A es de tipo error nulo si P1 = 1 y P2 = 0.
- A es de tipo error simple si P1 ρ c, siendo c una constante positiva, y P2 = 0
- A es de tipo error doble si P1 ρ ½ + ε y P2 β ½ - ε.

Definiremos también el tiempo esperado de ejecución de un algoritmo aleatorizado como el límite superior del tiempo de ejecución esperado para cada entrada, expresado en función del tamaño de la entrada. El tiempo de ejecución esperado para cada entrada será la media de los tiempos obtenidos para esa entrada y todas las posibles salidas del generador aleatorio. [26]

Las clases de complejidad probabilística son:

3.4.8.1 Clase ZPP

Es el conjunto de todos los problemas de decisión para los cuales existe un algoritmo de tipo error nulo que se ejecuta en un tiempo esperado de ejecución polinomial.

3.4.8.2 Clase RP

Es el conjunto de los problemas de decisión para los cuales existe un algoritmo de tipo error simple que se ejecuta en el peor caso en tiempo polinomial.

3.4.8.3 Clase BPP

Es el conjunto de los problemas de decisión para los cuales existe un algoritmo de tipo error doble que se ejecuta en el peor caso en tiempo polinomial. [26]

Finalmente, se dice que $P \subseteq ZPP \subseteq RP \subseteq BPP$ y $RP \subseteq NP$.

3.5 Patentes

Este criterio determina la capacidad de uso de determinado algoritmo y depende no solo de la propiedad intelectual de su autor sino también de las leyes de los diferentes países que permiten su o regulan su uso.

Capítulo 4

Análisis Comparativo de los Algoritmos Criptográficos de Mayor Relevancia en las Aplicaciones Actuales de Seguridad

4.1 Introducción

Antes de empezar con el análisis de los algoritmos es necesario tener en cuenta algunas definiciones que ayudarán a entender la aplicación de los criterios; en primera instancia se tiene que un algoritmo es todo proceso bien definido que a partir de una serie de valores de entrada proporciona un conjunto de valores de salida. En segunda instancia se tiene su clasificación general: un algoritmo es determinístico si el valor de entrada determina por completo al de salida. Si por el contrario, al ejecutar el algoritmo con el mismo valor de entrada pueden obtenerse distintos valores de salida, el algoritmo es probabilístico o aleatorio. Y por último, sus objetivos básicos son:

- Un algoritmo debe ser fácil de entender, codificar y depurar (Ingeniería de Software).
- Un algoritmo debe hacer uso eficiente de los recursos de la computadora (Análisis y Diseño de algoritmos). [17]

En la siguiente figura se presentará la clasificación más generalizada de los algoritmos modernos:

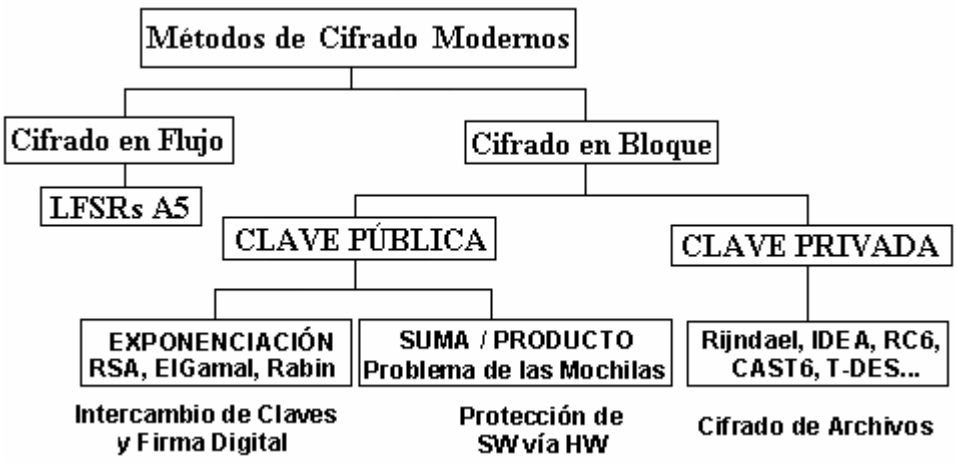

Figura No. 3 Clasificación de los Criptosistemas

Otros factores a tener en cuenta en los algoritmos criptográficos son las siguientes:

4.1.1 Características de un algoritmo de Cifrado

Algunos algoritmos utilizan las llamadas redes de Feistel, para entender la complejidad de las mismas es necesario examinar los siguientes parámetros de diseño:

- Tamaño de bloque. Bloques más grandes implican mayor seguridad pero decrece la velocidad de cifrado/descifrado. Se utilizan en todos los cifradores por bloques, conjuntos de datos de 64 bits de tamaño.

- Tamaño de la clave. Claves de tamaño más largo significan mayor seguridad, pero podrían causar menor velocidad de cifrado y descifrado. La longitud mas empleada en las claves son 128 bits.

- Número de vueltas (rondas). La esencia del cifrador Feistel es que en una sola vuelta ofrece seguridad poco adecuada, pero en múltiples vueltas ofrece mejor seguridad, es típico que se empleen 16 vueltas.

- Algoritmo de generación de sub-claves. Una mayor complejidad en el algoritmo implica mayor dificultad de criptoanálisis.

- Función de redondeo. Al igual que el algoritmo de sub-claves, mayor complejidad genera mayor resistencia al criptoanálisis.

Estas redes de Feistel son más seguras si trabajan con ciclos mayores que dos y tienen como principal característica que el recuperar su estado interno resulta ser un problema NP-completo. La estructura general para una red Feistel de cuatro ciclos, por ejemplo, es la siguiente:

$$R = R + F1(L)$$
$$L - L + F2(R)$$
$$R = R + F3(L)$$
$$L = L + F4(R)$$

Donde R y L son la mitad derecha e izquierda del bloque, respectivamente. F1, F2, F3 y F4 son funciones pseudoaleatorias (secretas) y el símbolo + representa la operación que se va a aplicar, usualmente el módulo 2n o el XOR. [28]

4.1.2 Consideraciones de los cifradores Feistel:

- El cifrado y descifrado por software debe ser rápido, ya que hay muchas utilidades que obstaculizan el buen desarrollo del algoritmo, para lograr una buena rapidez los expertos recomiendan usar hardware asociado como las FPGA´s las cuales además de mejorar el desempeño y de hacer independiente al algoritmo, también lo hacen más seguro a nivel de redes, ya que es muy posible que el futuro atacante desconozca la existencia del mismo.

- Facilidad de analizar. A pesar de que se desea que los algoritmos sean lo más difíciles de criptoanalizar que sea posible, es un gran beneficio que el algoritmo sea fácil de analizar ya que se puede explicar claramente y analizar si tiene vulnerabilidades de criptoanálisis y como consecuencia desarrollar niveles de fortaleza más altos. DES por ejemplo no es fácil de analizar.

- Los cifradores Feistel en el proceso de descifrado hacen esencialmente lo mismo que al cifrar, sucede lo siguiente: Se utiliza el texto cifrado como entrada del algoritmo, pero las subllaves K_i en orden inverso, por lo tanto utilizar K_n en el primer bucle, K_{n-1} en el segundo y así hasta llegar hasta K1 que es utilizado en la última vuelta. Esto es muy favorable ya que implica no tener que implementar dos algoritmos diferentes para el cifrado y descifrado.

- La estructura de Feistel opera en bloques de tamaño fijo (por ejemplo 64 bits), utiliza una clave secreta única y una clave compartida (de 56, 64, 128 ó 256 bits) y generalmente involucra varias rondas (de 8 a 32) en simples funciones no-lineales, en donde a la salida se le aplica la función XOR. [28]

4.1.3 Modos de Operación para Algoritmos Cifrados por Bloques:

Independientemente del método empleado para codificar, se ha de tener en cuenta lo que ocurre cuando la longitud de la cadena que se quiere cifrar no es un múltiplo exacto del tamaño de bloque; entonces se debe añadir información al final para que sí lo sea, el mecanismo más sencillo consiste en rellenar con ceros (o algún otro patrón) el último bloque que se codifica. El problema consiste en saber cuando se descifra el corte del bloque; lo que se suele hacer es añadir como último byte del último bloque el número de bytes que se han añadido. Esto tiene el inconveniente de que si el tamaño original es múltiplo del bloque, hay que alargarlo con otro bloque entero. Por ejemplo, si el tamaño de bloque fuera 64 bits, y sobraran cinco bytes al final, se añadirían dos ceros y un tres. Si por el contrario no sobrara nada, se tendría que añadir siete ceros y un ocho.

4.1.3.1 Modo ECB

El modo ECB (Electronic Codebook) es el método más sencillo y obvio de aplicar un algoritmo de cifrado por bloques. Simplemente se subdivide la cadena que se quiere codficar en bloques del tamaño adecuado y se cifran todos ellos empleando la misma clave. Entre las ventajas de este método están el hecho de que permite codificar los bloques independientemente de su orden, lo cual es adecuado para codificar bases de datos o archivos en los que se requiera un acceso aleatorio; y que es resistente a errores, pues si uno de los bloques sufriera una alteración, el resto quedaría intacto. Como desventaja se tiene que si el mensaje presenta patrones repetitivos, el texto cifrado también los presentará, y eso es peligroso, sobre todo cuando se codifica información muy redundante (como archivos de texto), o con patrones comunes al inicio y final (como el correo electrónico). Un contrincante puede en estos casos efectuar un ataque estadístico y extraer bastante información.[29]

Otro riesgo bastante importante que presenta el modo ECB es el de la sustitución de bloques. El atacante puede cambiar un bloque sin mayores problemas, y alterar los mensajes incluso desconociendo la clave y el algoritmo empleados.

4.1.3.2 Modo CBC

El modo CBC (Cipher Book Chaining Mode) incorpora un mecanismo de retroalimentación en el cifrado por bloques. Esto significa que la codificación de bloques anteriores condiciona la codificación del bloque actual, por lo que será imposible sustituir un bloque individual en el mensaje cifrado. Esto se consigue efectuando una operación XOR entre el bloque del mensaje que queremos codificar y el último criptograma obtenido. En cualquier caso, dos mensajes idénticos se codificarán de la misma forma usando el modo CBC. Más aún, dos mensajes que empiecen igual se codificarán igual hasta llegar a la primera diferencia entre ellos.

Para evitar esto se emplea un vector de inicialización, que puede ser un bloque aleatorio, como bloque inicial de la transmisión. Este vector será descartado en destino, pero garantiza que siempre los mensajes se codifiquen de manera diferente, aunque tengan partes comunes. [29]

4.1.3.3 Modo CFB y OFB

El modo CBC no empieza a codificar (o decodificar) hasta que no se tiene que transmitir (o se ha recibido) un bloque completo de información. Esta circunstancia puede convertirse en un serio inconveniente, por ejemplo en el caso de terminales, que deban transmitir cada caracter que pulsa el usuario de manera individual.

Una posible solución sería emplear un bloque completo para transmitir cada byte y rellenar el resto con ceros, pero esto haría que tengamos únicamente 256 mensajes diferentes en nuestra transmisión y que un atacante pueda efectuar un sencillo análisis estadístico para comprometerla. Otra opción sería rellenar el bloque con información aleatoria, aunque se seguiría desperdiciando gran parte del ancho de banda de la transmisión. El modo de operación CFB (Cipher-Feedback Mode) permite codificar la información en unidades inferiores al tamaño del bloque, lo cual permite aprovechar totalmente la capacidad de transmisión del canal de comunicaciones, manteniendo además un nivel de seguridad adecuado.[29]

Sea p el tamaño de bloque del algoritmo simétrico, y sea n el tamaño de los bloques que se desean transmitir (n ha de divisor de p). Sea m_i el i-esimo bloque del texto plano, de tamaño n. Se emplea entonces un registro de desplazamiento R de logitud p y se carga con un vector de inicialización. Se codifica el registro R con el algoritmo simétrico y se obtiene en r sus n bits más a la izquierda. El bloque que se debe enviar es $c_i = r \oplus m_i$. Se desplaza R n bits a la izquierda y se introduce c_i por la derecha.[29]

Para descifrar basta con cargar el vector de inicialización en R y codificarlo, calculando r. Entonces $m_i = r \oplus c_i$. Se desplaza luego R y se introduce c_i por la derecha.

si n = p, el modo CFB queda reducido al modo CBC.

Los modos CFB y OFB (output feedback) hacen que el cifrado en bloque opere como una unidad de flujo de cifrado: se generan bloques de flujo de claves, que son operados con XOR y el texto en claro para obtener el texto cifrado. Al igual que con otras unidades de

flujo de cifrado, al intercambiar un bit en el texto cifrado produce texto cifrado con un bit intercambiado en el texto en claro en la misma ubicación. [29]

4.1.3.4 Modo Counter (CTR)

Al igual que OFB, el modo contador convierte una unidad de cifrado por bloques en una unidad de flujo de cifrado. Genera el siguiente bloque en el flujo de claves cifrando valores sucesivos de un contador. El contador puede ser cualquier función sencilla que produzca una secuencia de números donde los resultados se repiten con muy baja frecuencia, si bien la operación más usada es un contador, el modo CTR tiene características similares al OFB, pero permite también el uso de una propiedad de acceso aleatorio para el descifrado. [29]

4.1.4 Consideraciones y Conceptos Matemáticos

La manera de contar operaciones elementales (explicada por el profesor de Matemáticas Mauricio Maca de la Universidad del Cauca), consta de los siguientes pasos: 1. Se toma el peor caso, es decir, cuando el algoritmo hace todos los procesos posibles. 2. Se enumeran las líneas de código y se empieza desde los procesos más internos a los más externos. 3. Se asigna a cada operación elemental (comparación, suma, resta, multiplicación, división, asignación, incremento, decremento) el valor de 1 (por ejemplo, cada bucle for tiene como mínimo un valor de 3 por las operaciones de asignación, comparación e incremento/decremento). 4. Se suman lo valores por línea de pseudocódigo y se deja en términos de la variable implicada en cada bucle, hasta que se llega al bucle más externo que por lo general está en términos de n; se debe tomar en cuenta que al valor final de cada bucle se le adiciona el valor correspondiente a la última comparación realizada para terminar el ciclo, es decir que los bucles siempre harán una vez más el mismo número de operaciones debido a que internamente necesitan desbordarse para terminar el proceso.

En general para desarrollar la función de complejidad propia de cada algoritmo simétrico, se debe tener en cuenta que para cada problema se debe determinar una medida **n** de su tamaño (por número de datos) para hallar respuestas en función de dicho **n**. El concepto exacto que mide **n** depende de la naturaleza del problema. Así, para un vector se suele utilizar como **n** su longitud; para una matriz, el número de elementos que la componen; para un grafo, puede ser el número de nodos, etc.

Varias operaciones en algunos algoritmos están definidas al nivel de bytes, tratados como elementos del campo finito $F_2{}^8$, es decir un byte $(b_7b_6b_5b_4b_3b_2b_1b_0)$ es considerado como un polinomio f(x) con coeficientes en $F_2 = Z_2 = \{0, 1\}$:

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0. \Leftrightarrow (b_7b_6b_5b_4b_3b_2b_1b_0)$$

En notación hexadecimal: $(01011110) = x^6 + x^4 + x^3 + x^2 + x = \{5E\}$.

Sumar dos polinomios es sumar sus coeficientes módulo 2, (or-exclusiva) de dos bytes: $(x^6+x^4+x^3+x^2+x)+(x^7+x^4+x^0+1) = x^7+x^6+x^2+x+1 \Leftrightarrow (01011110) \oplus (10011001) = (11000111)$.

La multiplicación en el cuerpo $F_2{}^8$ corresponde con la multiplicación de polinomios módulo un polinomio irreducible de grado 8.

El polinomio propuesto por Rijndael (AES) es : $m(x) = x^8 + x^4 + x^3 + x + 1$.

Multiplicar dos polinomios f(x) y g(x) módulo el polinomio m(x), consiste en tomar el resto de la división de f(x)g(x) entre m(x).

Cada elemento f(x) tiene inverso $f(x)^{-1}$ en el cuerpo $F_2^8 = F_2[x]/m(x)$. Para determinar este inverso, se utiliza el algoritmo extendido de Euclides a los polinomios f(x) y m(x) obteniendo $f(x)g(x)+m(x)n(x) = 1$, $f(x)g(x) = 1 \bmod m(x)$, $g(x) = f(x)^{-1}$.

La multiplicación por x es muy simple y útil: todas las multiplicaciones entre polinomios pueden ser expresadas por éstas y por or-exclusivas.

Por ejemplo, $f(x)(x^2 + 1) = x(x(f(x))) \oplus f(x)$.

Si $b_7 = 0$, la multiplicación es un desplazamiento a la izquierda: x(01101110) = (11011100).

En cambio si $b_7 = 1$, es un desplazamiento a la izquierda seguido de una or-exclusiva con (00011011) = {1B},
x(10101100) = (01011001) ⊕ (00011011) = (01000001).

Una palabra es un fila de 4 bytes, que es tratada como un polinomio de grado 3 en F_2^8: $[a_3a_2a_1a_0] \rightarrow a_3y^3 + a_2y^2 + a_1y + a_0$, donde los a_i son bytes.

Al sumar dos palabras se suman los coeficientes (or-exclusivo).

La multiplicación de dos palabras se reduce al módulo de un polinomio de grado 4.

Otra de las operaciones que se repetirán varias veces es la multiplicación de enteros grandes, por lo que es necesario comprobar que dicha operación se puede hacer de manera eficaz; la multiplicación por el método estándar, de dos enteros binarios de longitudes k y l precisa un máximo de kl sumas de un bit (operaciones bit), por lo tanto el producto de dos números enteros con a lo sumo r cifras decimales requiere a lo más $\log_2 r$ operaciones, es decir, es del orden $O(\log_2 r)$ y por lo tanto es una operación eficiente. [24]

Actualmente hay algoritmos de multiplicación más rápidos, el más conocido es el método de Karatsuba que permite multiplicar dos enteros de longitud binaria menor o igual que k en $O(k^{1,59})$ operaciones bit. El método más rápido se debe a Schönhage y Strassen y su complejidad temporal es O(k log k log log k).[24]

Dado que el tiempo de la multiplicación está ya estimado, entonces al repetir muchas veces dicha operación se denota M(r) la complejidad de multiplicar dos enteros menores o iguales que r. La complejidad de la división es la misma que la del producto, es decir, la división requiere también O(M(r)) operaciones. Por otro lado la conversión de binario a decimal (o viceversa) de un entero de longitud r es también es de complejidad O(M(r)).[24]

4.1.5 Tipos de Criptoanálisis Más Empleados

4.1.5.1 Criptoanálisis Diferencial

Descubierto por Biham y Shamir en 1990, permite efectuar un ataque de texto claro escogido a DES que resulta más eficiente que la fuerza bruta. Se basa en el estudio de los pares de criptogramas que surgen cuando se codifican dos textos claros con diferencias particulares, analizando la evolución de dichas diferencias a lo largo de las rondas de DES.

Para llevar a cabo un criptoanálisis diferencial se toman dos mensajes cualesquiera (incluso aleatorios) idénticos salvo en un número concreto de bits. Usando las diferencias entre los textos cifrados, se asignan probabilidades a las diferentes claves de cifrado. Conforme tenemos más y más pares, una de las claves aparece como la más probable. Esa será la clave buscada. [29]

4.1.5.2 Criptoanálisis Lineal

El criptoanálisis lineal, descubierto por Mitsuru Matsui, basa su funcionamiento en tomar algunos bits del texto claro y efectuar una operación XOR entre ellos, tomar algunos del texto cifrado y hacerles lo mismo, y finalmente hacer un XOR de los dos resultados anteriores, obteniendo un único bit. Efectuando esa operación a una gran cantidad de pares de texto claro y criptograma diferentes podemos ver si se obtienen más ceros o más unos.

Si el algoritmo criptográfico en cuestión es vulnerable a este tipo de ataque, existirán combinaciones de bits que, bien escogidas, den lugar a un sesgo significativo en la medida anteriormente definida, es decir, que el número de ceros (o unos) es apreciablemente superior .

Esta propiedad permite poder asignar mayor probabilidad a unas claves sobre otras y de esta forma descubrir la clave buscada. [29]

4.1.6 Algoritmos No Analizados

A continuación se muestran algunos algoritmos criptográficos conocidos, los cuales no analizaremos en detalle debido a su probada debilidad frente a diferentes tipos de ataques, estos son:

- MD2 (Message Digest 2). Se diseñó para computadores con procesador de 8 bits, y hoy apenas se utiliza. Se conocen ataques a versiones parciales de MD2.

- MD4 (Message Digest 4). Fue desarrollado por Ron Rivest, de RSA Data Security. Su diseño es la base de otros algoritmos de hash, aunque se le considera inseguro. Un ataque desarrollado por Hans Dobbertin permite generar colisiones (mensajes aleatorios con los mismos valores de la función hash) en cuestión de minutos para cualquier PC. Por ese motivo, está en desuso.

- MD5 (Message Digest 5). También fue creado por Ron Rivest. Hasta hace poco se consideraba un algoritmo de hash seguro, hasta tal punto que es el utilizado en el programa de cifrado PGP para firmar mensajes con claves de tipo RSA; también se utiliza como autentificador de mensajes en el protocolo SSL. Sin embargo, en 1996 el mismo Hans Dobbertin que logró romper la seguridad del MD4 consiguió demostrar que la función de compresión del algoritmo MD5 es insegura,

consiguiendo colisiones en ese caso. Sus ataques son parciales y no pueden extenderse a la totalidad del algoritmo MD5. Asimismo hay que resaltar que estos ataques comprometían la propiedad de no-colisión; esto significa que se puede obtener una función hash idéntica para dos mensajes obtenidos al azar, pero queda por demostrar que se pueda encontrar un mensaje con igual función hash a otro mensaje concreto; es decir, la reseña del mensaje que se ha firmado sigue siendo única. Por lo anterior, es inquietante que un algoritmo de hash considerado seguro tenga tales puntos débiles, hasta el punto en que no se recomienda su uso para generar firmas. Trabajos ulteriores en este campo podrían romperlo completamente, permitiendo que las firmas sean falsificadas. [27]

- El algoritmo Magenta no se analizará puesto que en el encuentro celebrado en Agosto de 1998 en el cual se presentó, fue criptoanalizado ese día dejando en entredicho su seguridad. [30]

- En noticias recientes publicadas en la página de Microsoft, se comenta que en dicha empresa se está estableciendo una política de desarrollo que prohíbe a sus programadores el uso de los algoritmos DES, MD4, MD5 y algunas implementaciones de SHA, hasta el punto de que, en caso de utilizarlos, serán marcados por los sistemas automáticos de escaneo de código para ser sustituidos de inmediato, como sustitutos más fiables, Microsoft recomienda el uso SHA-256 y AES, entre otros. Al conocer esta nueva política de desarrollo Bruce Schneier ha afirmado que Microsoft debería haber desechado MD4, MD5 y DES hace años.

