UNIVERSIDAD NACIONAL DE INGENIERIA

FACULTAD DE INGENIERIA INDUSTRIAL Y DE SISTEMAS

TECNOLOGÍAS PARA LA SEGURIDAD DIGITAL : PRIVACIDAD Y SEGURIDAD EN LA TRANSMISIÓN DE DATOS DE UN PROCESO ELECTORAL

INFORME DE SUFICIENCIA

PARA OPTAR EL TITULO PROFESIONAL DE

INGENIERO DE SISTEMAS

DINO MARCELO VEGA PAUL

LIMA – PERU
2002

Titulación por Actualización de Conocimientos

Segundo Programa

Ingeniería de Sistemas

Informe de Suficiencia

Tecnologías para la Seguridad Digital: Privacidad y Seguridad en la Transmisión de Datos de un Proceso Electoral.

AUTOR: DINO MARCELO VEGA PAUL

E-MAIL : dimavex@hotmail.com

TELEF : 4825860 / 8558469

Dedicatoria

A mis padres, Víctor y Consuelo que merecen esta muestra de la culminación de su esfuerzo en mí.

A mi esposa Patricia, que es el soporte y apoyo para la culminación de este proyecto.

A mis hijos Dino y Patricia, les digo que su padre les da el orgullo de la tarea postergada, pero cumplida.

Agradecimiento

Mi profundo agradecimiento a la Universidad Nacional de Ingeniería, a la Facultad de Ingeniería Industrial y de Sistemas, a mis padres, a mi esposa y mis hijos, por animarme y regalarme su tiempo.

INDICE

DESCRIPTORES TEMÁTICOS

Algoritmo Criptográfico Asimétrico.
Algoritmo Criptográfico Simétrico.
Autoridad de Certificación.
Certificado Digital.
Descifrar.
Digital Signature Standard (DSS).
Encriptación.
Estándar de Encriptación Avanzada (AES).
Estándar Encriptación de Datos (DES).
Firma Digital.
Función Hash.
Huella digital.
Message Digest 5 (MD5).
Rijndael (el nuevo AES).
Rivest-Shamir-Adleman (RSA).
Secure Hash Algorithm (SHA-1).
Sobre Electrónico.
Transmisión de Datos Seguro.
Triple DES.

RESUMEN EJECUTIVO

En el Proceso Electoral de Elecciones Presidenciales del año 2000, la Organización de Estados Americanos (OEA), realizó una evaluación técnica a la Oficina Descentralizada de Procesos Electorales (ONPE).

La OEA, como organismo veedor del proceso electoral, representado por el Sr. Stein y con la colaboración de consultores extranjeros, presentó un informe sobre las deficiencias encontradas durante el desarrollo del Proceso Electoral del 09 de abril del 2000.

Es necesario indicar, que la OEA, condicionaba su presencia como observador de la Segunda Vuelta Presidencial, si se daba solución integral a todas las deficiencias encontradas.

Dentro de las deficiencias encontradas a la Oficina Nacional de Procesos Electorales (ONPE), fue la vinculada especialmente a la fluidez de ciertos procedimientos informáticos y específicamente a la **falta de encriptación en la transmisión de los datos.**

Es decir, las Oficinas Descentralizadas de Procesos Electorales (ODPE's) que se encuentran instalados en todos los departamentos del Perú, a través de sus centros de cómputo (**emisor**), transmiten datos al centro de cómputo de la sede central de la ONPE (**receptor**).

Estos datos son enviados en texto claro (legible), por lo que no existe garantía de confidencialidad en las comunicaciones.

Después de realizar un análisis de riesgo, que nos permitió conocer mejor el sistema que deseábamos proteger, identificamos otros dos problemas adicionales, que hacían vulnerable la transmisión de datos:

- La imposibilidad de verificar si los datos transmitidos, han sido alterados en el trayecto, de forma fraudulenta.
- La imposibilidad de verificar que las partes que intervienen en el proceso de transmisión de datos, representen quienes dicen ser, es decir que no existen garantías de saber si están siendo suplantados por otra entidad, con la finalidad de introducir datos falso en la red.

Identificados los problemas, se desarrolla el protocolo TDS (Transmisión de Datos Seguro), como **solución a estos factores críticos en la transmisión de datos en un Proceso Electoral.**

Por lo tanto, los procesos definidos en TDS, dentro de los límites establecidos son:

- Garantizar la confidencialidad de la información sensible (número de mesa, cantidad de votos válidos, cantidad de votos en blanco, cantidad de votos nulos.).
- Preservar la integridad de la información transmitida.
- Proporcionar la autentificación necesaria entre las ODPE's (emisores) y la ONPE (receptor).
- Definir las técnicas y los algoritmos criptográficos necesarios, para soportar los servicios anteriores.

Aunque estos aspectos son importantes en cualquier red de transmisión de datos, su vulnerabilidad causa una preocupación mucho mayor, cuando se relaciona con un Proceso Electoral.

Podemos concluir que cada técnica criptográfica empleada en el protocolo TDS, permiten la implementación de una función determinada, ofreciendo en redes abiertas, mecanismos de protección y seguridad muy elevados, durante la transmisión de datos en procesos electorales.

La Organización de Estados Americanos después de la evaluación realizada a la ONPE, sobre la **falta de encriptación en la transmisión de datos**, aprobó las medidas correctivas realizadas sobre este punto.

INTRODUCCIÓN

El presente informe de suficiencia se presenta en el marco del Segundo Programa de Titulación por "Actualización de Conocimientos" organizado por la Facultad de Ingeniería Industrial y de Sistemas y el **objetivo del presente trabajo es dar solución** a unos de los problemas encontrados por la Organización de Estados Americanos (OEA), al centro de cómputo de la Oficina Nacional de Procesos Electorales (ONPE) durante el Proceso Electoral Presidenciales 2000. El problema específicamente es la **falta de encriptación en la transmisión de datos** entre los centros de cómputo de sus Oficinas Descentralizadas de Procesos Electorales (ODPE's) y su sede Central.

Las Oficinas Descentralizadas de Procesos Electorales, son órganos de ejecución temporal de la ONPE y su responsabilidad es llevar con eficiencia las acciones necesaria para el desarrollo del proceso electoral dentro del ámbito de una circunscripción electoral.

Uno de los requisitos para el informe de suficiencia es que el objeto del trabajo se relacione con algún curso que esté contenido dentro del programa de titulación, y en este caso concreto, el tema se relaciona directamente con el curso de "Negocios Electrónicos", en lo que se refiere a esquemas de seguridad y aspectos tecnológicos para la seguridad digital.

Los logros obtenidos fueron:

- Aprobación y levantamiento de las deficiencias encontradas al centro de cómputo de la ONPE en este punto.

- Reducción del riego de amenazas externas.
- Garantizar la confidencialidad en la transmisión de resultados electorales entre las ODPE's y su sede Central.
- Preservar la integridad de la información, es decir que el mensaje no ha sido manipulado.
- Garantizar la validación mutua entre las ODPE's y la Sede central ONPE.
- Mejora la credibilidad de ONPE, como organismo técnico para ejecutar Procesos electorales.

Las limitaciones fueron:

- No existen políticas de seguridad en lo referente a flujo de información.
- Poca información sobre el uso de la criptografía en el Perú.
- Restricciones y prohibiciones por parte de EEUU (ITAR), para vender software criptográfico fuerte a otros países. (con llaves de 128 bits a 256 bits tipo militar) (Prohibido a países como Cuba y Nicaragua) (Restringido a países de América Latina, sólo pueden importar software débil con longitud de clave de 40, 56 y 64 bits).
- El software criptográfico que se vende para América Latina, viene con Puertas traseras (backdoor), que le permite a los EEUU, descifrar cualquier mensaje, cuando ellos crean conveniente, por su Seguridad.
- En el Perú, existe la ley 27269, ley de Firmas y Certificados digitales, que tiene por objeto regular la utilización de la firma electrónica, pero el problema es que no está reglamentado.

I. ANTECEDENTES

1.1 Diagnóstico estratégico:

El Sistema Electoral Peruano dentro de un proceso electoral, tiene como finalidad, la de asegurar que las votaciones y escrutinios traduzcan la expresión auténtica, libre y espontánea de los ciudadanos y sean el reflejo **exacto y oportuno** de la voluntad del elector, expresada en las urnas por votación directa y secreta.

El Sistema Electoral está conformado por tres instituciones las cuales son:

El Jurado Nacional de Elecciones (JNE), La Oficina Nacional de Procesos Electorales (ONPE) y el Registro Nacional de Identificación y Estado Civil (RENIEC).

LA ONPE tiene a su cargo la planificación, organización y ejecución de los procesos electorales.

La **Visión** de la ONPE, es organizar procesos electorales honestos, justos y transparentes, que contribuya a la consolidación institucional de la democracia peruana, convirtiéndose en una institución modelo y referente en el ámbito de la región por su eficiencia, credibilidad y confianza en la gestión pública, al servicio de los ciudadanos y organizaciones políticas.

La **Misión** de la ONPE es ser un organismo autónomo encargado de planear, organizar y ejecutar procesos electorales y de consulta popular de manera eficiente, que **garantice** la participación, la libertad

de sufragio y el respeto a la voluntad de los electores, contando para ello con profesionales altamente calificados y comprometidos con los valores y principios democráticos de elecciones limpias y transparentes, que produzcan resultados incuestionables.

Dentro del análisis de ambiente, los factores críticos de éxito son:

Organizar procesos Electorales.

La transparencia e imparcialidad con la que se debe desarrollar un proceso electoral.

La Credibilidad y Confiabilidad de una institución ante la ciudadanía, para ser calificada como ejecutora de un proceso electoral. Esto se sustenta sobre la base que en todo proceso electoral se eligen autoridades que representan a las instituciones de gobierno del país. Si éstas nacen de elecciones no creíbles, no son respetadas, produciéndose un resquebrajamiento de la institucionalidad, por consiguiente se debilita la democracia y esto afecta el crecimiento económico y social de un país.

Oportunidad, seguridad, exactitud en la entrega de resultados del proceso electoral, por lo que **no** puede haber demora en la entrega de resultados, ya que la sociedad se sensibiliza y se presta a muchas especulaciones, pudiendo traerse a bajo, todo un proceso electoral.

Identificados los factores críticos de éxito, pasaremos a identificar las potencialidades, debilidades, oportunidades y amenazas para la ONPE.

1.1.1 Fortalezas y Debilidades

Dentro de las potencialidades tenemos:

Experiencia de la institución, en la realización de anteriores procesos electorales.

La Especialización del personal, en el desarrollo y ejecución de procesos electorales. Esto se sustenta en que el mundo moderno tiende cada vez más a la especialización

en todos los campos. Todas las áreas de la actividad humana tienden a una mayor complejidad de tal manera que la única vía para ser eficiente es la especialización.

Tecnología de Punta, en lo referente a equipos de cómputo de gran capacidad de almacenamiento y alta disponibilidad, también cuentan con avanzados equipos de impresión que permiten soportar las más altas exigencias técnicas.

Organismo Autónomo que forma parte de la estructura del estado, y goza de atribuciones en materia técnica, administrativa, económica y financiera, siendo la autoridad máxima en la organización y ejecución de todos los procesos electorales.

Dentro de las debilidades tenemos:

La poca Credibilidad en la ONPE, ocasiona que se gaste adicionales recursos económicos, para dotar con técnicas de seguridad, las actas electorales, que permitan recuperar la confianza de la población, en la institución.

Falta de transparencia del software informático, debiendo hacer público los programas fuentes y permitir auditorias del software.

Falta delimitar claramente: funciones, atribuciones y competencias con el Jurado Nacional de Elecciones, trayendo como consecuencia, la duplicación de funciones. El JNE distrae recursos y energías, contratando servicios de terceros para que hagan lo que hace la ONPE o el RENIEC, en rubros tales como publicidad, educación electoral y hasta del Padrón en nombre de la fiscalización.

1.1.2 Oportunidades y riesgos (Amenazas)

Dentro de las oportunidades tenemos a los observadores:

Organismos Internacionales (OEA, Centro Carter) como veedores del proceso electoral, es una oportunidad de la ONPE, para poder revertir la mala imagen, trabajando y coordinando con ellos, haciéndolos partícipe y que no sean simples espectadores.

Medios de Comunicación, coordinación con la prensa hablada y escrita, para que colaboren en el cambio de imagen y den confianza a la población.

Dentro de los riegos tenemos la:

Intromisión de Otras Instituciones de gobierno dentro de la ONPE, es decir pérdida de autonomía en sus funciones.

Clima Político de desconfianza durante el proceso electoral, debido a que los partidos políticos se sensibilizan y se produce la desconfianza.

1.2 Diagnóstico funcional

La Oficina Nacional de Procesos Electorales, tiene como función esencial, velar por la obtención de resultados, manifestada a través de los procesos electorales, del referéndum y de otros tipos de consulta popular.

1.2.1 Productos

La ONPE ofrece los siguientes productos:

- Elecciones Generales Presidenciales y de Congresistas.
- Elecciones Municipales
- Elecciones Regionales
- Referéndum
- Consulta Popular de Revocatoria del Mandato de Autoridades
- Elecciones de jueces de Paz a nivel Nacional.

1.2.2 Clientes

El cliente principal es la:

- La ciudadanía, es decir la población electoral del Perú que a la fecha, es aproximadamente de 15 millones de electores.

Pero últimamente han aparecido nuevos clientes como:

- Los Colegios Profesionales.
- Los Partidos Políticos.
- Los Sindicatos.

1.2.3 Proveedores

Los principales proveedores tenemos:

- Telefónica del Perú
- Microsoft
- Oracle
- IBM

1.2.4 Procesos

Los principales procesos son:

Proceso de educación y Capacitación electoral.

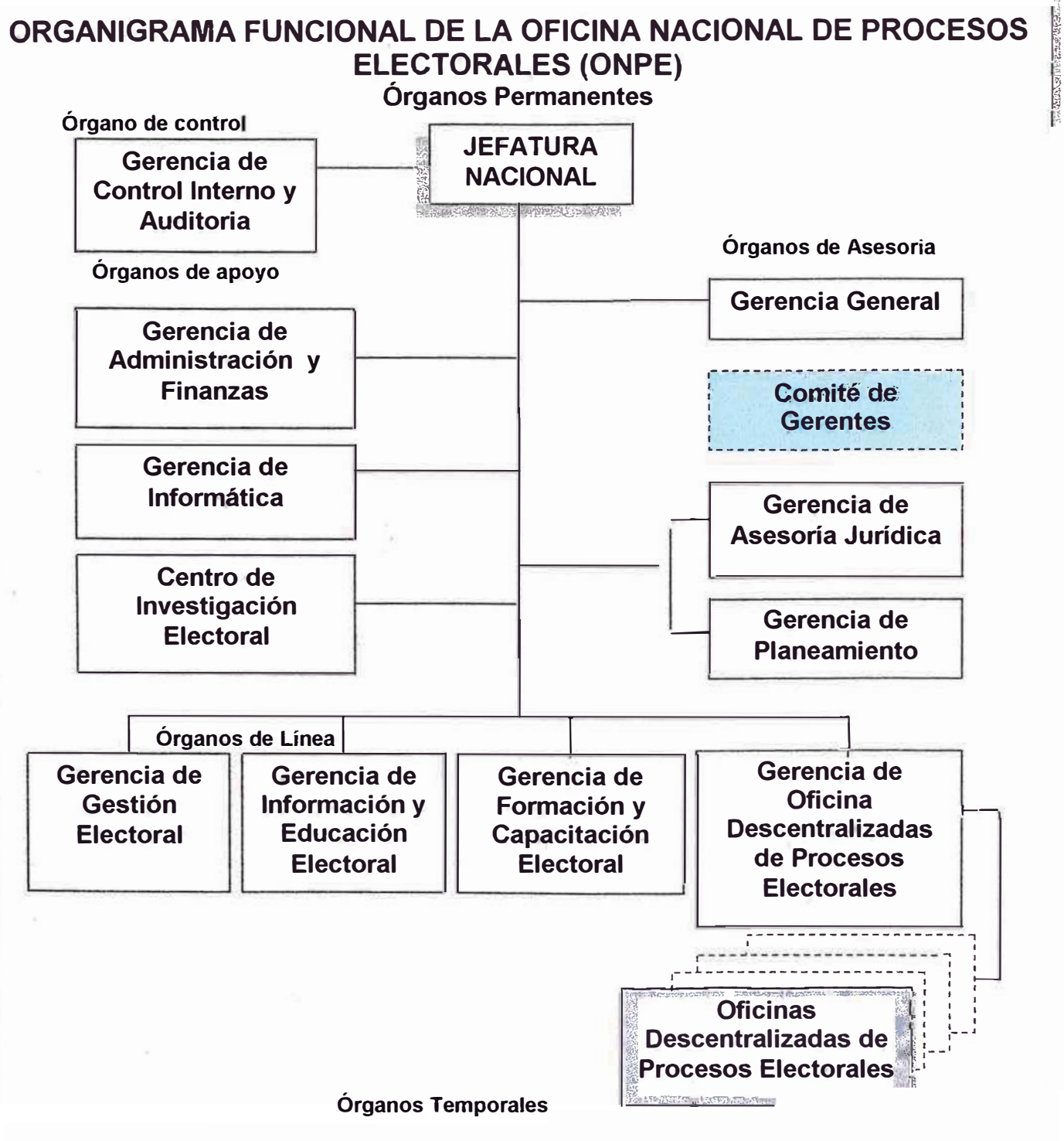
Proceso logísticas despliegue y repliegue del material electoral.

Proceso de Cómputo Electoral y entrega de resultados.

1.2.5 Organización de la empresa

La estructura orgánica de la Oficina Nacional de Procesos Electorales está conformada por Órganos Permanentes y Órganos Temporales de acuerdo al detalle siguiente:

Figura 1: Organigrama funcional de la ONPE



Órganos Permanentes

Órgano de la alta dirección:

Jefatura Nacional

Órgano de control :

Gerencia de Control Interno y Auditoría

Órganos de Asesoria:

Gerencia General

Gerencia de Asesoría Jurídica. Dentro de la cual tenemos:

- Sub Gerencia de Asesoría Electoral
- Sub Gerencia de Asesoría Administrativa.

Gerencia de Planeamiento

- Sub Gerencia de Planeamiento Electoral

Órganos de Apoyo:

Gerencia de Administración y Finanzas. Dentro de la cual tenemos:

- Sub Gerencia de Recursos Humanos
- Sub Gerencia de Logística
- Sub Gerencia de Finanzas

Gerencia de Informática. Dentro de la cual tenemos:

- Sub Gerencia de Plataforma Tecnológica.
- Sub Gerencia de Proyectos Informáticos.
- Sub Gerencia de Operaciones

Centro de Investigación Electoral

Órganos de Línea :

Gerencia de Gestión Electoral

Gerencia de Información y Educación Electoral.

- Sub Gerencia de Información y Educación Electoral.

Gerencia de Formación y Capacitación Electoral.

- Sub Gerencia Formación Ciudadana
- Sub Gerencia de Capacitación Electoral.

Gerencia de Oficinas Descentralizadas de Procesos Electorales

- Sub Gerencia de Oficinas Descentralizadas de Procesos Electorales

Órganos Temporales

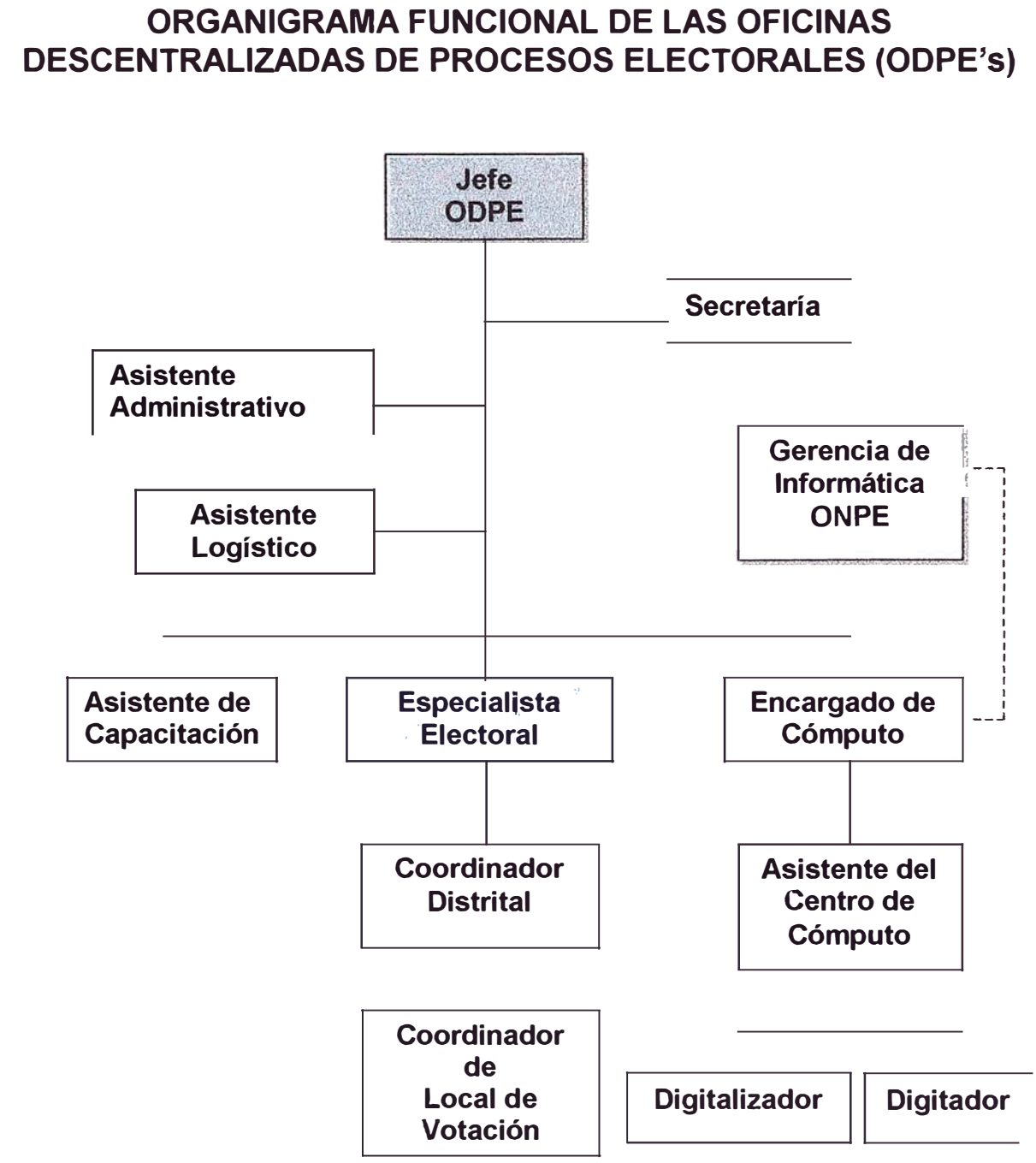
Comité de Gerencia de Procesos Electorales

Oficinas Descentralizadas de Procesos Electorales (ODPE's)

Órganos Consultivos

Comité de Gerentes

Figura 2: Organigrama funcional de la ODPE's (Órganos Temporales)



II. MARCO TEÓRICO

Pasaremos a describir la teoría en la que se sustenta el presente trabajo.

El concepto de seguridad de la información, es mucho más amplio que la simple protección de los datos en el ámbito lógico. Para proporcionar una seguridad real, hemos de tener en cuenta múltiples factores, tanto internos como externos.

En primer lugar, se debe caracterizar el sistema que va ha albergar la información para poder identificar las amenazas, y en ese sentido se hace la siguiente subdivisión:

- Sistemas aislados. Son los que no están conectados a ningún tipo de red. De unos años a esta parte, se han convertido en minoría, debido al auge que ha experimentado Internet.
- Sistema interconectados. Hoy por hoy casi cualquier ordenador pertenece a alguna red, enviando y recogiendo información del exterior casi constantemente. Esto hace que las redes de ordenadores sean cada día más complejas y supongan un peligro potencial que no puede en ningún caso ser ignorado.

El estudio de la seguridad informática puede plantearse bajo los siguientes enfoques:

- Seguridad física. Englobaremos dentro de ésta categoría a todos los asuntos relacionados con la salvaguarda de los soportes físicos de la información, más que de la información propiamente dicha. En este nivel estarían las medidas contra incendios y sobrecargas eléctricas, planes de contingencia, prevención de ataque terroristas, políticas de respaldo de información, restricción y

control de acceso físico a las computadoras de sólo y únicamente a personas autorizadas.

- Seguridad lógica. En este enfoque prestaremos atención a la preservación de la información frente a observadores no autorizados. Para ello podemos emplear la criptografía simétrica como asimétrica.
- Seguridad del canal de comunicación. Los canales de comunicación rara vez se consideran seguros. Debido a que en la mayoría de los casos escapan a nuestro control, ya que pertenecen a terceros y resulta imposible asegurar que no están siendo escuchados o intervenidos.

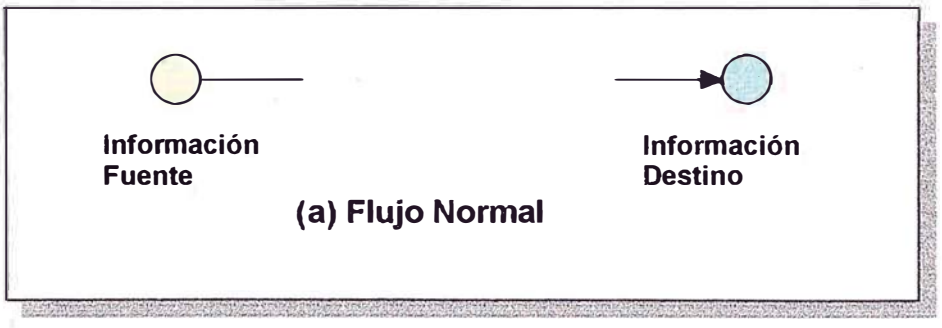
2.1 Seguridad de la Información

El objetivo de la seguridad de la información es:

- Mantener el **secreto**, evitando los accesos no autorizados.
- Mantener la **autenticidad**, evitando modificaciones no autorizadas.

La seguridad tiene su nacimiento con la aparición de los ataques a la información por parte de intrusos (hackers) interesados en el contenido de esta.

Figura 3: Flujo Normal de la información

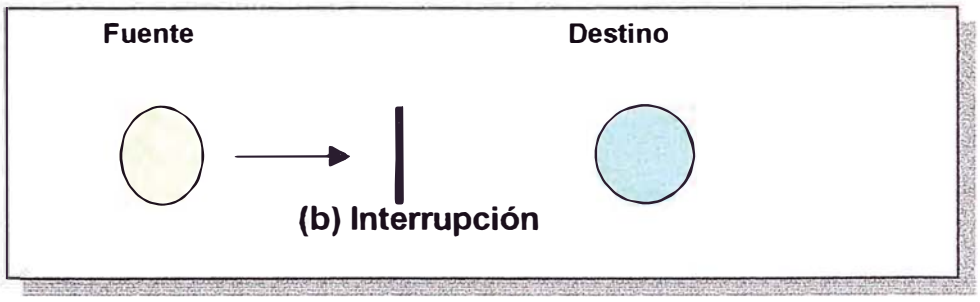


En general, en una comunicación hay un flujo de información desde una **fuente** hacia un **destino** remoto (ver figura anterior), por lo que en el

trayecto se pueden dar cuatro (4) categorías de ataque: interrupción, interceptación, modificación y fabricación.