- Los llamados Algoritmos Simétricos de Flujo o Stream Ciphers no se analizarán debido a que no son tan confiables por su misma estructura, ya que sin importar la clave pseudoaleatoria que se emplee, se puede mediante ataques de estadística (de las letras más usadas en los diferentes idiomas) quebrantar su seguridad por lo que los cifradores de bloque son más seguros.

Otro tipo de algoritmos que no se estudiarán en el presente trabajo son los de curva elíptica, debido a que en una consulta hecha al Doctor Julio López de la Universidad de Campinhas Brasil, se concluyó que es demasiada carga computacional para un sistema de transporte de documentos digitales el utilizar esta clase de algoritmos, ya que su implementación requiere además de hardware adicional para garantizar su fortaleza, a lo que se adiciona que es demasiada seguridad para algo que no lo requiere, comparando sería como transportar algo muy pequeño en un contenedor gigante, la relación es desproporcionada; además no se encuentra muy difundido por ahora entre los principales productos de seguridad.

Tampoco se analizarán los algoritmos cuánticos, ya que según el Ingeniero Físico Hernando Caicedo (universidad del Cauca) cuyo trabajo de grado se basó en este tipo de algoritmos, estos están aún en teoría y aunque se cree que son insuperables, la tecnología capaz de soportarlos no se encuentra desarrollada a plenitud hoy día.

Se tendrá especial interés por algunos de los algoritmos finalistas para el estándar AES, ya que son algoritmos bien probados por criptoanalistas de todo el mundo, realizados por doctores con años de experiencia y además revisados por algunos de ellos bajo supervisión del NIST.

4.2 ANÁLISIS DE LOS ALGORITMOS SIMÉTRICOS

Para establecer una jerarquía de algoritmos en orden de mayor a menor eficiencia, se consultó a algunos doctores en la materia, como Dr. Jorge Ramió Aguirre, Dr. José de Jesús Angel, Dr. Santiago Felici y Dr. Arturo Quirantes, los cuales basados en su larga experiencia en el manejo y conocimiento de artículos de referencia así lo recomendaron.

Es importante tener en cuenta que para este tipo de algoritmos criptográficos, se tiene que usan la misma clave para cifrar y para descifrar y por la misma razón en una comunicación se tiene que el mayor problema empleando solo este tipo de algoritmos es el acuerdo de la clave k y según el Profesor Leobardo Hernández Audelo del Laboratorio de seguridad informática del Centro Tecnológico Aragón de la UNAM (Universidad Nacional Autónoma de México) la complejidad del Acuerdo de la clave k es del orden de $O(n^2)$ siendo n, la cantidad de datos procesados.

Los algoritmos han sido divididos como buenos o malos algoritmos. La comunidad computacional acepta que un buen algoritmo es aquel para el cual existe un algoritmo polinomial determinístico que lo resuelva y si no existe tal algoritmo polinomial se considera mal algoritmo.

4.2.1 ANÁLISIS PRELIMINAR DE LOS FINALISTAS DEL AES

En la segunda ronda eliminatoria de los candidatos al AES, se realizaron las siguientes pruebas:

		RC6	Rijndael	MARS	Twofish	CAST - 256
Establecimiento de Clave	-128	1632	305:1389	4216	8414	4333
	-192	1885	277:1595	4277	11628	4342
	-256	1877	374:1960	4140	15457	4325
Cifrado	-128	270	374	369	376	633
	-192	267	439	373	376	633
	-256	270	502	369	381	639
Descifrado	-128	226	352	376	374	634
	-192	235	425	379	374	633
	-256	227	500	376	374	638
		Serpent	SAFER	Loki97	DEAL	MAGENTA
Establecimiento de Clave	-128	2402	4278	7430	8635	30
	-192	2449	7426	7303	8653	25
	-256	2349	11313	7166	11698	37
Cifrado	-128	952	1722	2134	2339	6539
	-192	952	2555	2138	2358	6531
	-256	952	3391	2131	3115	8711
Descifrado	-128	914	1709	2192	2365	6534
	-192	914	2530	2189	2363	6528
	-256	914	3338	2184	3102	8705

Tabla No.3 Desempeño de los Candidatos al AES – Segunda Ronda Eliminatoria [31]

En esta tabla presentada en el AES2, se muestran datos comparativos de 10 de los 15 finalistas, con eficiencia en lenguaje C++ con procesadores Pentium Pro y Pentium II. Se

exhiben datos, tanto de la función expansión E_K de la clave K (dependiendo de su tamaño), como de los algoritmos cifrado y descifrado. Los datos están en ciclos de reloj para Pentium Pro/II. Los dos valores para el Rijndael son para el cifrado y descifrado. [31]
En un análisis de desempeño realizado por un grupo de especialistas realizado por los Doctores Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall y Niels Ferguson titulado "Performance Comparison of the AES Submissions", se llegó a las siguientes conclusiones:

1. La velocidad de cifrado y de establecimiento de clave son independientes de la longitud de clave.

2. El tiempo requerido para establecer una clave y cifrar un bloque es el mismo, sin importar si la clave es de 128, 192 o 256 bits.

3. Algunos de los candidatos al AES tienen diferentes características de performance, según la longitud de clave usada, esto se resume en la siguiente tabla:

Nombre del Algoritmo	Establecimiento de Clave	Cifrado
CAST-256	Constante	Constante
Crypton	Constante	Constante
DEAL	Dependiente	128, 192: 6 rondas 256: 8 rondas
DFC	Constante	Constante
E2	Constante	Constante
Frog	Constante	Constante
HPC	Constante	Constante
Loki97	Decreciente	Constante
Magenta	Dependiente	128, 192: 6 rondas 256: 8 rondas
MARS	Constante	Constante
RC6	Constante	Constante
Rijndael	Dependiente	128: 10 rondas 192: 12 rondas 256: 14 rondas
SAFER	Dependiente	128: 8 rondas 192: 12 rondas 256: 16 rondas
Serpent	Constante	Constante
Twofish	Dependiente	Constante

Tabla No.4 Comparación del Establecimiento de Clave y de Cifrado de los Candidatos al AES [32]

Algunos datos interesantes de este análisis que se concentraron solo en claves de 128 bits para comparar los 15 algoritmos candidatos al AES, muestran que los algoritmos DEAL y Magenta cifran un 33% más lento usando claves de 256 bits, Rijndael cifra 20% más lento con claves de 192 bits y 40% más lento con claves de 256 bits y el más demorado, el SAFER+ cuya degradación mostró que cifraba 50% más lento con claves de 192 bits y 100% más lento con claves de 256 bits. [32]

En abril del 2000 se celebró la Tercera Conferencia de Candidatos AES (tercera ronda) en Nueva York, durante la cual los asistentes presentaron nuevos documentos de evaluación y criptoanálisis de los últimos cinco candidatos (MARS, RC6, RIJNDAEL, SERPENT, TWOFISH). Varios de los algoritmos recibieron fuertes ataques criptográficos; RC6 fue quién resultó más afectado: dos grupos se las ingeniaron para romper 15 de 20 ciclos más rápidamente que con fuerza bruta. Rijndael resistió algo mejor: 7 ciclos rotos de 10/12/14 ciclos. Se presentaron varios ataques contra MARS; el más interesante rompió 11 de 16 ciclos del núcleo criptográfico. Serpent y Twofish se comportaron mejor: el ataque más fuerte contra Serpent rompió 9 de 32 ciclos, y no se presentaron nuevos ataques contra Twofish. [33] [34]

Según los ataques publicados, los resultados obtenidos en cada uno de los cinco algoritmos finalistas fueron:

MARS: parece tener un margen de seguridad elevado, aunque recibió críticas basadas en su complejidad, la cual puede haber obstaculizado su análisis de seguridad durante el proceso de desarrollo del AES.

RC6: parece tener un margen de seguridad adecuado, sin embargo, RC6 recibió alguna crítica debido a su bajo margen de seguridad respecto al que ofrecieron otros finalistas. Por otro lado, RC6 ha sido elogiado por su simplicidad, la cual facilitó su análisis de seguridad durante el tiempo especificado en el proceso de desarrollo del AES.

RIJNDAEL: parece tener un margen de seguridad adecuado, su margen de seguridad es un poco difícil de medir, debido a que el número de rondas cambia con el tamaño de la clave. Rijndael recibió críticas sobre su margen de seguridad, ya que es de los más bajos, entre los finalistas, y que su estructura matemática puede conducir a ataques. Sin embargo, su estructura es bastante simple, esto ha facilitado su análisis de seguridad durante el tiempo especificado en el proceso de desarrollo del AES.

SERPENT: parece tener un margen de seguridad alto y también tiene una estructura simple, que ha facilitado su análisis de seguridad durante el tiempo especificado de desarrollo del AES.

TWOFISH: parece tener un margen de seguridad alto. El concepto de margen de seguridad tiene menos significado para este algoritmo que para los demás finalistas. La dependencia de las S-cajas de Twofish en solo K/2 bits de entropía en el caso de clave K-bits ha producido algunas especulaciones acerca de que este algoritmo puede ser sensible a un ataque divide y vencerás, aunque tal ataque no ha sido comprobado. Twofish ha recibido alguna crítica por su complejidad, haciendo difícil su análisis durante el tiempo establecido en el proceso de desarrollo del AES. [33] [34]

4.2.2. ANÁLISIS DEL ALGORITMO AES

Es el algoritmo ganador del concurso hecho por el Instituto Nacional de Estándares y Tecnología de los Estados Unidos (NIST) que buscaba un nuevo sistema de cifrado confiable y cuyos criterios de escogencia se basaban en básicamente tres categorías: seguridad, costo y características del algoritmo e implementación. El NIST invitó a toda la comunidad criptográfica a realizar ataques y tratar de criptoanalizar los diferentes candidatos, así como evaluar los costos de implementación, hasta obtener cinco

candidatos. El ganador fue anunciado en Octubre del 2000, siendo seleccionado como el AES el algoritmo “Rijndael”, (el nombre es una combinación de los nombres de sus dos autores, Vincent “Rijmen” y Joan “Daemen”). [30]
Este algoritmo ha sido implementado en todo tipo de plataformas, tanto de Hardware como de Software. Algunas de estas implementaciones, han manipulado las especificaciones del AES, con el fin de obtener un incremento en la eficiencia.

La aprobación por parte del gobierno norteamericano al algoritmo Rijndael como un estándar, le da una “certificación de calidad”. El AES ha sido enviado a la Organización Internacional para la Estandarización (ISO) y al grupo de trabajo para la ingeniería de Internet (IETF), así como a la IEEE para que lo adopten como estándar. [30]

Entre las características principales que favorecieron la rápida adopción de Rijndael están el hecho de ser un código abierto libre de derechos de autor y que gracias a su calidad, puede ser implementado de manera sencilla en una amplia variedad de plataformas sin que el rendimiento y la eficiencia se vean reducidos de manera considerable; además de que los datos de entrada y salida del AES son considerados como arreglos de bytes unidimensionales, además es fácilmente extensible a múltiplos de 32 bits tanto como para la clave como para la longitud de bloque.[30]

4.2.2.1 Eficiencia en el Uso de Recursos Computacionales

4.2.2.1.1 La Eficiencia en Software del AES

La operación mas costosa en tiempo es el obtener inversos multiplicativos del campo finito GF(2^8), en este caso esta operación es precalculada y la operación es substituida por un consulta de S-Box, con esta misma técnica varios cálculos del algoritmo son precalculados y entonces se evitan los cálculos reemplazándolos por consultas de tablas.

Respecto a la velocidad de los algoritmos la manera más simple de medirla es en MegaBits por segundo. Los tres procesos más comunes en un algoritmo son el cifrado, el descifrado y el programa de claves, estos tres se miden de manera independiente.

A continuación se presentarán algunos resultados representativos que se han obtenido a lo largo del estudio de Rijndael. Nos sirven de referencia para poder conocer las velocidades estándares con que se cuentan en nuestros días. [34]

Plataforma PIV 3.2GHz, assembly
Autor H. Lipmaa, Lipmaa Web Page
Longitud de clave 128
Velocidad de Cifrado 1537.9.7 Mb/s
Velocidad de Descifrado 1519.9 Mb/s

Plataforma PPro 200 MHz, C
Autor B.Gladman, Gladman Web Page
Longitud de clave 128
Velocidad de Cifrado 70.7 Mb/s
Velocidad de Descifrado 71.5 Mb/s

Plataforma PIV 1.8GHz, C++

Autor C. Devine, Devine Web Page
Longitud de clave 128
Velocidad de Cifrado 646 Mb/s

Plataforma PII 450MHz, MMX Assembly
Autor K. Aoki, H. Lipmaa [17]
Longitud de clave 128
Velocidad de Cifrado 243 Mb/s

4.2.2.1.2 La Eficiencia en Hardware del AES

Los diseños de hardware especialmente para la implementación de Rijndael requiere de un estudio más especializado que está fuera del alcance del presente proyecto, sin embargo se presentan algunos resultados representativos de esta otra parte de la eficiencia con AES. [34]

Tecnología ASIC
Autor H. Kuo, I. Verbauwhede, P. Schaumont [60]
Velocidad 2.29 Gb/s, cifrado, 128b.

Tecnología VLSI
Autor H. Kuo, I. Verbauwhede [37]
Velocidad 1.82 Gbits/s

Tecnología FPGA[24]
Autor A.J. Elbirt
Velocidad 2 Gb/s, cifrado, 128b.

Tecnología Smart Cards Z80
Autor F. Sano [7]
Velocidad 25 494 Clock (2 veces más rápido que DES)

Tecnología TMS320C62201 DSP
Autor T.Wollingr, M. Wang, J. Guajardo, C. Paar [14]
Velocidad 112 Mbits/s (200 Mhz)

4.2.2.2 Complejidad

Para este análisis y por razones de espacio y tiempo, se tomará el caso de claves k de 128 bits y bloques de 128 bits, teniendo en cuenta que BC(Byte Columns)=4 ya que se tienen 4 columnas de texto y 4 columnas de clave, si fueran de 192 y 256 tendrían 2 y 4 columnas más y el número de vueltas o rondas (ROUNDS) serían de 10 para 128 bits, 12 para 192 bits y 14 para 256 bits. [30]

Se empezará con algo de matemática preliminar, comprendiendo este algoritmo desde su representación polinomial: Se ha definido una representación en forma de polinomio con grado N-1 para los conjuntos de N unos y ceros, donde son estos los coeficientes de dicho polinomio. Para el caso de AES, dichos conjuntos son combinaciones de ocho elementos, dando origen a polinomios de grado siete.

Ej. El polinomio X^6+X^3+X+1 tiene como representación binaria: 01001011. y viceversa.

La matemática de AES es conocida como modular o de reloj, lo cual significa que el conjunto de elementos es finito. Cuando se llega al último elemento del conjunto, de una forma cíclica se pasa al elemento primero de éste. Debe existir un elemento neutro (N) tal que N por cualquier elemento (X) sea igual a X; además para cada X existe un Y tal que X por Y sea igual a N, por tanto $Y=X^{-1}$. Teniendo en cuenta lo anterior, se puede definir la operación suma como una operación XOR entre bytes, ya que dado que esta operación no tiene carry, no se tendrán problemas de sobre flujo, así los datos estarán siempre dentro del rango del conjunto. Se define también la multiplicación (Producto Bytes), pero en esta operación se pueden dar problemas de sobre flujo, ya que la multiplicación de dos polinomios de grado ocho puede dar un polinomio de grado 16; La forma de convertir este resultado a uno que pertenezca al rango del conjunto, es dividiendo por un valor específico, un polinomio irreducible, y tomar el residuo como el equivalente del polinomio de grado 16 en el conjunto de polinomios de grado ocho. [30]

Para el caso de AES, el módulo es el polinomio $X^8+ X^4+ X^3+ X+1$ (100011011), el cual, según las especificaciones de AES, es un polinomio que hace que el algoritmo de multiplicación sea eficiente, además de otras características que éste debe tener.

El algoritmo A.E.S. cifra bloques de 128, 192, o 256 bits usando claves de 128, 192 ó 256 bits. El proceso consiste en una serie de cuatro transformaciones matemáticas, las cuales se repiten 10, 12 o 14 veces, dependiendo de la longitud del bloque y de la longitud de la clave. Todos los ciclos, excepto el último son similares y consisten de las siguientes transformaciones:

- Transformación ByteSub (Sustitución de bytes).
- Transformación ShiftRow (Desplazamiento de filas).
- Transformación MixColumns (Multiplicación de columnas).
- Transformación AddRoundKey (Se aplica una or-exclusiva entre los bits del texto ya modificado y la llave).

En el último ciclo sólo se ejecutan las siguientes transformaciones:

- Transformación ByteSub.
- Transformación ShiftRow.
- Transformación AddRoundKey.

El resultado intermedio del cifrador es llamado estado y las diferentes transformaciones operan sobre este. El estado puede ser graficado como un arreglo rectangular de bytes. Este arreglo tendrá 4 filas, el número de columnas (Nb) es igual al tamaño del bloque dividido entre 32. [30]

Cada una de estas funciones tiene un propósito preciso:

- Una capa de mezcla lineal - funciones DesplazarFila y MezclarColumnas –que permite obtener un alto nivel de difusión a lo largo de las diferentes rondas.
- Una capa no lineal – función ByteStub – consiste en la aplicación paralela de cajas-S con propiedades óptimas de no linealidad.

- Una capa de adición de clave es una operación or-exclusivo entre la entrada ya manipulada y la subclave correspondiente a cada ronda.

En forma más detallada:

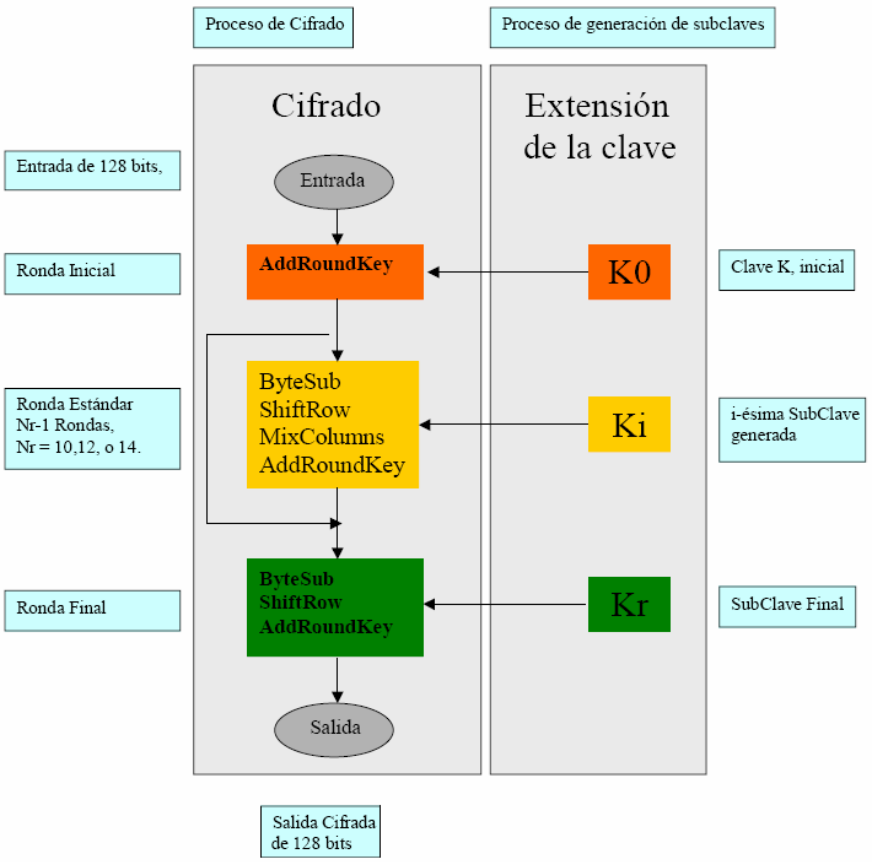

Figura No. 4 Diagrama de Flujo Algoritmo AES [30]

La llave del cifrador puede ser similarmente dibujada como un arreglo rectangular de 4 filas. El número de columnas de la llave del cifrador (Nk) es igual al tamaño de la llave divido entre 32. [30]

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$

Figura No. 5 Ejemplo de Estado con Nb = 6 y Nk = 4 [30]

El número de ciclos o rondas (Nr) depende de los valores de Nb y Nk, según la tabla:

Nr	Nb = 4	Nb = 6	Nb = 8
Nk = 4	10	12	14
Nk = 6	12	12	14
Nk = 8	14	14	14

Tabla No. 5 Valores de ciclos Nr en función de Nk y Nb [30]

Este número de ciclos fue seleccionado por los diseñadores del algoritmo y para ello tuvieron en cuenta el numero máximo de ciclos en los que un ataque de atajo ("shortcut attacks" que consiste en atacar las debilidades de ciertos procesos del algoritmo en los cuales se pueden filtrar datos o texto plano.) encontraba solución, y adicionando un margen considerable de seguridad.

Las sucesivas transformaciones se pueden imaginar como aplicadas a una matriz A de cuatro filas por cuatro columnas, representada en la Ec. 1.

$$A=\begin{matrix} A_{00} & A_{10} & A_{20} & A_{30} \\ A_{01} & A_{11} & A_{21} & A_{31} \\ A_{02} & A_{12} & A_{22} & A_{32} \\ A_{03} & A_{13} & A_{23} & A_{33} \end{matrix} \qquad \text{Ec. (1)}$$

Los sucesivos A_{ij} son los 16 caracteres del texto del bloque de 128 bits a cifrar.

Todas las transformaciones son aplicadas a la matriz A, excepto la transformación AddRoundKey.

Seguidamente está el proceso de Expansión de la clave, que consiste en la generación de diez claves distintas, ya que el proceso de cifrado mencionado previamente se repetirá diez veces o, como se describe en el algoritmo, a lo largo de diez rondas. De esta manera se genera una clave diferente para cada ronda.

A continuación se mostrará una tabla extraída de investigaciones sobre la fortaleza de las rondas del AES:

CIFRADOR	Tamaño de la Clave	Complejidad	
		Memoria	Tiempo (seg)
Rijndael - 6	(Todos)	2^{32} CP	2^{72}
Rijndael – 6	(Todos)	$6 * 2^{32}$ CP	2^{44}
Rijndael – 7	(192)	$19 * 2^{32}$ CP	2^{155}
Rijndael – 7	(256)	$21 * 2^{32}$ CP	2^{172}
Rijndael – 7	(Todos)	$2^{128} - 2^{119}$ CP	2^{120}
Rijndael – 8	(192)	$2^{128} - 2^{119}$ CP	2^{188}
Rijndael – 8	(256)	$2^{128} - 2^{119}$ CP	2^{204}
Rijndael – 9	(256)	2^{85} RK - CP	2^{224}

Tabla No. 6 Complejidad en el criptoanálisis de Rijndael – CP= Texto Plano Escogido, RK=clave Relacionada, Tiempo (Segundos) [35]

Esta tabla nos muestra la poca factibilidad actual de lograr un ataque exitoso contra el algoritmo de Rijndael, por ejemplo para el algoritmo con 6 ciclos el ataque tiene una demora de 2^{72} segundos esto es 149.745.258.842.898 años. [35]

Para entender la fortaleza de este algoritmo es necesario examinar cada una de las transformaciones aplicando la regla de "divide y vencerás" para determinar la complejidad total.