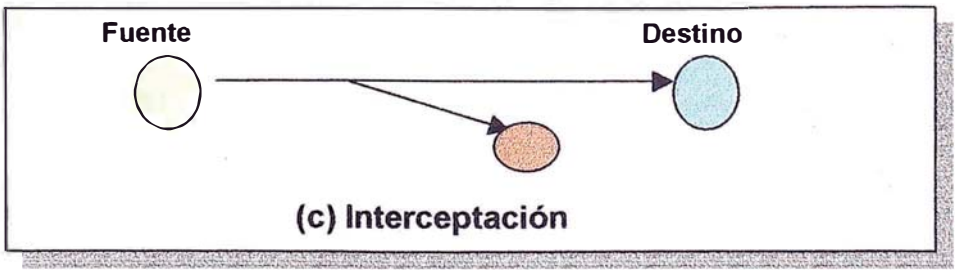
Interrupción.- la información del sistema es destruida o llega a ser inutilizable. Éste es un ataque sobre la disponibilidad. En este ataque se puede incluir ejemplos como el corte de una línea de comunicación, destrucción de una pieza de hardware (Disco Duro), borrado de programas o datos, fallos en el sistema operativo.

Figura 4: Interrupción de la Información



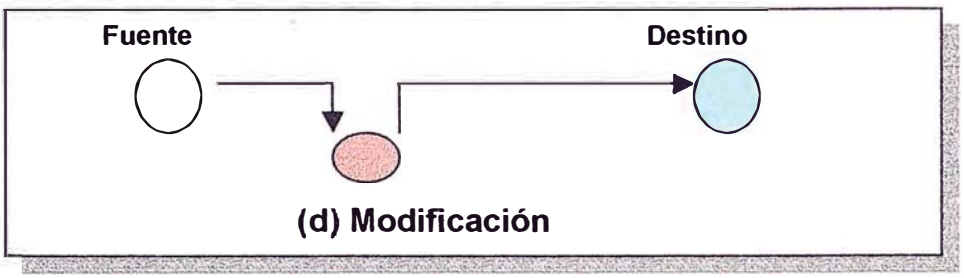
Interceptación.-Se refiere a la participación sin autorización por parte de una persona, computadora o programa en una comunicación. Este es un ataque sobre la confidencialidad. Ejemplo: escucha en línea de datos, copias ilícitas de programas.

Figura 5: Interceptación de la información



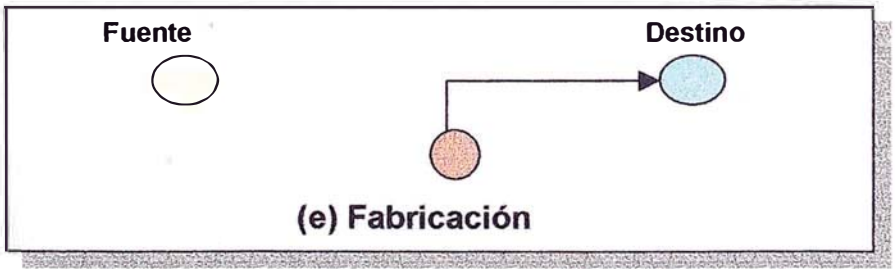
Modificación.- Se refiere a la participación sin autorización, pero no solo accediendo a la información sino alterándola. Los ejemplos incluidos podrían ser los cambios de valores en archivos y programas o la modificación de mensajes transmitidos en una red, modificación de bases de datos.

Figura 6: Modificación de la Información



Fabricación.- Introducción de objetos falsificados en un sistema sin autorización. Este es un ataque sobre la autenticidad. Un ejemplo sería la introducción de mensajes falsos en una red, añadir registros a la base de datos.

Figura 7: Fabricación de Información



Los ataques comentados anteriormente, también se pueden categorizar en dos tipos de ataque, los pasivos y los activos:

Ataques Pasivos.- Son simplemente observaciones de datos sensibles durante una transmisión. La finalidad del intruso es la obtención de la información transmitida. Dentro de las cuales encontramos dos tipos de ataque: **la observación del contenido del mensaje**, que sería el entendimiento por parte del intruso del contenido de una transmisión que contiene información confidencial, como el correo electrónico o una conversación telefónica y el otro tipo de ataque es el **análisis de tráfico,** que sería la observación por parte del intruso sobre la longitud del mensaje, la identificación de los usuarios y la frecuencia de transmisión; pero en ningún caso puede entender la información, pues va cifrada. Los

ataque pasivos son difícilmente detectables, porque no producen una alteración de la información, no obstante son factibles de prevenir.

Ataques Activos.- Incluyen alguna modificación del mensaje o la creación de falsos mensajes y existen varios tipos de ataques:

- **Cambiar la identidad del emisor o receptor**:- ocurre cuando una entidad pretende hacerse pasar por otra.
- **Manipulación de datos**.
- **Repetición**.- capturar una información, guardarla un tiempo y volverla a enviar, produciendo un efecto no autorizado.
- **Denegación de servicio**.- Impedir una comunicación, una respuesta, causar un repudio de usuarios.
- **Encaminamiento incorrecto**.- Atacan a los nodos de la red, pues no están tan protegidos, como los terminales.

2.2 Delitos Informáticos

Son acciones que vulneran la confidencialidad, integridad y disponibilidad de la información. Siendo alguno de ellos los siguientes:

Fraude. Acto deliberado de manipulación de datos perjudicando a una persona física o jurídica que sufre de esta forma una pérdida económica. El autor del delito logra de esta forma un beneficio normalmente económico.

Sabotaje. Acción con la que se desea perjudicar a una empresa entorpeciendo deliberadamente su marcha, averiando sus equipos, herramientas, programas, etc. El autor no logra normalmente con ello beneficios económicos, pero pone en jaque a la organización.

Chantaje. Acción que consiste en exigir una cantidad de dinero a cambio de no dar a conocer información privilegiada o confidencial y que puede afectar gravemente a la empresa, por lo general a su imagen corporativa.

Mascarada. Utilización de una clave por una persona no autorizada y que accede al sistema suplantando una identidad. De esta forma el intruso se

hace dueño de la información, documentación y datos de otros usuarios con los que puede, por ejemplo, chantajear a la organización.

Virus. Código diseñado para introducirse en un programa, modificar o destruir datos. Se copia automáticamente a otros programas para seguir su ciclo de vida. Es común que se expanda a través de plantillas, macros de aplicaciones y archivos ejecutables.

Gusanos. Virus que se activa y transmite a través de la red. Tiene como finalidad su multiplicación hasta agotar el espacio en disco o memoria RAM. Suele ser uno de los ataque más dañinos porque normalmente produce un colapso en la red.

Caballos de Troya. Virus que entra al computador y posteriormente actúa de forma similar a este hecho de la mitología griega. Así, parece ser un programa inofensivo, cuando en realidad esta haciendo otra y expandiéndose.

La solución para evitar estos delitos informáticos es poner en marcha una adecuada política de seguridad en la empresa o institución.

El sustento teórico del presente trabajo, es la criptografía.

2.3 Criptología.

Es la disciplina dedicada a comunicarse secretamente.

Criptografía Rama de las matemáticas y en la actualidad de la informática y la telemática que hace uso de métodos y técnicas matemáticas con el objeto de cifrar un mensaje o archivo, por medio de un algoritmo, usando una o m'as claves.

Criptoanálisis la que se dedica a "quebrar" o comprometer dichos sistemas.

Los criptosistemas se pueden dividir en dos grandes grupos principales: criptosistemas de **clave privada o simétricos** y criptosistemas de **clave pública o asimétricos.**

Cifrar o encriptar consiste en aplicar un algoritmo matemático a un texto plano, claro o legible, para convertirlo en algo totalmente ininteligible.

2.4 Sistemas Simétricos

La clave utilizada para descifrar es la misma clave que se utilizó para cifrar. De esta forma, si usted quiere cifrar un mensaje, debe poseer el programa (algoritmo de encriptación) y su clave personal, y esa clave personal debe distribuirla a todas aquellas personas, a las que usted pretende enviarle mensajes cifrados o encriptados. Como se podrá observar, presentan el inconveniente de que para ser empleados en canales de comunicación, la clave ***K,*** debe estar tanto en el emisor, como el receptor, lo cual nos lleva a preguntarnos cómo transmitir la clave de forma segura.

Algunos algoritmos de clave privada:

DES (Data Encription Standard): Es un algoritmo diseñado por IBM y utilizado habitualmente en los años 70. Es un método de cifrado altamente resistente frente a los ataques criptoanalíticos diferenciales. Por desgracia su tamaño de clave (56 bits) la hace vulnerable a ataques de la fuerza bruta, de que incluso se utilizó para cifrar las comunicaciones entre la NASA y la Estación Espacial Liberty. Actualmente DES, está condenado al ostracismo, debido a que la longitud de la clave está limitada a 56bits (en un principio, lo deseable era que cuanto más larga sea la clave, mejor. Aunque a partir de un punto de suficiente longitud, da casi igual, aparte de tener que compartir la clave).

La variante Triple DES, es muy difícil de quebrar.

Triples DES: Es un DES múltiple y consiste en aplicar varias veces el algoritmo DES, con diferentes claves al mensaje original. Esto se puede realizar ya que DES, **no** presenta estructura de grupo y responde a la siguiente estructura:

$$C = E_{k1}\left(D_{k2}\left(E_{K1}\left(M\right)\right)\right)$$

Es decir, codificamos con la subclave ***k1***, decodificamos con ***k2*** y volvemos a codificar con ***K1***, la clave resultante es la concatenación de ***k1*** y ***k2***, con una longitud de 112 bits.

Algunos algoritmos de clave privada: IDEA, CAST, TWOFISH, BLOWFISH, SAFER, RC5, SERPENT, RIJNDAEL.

2.5 RIJNDAEL

Rijndael, es el Nuevo AES (Advanced Encryption Standard), (ver anexo FIPS – Federal Information Processing Standards Publication 197). Es un algoritmo de libre distribución y este estándar puede ser usado por cualquier institución cuando determinen que la información sensible requiere de protección criptográfica.

Tras un largo proceso de casi cuatro años, el NIST (instituto nacional de estándares y tecnología) norteamericano ha seleccionado al algoritmo "RIJNDAEL" como futuro estándar AES.

El 2 de Octubre de 2000, el NIST hizo público el algoritmo ganador de la convocatoria AES (estándar de cifrado avanzado), que sustituirá al actual DES (estándar de cifrado de datos) hasta bien entrado el siglo XXI. Para sorpresa de muchos, "RIJNDAEL" es un algoritmo belga, venciendo a algoritmos norteamericanos y a criptólogos de reputada fama mundial. "RIJNDAEL", por ejemplo, ha vencido al algoritmo "twofish", diseñado por un equipo liderado por el conocido Bruce Schneier (autor, entre otras cosas, del libro "Applied Cryptography"), y al algoritmo "RC6", diseñado, entre otros, por Ronald Rivest (la "R" del algoritmo RSA).

Al contrario que en la convocatoria DES, dos décadas atrás, AES ha sido un concurso abierto a cualquier participante mundial que cumpliese con una serie de requisitos, como velocidad tanto en software como en hardware, reducida ocupación de memoria, implementación eficiente tanto en procesadores modernos como en CPUs de 8 bits (utilizadas, por ejemplo, en tarjetas inteligentes), etc. El ganador debe, por otra parte,

liberar cualquier tipo de patente en el algoritmo, para potenciar así el desarrollo de soluciones tecnológicas de buena calidad.

El tamaño de clave debía ser de, al menos, 128, 192 y 256 bits (debe admitir los tres), y el tamaño de bloque de cifrado (se trata de un sistema de cifrado en bloque, no un sistema de cifrado en flujo) debía ser de 128 bits. Esto se compara muy favorablemente con el DES, de 56 bits de clave y 64 bits de tamaño de bloque.

Los motivos para seleccionar a "RIJNDAEL", según el web del NIST son su buena combinación de seguridad, velocidad, eficiencia (en memoria y puertas lógicas), sencillez y flexibilidad.

Los otros cuatro algoritmos finalistas ("TwoFish", "RC6", "MARS" y "Serpent") son considerados como "seguros", por lo que se tiene certeza sobre su calidad. Como contrapartida, algunos de ellos (por ejemplo, RC6) pueden estar cubiertos por patentes a las que, al no resultar vencedores, no se les exige renuncia. Esto es algo a valorar por parte de los desarrolladores.

Durante todo el desarrollo del proceso AES, todos los algoritmos y criterios de diseño estuvieron disponibles de forma pública y abierta, por lo que el escrutinio al que han sido sometidos todos los finalistas ha sido enorme, acorde con la importancia del nuevo AES. Todos los participantes han contribuido al proceso, analizando las posibles vulnerabilidades de sus competidores. De hecho se han dado casos curiosos; el algoritmo "magenta", por ejemplo, fue "criptoanalizado" en el mismo encuentro en el que se presentó.

2.5.1 Introducción al Rijndael

Este estándar define el algoritmo Rijndael, como un cifrado de bloque simétrico que puede procesar, bloques de datos de 128 bits, usando claves de cifrado con longitud de 128, 192 y 256 bits.

Rijndael fue diseñado para manejar bloques de mayor tamaño y claves de mayor longitud, sin embargo ellos no son adoptados dentro de este estándar.

Por todas partes de el resto de este estándar, el algoritmo especificado aquí, debería ser referido como " **El AES algoritmo**".

El algoritmo puede ser usado con las tres diferentes longitudes de claves indicadas después y por lo tanto estas formas diferentes pueden ser referidas como "AES-128", "AES-192" y "AES-256".

Esta especificación incluye las siguientes secciones:

- Definición de términos , acrónimos, parámetros de algoritmos, símbolos y funciones.
- Notación y convención usada en la especificación del algoritmo, incluyendo el ordenamiento y numeración de bits, bytes y palabras (words).
- Propiedades matemáticas que son usadas para el entendimiento del algoritmo.
- Especificación del algoritmo, cubre la expansión de claves, rutina de encriptación y desencriptación.

Este estándar dentro de su documentación FIPS-197, concluye con varios apéndices que incluye ejemplos paso a paso para la expansión de claves y el cifrado, ejemplo de vectores para el cifrado y el cifrado inverso.

2.5.2 Glosario de Términos y Acrónimos

Las siguientes definiciones son usadas a través de este estándar.

AES	Estándar de Encriptación Avanzada
Affine multiplicación	Una transformación affine consistente de

Transformation	por una matriz seguido por la adición de un vector.
Array	Una colección enumerada de entidades idénticas (un arreglo de bytes).
BIT	Un digito binario teniendo un valor 0 o 1.
Block (bloque)	Secuencia de bits binarios que comprende o esta incluida, la entrada (input), salida (output), estado (state) y una ronda de clave (round key). La longitud de una secuencia es el número de bits que eso contiene. Los blocks (bloques) también son interpretados como arreglo de bytes.
Byte	Un grupo de ocho bits que es tratado como una sola entidad o como un arreglo de 8 bits individuales.
Cipher (Cifrado)	Serie o conjunto de transformaciones que convierte texto legible a texto cifrado usando la clave de cifrado.
Cipher Key	Clave de cifrado, clave criptográfica secreta que es usada por la rutina de expansión de claves, para generar un conjunto claves por ronda, puede ser representado como un arreglo rectangular de bytes, teniendo cuatro filas y ***Nk*** columnas (de 32 bits, word).
Ciphertext	texto cifrado, Data de salida del cifrador o entrada a el cifrador inverso.
Inverse Cipher	Serie o conjunto de transformaciones que convierte texto cifrado a texto legible o texto plano, usando la clave de cifrado.
Key Expansion	Rutina usada para generar una serie de claves por ronda desde o de la clave de cifrado.

Plaintext	Texto legible, datos de entrada al cifrador o salida del cifrador inverso.
Round Key	Claves por ronda, son valores derivados de la clave de cifrado, usando la rutina de expansión; ellos son aplicados al estado (state) dentro del cifrador y del cifrador inverso.
State (Estados)	Resultado intermedio del cifrado, que puede ser representado como un arreglo rectangular de bytes, teniendo cuatro filas y ***Nb*** columnas (de 32 bits, word).
S-box	Tabla de sustitución no lineal, usada dentro de varias transformaciones de sustitución de bytes y dentro de la rutina de expansión de claves a ejecutar sustitución de uno por uno del valor de un byte.
Word (palabra)	Un grupo de 32 bits que es tratado como una sola entidad o como un arreglo de 4 bytes.

2.5.3 Parámetros del Algoritmo, Símbolos y Funciones

Las siguientes parámetros de algoritmo, símbolos y funciones son usadas a través de este estándar.

AddRoundKey()	Transformación dentro del cifrador o el cifrador inverso dentro del cual una clave de ronda es adicionada a el estado usando un operación XOR. (ejemplo, para ***Nb*** = 4, la longitud de la clave por ronda es igual a 128 bits/16 bytes).
InvMixColumns()	Transformación dentro del cifrador inverso que es la inversa de MixColumns().
InvShiftRows()	Transformación dentro del cifrador inverso que es la inversa de ShiftRows().

InvSubBytes()	Transformación dentro del cifrador inverso que es la inversa de SubBytes().
K	Clave del cifrado.
MixColumns()	Transformación dentro del cifrador que toma todas las columnas de el estado y mezcla su data (independiente del otro) para producir nuevas columnas.
Nb	Número de columnas (de 32 bits, words) que comprende el estado. Para este estándar, Nb = 4.
Nk	Número palabras de 32 bits que comprende la clave de cifrado. Para este estándar, Nk = 4, 6 ó 8.
Nr	Número de rondas, el cual es una función de Nk y Nb (que es fijo). Para este estándar, Nr = 10, 12 ó 14.
Rcon[ ]	Un arreglo de palabra constante por ronda.
Rotword()	Función usada dentro de la rutina de expansión de clave que toma una palabra de cuatro bytes y ejecuta una permutación cíclica.
ShiftRows()	Transformación dentro del cifrador que procesa el estado por desplazamiento cíclico de las tres últimas filas del estado por diferentes offsets.
SubBytes()	Transformación dentro del cifrador que procesa el estado usando una tabla de sustitución de byte no lineal (S-box), que opera sobre cada uno de los bytes del estado, independientemente.

SubWord()	Función usada dentro de la rutina de expansión que toma una palabra de entrada de cuatro bytes y aplica un S-box a cada uno de los cuatro bytes para producir un palabra de salida.
XOR	Operación con OR Exclusivo.
$\oplus$	Operación OR exclusivo.
	Multiplicación de dos polinomios (cada uno con grado menor de 4) modulo $X^4 + 1$.
•	Multiplicación de Campos Finitos.

2.5.4 Notaciones y Convenciones

Inputs y Outputs (Entradas y Salidas)

Las entradas y salidas para el algoritmo AES, cada uno consiste de secuencias de 128 bits (dígitos con valores de 0 ó 1). Estas secuencias deberían algunas veces ser referidas como bloques y el numero de bits que ellos contienen deberían ser referidos como su longitud. La clave de cifrado para el algoritmo AES es una secuencia de 128, 192 ó 256 bits. Otras longitudes de entradas, salidas y clave de cifrado, no son permitidas por este estándar.

Los bits dentro de tal secuencia deberían ser numerados comenzando en cero y finalizando en uno menos que la longitud de secuencia (longitud de bloque o longitud de clave). El numero ***i*** atachado a un bit es conocido como índice y debería estar dentro de uno de los rangos $0 \leq i < 128$, $0 \leq i < 192$ o $0 \leq i < 256$ dependiendo de la longitud de bloque y longitud de clave (especificado antes).

Bytes

La unidad básica para el procesamiento dentro del algoritmo AES es un byte, una secuencia de ocho bits, tratados como una

sola entidad. Las secuencia de bits de entradas, salidas y clave de cifrado descritas anteriormente son procesadas como arreglos de bytes, que son formadas dividiendo estas secuencias dentro de grupos de ocho bits contiguos, para formar arreglos de bytes. Para una entrada, salida o clave de cifrado denotado por **a,** los bytes dentro del arreglo resultante deberían ser referenciado; usando uno de las dos formas, a_n o a[n], donde **n** debería estar dentro de uno de los siguientes rangos:

Longitud de la clave (Key length), Longitud de bloque (Block length).

Key length = 128 bits, $0 \leq \mathbf{n} < 16$; Block length = 128 bits, $0 \leq \mathbf{n} < 16$;

Key length = 192 bits, $0 \leq \mathbf{n} < 24$;

Key length = 256 bits, $0 \leq \mathbf{n} < 32$;

Todos los valores del byte dentro del algoritmo AES, deberían ser Presentados, como una concatenación de bits individuales con valores (0 ó 1) entre corchetes en el siguiente orden {b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0}. Estos bytes son interpretados como elementos de campos finitos (finite fiels) usando una **representación polinomial**.

$b_7x^7 + b_6X^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0 = \Sigma^7_{i=0}\ \mathbf{b_i\ x^i}$.

Por ejemplo, {01100011} identifica el especifico elemento del campo finito $x^6 + x^5 + x + 1$.

Es conveniente notar que los valores del byte usando notación hexadecimal se divide en **dos grupos de cuatro bits**, siendo denotado por un solo carácter, como sigue en la figura.

Tabla 1. Representación hexadecimal de conjunto de bits

Bits	Caracter	Bits	Caracter	Bits	Caracter	Bits	Caracter
0000	0	0100	4	1000	8	1100	c
0001	1	0101	5	1001	9	1101	d
0010	2	0110	6	1010	a	1110	e
0011	3	0111	7	1011	b	1111	f

En adelante el elemento {01100011} puede ser representado como {63}, donde el carácter denotando el grupo de cuatro bits, conteniendo el numero mas alto esta otra vez a la izquierda.

Algunas operaciones del campo finito involucra un Bit adicional (b_8) a la izquierda de un byte de 8-bits. Cuando este bit extra este presente, eso debería aparecer como '{01}' inmediatamente antes o precediendo el byte de 8-bits; por ejemplo, una secuencia con 9-bit debería ser representado como {01}{1b}.

Arreglos de bytes (Arrays of Bytes)

Los arreglos de bytes deberían ser representados de la siguiente forma: $a_0a_1a_2a_3.....a_{15}$

El ordenamiento de los bytes y los bits dentro de los bytes, son derivados desde la secuencia de entrada 128-bit.

$Input_0\ input_1\ input_2\ .....\ input_{126}\ input_{127}$

Como sigue:

$a_0 = \{ Input_0\ input_1\ .....\ input_7\};$

$a_1 = \{ Input_8\ input_9\ .....\ input_{15}\};$

•

•

$a_{15} = \{ Input_{120}\ input_{121}\ .....\ input_{127}\}.$

El diseño puede ser extendido a secuencia de longitud mayor (ejemplo. Para claves de 192 y 256 bits).

Tabla 2. Indice para los bytes y bits.

Secuencia de bits de entrada	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Numero de bytes	0								1								2							
Numero de bit dentro de un byte	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

El estado o cifrado intermedio (State)

Internamente, las operaciones del algoritmo AES, son ejecutadas sobre un arreglo de bytes de dos dimensiones llamado el estado (State). El estado o resultado intermedio del cifrado, consiste de cuatro (4) filas de bytes, cada uno conteniendo ***Nb*** bytes, donde ***Nb*** es la longitud de bloque dividida por 32. Dentro del arreglo estado (state), denotado por **s**, cada byte individual tiene dos índices, con número de fila **r** dentro del rango $0 \leq \mathbf{r} < 4$; y con número de columna **c** dentro del rango $0 \leq \mathbf{c} < \mathbf{Nb}$. Esto permite que un byte individual del estado (state), sea referido como $S_{r,c}$ o $S[\mathbf{r},\mathbf{c}]$. Para este estándar, $\mathbf{Nb}=4$, ejemplo $0 \leq \mathbf{c} < 4$.

En el inicio de la descripción del cifrador y el cifrador inverso, la entrada (input), el arreglo de bytes in_0. in_1, ... in_{15} es copiado dentro de un arreglo estado (state) como se muestra en las siguientes tablas. Las operaciones del cifrador y el cifrador inverso son entonces conducidos sobre el arreglo estado (state), después del cual, el valor final es copiado a la salida (output), el arreglo de bytes out_0. out_1, ... out_{15}.

Tabla 3. Arreglo Estado, entrada y salidad (input y output)

Bytes entrada $Nb_1\ Nb_2\ Nb_3\ Nb_4$					**Arreglo estado** w_0 w_1 w_2 w_3					**Bytes salida**			
In_0	In_4	In_8	In_{12}		$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$		Out_0	Out_4	Out_8	Out_{12}
In_1	In_5	In_9	In_{13}	⇒	$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$	⇒	Out_1	Out_5	Out_9	Out_{13}
In_2	In_6	In_{10}	In_{14}		$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$		Out_2	Out_6	Out_{10}	Out_{14}
In_3	In_7	In_{11}	In_{15}		$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$		out_3	out_7	Out_{11}	Out_{15}

Dado que, en el comienzo del cifrador o del cifrador inverso, el arreglo de entrada, **in**, es copiado al arreglo estado, de acuerdo al siguiente esquema: donde **r** = fila y **c** = columna.

$S[\mathbf{r}, \mathbf{c}] = in[\mathbf{r}+4\mathbf{c}]$ para $0 \leq \mathbf{r} < 4$ y $0 \leq \mathbf{c} < \boldsymbol{Nb}$,

Y en el fin del cifrador y cifrador inverso, el estado es copiado a un arreglo de salidad out como sigue:

$out[\mathbf{r}+4\mathbf{c}] = s[\mathbf{r}, \mathbf{c}]$ para $0 \leq \mathbf{r} < 4$ y $0 \leq \mathbf{c} < \boldsymbol{Nb}$.

El Estado como un arreglo de columnas

Los cuatro (4) bytes dentro de cada columna del arreglo estado (state) forman palabras (Word) de 32 bits, donde el número de fila r provee un índice para los cuatro bytes dentro de cada palabra (Word). El estado puede por lo tanto ser interpretado como un arreglo unidimensional de palabras de 32 bits (columnas), $w_0 \ldots w_3$, donde el número de columna c, provee un índice dentro de este arreglo. Por lo tanto para la tabla anterior, el estado puede ser considerado como un arreglo de cuatro (4) palabras, como sigue:

$W_0 = s_{0,0}\ s_{1,0}\ s_{2,0}\ s_{3,0}$ $\quad W_2 = s_{0,2}\ s_{1,2}\ s_{2,2}\ s_{3,2}$

$W_1 = s_{0,1}\ s_{1,1}\ s_{2,1}\ s_{3,1}$ $\quad W_3 = s_{0,3}\ s_{1,3}\ s_{2,3}\ s_{3,3}$.

2.5.5 Matemática Preliminar.

Todos los bytes dentro del algoritmo AES, son interpretados como elementos de campos finitos usando la notación polinomial, visto anteriormente. Los elementos de los campo finitos pueden ser adicionados y multiplicados, pero estas operaciones son diferentes de aquéllas usadas para los números. Por lo que haremos una introducción de los conceptos de matemática básica, necesitados para entender la especificación del algoritmo.
Cuando trabajamos en un cuerpo primo p, sabemos que se asegura la existencia de un único inverso multiplicativo. En este caso se dice que estamos trabajando en Campos de Galois GF(p).
Algunos usos en criptografía:

- Sistemas de clave pública cuando la operación de cifra es $C = M^{\bullet} \bmod p$ (cifrador El Gamal)
- Aplicaciones en $GF(q^n)$, polinomios módulo q y de grado n:
  $a(x) = a_{n-1}*x^{n-1} + a_{n-2}*x^{n-2} + \ldots + a_1*x + a_0$ (cifrador RIJNDAEL).