4.2.2.2.1. La Suma de la Llave: AddRoundKey (ARK)

Esta ronda consiste en la combinación de la llave de ronda con la matriz de estado mediante la operación lógica XOR. La llave de ronda tiene la misma longitud que la matriz de estado (128 bits). El AddRoundKey es ilustrado en la figura En la figura ai,j denota el byte correspondiente a la matriz de estado, mientas que ki,j al de la llave de ronda, así mismo bi,j simboliza al byte de la matriz de estado resultante.

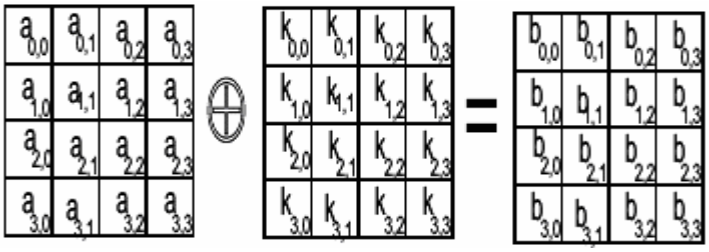

Figura No. 6 Adición de Llave [30]

Este procedimiento es el más sencillo, y recibe la matriz a[4][4] y rk[4][4], BC=4, entonces actualiza los valores de a[4][4], haciendo una operación xor, byte por byte, a[i][j] xor rk[i][j].

Para esta función se tienen 107 operaciones para el caso de una clave de 128 bits por lo que las constantes de BC y ROUNDS son 4 y 10 respectivamente si seguimos la secuencia, es decir para claves de 192 bits y 256 bits (BC= 6 y 8; ROUNDS= 10 y 12) daría: 107, 139 y 171 procesos respectivamente, lo que indica que a medida que aumenta la clave en 64 bits, se aumentan en 32 el número de procesos. [29]

4.2.2.2.2. Sustitución de Bytes: SubBytes (BS)

La transformación SubBytes, es una sustitución no lineal, que opera independientemente en cada byte de la matriz. Se sustituye la matriz [a_{ij}] por ij [$\mathbf{s}_{ij}$], donde $\mathbf{s}_{ij}$ es el resultado de aplicar dos funciones a a_{ij}, 1) Calcular su inverso multiplicativo $a_{ij} \rightarrow a_{ij}^{-1} \in GF(2^8)$ y 2) Aplicar posteriormente una transformación lineal.

1) Todo byte, conjunto de 8 bits, puede verse como un elemento del campo finito $GF(2^8)$, como todo elementotiene inverso multiplicativo, entonces esta primera función de bytesub, asocia el inverso multiplicativo en $GF(2^8)$, es decir $a_{ij} \rightarrow a_{ij}^{-1} \in GF(2^8)$; al elemento cero se le asocia el mismo cero.

2) Ahora la transformación lineal que sigue al inverso multiplicativo se aplica bit por bit y sigue la siguiente regla: Transformacion lineal en $GF(2^8) \rightarrow GF(2^8)$

$b_i' = b_i \oplus b_{(i+4)mod8} \oplus b_{(i+5)mod8} \oplus b_{(i+6)mod8} \oplus b_{(i+7)mod8} \oplus c_i$; 0< i<8, c = 01100011 = 63_x

El valor de S correspondiente a a[i][j] = xy en hexadecimal, es el valor que está en la fila x y la columna y. [29]

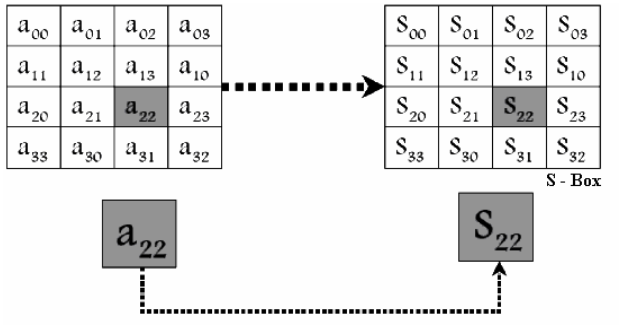

Figura No. 7 Función Sustitución de Bytes [30]

Puntos a Considerar:

- La No linealidad.
- La Correlación entre la entrada y la salida debe tener la menor amplitud posible.
- La Probabilidad de la propagación de diferencias, la cual debe ser lo más pequeña posible.
- La Complejidad Algebraica: La expresión algebraica de S-Box en GF(2^8) tiene que ser una expresión lo suficientemente fuerte para resistir ataques de interpolación

Este proceso cuenta con 91 operaciones.

Este procedimiento también es simple, y sólo reemplaza cada byte a[i][j] por el correspondiente box(a[i][j]), es decir: box(00)=S[0], box(01)=S[1], box(11)=S[17].

4.2.2.2.3. Corrimiento de Renglones: ShiftRows (SR)

Se aplica a la matriz [a_{ij}], shifts(corrimientos izquierdos circulares de bytes) a las renglones, de la siguiente manera, recorre 0 bytes a el primer renglón, 1 byte al segundo renglón, 2 bytes al tercer renglón, y 3 bytes recorridos al cuarto renglón.

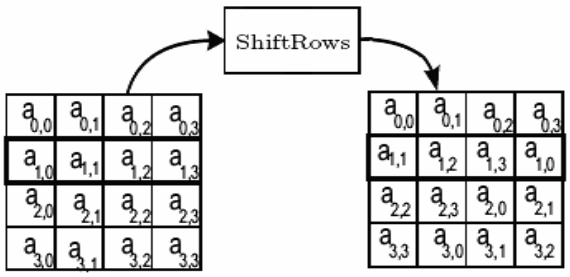

Figura No. 8 Corrimiento de Renglones [30]

El proceso cuenta con 174 operaciones y para claves de 192bits y 256 bits se tienen 246 y 318 operaciones, es decir que a medida que aumenta la clave en 64 bits, se aumentan en 72 el número de procesos. [29]

4.2.2.2.4. El mezclado de Columnas: MixColumns (MC)

A cada columna de la matriz $[a_{ij}]$, la multiplica por una columna constante en $GF(2^8)[x]/(x^4+1)$.

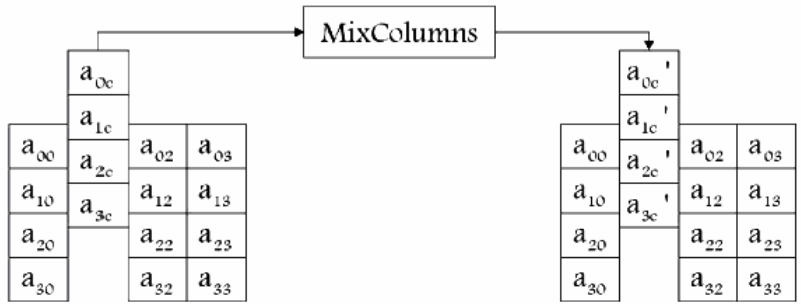

Figura No. 9 Mezclado de Columnas [30]

Aquí se toma cada columna A, y la manda a otra columna A', que se obtiene al multiplicar A por un polinomio constante $c(x) \in GF(2^8)[x]/(x^4 + 1)$, $c(x) = 03\ x^3 + 01\ x^2 + 01\ x + 02$, entonces $A' = A \cdot c(x)$, como se vio en los antecedentes, este producto se puede representar por la siguiente matriz.

$$\begin{bmatrix} a_0' \\ a_1' \\ a_2' \\ a_3' \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Figura No. 10 Matriz Característica [30]

La anterior función hace uso de dos tablas para logaritmos y antilogaritmos para multiplicar dos elementos del campo finito $GF(2^8)$.

Para una clave de 128 bits se tienen 566 operaciones, para claves de192 bits se tienen 834 operaciones y para claves de 256 bits se tienen 1102 operaciones, lo que indica que a medida que aumenta la clave en 64 bits, se aumentan en 268 el número de procesos.

En esta función se multiplica cada columna de entrada por una columna constante. [29]

4.2.2.2.5. Proceso de Cifrado

Esta función cuenta con 9833 operaciones para la clave de 128 bits, para claves de 192 bits se tienen 17953 operaciones y para claves de 256 bits se tienen 26073 lo que indica que a medida que aumenta la clave en 64 bits, se aumentan en 8120 el número de procesos.

En a[4][4] esta el texto a cifrar como matriz de 16 bytes, en rk[10+1][4][4] se guarda la expansión de la clave K. En este caso, BC=4 para el AES de 128 bits y tiene 4 columnas de texto y 4 columnas de clave, por lo tanto ROUNDS=10. [29]

4.2.2.2.6. Función Keysched

Esta función recibe una clave k, de 4 columnas y regresa un arreglo W de 44 columnas y consta de 523 operaciones.

En el primer bucle for anidado se copia el valor de la clave en la variable tk, en el segundo bucle for anidado se copia el valor de la primera subclave en el arreglo W, en el bucle mientras se generan las siguientes subclaves y en el último bucle Para anidado se copia la r-ésima subclave en el arreglo W. [29]

4.2.2.2.7. Función MixColumns para Descifrado

Realiza las mismas operaciones que la función MixColumns, lo que varía son las constantes enviadas a la función mul. Cuenta con 566 operaciones.

4.2.2.2.8. Proceso de Descifrado

La transformación AddRoundKey, igual que en el proceso de cifrado, consiste de una EXOR de 128 bits. Las demás transformaciones son las inversas de las transformaciones explicadas anteriormente en el proceso de cifrado.

El diagrama de flujo para este proceso se expone a continuación:

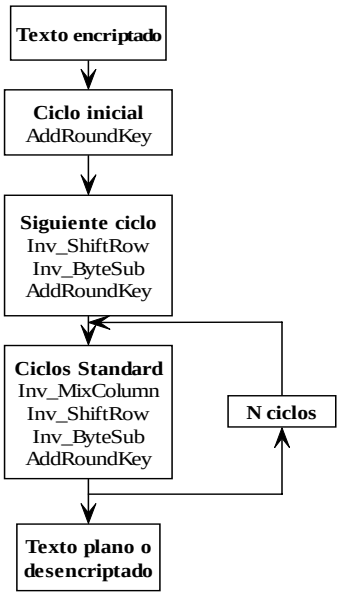

Figura No. 11 Proceso de Descifrado

El proceso de descifrado se diferencia del de cifrado por el procedimiento de invertir columnas y la caja s invertida (Si). Y cuenta con 8951 operaciones.

En síntesis la complejidad total de este algoritmo a pesar de encontrarse en el campo de la matemática discreta, se clasifica en la clase de problemas NP- completos, ya que no se han encontrado funciones polinómicas capaces de romper su integridad, además su mayor fortaleza radica en la matemática empleada, ya que aprovecha las propiedades del campo finito F_2^8 y el polinomio empleado para el módulo tiene el aval de la comunidad de criptoanálistas, por último sobra decir que las pruebas Criptoanalíticas realizadas para solo 6 rondas confirman su robustez , a lo que hay que añadir la complejidad agregada del Acuerdo de la clave $O(n^2)$ y la complejidad del algoritmo de generación de claves pseudoaleatorias, que en el pero caso es $O(\log_2 r)$.

4.2.2.3 Patente

El documento FIPS especifica que AES se puede implementar para la seguridad de datos o la seguridad de cualquier sistema computacional que requiera protección criptográfica, siendo estos datos no clasificados de acuerdo con acta de seguridad nacional de 1947 y el acta de energía atómica de 1954, en ambos casos se hace también referencia a las correcciones de las mismas. Además este estándar puede ser adoptado y usado por una organización no federal, cuando de seguridad comercial y privacidad de la misma se trata. Esto hace posible usar este algoritmo libremente en este trabajo de grado.

Con respecto a las implementaciones, el documento dice que el algoritmo puede ser usado para software, firmware, hardware o cualquier combinación de estas, donde la implementación puede depender de tantos factores como la aplicación, el medio ambiente, la tecnología usada, etc. [27] [29]

4.2.3 ANÁLISIS DEL ALGORITMO IDEA

4.2.3.1 Descripción y Funcionamiento

El algoritmo IDEA (International Data Encryption Algorithm) fue diseñado por Xuejia Lai y James L. Massey en 1992 y se considera uno de los más seguros algoritmos simétricos disponibles en la actualidad (lo recomiendan Bruce Schneier y Microsoft, entre otros). Trabaja con bloques de 64 bits de longitud y emplea una clave de 128 bits. Como en el caso de DES, se usa el mismo algoritmo tanto para cifrar como para descifrar. [29]

El IDEA es un algoritmo bastante seguro, y hasta ahora se ha mostrado resistente a multitud de ataques, entre ellos el criptoanálisis diferencial. No presenta claves débiles, y su longitud de clave hace imposible en la práctica un ataque por la fuerza bruta. [29]

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

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

El algoritmo IDEA consta de ocho rondas. Se divide el bloque X a codificar, de 64 bits, en cuatro partes X_1, X_2, X_3 y X_4 de 16 bits. Se llama Z_i a cada una de las 52 subclaves de 16 bits que vamos a necesitar. Las operaciones que llevaremos a cabo en cada ronda son las siguientes:
1. Multiplicar X_1 por Z_1.
2. Sumar X_2 con Z_2.
3. Sumar X_3 con Z_3.
4. Multiplicar X_4 por Z_4.
5. Hacer un XOR entre los resultados del paso 1 y el paso 3.
6. Hacer un XOR entre los resultados del paso 2 y el paso 4.
7. Multiplicar el resultado del paso 5 por Z_5.
8. Sumar los resultados de los pasos 6 y 7.
9. Multiplicar el resultado del paso 8 por Z_6.
10. Sumar los resultados de los pasos 7 y 9.
11. Hacer un XOR entre los resultados de los pasos 1 y 9.
12. Hacer un XOR entre los resultados de los pasos 3 y 9.
13. Hacer un XOR entre los resultados de los pasos 2 y 10.
14. Hacer un XOR entre los resultados de los pasos 4 y 10.

La salida de cada iteración serán los cuatro sub-bloques obtenidos en los pasos 11, 12, 13 y 14, que serán la entrada del siguiente ciclo, en el que se emplean las siguientes seis subclaves, hasta un total de 48. Al final se intercambian los dos bloques centrales. [29]

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

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

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

Las subclaves necesarias para descifrar se obtienen cambiando de orden las Z_i y calculando sus inversas para la suma o la multiplicación. Puesto que $2^{16} + 1$ es un número primo, nunca podremos obtener cero como producto de dos números, por lo que no se necesita representar dicho valor. Cuando se calculen los productos, se utiliza el cero para expresar el número 2^{16}. Esta representación es necesaria puesto que los registros que se emplean internamente en el algoritmo poseen únicamente 16 bits. [29]

4.2.3.2 Complejidad y Seguridad

En primer lugar, el ataque por fuerza bruta resulta impracticable, ya que sería necesario probar 1038 claves, cantidad imposible de manejar con los medios informáticos actuales. Al igual que Blowfish, utiliza operaciones de bajo coste computacional (XOR, suma módulo 2^{16} y producto módulo $2^{16} + 1$). [17]

No tiene claves débiles, aunque hay algunas que pueden dar cierta ventaja a la hora del criptoanálisis, sin embargo encontrarlas es prácticamente imposible (probabilidad de 2^{-96}).

Los diseñadores analizaron IDEA para medir su fortaleza frente al criptoanálisis diferencial y concluyeron que lo es bajo ciertos supuestos. No se han reportado debilidades frente criptoanálisis lineal o algebraico. Se han encontrado algunas claves débiles, las cuales en la práctica son poco usadas siendo necesario evitarlas explícitamente. [17]

Por lo anterior también se debe incluir en el grupo de los algoritmos con problema NP-Completo, además de la complejidad añadida del Acuerdo de la clave $O(n^2)$ y la complejidad del algoritmo de generación de claves pseudoaleatorias, que en el pero caso es $O(\log_2 r)$.

4.2.3.3 Patente

IDEA es un algoritmo que si bien es de uso libre para fines no comerciales, si está cubierto por patentes, concretamente:

- USA y Canadá - Patente 5.214.703 - Expira el 25 de Mayo de 2010.
- Europa (Austria, Francia, Alemania, Italia, Holanda, España, Suecia, Suiza, Reino Unido) - Patente 0482154 - Expira el 16 de Mayo de 2011.
- Japón - Patente JP3225440B2 - No hay información de la fecha de expiración.

Precisamente a causa de esta patente, el sistema gnuPG no lo incluye como algoritmo de cifrado simétrico (a pesar de que PGP sí lo hace), siendo necesario añadirlo mediante un parche al software. [29]

4.2.4 ANÁLISIS DEL ALGORITMO RC6

4.2.4.1 Descripción y Funcionamiento

Desde que se propusieron RC5 en 1995, varios estudios han proporcionado una comprensión mayor de cómo la estructura y funcionamiento de RC5 contribuye a su seguridad. Mientras ningún ataque práctico en RC5 ha sido encontrado, los estudios proporcionan algunos ataques teóricos interesantes, generalmente basado en el hecho que la suma de rotaciones en RC5 no depende de todos los fragmentos de información en un registro.[29]

RC6 ha sido diseñado para frustrar este tipo de ataques, y de hecho para frustrar todos los ataques conocidos, proporcionando un cifrado que obtiene la seguridad requerida.

Para reunir los requisitos del AES, un cifrador de bloque debe manejar bloques de128 bits de Entrada/Salida. Ya que RC5 es un cifrador de bloque excepcionalmente rápido actuando sobre bloques de 64 bits, para actuar en bloques de 128 bits, la manera más natural sería usando dos registros de 64 bit. Sin embargo el diseño de RC6 se realizó utilizando 4 registros de 32 bits, esto se debe a que los lenguajes y arquitecturas, objetivos de AES, no manejan de manera eficiente registros 64 bits. [29]

Esta forma, tiene la ventaja de que se realizaran dos rotaciones por ronda, el lugar de una media rotación que se realiza en el RC5, la filosofía de RC5 es aprovecharse de funciones (como rotaciones) las cuales se llevan a cabo en procesadores modernos en forma eficiente. RC6 continúa en esta tendencia, y con ventaja en la mayoría de los procesadores, la multiplicación entera de 32 bits es muy eficiente. [29]

Como RC5, RC6 es una familia de algoritmos totalmente parametrizados. Una versión de RC6 especificada mas adecuadamente sería RC6-w/r/b donde el tamaño de la palabra es de w bits, el cifrado consiste en un número no negativo de rondas r, y b denota el largo de la clave de cifrado en bytes, según los requerimientos de AES w es de 32 bits y r=20. [29]

RC6 trabaja con cuatro registros de w bits, A, B, C y D, los cuales contienen la entrada inicial es decir o bien el texto plano o el texto cifrado. El primer byte del texto plano o el texto cifrado es colocado en el byte menos significante de A; el ultimo byte del texto plano o cifrado se coloca en el byte más significante de D. [29]

4.2.4.2 Seguridad

Se dice que para atacar al RC6, la mejor aproximación disponible es la búsqueda exhaustiva de los b-byte de la clave de cifrado. El esfuerzo requerido para realizar esto tiene una cantidad de operaciones que esta dada por el mín$\{2^{8b}, 2^{1408}\}$.[28]

Los ataques más avanzados de criptoanálisis diferencial y lineal, que podrían llegar a ser posibles en versiones de pocas rondas, quedan si efecto para versiones de 20 rondas. La principal dificultad es que es tedioso encontrar buenas características iterativas o aproximaciones lineales con las que el ataque pueda ser llevado a cabo.

Los ataque más avanzados típicamente requieren de cantidades grandes de datos, y obteniendo 2^a bloques de pares texto plano- texto cifrado, conocido o escogido, se puede intentar recobrar la clave desde una 2^a posibilidades (esta última tarea puede ser paralelizada). Vale la pena destacar que con un cifrador corriendo a una razón de un terabit por segundo (esto es 10^{12} bits/sec.), el tiempo requerido para 50 computadoras, trabajando en forma paralela para cifrar 2^{64} bloques de datos es algo más de un año, para cifrar 2^{80} bloques de datos es mas de 98000 años, y finalmente para cifrar 2^{128} bloques de datos es mas de 10^{19} años.[28]

Para atacar una versión de 8 rondas, RC6-32/8/b, se puede construir características seis-redondas o las aproximaciones lineales, la estimación de datos requeridas para montar un criptoanálisis diferencial sobre este, debería ser alrededor de 2^{76} elecciones de pares de texto plano – texto cifrado, y para realizar un ataque con criptoanálisis lineal debería ser aproximadamente 2^{60} textos conocidos.[28]

4.2.4.3 Complejidad

Debido a lo anterior, este algoritmo debe ubicarse en el grupo de los NP-completos, debido a su alta calidad de operaciones elementales y de bit.

Los requerimientos de datos para ataques más sofisticados sobre el RC6 son tales que los criptoanálisis lineales o diferenciales exceden la cantidad de datos disponibles y no hay aun ejemplos los cuales en los cuales se pueda encontrar la clave en tiempos prácticos.

4.2.4.4 Eficiencia Software

Los valores para la implementación sobre JAVA de RC6 medida sobre un computador Pentium Pro con 64 Mb de RAM corriendo en Windows NT tomando un promedio de 10 mediciones, son: 16200ciclos/bloque, con 12300 bloques/seg y 0.197 MBytes/seg. [28]

Algoritmo RC6	Código Assembler Pentium	ANSI (Microsoft)	ANSI C (Borland)	JAVA (JDK)
RC6-32/20/16	5.5	12.2	23.5	537
RC6-32/20/24	5.5	12.8	23.5	542
RC6-32/20/32	5.5	13.1	23.6	548

Tabla No. 7 Eficiencia en Hardware del Algoritmo RC6
Establecimiento de la Clave (mseg) [28]

4.2.5 ANÁLISIS DEL ALGORITMO TWOFISH

4.2.5.1 Descripción

El algoritmo Twofish nació como una evolución de otro algoritmo llamado Blowfish, que trata de cumplir con todos los requerimientos del NIST para el nuevo AES añadiendo un valor agregado fue diseñado por Bruce Schneier, John Kelsey, Doug Withing, David Wagner, Chris Hall y Niels Ferguson. Específicamente, las metas de diseño radican en:

- Cifrar un bloque de cifrado simétrico de 128 bits
- Longitud de llaves de 128, 192 y 256 bits
- No existencia de claves débiles
- Eficiencia en múltiples plataformas de hardware y software
- Diseño flexible
- Diseño simple
- Valor agregado: soporte de claves variables de longitudes mayores y menores a 256 bits, alta velocidad de codificación, no contener operaciones que sean ineficientes en otros procesadores de 8, 16, 32 o 64 bits, no incluir componentes que lo hagan ineficiente al implementarlo en hardware, número variable de rondas de cifrado e incluir un horario de claves. [29]

4.2.5.1.2 Bloques de construcción

4.2.5.1.2.1 Redes de Feistel

Una red de Feistel es un método general de transformación de cualquier función (usualmente llamada función F) dentro de una permutación. 4 bits de un bloque de texto plano son tomados de dos en dos y son transformados dentro de una permutación; en consecuencia, dos rondas de la red de Feistel es llamada un ciclo de cambio en donde cada bit de el bloque de texto ha sido transformado por lo menos una vez.