Campos de Galois del tipo $GF(q^n)$

$$a(x) = a_{n-1}*x^{n-1} + a_{n-2}*x^{n-2} + \ldots + a_1*x + a_0$$

Es un polinomio de grado n-1 o menor.
Los elementos a_i son parte del CCR (conjunto completo de restos) del módulo q.
Cada elemento a(x) es un resto módulo p(x), siendo p(x) un polinomio irreducible de grado n (que no puede ser factorizado en polinomios de grado menor que n).
$GF(2^n)$ es interesante porque CCR(2) = {0, 1} Þ bits.

GF(2^3) tiene 8 elementos o restos polinómicos que son: **0**, **1**, **x**, **x+1**, x^2, x^2+1, x^2+x, x^2+x+1, los 8 restos de un polinomio de grado n-1 (n=3).

Suma de campos de Galois GF(2^n)

Si el modulo de trabajo es 2 (con restos igual a bits 0 y 1), las operaciones suma y resta serán un OR Exclusivo:

$0 \oplus 1 \bmod 2 = 1$ $\quad$ $1 \oplus 0 \bmod 2 = 1$

$0 \oplus 0 \bmod 2 = 0$ $\quad$ $1 \oplus 1 \bmod 2 = 0$

CG(2^2)

⊕	0	1	X	X+1
0	0	1	X	X+1
1	1	0	X+1	X
X	X	X+1	0	1
X+1	X+1	X	1	0

Restos: 0, 1, x, x+1

Como los resultados deberán pertenecer al cuerpo, aplicaremos reducción por coeficientes:

$x + (x + 1) = 2x + 1 \bmod 2 = 1$

$1 + 1 = 2 \bmod 2 = 0$

Producto en campos de Galois GF (2^n)

La operación de multiplicación puede entregar elementos que no pertenezcan al cuerpo, potencias iguales o mayores que n, por lo que es necesario la reducción por exponente.

CG(2^2)

⊗	0	1	X	X+1
0	0	0	0	0
1	0	1	X	X+1
X	0	X	X+1	1
X+1	0	X+1	1	X

Restos: 0, 1, x, x+1

Sea el polinomio irreducible de grado n = 2, $p(x)= x^2 + x +1$

Luego: $x^2 = x +1$

Entonces la multiplicación en la matriz de $(x+1) \otimes (x+1)$ será:

$(X+1)*(X+1) = x^2 + 2x +1$, pero $x^2 = x + 1$, entonces reemplazamos:

$= (x+1) + 2x + 1$

$= 3x + 2 \bmod 2 =$

En el algoritmo AES, se trabaja con 8 bits, por lo que las operaciones se realizan en $GF(2^8)$.

Operaciones a nivel byte en campos finitos de galois $GF(2^8)$

- Suma y multiplicación. Son cálculos en campos de Galois $GF(2^8)$ con 8 bits. Para la reducción del exponente se usara un polinomio primitivo p(x).
- Producto por **x.** Esta operación conocida como xtime(a), al igual que el caso anterior, usa la reducción de exponente. Puede implementarse fácilmente con desplazamientos y operaciones OR exclusivo.

Adición

La adición de dos (2) elementos dentro de un campo finito es alcanzado adicionando los coeficientes por el correspondiente exponente dentro del polinomial para los dos elementos. La adición es ejecutada con la operación XOR (denotada por $\oplus$) modulo 2, tal que:

a	b	XOR ($\oplus$)
0	0	0
0	1	1
1	0	1
1	1	0

Consecuentemente, la substracción de polinomios es idéntica a la adición de polinomios.

Alternativamente, la adición de elementos de campo finitos puede ser descrito como la adición modulo 2, de los correspondientes bits dentro del byte. Para dos bytes $\{a_7a_6a_5a_4a_3a_2a_1a_0\}$ y $\{b_7b_6b_5b_4b_3b_2b_1b_0\}$, la suma es $\{c_7c_6c_5c_4c_3c_2c_1c_0\}$, donde cada $c_i = a_i \oplus b_i$ (ejemplo

$c_7 = a_7 \oplus b_7$, $c_6 = a_6 \oplus b_6$, ... $c_0 = a_0 \oplus b_0$).

Por ejemplo, la siguientes expresiones son equivalentes el uno del otro:

$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2$ (notación polinomial);

$\{01010111\} \oplus \{10000011\} = \{11010100\}$ (notación binaria);

$\{57\} \oplus \{83\} = \{d4\}$ (notación Hexadecimal).

Vamos a sumar los valores hexadecimales 57 y 83:

$A = 57_{16} = 0101\ 0111_2$ $B = 83_{16} = 1000\ 0011_2$

que expresados en polinomios dentro de $GF(2^8)$ serán:

$A = 0101\ 0111_2 = x^6 + x^4 + x^2 + x + 1$

$B = 1000\ 0011_2 = x^7 + x + 1$

Sumando: $A + B = (x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) \bmod 2$

$A + B = (x^7 + x^6 + x^4 + x^2 + 2x + 2) \bmod 2$

$A + B = x^7 + x^6 + x^4 + x^2 = 1101\ 0100 = D4_{16}$

Y lo mismo se obtiene con la suma OR exclusivo:

$0101\ 0111 \oplus 1000\ 0011 = 1101\ 0100 = D4_{16}$

Multiplicación

Dentro de la representación polinomial, multiplicación en $GF(2^8)$ (denotado por •) corresponde con la multiplicación de polinomios, modulo un polinomio irreducible de grado 8. Un polinomio es irreducible si sólo es divisible por 1 y por el mismo. Para el

algoritmo AES, este polinomio irreducible es: $m(x) = x^8 + x^4 + x^3 + x + 1$, o

{01}{1b} dentro de la notación hexadecimal.

Por ejemplo, {57} • {83} = {c1}, porque

Vamos a multiplicar los valores hexadecimales 57 y 83:

$A = 57_{16} = 0101\ 0111_2$ $B = 83_{16} = 1000\ 0011_2$

que expresados en polinomios dentro de $GF(2^8)$ serán:

$A = 0101\ 0111_2 = x^6 + x^4 + x^2 + x + 1$

$B = 1000\ 0011_2 = x^7 + x + 1$

Sea $m(x) = x^8 + x^4 + x^3 + x + 1 \Rightarrow x^8 = x^4 + x^3 + x + 1$

$A \bullet B = (x^6 + x^4 + x^2 + x + 1)*(x^7 + x + 1) \bmod 2$

$A \bullet B = x^{13} + x^{11} + x^9 + x^8 + 2x^7 + x^6 + x^5 + x^4 + x^3 + 2x^2 + 2x + 1$

$A \bullet B = x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$

Este resultado hay que reducirlo por **$m(x) = x^8 + x^4 + x^3 + x + 1$**

m(x): $\mathbf{x^8} = x^4 + x^3 + x + 1$.

$x^{13} = x^5 * x^8 = x^5 * (x^4 + x^3 + x + 1) = x^9 + x^8 + x^6 + x^5$

$x^{13} = x^9 + x^8 + x^6 + x^5 = x * (x^4 + x^3 + x + 1) + (x^4 + x^3 + x + 1) + + x^6 + x^5$

$x^{13} = (x^5 + x^4 + x^2 + x) + (x^4 + x^3 + x + 1) + + x^6 + x^5$

$\mathbf{x^{13}} = x^6 + x^3 + x^2 + 1$

$x^{11} = x^3 * x^8 = x^3 * (x^4 + x^3 + x + 1)$ $\mathbf{x^{11}} = x^7 + x^6 + x^4 + x^3$

$x^9 = x * x^8 = x * (x^4 + x^3 + x + 1)$ $\mathbf{x^9} = x^5 + x^4 + x^2 + x$

reemplazando los cálculos anteriores, en la expresión A • B, tenemos

$A \bullet B = (x^6 + x^3 + x^2 + 1) + (x^7 + x^6 + x^4 + x^3) + (x^5 + x^4 + x^2 + x) + (x^4 + x^3 + x + 1) + x^6 + x^5 + x^4 + x^3 + 1 \bmod 2$

$A \bullet B = x^7 + x^6 + 1$ = 1100 0001 = {c1} = $c1_{16}$

La reducción modular por m(x), asegura que el resultado debería ser un polinomio binario de grado menor que 8, y así puede ser representado por un byte.

La multiplicación definida anteriormente es asociativa, y el elemento {01}, es la identidad multiplicativa. Para cualquier polinomio binario no cero b(x) de grado menor que 8, la inversa multiplicativa de b(x), denotado por $b^{-1}(x)$, puede ser encontrado como sigue: el Algoritmo Extendido de Euclides, es usado para calcular polinomios a(x) y c(x) tal que: $b(x)a(x) + m(x)c(x) = 1$

Dado que $a(x) \bullet b(x) \bmod m(x) = 1$, lo cual significa

$$b^{-1}(x) = a(x) \bmod m(x)$$

además, para cualquier a(x), b(x) y c(x) dentro del campo, se mantiene que: $a(x) \bullet (b(x) + c(x)) = a(x) \bullet b(x) + a(x) \bullet c(x)$.

Es seguido que el conjunto de 256 posible valores del byte, con XOR usado como adición y la multiplicación definidas anteriormente, tiene la estructura de los campos finitos $GF(2^8)$.

Multiplicación por x

Multiplicar el polinomio binario definido anteriormente

$b(x) = b_7x^7 + b_6X^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$, con el polinomial **x**, resulta así:

$$b_7x^8 + b_6X^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0\mathbf{x}$$

el resultado de $x \bullet b(x)$, es obtenido, reduciendo el anterior resultado por modulo $m(x) = x^8 + x^4 + x^3 + x + 1$. Si $b7 = 0$, el resultado esta ya en forma reducida. Si $b_7 = 1$, la reducción es consumada por substracción (XORing) del polinomio m(x). Eso sigue que multiplicando por **x** (ejemplo, {00000010} ó {02}), puede ser implementado en el nivel byte, como un desplazamiento a la izquierda y un consiguiente condicional XOR con {1b}. Esta operación sobre los bytes es denotado como xtime().

Multiplicación con grandes exponentes de **x**, puede ser implementado por aplicación repetida de xtime().

Adicionando resultados intermedios, multiplicando por cualquier constante puede ser implementado.

Por ejemplo, {57} • {13} = {fe}, porque

{57} • {02} = xtime({57}) = {ae}

{57} • {04} = xtime({ae}) = {47}

{57} • {08} = xtime({47}) = {8e}

{57} • {10} = xtime({8e}) = {07}

así

{57} • {13} = {57} • ({01} ⊕ {02} ⊕ {10})

= {57} ⊕ {ae} ⊕ {07}

= {fe}

Polinomios con coeficientes en GF(2^8)

Polinomios de cuatro términos, pueden ser definidos, con coeficientes que son elementos de campos finitos, como:

a(x) = $a_3x^3 + a_2x^2 + a_1x + a_0$

el cual debería ser denotado como una palabra (Word), dentro de la forma [a_0, a_1, a_2, a_3]. Note que los polinomios dentro de esta sección, se comportan algo diferente que los polinomios usados dentro de la definición de elementos campos finitos, pero ambos tipos de polinomios usan el mismo indeterminante, **x**. Los coeficientes dentro de esta sección son ellos mismos elementos de campos finitos. Ejemplo, bytes, en vez de bits; también la multiplicación de polinomios de cuatro términos usa un polinomio de reducción diferente, definido más adelante.

La diferencia debería siempre ser claro desde el contexto.

Para ilustrar las operaciones de adición y multiplicación, dejar que:

b(x) = $b_3x^3 + b_2x^2 + b_1x + b_0$, definido como el segundo polinomial de cuatro términos. La adición es ejecutada adicionando los coeficientes campos finitos de similar potencia de x. Esta adición corresponde a una operación XOR entre los

correspondientes bytes dentro de cada palabra (Word), es decir, el XOR de los valores de la palabra completa.

Así, usando la ecuaciones anteriores tenemos:

a(x) + b(x) = $(a_3 \oplus b_3)x^3 + (a_2 \oplus b_2)x^2 + (a_1 \oplus b_1)x + (a_0 \oplus b_0)$

La multiplicación es alcanzada en dos pasos. El primer paso, el polinomio producto c(x) = a(x) • b(x), es algebraicamente expandido, y las potencia similares son agrupadas para dar:

$C(x) = c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0$

Donde:

$c_0 = a_0 \bullet b_0$ $c_4 = a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_3$

$c_1 = a_1 \bullet b_0 \oplus a_0 \bullet b_1$ $c_5 = a_3 \bullet b_2 \oplus a_2 \bullet b_3$

$c_2 = a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2$ $c_6 = a_3 \bullet b_3$

$c_3 = a_3 \bullet b_0 \oplus a_2 \bullet b_1 \oplus a_1 \bullet b_2 \oplus a_0 \bullet b_3$

el resultado, c(x), **no** representa una palabra de cuatro bytes. Por lo tanto el segundo paso de la multiplicación es reducir c(x) modulo un polinomial de grado 4; el resultado puede ser reducido a un polinomial de grado menor que 4. Para el algoritmo AES, éste es logrado con el polinomial $x^4 + 1$, tal que: $x^i \bmod (x^4 + 1) = x^{i \bmod 4}$

El producto modular de a(x) y b(x), denotado por a(x) ⊗ b(x), es dado por el polinomial de cuatro términos d(x), definido como sigue:

d(x) = $d_3x^3 + d_2x^2 + d_1x + d_0$

con:

$d_0 = (a_0 \bullet b_0) \oplus (a_3 \bullet b_1) \oplus (a_2 \bullet b_2) \oplus (a_1 \bullet b_3)$

$d_1 = (a_1 \bullet b_0) \oplus (a_0 \bullet b_1) \oplus (a_3 \bullet b_2) \oplus (a_2 \bullet b_3)$

$d_2 = (a_2 \bullet b_0) \oplus (a_1 \bullet b_1) \oplus (a_0 \bullet b_2) \oplus (a_3 \bullet b_3)$

$d_3 = (a_3 \bullet b_0) \oplus (a_2 \bullet b_1) \oplus (a_1 \bullet b_2) \oplus (a_0 \bullet b_3)$

Cuando a(x) es un polinomio fijo, la operación definida en d(x), puede ser escrita en forma matricial como sigue:

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \bullet \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Porque $x^4 + 1$, **no** es un polinomio irreducible sobre GF(2^8), la multiplicación por un polinomio de cuatro términos, no es necesariamente invertible. Sin embargo, el algoritmo AES, especifica un polinomio de cuatro términos, **que tiene una inversa**.

a(x) = $\{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$

a^{-1}(x) = $\{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$.

Otro polinomial usado dentro del algoritmo AES (ver la función RotWord()) , tiene:

$a_0 = a_1 = a_2 = \{00\}$ y $a_3 = \{01\}$, el cual es el polinomial x^3.

Inspeccionado la ecuación de multiplicación de matrices, debería mostrar que el efecto es formar la palabra de salida (output), rotando bytes dentro de la palabra de entrada (input). Esto significa que

$[b_0, b_1, b_2, b_3]$ es transformado dentro de $[b_1, b_2, b_3, b_0]$.

2.5.6 Especificaciones del Algoritmo

Para el algoritmo AES, la **longitud de el bloque de entrada (Input), el bloque de salida (output) y el estado (state) es de 128 bits**. Esto es representado por ***Nb*** = 4, que refleja el número

de palabras de 32 bits (número de columnas) dentro del estado (state).

Para el algoritmo AES, la **longitud de la clave de cifrado, *K*, es 128, 192 ó 256 bits.** La longitud de la clave es representada por ***Nk*** = 4, 6, ó 8, que refleja el número de palabras de 32 bits (número de columnas) dentro de la clave de cifrado.

Para el algoritmo AES, el número de rondas a ser representada durante la ejecución del algoritmo, es dependiente del tamaño de la clave. El número de rondas es representado por ***Nr***, donde ***Nr*** = 10, cuando ***Nk*** = 4, ***Nr*** = 12 cuando ***Nk*** = 6, y ***Nr*** = 14 cuando ***Nk*** = 8.

Solo las combinaciones de Ronda- bloque –clave que conforman este estándar están dadas en la siguiente tabla:

Tabla 4. Combinación de Rondas, bloques y claves

	long. De Clave (***Nk*** words)	Tamaño Bloque (***Nb*** words)	Nr. De Rondas (***Nr***)
AES-**128**	4	4	10
AES-**192**	6	4	12
AES-**256**	8	4	14

Para ambos, tanto como para el cifrador, como para el cifrador inverso, el algoritmo AES, usa una función ronda (round), que esta compuesta de cuatro (4) transformaciones diferentes orientadas al byte:

1) Sustitución del byte, usando una tabla de sustitución (S-box).
2) Desplazamiento de filas de el arreglo estado, por diferentes offsets.
3) Mezcla el dato dentro de cada columna del arreglo estado.
4) Adicionando una clave para la ronda (Round key) al estado.

Estas transformaciones (y sus inversas) son descritas seguidamente.

Cifrado

En el inicio del cifrado, la entrada (input) es copiada al arreglo estado (state) usando la convención descrita en la tabla 3. después de una adición inicial de clave para la ronda (Round key), el arreglo estado es transformado implementando la función ronda (round()) 10, 12 ó 14 veces (dependiendo de la longitud de la clave), con la ronda final ligeramente diferente de los primeros **Nr** –1 rondas. El estado final es después copiado a la salida (output), como se describió en la tabla 3.

La función ronda (round()), esta parametrizada usando un inventario de claves, que consiste de un arreglo de una dimensión de cuatro (4) palabras derivada del uso de la rutina de expansión de clave (Key Expansion routine), descrita en la siguiente sección.

El cifrado es descrito en seudo código en la siguiente figura. Las transformaciones individuales - **SubBytes(), ShiftRows(), MixColumns(), y AddRoundKey()** – procesan el estado y son descritas en la subsiguientes secciones. En la siguiente figura el arreglo **w[ ]**, contiene el inventario de claves.

Como se muestra en la siguiente figura, todas la ***Nr*** rondas son idénticas, con la excepción de la ronda final, el cual no incluye la transformación **MixColumns().**

(ver anexo: FIPS – Federal Information Processing Standards Publication 197 – Donde el apéndice B, presenta un ejemplo del cifrado, mostrando valores para el arreglo Estado, en el comienzo de cada ronda y después la aplicación de cada una de las cuatro transformaciones, que se describirán en las siguientes secciones.)

Figura 8. Seudo código del cifrado Rijndael

```
Cipher ( byte in[4*Nb], byte out[4*Nb], word w[Nb* (Nr+1)] )
Begin
   Byte state[4, Nb]

   State = in

   AddRoundKey ( state, w[0, Nb-1] )

   For round = 1    step 1    to Nr-1
         SubBytes (state)
         ShiftRows (state)
         MixColumns (state)
         AddRoundKey (state, w[ round*Nb,  (round+1) * Nb-1 ] )
   End for

   SubBytes (state)
   ShiftRows (state)
   AddRoundKey (state,  w [ Nr * Nb,  (Nr-1) * Nb - 1 ] )

   Out = state
End
```

Transformación SubBytes ()

La transformación SubBytes(), es una sustitución no lineal que opera independientemente sobre cada byte de el estado, usando una tabla de sustitución (S-box). Esta caja S-box, que es invertible, es construido por composición de dos (2) transformaciones:

1) Toma la inversa multiplicativa dentro del campo finito GF(2^8), descrito anteriormente; el elemento {00} es mapeado a si mismo.
2) Aplica la siguiente transformación affine (sobre GF(2)):

$$b'_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i$$

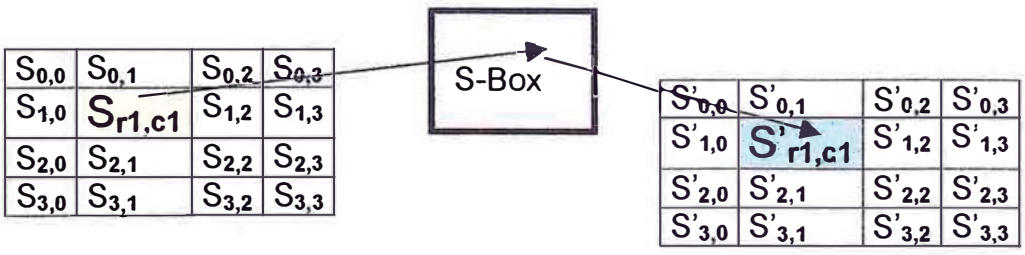
Para $0 \leq i < 8$, donde b_i es el i –esimo bit de el byte, y c_i, es el i –esimo bit de un byte c con el valor {63} ó {01100011}. Aquí y en cualquier otra parte, un apostrofe sobre una variable (ejemplo b'), indica que la variable, esta para ser actualizable con el valor de la derecha.

En forma matricial, el elemento de la transformación affine de la caja S-box, puede ser expresada como:

$$\begin{pmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \bullet \begin{pmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \\ B_6 \\ b_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

La siguiente figura, ilustra el efecto de la transformación SubBytes() sobre el estado.

Figura 9. SubBytes(), aplica la cja S-box a cad abytes del estado



La caja S-box usada dentro de la transformación SubBytes(), es presentada en forma Hexadecimal.

Por ejemplo, Si $s_{1,1}$ **={ 53 },** entonces el valor de la substitución debería estar determinada por la intersección de la fila '5' y la

columna con índice '3' en la siguiente figura. Esto debería resultar en $s'_{1,1}$, obteniendo el valor de {ed}.

		y														
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5ª	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	f	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	A	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	B	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	C	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	D	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	E	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	F	8c	a1	89	0d	bf	e6	42	68	41	99	2d	of	b0	54	bb	16

Transformación ShiftRows()

Dentro de la transformación ShiftRows(), los bytes dentro de las tres (3) últimas filas del estado son desplazados cíclicamente sobre diferentes números de bytes **(offset),** La primera fila, r = 0, no es desplazada.

Específicamente, la transformación ShiftRows() procede como sigue:

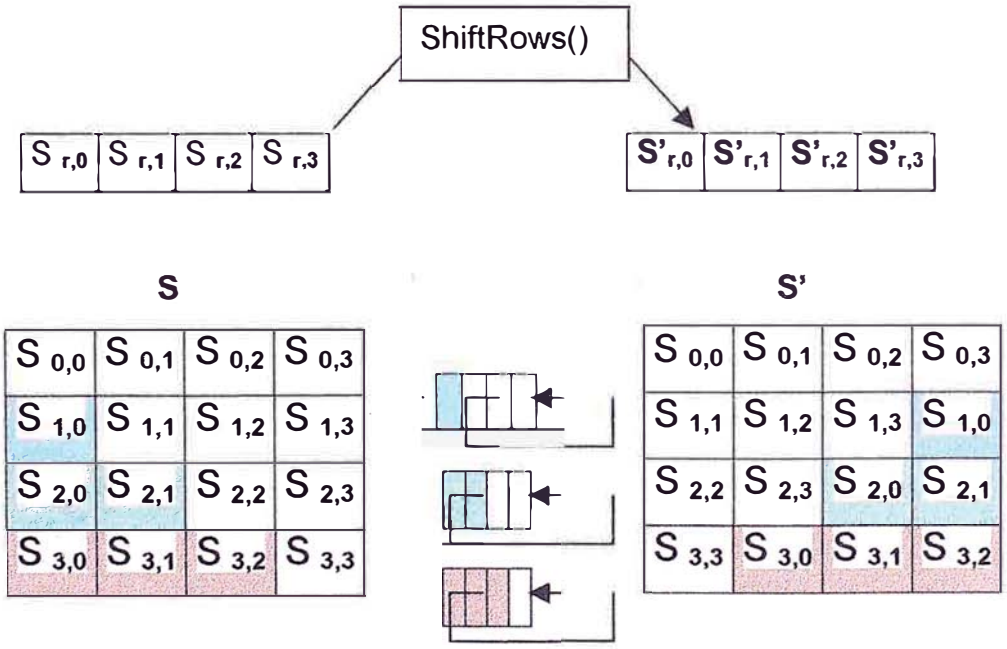
$S'_{r,c} = S_{r,(c + shift(r, Nb)) \bmod Nb}$ Para $0 < \mathbf{r} < 4$ y $0 \leq \mathbf{c} < NB$,

Donde el valor shift (r, *Nb*) depende del número de fila, **r**, como sigue (acordarse de que Nb = 4):

Shift (1,4) = 1; shift (2,4) = 2; shift (3,4) = 3 .

Esto tiene el efecto de mover bytes, a posiciones bajas dentro de la fila (ejemplo, valores inferiores de c dentro de una fila dada), mientras que los bytes más bajos se envuelven en la parte superior de la fila (ejemplo, valores altos de c dentro de una fila dada). Ver siguiente figura:

Figura 10. Transformación ShiftRows() de Rijndael



Transformación MixColumns()

La transformación MixColumns(), opera sobre el estado columna por columna, tratando cada columna como un polinomial de cuarto término, descrito anteriormente. Las columnas son consideradas como polinomial sobre GF(2^8) y multiplicado por modulo $x^4 + 1$ con un polinomio fijo a(x), dado por : $a(x) = \{03\} x^3 + \{01\} x^2 + \{01\} x + \{02\}$.

Como fue descrito anteriormente, esto puede ser escrito como una multiplicación de matrices. Veamos $s'(x) = a(x) \otimes s(x)$:

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \bullet \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

Como resultado de esta multiplicación, los cuatro bytes dentro de una columna son reemplazados por lo siguiente:

$S'_{0,c} = (\{02\} \bullet S_{0,c}) \oplus (\{03\} \bullet S_{1,c}) \oplus S_{2,c} \oplus S_{3,c}$

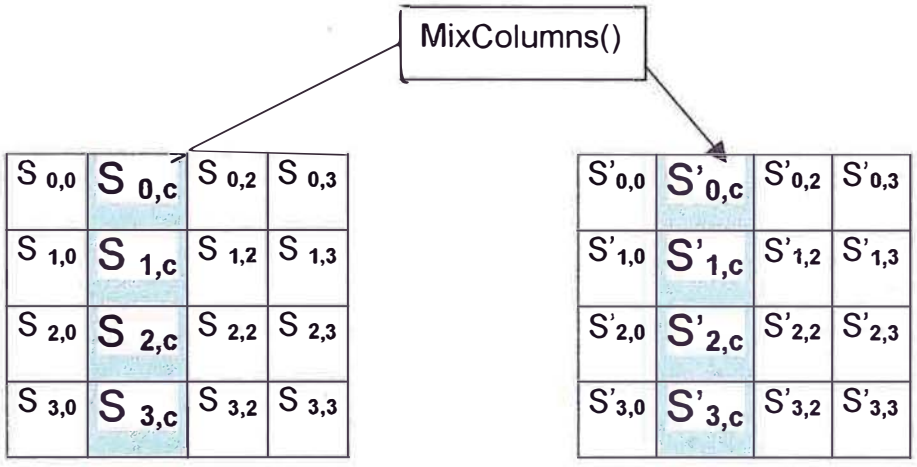
$S'_{1,c} = S_{0,c} \oplus (\{02\} \bullet S_{1,c}) \oplus (\{03\} \bullet S_{2,c}) \oplus S_{3,c}$

$S'_{2,c} = S_{0,c} \oplus S_{1,c} \oplus (\{02\} \bullet S_{2,c}) \oplus (\{03\} \bullet S_{3,c})$

$S'_{3,c} = (\{03\} \bullet S_{0,c}) \oplus S_{1,c} \oplus S_{2,c} \oplus (\{02\} \bullet S_{3,c})$

La siguiente figura ilustra la transformación MixColumns().

Figura 11. MixColumns() opera sobre el estado columna por columna en rijndael



Transformación AddRoundKey()

Dentro de la transformación AddRoundKey(), una clave a la ronda es adicionada al estado por una simple operación de bits XOR. Cada clave para la ronda consiste de ***Nb*** palabras (words) del inventario de claves (descrita en la sección siguiente). Estas

palabras ***Nb*** (words) son cada una adicionadas dentro de la columna de el estado, tal que:

$[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}] = [s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}] \oplus [w_{round * Nb + c}]$;

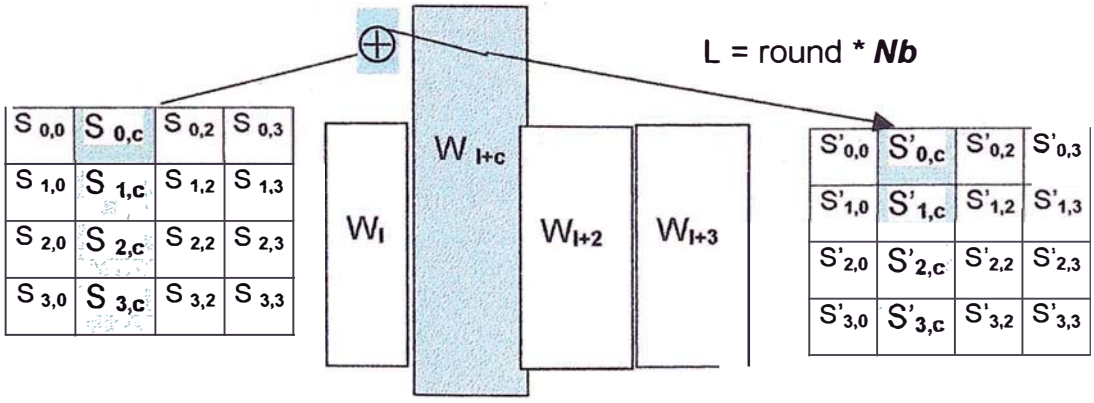
para $0 \le c < Nb$,

donde $[w_i]$ son las palabras del inventario de claves, descritas en la sección siguiente, y ronda (round) es el valor dentro del rango $0 \le round \le Nr$. Dentro del cifrado, la inicial adición de clave para la ronda ocurre cuando la ronda round = 0, anterior a la primera aplicación de la función round (ver figura 8). La aplicación de la transformación AddRoundKey() a las ***Nr*** rondas de el cifrado ocurre cuando $1 \le round \le Nr$.

La acción de esta transformación es ilustrada en la siguiente figura, donde

l = round * ***Nb***. La dirección del byte dentro de las palabras del inventario de clave fue descrita anteriormente en notaciones y convenciones - input y output.

Figura 12. AddRoundKey() XORs de cada columna del estado con una palabra del inventario de claves.



Expansión de la clave

El algoritmo AES toma la clave de cifrado, ***K***, y ejecuta una rutina de expansión de clave, para generar un inventario de claves. La expansión de claves genera un total de ***Nb*** * (***Nr*** + 1) palabras: el algoritmo requiere de un inicial conjunto de ***Nb*** palabras (words), y cada uno de las ***Nr*** rondas requiere ***Nb*** palabras (words) de data de la clave. El resultante del inventario de claves consiste de un arreglo lineal de palabras de 4-bytes, denotados por $[w_i]$, con i dentro del rango $0 \leq i < Nb * (Nr + 1)$.

La expansión de la clave de entrada dentro del inventario de claves procede de acuerdo al seudo código presentado en la siguiente figura:

SubWord () es una función que toma una palabra de entrada de 4-bytes y aplica la caja S-box (de la sección anterior), a cada uno de los cuatro (49 bytes, para producir una palabra de salida. La función **RotWord()** toma una palabra $[a_0,a_1,a_2,a_3]$ como entrada, ejecuta una permutación cíclica, y retorna la palabra$[a_1,a_2,a_3\ a_0]$. La constante ronda, que es un arreglo de palabras, **Rcon [i]**, contiene los valores dados por [x^{i-1}, {00},{00},{00}], con x^{i-1} siendo potencia de x (x es denotado como {02} dentro del campo $GF(2^8)$, note que i inicia en 1 y no en 0).

Figura 13. Seudo código para la expansión de claves en rijndael

```
KeyExpansion ( byte Key[ 4*Nk ] , word w [ Nb * (Nr + 1) ] , Nk )
Begin
   word    temp

   i = 0

   while ( i < Nk )
      w [ i ] = word( key[ 4* i ], key[ 4* i + 1], key[ 4* i + 2 ], key[ 4* i + 3] )
      i = i + 1
   end while

   i = Nk

   while ( i < Nb * (Nr + 1) )
      temp = w [ i –1]
      if ( i mod Nk = 0 )
         temp = SubWord ( RotWord (temp) ) xor Rcon [ i / Nk ]
      else if  (Nk > 6 and  i mod Nk = 4 )
               temp = SubWord (temp)
      end if
      w [ i ] = w [ i – Nk ]  xor  temp
      i = i + 1
   end while
end
```

Note que Nk = 4, 6 y 8, no todos tienen que ser implementados; ellos están todos incluidos dentro de la sentencia condicional breve y conciso. Para especificas implementaciones requeridos por la clave de cifrado, son presentados en la siguientes secciones.

ver anexo: FIPS – Federal Information Processing Standards Publication 197 – Donde el apéndice B, presenta un ejemplo de expansión de claves.

Cifrado Inversa

La transformación Cifrado vista en la sección anterior, puede ser invertida y después implementada en orden reverso para producir

un directo cifrado inverso, para el algoritmo AES. Las transformaciones individuales usados dentro del cifrado inverso son **InvShiftRows(), InvSubBytes(), InvMixColumns() y AddRoundKey()**, que procesa el estado y son descritos en las subsiguientes secciones.

El Cifrado Inverso es descrito en seudo código dentro de la siguiente figura:

El arreglo **w[ ]**, contiene el inventario de claves, el cual fue descrito anteriormente:

Figura 14. Seudo código para el Cifrado Inverso de Rijndael.

```
InvCipher ( byte in[4*Nb], byte out[4*Nb], word w[Nb* (Nr+1)] )
Begin
   Byte state[4, Nb]

   State = in

   AddRoundKey ( state, w [Nr * Nb,  (Nr+1) * Nb-1] )

   For round = Nr - 1 step  -1  downto 1
         InvShiftRows (state)
         InvSubBytes (state)
         AddRoundKey (state,  w[ round*Nb,  (round+1) * Nb-1 ] )
         InvMixColumns (state)
   End for

   InvShiftRows (state)
   InvSubBytes (state)
   AddRoundKey (state,  w [ Nr * Nb,  (Nr-1) * Nb - 1 ] )

   Out = state
End
```

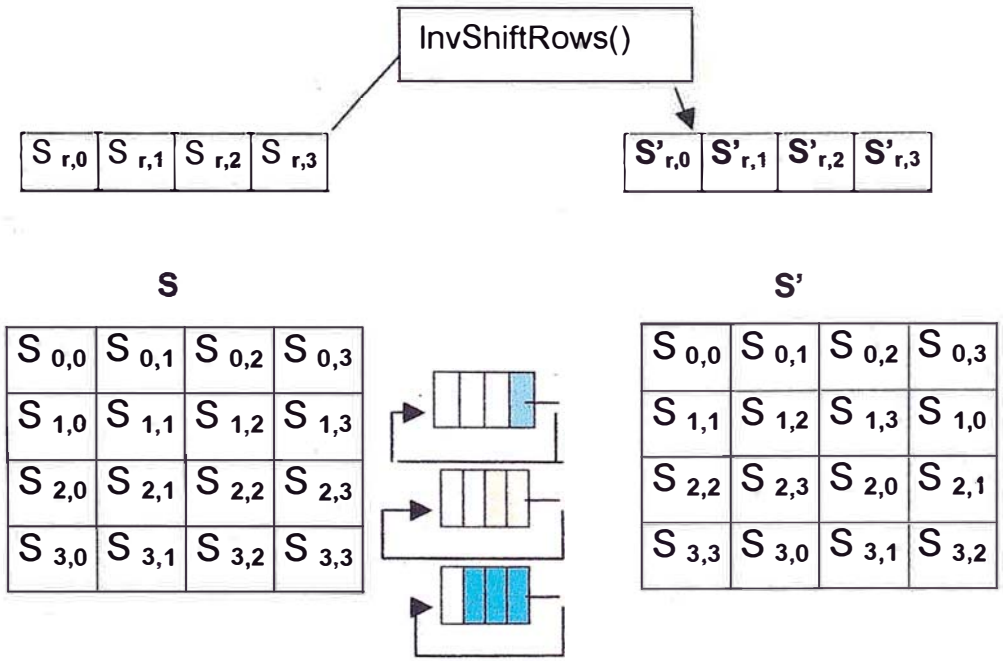
Transformación InvShiftRows()

InvShiftRows(), es la inversa de la transformación ShiftRows(), los bytes dentro de las tres (3) últimas filas de el estado son cíclicamente desplazados sobre diferentes números de bytes **(offset),** La primera fila, r=0, no es desplazada. Las tres filas de abajo son cíclicamente desplazada por Nb – shift(r, *Nb*) bytes, donde el valor de desplazamiento shift(r, *Nb*) depende del número de fila, y está dado en la ecuación de transformación ShiftRows(). Específicamente, la transformación InvShiftRows() procede como sigue:

$$S'_{r,(c+shift(r,Nb))\bmod Nb} = S'_{r,c} \quad \text{Para } 0 < r < 4 \text{ y } 0 \leq c < NB,$$

La siguiente figura, ilustra la transformación InvShiftRows().

Figura 15. Transformación ShiftRows() de Rijndael



Transformación InvSubBytes ()

InvSubBytes(), es la inversa de la transformación del byte, dentro del cual, la caja inversa S-box es aplicada a cada byte del estado. Esto es obtenido aplicando la inversa de la transformación affine, seguido de tomar la multiplicativa inversa dentro de GF(2^8).

La inversa S-box usada dentro de la transformación InvSubBytes(), es presentada en la siguiente figura:

Figura 16. Caja inversa S-box: valores de sustitución para el byte xy (en formato hexadecimal).

		y														
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	52	09	6ª	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3ª	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	A	47	f1	1ª	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	B	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	C	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	D	60	51	7f	a9	19	b5	4a	0d	2d	e5	7ª	9f	93	c9	9c	ef
	E	a0	e0	3b	4d	ae	2ª	f5	b0	c8	eb	bb	3c	83	53	99	61
	F	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Transformación InvMixColumns()

InvMixColumns(), es la inversa de la transformación MixColumns().

InvMixColumns(), opera sobre el estado columna por columna, tratando cada columna como un polinomial de cuatro termino, descrito anteriormente. Las columnas son consideradas como polinomial sobre GF(2^8) y multiplicado por modulo $x^4 + 1$, con un polinomio fijo a(x), dado por: $a^{-1}(x) = \{0b\}\ x^3 + \{0d\}\ x^2 + \{09\}\ x + \{0e\}$.

Como fue descrito anteriormente, esto puede ser escrito como una multiplicación de matrices. Veamos $s'(x) = a^{-1}(x) \otimes s(x)$:

Para $0 \leq c < Nb$

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \bullet \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

Como resultado de esta multiplicación, los cuatro bytes dentro de una columna son reemplazados por lo siguiente:

$S'_{0,c} = (\{0e\} \bullet S_{0,c}) \oplus (\{0b\} \bullet S_{1,c}) \oplus (\{0d\} \bullet S_{2,c}) \oplus (\{09\} \bullet S_{3,c})$

$S'_{1,c} = (\{09\} \bullet S_{0,c}) \oplus (\{0e\} \bullet S_{1,c}) \oplus (\{0b\} \bullet S_{2,c}) \oplus (\{0d\} \bullet S_{3,c})$

$S'_{2,c} = (\{0d\} \bullet S_{0,c}) \oplus (\{09\} \bullet S_{1,c}) \oplus (\{0e\} \bullet S_{2,c}) \oplus (\{0b\} \bullet S_{3,c})$

$S'_{3,c} = (\{0b\} \bullet S_{0,c}) \oplus (\{0d\} \bullet S_{1,c}) \oplus (\{09\} \bullet S_{2,c}) \oplus (\{0e\} \bullet S_{3,c})$

Inversa de la transformación AddRoundKey( )

El AddRoundKey( ), que fue descrito en la sección anterior, es su propia inversa, desde que eso solo involucra una aplicación de la operación XOR.

Equivalente del Cifrado Inverso

Dentro del cifrado inverso directo presentado anteriormente, la secuencia de las transformaciones difiere de la del Cifrado, mientras la forma del inventario de claves para el cifrado y descifrado queda igual. Sin embargo, varias propiedades del algoritmo AES, permite para un equivalente cifrado inverso que tiene la misma secuencia de transformación como el cifrado (con la transformaciones reemplazadas por sus inversas). Esto es alcanzado con un cambio dentro del inventario de claves.

Las dos propiedades que permite un equivalente de cifrado inversa son los siguientes:

- Las transformaciones SubBytes() y ShiftRows() se conmuta; esto es, una transformación SubBytes() inmediatamente seguido por una transformación ShiftRows(), es equivalente a una transformación ShiftRows() inmediatamente seguido por una transformación SubBytes(). Lo mismo es verdadero para sus inversas, InvSubBytes() y InvShiftRows().
- La operación de mezcla de columnas MixColumns() y InvMixColumns(), son lineales con respecto a la columna input, lo cual significa

  InvMixColumns (state XOR Round Key) = InvMixColumns (state) XOR InvMixColumns (Round Key) .

Estas propiedades permiten el ordenamiento de las transformaciones InvSubBytes() y InvShiftRows(), para ser reversas. El orden de la transformaciones AddRoundKey() y InvMixColumns(), pueden también ser reversas, con tal que las columnas (words) del inventario de claves de desencriptacion son modificadas usando la transformación InvMixColumns().

El equivalente de la inversa de cifrado es definido, cambiando el orden de las transformaciones InvSubBytes() y InvShiftRows(), mostrada en la siguiente figura, y cambiando el orden de las transformaciones AddRoundKey() y InvMixColumns() usado dentro de la ronda "round loop" después, primero modificar el inventario de claves de desencriptacion para round = 1 a *Nr* –1, usando la transformación InvMixColumns(). La primera y la última palabra Nb del inventario de claves de desencriptacion, no debería ser modificado de esta manera.

Dados estos cambios, el resultante equivalente de la inversa de cifrado, ofrece una estructura más eficiente que el cifrado inverso descrito anteriormente.

El seudo código para el equivalente del cifrado inverso aparece en la siguiente figura. (el arreglo palabra (Word) **dw [ ]** contiene el modificado inventario de claves para la desencriptacion. La modificación a la rutina de expansión de claves está también prevista en la siguiente figura.

Figura 17. Seudo código, para el equivalente del cifrado inverso.

```
EqInvCipher ( byte in[4*Nb], byte out[4*Nb], word dw[Nb* (Nr+1)] )
Begin
   Byte state[4, Nb]

   State = in

   AddRoundKey ( state, dw [Nr * Nb, (Nr+1) * Nb-1] )

   For round = Nr - 1 step -1 downto 1
         InvSubBytes (state)
         InvShiftRows (state)
         InvMixColumns (state)
         AddRoundKey (state, dw[ round * Nb, (round+1) * Nb -1 ] )
   End for

   InvSubBytes (state)
   InvShiftRows (state)
   AddRoundKey (state, dw [ 0, Nb - 1 ] )

   Out = state
End
```

Note que para el equivalente del cifrado inverso, el siguiente seudo código es adicionado al final de la rutina de expansión de claves

```
   For i = 0    step 1  to (Nr + 1) * Nb – 1
      dw [i] = w [i]
   end for

   for round = 1 step 1 to Nr – 1
      InvMixColumns ( dw [round * Nb,  (round + 1) * Nb -- 1 ] )
      // note el cambio de tipo
   end for
```

Note que, solo InvMixColumns opera sobre un arreglo dimensional de bytes, mientras que las claves por ronda son held dentro de un arreglo de palabras, el llama a InvMixColumns dentro de este código de secuencia que involucra un cambio de tipo (ejemplo, el input a InvMixColumns() es normalmente el arreglo estado, dentro del cual esta considerado ser un arreglo de bytes de dos dimensiones, donde el input aquí es un clave para la ronda, calculado como un arreglo de palabras de una dimensión).

2.6 Sistemas Asimétricos

Cada usuario dispone de dos (2) claves, una privada y otra publica (K_p, K_P), , de tal forma que, con una cifra y con la otra descifra y viceversa. Lo importante de este proceso es que la clave privada sólo la conoce el usuario propietario de ella, y es la clave pública, la que se distribuye para que el resto del mundo. Lo utilice para enviarnos mensaje. Si alguien capta el mensaje no podrá descifrarlo, ya que sólo se descifra con la clave privada. Esto ofrece un abanico de posibilidades, pudiendo emplearse para establecer comunicaciones seguras, sobre canales de comunicación inseguros, puesto que únicamente viaja la clave pública, que solo sirve para cifrar. Algunos algoritmos de clave pública son: RSA, DSS, Diffie & Hellman.

En la práctica, lo que se emplea es una combinación de estos dos tipos de criptosistemas, puesto que los de clave asimétrica, presentan el inconveniente de ser computacionalmente costosos y lentos que los de clave simétrica. Lo que se hace en el mundo real es codificar los mensajes (largos) mediante algoritmos simétricos, que suelen ser muy eficientes y rápidos, y luego emplear criptografía asimétrica, para codificar solo la clave, que suele ser corta en comparación con los mensajes.

2.6.1 Introducción a RSA

En los sistemas tradicionales de cifrado, debe comunicarse una clave entre el emisor y el receptor del mensaje. El problema aquí es encontrar un canal seguro para transmitir dicha clave. Este problema viene a resolverse en los sistemas de clave pública, haciendo público la clave de cifrado, pues un tiempo enormemente grande de ordenador, es necesario para encontrar una transformación de descifrado a partir de la del cifrado.

RSA, es un sistema de clave pública, basado en la exponenciación modular, donde las claves son pares de

números (**e**, **n**), formados por un exponente **e** y un **módulo n**, que es el producto de dos grandes números primos p y q, tales que el **mcd (**e, fi(n) **)** = 1, (donde fi(n) es el número de enteros, menores que n y primos con él).

Para cifrar un mensaje convertimos las letras que lo forman, en sus equivalentes numéricos y formamos bloques del mayor tamaño posible, con un número par de dígitos. Entonces el cifrado del bloque P, sería

(P) = C = P^{e} **(mod n),** donde C, es el mensaje cifrado. El procedimiento de descifrado, requiere el conocimiento de un inverso **d**, de **e mod fi(n)**. Entonces D(C) = P = C^{d}**(mod n),** el par (d, n), es la clave de descifrado.

Desarrollemos un ejemplo donde n = 2537 = 43 * 59 (números primos muchísimos m'as pequeños de los que usaríamos realmente). Tomemos **e** = 13 como exponente, observar que es v'alido, pues el **mcd (** e, fi(n) **)** = **mcd (**13, 42*58 **)** = 1. Para cifrar la Frase "PUBLIC KEY CRIPTOGRAPHY", obtenemos su equivalentes numéricos1520 0111 08021004 2402 1724 1519 1406 1700 1507 2423 (hemos usado A = 00, B = 01, ...), donde hemos añadido una X(23), para rellenar el ultimo bloque. Ahora transformamos bloque a bloque, usando

C = P^{13} **(mod 2537)** , con el primer C = 1520^{13} (mod 2537) = 95 y el resultado final seria: 0095 1648 1410 1299 0811 2333 2132 0370 1185 1457 1084.

Para recuperar el texto original, necesitamos un inverso de 13 mod fi(2537) = 42 * 58 = 2436. Un cálculo bastante breve muestra que d = 937, es el inverso, por lo que el descifrado sería P = C^{937} (mod 2537). Por ejemplo, con el primer bloque 0095^{937} (mod 2537) = 1520 y así procederemos con todos los bloques que forman el texto cifrado.

Para entender por qué RSA, constituye un sistema útil de clave pública, notemos en primer lugar que cualquier individuo puede encontrar dos grandes números primos **p** y **q**, con por ejemplo 100 dígitos, en poco segundos de computador. Se pueden elegir simplemente enteros impares de 100 dígitos aleatoriamente y el teorema de distribución de los números primos, nos asegura que la probabilidad de que dicho número sea primo es aproximadamente de 2/(log 10 100), lo que quiere decir, uno de cada 115 en promedio. Un test de tipo probabilística, nos permitirá descartar los compuestos muy rápidamente.

Una vez elegidos los primos **p** y **q**, necesitamos un exponente **e**, que verifique mcd (e, fi(p * q)) = 1, como se explico anteriormente.

Una forma posible es elegir un primo **e** mayor que **p** y **q**

De cualquier modo debe cumplirse que $2^{e} > n = p * q$, para que sea imposible obtener el texto legible P, calculando simplemente la raíz de índice **e** del texto cifrado $C = (P^{e} \pmod n)$. Con la condición antes citada cualquier P (excepto 0 y 1), sufrirá una reducción módulo n.

La exponenciación modular, es también rapidísima, incluso cuando el módulo, el exponente y la base tienen 200 dígitos. Con el algoritmo de Euclides, encontramos rápidamente el inverso **d** del exponente **e módulo fi(n)**, cuando **p** y **q**, son conocidos, pues en este caso

fi(**n**) = fi(**p** *q) = (**p**-1)(**q**-1).

Por el contrario, el conocimiento de la clave de cifrado (**e**, **n**), no conduce, al de la clave de descifrado. En efecto, para hallar **d**, un inverso de **e modulo fi(n)**, necesitamos encontrar fi(n) y esto No es más sencillo que factorizar **n**, lo que parece ser un problema intrínsicamente costoso computacionalmente hablando.

Aunque no se ha probado que sea imposible romper un cifrado RSA sin factorizar n, aun nadie ha descubierto una alternativa, pues los métodos existente son en general equivalentes en complejidad a factorizar un número entero.

Unas precauciones extras, para escoger **p** y **q**, evitando así métodos de factorización especiales:

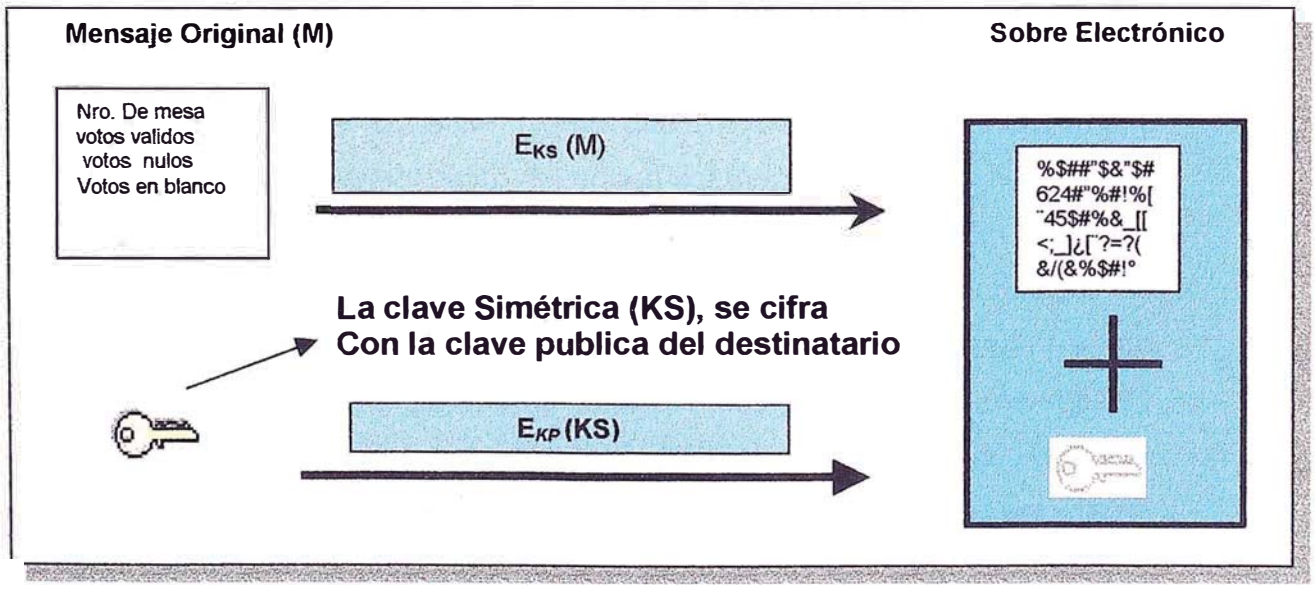
- **p-1** y **q-1**, deben tener grandes factores primos.
- **mcd**(**p-1**, **q-1**) debe ser "pequeño".
- **p** y **q**, deberían tener un número similar de dígitos.

Los sistemas de clave pública, se utilizan también para firmar mensajes. El uso de firmas digitales hace que el receptor de un mensaje, esté seguro de que realmente partió del emisor anunciado. Se utilizan mucho en correo electrónico y en transacciones bancarias.

2.7 Sobres electrónicos.

Este concepto se utiliza para darle confidencialidad a los mensajes y está soportada por el uso de algoritmos simétricos, para encriptar el contenido del mensaje. El proceso es el siguiente: primero se cifra el archivo o mensaje original (M), con un algoritmo simétrico, que utiliza una sola clave simétrica (**KS**), tanto para cifrar como para descifrar. Luego la clave simétrica (**KS**), es cifrada con la clave pública (***KP***) del par de claves asimétricas del destinatario (como se puede apreciar en este punto se esta utilizando el algoritmo asimétrico del destinatario, que tienen dos claves una pública y otra privada). La unión de la **clave simétrica cifrada**, junto con los **datos del mensaje cifrados** con esta, se conoce como sobre electrónico (digital envelope). Ver figura.

Figura 18: Sobre Electrónico



A su recepción, **el destinatario** utiliza su clave privada (***Kp***), de su par de claves asimétricas, para descifrar la clave simétrica (**KS**), que a su vez permitirá descifrar los datos del mensaje.

La generación de las claves simétricas aleatorias es un proceso de gran importancia. La programación y métodos empleados para ello, deben

garantizar que tales claves, no serán inferidas del contenido del mensaje, ni del entorno en el que se ha producido.