Este mecanismo de transformación de bits de información ha sido utilizado en la mayoría de los nuevos bloques de cifrado existentes, ejemplo de ello se encuentra en los siguientes algoritmos: FEAL, GOST, Khufu and Kafre, LOKI, CAST-128, Blowfish y RC5.

En el algoritmo Twofish se usa una conjunción de una red de Feistel de 16 rondas, o ciclos, con una función F biyectiva. [36]

4.2.5.1.2.2 Cajas – S

Una caja-s es una operación de sustitución no linear manejada por tablas usada también en la mayoría de los bloques de cifrado. Estas varían tanto en su cantidad de entradas como en su cantidad de salidas., y pueden ser creadas randómica o algorítmicamente. Las cajas-s fueron usadas en Lucifer, en DES y posteriormente en la mayoría de algoritmos de cifrado. Además Twofish usa cuatro diferentes cajas-S, biyectivas y dependientes de la clave de 8x8 bits en su implementación. Estas cajas-S son construidas usando dos permutaciones fijas de 8x8 bits y material de clave.[36]

4.2.5.1.2.3 Matrices MDS

Estas matrices fueron propuestas por primera vez en 1995 por Serge Vaudenay, publicada en el algoritmo de cifrado no publicado Manta (1996) y utilizado en Twofish con una simple matriz de 4x4 sobre GF.[36]

4.2.5.1.2.4 Transformadas Pseudo-Hadamard

Estas son simples operaciones de mezcla que corren rápidamente por software. Dadas dos entradas, a y b, la PHT de 32 bits usada en Twofish, es definida como:
a’ = a + b mod 32
b’ = a + 2b mod 32

4.2.5.1.2.5 Blanqueamiento

El uso de operaciones XOR con material de clave antes de la primera ronda y después de la última ronda de Twofish, incrementa sustancialmente la dificultad del criptoanalista de buscar los restos o huellas del cifrado.

Twofish usa este blanqueamiento con subclaves de 128 bits, y estas subclaves no son usadas para el cifrado. [36]

4.2.5.1.2.6 Horario de clave

Esta técnica ayuda a definir que bits de la clave pueden ser usados dentro de las rondas de cifrado. Twofish usa gran cantidad de material de clave y con ello provee un complicado horario de clave.

El núcleo central del Twofish lo conforma la función g, en ella se procesan los bloques de las cajas-s mas las matrices MDS utilizando dos bits de cuatro cada caja-s, la próxima tomarán los otros dos y se completará una ronda. Este complejo diseño matemático que puede ser implementado por software o hardware provee una excelente base para las generaciones futuras de algoritmos criptográficos. [36]

4.2.5.2 Complejidad

Este algoritmo por tener redes de Feistel también se clasifica dentro del grupo de los problemas NP-completos, además de las complejidades añadidas del Acuerdo de la clave y del algoritmo de generación de claves pseudoaleatorias.

4.2.5.3 Eficiencia Hardware

Se dice que, por lo general para tamaños de clave muy grandes, se requiere más memoria RAM para su procesamiento: 36 bytes para claves de 192 bits y 48 bytes para claves de 256 bits; y si se puede de alguna manera grabar información no delicada (texto plano) en la ROM o EEPROM, entonces se podría pensar en utilizar tarjetas con memorias de solo 36 bits de RAM; lo anterior se utliza también en los casos en que se requiera implementar un algoritmo en hardware diferente a un pc. Además se tiene que toda esta RAM se puede reutilizar para otros propósitos entre las operaciones del cifrado del bloque.

Para las tarjetas smart con una memoria más grande para llevar a cabo datos clave-dependientes, la velocidad del cifrado puede aumentar considerablemente. Esto es porque los claves redondos pueden ser precomputed como parte del clave ampliado, requiriendo un total de 184 octetos de memoria dominante. Según las indicaciones del vector 5.4, esta opción parte en dos casi el tiempo del cifrado. Si la tarjeta smart tiene bastante memoria adicional disponible almacenar 1 kB de la caja S precomputada en la RAM, ROM, o EEPROM (para un total de 1208 bytes), su funcionamiento mejora en gran manera. Si la caja S entera precomputada más la tabla MDS se pueden almacenar en la memoria (3256 bytes), la velocidad aumentará. En la práctica, el uso de esta cantidad de RAM guarda 512 bytes de espacio del código si se asume que ciertos valores no se almacenan en la ROM. Si la tarjeta smart tiene que realizar su propia extensión dominante el tamaño del código aumentará. Este aumento tiene sus propias opciones de compensación espacio/tiempo.

Esta flexibilidad hace que el algoritmo Twofish esté bien adaptado para los procesadores pequeños y grandes de tarjetas smart: Twofish trabaja en los ambientes RAM más pobres, mientras que al mismo tiempo puede aprovecharse en tarjetas con mediana y alta capacidad de RAM. [36]

4.2.5.4 Agilidad de la velocidad y del clave del cifrado

El rendimiento del algoritmo en sistemas Pentium o Mac cumple con las especificaciones del concurso para el nuevo AES, como se muestra en la siguiente tabla:

Procesador	Lenguaje	Tamaño Código	Clocks Llave			Clocks Cifrado		
			128	192	256	128	192	256
PPro/II	ASM	9000	8600	11300	14100	258	258	258
PPro/II	ASM	8500	7600	10400	13200	315	315	315
PPro/II	ASM	10700	4900	7600	10500	460	460	460
PPro/II	ASM	13600	2400	5300	8200	720	720	720
PPro/II	ASM	9100	1250	1600	2000	860	1130	1420
Pentium	ASM	9100	12300	14600	17100	290	290	290
Pentium	ASM	8200	11000	13500	16200	315	315	315
Pentium	ASM	10300	5500	7800	9800	430	430	430
Pentium	ASM	12600	3700	5900	7900	740	740	740
Pentium	ASM	8700	1800	2100	2600	1000	1300	1600

Tabla No 8 Eficiencia en Hardware del Algoritmo Twofish [36]

Sobre un microprocesador 6805 con solamente 60 bytes de RAM, Twofish cifra a las velocidades de 26500 a 37100 ciclos de reloj por bloque, dependiendo de la cantidad de ROM disponible para el código. Sobre un chip de 4 MHz, esto se traduce en 6.6 milisegundos a 9.3 milisegundos por cifrado. En estas implementaciones, el tiempo del precomputación del establecimiento de la clave es mínimo: levemente sobre 1750 ciclos de reloj por clave. [36]

Si la ROM es costosa, Twofish se puede poner en ejecución en menos espacio a velocidades más reducidas.

En pruebas realizadas por el equipo del Doctor Bruce Schneier se concluye que las claves más largas son más lentas, pero no muy significativamente; para memorias pequeñas, el tiempo del cifrado de Twofish por bloque aumenta en menos de 2600 ciclos de reloj por bloque para claves de 192 bits, y en cerca de 5200 ciclos de reloj por bloque para claves de 256bits. Similarmente, el establecimiento de la clave aumenta a 2550 ciclos de reloj para claves de 192 bits, y a 3400 ciclos de reloj para claves de bits. [36]

Con suficiente almacenaje adicional de la RAM para procesar el conjunto entero de subclaves, el rendimiento de procesamiento mejora perceptiblemente, aunque el tiempo dominante de la disposición también aumenta. Los ahorros del tiempo por bloque están sobre 11000 ciclos de reloj, cortando el tiempo del cifrado del bloque a cerca de 15000 ciclos de reloj; es decir, casi doblando la velocidad del cifrado. El tiempo de establecimiento de la clave aumenta a lo sumo en el mismo número de ciclos de reloj, lo que se traduce en una disminución de código por algunos cientos de bytes. [36]

La variedad amplia de velocidades posibles ilustra otra vez la flexibilidad de Twofish en ambientes obligados. El algoritmo no tiene una velocidad; tiene muchas velocidades, dependiendo de recursos disponibles.

4.2.5.4.1 Tamaño de código

El código de Twofish es muy compacto: 1760 a 2200 bytes usados de RAM, dependiendo de la puesta en práctica. La misma base del código se puede utilizar para el cifrado y el descifrado. Si solamente se requiere el cifrado, las mejoras de menor importancia de tamaño de código pueden ser obtenidas (en la orden de 150 bytes). El código adicional requerido para claves más grandes es bastante insignificante: menos de 100 bytes adicionales para una clave de 192 bits, y menos de 200 bytes para un clave de 256 bits. Además es posible separar espacio adicional de la ROM para las operaciones de búsqueda que computan q0 y q1 usando la construcción subyacente de 4 bits. [36]

Algoritmo	Longitud de la Clave	Bloque (bits)	Rondas	Ciclos	Clocks/Byte
Twofish	variable	128	16	8	18.1
Blowfish	variable	64	16	8	19.8
Square	128	128	8	8	20.3
RC5-32/16	variable	64	32	16	24.8
CAST-128	128	64	16	8	29.5
DES	56	64	16	8	43
Serpent	128, 192, 256	128	32	32	45
SAFER (S)K-128	128	64	8	8	52
FEAL-32	64, 128	64	32	16	65
IDEA	128	64	8	8	74
Triple-DES	112	64	48	24	116

Tabla No 9 Comparación de rendimiento de Algunos Algoritmos Simétricos [36]

4.2.6 ANÁLISIS DELALGORITMO MARS

4.2.6.1 Descripción y Funcionamiento

El MARS es un cifrador de bloques de secreto compartido, con un tamaño de bloque de 128 bits y un tamaño de clave variable que va desde los 128 bits a los 1248 bits. Fue diseñado para alcanzar y superar los requisitos de un estándar de cifrado de secreto compartido en las próximas décadas. El objetivo principal en el diseño del MARS es alcanzar la mejor relación seguridad / performance utilizando las herramientas más fuertes y las tecnologías disponibles de hoy en día para el diseño de cifradores. [38]

El diseño del MARS toma amplias ventajas de la capacidad de las computadoras de hoy de ejecutar rápidas multiplicaciones y rotaciones de acuerdo al dato.

La estructura del cifrador: Dos décadas de experiencia en criptoanálisis han enseñado que diferentes partes de un cifrador cumplen muy diferentes roles en alcanzar la seguridad de un cifrador. El MARS se diseñó usando una estructura mixta, donde las vueltas del principio y del final son distintas de las del medio. [38]

Análisis: Un aspecto importante del MARS es que sus componentes se diseñan para permitir análisis exhaustivos. En cada paso del diseño, se recalca el uso de operaciones y estructuras que parecen “muy difíciles” de analizar. En su lugar, se insiste en proveer un análisis de cada aspecto del cifrador, y se usa este análisis como guía a través de muchas opciones de diseño. [38]

Especificaciones de Diseño

Se trabaja con palabras de 32 bits: Dado que la mayoría de las computadoras de hoy, usan palabras de 32 bits, todas las operaciones en el MARS se realizan con este tamaño de palabra.

Con el estado actual de la tecnología, esta opción ofrece una buena relación entre la capacidad para correr en computadoras actuales y la capacidad de tomar ventaja de tamaño de palabras más grandes en arquitecturas futuras. [38]

Red Feistel tipo 3: Dado que el MARS tiene una longitud de bloque de 128 bits y un tamaño de palabra de 32 bits, se tiene que cada bloque consiste de cuatro palabras. Entre las varias estructuras de redes que son capaces de manejar cuatro palabras en un bloque, la red Feistel tipo 3 brinda una mejor relación entre velocidad, fortaleza y conveniencia para el análisis. Una red Feistel tipo 3 consiste de muchas vueltas, donde en cada vuelta una palabra dato (y unas pocas palabras clave) se usan para modificar todas las otras palabras datos. Comparado con la red Feistel tipo 1 (donde en cada vuelta una palabra de datos se usa para modificar una sola palabra dato), esta construcción provee mejores propiedades de difusión con sólo un pequeño costo adicional. [38]

Por lo tanto se pueden usar menos vueltas para obtener la misma fortaleza. Además, una red Feistel tipo 3 tiene ventajas sobre estructuras en las cuales palabras datos se usan "una vez" para modificar otras palabras datos, en las que las estructuras son más difíciles de analizar. [38]

Simetría de cifrado y descifrado: Se diseñó el MARS para ser tan seguro contra ataques de texto cifrado elegido como contra ataques de texto plano elegido. Esto dictamina a hacer el cifrador muy simétrico de forma tal que la última mitad de las vueltas sea casi una "imagen espejo" de la primera mitad. [38]

4.2.6.2 Complejidad

Sumas, restas y xors: Estas son las operaciones más simples, las cuales se usan para "mezclar" datos (y claves), Estas operaciones son muy rápidas ya sea en software o hardware y generalmente no ofrecen "fuerza criptográfica". A lo largo del cifrador se intercalan xors con sumas y restas para asegurar que las operaciones en el cifrador no se conmuten unas con otras; su complejidad pertenece al grupo de los NP_completos, no solo por las redes de Feistel, sino también por su capacidad de tener rotaciones variables y las multiplicaciones no lineales lo que amplía el rango de posibilidades. [38]

Búsqueda en tablas: Las operaciones de búsquedas en las tablas provee la base para la seguridad del DES, como también de muchos cifradores. El MARS usa una tabla simple de 512 palabras de 32 bits (caja-S). A veces la caja-S se ve como dos tablas, cada una de 256 entradas.

En principio, una caja-S elegida cuidadosamente puede proveer una buena resistencia contra ataques diferenciales y lineales, lo malo es que es relativamente lento para implementaciones en software.

En un cifrador orientado a la palabra como el MARS, una operación de búsqueda en una caja-S toma tres instrucciones (una para copiar la palabra fuente en un registro índice,

una para obtener los bits de orden superior del índice a través de una máscara y la otra para acceder a la tabla misma). También una caja-S puede ocupar una gran cantidad de espacio en implementaciones de hardware. [38]

Otro problema es que el índice de la tabla consiste de solo unos pocos bits (de otra forma la tabla sería muy grande). De ahí que solo se usan unos pocos bits de la palabra de datos.

Rotaciones fijas: Las rotaciones por cantidades fijas se usan para obtener algunos bits de datos para otras aplicaciones (Por ejemplo para usar los bits de mayor orden de una palabra de datos como índice de la caja-S). [38]

Rotaciones de acuerdo al dato: Este tipo de rotaciones fueron usadas para cifrar en un cifrador desarrollado por IBM en los finales de los ’70. Esta operación ganó reconocimiento en años recientes después de ser usada por Rivest como el principal constructor de bloques para el cifrador RC5. Las rotaciones de acuerdo a los datos se pueden llevar a cabo rápidamente en software y hardware. [38]

Un problema con las rotaciones de acuerdo al dato es que para especificar una cantidad de rotación para una palabra de w bits sólo toma log w bits.

De ahí que mientras el resultado de esta operación de todos los bits en el operando, sólo depende de unos pocos bits en el otro. Esto lleva a debilidades diferenciales.

En el MARS, se hace uso extenso de las rotaciones de acuerdo al dato, pero resuelve el problema mencionado, combinando estas operaciones con multiplicaciones.

Multiplicaciones: Las operaciones fueron usadas para cifrar en el cifrador IDEA y sus variantes. Sin embargo, hasta hace poco las multiplicaciones fueron consideradas caras para un cifrado rápido. Esto era cierto dado que, a las máquinas antiguas les tomaba muchos ciclos para ejecutar una simple operación de multiplicación. [38]

Hoy en día, no es el caso, como todas las arquitecturas modernas soportan una instrucción múltiple la cual toma cerca de dos ciclos para completarla.

Otra razón para que las multiplicaciones se las considerase caras era que el IDEA y sus variantes insistieron en ejecutar multiplicaciones en el campo de los enteros módulo 2^{16} +1. De ahí, que cada multiplicación tenía que ser codificada en software como una secuencia de operaciones, incluyendo una “multiplicación nativa” módulo 2^{16} y unas operaciones adicionales.

En el MARS se usan “multiplicaciones nativas“ módulo 2^{32} , en conjunto con rotaciones de acuerdo al dato, para obtener alta seguridad. La fuerza principal criptográfica de la multiplicación módulo 2^{32}, es en los bits de mayor orden del producto, ya que cada uno de estos bits depende de casi todos los bits en los operandos en una forma no lineal. También estos bits tienen excelentes propiedades diferenciales. Por lo tanto, en el MARS se usan los bits de mayor orden para especificar la cantidad de rotación en las operaciones de rotación de este tipo. [38]

Esta combinación original es lo que da al MARS su buena resistencia al criptoanálisis diferencial. Se debe hacer notar que la multiplicación es todavía una operación costosa;

aún en modernos procesadores toma cerca de dos veces otras operaciones y en hardware es aun más costoso. De que se usa esta operación con moderación; en todo el cifrador se ejecutan sólo dieciséis multiplicaciones (comparado con las treinta y dos del IDEA). [38]
Como resultado, se estima que la multiplicación sólo toma cerca del treinta por ciento del tiempo y menos del veinte por ciento en una implementación típica de hardware del MARS, y le toma menos del diez por ciento del tiempo en la implementación del software.

Analizar una multiplicación de dos palabras de datos resulta una tarea muy dura. Como resultado, en el MARS se multiplican palabras de datos con palabras de claves. [38]

Además, en el proceso de expansión de claves, se verifica que las palabras claves, usadas para la multiplicación, no sean “débiles” (como 1, o –1).

4.2.6.3 Seguridad

El MARS provee un alto nivel de seguridad, combinado con una mejor performance que la de otros cifradores existentes. Se estima que el MARS ofrece mejor seguridad que el DES triple. En particular se estima que todos los ataques criptoanalíticos conocidos (incluyendo criptoanálisis lineal y diferencial) requiere más datos de lo que se dispone (2^{128}), y por lo tanto estos ataques son imposibles contra el MARS.

También, los principios en diseño del MARS, hacen que este sea resistente a nuevas técnicas de criptoanálisis. Con respecto a la eficiencia, se calcula que una implementación de software completamente optimizado del MARS se puede correr a 100 Mbits/segundo en las computadoras disponibles de hoy.

Se tienen implementaciones en C, las cuales corren a 65 Mbits/segundo en Pentium-Pro de 200 Mhz y 85 Mbits/segundo en Power PC de 200 Mhz.

4.2.7 ANÁLISIS DEL ALGORITMO CAST 6

4.2.7.1 Descripción y Funcionamiento

El CAST6 es similar a DES, una red de Substitución-Permutación (SPN) construido basándose en el algoritmo de cifrado CAST-128 que aparentemente tiene buena resistencia a criptoanálisis diferencial, lineal y de claves relacionadas. Este cifrador también posee una serie de propiedades criptoanalíticas deseables. Fue creado en 1998 por Carlisle Adams and Stafford Tavares y fue candidato para el Advanced Encryption Standard (AES) y se lo describe en el RFC2612. [37]

El CAST-256 tiene un tamaño de bloques de 128 bits y un tamaño variable de claves (128, 160, 192, 224, o 256 bits) y se compone de 48 rondas.

4.2.7.2 Complejidad

Se le considera altamente seguro debido a la matemática empleada; su complejidad es de O(n!) debido a la complejidad de las permutaciones.

4.2.7.3 Patente

Este cifrador puede ser empleado en todo el mundo por una licencia gratuita con o sin fines comerciales.

4.2.7.4 Resistencia a Ataques

David Wagner, John Kelsey y Bruce Schneider han descubierto un ataque relativo a las llaves en la versión de 64 bits de CAST, que requiere aproximadamente 217 textos planos seleccionados, una consulta relacionada y 248 cálculos fuera de línea. El ataque es infaliblemente el mejor. [37]

4.2.8 Análisis del Algoritmo Blowfish

4.2.8.1 Descripción y Funcionamiento

El algoritmo Blowfish fue diseñado por el Doctor Bruce Schneier en 1993 es rápido, cuenta con llaves de 128, 256 y de 448 bits y es un codificador de bloques simétricos incluido en un gran número de conjuntos de codificadores y productos de cifrado, aunque ningún analizador de cifrados de Blowfish efectivo ha sido encontrado hoy en día. El Dr. Schneier diseñó Blowfish como un algoritmo de uso general (intentando reemplazar al antiguo DES) y libre de problemas asociados con otros algoritmos; además declaró: “Blowfish no tiene patente, y así se quedará en los demás continentes. El algoritmo está a disposición de dominio público, y puede ser usado libremente por cualquiera”.

Algoritmo	Ciclos de Reloj por Ronda	No. de Rondas	No. de Ciclos de Reloj por Byte Cifrado	Patente
Blowfish	9	16	18	Libre, No patentado
Khufu/Khafre	5	32	20	Patentado por Xerox
RC5	12	16	23	Patentado por RSA Data Security
DES	18	16	45	56-bit key
IDEA	50	8	50	Patentado por Ascom-Systec
Triple-DES	18	48	108	

Tabla No. 10 Comparaciones de Velocidad de algunos Cifradores de Bloque sobre un Pentium [40]

Es un cifrador de bloque de 64 bits, tiene una llave escalable desde 32 bits hasta 256 bits, utiliza operaciones simples que son eficientes en los microprocesadores: or-exclusivo, adición, operaciones de búsqueda de tabla, multiplicación modular. No está diseñado para utilizar cambios de longitud de variable o permutaciones bit-wise o saltos condicionales. Se puede implementar en procesadores de 8 bits con un mínimo de 24 bytes de RAM (además de la RAM requerida para el almacenamiento de la clave) y 1KB de ROM. Además emplea claves predefinidas para agilizar las operaciones, consta de un número

variable de iteraciones, no tiene estructura lineal como el DES, posee grandes cajas-S dependientes de la clave lo cual vuelve al algoritmo más resistente a ataques de criptoanálisis diferencial y lineal, estas cajas-S son usadas también por algoritmos como el khufu y el GOST. [40]

El algoritmo consta de dos partes: una de expansión de clave (convierte una llave de 448 bits a un arreglo de subclaves que suman 4168 bytes) y otra de cifrado de datos (utiliza una red Feistel de 16 rondas, cada ronda consta de una permutación dependiente de la clave y una sustitución de pendiente de la llave y de los datos)

4.2.8.2 Complejidad

Debido al uso de redes de Feistel y las operaciones de or-exclusivo, adición, operaciones de búsqueda de tabla y multiplicación modular se clasifica dentro del grupo de algoritmos con problema tipo NP-completo.