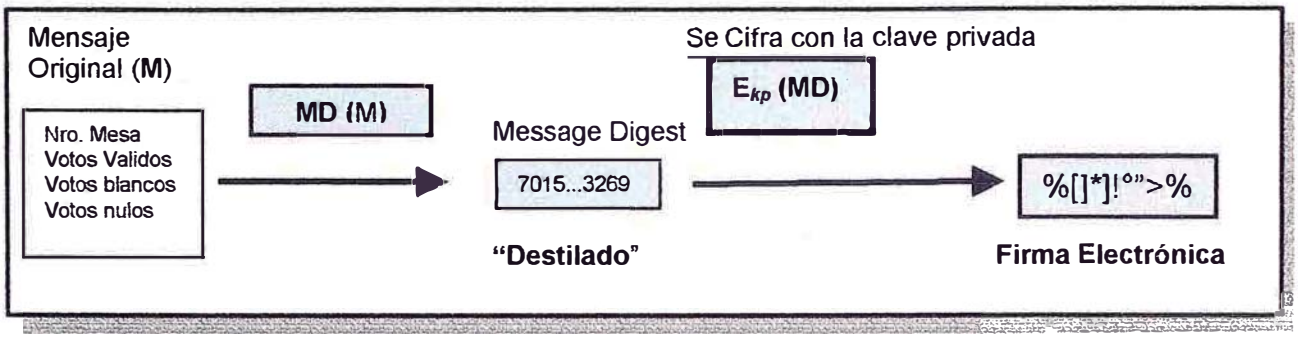
2.8 Firma Digital

Este concepto se utiliza para garantizar la integridad y autenticidad de los mensajes. La integridad, es la garantía de que el contenido de los mensajes no ha sido alterado de forma fraudulenta. La autenticidad de los mensajes, garantiza que las partes intervinientes en el proceso de transmisión de datos lo hacer de forma legal y representan quienes dicen ser. Estos se basan en la generación de firmas electrónicas (digital signature).

La firma electrónica, está sustentada en las relaciones matemáticas, entre la clave pública (***KP***) y la clave privada (***Kp***) del algoritmo asimétrico utilizado. Así, un mensaje cifrado con una de las claves, sólo puede ser descifrado, con la otra clave. El Remitente de un mensaje, cifra su contenido con su propia clave privada (***Kp***); el Destinatario puede descifrarlo, con la correspondiente clave pública (***KP***) del remitente y determinar así la autenticidad del origen del mensaje.

Dentro del proceso para garantizar la integridad del contenido del mensaje y al mismo tiempo acelerar el tratamiento del mismo, se incorpora otro proceso adicional que consiste en **generar un valor único** y **representativo de los datos**. Este proceso, denominado literalmente como **Digestión del mensaje (message digest),** consiste en hacer pasar los datos a través de una función irreversible (**one–way hash function**), como MD5 o SHA-1, que produce un destilado del mensaje original, que es único, para un contenido dado. Es computacionalmente imposible producir el mismo destilado, a partir de dos mensajes diferentes. El algoritmo SHA-1, produce un destilado de 160 bits (20 dígitos) y es tal que el cambio de un solo BIT en el mensaje original, produce en promedio, el cambio de la mitad de los bits del producto. Ver figura.

Figura 19: Firma Digital



El "**destilado**" del mensaje, se cifra ahora con la clave privada del remitente ***Kp,*** y el resultado se añade al mensaje original que se envía, constituyendo la firma electrónica del mismo, es decir del mensaje.

El Destinatario del mensaje descifra el "**destilado**" con la clave pública del remitente ***KP,*** aplica la misma función hash al mensaje original y compara ambos resultados. Si son iguales, la integridad y autenticidad del mensaje son correctas. Si el proceso de descifrado, no es satisfactorio, el remitente no puede ser autenticado; si el "**destilado**" generado, no es coincidente con el extraído de la firma electrónica, se ha producido una modificación en el contenido del mensaje.

2.9 Función Hash

Se puede definir la función hash (función de comprobación aleatoria), como aquella que reduce el mensaje a un conjunto de datos, denominado "**resumen**" o "**destilado**", de longitud mucho menor que el mensaje (usualmente a 128, 160 o 254 bits). Un valor hash es generado por una función de la forma **h = H(M)**, donde **M**: es el mensaje de longitud variable, y **H(M)**, es el valor Hash de longitud fija. El valor hash, es añadido al mensaje en el emisor, cundo se conoce que su valor es correcto. El receptor autentifica el mensaje, volviendo a generar el valor hash. Como una función Hash no está protegida o considerada como secreta es necesaria cifrarla.

El propósito de una función hash, es producir una "**huella digital**" de un mensaje, o un bloque de datos. Para poder ser utilizada en la autentificación y en el proceso de firmas digitales, una función Hash, debe poseer las siguientes propiedades:

- H, puede ser aplicada a un bloque de datos de cualquier longitud.
- H, produce una salida de longitud fija.
- H(M), es relativamente fácil de computar dado M.
- Dado un "código" o "destilado" x, es computacionalmente in factible encontrar M de la forma H(M)=x.
- Dado un bloque cualquiera M, es computacionalmente in factible encontrar un Y, diferente de M con H(Y)=H(M).
- Es computacionalmente in factible, encontrar un par (x, y), tal que H(x)=H(y).

Las tres (3) primeras propiedades son requeridas para poner en práctica sistemas de autentificación de mensajes en aplicaciones de comunicaciones.

La cuarta propiedad es "one-way": es decir, es fácil generar un código, dado un mensaje, pero es virtualmente imposible generar el mensaje, dado el código.

La quinta propiedad, garantiza que la realización de una función Hash, a un mensaje alternativo, no puede ofrecer el mismo valor que el mensaje original.

La función Hash, que cumpla las cinco (5) primeras propiedades, es una función débil. La función hash, que además cumpla con la sexta propiedad, se denomina una función Hash fuerte.

Entre los principales algoritmos resumen tenemos:

MD5 (Message Digest 5). Creado por Ron Rivest. Hasta hace poco se consideraba un algoritmo resumen seguro, hasta tal punto que es 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 Hans Dobbertin que quebró MD4, consiguió demostrar que la función de comprensión 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 idéntico "resume" o "destilado" para dos mensajes obtenidos al azar, pero queda por demostrar que se pueda encontrar un mensaje con igual resumen a otro mensaje concreto; es decir, el resumen del mensaje que yo he firmado sigue siendo único. Con todo, resulta inquietante que un algoritmo de resumen, considerado seguro tenga tales puntos débiles. A menos que se le refuerce, su uso ira decreciendo en el futuro (Ver anexo).

2.10 Secure Hash Algorithm (SHA-1)

Fue desarrollado como estándar de resumen seguro (Secure Hash Standard, SHS) y el estándar de cifrado digital (Digital Signature Standard, DSS) por la Agencia de Seguridad Nacional Norteamericana (NSA). A pesar de su origen, parece un resumen seguro y sin fisuras, al menos por ahora. La primera versión, conocida como SHA, fue mejorada como protección a un tipo de ataque que nunca fue revelado. El documento FIPS 180-1(Federal Información Processing Standard) que oficialmente lo describe, afirma que los principios subyacentes al SHA-1, son similares a los del MD4 de Rivest. (Esperemos que no en lo referente a la vulnerabilidad). Su implementación puede estar cubierta por patentes en EU y el extranjero. A falta de ataques ulteriores, se le puede considerar seguro. (ver anexo).

2.10.1 Introducción a SHA-1

Este estándar especifica un algoritmo hash seguro, SHA-1, para calcular una representación condensada de un mensaje o un archivo de datos.

Cuando un mensaje de cualquier longitud $< 2^{64}$ bits, como entrada (input), el SHA-1, produce una salida (output) de 160-bits llamado digestión del mensaje o resumen (message digest). El resumen o destilado, puede después ser entrada al algoritmo de firma digital (DSA), que genera o verifica la firma de un mensaje. Firmando el resumen o destilado, más que el mensaje, frecuentemente mejora la eficiencia del proceso porque el resumen es usualmente más pequeño en tamaño que el mensaje. El mismo algoritmo hash debería ser usado por el verificador de una firma digital, como fue usado por el creador de la firma digital.

La función hash SHA-1 es llamada segura, y fue diseñado para que sea computacionalmente imposible encontrar un mensaje que corresponda a un resumen dado, o para encontrar dos mensaje diferentes, el cual produzca el mismo resumen. Cualquier cambio a un mensaje en tránsito debería con muy alta probabilidad, resultar un resumen diferente y la firma debería fallar al verificar. El SHA-1, es una revisión técnica del SHA (FIPS 180). Una operación de desplazamiento circular a la izquierda, tuvo que ser adicionada a las especificaciones.

2.10.2 Cadena de bits y enteros (BIT Strings and Integers)

La siguiente terminología relacionada a cadena de bits y enteros debería ser usada:

- Un dígito hexadecimal es un elemento de el conjunto de {0, 1, 2, ..., 9, A, B, C, D, E, F}. Un digito hexadecimal es la representación de una cadena de 4-bits.
  Ejemplo: 7 = 0111, A = 1010.
- Una palabra (Word) es igual a una cadena de 32 bits, que puede ser representada por una secuencia de 8 dígitos hexadecimales.
  Para convertir una palabra a 8 dígitos hexadecimales, cada cadena de 4 bits, es convertida a su equivalente hexadecimal como sigue: ejemplo:
  1010 0001 0000 0011 1111 1110 0010 0011 = A103FE23
- Un entero entre 0 y 2^{32} –1 inclusive, puede ser representado como una palabra (Word). Los cuatro bits menos significativos del entero son representados por el dígito hexadecimal más a la derecha de la palabra representada. Ejemplo:
  el entero 291 = $2^8 + 2^5 + 2^1+2^0$ = 256+32+2+1 es representado por la palabra hex., 00000123.
  Si z es un entero, $0 \leq z < 2^{64}$, entonces $z = 2^{32}x + y$ donde
  $0 \leq x < 2^{32}$ y $0 \leq y < 2^{32}$. desde que x e y, pueden ser representados como palabras X e Y, respectivamente, z puede ser representado como un par de palabras (X, Y).
- Block = 512 bits (cadena). Un block puede ser representado como una secuencia de 16 palabras.

2.10.3 Operaciones sobre palabras

Las siguientes operaciones lógicas deberían ser aplicadas a las palabras:

Operación lógica entre bits de una palabra

$X \wedge Y$ = operación lógica 'AND' De X e Y.

$X \vee Y$ = operación lógica 'OR - inclusivo' De X e Y.

X XOR Y = operación lógica 'OR - exclusivo' De X e Y.

$\sim X$ = operación lógica 'complemento' de X.

Ejemplo:

01101100101110011101001001111011

XOR 01100101110000010110100110110111

00001001011110001011101111001100

La operación X + Y es definida como sigue: las palabras X e Y representan enteros x e y, donde $0 \leq x < 2^{32}$ y $0 \leq y < 2^{32}$. Para enteros positivos n y m, dejamos **n mod m,** ser el resto sobre la división de n por m. Calculamos

$$Z = (x + y) \bmod 2^{32}.$$

Entonces $0 \leq z < 2^{32}$. Convertir z a palabra, Z y definir Z = X + Y.

La operación desplazamiento circular a la izquierda S^n (X), donde X es una palabra y **n** es un entero con $0 \leq n < 32$, es defino por:

$$S^n (X) = (X << n) \vee (X >> 32 - n).$$

X << n, es obtenido como sigue: desechar los n bits mas a la izquierda de X y después llenar el resultado con n ceros sobre la derecha (el resultado debería todavía ser de 32 bits). X >> n, es obtenido desechando los n bits mas a la derecha de X y después rellenar con n ceros sobre la izquierda. Puesto S^n(X)

es equivalente a un desplazamiento circular de X por **n** posiciones a la izquierda.

2.10.4 Llenado del mensaje (Message padding)

El SHA-1, es usado para calcular un resumen (message digest), para un mensaje o archivo de dato que es proveído como input. El mensaje o archivo de datos debería ser considerado como una cadena de bits. La longitud del mensaje es el número de bits dentro del mensaje (el mensaje vacío tiene longitud cero). Si el número de bits dentro de un mensaje es un múltiplo de 8, para que sea más compacto nosotros podemos representar el mensaje en hex.

El propósito del relleno (message padding) es hacer la longitud total de un mensaje rellenado múltiplo de 512. El SHA-1, secuencialmente procesa bloques de 512 bits, cuando calcula el resumen. Lo siguiente especifica cómo el llenado debería ser ejecutado. Como un resumen, un '1' seguido por **m** '0's, seguido por un entero de 64 bits, son adicionados al final del mensaje para producir el llenado de mensaje de longitud 512 * n. El entero de 64 bits es ***l***, la longitud del mensaje original. El mensaje rellenado es después procesado por el SHA-1, como un bloque de 512 bits.

Supongamos que el mensaje tiene longitud de $i < 2^{64}$. Antes que entre al SHA-1, los mensajes son llenados sobre la derecha como sigue:

a. "**1**" es adicionado. Ejemplo: si el mensaje original es "01010000", este es llenado a: "01010000**1**".
b. "0"s son adicionados. El número de ceros "0"s debería depender de la longitud original del mensaje. Los últimos 64 bits, del ultimo bloque de 512 bits, son reservados para la longitud ***l*** del mensaje original. Ejemplo: suponga

que el mensaje original es la cadena de bits siguiente: ***l*** = 40 bits.

01100001 01100010 01100011 01100100 01100101.

Después de dar el paso (a) tenemos:

01100001 01100010 01100011 01100100 01100101 1

desde que ***l*** = 40, el número de bits con el paso (a) adicionado es 41 y 407 "0"s son adicionados, haciendo ahora un total de 448. esto dado en hexadecimal tenemos:

61626364 65800000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000.

c. Obteniendo las dos(2) palabras (word) de la representación de ***l***, el número de bits dentro del mensaje original. Si ***l*** < 2^{32} entonces la primera palabra es todo ceros. Adicione estas dos palabras al relleno del mensaje. Ejemplo: suponga que el mensaje original es como en el paso (b). Entonces ***l*** = 40 (note que ***l*** es computado antes de cualquier relleno (padding)). Las representación de 40 en dos (2) palabras (word) en hexadecimal es: 00000000 00000028.

Entonces el mensaje de relleno final en hexadeciamal es:

61626364 65800000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000028.

El mensaje rellenado debería contener 16**n** palabras para **n** > 0. el mensaje rellenado, es considerado como una secuencia de **n** bloques M_1, M_2, ... , M_n, donde cada M_i, contiene 16 palabras y M_1, contiene los primeros caracteres (o bits) del mensaje.

2.10.5 Funciones usadas

Una secuencia de funciones lógicas f_0, f_1, ... , f_{79} es usada dentro de SHA-1. Cada f_t, es definido como sigue: para palabras **B, C, D**,

F_t (**B, C, D**) = (**B** $\wedge$ **C**) $\vee$ (~**B** $\wedge$ **D**) (0 $\leq$ t $\leq$ 19)

F_t (**B, C, D**) = **B** XOR **C** XOR **D** (20 $\leq$ t $\leq$ 39)

F_t (**B, C, D**) = (**B** $\wedge$ **C**) $\vee$ (**B** $\wedge$ **D**) $\vee$ (**C** $\wedge$ **D**) (40 $\leq$ t $\leq$ 59)

F_t (**B, C, D**) = **B** XOR **C** XOR **D** (60 $\leq$ t $\leq$ 79).

2.10.6 Constantes usadas

Una secuencia de palabras constantes K_0, K_1, ... , K_{79}, es usado dentro de SHA-1. En hexadecimal están dadas por:

K_t = 5A827999 (0 $\leq$ t $\leq$ 19)

K_t = 6ED9EBA1 (20 $\leq$ t $\leq$ 39)

K_t = 8F1BBCDC (40 $\leq$ t $\leq$ 59)

K_t = CA62C1D6 (60 $\leq$ t $\leq$ 79).

2.10.7 Calculando el Resumen o destilado (message digest)

El resumen o destilado es calculado usando el mensaje relleno final. El calculo usa dos (2) almacenamientos intermedios (buffers), cada uno consistente de 5 palabras de 32 bits, y adicionalmente de una secuencia de 80 palabras de 32 bits. Las 5 palabras del primer almacenamiento intermedio (buffer),

son etiquetadas como A, B, C, D, E. Las 5 palabras del segundo almacenamiento intermedio son etiquetados como H_0, H_1, H_2, H_3, H_4. Las 80 palabras de la secuencia adicional son etiquetadas como W_0, W_1, ..., W_{79}.

Una sola palabra (buffer) **TEMP**, es también empleada.

Para generar el resumen , el bloque de 16 palabras M_1, M_2, ... , M_n, definidas en la sección 2.10.4, son procesadas en orden. El procesamiento de cada M_i involucra 80 pasos.

Antes de procesar cualquier bloque, los $\{H_j\}$ son inicializados como sigue: en hex.

H_0 = 67452301

H_1 = EFCDAB89

H_2 = 98BADCFE

H_3 = 10325476

H_4 = C3D2E1F0.

Ahora M_1, M_2, ... , M_n, son procesados. Para procesar M_i , nosotros procederos como sigue:

a. Dividir M_i en 16 palabras W_0, W_1, ... , W_{15}, donde W_0 es la palabra mas a la izquierda.
b. Para t = 16 a 79, tenemos:
   $W_t = S^1 (W_{t-3}$ XOR W_{t-8} XOR W_{t-14} XOR $W_{t-16})$.
c. Dejamos $A = H_0$, $B = H_1$, $C = H_2$, $D = H_3$, $E = H_4$.
d. Para t = 0 a 79, hacer
   $TEMP = S^5 (A) + F_t (B,C,D) + E + W_t + K_t$;
   $E = D$; $D = C$; $C = S^{30}(B)$; $B = A$; $A = TEMP$;
e. Dejemos que $H_0 = H_0 + A$, $H_1 = H_1 + B$, $H_2 = H_2 + C$, $H_3 = H_3 + D$,
   $H_4 = H_4 + E$.
f. Después del procesamiento de M_n, el resumen o destilado (message digest), es la cadena de 160 bits, representado por 5 palabras, H_0 H_1 H_2 H_3 H_4.

2.10.8 Método de cálculo alternativo

Lo anterior asumen que la secuencia W_0, ..., W_{79}, es implementado como un arreglo de 80 palabras de 32 bits. Éste es eficiente desde el punto de vista de minimización del tiempo de ejecución, desde que las direcciones de W_{t-3}, ..., W_{t-16} dentro del paso (b) son fáciles de calcular.

Si el espacio es apremiante, una alternativa a esto es para { W_t }, como una cola circular el cual puede ser implementado usando un arreglo de 16 palabras de 32 bits W[0], ... W[15]. Dentro de este caso, en hex. dejamos la máscara MASK = 0000000F. Entonces el procesamiento de M_i, es como sigue:

a. Dividir M_i en 16 palabras W[0], W[1], ... , W[15], donde W[0], es la palabra más a la izquierda.
b. Dejamos $A = H_0$, $B = H_1$, $C = H_2$, $D = H_3$, $E = H_4$.
c. Para t = 0 a 79, hacer

S = t $\wedge$ MASK;

If (t $\geq$ 16)

W[s] = S^1 (W [(s+13) $\wedge$ MASK] XOR W [(s+8) $\wedge$ MASK] XOR

W [(s+2) $\wedge$ MASK] XOR W [s]);

TEMP = S^5 (A) + f_t (B, C,D) + E + W[s] + K_t ;

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

d. Dejamos que $H_0 = H_0 + A$, $H_1 = H_1 + B$, $H_2 = H_2 + C$, $H_3 = H_3 + D$,

$H_4 = H_4 + E$.

2.10.9 Comparación de Métodos

Los métodos de las secciones anteriores 2.10.7, 2.10.8, producen el mismo resumen o destilado (message digest). Aunque usando el método de la sección 2.10.8, ahorra 64 palabras de 32 bits de almacenamiento, eso es probable tome más tiempo de ejecución, debido a la complejidad aumentada de los cálculos de las direcciones para el { W[t] dentro del paso (c). Otro método de cálculo que dé idéntico resultado puede ser implementado en concordancia con el estándar.

2.11 Autoridad de Certificación (Certificación Authority)

La autoridad de certificación es la responsable de proveer y asignar las claves de encriptación, desencriptacion y autenticación. Una autoridad de certificación distribuye claves, emitiendo certificados el cual contiene la clave pública y un conjunto de atributos.

Una autoridad de certificación puede emitir certificados para una computadora, una cuenta de usuario o un servicio.

Certificados. Son documentos firmados por las autoridades de Certificación que asocia la clave publica a otra información, tal como un nombre o una dirección e-mail. Una firma de la autoridad de certificación garantiza que la clave pública, pertenece a la parte que presenta eso.

Las autoridades de certificación pueden ser externas o internas, tal como un departamento de una compañía que ha instalado su propio servidor para emitir y verificar certificados. Cada autoridad de certificación decide que atributos incluir en un certificado y que mecanismos usa para verificar estos atributos, antes de emitir un certificado.

Adicionalmente cada Autoridad de Certificación tiene un certificado para confirmar su propia identidad, emitido por otra Autoridad de certificación o por sí mismo.

Proceso para emitir un certificado son:

- La Autoridad de Certificación acepta el pedido de certificado.
- La Autoridad de Certificación verifica que la información pedida está de acuerdo a la prueba de identidad requerida de la Autoridad de Certificación.
- La Autoridad de Certificación usa la clave privada para aplicar la firma digital a el certificado.
- La Autoridad de Certificación emite el certificado para usar como credencial de seguridad con una infraestructura de clave pública.

Una Autoridad de Certificación es responsable de la revocación de certificados y de la publicación de una lista de certificados revocados. La revocación de un certificado invalida el certificado como una credencial de seguridad de confianza ante la expiración de un periodo valido del certificado.

Un certificado debería ser revocado por:

- Riesgo o supuesto riesgo de la clave privada.
- Descubrir que un certificado fue obtenido fraudulentamente.
- Cambio en el estado del certificado, sujeto como entidad de confianza.

III PROCESO DE TOMA DE DECISONES

3.1 Planteamiento del problema

La Oficina Nacional de Procesos Electorales ONPE, en todo proceso electoral, de referéndum o Consultas Populares, crea temporalmente a nivel nacional, Oficinas Descentralizadas de Procesos Electorales (ODPE's).

Dichas Oficinas Descentralizadas de Procesos Electorales (ODPE's) a través de sus respectivos centros de c'omputo, tienen la responsabilidad de procesar y transmitir información de los resultados electorales a la sede central.

Siendo el principal problema, la vulnerabilidad de la información en la transmisión de datos, desde las ODPE's (emisores) a la sede central (receptor).

Esto se debe a que los datos son enviados en texto claro o legible, por lo que no existe garantía de confidencialidad en las comunicaciones y además los canales usados para la transmisión de los datos no son seguros, debido a que en la mayoría de los casos escapan a nuestro control, ya que pertenecen a terceros (Telefónica, AT&T, FirstCom) y resulta imposible asegurarse de que no están siendo escuchados o intervenidos.

Pero adicionalmente se identificó otros dos problemas en la transmisión de datos:

- La falta de integridad de los datos; es decir, la imposibilidad de verificar si los datos transmitidos, han sido alterados en el trayecto, de forma fraudulenta.
- La falta de autenticidad entre las ODPE's y la sede central ONPE, es decir, que las partes intervinientes en el proceso de transmisión de datos, lo hacen de forma legal y representan quienes dicen ser.

3.2 Alternativa de Solución

Como solución a los factores críticos encontrados en la transmisión de datos de un Proceso Electoral, se desarrolla el **protocolo TDS (Transmisión de Datos Seguro).**

Por lo tanto, los procesos definidos en TDS, dentro de los límites establecidos son:

- Garantizar la confidencialidad de la información sensible (número de mesa, cantidad de votos válidos, cantidad de votos en blanco, cantidad de votos nulos.).
- Preservar la integridad de la información transmitida.
- Proporcionar la autentificación necesaria entre las ODPE's (emisores) y la sede central ONPE (receptor).
- Definir los algoritmos criptográficos necesarios para los servicios anteriores.

Aunque estos aspectos son importantes en cualquier red de transmisión de datos, su vulnerabilidad causa una preocupación mucho mayor cuando se relaciona con un Proceso Electoral.

¿Cómo se implementan en TDS, todos los procesos de autentificación, confidencialidad e integridad enunciadas anteriormente?

Estas definiciones constituyen el núcleo de TDS:

- La confidencialidad (no-vulnerabilidad de la información conteniendo los datos como número de mesa, ubicación, cantidad de votos válidos, cantidad votos en blanco, cantidad de votos nulos), se alcanza con la **encriptación de los mensajes utilizando el algoritmo Rijndael (AES).**
- La integridad de los datos contenidos en las transacciones de los votos, garantizando que no han sido modificados a lo largo de su trayecto, se consigue mediante el uso de **firmas digitales utilizando la función Hash SHA-1.**
- La autenticidad de las Oficinas Descentralizadas de Procesos Electorales (ODPE's), garantizando que forma parte de la organización de la ONPE, y lo identifica como único responsable para la transmisión de datos, se consigue mediante la **emisión de certificados digitales del tipo X.509 v3 para las ODPE's y las correspondientes firmas digitales, utilizando el servidor de certificados digitales de Windows NT, quedando la ONPE como una Autoridad de Certificación interna para la Institución.**
- Utilizar un protocolo que no depende de otros mecanismos de seguridad de transporte, así como tampoco evite su utilización.

3.3 Metodología de Solución

Teniendo en consideración, que la solución propuesta, resuelve un conjunto de problemas, con diferentes técnicas y metodologías, es necesario describir los pasos a seguir, para que el protocolo funcione.

- Generar certificados digitales, al jefe de centro de cómputo de cada ODPE a nivel nacional. Por consiguiente cada ODPE, contará con una clave privada (K_p) y otra clave pública (*KP*). La clave privada la tendrá única y exclusivamente el jefe de centro de cómputo de la respectiva ODPE (emisor) y la clave pública la tendrá la sede central (receptor).
- A través de un procedimiento automatizado, cada centro de cómputo antes de enviar el archivo de datos del avance de resultados electorales, primero ejecuta la función Hash (SHA-1) sobre el archivo de datos, generando un "**resumen**" o "**destilado**" de 160 bits.
- Luego se cifra el archivo de datos, con el algoritmo Rijndael (AES), para el cual se utiliza una clave simétrica (**Ks**) de 256 bits, que es manejada por la sede central.
- Con la clave privada (K_p) del certificado digital emitida a la ODPE (emisor), se cifra el "**resumen",** utilizando el algoritmo asimétrico RSA, obteniendo la **firma digital** del archivo de datos.
- Con la clave privada (K_p) del certificado digital emitida a la ODPE (emisor), se cifra la clave simétrica (**Ks**), utilizando el algoritmo asimétrico RSA. (Sobre electrónico = Archivo de datos cifrado + clave simétrica (**Ks**) cifrada).
- El procedimiento transfiere vía FTP (File Transfer Protocolo), el archivo de datos cifrado, más la clave simétrica cifrada, mas el "resumen" cifrado al centro de cómputo de la sede central.
- En la sede central, el receptor del mensaje, utilizando otro procedimiento automatizado, primero descifra el "resumen" y luego descifra la clave simétrica (**Ks**), con la clave pública (K_P) del emisor.

- Con la clave simétrica (Ks) ya desencriptada, se descifra el archivo de datos, y se le aplica la misma función Hash, para luego comparar ambos "resumen".
  Si son iguales, la integridad y autenticidad del archivo de datos son correctas.
  Si el proceso de descifrado no es satisfactorio, el trasmisor no puede ser autenticado.
  Si el "destilado" o "resumen" generado no es coincidente con el extraído de la firma digital, se ha producido una modificación en el contenido del archivo.

3.4 Toma de Decisiones

Los algoritmos criptográficos empleados en TDS para los procesos de encriptación, emisión de certificados y generación de firmas digitales son de doble naturaleza. Por un lado se define un algoritmo de clave privada, de fortaleza contrastada y excelente rendimiento: Rijndael (ver anexo).

Por otro lado, se hace imprescindible contar con un algoritmo que permita el intercambio de claves en una red pública, con total seguridad, entre múltiples participantes sin ninguna relación previa; un algoritmo como el descrito se define de clave pública, y el escogido para TDS fue diseñado por Rivest, Shamir y Adleman, cuyas iniciales componen su nombre: RSA (ver anexo).

Esencialmente, cada algoritmo criptográfico empleado en TDS, permite la implementación de una función determinada. Rijndael, se emplea para garantizar la confidencialidad de los mensajes transmitidos; RSA se emplea para garantizar la integridad de los datos y la autenticidad de los participantes. RSA desempeña todavía una función adicional, posible gracias a su definición como algoritmo de clave pública (también se le conoce como algoritmo asimétrico, por emplear dos claves diferentes, una para

la encriptación y otra para la desencriptacion): permite la distribución y utilización de una clave secreta entre participantes sin ninguna relación previa y, lo que es mas importante, sobre canales de comunicación no asegurados.

Para finalizar la alternativa de solución, veremos más detenidamente las diversa utilizaciones que se hacen de los algoritmos criptográficos definidos.

Rijndael, como algoritmo de clave privada (algoritmo simétrico) requiere que las partes intervinientes en un proceso de encriptación /desencriptacion compartan la misma clave. Esto plantea problemas de distribución de claves en entornos no seguros. A nadie se le pasa por la mente distribuir una clave Rijndael, mediante un mensaje de correo electrónico. Incluso por algunos canales "menos públicos" como puede ser el correo ordinario o el teléfono son sumamente vulnerables. Sólo la entrega a mano garantiza que una clave no ha sido descubierta durante la distribución.

Los algoritmos de clave pública, como RSA, están sustentados en una base matemática tal que cada una de las partes intervinientes dispone de un par de claves: una se denomina clave pública, y está destinada a ser distribuida libremente. Es más, cuanto más ampliamente se haya distribuido esta clave, más garantías existen de que no es posible la "usurpación de personalidad". La otra clave, la clave privada será conocida solamente por su legítimo propietario, y debe ser custodiada con el mismo celo con que se haría para una clave Rijndael. La base matemática aludida anteriormente hace que mientras que un mensaje puede ser cifrado con la clave pública, es necesaria la clave privada para su desencriptacion. Se resuelve así el problema de la distribución de claves sobre canales no seguros.

3.5 Estrategias Adoptadas

La principal estrategia adoptada:

- Crear una extranet, entre las ODPE's y la sede central ONPE, instalando un servidor remoto VPN (Virtual Paht, Network). VPN crea una conexión virtual, punto a punto con un gateway VPN que reside en la red privada de ONPE. Típicamente usted conecta primero a Internet y después crea la conexión VPN.
- Instalar el servidor de Servicio de Certificado digital, para que la sede central ONPE, sea una autoridad de certificación interna y pueda emitir certificados a todas las ODPE's, que va a permitir autenticar y verificar al originador o emisor de la data transmitida.
- La Sede central es la única que maneja las claves simétricas, para cifrar el archivo de datos.
- Para la transferencia de archivos utilizar FTP (File Transfer Protocol).

IV EVALUACIÓN DE RESULTADOS

Estado inicial.

Es importante realizar un breve diagnostico de cómo eran antes, las transmisiones de datos, en procesos electorales.

- Los datos eran enviados en texto claro y legibles.
- No se protegía la integridad de los datos.
- No había forma, de estar seguro, de que los datos recibidos vienen de quien realmente creemos que viene, es decir de las ODPE's.
- Una ODPE, a pesar de haber enviado datos, podían negarse, ya que no existía modo de probarlo.

Es decir, la vulnerabilidad de los datos, era una constante preocupación, especialmente cuando esta se relaciona con un proceso electoral.

Después de tomada la decisión, de utilizar técnicas y algoritmos criptográficos como medio de seguridad y protección para las transmisiones de datos en proceso electorales, podemos aseverar que esta solución, ofrece mecanismos de seguridad muy elevados.

Debido que asegura cuatro aspectos fundamentales de la seguridad informática:

- Se garantiza la confidencialidad de la información sensible.
- Se preserva la integridad de la información transmitida.
- Se garantiza la autenticidad de las ODPE's como parte de la organización de ONPE.

- Se garantiza el no repudio de la información.

Si se cumplen con los principios anteriormente mencionados, diremos en general que los datos trasmitidos están protegidos y seguros.

V CONCLUSIONES Y RECOMENDACIONES

- Podemos concluir que cada técnica criptográfica empleada en el protocolo TDS, permiten la implementación de una función determinada, ofreciendo en redes abiertas, mecanismos de seguridad muy elevados, durante la transmisión de datos en procesos electorales.
- Se concluye que las tecnologías de encriptación debería ser un componente clave de las arquitecturas globales de seguridad de la información de las empresas.
- Existe un creciente interés en este campo de la encriptación, tanto entre las empresas clientes como entre los fabricantes.
- Esta tecnología de seguridad actuará como impulsor del comercio electrónico.
- Se recomienda que la institución elabore políticas de seguridad en lo referente a políticas de flujo de información, el cual definiría las normas bajo las cuales se comunican los sujetos dentro del sistema.
- Se recomienda que el estado peruano reglamente o norme la ley 27269, ley de firmas y certificados digital, para que estas tecnologías puedan masificarse en nuestro medio, a través de los negocios ínter empresas y entre las instituciones del estado.
- Se recomienda que a nivel de estado, se estandarice al uso de:
  Un sólo algoritmo criptográfico asimétrico fuerte, con claves de 1024 a 2048 bits y un sólo algoritmo criptográfico simétrico robusto, con claves de 256 bits (tipo militar), para la

comunicaciones entre Ministerios, Embajadas diplomáticas, Bancos, etc.

BIBLIOGRAFÍA

- Cryptography Theory and Practice Douglas R. Stinson.
- Fundamentos Matemáticos de la Criptografía Antonio Portilla Silva.
- Criptografía y Seguridad en Computadoras Manuel J. Lucena Lopez.
- Federal Information Processing Standards Publication 197 AES – Rijndael
- Federal Information Processing Standards Publication 180-1 SHA –1
- http://www.nist.gov
- http://www.counterpane.com
- http://www.kriptopolis.com
- http://www.criptored.upm.es

ANEXOS

DES (Data Encryption Standard)

DES (Estándar de Cifrado de Datos), es un algoritmo desarrollado por IBM a requerimiento del NBS (Nacional Bureau of Standards, Oficina Nacional de Estandarización, en la actualidad denominado NIST, National Institute of Standards and Technology, Instituto Nacional de Estandarización y Tecnología) de EEUU y posteriormente modificado y adoptado por el gobierno de EEUU en 1977 como el estándar de cifrado de datos de todas las informaciones sensibles no clasificadas. Posteriormente en 1980, el NIST estandarizó los diferentes modos de operación del algoritmo. Es el mas estudiado y utilizado de los algoritmos de clave simétrica.

El nombre original del algoritmo, tal como lo denomino IBM, era Lucifer. Trabajaba sobre bloques de 128 bits, teniendo la clave igual longitud. Se basaba en operaciones lógicas booleanas y podía ser implementado fácilmente, tanto en software como en hardware.

Tras la modificaciones introducidas por el NBS, consistentes básicamente en la reducción de la longitud de la clave y el de los bloques, DES cifra bloques de 64 bits, mediante permutación y sustitución , usando una clave de 64 bits, de los cuales 8 son de paridad (Esto es, en realidad usa 56 bits), produciendo así 64 bits cifrados.

Una completa descripción de DES es dada por el Federal Información Processing Standards Publication 46, fecha 15 de Enero de 1977.

DES encripta un texto plano (texto claro o texto legible). Una cadena de bits **x** de longitud 64 (8 bytes), usando una clave ***K,*** el cual es una cadena de bits de longitud 56 (7 bytes).

Primero vamos a dar una descripción de alto nivel del sistema. El algoritmo procede en tres etapas o fases.

1. Dado un texto plano (texto legible) x, una cadena de bits x_0 , que es construida por permutaciones de bits de x, según una permutación fija inicial, llamada IP. Nosotros escribimos $x_0 = IP(x) = L_0R_0$, donde L_0 ,

comprende o incluye los primeros 32 bits de x_0 y R_0 los últimos 32 bits.

2. Dieciséis (16) interacciones de una cierta función son después calculados. Nosotros calculamos L_iR_i, $1 \leq i \leq 16$, según la siguiente regla:

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

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

Una vuelta de encriptación:

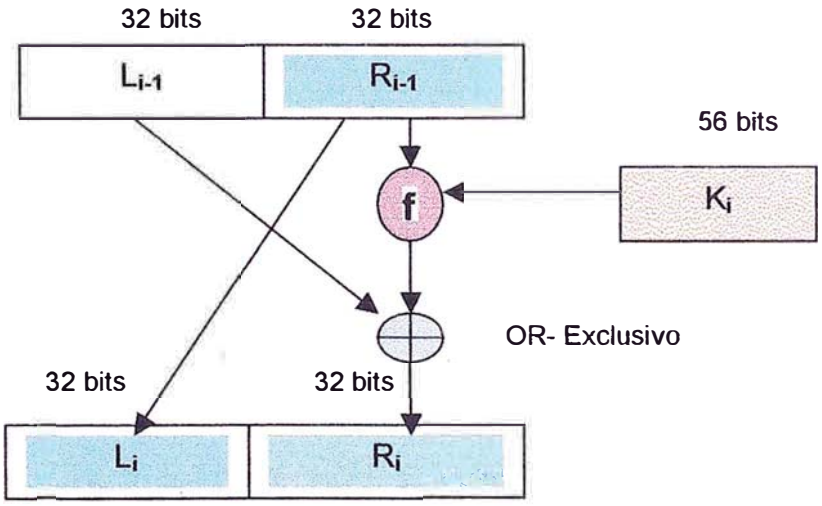


Figura 1: Una ronda de encriptación DES

Nosotros denotamos:

⊕ el OR –exclusivo de 2 cadenas de bits. **f** es una función que nosotros podríamos describir después, y *K1, k2, ...K16* son cada uno, cadenas de bits de longitud 48 (6 bytes). Calculados como una función de clave **K.** (Actualmente cada ***Ki***, es una permutación. Selección de bits de **K**). *K1, K2, K3, ...K16* comprende la lista de claves. Una vuelta de encriptación es descrita en la figura anterior.

3. Aplicando la permutación inversa IP^{-1} a la cadena de bits $R_{16}L_{16}$ obteniendo el texto cifrado **y**. Esto es $y = IP^{-1}(R_{16}L_{16})$.

   Note el orden invertido de L_{16} y R_{16}.

La función **f** toma como entrada un primer argumento **A,** el cual es una cadena de bits de longitud 32, y un segundo argumento **J,** que es una

cadena de bits de longitud 48 y produce una salida, una cadena de bits de longitud 32. Los siguientes pasos son ejecutados.

- El primer argumento **A**, es expandido a una cadena de bits de longitud 48, según una función de expansión fija **E. E(A)** consiste de los 32 bits de A, permutando dentro de un cierto camino, con 16 de los bits de entrada dos veces (Es decir se repiten dos veces). La función f, esta descrita en la siguiente figura:

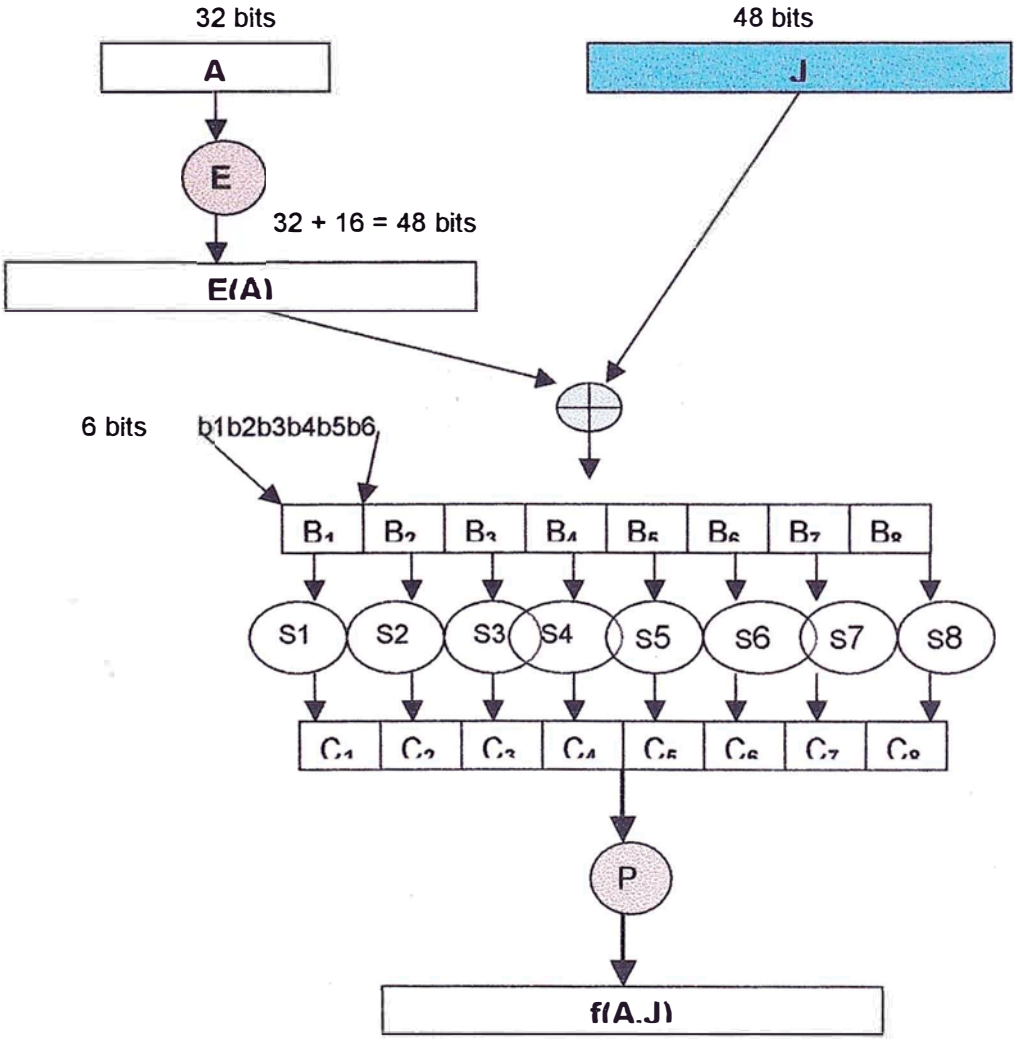


Figura 2: La función f de DES

- Computar o calcular **E(A)** ⊕ **J** y escribe el resultado como la concatenación de ocho de 6, cadenas de bits. B = B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8.

- El siguiente paso usa ocho S-cajas S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8. Cada Si, es un arreglo fijo de 4x16, cuya entrada viene de los enteros 0 – 15. dado una cadena de bits de longitud 6 (ejemplo los B_j). Decimos B_j = b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8. Nosotros calculamos S_j (B_j) como sigue: los dos bits b_1b_6, determinan la representación binaria, de una fila **r** de Sj ($0 \leq c \leq 3$), y los cuatro bits b_2 b_3 b_4 b_5, determina la representación binaria una columna c de Sj ($0 \leq c \leq 15$), entonces S_j (r, c) es escrito en binario, como una cadena de bits de longitud cuatro (4). (Entonces cada Sj puede ser pensado como una función que acepta como entrada una cadena de bits de longitud dos (2) y una de longitud cuatro (4) y produce una salida de una cadena de bits de longitud cuatro (4). Dentro de este modo nosotros computamos Cj = Sj (Bj), $1 \leq j \leq 8$.
- La cadena de bits C = C_1 C_2 C_3 C_4 C_5 C_6 C_7 C_8 de longitud 32,es permutado, según una permutación fija **P**. La cadena de bits resultante de P(C) es definido par ser **f (A, J)**.

La función básicamente consiste de una substitución (usando S-cajas), seguido por una permutación fija **P**. Las 16 iteraciones de **f** comprende un producto de sistema.

En el resto de esta explicación, nosotros presentaremos una función especifica usada dentro de DES.

Tabla 1: Permutación inicial (IP) de DES

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

Esto quiere decir que el 58avo bit de x, es el primer bit de IP(x), el 50avo bit de x, es el segundo bit de IP(x).La permutación inversa IP^{-1} es:

Tabla 2: La Permutación inversa de (IP)

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

La función de expansión E:

Tabla 3: Función de Expansión E de DES

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	33

La permutación P y las ocho S-cajas, son mostradas

Tabla 4: Permutación P de DES

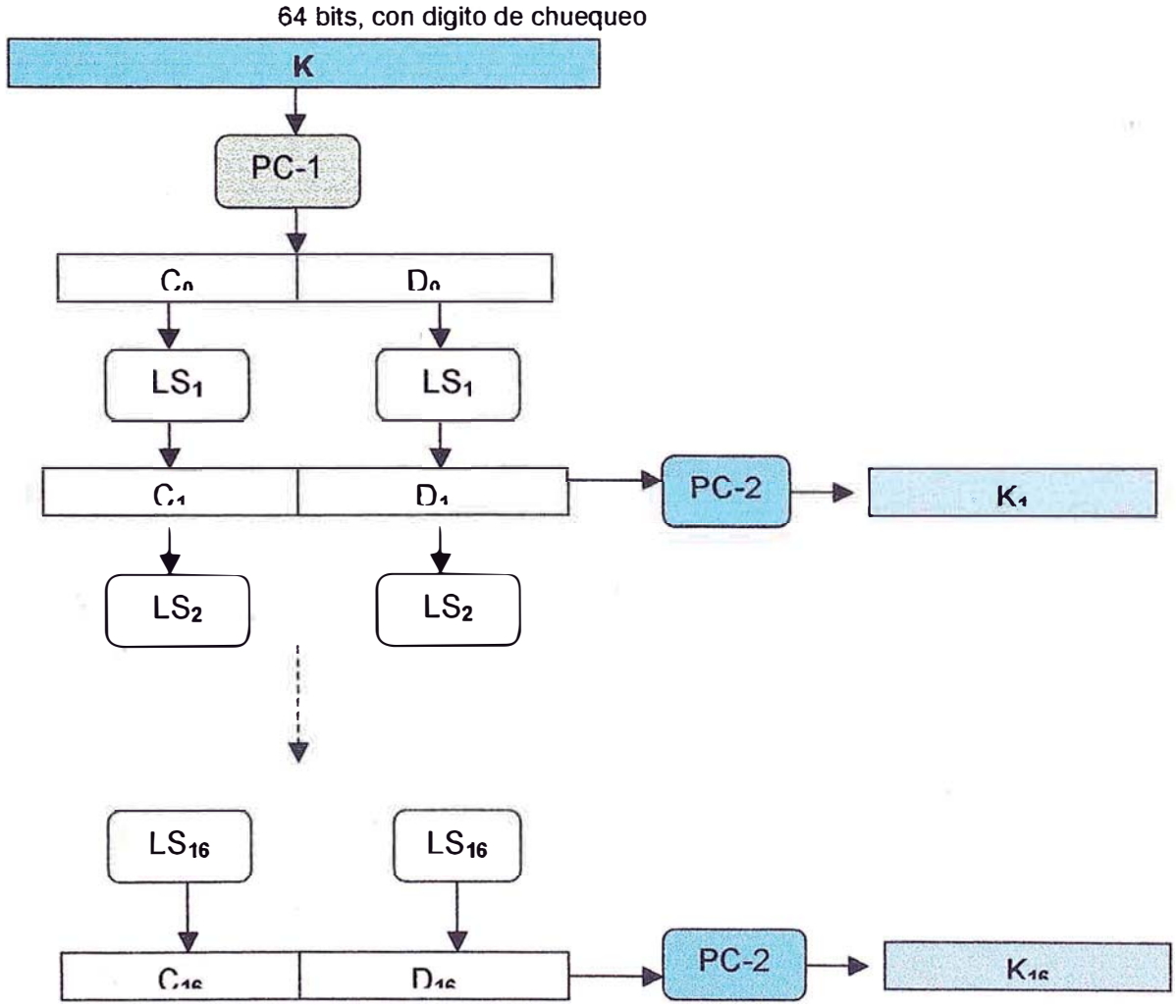
P			
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

	Columna																
Fila	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	S-Cajas
00	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	
01	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	
10	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	S1
11	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
00	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	
01	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	S2
10	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
11	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
00	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	
01	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	S3
10	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
11	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
00	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	
01	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	S4
10	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
11	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	
00	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	
01	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	S5
10	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
11	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
00	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	
01	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	
10	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	S6
11	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
00	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	
01	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	
10	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	S7
11	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
00	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	
01	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	
10	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	S8
11	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

Finalmente, nosotros necesitamos describir el calculo de la lista de claves de la clave **K**. Actualmente **K**, es una cadena de bits de longitud 64, de los cuales 56 bits, comprende la llave y 8 bits son bits de chequeo de paridad (para detectar errores). Los bits dentro de la posición 8,16,24,32,40,48,56,64, son definidos tal que cada byte contiene un numero impar.

El calculo del inventario de claves, es descrito en la siguiente figura.

Figura 3: Calculo del inventario de claves de DES



- Dado una clave de 64 bits, descartando el bit de chequeo y permutando los restos de bits de **K**, de acuerdo a una permutación fija

PC-1. Nosotros podríamos escribir PC-1(**K**) = C_0d_0, donde C_0, comprende los primeros 28 bits de PC-1(**k**) y D_0, los últimos 28 bits.

- Para un rango de **i**, de 1 a 16, calcular:

$$C_i = LS_i\ (C_{i-1})$$

$$D_i = LS_i\ (D_{i-1})$$

y K_i = PC-2 ($C_{ii}D_i$). LS_i representa un desplazamiento cíclico (a la izquierda) de una o dos posiciones, dependiendo del valor de **i**: desplazando una posición si **i** = 1, 2, 9, o 16 y desplazando dos posiciones si es otro caso. PC-2, es otra permutación fija.

Las permutaciones PC-1 y PC-2, usadas dentro del inventario de claves, son como sigue:

Tabla 5: Permutación PC-1, del inventario de claves de DES

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

Tabla 6: Permutación PC-2, del inventario de claves de DES.

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

Nosotros ahora mostramos el resultado del inventario de claves. Como mencionamos anteriormente, cada ronda usa una clave de 48 bits, comprendido de 48 bits dentro de K. Las entradas dentro de las tablas abajo, se refiere a los bits dentro de K, que son usados dentro de varias rondas.

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

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

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

La desencriptacion es hecha usando el mismo algoritmo como la encriptación, comenzando con **Y** como entrada, pero usando el inventario de claves K_{16}, K_{15}, K_{14}, ...K_1 EN ORDEN INVERSO. La salida debería dar el texto claro **x.**

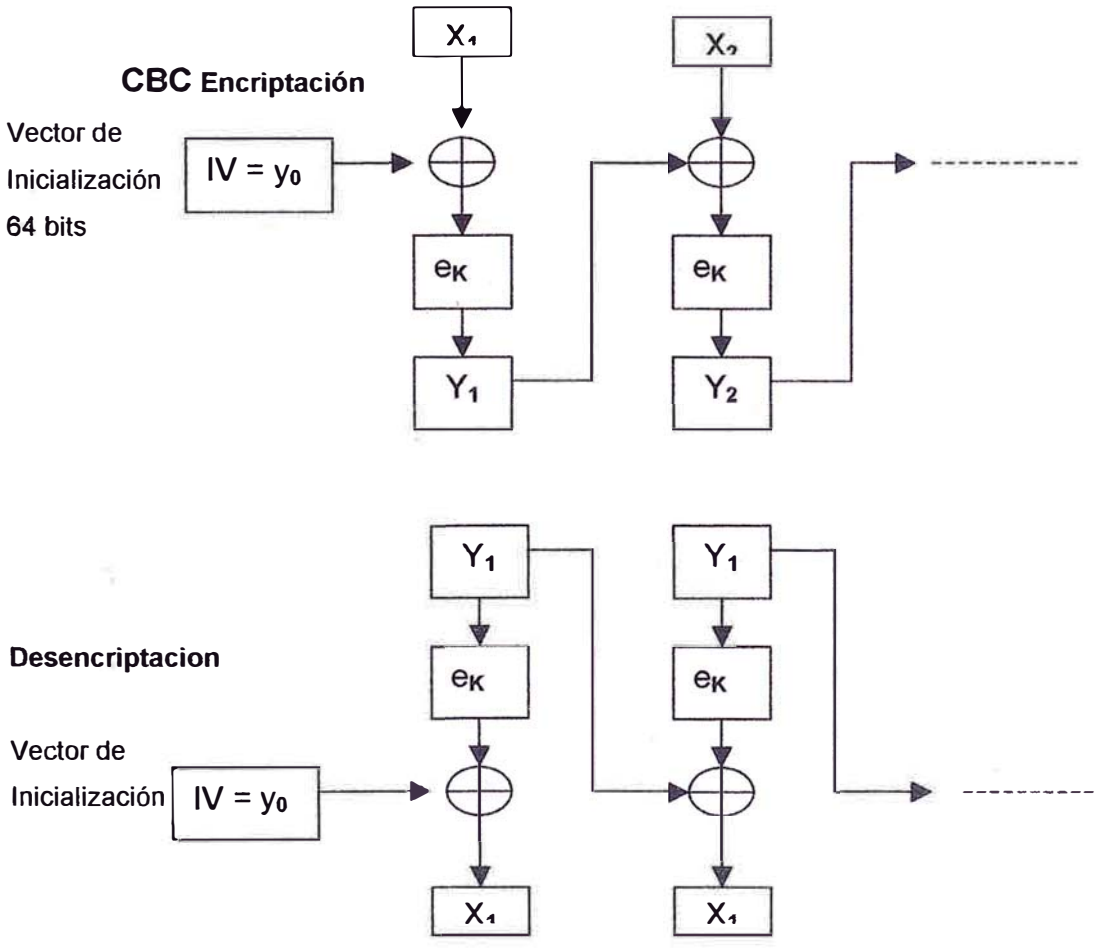
MODO DE OPERACIÓN DE DES:

Cuatro modos de operación han sido desarrollado para DES. Electronic Code Mode (**ECB**), Cipher Feedback Mode (**CFB**), Cipher Block Chaining Mode (**CBC**) I Output Feedback Mode (**OFB**).

El Modo ECB, corresponde al usual uso de un bloque de cifrado, dado una secuencia x_1, x_2, x_3, ...de bloques de 64 bits de texto plano, cada x_i, es encriptado con la misma clave **K,** produciendo una cadena de bloque de texto cifrado y_1, y_2, y_3, ... A favor de este método, podemos decir que permite codificar bloques independientemente de su orden, lo cual es adecuado para codificar bases de datos o ficheros en los que se requiera

acceso aleatorio. También es resistente a errores, pues si uno de los bloques sufriera una alteración, el resto quedaría intacto.

El Modo CBC, (Cipher Block Chianig), cada Bloque de texto cifrado y_i, es aplicado el Xor con el siguiente bloque de texto plano x_{i+1}, antes de ser cifrado con la clave **K.** Formalmente nosotros comenzamos con un vector de inicialización (**IV**) de 64 bits y definimos y_0 = IV. Después nosotros construimos y_1, y_2, y_3, ... de la regla $y_i = e_K (y_{i-1} \oplus x_i)$, $i \geq 1$. ver la siguiente figura.