4.2.9 ANÁLISIS DEL ALGORITMO CAST 5

4.2.9.1 Descripción y Funcionamiento

El CAST, fue diseñado en Canadá en 1993 por Carlisle Adams y Staffort Tavares, toma el nombre por las iniciales de sus diseñadores.

Este cifrador por bloques ha sido considerado como un algoritmo sólido y es muy utilizado en versiones de GPG y PGP. Su diseño es muy similar al de Blowfish, emplea seis Caja-S (igual que en el algoritmo Data Encryption Standard, pero más largas) de 8*32 bits dependientes de la llave, una función F no reversible y una estructura similar a la red de Feistel (también llamada red de permutaciones). El CAST codifica bloques de 64 bits empleando claves de 64 bits, consta de ocho rondas y deposita prácticamente toda su fuerza en las Caja-S; existen muchas variedades de CAST, cada una con sus Caja-S correspondientes (algunas de ellas secretas). Este algoritmo se ha demostrado resistente a las técnicas habituales de criptoanálisis, y sólo se conoce la fuerza bruta como mecanismo para atacarlo. [37]

Tres diferentes funciones de bucle son utilizadas en CAST-128. Las rondas son como se indica a continuación (donde “D” es la entrada de datos a la función F y “Ia” - “Id” son los bytes más significativos y bytes menos significativos de I, respectivamente). “Si” es la i-ava S-Box (Caja S) y 0 es la salida de la operación.

Las Cajas-S utilizadas en el algoritmo fueron diseñadas de manera no lineal para que no sean vulnerables al criptoanálisis.[37]

El proceso de generación de sub-llaves utilizado en CAST-128 es diferente a los empleados en los cifradores de bloque convencionales. Los diseñadores de CAST han hecho a las sub-llaves tan resistentes a ataques criptoanalíticos, como se tiene conocimiento. Otra función interesante del CAST-128 es la función F que difiere de bucle en bucle, agregando fortaleza criptoanalítica.[37]

Ha sido aprobado por el gobierno canadiense para ser usado por el Communications Security Establishment.

El CAST-128 es un cifrador de 12 o 16 rondas que se basa en la red de Feistel con bloques de 64 bits y tamaños de clave entre 40 y 128 bits (pero con sólo incrementos de 8 bits). Las 16 rondas completas se usan cuando la clave tiene un tamaño mayor de 80 bits. Incluye unas largas S-Boxes de 8x32 bits basadas en funciones bent, rotaciones dependientes de clave, adición y sustracción modular y operaciones XOR. Hay tres tipos alternativos de funciones de ronda, pero son de estructura similar y se diferencian sólo en la elección del tipo exacto de operación (XOR, adición o sustracción) en varios puntos.[37]

4.2.9.2 Patente

A pesar de que Entrust Technologies (Parte de Nortel Telecom) posee la patente del procedimiento de diseño de CAST; el CAST 5 y el CAST 6 están disponibles globalmente con licencias para usos comerciales y no comerciales ya que generosamente lo han permitido utilizar.

4.2.9.3 Resistencia a Ataques

La manera más directa para intentar quebrar un algoritmo de cifrado es utilizando la técnica de fuerza bruta. Dado un mensaje cifrado, intentar todas las claves posibles, que aproximadamente son 7.2×10^{16} claves. Ante esto, un ataque de fuerza bruta parece impráctico, asumiendo que, en promedio, la mitad del espacio de claves debe ser alcanzado. [37]

En 1977, Diffie and Hellman postularon que existía la tecnología para construir una máquina de procesamiento paralelo con un millón de dispositivos de cifrado, cada uno podría procesar un proceso de cifrado por microcontroladores. Esto nos aproximaría a las 10 horas de tiempo de búsqueda de la clave correcta. Los autores estimaron que el costo de construir una máquina semejante costaría aproximadamente $20.000.000 de dólares en 1977.

Si la única forma de ataque a un algoritmo criptográfico es la fuerza bruta, se puede estimar el tiempo que tardará en quebrarse este algoritmo. Por ejemplo, para una clave de 128 bits (lo cual es común y utilizado en CAST5) tomaría más de 1019 años quebrantar la seguridad del código, utilizando el cracker de la Electronic Frontier Foundation, incluso si se acelera al crackeador en un factor de 1 trillón, tomaría más de 10 millones de años quebrantar el código. Por lo tanto una clave de 128 bits da como resultado un algoritmo inquebrantable por fuerza bruta.[37]

Pero hay otra línea de ataque llamada criptoanálisis. El criptoanálisis es el intento de explotar la estructura interna del algoritmo de cifrado, reduciendo la clave de cifrado dados un número de pares de textos cifrados y textos planos. Es un área con mucho movimiento y en desarrollo y como resultado hay que es difícil diseñar un algoritmo altamente resistente al criptoanálisis. Es importante destacar que no todos los algoritmos con igual tamaño de claves son iguales. [37]

Biham y Shamir anunciaron el descubrimiento del criptoanálisis diferencial (una técnica poderosa de ataque a cifradores de iteración). Subsecuentemente, Matsui anunció el descubrimiento del criptoanálisis lineal y luego Biharn el ataque de claves relacionadas. Todos estos ataques utilizan conocimiento de la estructura del cifrador para acumular

información de un número extremadamente largo de operaciones de cifrado para obtener información de la clave utilizada.[37]

Es de observar que el éxito de estos ataques depende del número de operaciones de cifrado y decrece rápidamente con el número creciente de iteraciones.

4.2.9.4 Complejidad

Existen diferentes modos de operación que garantizan diversos grados de confidencialidad e integridad en los datos manejados.

Modo ECB

El más sencillo de los modos de uso es el modo ECB, en el cual los mensajes se dividen en bloques y cada uno de ellos es cifrado por separado. La desventaja de este método es que a bloques de texto en claro idénticos les corresponde bloques idénticos de texto cifrado, de manera que se pueden reconocer estos patrones como guía para descubrir el texto en claro a partir del texto cifrado. De allí que no sea recomendado para su uso en protocolos cifrados. [37]

Modo CBC

En el modo CBC, a cada bloque de texto en claro se le aplica la operación XOR con el bloque cifrado anterior antes de ser cifrado. De esta forma, cada bloque de texto cifrado depende de todo el texto en claro procesado hasta este punto. [37]

Modos CFB y OFB

Estos modos hacen que el cifrado en bloque opere como una unidad de flujo de cifrado: se generan bloques de flujo de claves, que son operados con XOR y el texto en claro para obtener el texto cifrado. Al igual que con otras unidades de flujo de cifrado, al intercambiar un bit en el texto cifrado produce texto cifrado con un bit intercambiado en el texto en claro en la misma ubicación.[37]

Modo CTR

Al igual que OFB, el modo contador convierte una unidad de cifrado por bloques en una unidad de flujo de cifrado. Genera el siguiente bloque en el flujo de claves cifrando valores sucesivos de un contador. El contador puede ser cualquier función sencilla que produzca una secuencia de números donde los resultados se repiten con muy baja frecuencia, si bien la operación más usada es un contador, el modo CTR tiene características similares al OFB, pero permite también el uso de una propiedad de acceso aleatorio para el descifrado. [37]

Es un algoritmo fuerte y su complejidad es de O(n!), lo que indica que se considera que se basa en operaciones que forman un problema intratable desde el punto de vista computacional.

4.2.9.5 Eficiencia Hardware

A continuación se presentan algunos de estos algoritmos que fueron implementados en Java (utilizando la librería Cryptix). En una prueba efectuada en un computador Pentium2/266 con Linux, se obtuvo la siguiente tabla comparativa: [37]

Algoritmo	Cifrado (1MByte)		Establecimiento de la Clave (ms)
	Tiempo (ms)	Velocidad (Kbps)	
Blowfish	20506	409	79290
CAST5	23772	352	976
DES	48629	172	519
TripleDES	160807	52	1790
IDEA	43409	193	734
LOKI91	31071	269	76
RC2	43329	193	790
RC4	12945	648	2382
SAFER	41442	202	2219
Square	29610	283	2166

Tabla No 11 Comparativa de Velocidades de Algunos Cifradores de Bloque [37]

4.2.10 ANÁLISIS DEL ALGORITMO DES

4.2.10.1 Descripción y Funcionamiento

Este algoritmo simétrico cifra bloques de 64 bits de longitud con una clave de 64 bits de longitud. Dentro de la clave el último bit de cada byte es de paridad, con lo que se tiene que la clave en realidad es de 56 bits, esto hace que haya 256 posibles claves para este algoritmo. Este algoritmo utiliza una función denominada SBB (Standard Building Block o constructor estándar de bloques), la cual requiere como entrada un bloque de 64 bits y una clave de 48 bits, produciendo una salida de 64 bits. El DES requiere 16 funciones SBB.[39]

Se tiene la clave original, K, de 64 bits (56 en realidad). De ella se extraen 16 subclaves K_i de 48 bits de longitud. Los pasos a seguir son:

1. Aplicar una permutación original, IP, a cada bloque de 64 bits. Produciendo una salida B_j de 64 bits.

2. Se pasa B_j con la subclave K_1 por la primera función SBB, la salida se pasa por la segunda función SBB con la subclave K_2 y así sucesivamente con los 16 SBB.

3. A la salida del último SBB se le aplica la permutación IP^{-1}. De donde se obtiene el texto cifrado.

Para descifrar se toma como entrada el texto cifrado y se le aplica las subclaves K_i en orden inverso, es decir en la primera función SBB se utiliza K_{16}, en el segundo y así sucesivamente. [39]

Antes de entrar en un estudio detallado de este algoritmo es necesario repasar sus modos de operación y su uso.

Sus modos de operación son:

1. ECB (Electronic Code Book): Siempre se cifra un bloque del mismo modo, se puede crear un diccionario o libro de códigos, se usa para mensajes cortos, de menos de 64 bits.

2. CBC (Cipher Block Chaining): Agrega retroalimentación. Al bloque siguiente se le hace XOR con el bloque de texto cifrado anterior, se usa para mensajes largos.

3. CFB (Cipher Feedback): Como cifrado de flujo auto-sincronizado. Puede procesar varios bloques pequeños a la vez, se usa para cifrar bit por bit ó byte por byte.

4. OFB (Output Feedback): Corre un cifrador de bloque como un difrador de flujo síncrono. Similar a CFB pero n bits del bloque de salida anterior sirven de retroalimentación interna, el mismo uso anterior pero evitando propagación de error.

4.2.10.2 Resistencia a Ataques

El DES (Data Encryption Standard) ya no es una opción viable para cifrado, su llave de 56 bits puede ser encontrada en cuestión de días con una máquina crackeadora económica. [39]

El DES, fue un diseño de unidad de cifrado por bloques de gran influencia. Fue desarrollado y publicado por IBM y publicado como estándar en 1977. El Advanced Encryption Standard (AES) es un sucesor de DES, adoptado en 2001.

Una demostración que evidencia la vulnerabilidad de DES fue el concurso “secret-key” sugerido por los Laboratorios RSA. El concurso, donde se ofrecía un premio de US 10.000 dólares, implicaba encontrar una llave DES dado un texto plano cifrado de un mensaje desconocido. RSA propuso este concurso el 29 de Enero de 1997. En respuesta al concurso, Rock Verser (un consultor independiente) desarrolló un programa y lo distribuyó por Internet. El proyecto unió numerosas máquinas en Internet, creció hasta más de 70000 sistemas, cada computadora era un voluntario suscripto. El proyecto comenzó el 18 de Febrero de 1997 y finalizó 96 días más tarde cuando la clave correcta fue encontrada tras examinar cerca de un cuarto de las claves posibles. Esta competencia demuestra el poder de la computación distribuida compuesta por numerosas computadoras personales con el objetivo de atacar problemas criptográficos. [39]

El DES fue declarado muerto en Julio de 1998, cuando la Electronic Frontier Foundation (EFF) anunció que ellos habían quebrado una nueva competencia de DES usando una máquina crackeadora de DES especializada construida por US 250.000 dólares; el ataque llevó solamente 3 días. La EFF publicó una descripción detallada de la máquina,

posibilitando a otros construir su propio crackeador. Con el transcurso del tiempo el precio y potencia del hardware disminuye, quitándole sentido a DES. [39]

Dada la potencial vulnerabilidad de DES a un ataque de fuerza bruta ha habido considerable interés en encontrar alternativas de utilizar claves más largas e invulnerabilidad a los ataques de criptoanálisis ya que no se le ha encontrado alguna debilidad grave desde el punto de vista teórico y porque su principal debilidad radica en el tamaño de la clave (codifica bloques de 64 bits empleando claves de 56 bits). [39]

A continuación se mostrará una comparación de los diferentes ataques realizados al algoritmo DES: [39]

No. de Rondas	Texto Plano Escogido	Texto Plano Conocido	Texto Plano Analizado	Análisis de Complejidad
8	2^{14}	2^{38}	4	2^{9}
9	2^{24}	2^{44}	2	2^{32}
10	2^{24}	2^{43}	2^{14}	2^{15}
11	2^{31}	2^{47}	2	2^{32}
12	2^{31}	2^{47}	2^{21}	2^{21}
13	2^{39}	2^{52}	2	2^{32}
14	2^{39}	2^{51}	2^{29}	2^{29}
15	2^{47}	2^{56}	2^{7}	2^{37}
16	2^{47}	2^{55}	2^{36}	2^{37}

Tabla No. 12 Ataques Diferenciales Criptoanalíticos contra el DES [39]

4.2.10.3 Complejidad

Debido a las operaciones realizadas con la entrada de 64 bits, su orden de complejidad es constante O(n), ya que las operaciones de permutación y de inserción son lineales en tiempo de ejecución, además se le suma la complejidad del Acuerdo de clave $O(n^2)$ y se le clasifica como perteneciente a los algoritmos con problema P.

4.2.11 ANÁLISIS DE LAS VARIANTES DEL DES

A pesar de su caída, DES sigue siendo ampliamente utilizado en multitud de aplicaciones, como por ejemplo las transacciones de los cajeros automáticos. De todas formas, el problema real de DES no radica en su diseño, sino en que emplea una clave demasiado corta (56 bits), lo cual hace que con el avance actual de las computadoras los ataques por la fuerza bruta comiencen a ser opciones realistas. Mucha gente se resiste a abandonar este algoritmo, precisamente porque ha sido capaz de sobrevivir durante veinte años sin

mostrar ninguna debilidad en su diseño, y prefieren proponer variantes que, de un lado evitarán el riesgo de tener que confiar en algoritmos nuevos, y de otro permitirán aprovechar gran parte de las implementaciones por hardware existentes de DES. [39]

4.2.11.1 DES Múltiple

Consiste en aplicar varias veces el algoritmo DES con diferentes claves al mensaje original. Se puede hacer ya que DES no presenta estructura de grupo. El más común de todos ellos es el Tripe-DES (TDES), que responde al siguiente proceso: se codifica con la subclave k_1, se decodifica con k_2 y se vuelve a codificar con k_1. La clave resultante es la concatenación de k_1 y k_2, con una longitud de 112 bits. [39]

El TDES usa una clave de 168 bits efectivos y 24 bits de paridad, aunque se ha podido mostrar que los ataques actualmente pueden romper a TDES con una complejidad de 2^{112}, es decir se deben efectuar al menos 2^{112} operaciones para obtener la clave a fuerza bruta, además de la memoria requerida. En muchas aplicaciones se ha optado por TDES ya que es muy fácil Interoperar con DES y proporciona seguridad a mediano plazo, según lo publicó en unas notas de clase el profesor Rafael Barzanallana de la Universidad de Murcia (España). [39]

En cuanto a su seguridad se ha podido mostrar que los ataques actualmente pueden romper al TDES con una complejidad de 2^{112}, es decir efectuar al menos 2^{112} operaciones para obtener la clave a fuerza bruta, además de la memoria requerida. [39]

En cuanto a su complejidad del TripleDes se considera como un algoritmo probabilístico ya que incorpora claves pseudoaleatorias generadas en ciertos estados de su proceso.

4.2.11.2 DES con Subclaves Independientes

Consiste en emplear subclaves diferentes para cada una de las 16 rondas de DES. Puesto que estas subclaves son de 48 bits, la clave resultante tendría 768 bits en total. Empleando criptoanálisis diferencial, esta variante podría ser rota con 261 textos planos escogidos, por lo que en la práctica no presenta un avance sustancial sobre DES estándar.

4.2.11.3 DES Generalizado

Esta variante emplea n trozos de 32 bits en cada ronda en lugar de dos, por lo que se aumenta tanto la longitud de la clave como el tamaño de mensaje que se puede codificar, manteniendo sin embargo el orden de complejidad del algoritmo. Se ha demostrado sin embargo que no sólo se gana poco en seguridad, sino que en muchos casos incluso se pierde.

4.2.11.4 DES con S-Cajas Alternativas

Consiste en utilizar S-Cajas diferentes a las de la versión original de DES. En la práctica no se han encontrado S-Cajas mejores que las propias del DES. De hecho, algunos estudios han revelado que las S-Cajas originales presentan propiedades que las hacen

resistentes a técnicas de criptoanálisis que no fueron conocidas fuera de la NSA hasta muchos años después de la aparición del algoritmo. [39]

A continuación se presenta un resumen de los algoritmos anteriormente explicados, mediante la Tabla No.13

Algoritmos	Eficiencia (Clocks)	Longitud de las Claves (bits)	Resistencia a Ataques	Nivel de Complejidad	Orden de Complejidad	Patente
AES	440	128, 192, 256,…	Alta	Alto	Clase NP-C	No
IDEA	250	128	Alta	Alto	Clase NP-C	Si
RC6	260	128, 192, 256	Alta	Alto	Clase NP-C	No
TWOFISH	400	256	Alta	Alto	Clase NP-C	No
MARS	390	128 - 1248	Alta	Alto	Clase NP-C	No
CAST6	660	128 - 256	Alta	Alto	O(n!)	No
BLOWFISH	409	128, 256, 448	Alta	Alto	Clase NP-C	No
CAST5	352	40 - 128	Alta	Alto	Clase NP-C	No
DES	172	64	Muy Baja	Bajo	Clase P, $O(n^2)$	No
Triple-DES	516	112	Media	Medio	Clase P, $O(n^2)$	No

Tabla No. 13 Tabla Comparativa de los Algoritmos Simétricos

4.3 ANÁLISIS DE LOS ALGORITMOS ASIMÉTRICOS

En este tipo de algoritmos al conocerse la clave pública, el problema común sería conocer la clave privada, por esta razón se considera que tienen una complejidad agregada de O(n) (n: cantidad de datos a procesar) ya que la clave pública se conoce y solo está la probabilidad de encontrar la clave privada.

4.3.1 ANÁLISIS DEL ALGORITMO RSA

4.3.1.1 Descripción y Funcionamiento

Rivest, Shamir and Adleman lograron en 1978 el que en la actualidad es el más difundido y más versátil criptosistema de clave pública; que ofrece secreto y autenticación, proporcionando un soporte completo para la distribución de claves y de firmas.

Su funcionamiento, de manera general se muestra a continuación:

1. Cada usuario i elige dos numeros primos, p y q, que han de ser suficientemente grandes (hoy en día todos los computadores tienen métodos para fabricar números primos pseudoaleatorios), dependiendo de la capacidad de computación de los posibles criptoanalistas, aproximadamente de 150 cifras. [2][41]

2. Se halla n, como producto $n = p \cdot q$

3. Se halla $z = \Phi = (p-1)(q-1)$, que es la función del indicador de Euler.

4. Se elige un número d menor que z, tal que su mcd sea zd, es decir, d y z sean primos entre sí, o primos relativos: en la práctica se coge d primo directamente, y mayor que p y q, (mayor que el mayor de los dos), con lo que no caben dudas de que sean primos entre ellos. Otro método es el algoritmo de Euclides, para hallar el mcd, lo que evita factorizar ambos números. [2][41]

5. Se obtiene un número e, tal que $1<e<z$, $e \cdot d = 1 \pmod z$, número que existe y es único.

6. La clave pública está constituida por P(n, e), la cual es enviada a los demás usuarios, y la secreta es S(n, d), que ha de ser conservada por i.

7. Para cifrar, se utiliza la función: $C \equiv M^e \pmod n$, siendo M el mensaje original y C el mensaje codificado.

8. El usuario i puede descifrar cualquier mensaje que le haya enviado otro usuario con: $M = C^d \pmod n = (M^e \bmod n)^d \bmod n$.

4.3.1.2 Complejidad

El algoritmo RSA se fundamenta en el hecho de que factorizar números muy grandes es un problema de difícil solución. Si un intruso que quisiera quebrar el algoritmo fuese capaz de factorizar n (parte de la clave pública), entonces podría utilizar estos factores para deducir rápidamente e y d. Por lo tanto, si fuese fácil factorizar números grandes, sería fácil romper RSA. Al ser difícil factorizar números muy grandes se cree que es difícil romper el algoritmo RSA (aunque no se ha demostrado que factorizar números muy grandes sea en realidad un problema difícil).

4.3.1.3 Seguridad

Para lograr la máxima seguridad, es necesario utilizar enteros de más de 100 dígitos de longitud, pues factorizar números más pequeños es posible. Según asegura el Dr. Bruce Schneier en Applied Cryptography, en 1996, un número de 129 dígitos decimales está en el borde de la tecnología factorizadora. Además, se debe asegurar que el producto (p-1)(q-1) no tiene factores primos pequeños. [2][41]

La utilidad específica de este criptosistema es idónea para la documentación y firma electrónica, y para la generación y transacción del dinero electrónico. Desde el punto de vista tecnológico y matemático, la seguridad del criptosistema RSA depende de la suposición de que calcular D(d,n) a partir de C(e,n) es equivalente a factorizar n = pq. Si un criptoanalista puede factorizar n, es decir, encontrar los p y q usados, entonces puede calcular (p-1)(q-1) y le es fácil encontrar el "exponente secreto" d a partir del exponente público e; aunque no se ha demostrado que la única manera de llegar a D(d,n) a partir de C(e,n) sea ésta. La factorización de n no es, de momento, computacionalmente factible si p y q son del orden de 100 dígitos decimales cada uno. [2][41]

El aumento de la potencia de cálculo en estos últimos años no puede ser ignorada, ni el hecho de que cientos de estaciones de trabajo en todo el mundo hayan dedicado sus ciclos ociosos a esta tarea durante más de seis meses.

Es una especulación afirmar que la seguridad del RSA dependa sólo del problema de la factorización de grandes números, puesto que nunca se ha demostrado que se necesiten los factores primos de n para calcular m partiendo de c y e, y son concebibles formas completamente diferentes de criptoanalizar el RSA que todavía puedan ser descubiertas.

En general, se elige como clave pública el menor de los dos exponentes a fin de conseguir que el proceso de cifrado sea el más rápido (más que el descifrado, el cual, a su vez lo es más que la generación de claves). El cifrado, que utiliza la clave pública, tiene una complejidad de $O(k^2)$, el descifrado $O(k^3)$ y la generación de claves $O(k^4)$, en donde k es la longitud en bits del módulo. [2][41]

Una práctica habitual es elegir como clave pública un exponente pequeño, típicamente 1001 ó 10001, variando el módulo.