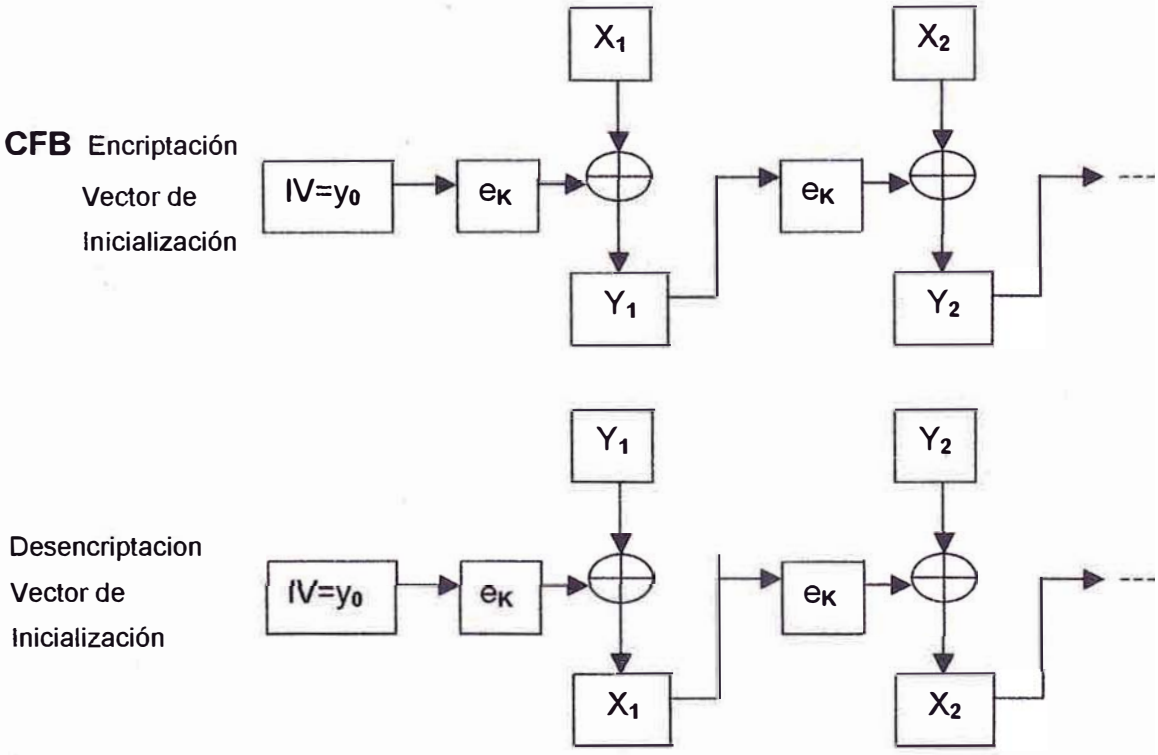


Dentro de los modos **OFB** y **CFB**, una cadena de clave es generado, el cual es después realizado el Xor con el texto plano.

OFB, es actualmente un cifrado de cadena **sincrono** (es decir la generación de la cadena de clave es independiente del texto plano). La cadena de clave es producida por repetida encriptación en un vector de inicialización de 64 bits (**IV**). Nosotros definimos Z_0 = **IV** y después computamos la cadena de claves z**1**, z**2**, z**3**, ... donde la regla es $z_i = e_K (z_{i-1})$, $i \geq 1$. La secuencia de texto plano $x_1 x_2 x_3$ ..... es encriptada por el calculo de $y_i = x_i \oplus z_i$, $i \geq 1$.

El Modo CFB, (Cipher Feedback Mode), nosotros comenzamos con y_0 = IV (vector de inicialización) y nosotros producimos un elemento de la cadena de claves z_i, por encriptación del previo bloque de texto cifrado.

Esto es $z_i = e_K (y_{i-1})$, $i \geq 1$. Como dentro del modo OFB $y_i = x_i \oplus z_i$, $i \geq 1$. El uso de CFB es descrito en la siguiente figura: (note que la función de encriptación e_K es usado por ambos, encriptación y desencriptacion en modo CFB y OFB).



Los cuatro modos de operación tienen diferentes ventajas y desventajas.

Dentro del modo ECB y OFB, un cambio en el bloque de texto plano de 64 bits, causa el correspondiente bloque de texto cifrado y_i, a ser alterado, pero otro bloque de texto cifrado, no son afectados. En algunas situaciones esta podría ser una deseable propiedad. Ej. El modo OFB es frecuentemente usado para encriptar transmisiones de satélite. De otra manera, si un bloque de texto plano x_i, es cambiado a los modos CBC y CFB, entonces y_i y todos los bloques de texto cifrado subsecuentemente podrían ser afectados. Esta propiedad quiere decir que los modos CBC CFB, son usados usualmente para propósitos de autenticación. Mas específicamente, estos modos pueden ser usados para producir un mensaje de autenticación (message Authentication code o MAC).

Function Hash MD5 "Message Digest 5"

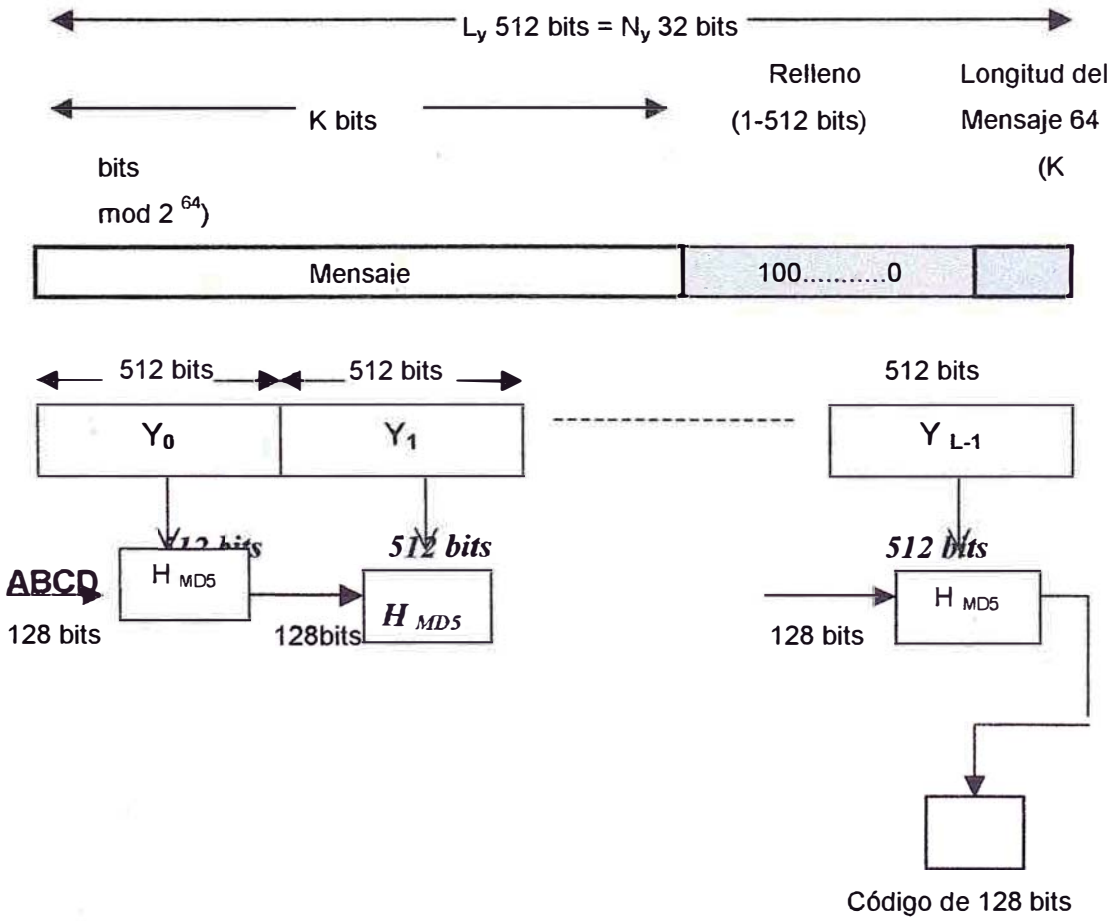
El algoritmo Mensaje- resumen MD5 fue desarrollado por Ron Rivest en el MIT.

Lógica del MD5

El algoritmo toma como entrada un mensaje de longitud arbitraria y produce como salida un mensaje resumen de 128 bits. La entrada es procesada en bloques de 512-bits. La siguiente figura describe el procedimiento completo para producir un resumen o destilado a partir de un mensaje.

El procedimiento se compone de los siguientes pasos:

Figura 4: Generación de un mensaje resumen utilizando MD5



Paso 1: Añadir relleno de bits

El mensaje es rellenado para que la longitud en bits sea congruente con 448 modulo 512 (longitud =448 mod 512). Esto quiere decir, que la longitud del mensaje relleno es 64 bits mas que un entero múltiple de 512 bits. El relleno es añadido siempre, aunque el mensaje ya sea de la longitud deseada. Por ejemplo, si el mensaje es de 448 bits de longitud, este es rellenado con 512 bits, sumando un total de 960 bits. Por lo tanto, el numero de bits de relleno se sitúa entre 1 – 512 bits. El relleno consiste de un único 1 – bit, seguido del numero necesario de 0 – bits.

Paso 2: Añadir la longitud

Una representación de la longitud del mensaje original en 64 bits, sin contar el relleno, es añadida al resultado del paso 1. Si la longitud original es mayor de 2^{64}, entonces solo será utilizada la longitud por debajo de 64 bits. De esta manera, el campo contiene la longitud original del mensaje, modulo 2^{64}.

El resultado de los dos primeros pasos, produce un mensaje de longitud, un entero múltiple de 512 bits. En la figura anterior, el mensaje extendido es representado como una secuencia de bloques de 512 bits, Y_0 Y_1, Y_{L-1}, por lo que la longitud extendido es L-512 bits.

Equivalentemente, el resultado es un múltiplo de 16 palabras de 32 bits. Podemos representarlo como, M(0...N-19), con N un entero múltiple de 16. De esta manera, N = L * 16.

Paso 3: Inicializar el buffer MD

Un buffer de 128 bits, es utilizado para soportar resultados intermedios y finales de la función Hash. El buffer puede ser representado como 4 registros de 32 bits (A,B,C,D). Estos registros están inicializados con los siguientes valores hexadecimales (los octetos menos significativos primero).

A = 01234567

B = 89ABCDEF

C = FEDCBA98

D = 76543210

Paso 4: Proceso De un mensaje en bloques de 512 bits (16 palabras de 32 bits)

El corazón del algoritmo es un modulo que consiste en 4 ciclos de procesamiento. Este modulo es llamado como H_{MD5} en la figura anterior y su lógica esta ilustrada en la siguiente figura. Los 4 ciclos tiene una estructura similar, pero cada uno utiliza una funcion primitiva logica diferente, referidas con **F, G, H** y **I**. En la figura los cuatro ciclos son etiquetados como f_F, f_G, f_H, f_I, para indicar que uno de los ciclos tiene en general, la misma funcion estructurada f, pero que depende de las diferentes funciones primitivas (**F,G,H,I**).

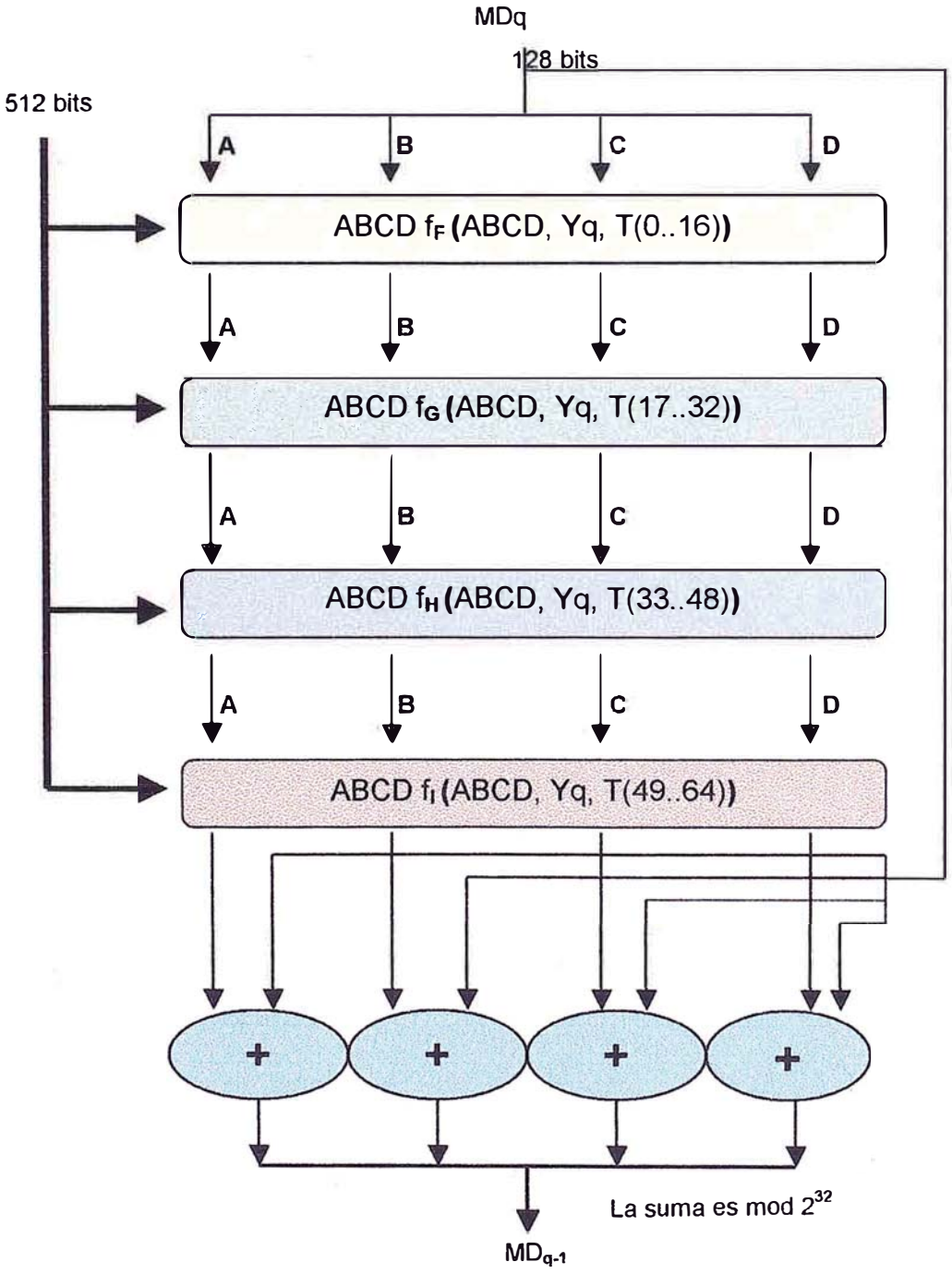


Figura 5: MD5, Actualización de un único bloque de 512 bits (HD)

Anotar que cada uno de los ciclos coge como entrada un bloque de 512 bits, para ser procesado (Yq) y el buffer de 128 bits, con valor ABCD, y actualiza el contenido del buffer. Cada ciclo también utiliza 16 palabras de 32 bits de una tabla de 64, T[1,….,64], construidos a partir de una función

Sinus. El elemento i-esimo de T, denotado como T[i], tiene el valor igual a la parte entera de 2^{32} * abs(sin(i)), donde i-esimo esta en radianes. Ya que el abs(sin(i)), es un numero entre 0 y 1, el elemento de T, es un entero que puede ser representado en 32 bits. La tabla proporciona aleatoriamente palabras de 32 bits, con lo cual debería eliminar cualquier irregularidad en la entrada de datos. La tabla siguiente lista los valores de T.

FF (a, b, c, d, M_j, t_j, s)

FF(a, b, c, d, M_0, D76AA478, 7)
FF(d, a, b, c, M_1, E8C7B756, 12)
FF(c, d, a, b, M_2, 242070DB, 17)
FF(b, c, d, a, M_3, C1BDCEEE, 22)
FF(a, b, c, d, M_4, F57C0FAF, 7)
FF(d, a, b, c, M_5, 4787C62A, 12)
FF(c, d, a, b, M_6, A8304613, 17)
FF(b, c, d, a, M_7, FD469501, 22)
FF(a, b, c, d, M_8, 698098D8, 7)
FF(d, a, b, c, M_9, 8B44F7AF, 12)
FF(c, d, a, b, M_{10}, FFFF5BB1, 17)
FF(b, c, d, a, M_{11}, 895CD7BE, 22)
FF(a, b, c, d, M_{12}, 6B901122, 7)
FF(d, a, b, c, M_{13}, FD987193, 12)
FF(c, d, a, b, M_{14}, A679438E, 17)
FF(b, c, d, a, M_{15}, 49B40821, 22)

GG (a, b, c, d, M_j, t_j, s)

GG(a, b, c, d, M_1, F61E2562, 5)
GG(d, a, b, c, M_6, C040B340, 9)
GG(c, d, a, b, M_{11}, 265E5A51, 14)
GG(b, c, d, a, M_0, E9B6C7AA, 20)
GG(a, b, c, d, M_5, D62F105D, 5)
GG(d, a, b, c, M_{10}, 02441453, 9)
GG(c, d, a, b, M_{15}, D8A1E681, 14)
GG(b, c, d, a, M_4, E7D3FBC8, 20)
GG(a, b, c, d, M_9, 21E1CDE6, 5)
GG(d, a, b, c, M_{14}, C33707D6, 9)
GG(c, d, a, b, M_3, F4D50D87, 14)
GG(b, c, d, a, M_8, 455A14ED, 20)
GG(a, b, c, d, M_{13}, A9E3E905, 5)
GG(d, a, b, c, M_2, FCEFA3F8, 9)
GG(c, d, a, b, M_7, 676F02D9, 14)
GG(b, c, d, a, M_{12}, 8D2A4C8A, 20)

HH (a, b, c, d, M_j, t_j, s)

HH(a, b, c, d, M_5, FFFA3942, 4)
HH(d, a, b, c, M_8, 8771F681, 11)
HH(c, d, a, b, M_{11}, 6D9D6122, 16)
HH(b, c, d, a, M_{14}, FDE5380C, 23)
HH(a, b, c, d, M_1, A4BEEA44, 4)
HH(d, a, b, c, M_4, 4BDECFA9, 11)
HH(c, d, a, b, M_7, F6BB4B60, 16)
HH(b, c, d, a, M_{10}, BEBFBC70, 23)
HH(a, b, c, d, M_{13}, 289B7EC6, 4)
HH(d, a, b, c, M_0, EAA127FA, 11)
HH(c, d, a, b, M_3, D4EF3085, 16)
HH(b, c, d, a, M_6, 04881D05, 23)
HH(a, b, c, d, M_9, D9D4D039, 4)
HH(d, a, b, c, M_{12}, E6DB99E5, 11)
HH(c, d, a, b, M_{15}, 1FA27CF8, 16)
HH(b, c, d, a, M_2, C4AC5665, 23)

II (a, b, c, d, M_j, t_j, s)

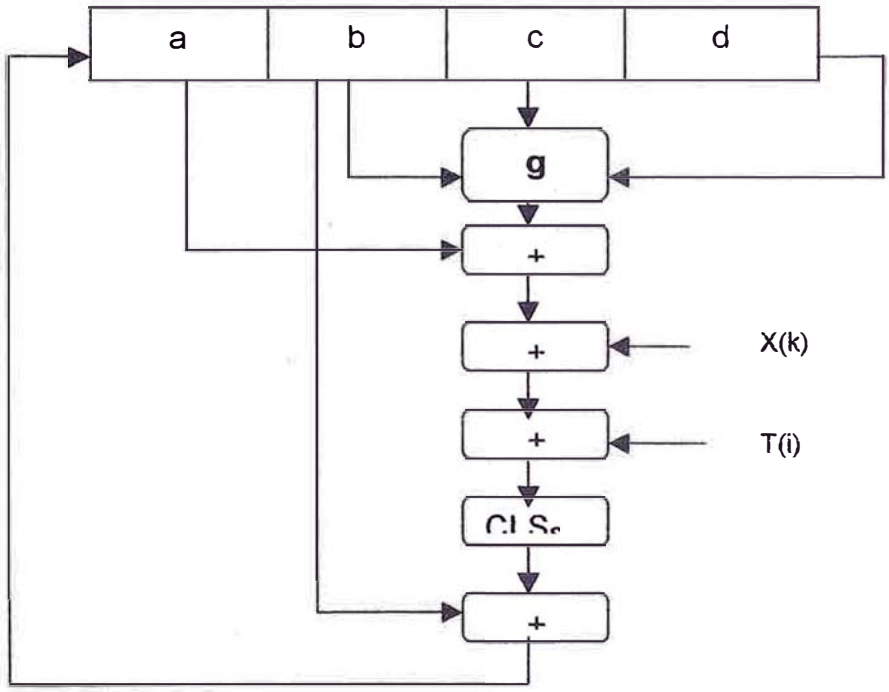
II(a, b, c, d, M_0, F4292244, 6)
II(d, a, b, c, M_7, 411AFF97, 10)
II(c, d, a, b, M_{14}, AB9423A7, 15)
II(b, c, d, a, M_5, FC93A039, 21)
II(a, b, c, d, M_{12}, 655B59C3, 6)
II(d, a, b, c, M_3, 8F0CCC92, 10)
II(c, d, a, b, M_{10}, FFEFF47D, 15)
II(b, c, d, a, M_1, 85845DD1, 21)
II(a, b, c, d, M_8, 6FA87E4F, 6)
II(d, a, b, c, M_{15}, FE2CE6E0, 10)
II(c, d, a, b, M_6, A3014314, 15)
II(b, c, d, a, M_{13}, 4E0811A1, 21)
II(a, b, c, d, M_4, F7537E82, 6)
II(d, a, b, c, M_{11}, BD3AF235, 10)
II(c, d, a, b, M_2, 2AD7D2BB, 15)
II(b, c, d, a, M_9, EB86D391, 21)

Tabla 7: Tabla T, construida a partir de la función Sinus.

Paso 5: Salida

Después de que todos los **L** bloques de 512 bits, hayan sido procesados, la salida de la L-esima fase es un mensaje resumen de 128 bits. A continuación se estudia con mas detalles la lógica de cada uno de los cuatro ciclos de procesamiento de un bloque de 512 bits. Cada ciclo consiste de una secuencia de 16 pasos por los cuales operan sobre un buffer **ABCD.** Cada paso sigue el esquema ilustrado en la siguiente figura.

Figura 6: Operación elemental del MD5: (abcd k s i)



Anotación: La suma (+) es mod 2^{32} . X(k) es los 32 bits de una palabra, de los 512 bits de un bloque de entrada.

$$\mathbf{a\ b + CLS_s\ (a + g(b, c, d) + X[k] + T[i])}$$

Donde:

a,b,c,d = las cuatro palabras del buffer, en un orden que varia a través de los pasos.

g = una de las funciones primitivas F, G, H, I.

CLS_S = desplazamiento circular a la izquierda de s bits sobre palabras de 32 bits.

X[k] = M[qx16+k] = la k-esima palabra de 32 bits en el q-esimo bloque de 512 bits de un mensaje.

T[i] = la i-esima palabra de 32 bits en un arreglo T.

+ = Suma mod 2^{32}.

Una de las cuatro funciones lógicas primitivas es utilizada por cada uno de los cuatro ciclos del algoritmo. Cada función primitiva toma tres palabras de 32 bits como entrada y produce palabra de 32 bits de salida. Cada función ejecuta una serie de operaciones lógicas; esto quiere decir, que n-esimo bit de salida es una función del n-esimo bit de las tres entradas. Las funciones pueden ser resumidas tal y como se muestra a continuación:

Ciclo	Función Primitiva g	g(b, c, d,)
f_F	F(b, c, d)	(b . c) v (~b .d)
f_G	G(b, c, d)	(b . c) v (b . ~d)
f_H	HG(b, c, d)	b # c # d
f_I	I(b, c, d)	c # (b . ~d)

Los operadores lógicos (AND, OR, NOT, XOR) son representados por los símbolos (., v, °, #). Función F es una función condicional: si b entonces c, sino d. Semejante, G puede empezar: si d, entonces b, sino c. La función H, produce un bit de paridad. La siguiente, es una tabla de verdad de las cuatro funciones.

Tabla 8: Tabla de verdad de las funciones lógicas

b	c	d	FG	H	I
0	0	0	0	0	1
0	0	1	1	0	1
0	1	0	0	1	1
0	1	1	1	0	1
1	0	0	0	1	1
1	0	1	0	1	1
1	1	0	1	1	0
1	1	1	1	1	0

La siguiente figura , adaptada de RFC 1321, define el procedimiento del algoritmo del paso 4. La expresión (w <<< s), denota el valor de 32 bits, obtenidos por la ejecución del desplazamiento circular a la izquierda (rotación) de **w** por **s** posiciones de bit. El arreglo de palabras de 32 bits, X[0..15], mantiene el valor actual del bloque de entrada, 512 bits, que serán procesados. Se debe mencionar que cada ciclo consiste en 16, de un total de 64 pasos. Cada palabra de 32 bits de entrada, es utilizada cuatro veces, una por ciclo y cada una de las 64 palabras de 32 bits pertenecientes a **T**, es utilizada exactamente una vez. También, anotar que por cada paso, solo uno de los cuatro bytes de buffer ABCD es actualizado. De aquí en adelante, cada byte del buffer, es actualizado 16 veces durante el ciclo, y entonces el decimoséptimo paso, el final produce la salida final del bloque.

Figura 7: Funcionamiento básico del algoritmo MD5.

Procesa bloques de 16 palabras de 32 bits (512 bits)

For i = 0 to N/16-1 do

/* Copiar bloque i en X.*/

For j = 0 to 15 do

Copiar en X[j] => M[i*16+j].

End

/* Guardar A como AA, B como BB, C como CC, y D como DD.*/

AA = A

BB = B

CC = C

DD = D

/* Ciclo No. 1 */

/* Con [abcd k s I] realiza operaciones a = b + ((a + F(b, c, d) + X[k] + T[i]) <<<s).*/ /* Hace las siguientes 16 operaciones */

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

/* Ciclo No. 2 */

/* Con [abcd k s I] realiza operaciones a = b + ((a + G(b, c, d) + X[k] + T[i]) <<<s).*/

/* Hace las siguientes 16 operaciones */

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

/* Ciclo No.3 */

/* Con [abcd k s t] realiza operaciones a = b + ((a + H(b, c, d) + X[k] + T[i]) <<<s).*/

/* Hace las siguientes 16 operaciones */

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

[ABCD 9 4 45] [DABC 12 11 46] [CDAB 15 16 47] [BCDA 2 23 48]

/* Ciclo No.4 */

/* Con [abcd k s t] realiza operaciones a = b + ((a + I(b, c, d) + X[k] + T[i]) <<<s).*/

/* Hace las siguientes 16 operaciones */

[ABCD 0 6 49] [DABC 7 10 50] [CDAB 14 15 51] [BCDA 5 21 52]

[ABCD 12 6 53] [DABC 3 10 54] [CDAB 10 15 55] [BCDA 1 21 56]

[ABCD 8 6 57] [DABC 15 10 58] [CDAB 6 15 59] [BCDA 13 21 60]

[ABCD 4 6 61] [DABC 11 10 62] [CDAB 2 15 63] [BCDA 9 21 64]

/* incrementa cada uno de los cuatro registros con el valor que tenia antes de empezar el proceso.*/

A = A + AA

B = B + BB

C = C + CC

D = D + DD

end /* final del ciclo no. i */

El comportamiento anterior del MD5, se resumen como sigue:

$MD_0 = IV$

$MD_{q+1} = MD_q + f_I [Y_q, f_H[Y_q, f_G[Y_q, MD_q]]]$

$MD = MD_{L-1}$

Donde :

IV = valor inicial del buffer ABCD, definido en el paso 3.