La función $y \equiv g^x \pmod p$, donde g, $x \in \zeta$ y p es un número primo (de más de 100 dígitos, según recomendaciones) se conoce como “exponenciación modular” y su complejidad computacional es O(log p).
Su función inversa es: $x \equiv \log_g y \pmod p$ y su complejidad computacional es de tipo exponencial $O(e^{\sqrt{(\log p \log \log p)}})$ y cuando p cumple con las recomendaciones de tamaño se hace prácticamente imposible su cálculo, esta función se conoce como “logaritmo discreto”.

Pero la vulnerabilidad de un criptosistema basado en el algoritmo RSA puede encontrarse en el protocolo. Existen varios escenarios ideales para el ataque con texto en claro a elegir y usos imprudentes que debilitan el criptosistema y que hacen recomendable tener en cuenta las siguientes restricciones que contemplan éxitos criptoanalíticos ya conocidos y reproducibles:

1. El conocimiento de una pareja de exponentes de cifrado/descifrado para un módulo dado permite el ataque por factorialización del módulo.

2. El conocimiento de una pareja de exponentes de cifrado/descifrado para un módulo dado permite también calcular otros pares de cifrado/descifrado sin haber tenido que factorizar n.

3. No debe utilizarse un módulo común en una red de comunicaciones por los riesgos obvios de las restricciones 1ª y 2ª.

4. Los mensajes deben de completar la parte no utilizada de los bloques con secuencias aleatorias para evitar ataques sobre pequeños exponentes de cifrado.

5. El exponente de descifrado debe ser grande.

Cumpliéndose los requisitos anteriores, nadie ha demostrado que la seguridad del criptosistema RSA sea cuestionable con carácter general, ya que se confía en la dificultad de la factorización y de los logaritmos discretos, problemas que podrían dejar de ser “problemáticos” si avanza la computación cuántica. En teoría puede haber problemas en cuanto a los primos generados para obtener la clave, ya que estos números son generados con pruebas estadísticas, es decir, no se sabe a ciencia cierta si el número generado es primo, sino que se sabe que lo es con alta probabilidad. [2][41]

Si lo números generados no son realmente primos, el algoritmo simplemente deja de funcionar.

Otras vulnerabilidades a considerar son:

- Claves demasiado cortas: si la clave es menor de 768 bits, se podría llegar a descifrar el mensaje en un lapso de tiempo suficiente. Actualmente se recomiendan claves de al menos 1024 bits.

- Claves débiles: en RSA siempre existe un número de claves que dejan el mensaje tal como estaba. Realmente el número de estas claves es muy pequeño y no suponen un problema.

- Ataques de suplantación de identidad (man in the middle): Un atacante podría situarse entre los dos extremos de la comunicación, e intercambiar las claves que se envían ambos extremos introduciendo las suyas. Esto provoca que ni A ni B se enteren realmente de que alguien está escuchando sus mensajes (o incluso modificándolos). Para solucionar estos ataques, se han implementado grupos llamados anillos de confianza, que son grupos de personas que firman sus claves para garantizar su autenticidad (lo veremos más adelante). Este ataque no sólo afecta a RSA sino a cualquier otro algoritmo asimétrico.

4.3.1.3.1 Ataques que miden el tiempo

Kocher describió un nuevo ataque sobre RSA en 1995: si el atacante Eve conoce suficientemente detalles sobre el hardware de Alicia y conoce el tiempo en descifrar varios textos cifrados conocidos, ella puede deducir que el descifrado de la clave d es rápida. Este ataque puede también ser aplicado contra la autentificación RSA scheme. En el 2003, Boneh y Brumley demostraron un ataque práctico capaz de recuperar factorizaciones RSA a través de una conexión de red (ej: Secure socket Layer (SSL) – transmisión segura de páginas web cifradas). Este ataque toma información escapada por la optimización del Teorema chino del resto usado en muchas implementaciones del RSA.

Una forma de frustrar estos ataques es asegurar que la operación de descifrado tome un tiempo constante en todos los textos cifrados. Sin embargo, esta aproximación puede significar la reducción del funcionamiento. En vez, de más implementaciones RSA una técnica alternativa conocida como blindaje criptográfico. RSA usa propiedades multiplicativas del RSA y con el blindaje aplicado, el tiempo de descifrado se relaciona con el tiempo del texto cifrado así que el ataque de sincronización falla. [27]

4.3.1.3.2 Ataques Adaptativos al Texto Cifrado Elegido

En 1998, Daniel Bleichenbacher describió el primer ataque adaptativo eligiendo textos cifrados contra mensajes cifrados con RSA usando el PKCS #1 v1 padding scheme (un padding scheme aleatorio, que crea estructura para un mensaje cifrado-RSA, así es posible determinar si el mensaje descifrado es válido) Debido a defectos con PKCS #1 scheme, Bleichenbacher hizo posible montar un ataque práctico contra la implementación RSA del protocolo Secure Socket Layer, y recupero la sesión de claves. Como resultado

de este trabajo, los criptógrafos ahora recomiendan el uso de seguridad padding scheme con Optimal Asymmetric Encryption Padding, los laboratorios RSA han realizado nuevas versiones de PKCS #1 que no son vulnerables a estos ataques. [27]

4.3.1.4 Patente

Este criptosistema estuvo patentado sólo en los EEUU hasta el 2000. El algoritmo RSA es considerado como estratégico por el Departamento de Defensa Norteamericano. El International Traffic in Arms Regulations (ITAR) considera a la criptología como alta tecnología de potencial doble uso, y obliga a cualquier fabricante o comerciante de material criptográfico a registrarse en el Departamento de Estado, incluso si no tiene intención de exportar sus productos. [29]

También se conocen protestas y presiones por la simple publicación de artículos y trabajos de investigación sobre criptología, especialmente sobre la de clave pública y el algoritmo RSA en particular.

4.3.2 ANÁLISIS DE ELGAMAL

4.3.2.1 Descripción y Funcionamiento

ElGamal consta de tres componentes : el generador de claves, el algoritmo de cifrado, y el algoritmo de descifrado. El algoritmo ElGamal fue ideado en un principio para producir firmas digitales, aunque después se extendió su uso para utilizarlo en el cifrado de mensajes.

Para generar un par de llaves, se escoge un número primo n y dos números aleatorios p y x menores que n. Se calcula entonces: $y = p^x \bmod n$; la llave publica es (p, y, n), mientras que la llave privada es x. Escogiendo n primo, se garantiza que sea cual sea el valor de p, el conjunto {p, p2, p3 . . .} es una permutación del conjunto {1, 2,..., n − 1}. Aunque este algoritmo es bastante versátil y se puede emplear también un n no primo, siempre que el conjunto generado por las potencias de p sea lo suficientemente grande. [4] [17]

4.3.2.2 Seguridad y Complejidad

ElGamal es un ejemplo simple de un algoritmo dominante y seguro para cifrar; es probabilístico, significando que un solo texto plano se puede cifrar a muchos textos cifrados posibles, con la consecuencia que un cifrado de general ElGamal produce una extensión de 2:1 de tamaño de texto plano al texto cifrado.

La seguridad de ElGamal se reclina, en parte, en la dificultad de solucionar el problema discreto del logaritmo, el cual como se vió tiene una complejidad de $O(e^{\sqrt{(\log p \log \log p)}})$. Específicamente, si el problema discreto del logaritmo se pudiese solucionar eficientemente, entonces ElGamal estaría roto; sin embargo, la seguridad de ElGamal confía realmente en los estudios y teorema dee Diffie-Hellman. [4] [17]

En comparación con el algoritmo RSA, al emplear ambos la exponenciación, la velocidad de cifrado y descifrado es similar. La generación de claves con ambos métodos también lo es, y el la búsquedas de números primos apropiados es la etapa más crítica para ambos.

La seguridad de ElGamal es en la actualidad algo más difícil de garantizar que la del RSA. El mayor ataque al RSA, la factorización, es un problema que ha sido estudiado durante siglos, pero son mucho más recientes, y menos profundos, los estudios para criptoanalizar el esquema del ElGamal.

4.3.3 Análisis del Algoritmo Rabin

4.3.3.1 Descripción y Funcionamiento

Este algoritmo se basa en la complejidad de la factorización. Sin embargo, la ventaja del algoritmo Rabin es que se ha demostrado que la complejidad del problema en el que se basa es tan duro como la factorización de enteros, cosa que se desconoce si es cierto en el caso del RSA simple. El inconveniente que tiene es que cada salida de la función de Rabin puede ser generado por 4 posibles entradas, y si cada salida es un texto cifrado se requiere un tiempo extra en el descifrado para identificar cual de las 4 posibles entradas era el correcto texto en claro. El algoritmo se publicó en enero de 1979 por Michael O. Rabin. [4] [17]

Cada usuario elige una pareja de primos p y q cada uno igual a 3 módulo 4 y forma el producto n = p * q.

La clave pública será n y la clave privada p y q

Función de cifrado: $c = E(m) = m^2 (\text{mod } n)$

Función de descifrado: Dado un texto cifrado c, se calculan las 4 raíces cuadradas de c mod n. Una de ellas será el propio m, otra es n-m y las otras son sus negativas. Para hallar m deberá hacerse a partir de las otras 3 raíces.

Si, como se dijo antes, los primos p y q son iguales a 3 módulo 4 existen formulas sencillas para hallar las raíces, calculando previamente mediante el TCR los valores para a y b que satisfacen la ecuación: a * b + p * q = 1 [4] [17]

$s = c((q + 1) / 4) \text{mod} q$

$r = c((p + 1) / 4) \text{mod} p$

$x = (a * p * s + b * q * r) \text{mod} n$

$y = (a * p * s - b * q * r) \text{mod} n$

Las 4 raíces serán m1=x, m2=-x , m3=y y m4=-y

Para distinguir cual m codifica el mensaje, será necesario incluir información redundante.

4.3.3.2 Seguridad

Es interesante el hecho de que para el caso en que un primo p sea congruente a 1 módulo 4 no se conoce ningún algoritmo polinomial determinista para hallar las raíces cuadradas de residuos cuadráticos modulo p.

Un atacante no sabría cual de las cuatro raíces es la correcta, pero el receptor tampoco. Esto se resuelve acordando ciertas reglas de redundancia en el mensaje para poder distinguir cual de las cuatro raíces corresponde al mensaje original.

4.3.3.3 Complejidad

Este es un problema NP-completo y se cree que es un problema equivalente a factorizar el entero n, aunque no ha sido probado.

A continuación, en la Tabla No. 14, se presenta el resumen de las principales características de los principales algoritmos asimétricos:

Algoritmos	Longitud de las Claves (bits)	Resistencia a Ataques	Nivel de Complejidad	Orden de Complejidad	Patente
RSA	1024-2048	Alta	Alto	$O(k^2), O(k^3), O(k^4), O(e^{\sqrt{(\log p \log\log p)}})$	No
ElGamal	1024-2048	Alta	Alto	$O(e^{\sqrt{(\log p \log\log p)}})$	No
Rabin	1024-2048	Alta	Alto	Clase NP-Completo	No

Tabla No. 14 Tabla Comparativa de los Algoritmos Asimétricos

4.4 ANÁLISIS DE LOS ALGORITMOS DE HASH

Estos algoritmos deben analizarse desde el punto de vista computacional y no matemático puesto que son funciones diseñadas para ser contenidas dentro de una llave privada del emisor para generar la firma digital.

Antes de analizar este tipo de algoritmos se definirán algunos tipos de ataque para los mismos, teniendo en cuenta que es inevitable que en esta clase de funciones que transforman cadenas de longitud arbitraria (preimágenes) en cadenas de longitud fija (imágenes), más de una preimágen dará lugar a una misma imágen. Esa clase de conflictos se conoce como “colisiones” entre valores de hash. Para entender los procesos que se describirán a continuación resulta conveniente aclarar que una operación es computacionalmente no factible si no existe un algoritmo de tiempo polinomial para resolverla (complejidad Clase P), o sea esa operación es prácticamente irrealizable con los recursos computacionales disponibles actualmente (complejidad Clase NP).

Se definen cinco tipos de resistencias a los ataques criptoanalíticos contra las funciones hash:

1. Función de hashing resistente a preimágenes: Una función de hash es resistente a preimágenes si dado el digesto z es computacionalmente no factible hallar algún mensaje x tal que h(x) = z. La complejidad de este ataque contra un hashing de n-bits es O(2n).

2. Función de hash resistente a segundas preimágenes: Una función de hash es resistente a segundas preimágenes si dado un mensaje x es computacionalmente imposible hallar un mensaje x' ≠ x tal que h(x) = h(x'). La complejidad de este ataque contra un hash de n-bits es O(2n).

3. Función de hash resistente a colisiones: Una función de hash es resistente a colisiones si es computacionalmente no factible hallar un par de mensajes distintos x, x' tal que h(x) = h(x'). Hay dos variantes: con IV's (initialization vectors) fijos (la habitual) o de IV aleatorio en la cual uno de esos vectores puede variar libremente. La complejidad de este ataque (IV fijo) contra un hash de n-bits es O(2n/2).

4. Función de hash resistente a seudocolisiones: Una función de hash es resistente a seudocolisiones si es computacionalmente no factible hallar un par de mensajes distintos x, x' tal que h(x) = h(x') en las cuales se puedan usar distintos IV's para ambos mensajes x, x'.

5. Función de hash resistente a seudopreimágenes: Una función de hash es resistente a seudopreimágenes si dado el hash z es computacionalmente no factible hallar algún mensaje x y algún IV, tal que h(x) = z.

Una función de hash que cumpla 1. y 2. se denomina función de hash de una vía o función débil de una vía (OWHF) si cumple 2. y 3. (y eventualmente 1.) recibe el nombre de resistente a colisiones o función fuerte de una vía (CRHF). Además ambas categorías se agrupan genéricamente como códigos de detección de modificaciones (MDC's) (usados sólo para control de integridad), otra categoría especial la forma una combinación de las funciones de hash con claves simétricas y que se conocen como códigos de autenticación de mensajes (MAC´s) (una especie primitiva de firma digital, usada para asociar identidad con datos). [42] [29]

Según el Dr Arturo Quirantes en un artículo para el Boletín ENIGMA (publicado en Septiembre del año pasado): Una buena función hash tendrá que cumplir ciertas características. En primer lugar, la discreción: conocido el hash h, debe ser difícil, si no imposible, obtener el valor del mensaje M, cualquier alteración, por pequeña que sea, cambie el valor de h de forma significativa. Idealmente, un cambio de un solo bit en el mensaje debería cambiar al menos la mitad de los bits en h. Esto no es muy difícil, si se sabe mezclar bien los bits… En la práctica, lo que se hace es que el número de bits de la función hash sea lo bastante alto para que la probabilidad de hacer dos mensajes con el mismo hash sea ínfima. En general, para un hash de n bits, la probabilidad de encontrar otro mensaje con el mismo hash que el primero ha de ser el que marca la probabilidad pura y dura: 2^{-n}. Esto es lo que se conoce como resistencia débil a las colisiones: Para un mensaje M dado, ha de ser imposible (o al menos muy difícil) encontrar otro mensaje M' con el mismo resumen. Es decir, no ha de darse H(M)=H(M'). Sin embargo, existe otra condición más restrictiva, conocida como resistencia a las colisiones: Ha de ser difícil hallar al azar dos mensajes M, M' tales que H(M)=H(M').

Los algoritmos MD4 (año 1990), MD5 (año 1991) fueron los primeros de su serie. Luego surgieron como mejoras el SHA-0 (Secure Hash Algorithm, año 1993), SHA-1 (año 1995) y SHA-2 (año 2004) y que forman una familia de algoritmos vinculados por su estructura. El algoritmo SHA-1 (SHA-160) y la familia informalmente conocida como SHA-2 (estándars SHA-224, SHA-256, SHA-384 y SHA-512) surgen para eliminar debilidades halladas en el SHA-0. Estas funciones de hash han sido y son las recomendadas por la Administración Federal de EEUU (SHS: Secure Hash Standard) y por reflejo en gran parte del resto del mundo. El SHA-224 es idéntico al SHA-256 excepto que varían los IV's (initialization vectors) y que se trunca el hash a los 224 bits de la derecha, el SHA-384 es idéntico al SHA-512 con las mismas salvedades del caso anterior. El SHA-224/256 está basado en palabras de 32 bits y el SHA-384/512 en palabras de 64 bits.

El cálculo de una colisión para MD4, según el método de Dobbertin, tiene una complejidad computacional de unas 2^{20} evaluaciones de la función de compresión; si se tiene en cuenta que esta función es extremadamente rápida, se habla de un cálculo de unos pocos segundos. El hecho de que tal compresión solo tenga tres etapas se considera totalmente insuficiente; de hecho, la principal diferencia entre MD4 y MD5 es que este último incluye una cuarta etapa en la función de compresión. [42] [29]

El MD5 fue considerado criptográficamente seguro en un principio, pero ciertas investigaciones han revelado vulnerabilidades que hacen cuestionable el uso futuro del MD5. En agosto del 2004, Xiaoyun Wang, Dengguo Feng, Xuejia Lai y Hongbo Yu anunciaron el descubrimiento de colisiones de hash para MD5. Su ataque se consumó en una hora de cálculo con un clúster IBM P690. [42] [29]

Aunque dicho ataque era analítico, el tamaño del hash (128 bits) es lo suficientemente pequeño como para que resulte vulnerable frente a ataques de fuerza bruta tipo cumpleaños. El proyecto de computación distribuida MD5CRK se inició en marzo del 2004 con el propósito de demostrar que el MD5 es inseguro frente a uno de tales ataques, acabó poco después del aviso de la publicación de la vulnerabilidad del equipo de Wang.

El proceso de relleno y codificación del SHA es igual al del MD5 solo que en vez de trabajar un buffer de 128 bits se trabaja con uno de 160 bits; pero el SHA es 232 veces más seguro que el MD5, pero es más lento su cálculo que el MD5. [42] [29]

Debido al descubrimiento de métodos sencillos para generar colisiones de hash, muchos investigadores, como el equipo de seguridad de Microsoft y el Dr. Bruce Schneier recomiendan su sustitución por algoritmos alternativos tales como SHA-1 o RIPEMD-160.

El SHA-1 (Secure Hash Algorithm) es actualmente el estándar mundial de facto de las funciones de hashing; fue desarrollado como parte del Estándar de Hash Seguro (Secure Hash Standard, SHS) y el Estándar de Cifrado Digital (Digital Signature Standard, DSS) por la Agencia de Seguridad Nacional norteamericana (NSA). La primera versión, conocida como SHA, fue mejorada como protección ante un tipo de ataque que nunca fue revelado. El SHA-0 y el SHA-1 producen una salida resumen de 160 bits de un mensaje que puede tener un tamaño máximo de 2^{64} bits. El documento FIPS (Federal Information Processing Standard) que oficialmente lo describe afirma que los principios subyacentes al SHA-1 son similares a los del MD4 de Rivest, dice además que su implementación "puede estar cubierta por patentes en EEUU y el extranjero". En la tecnología PKI se usan extensivamente MD5 y SHA1; por ejemplo forman parte constante del protocolo SSL

v3. El MD5 y SHA-1 son usados por sistemas UNIX para almacenar las contraseñas de los usuarios en forma segura. A su vez la familia RIPEMD128 y RIPEMD160, desarrollados para el proyecto RIPE (RACE Integrity Primitives Evaluation) en 1992 y 1996, se hallan muy distribuidos en Europa. En nuestro País, el MD5 es el principal algoritmo de hashing usado por el sistema bancario, y en general, a nivel mundial, MD5 y SHA-1 son las funciones más empleadas hoy día en la firma digital de documentos para todo tipo de transacciones. [42] [29]

4.4.1 Ataques de Colisión

En la reunión anual de criptología CRYPTO'04 surgieron a la luz una serie de vulnerabilidades de los hashings actuales empleando variantes computacionalmente eficientes del ataque de colisiones diferenciales de Chabaud-Joux. .Un equipo chino acaba de publicar un trabajo interesante: halló una variante computacional eficiente para generar colisiones reales. Los algoritmos involucrados son el MD4, MD5, HAVAL128 y RIPEMD128. Además se han comprometido los algoritmos SHA-0 y RIPEMD160. De hecho se generan pares de mensajes x y x' con igual imagen z = h(x) = h(x'). Esto fue seguido por la publicación de un segundo trabajo aún más significativo por parte de un equipo australiano quien halló por variantes del método Chabaud-Joux ataques de segunda preimagen contra la familia SHA-2. [42] [29]

Los ataques contra HAVAL128, MD4, RIPEMD128 que han sido publicados, son prácticamente idénticos ya que todos emplean este método de "máscara" de DWords, sólo difieren en las potencias de dos empleadas en determinadas posiciones de esa máscara y las posiciones de los valores no nulos dentro de esas máscaras. En síntesis, se trata de una "familia" de ataques de colisiones diferenciales de Chabaud-Joux eficaces contra una "familia" de funciones. [42] [29]

El equipo australiano realizó un ataque de colisiones diferenciales tipo Chabaud-Joux sobre la familia SHA-2, la que hasta ahora se considera invulnerable y reemplazante potencial del SHA-1. El método está basado en la búsqueda de patrones correctivos sobre el registro del digesto para lograr dichas colisiones ronda a ronda. Previamente, otro equipo de investigadores como Gilbert y Hanshuh había hallado colisiones por esta variante de patrones correctivos usando diferencias por función XOR, pero terminó concluyendo que la familia SHA-2 es inmune al ataque de colisiones porque su probabilidad de su ataque era inferior al límite dado por el ataque de cumpleaños. Sin embargo, en el trabajo del equipo australiano se amplió el esquema previo usando diferencias aditivas y esto resultó ser fructífero porque permitió pasar del ataque original por colisiones a un ataque mucho mas severo de segundas preimágenes, razón por la cual este equipo llega a la conclusión que la familia SHA-2 dejó de ser inmune al ataque Chabaud-Joux, no sólo por sus propios métodos sino por lo hallado por Gilbert y Hanshuh quienes usaran los límites teóricos de colisiones. A pesar que este equipo señaló haber obtenido un ataque derivado de segunda preimagen, sólo se indican resultados de los ataques de colisiones y no se indica cómo pasar del mismo a las segundas preimágenes el que queda en términos potenciales hasta que se aporte la suficiente evidencia. [42] [29]

Según el NIST: "Este ataque es de particular importancia para las aplicaciones que usan firmas digitales tales como marcas de tiempo y notarías. Sin embargo, muchas aplicaciones que usan firmas digitales incluyen información sobre el contexto que hacen este ataque difícil de llevar a cabo en la práctica."

A pesar de que 2^{64} suponen aún un número alto de operaciones, se encuentra dentro de los límites de las capacidades actuales de cálculos, y es previsible que con el paso del tiempo romper esta función sea trivial, al aumentar las capacidades de cálculo y al ser más serios los ataques contra SHA-1.

La importancia de la rotura de una función hash radica en que un hash permite crear una huella digital, en teoría única, de un archivo. Si un hash fuese roto podría haber otro documento con la misma huella. La similitud sería equivalente a hacer que hubiese personas que compartiesen las mismas huellas digitales, o peor aún, el mismo ADN: No habría manera de poder diferenciarlos usando su hash. [42] [29]

A pesar de que el NIST contempla funciones de SHA de mayor tamaño (por ejemplo, el SHA-512, de 512 bits de longitud), expertos como el Dr. Bruce Schneier apoyan la idea de buscar una nueva función hash estandarizada que permita sustituir a SHA-1. Otros nombres que se mencionan al respecto son el algoritmo Tiger (de los creadores del algoritmo Serpent) y WHIRLPOOL (de los creadores del Rijndael - AES).

El algoritmo Tiger fue diseñado por Ross Anderson y Eli Biham en 1996, en previsión de eficiencia para plataformas de 64 bits. El tamaño de la función es de 192 bits, aunque hay versiones de 128 y 169 bits de la misma, llamadas Tiger/128 y Tiger/160, que devuelven versiones cortas de la versión Tiger/192.

Se usa frecuentemente en los árboles de hash de Merkle TTH (Tiger Tree Hash). TTH es usado por redes P2P, como Direct Connect y Gnutella.

El algoritmo Tiger tenía la posibilidad de ser incluido en el estándar OpenPGP, aunque ha sido abandonado en favor de RIPEMD-160.

El algoritmo Whirlpool fue diseñado por Vincent Rijmen y Paulo S. L. M. Barreto. El hash ha sido recomendado por el proyecto NESSIE y ha sido adoptado por la Organización Internacional de Estandarización (ISO) y la Comisión Electrotécnica Internacional (IEC) como parte del estándar internacional ISO/IEC 10118-3. [42] [29]

El algoritmo Whirlpool es una construcción Miyaguchi-Preneel basada en una modificación del Advanced Encryption Standard (AES). Dado un mensaje de un tamaño menor de 2^{256} bits, devuelve un hash de 512 bits.

Los autores han declarado que Whirlpool no está patentado (ni lo estará). Puede ser usado libremente para cualquier propósito y las implementaciones de referencia son de dominio público.

El RipeMD-160 (acrónimo de RACE Integrity Primitives Evaluation Message Digest, primitivas de integridad del resumen del mensaje) es un algoritmo de hash de 160 bits desarrollado en Europa por Hans Dobbertin, Antoon Bosselaers y Bart Preneel, y publicados primeramente en 1996. Es una versión mejorada de RipeMD, que estaba basado sobre los principios del diseño del algoritmo MD4, y es similar en seguridad y funcionamiento al más popular SHA-1.[43]

También existen versiones de 128, 256 y 320 bits de este algoritmo, llamadas RipeMD-128, RipeMD-256 y RipeMD-320 respectivamente. La versión 128 bits fue pensada solamente como un reemplazo para el RipeMD original, que eran también de 128 bits,

para mejorar la seguridad, debido al rápido crecimiento de la tecnología. Las versiones de 256 y 320 bits solamente disminuyen la posibilidad de colisiones hash accidentales, y no tienen niveles más altos de seguridad con respecto a RipeMD-128 y RipeMD-160.[43]

RipeMD-160 fue diseñado en la comunidad académica abierta, en contraste con el algoritmo SHA-1, diseñado por la Agencia de Seguridad Nacional estadounidense (NSA). Por otra parte, RipeMD-160 es un diseño menos popular y no ha sido bien estudiado aunque Doctores como Arturo Quirantes y Jorge Ramió Aguirre lo recomiendan. Ninguna patente está asociada al RIPEMD-160.

A continuación se presentarán algunas tablas que muestran de manera más concisa, el rendimiento de los diferentes algoritmos de hash.

FUNCION	bits	Rounds	Velocidad Relativa *estimado*	Fuerza Preimagen	Fuerza contra ataque de Colisión *ataque cumpleaños*
MD4	128	48	1.00	2^{128}	2^{20}
MD5	128	64	0.68	2^{128}	2^{64}
SHA-0	160	80	0.28	2^{160}	2^{80}
SHA-1	160	80	0.28	2^{160}	2^{80}
SHA-224	224	64	0.20	2^{224}	2^{112}
SHA-256	256	64	0.20	2^{256}	2^{128}
SHA-384	384	80	0.15	2^{384}	2^{192}
SHA-512	512	80	0.15	2^{512}	2^{256}
RIPEMD128	128	64	0.39	2^{128}	2^{64}
RIPEMD160	160	80	0.24	2^{160}	2^{80}

Tabla No. 15 Comparación de Rendimiento de Algunos Algoritmos de Hash[44]

FUNCION	bits	Fuerza Ideal de Colisión	Fuerza Residual Post-Ataque de Colisión
MD4	128	2^{20}	1
MD5	128	2^{64}	2^{32}
SHA-0	160	2^{80}	2^{40}
SHA-1	160	2^{80}	2^{39}
SHA-224	224	2^{112}	2^{39} (a) y 2^{9} (b)
SHA-256	256	2^{128}	2^{39} (a) y 2^{9} (b)
SHA-384	384	2^{192}	2^{39} (a) y 2^{9} (b)
SHA-512	512	2^{256}	2^{39} (a) y 2^{9} (b)
RIPEMD128	128	2^{64}	2^{32}
RIPEMD160	160	2^{80}	2^{40}
HAVAL128	128	2^{64}	2^{6}
HAVAL160	160	2^{80}	2^{32}

Tabla No. 16 Cálculos Teóricos de Ataques a Algunos Algoritmos de hash
(a) Valores con estados iniciales por ronda desconocidos del registro de hash
(b) Valores con estados iniciales por ronda conocidos del registro de hash. [44]

Algoritmo	Ciclos de Reloj	Velocidad (MBits/s)	Velocidad (Mbyte/sec)	Performance relativo
MD4	241	191.2	23.90	1.00
MD5	337	136.7	17.09	0.72
RIPEMD	480	96.0	12.00	0.50
RIPEMD-128	592	77.8	9.73	0.41
SHA-1	837	55.1	6.88	0.29
RIPEMD-160	1013	45.5	5.68	0.24

Tabla No. 17 Performance de Implementaciones de Lenguaje Assembler Optimizadas de Funciones Hash sobre un Pentium de 90 MHz con Memoria Flat de 32 bits.[44]

A continuación se presenta un resumen de los algoritmos anteriormente explicados, mediante la Tabla No.18

Algoritmos	**bits**	**Resistencia a Ataques**	**Nivel de Complejidad**	**Patente**
RipeMD320	320	Alta	Alto	No
SHA - 512	512	Alta	Alto	No
Whirlpool	512	Alta	Alto	No
Tiger	192	Alta	Alto	No
RipeMD160	160	Alta	Alto	No
Haval160	160	Alta	Alto	No

Tabla No. 18 Tabla Comparativa de los Algoritmos de Hash

Capítulo 5

Análisis de la Aplicación Desarrollada y Aplicaciones libres existentes.

5.1 Antecedentes

El lenguaje de programación Java presenta una respuesta práctica a las cuestiones de seguridad sobre Internet y empresas como Telefónica Móviles utiliza a Java como núcleo para proporcionar servicios sobre su red GPRS a través de su red europea; además incorpora un robusto modelo de seguridad de principio a fin, desarrollado en estándares abiertos que protegen las redes, las aplicaciones y los dispositivos móviles. Soportando HTTPS y aprovechando los estándares existentes tales como SSL y WTLS (Wireless Transport Layer Security) para permitir la transmisión de datos cifrados, ya que la seguridad sobre la información es uno de los más importantes retos de hoy, requiriendo esquemas de seguridad más flexibles, personalizables y óptimos; los cuales están fuera del alcance de HTTPS. [45]

Es por todos estos requerimientos de seguridad, que tanto el programa JCP (Java Community Proccess) como terceras partes, han desarrollado APIs de seguridad y servicios confiables para Java, algunas de estas son: la especificaciones JSR 177 (Java Specification Request) comúnmente conocida como SATSA (The Security and Trust Services APIs), y la Bouncy Castle lightweight API. [45]

5.1.1 Bouncy Castle Lightweight API

BC (Bouncy Castle) se inició como el esfuerzo de una comunidad para implementar un proveedor JCE (Java Cryptographic Extension) libre y de código abierto. Los desarrolladores de BC han desarrollado su propia API de criptografía liviana, para ser incluida en las clases del proveedor JCE BC. Esta API también puede ser usada de forma standalone, con mínimas dependencias sobre las otras clases de J2SE (Java Standard Edition v2). El paquete BC trabaja especialmente con dos clases núcleo de Java: java.math.BigInteger y java.security.SecureRandom. [45]

La comunidad de BC constantemente optimiza las implementaciones existentes. Por ejemplo, BC 1.16 tiene tres implementaciones AES que proveen un rango de compromisos entre el uso de velocidad y memoria del dispositivo. Desde BC 1.11 a 1.16, la implementación de BigInteger ha mejorado a tal grado que el tiempo utilizado para cifrado RSA es únicamente un cuarentavo del inicial. [45]

La API de criptografía de Bouncy Castle consiste entre otras cosas de: Una API de criptografía liviana en Java, un proveedor para la JCE y JCA (Autoridad Certificadora de Java), una implementación de la JCE 1.2.1, una librería para lectura y escritura de objetos codificados en ASN.1, generadores de certificados X.509 (para las versiones 1 y 3), CRL (Certificate Revocation List. Lista de revocación de certificados.) (versión 2), archivos PKCS12 y atributos de certificados X.509 (para la versión 2).

El modelo de desarrollo ad-hoc de BC, trae algunos problemas: Muchas implementaciones de algoritmos BC viene directamente de libros y existen demasiados

algoritmos y muy pocos desarrolladores voluntarios para optimizarlos. La falta de optimización resulta en un relativamente pobre desempeño, especialmente para algunos algoritmos de llave pública. Sin embargo, BC presenta algunas ventajas, en especial al implementar un proveedor JCE de fuente abierta, ya que permite ver el código fuente, para conocer claramente la forma de cómo usar el API BC en varias tareas y dando a la vez una poderosa herramienta de aprendizaje para desarrollares avanzados.[45]

El archivo .jar de la API BC pesa alrededor de 1 MB. Los términos de licencia libre de BC permiten empaquetar y redistribuir únicamente clases requeridas en las aplicaciones desarrolladas.

5.2 LA APLICACIÓN SEGURIDAD

Este prototipo tiene el propósito de mostrar el funcionamiento de los principales y mejores algoritmos criptográficos existentes; su núcleo central es el paquete prueba, el cual consta de clases como Seguridad.java, que es la aplicación principal, las clases EncArch.java, DecArch.java, CifAsArch.java, DecSimArch.java, Cifres.java, CifAsArch.java y DecAsArch.java, para el cifrado/descifrado simétrico y asimétrico de archivos, las clases Cifradora.java y DescifradoraM.java, PanelPruebas.java y PanelPruebaIn para cifrado/descifrado de texto, las clases Client.java y Server.java para el envío de archivos a través de una red, las clases GenHash.java, HashArch.java e IUcompHash.java para la creación y manejo de funciones hash y finalmente la clase Generado.java que sirve de clase auxiliar para el transporte de información de una clase a otra y para manejar los archivos de información de los parámetros de los algoritmos. (Ver ANEXOS 1 y 3)

El proyecto se basa en un compendio de librerías escogidas mediante consultas realizadas a diferentes Ingenieros que han trabajado en el área de la seguridad computacional con algoritmos criptográficos y con amplia experiencia en el entorno de desarrollo Java, las librerías usadas son:

5.2.1 Librería Fastper

Es la encargada de guardar información en los diferentes archivos (entrada, secret, entArchHash, CifASHash, entradaArch y CifAsAr) con etiquetas xml, las cuales delimitan la información correspondiente a los parámetros de los diferentes algoritmos criptográficos y facilitan la búsqueda de dicha información. Esta librería fue diseñada y creada en la empresa Seratic perteneciente al grupo de Parquesoft de Popayán, la cual autorizó el uso de la misma para esta aplicación.

5.2.2 Paquete Rabin

Es un proyecto descargado de Internet el cual implementa el algoritmo asimétrico de Rabin, el cual no viene incluido en los proyectos de cifrado de java, este proyecto fue desarrollado por Robin Abraham y es de uso libre.

5.2.3 Librería Cryptix

Es una librería muy mencionada en los distintos proyectos de criptografía con java y que ahora viene por defecto en el JDK 1.6; de ésta se utilizaron los siguientes algoritmos simétricos: DES, Blowfish, SKIPJACK, Square, MARS y TripleDES (DESede).

5.2.4 Librería BouncyCastle

Esta librería también es bastante utilizada en el desarrollo de aplicaciones de seguridad con java, también está incluida ya en el JDK1.6; de ésta se utilizaron los siguientes algoritmos:

Simétricos: AES(Rijndael), Camellia, CAST6, IDEA, RC6, RC4, Serpent, Twofish, TEA, SEED y Noekeon.

Asimétricos: RSA y ElGamal (solo para cifrado/descifrado de texto, ya que no se consiguió una aplicación que creara certificados con este algoritmo).

Hash: MD2, MD4, MD5, RIPEMD128, RIPEMD160, RipeMD320, SHA, SHA0, SHA1, SHA-512, Tiger, Whirlpool y GOST3411.

5.2.5 Librerías Javax-crypto y Javax-Security

Son las encargadas de controlar las funciones, parámetros, comunicaciones y las gestiones para invocar los diferentes algoritmos criptográficos.

5.2.6 El Paquete Java.Security

Este API de seguridad está incluido en Java API y provee dos API, uno para los usuarios de los algoritmos de seguridad y otro para implementadores o proveedores de estos algoritmos.

En los últimos 50 años los matemáticos y los científicos de la informática han desarrollado algoritmos que aseguran la integridad de los datos y que permiten hacer firmas digitales. El paquete java.security contiene implementaciones para muchos de estos algoritmos.

El API para usuarios está diseñado para que los distintos algoritmos criptográficos sean utilizados en una aplicación, sin tener que preocuparse por la manera en la que éstos han sido implementados. Lo único que se necesita saber es el nombre del algoritmo. Una compañía o algún programador puede añadir sus propias implementaciones de los algoritmos usando la interfaz Provider.

El API de seguridad de Java es extendido por el paquete JCE (*Java Cryptography Extension*) que añade interfaces de cifrado. Este paquete está separado del API principal porque utiliza código que no es exportable fuera de Estados Unidos y Canadá.

En general el API de seguridad incluye interfaces para hacer uso de identidades, para utilizar firmas digitales y para cifrado de datos.

5.2.7 Pruebas Efectuadas en una Empresa

Desafortunadamente, por razones de tiempo de los empleados y debido a la falta de un manual de funciones y procedimientos, se dificultó la colaboración para la realización de pruebas del prototipo Seguridad en la Universidad del Cauca, razón por la cual se efectuaron dichas pruebas en empresas externas, tales como la Fundación Mundo Mujer (en donde se presentó la aplicación para ilustrar la utilización de los algoritmos criptográficos como solución al problema de la seguridad en el transporte de documentos en formato digital) y Cable Unión Telecomunicaciones, en donde se hicieron pruebas de envío entre la sucursal ubicada en el Centro Comercial Ferrocarril Local 114 y la sucursal que atiende las solicitudes de nuevos clientes en la Ciudad de Cali.

Se mostrarán a continuación, los documentos utilizados en el proceso más común de trámite de documentos, el del manejo comercial, en donde se expone solo información de carácter general debido a las políticas de confidencialidad de la empresa, como se muestra en la Tabla No. 16 (cortesía de la empresa), en la cual se observan los documentos utilizados en la afiliación y posterior prestación del servicio de Internet a través de Fibra Óptica, según las necesidades de cada usuario y en los cuales se adjuntan los detalles de las operaciones técnicas y estándares de calidad del servicio.

A continuación se presentarán detalles de la prueba efectuada el día 13 de Septiembre del presente año, a las 5:00 p.m., en la empresa Cable Unión Telecomunicaciones, en la que se envió un formato de Previabilidad, tres formatos de cotizaciones y tres formatos de precios y se recibió un formato de orden de servicio; en este trámite de documentos, según comentarios del Ingeniero de Soporte, el envío se realizó a través de la red de datos de la empresa, en la cual se guardan los archivos en una carpeta compartida que se encuentra en un servidor en Cali, la información no se envía cifrada porque la seguridad se le confía al esquema de VLAN establecido entre los nodos de Popayán, Cali, Buenaventura, Palmira, Buga, Tulúa, Pereira, Manizales, Medellín, Rionegro, Envigado, entre otras, además de la dificultad que tiene intervenir físicamente una fibra óptica; además la seguridad hacia el exterior, radica en el manejo de servidores Proxy para los clientes y el uso de un firewall soportado en Linux y un filtro WEB para restringir el acceso a contenidos no permitidos; en cuanto a la seguridad locativa, se restringe la entrada a personal no autorizado, pero en caso de que un intruso logre entrar, éste puede acceder fácilmente a las carpetas compartidas, en las cuales está la información sin cifrar, también se puede vulnerar la seguridad desde algún punto de la red mal ubicado o escondido con el programa “olfateador” Ethercap, que según pruebas puede capturar el texto plano o documentos que circulen a través de dicha red, ya que el programa puede hacer un test mediante el modo promiscuo para tal fin; además se notó que lo que antes demoraba minutos entre ciudades, se puede realizar en segundos y con información cifrada.

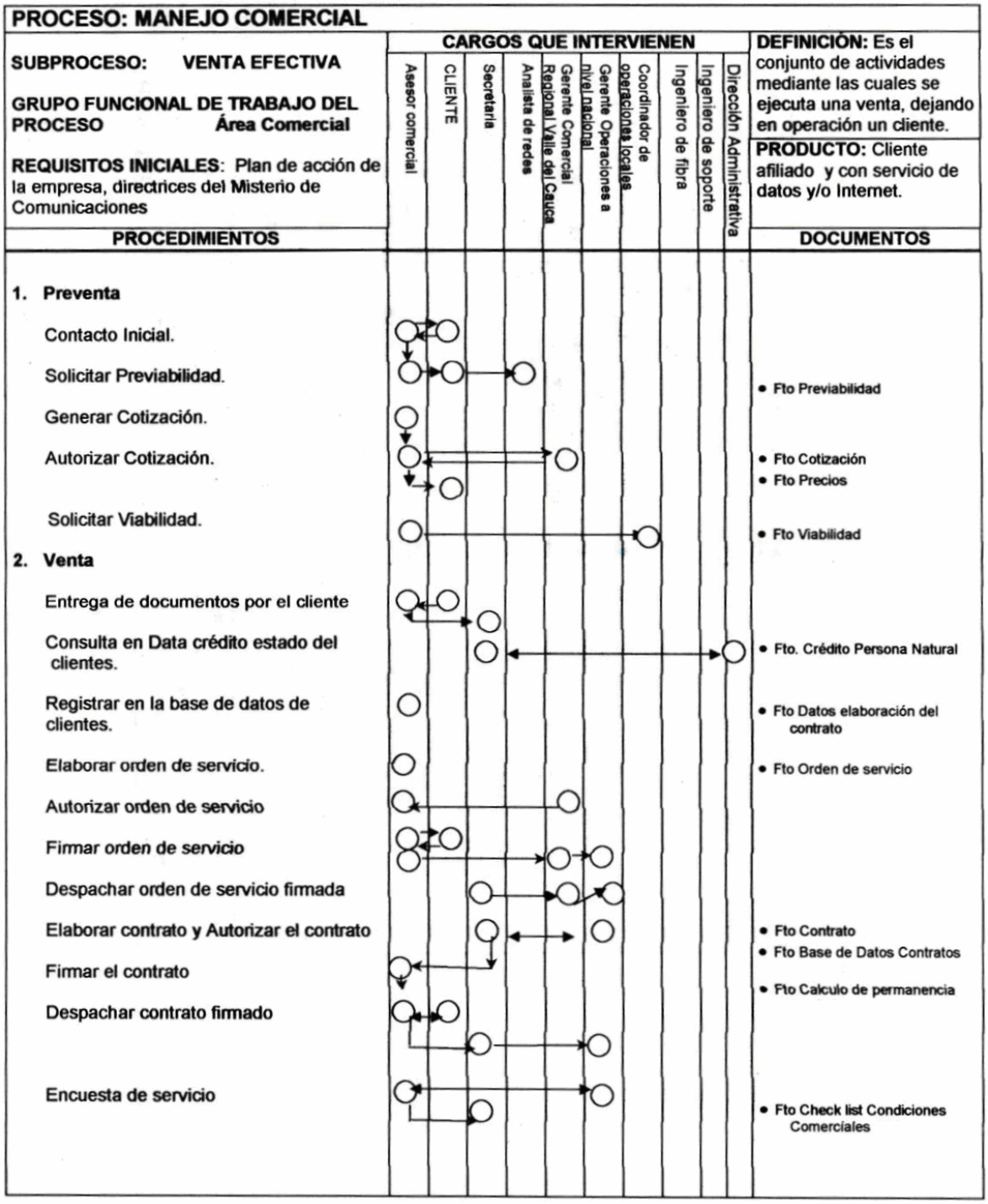

Tabla No. 19 Manual de Procesos de Cable Unión Telecomunicaciones

A continuación se dará un breve resumen del contenido de cada uno de lo documentos manejados entre los diferentes empleados solo para el proceso de manejo comercial:

- Formato de Previabilidad: Este documento se emite después de que el asesor ya ha establecido un contacto con el cliente y este se ha mostrado interesado en el

servicio, contiene los datos del cliente y sus requerimientos generales, además de posibles observaciones.

- Formato de Cotización: Contiene los detalles de los materiales, equipos, además de gastos extra generados por la instalación.

- Formato de Precios: Contiene información correspondiente al servicio que se va a prestar, se incluyen planes especiales.

- Formato de Viabilidad: Contiene información del cliente junto con el tipo de servicio, la capacidad del ancho de banda, cantidad de metros de fibra óptica a instalar, proporción de reuso, materiales a utilizar y la cantidad de los mismos, además de posibles observaciones.

- Formato Crédito Persona Natural: Es el documento mediante el cual la Dirección Administrativa de la empresa avala al cliente en los casos en los que se requieran garantías para pagos por crédito.

- Formato Datos de Elaboración del Contrato: Lo maneja cada asesor y lo guarda en la base de datos de la empresa para dejar constancia del proceso realizado.

- Formato Orden de Servicio: Es un formato emitido por el Asesor (según políticas de la empresa) para el cliente, detallando el servicio contratado por el cliente, la duración del contrato, la forma de pago y detalles de la instalación.

- Formato Contrato: Contiene los detalles de las obligaciones de ambas partes: prestadora del servicio y cliente, además de cláusulas adicionales con multas o sanciones por incumplimiento de una de ellas.

- Formato Base de Datos Contratos: Documento diligenciado por el asesor, una vez firmado el contrato para el control de pagos por comisiones.

- Formato Check List Condiciones Comerciales: Es una encuesta para medir el grado de satisfacción del cliente con el servicio prestado.

5.3 SOLUCIONES CRIPTOGRÁFICAS CON SOFTWARE LIBRE

5.3.1 XML Encryption

Existen situaciones en las que partes de un mismo documento necesitan un tratamiento diferente, como ocurren en documentos con secciones cuyo contenido puede ser visto por unos usuarios pero no por otros. En estos casos el cifrado juega un papel muy importante ya que es lo que va a confirmar la integridad del texto.

XML Encryption provee seguridad “extremo a extremo” a las aplicaciones que requieren intercambio de datos estructurados. XML por si misma, es la tecnología más popular para el estructurado de datos, y además el cifrado basado en XML es la forma natural de manejar requerimientos complejos de seguridad en aplicaciones que intercambian datos.

La función principal de XML Encryption es asegurar la confidencialidad de partes de documentos XML, a través del cifrado parcial del documento.[45]

XML Encryption es el resultado de un esfuerzo exclusivo de la W3C. Ésta especificación no pretende reemplazar o sustituir a SSL/TLS, más que esto, provee mecanismos para cubrir requerimientos de seguridad que no son tratados por SSL, como se mencionó en las limitaciones del SSL.

Con XML Encryption, cada parte de una comunicación (emisor y receptor) puede mantener un estado seguro o inseguro con la otra parte. Ambos, datos seguros y no seguros pueden ser intercambiados en el mismo documento. Por ejemplo, En una aplicación de Chat segura, entre un número de salas de Chat con gente en cada salón, los archivos XML-cifrados pueden ser intercambiados confiadamente entre los compañeros del Chat, garantizando que los datos dirigidos a un salón no serán visibles a los otros salones.[45]

WSDL (Web Services Definition Language) y SOAP (Simple Object Access Protocol) por estar gramaticalmente basados en XML pueden usar XML Encryption para proveer comunicaciones seguras.

5.3.2 TrueCrypt (TCCL)

En respuesta a nuestra solicitud de ayuda, el Dr. Jorge Ramió Aguirre, experto en el área de los algoritmos criptográficos, recomienda para el uso de algoritmos simétricos el software TrueCryp 4.2a, el cual es un sistema multiplataforma de unidades cifradas virtuales que trabaja de forma transparente al usuario, su seguridad como sistema radica en la creación de dichas unidades virtuales que permanecen ocultas y que no se pueden encontrar si no se conocen; utiliza como algoritmos de cifrado asimétrico: AES, Blowfish, CAST5, Serpent y 3DES.

Lo que hace TrueCrypt es crear un "volumen secreto", que consiste en un archivo que puede tener cualquier nombre y que TrueCrypt puede montar como una unidad de disco, con su identificación respectiva, según el sistema operativo utilizado. El contenido de ese archivo tiene su propio filesystem y todo lo necesario para operar como una unidad común de almacenamiento. Lo que se grabe en esa unidad virtual se cifra usando los algoritmos y la potencia de cifrado que el usuario elija. Cuando se "monta" la unidad a través de TrueCrypt, se pide la contraseña que el usuario escogió al momento de crear este archivo secreto.

5.3.2.1 Keyfiles

En teoría, la contraseña sería el punto débil del sistema, ya que los usuarios rara vez escogen una contraseña compleja y se contentan con palabras de asociación familiar. Debido a esto, además de la contraseña está la opción de usar un keyfile, es decir seleccionar uno de los miles de archivos que hay en el equipo como segunda contraseña. Si archivo y password coinciden, el volumen se monta y queda disponible como una unidad de disco estándar. Es importante tener presente que este archivo debe ser inmutable. Si su contenido cambia, TrueCrypt no lo reconocerá como el "archivo llave" correcto y negará el acceso al volumen cifrado. Es recomendable, por ello, usar algún archivo que eventualmente se pueda recuperar de otra fuente, como Internet. Una imagen

o un archivo de música serían muy aptos. También se le puede pedir a TrueCrypt que genere un Keyfile ad-hoc, pero ese tiene el inconveniente de que es único. Si se malogra y no hay copia, no será posible recuperar el archivo cifrado.
Opciones de almacenamiento

Es posible también grabar un respaldo del archivo (volumen) cifrado en otra parte, por ejemplo en un CD y dejarlo en algún lugar seguro. Aunque se destruya el computador original o sea robado, el archivo del disco de respaldo se puede montar perfectamente en otro equipo que tenga TrueCrypt y el archivo secreto que sirve como Keyfile, si es que se escogió esta modalidad. También es posible crear un "volumen movible" con todo lo necesario en el mismo disco de respaldo para no perder tiempo en caso de emergencia.

El tamaño del archivo cifrado es definible por el usuario, así que si se crea uno de 700 Mb, se puede almacenar sin problemas en un CD. En último caso se puede seccionar con algún programa archivador si es necesario.

El volumen una vez montado, puede desmontarse automáticamente según parámetros que elija el usuario. Por defecto se desmonta al cerrar la sesión de trabajo (logout) en el equipo, pero es posible que se desmonte también al activarse el salvapantallas, por ejemplo, si se requiere una alta seguridad.

Este volumen puede pasar totalmente inadvertido, ya que puede tener el nombre de un archivo común y que no llame la atención, por ejemplo "discurso.avi" o "dump1.dat" o lo que se estime adecuado.

5.3.3 GNU Privacy Guard (GPL)

También conocido como GPG es la versión libre del conocido PGP de Phil Zimmermann. Es libre para uso no comercial y para uso comercial. Además se dispone del código fuente con lo cual es posible auditar el código y comprobar si tiene fallos de seguridad y/o funcionalidades no documentadas. Es un estándar de facto para correo electrónico y contiene una implementación libre de OpenPGP (RFC #2440), usa RSA, ElGamal y DSA como algoritmos para cifrado asimétrico; utiliza DES, CAST5, Blowfish, AES y Twofish para cifrado simétrico y utiliza MD5, SHA1, RIPEMD160 y para generar las funciones hash. [46]

5.3.4 OpenSSH (BSD)

Es una suite de seguridad con una implementación libre de Secure Shell (Parte del proyecto OpenBSD) y es también un estándar en administración remota que consta de las siguientes herramientas: ssh, scp, sshd, ssh-keygen, ssh-agent, entre otras, usa como algoritmos para cifrado asimétrico RSA y DSA y para cifrado simétrico TDES, Blowfish, AES, Arcfour (parecido al RC4) [46]

5.3.5 OpenSSL (Apache)

Es una suite de seguridad con una implementación libre de TLS y SSL (Basado en SSLeay) cuya principal característica es proporciona un canal de comunicación seguro a través de una red; usa como algoritmos para cifrado asimétrico RSA, DH y DSA, para

cifrado simétrico RC2, RC4, (IDEA), DES, 3DES, AES y Camellia y para generar las funciones hash MD2, MD4, MD5 y SHA-1. [46]

5.3.6 P.A.M. (Pluggable Authentication Modules)

P.A.M. es una librería para Linux que permite al administrador de un sistema elegir que métodos de autenticación va a utilizar el sistema. Esto permite adaptar la seguridad del sistema según las necesidades. Utilizando esta librería se pueden personalizar los métodos que utilizaran las aplicaciones para autenticar a los usuarios, siempre y cuando dispongamos del código fuente de las aplicaciones. Con esta librería no es necesario realizar muchos cambios en el código fuente de las aplicaciones. [46]

5.3.7 C.F.S. (Cryptographic File System)

Esta utilidad para Linux permite la creación de archivos y directorios cifrados dentro del sistema de archivos que estamos utilizando. Para poder acceder a estos archivos se necesitará una contraseña que habrá sido establecida cuando se creó el directorio. Este software esta basado en D.E.S. y está sometido a las leyes de exportación de los Estados Unidos.[46]

5.3.8 T.C.F.S (Transparent Cryptographic File System)

Desarrollado por la Universidad de Salerno específicamente para Linux. Su propósito es proporcionar seguridad en sistemas de archivos NFS. El descifrado se realiza en el lado del cliente. Este software funciona a nivel de núcleo, no a nivel de usuario, con lo cual se gana en seguridad. Esta basado en D.E.S. y para hacer funcionar las máquinas clientes necesitan una versión compilada del núcleo para soporte de T.C.F.S. [46]

Conclusiones y Recomendaciones

Este capítulo contiene las consideraciones finales del proyecto indicando en términos generales el desempeño de los algoritmos criptográficos empleados en una herramienta software para ilustrar el análisis previo. Se plantea además algunas sugerencias en cuanto a trabajos futuros orientados al mejoramiento de la aplicación, así como en lo referente al campo de investigación de la criptografía y sus aplicaciones.

- La seguridad computacional en sus diferentes aspectos (autenticación, autorización, confidencialidad e integridad) es de vital importancia para proteger la información, en especial mediante las aplicaciones de gestión documental, sin dejar de lado la fortaleza de los diferentes algoritmos que ayudan a garantizar un canal de transferencia seguro.

- Antes de realizar una aplicación criptográfica conviene elegir buenos algoritmos, lo que significa que utiliza pocos recursos, siendo usualmente los más importantes: el tiempo que lleva ejecutarse y la cantidad de espacio en memoria que requiera. No se debe confiar en la creciente potencia de las máquinas y el abaratamiento de las mismas como remedio de todos los problemas que puedan aparecer. Además se debe consultar siempre fuentes fidedignas con resultados concretos a fin de lograr establecer la seguridad y complejidad de los diferentes algoritmos criptográficos. Sería de gran utilidad que en el futuro se complementaran las materias de programación con fundamentos matemáticos necesarios como la teoría de la complejidad y conceptos de matemática discreta junto con teoría de números, para este campo de la criptografía que está en pleno auge.

- El análisis de algoritmos ayuda a fortalecer los conocimientos en la teoría de la computación y en la solución de los problemas informáticos. Entrar en contacto con algunas técnicas fundamentales de diseño y análisis de algoritmos, incentiva la capacidad para el razonamiento crítico y lógico-matemático.

- Con el desarrollo del presente proyecto se ha llevado a cabo la adquisición de nuevos conceptos tanto matemáticos como prácticos de la implementación de los diferentes tipos de algoritmos, además de ganar experiencia en el aprendizaje del manejo de las diferentes APIs de criptografía de Java, con lo cual se refuerzan los criterios de evaluación escogidos.

- Es de resaltar que en el presente trabajo no se hicieron pruebas Criptoanalíticas propias, debido a la falta de experiencia y conocimientos previos al respecto, pero gracias a los trabajos de Doctores y a la comunidad criptográfica mundial se logró extractar importante documentación de pruebas al respecto, además de los contactos establecidos, los cuales aportaron en gran manera al desarrollo del mismo.

- Finalmente y considerando que en la Universidad del Cauca solo hasta ahora se empieza a difundir la criptografía, resta incentivar el desarrollo de nuevos estudios referentes a la aplicación de la misma, pues son muchas las áreas donde puede representar importantes logros, una de ellas es la referente al manejo de documentación interna, que podría aprovechar mejor el potencial de las redes telemáticas para mejorar la eficiencia de los procesos, además se pueden desarrollar pluggins criptográficos para el servicio de correo electrónico en los cuales cada estudiante pueda manejar certificados y firmas digitales propios junto con claves privadas y públicas.

- Como propuestas futuras vale la pena considerar proyectos que involucren tanto llaves software como hardware, ya que éstas aumentan considerablemente la seguridad, también se podría considerar el estudio e implementación de herramientas criptoanalíticas para medir la fortaleza de los algoritmos, además se podría pensar en estudiar o analizar los protocolos criptográficos mediante herramientas de simulación o aplicándolos a problemas específicos de la Universidad.

BIBLIOGRAFIA

[1] S. Felici Castell. (2003). *"Seguridad Distribuida en la Red y Centralizada en los Sistemas*". [En Línea]. Disponible: http://informatica.uv.es/guia/asignatu/R/.

[2] D. Brink, W. Duane, C. Joseph, A. Nash. *"PKI Infraestructura de Claves Públicas"*. Estados Unidos. McGraw Hill. (2003). PP 13-259.

[3] D. De la Guía Martinez, A. Fuster Sabater, L. Hernández Encinas, F. Montoya Vitini, J. Muñoz Masqué. *"Técnicas Criptográficas de Protección de Datos"*. AlfaOmega Ra-ma. (2003). PP 243-351.

[4] B. M. García Lobo. España. (2003). *"Algoritmos Criptográficos"* .

[5] M. J. Lucena López. Madrid, España. (2006, 05). *"Criptografía y Seguridad en Computadores".*

[6] B. Schneier. Artículo IEEE. (2000,09). *"Cryptographic Design Vulnerabilities".*

[7] J. Herrera Joancomartí, J. García Alfaro, X. Perramón Tornil. (2003). *"Aspectos Avanzados de Seguridad en Redes".* PP 5-42.

[8] I. Alshanetsky. "*Vulnerabilidades del PHP"*. (2006,06) [En Línea]. Disponible: http://es.phpsolmag.org.

[9] M. Camps. España. (2000,05). *"La Protección de la Información por Métodos Criptográficos"* .

[10] J. F. Flores Barahona. Escuela Superior Politécnica del Litoral, Ecuador. (2000) *"Diseño de un sistema criptográfico de Seguridad para las comunicaciones navales"*. PP 53-100.

[11] J. Gutiérrez. Universidad de Cantabria. España. (2000) *"Criptografía (Matemáticas e Internet)" .*

[12] J. J. Angel Angel. España. (2005). *"AES – Advanced Encryption Standard para Principiantes"* .

[13] J. Kelsey. AES4Horst Görtz Institute. NIST. (2004) *"NIST and AES in 2004".*

[14] B. Cohen, B. Laurie. (2001). NIST *"AES-hash"*.

[15] A. Biryukov. (2004). AES4 Horst Görtz Institute. *"The Boomerang attack on 5- and 6-round AES".*

[16] I. Toli, A. Zanoni. (2004). AES4 Horst Görtz Institute *"An algebraic interpretation of AES-128"*.

[17] J. Ramió Aguirre. Universidad Politécnica de Madrid, España. (2006,03). *"Libro Electrónico de Seguridad Informática y Criptografía Versión 4.1"* .

[18] J. J. Ángel Ángel. España. (2003). *"Introducción a los Criptosistemas con Curvas Elípticas"*.

[19] Grupo de Trabajo Pentazip. (2006, 03). *"Descripción de los Algoritmos de Pentazip".* [En Línea].Disponible: http://www.pentazip.com/pw_es/PentaSuite_Security.htm.

[20] Grupo de Trabajo Infodoc. España. (2006,03). *"Siam IRS"*. [En Línea]. Disponible: http://www.infodoc.es/software/software.asp.

[21] Grupo de Trabajo Neodoc. España. (2006). *"Software de Gestión Documental".* [En Línea]. Disponible: http://www.neodoc.es/productos/gestion_documental.

[22] R. Baeza-Yates. Depto. Ciencias de la Computación, Universidad de Chile (1991). *"Algoritmia".*

[23] S. Baase; A. Van Gelder. Addison-Wesley, (2000). *"Computer Algorithms. Introduction to Design and Analysis".*

[24] A. M. Mancilla Herrera. España. (2002). *"Análisis de un Cifrador Simétrico".*

[25] J. A. Mañas. España. (1997). *"Análisis de Algoritmos: Complejidad".* [En Línea]. Disponible: http://www.lab.dit.upm.es/~lprg/material/apuntes/o/index.html.

[26] N. Koblitz. Estados Unidos. (2002). "A Course in Number Theory and Cryptograhy"

[27] J. Ramió Aguirre. Departamento de Publicaciones de la Escuela Universitaria de Informática de la Universidad Politécnica de Madrid, España. (2000). *"Aplicaciones Criptográficas".*

[28] M. Blaze. AT&T Laboratories. (2005). *"Efficient Symmetric - Key Ciphers Based on an NP- Complete Subproblem".*

[29] A. Menezes. Estados Unidos. (2000). *"Cripto CRC Handbook Of Applied Cryptography (Algorithms, Mathematics, Science, Source Code)".*

[30] J. J. Angel Angel. España. (2005). *"AES – Advanced Encryption Standard Versión 2005 para Principiantes".*

[31] B. Gladman. Roma, Italia (1999,03) *"Implementtaion Experience with AES Candidates Algorithms. Proceedings of the Second Advanced Encryption Standard Candidate Conference National Institute of Standards and Technology (NIST)".*

[32] B. Schneier, J. Kelsey, D. Whitingz, D. Wagner. Estados Unidos. (1999). *"Performance Comparison of the AES Submissions"* .

[33] Álvarez Marañón, Gonzalo. España. (2000) Revista Criptonomicón nº34. *"El nuevo estándar de cifrado".*

[34] J. Nechvatal, E. Barker, L. Bassham, W. Burr, M. Dworkin, J. Foti, E. Roback. Estados Unidos. (2000). "*Report on the Development of the Advanced Encryption Standard (AES)".*

[35] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, D. Whiting,. Estados Unidos. (2001). *"Improved Cryptoanalysis of Rijndael".*

[36] N. Ferguson, J. Kelsey, B. Schneier, D. Wagner, D. Whiting, C. Hall. Estados Unidos. (2001). *"New Results on the Twofish Encryption Algorithm".*

[37] M. Lamagna Walter. Universidad Abierta Interamericana, Argentina. (2005). *"Trabajo de investigación. Algorítmo Criptográfico CAST".*

[38] C. Burwick, D. Coppersmith, E. D'Avignon, R. Gennaro, S. Halevi, C. Jutla, S. M. Matyas Jr., L. O'Connor, M. Peyravian, D. Safford, N. Zunic. IBM Corporation, Esatdos Unidos. (1999,08). *"MARS - a candidate cipher for AES"*

[39] J. Sánchez Arriazu. España. (1999). *"Descripción del Algoritmo DES (Data Encryption Standar)".*

[40] B. Schneier. Estados Unidos. (2006). *"Algoritmos Criptográficos".* [En Línea]. Disponible: http://www.schneier.com/blowfish-speed.html.

[41] M. A. Gallardo Ortiz. España. (2006). *"Criptosistema RSA"*. [En Línea]. Disponible: http://www.cita.es/textos/protesis.htm.

[42] H. D. Scolnik, J. P. Hecht. México. (2004,09) *"Impacto de Recientes Ataques de Colisiones Contra Funciones de Hashing de Uso Corriente".*

[43] B. Preneel, A. Biryukov, E. Oswald, B. Van Rompay, L. Granboulan, E. Dottax, S. Murphy, A. Dent, J. White, M. Dichtl, S. Pyka, M. Schafheutle, P. E. Biham, E. Barkan, O. Dunkelman, J.-J. Quisquater, M. Ciet, F. Sica, L. Knudsen, M. Parker, H. Raddum. Estados Unidos. (2003). "NESSIE Security Report".

[44] National Institute of Standards and Technology (NIST). Esatdos Unidos. (2004,02). *"Federal Information Processing Standards (FIPS), Publication 180-2, Secure Hash Standard (SHS)".*

[45] D. Cerón Imbachí, D. I. Chamorro Salas. Universidad del Cauca, Colombia. (2005). *"Plataforma de acceso seguro a servicios de 2.5 y 3G".* PP 28-38, 46-55.

[46] J. A. De Bustos Pérez. España. (2002). "*Criptografía*".

GLOSARIO

A

AES: Advanced Encryption Standard.

B

BC: Bouncy Castle

C

CA: Certificate Authority. Autoridad Certificadora.

CBC: Cipher Book Chaining Mode

CDMA: Acceso Múltiple por División de Código

CERT-CC : Computer Emergency Response Team Coordination Center.

CFB : Cipher-Feedback Mode

CRL: Certificate Revocation List. Lista de revocación de certificados

CSRF : Cross-Site Request Forgery

CTR: Modo Counter

D

DEA : Algoritmo de Cifrado de Datos

DES: Data Encryption Standar

E

ECB : Electronic Codebook

F

FTP: File Transfer Protocol. Protocolo de transferencia de archivos.

G

GSM: Global System for Mobile.

H

HE: Home Environment

HPC: Handheld PC

HTML: acrónimo de HyperText Markup Language, lenguaje de marcas de hipertexto

HW: Hardware

I

IDEA: International Data Encryption Algorithm. Algoritmo Internacional de cifrado de datos.

IEEE: Instituto de Ingenieros Electrónicos y Eléctricos

IETF: Internet Engineering Task Force. Es una gran comunidad abierta de desarrolladores de redes, operadores, vendedores e investigadores relacionados con la evolución de la arquitectura de Internet y su operación.

ISO: Organización Internacional de Estándares

ITAR : Normas que regulan el tráfico de armamento a nivel internacional en los Estados Unidos

J

JCE: Java Cryptographic Extension

JCP: Java Community Proccess, Creado por un grupo experto que abarca la mayor parte de fabricantes de dispositivos móviles, operadores inalámbricos y vendedores de software móvil.

JDK: Java Development Kit

JSR 177: Java Specification Request

J2ME: Java 2 Micro Edition

J2SE: Java Standard Edition v2

K

L

M

MD5: Algoritmo Message Digest 5

MMS: Multimedia Messaging Service

N

NIST: Instituto Nacional de Estándares y Tecnología

O

OFB : Output Feedback

P

PDA: Personal Digital Assistent

PGP: Pretty Good Privacy

PHP: Página de Hipertexto

PKI: Public Key Infrastructure

PPTP: Protocolo de Entunelamiento Punto a Punto de Microsoft

Q

R

RA: Registration Authority. Autoridades Registradoras.

Record Protocol: protocolo de registro.

RC5: "RC" simboliza "Ron's Code" o "Rivest Cipher". RC es un bloque de cifrado diseñado por Ron Rivest .

RIJNDAEL: Algoritmo cifrador de bloque propuesto por los belgas Vincent Rijmen y Joan Daemen.

RSA: Algoritmo de Rivest, Shamir y Adleman

S

SAML: Secure Assertion Markup Language

SATSA: The Security and Trust Services APIs

SET: Secure Electronic Transactions.

SGSN: Nodo de servicio GPRS

SIM: subscriber identity module. Módulo de identidad del suscriptor

SkipJack: Algoritmo diseñado para alcanzar alto rendimiento de los datos usados en sistemas de comunicación de tiempo real.

SMS: Short Messaging Service

SMTP: Simple Mail Transfer Protocol

SN: Serving Network

SOAP: Simple Object Access Protocol

SSL: Secure Socket Layer

T

U

UDP: User Datagram Protocol

URL: acrónimo de Universal Resource Locator (localizador universal de recursos)

V

W

WSDL: Web Services Definition Language

X

XKMS: XML Key Management Specification

XSS : Cross-Site Scripting