Y_q = El q-esimo bloque de 512 bits del mensaje.

L = El numero de bloques en el mensaje (incluyendo el relleno y el campo de longitud).

MD = valor final del mensaje resumen.

El algoritmo MD5, tiene la propiedad de que cada bit del código Hash, es una función de cadena de entrada. La compleja repetición de las funciones básicas (F, G, H, I), produce unos resultados que están bien mezclados; siendo desfavorable que dos mensajes elegidos aleatoriamente demuestren similares regularidades, tengan el mismo

código Hash. Rivest conjetura que el MD5, es tan fuerte como es posible para un Hash de 128 bits, es decir la probabilidad de obtener dos mensajes con el mismo resumen es de 2^{64} operaciones, mientras que la dificultad de encontrar un mensaje a partir de un resumen dado se calcula en alrededor de 2^{128} operaciones.

FIPS PUB 180-1

FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION
(Supersedes FIPS PUB 180 - 1993 May 11)

SECURE HASH STANDARD

CATEGORY: COMPUTER SECURITY

Computer Systems Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899

Issued April 17, 1995

U.S. Department of Commerce
Ronald H. Brown, Secretary

Technology Administration
Mary L. Good, Under Secretary for Technology

National Institute of Standards
and Technology
Arati Prabhakar, Director

Federal Information
Processing Standards Publication 180-1

1995 April 17

Announcing the

SECURE HASH STANDARD

Federal Information Processing Standards Publications (FIPS PUBS) are issued by the National Institute of Standards and Technology (NIST) after approval by the Secretary of Commerce pursuant to Section 111(d) of the Federal Property and Administrative Services Act of 1949 as amended by the Computer Security Act of 1987, Public Law 100-235.

Name of Standard: Secure Hash Standard.

Category of Standard: Computer Security.

Explanation: This Standard specifies a secure hash algorithm, SHA-1, for computing a condensed representation of a message or a data file. When a message of any length $< 2^{64}$ bits is input, the SHA-1 produces a 160-bit output called a message digest. The message digest can then be input to the Digital Signature Algorithm (DSA) which generates or verifies the signature for the message (see Figure 1). Signing the message digest rather than the message often improves the efficiency of the process because the message digest is usually much smaller in size than the message. The same hash algorithm must be used by the verifier of a digital signature as was used by the creator of the digital signature.

The SHA-1 is called secure because it is computationally infeasible to find a message which corresponds to a given message digest, or to find two different messages which produce the same message digest. Any change to a message in transit will, with very high probability, result in a different message digest, and the signature will fail to verify. SHA-1 is a technical revision of SHA (FIPS 180). A circular left shift operation has been added to the specifications in section 7, line b, page 9 of FIPS 180 and its equivalent in section 8, line c, page 10 of FIPS 180. This revision improves the security provided by this standard. The SHA-1 is based on principles similar to those used by Professor Ronald L. Rivest of MIT when designing the MD4 message digest algorithm[1], and is closely modelled after that algorithm.

---

[1]"The MD4 Message Digest Algorithm," Advances in Cryptology - CRYPTO '90 Proceedings, Springer-Verlag, 1991, pp. 303-311.

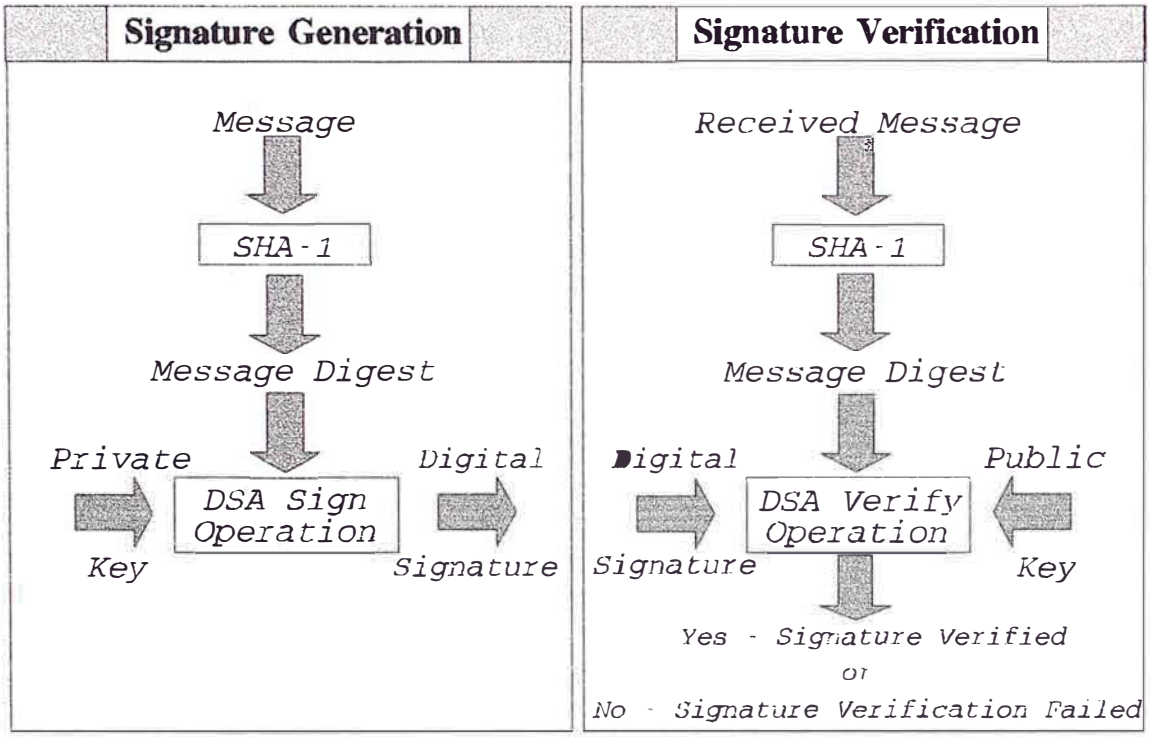


Figure 1: Using the SHA-1 with the DSA

Approving Authority: Secretary of Commerce.

Maintenance Agency: U.S. Department of Commerce, National Institute of Standards and Technology, Computer Systems Laboratory.

Applicability: This standard is applicable to all Federal departments and agencies for the protection of unclassified information that is not subject to section 2315 of Title 10, United States Code, or section 3502(2) of Title 44, United States Code. This standard is required for use with the Digital Signature Algorithm (DSA) as specified in the Digital Signature Standard (DSS) and whenever a secure hash algorithm is required for Federal applications. Private and commercial organizations are encouraged to adopt and use this standard.

Applications: The SHA-1 may be used with the DSA in electronic mail, electronic funds transfer, software distribution, data storage, and other applications which require data integrity assurance and data origin authentication. The SHA-1 may also be used whenever it is necessary to generate a condensed version of a message.

Implementations: The SHA-1 may be implemented in software, firmware, hardware, or any combination thereof. Only implementations of the SHA-1 that are validated by NIST will be considered as complying with this standard. Information about the requirements for validating implementations of this standard can be obtained from the National Institute of Standards and Technology, Computer Systems Laboratory, Attn: SHS Validation, Gaithersburg, MD 20899.

Export Control: Implementations of this standard are subject to Federal Government export controls as specified in Title 15, Code of Federal Regulations, Parts 768 through 799. Exporters are advised to contact the Department of Commerce, Bureau of Export Administration for more information.

Patents: Implementations of the SHA-1 in this standard may be covered by U.S. and foreign patents.

Implementation Schedule: This standard becomes effective October 2, 1995.

Specifications: Federal Information Processing Standard (FIPS) 180-1, Secure Hash Standard (affixed).

Cross Index:

a. FIPS PUB 46-2, Data Encryption Standard.

b. FIPS PUB 73, Guidelines for Security of Computer Applications.

c. FIPS PUB 140-1, Security Requirements for Cryptographic Modules.

d. FIPS PUB 186, Digital Signature Standard.

e. Federal Information Resources Management Regulations (FIRMR) subpart 201.20.303, Standards, and subpart 201.39.1002, Federal Standards.

Objectives: The objectives of this standard are to:

a. Specify the secure hash algorithm required for use with the Digital Signature Standard (FIPS 186) in the generation and verification of digital signatures;

b. Specify the secure hash algorithm to be used whenever a secure hash algorithm is required for Federal applications; and

c. Encourage the adoption and use of the specified secure hash algorithm by private and commercial organizations.

Qualifications: While it is the intent of this standard to specify a secure hash algorithm, conformance to this standard does not assure that a particular implementation is secure. The responsible authority in each agency or department shall assure that an overall implementation provides an acceptable level of security. This standard will be reviewed every five years in order to assess its adequacy.

Waiver Procedure: Under certain exceptional circumstances, the heads of Federal departments and agencies may approve waivers to Federal Information Processing Standards (FIPS). The head of such agency may redelegate such authority only to a senior official designated pursuant to section 3506(b) of Title 44, United States Code. Waiver shall be granted only when:

a. Compliance with a standard would adversely affect the accomplishment of the mission of an operator of a Federal computer system; or

b. Compliance with a standard would cause a major adverse financial impact on the operator which is not offset by Governmentwide savings.

Agency heads may act upon a written waiver request containing the information detailed above. Agency heads may also act without a written waiver request when they determine that conditions for meeting the standard cannot be met. Agency heads may approve waivers only by a written decision which explains the basis on which the agency head made the required finding(s). A copy of each decision, with procurement sensitive or classified portions clearly identified, shall be sent to: National Institute of Standards and Technology; ATTN: FIPS Waiver Decisions, Technology Building, Room B-154, Gaithersburg, MD 20899.

In addition, notice of each waiver granted and each delegation of authority to approve waivers shall be sent promptly to the Committee on Government Operations of the House of Representatives and the Committee on Governmental Affairs of the Senate and shall be published promptly in the Federal Register.

When the determination on a waiver applies to the procurement of equipment and/or services, a notice of the waiver determination must be published in the Commerce Business Daily as a part of the notice of solicitation for offers of an acquisition or, if the waiver determination is made after that notice is published, by amendment to such notice.

A copy of the waiver, any supporting documents, the document approving the waiver and any accompanying documents, with such deletions as the agency is authorized and decides to make under 5 United States Code Section 552(b), shall be part of the procurement documentation and retained by the agency.

Where to Obtain Copies of the Standard: Copies of this publication are for sale by the National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161. When ordering, refer to Federal Information Processing Standards Publication 180-1 (FIPSPUB180-1), and identify the title. When microfiche is desired, this should be specified. Prices are published by NTIS in current catalogs and other issuances. Payment may be made by check, money order, deposit account or charged to a credit card accepted by NTIS.

APPENDIX A. A SAMPLE MESSAGE AND ITS MESSAGE DIGEST

This appendix is for informational purposes only and is not required to meet the standard.

Let the message be the ASCII binary-coded form of "abc", i.e.,

```
01100001  01100010  01100011.
```

This message has length $l = 24$. In step (a) of Section 4, we append "1". In step (b) we append 423 "0"s. In step (c) we append hex `00000000 00000018`, the 2-word representation of 24. Thus the final padded message consists of one block, so that n = 1 in the notation of Section 4.

The initial hex values of $\{H_i\}$ are

H_0 = 67452301

H_1 = EFCDAB89

H_2 = 98BADCFE

H_3 = 10325476

H_4 = C3D2E1F0.

Start processing block 1. The words of block 1 are

```
W[0]  = 61626380
W[1]  = 00000000
W[2]  = 00000000
W[3]  = 00000000
W[4]  = 00000000
W[5]  = 00000000
W[6]  = 00000000
W[7]  = 00000000
W[8]  = 00000000
W[9]  = 00000000
W[10] = 00000000
W[11] = 00000000
W[12] = 00000000
W[13] = 00000000
W[14] = 00000000
W[15] = 00000018.
```

The hex values of A,B,C,D,E after pass t of the "for t = 0 to 79" loop (step (d) of Section 7 or step (c) of Section 8) are

	A	B	C	D	E
t = 0:	0116FC33	67452301	7BF36AE2	98BADCFE	10325476
t = 1:	8990536D	0116FC33	59D148C0	7BF36AE2	98BADCFE
t = 2:	A1390F08	8990536D	C045BF0C	59D148C0	7BF36AE2
t = 3:	CDD8E11B	A1390F08	626414DB	C045BF0C	59D148C0
t = 4:	CFD499DE	CDD8E11B	284E43C2	626414DB	C045BF0C
t = 5:	3FC7CA40	CFD499DE	F3763846	284E43C2	626414DB
t = 6:	993E30C1	3FC7CA40	B3F52677	F3763846	284E43C2
t = 7:	9E8C07D4	993E30C1	0FF1F290	B3F52677	F3763846
t = 8:	4B6AE328	9E8C07D4	664F8C30	0FF1F290	B3F52677
t = 9:	8351F929	4B6AE328	27A301F5	664F8C30	0FF1F290
t = 10:	FBDA9E89	8351F929	12DAB8CA	27A301F5	664F8C30
t = 11:	63188FE4	FBDA9E89	60D47E4A	12DAB8CA	27A301F5
t = 12:	4607B664	63188FE4	7EF6A7A2	60D47E4A	12DAB8CA
t = 13:	9128F695	4607B664	18C623F9	7EF6A7A2	60D47E4A
t = 14:	196BEE77	9128F695	1181ED99	18C623F9	7EF6A7A2
t = 15:	20BDD62F	196BEE77	644A3DA5	1181ED99	18C623F9
t = 16:	4E925823	20BDD62F	C65AFB9D	644A3DA5	1181ED99
t = 17:	82AA6728	4E925823	C82F758B	C65AFB9D	644A3DA5
t = 18:	DC64901D	82AA6728	D3A49608	C82F758B	C65AFB9D
t = 19:	FD9E1D7D	DC64901D	20AA99CA	D3A49608	C82F758B
t = 20:	1A37B0CA	FD9E1D7D	77192407	20AA99CA	D3A49608
t = 21:	33A23BFC	1A37B0CA	7F67875F	77192407	20AA99CA
t = 22:	21283486	33A23BFC	868DEC32	7F67875F	77192407
t = 23:	D541F12D	21283486	0CE88EFF	868DEC32	7F67875F
t = 24:	C7567DC6	D541F12D	884A0D21	0CE88EFF	868DEC32
t = 25:	48413BA4	C7567DC6	75507C4B	884A0D21	0CE88EFF
t = 26:	BE35FBD5	48413BA4	B1D59F71	75507C4B	884A0D21
t = 27:	4AA84D97	BE35FBD5	12104EE9	B1D59F71	75507C4B
t = 28:	8370B52E	4AA84D97	6F8D7EF5	12104EE9	B1D59F71
t = 29:	C5FBAF5D	8370B52E	D2AA1365	6F8D7EF5	12104EE9
t = 30:	1267B407	C5FBAF5D	A0DC2D4B	D2AA1365	6F8D7EF5
t = 31:	3B845D33	1267B407	717EEBD7	A0DC2D4B	D2AA1365
t = 32:	046FAA0A	3B845D33	C499ED01	717EEBD7	A0DC2D4B
t = 33:	2C0EBC11	046FAA0A	CEE1174C	C499ED01	717EEBD7
t = 34:	21796AD4	2C0EBC11	811BEA82	CEE1174C	C499ED01
t = 35:	DCBBB0CB	21796AD4	4B03AF04	811BEA82	CEE1174C
t = 36:	0F511FD8	DCBBB0CB	085E5AB5	4B03AF04	811BEA82
t = 37:	DC63973F	0F511FD8	F72EEC32	085E5AB5	4B03AF04
t = 38:	4C986405	DC63973F	03D447F6	F72EEC32	085E5AB5
t = 39:	32DE1CBA	4C986405	F718E5CF	03D447F6	F72EEC32
t = 40:	FC87DEDF	32DE1CBA	53261901	F718E5CF	03D447F6
t = 41:	970A0D5C	FC87DEDF	8CB7872E	53261901	F718E5CF
t = 42:	7F193DC5	970A0D5C	FF21F7B7	8CB7872E	53261901
t = 43:	EE1B1AAF	7F193DC5	25C28357	FF21F7B7	8CB7872E
t = 44:	40F28E09	EE1B1AAF	5FC64F71	25C28357	FF21F7B7
t = 45:	1C51E1F2	40F28E09	FB86C6AB	5FC64F71	25C28357
t = 46:	A01B846C	1C51E1F2	503CA382	FB86C6AB	5FC64F71
t = 47:	BEAD02CA	A01B846C	8714787C	503CA382	FB86C6AB

```
t = 48: BAF39337    BEAD02CA    2806E11B    8714787C    503CA382
t = 49: 120731C5    BAF39337    AFAB40B2    2806E11B    8714787C
t = 50: 641DB2CE    120731C5    EEBCE4CD    AFAB40B2    2806E11B
t = 51: 3847AD66    641DB2CE    4481CC71    EEBCE4CD    AFAB40B2
t = 52: E490436D    3847AD66    99076CB3    4481CC71    EEBCE4CD
t = 53: 27E9F1D8    E490436D    8E11EB59    99076CB3    4481CC71
t = 54: 7B71F76D    27E9F1D8    792410DB    8E11EB59    99076CB3
t = 55: 5E6456AF    7B71F76D    09FA7C76    792410DB    8E11EB59
t = 56: C846093F    5E6456AF    5EDC7DDB    09FA7C76    792410DB
t = 57: D262FF50    C846093F    D79915AB    5EDC7DDB    09FA7C76
t = 58: 09D785FD    D262FF50    F211824F    D79915AB    5EDC7DDB
t = 59: 3F52DE5A    09D785FD    3498BFD4    F211824F    D79915AB
t = 60: D756C147    3F52DE5A    4275E17F    3498BFD4    F211824F
t = 61: 548C9CB2    D756C147    8FD4B796    4275E17F    3498BFD4
t = 62: B66C020B    548C9CB2    F5D5B051    8FD4B796    4275E17F
t = 63: 6B61C9E1    B66C020B    9523272C    F5D5B051    8FD4B796
t = 64: 19DFA7AC    6B61C9E1    ED9B0082    9523272C    F5D5B051
t = 65: 101655F9    19DFA7AC    5AD87278    ED9B0082    9523272C
t = 66: 0C3DF2B4    101655F9    0677E9EB    5AD87278    ED9B0082
t = 67: 78DD4D2B    0C3DF2B4    4405957E    0677E9EB    5AD87278
t = 68: 497093C0    78DD4D2B    030F7CAD    4405957E    0677E9EB
t = 69: 3F2588C2    497093C0    DE37534A    030F7CAD    4405957E
t = 70: C199F8C7    3F2588C2    125C24F0    DE37534A    030F7CAD
t = 71: 39859DE7    C199F8C7    8FC96230    125C24F0    DE37534A
t = 72: EDB42DE4    39859DE7    F0667E31    8FC96230    125C24F0
t = 73: 11793F6F    EDB42DE4    CE616779    F0667E31    8FC96230
t = 74: 5EE76897    11793F6F    3B6D0B79    CE616779    F0667E31
t = 75: 63F7DAB7    5EE76897    C45E4FDB    3B6D0B79    CE616779
t = 76: A079B7D9    63F7DAB7    D7B9DA25    C45E4FDB    3B6D0B79
t = 77: 860D21CC    A079B7D9    D8FDF6AD    D7B9DA25    C45E4FDB
t = 78: 5738D5E1    860D21CC    681E6DF6    D8FDF6AD    D7B9DA25
t = 79: 42541B35    5738D5E1    21834873    681E6DF6    D8FDF6AD.
```

Block 1 has been processed. The values of $\{H_i\}$ are

H_0 = 67452301 + 42541B35 = A9993E36

H_1 = EFCDAB89 + 5738D5E1 = 4706816A

H_2 = 98BADCFE + 21834873 = BA3E2571

H_3 = 10325476 + 681E6DF6 = 7850C26C

H_4 = C3D2E1F0 + D8FDF6AD = 9CD0D89D.

Message digest = A9993E36 4706816A BA3E2571 7850C26C 9CD0D89D

APPENDIX B. A SECOND SAMPLE MESSAGE AND ITS MESSAGE DIGEST

This appendix is for informational purposes only and is not required to meet the standard.

Let the message be the binary-coded form (cf. Appendix A) of the ASCII string

"abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq".

Since each of the 56 characters is converted to 8 bits, the length of the message is $l = 448$. In step (a) of Section 4, we append "1". In step (b) we append 511 "0"s. In step (c) we append the 2-word representation of 448, i.e., hex 00000000 000001C0. This gives n = 2.

The initial hex values of $\{H_i\}$ are

H_0 = 67452301

H_1 = EFCDAB89

H_2 = 98BADCFE

H_3 = 10325476

H_4 = C3D2E1F0.

Start processing block 1. The words of block 1 are

```
W[0]  = 61626364
W[1]  = 62636465
W[2]  = 63646566
W[3]  = 64656667
W[4]  = 65666768
W[5]  = 66676869
W[6]  = 6768696A
W[7]  = 68696A6B
W[8]  = 696A6B6C
W[9]  = 6A6B6C6D
W[10] = 6B6C6D6E
W[11] = 6C6D6E6F
W[12] = 6D6E6F70
W[13] = 6E6F7071
W[14] = 80000000
W[15] = 00000000.
```

The hex values of A,B,C,D,E after pass t of the "for t = 0 to 79" loop (step (d) of Section 7 or step (c) of Section 8) are

	A	B	C	D	E
t = 0:	0116FC17	67452301	7BF36AE2	98BADCFE	10325476
t = 1:	EBF3B452	0116FC17	59D148C0	7BF36AE2	98BADCFE
t = 2:	5109913A	EBF3B452	C045BF05	59D148C0	7BF36AE2
t = 3:	2C4F6EAC	5109913A	BAFCED14	C045BF05	59D148C0
t = 4:	33F4AE5B	2C4F6EAC	9442644E	BAFCED14	C045BF05
t = 5:	96B85189	33F4AE5B	0B13DBAB	9442644E	BAFCED14
t = 6:	DB04CB58	96B85189	CCFD2B96	0B13DBAB	9442644E
t = 7:	45833F0F	DB04CB58	65AE1462	CCFD2B96	0B13DBAB
t = 8:	C565C35E	45833F0F	36C132D6	65AE1462	CCFD2B96
t = 9:	6350AFDA	C565C35E	D160CFC3	36C132D6	65AE1462
t = 10:	8993EA77	6350AFDA	B15970D7	D160CFC3	36C132D6
t = 11:	E19ECAA2	8993EA77	98D42BF6	B15970D7	D160CFC3
t = 12:	8603481E	E19ECAA2	E264FA9D	98D42BF6	B15970D7
t = 13:	32F94A85	8603481E	B867B2A8	E264FA9D	98D42BF6
t = 14:	B2E7A8BE	32F94A85	A180D207	B867B2A8	E264FA9D
t = 15:	42637E39	B2E7A8BE	4CBE52A1	A180D207	B867B2A8
t = 16:	6B068048	42637E39	ACB9EA2F	4CBE52A1	A180D207
t = 17:	426B9C35	6B068048	5098DF8E	ACB9EA2F	4CBE52A1
t = 18:	944B1BD1	426B9C35	1AC1A012	5098DF8E	ACB9EA2F
t = 19:	6C445652	944B1BD1	509AE70D	1AC1A012	5098DF8E
t = 20:	95836DA5	6C445652	6512C6F4	509AE70D	1AC1A012
t = 21:	09511177	95836DA5	9B111594	6512C6F4	509AE70D
t = 22:	E2B92DC4	09511177	6560DB69	9B111594	6512C6F4
t = 23:	FD224575	E2B92DC4	C254445D	6560DB69	9B111594
t = 24:	EEB82D9A	FD224575	38AE4B71	C254445D	6560DB69
t = 25:	5A142C1A	EEB82D9A	7F48915D	38AE4B71	C254445D
t = 26:	2972F7C7	5A142C1A	BBAE0B66	7F48915D	38AE4B71
t = 27:	D526A644	2972F7C7	96850B06	BBAE0B66	7F48915D
t = 28:	E1122421	D526A644	CA5CBDF1	96850B06	BBAE0B66
t = 29:	05B457B2	E1122421	3549A991	CA5CBDF1	96850B06
t = 30:	A9C84BEC	05B457B2	78448908	3549A991	CA5CBDF1
t = 31:	52E31F60	A9C84BEC	816D15EC	78448908	3549A991
t = 32:	5AF3242C	52E31F60	2A7212FB	816D15EC	78448908
t = 33:	31C756A9	5AF3242C	14B8C7D8	2A7212FB	816D15EC
t = 34:	E9AC987C	31C756A9	16BCC90B	14B8C7D8	2A7212FB
t = 35:	AB7C32EE	E9AC987C	4C71D5AA	16BCC90B	14B8C7D8
t = 36:	5933FC99	AB7C32EE	3A6B261F	4C71D5AA	16BCC90B
t = 37:	43F87AE9	5933FC99	AADF0CBB	3A6B261F	4C71D5AA
t = 38:	24957F22	43F87AE9	564CFF26	AADF0CBB	3A6B261F
t = 39:	ADEB7478	24957F22	50FE1EBA	564CFF26	AADF0CBB
t = 40:	D70E5010	ADEB7478	89255FC8	50FE1EBA	564CFF26
t = 41:	79BCFB08	D70E5010	2B7ADD1E	89255FC8	50FE1EBA
t = 42:	F9BCB8DE	79BCFB08	35C39404	2B7ADD1E	89255FC8
t = 43:	633E9561	F9BCB8DE	1E6F3EC2	35C39404	2B7ADD1E
t = 44:	98C1EA64	633E9561	BE6F2E37	1E6F3EC2	35C39404
t = 45:	C6EA241E	98C1EA64	58CFA558	BE6F2E37	1E6F3EC2
t = 46:	A2AD4F02	C6EA241E	26307A99	58CFA558	BE6F2E37
t = 47:	C8A69090	A2AD4F02	B1BA8907	26307A99	58CFA558
t = 48:	88341600	C8A69090	A8AB53C0	B1BA8907	26307A99
t = 49:	7E846F58	88341600	3229A424	A8AB53C0	B1BA8907

```
t = 50: 86E358BA    7E846F58    220D0580    3229A424    A8AB53C0
t = 51: 8D2E76C8    86E358BA    1FA11BD6    220D0580    3229A424
t = 52: CE892E10    8D2E76C8    A1B8D62E    1FA11BD6    220D0580
t = 53: EDEA95B1    CE892E10    234B9DB2    A1B8D62E    1FA11BD6
t = 54: 36D1230A    EDEA95B1    33A24B84    234B9DB2    A1B8D62E
t = 55: 776C3910    36D1230A    7B7AA56C    33A24B84    234B9DB2
t = 56: A681B723    776C3910    8DB448C2    7B7AA56C    33A24B84
t = 57: AC0A794F    A681B723    1DDB0E44    8DB448C2    7B7AA56C
t = 58: F03D3782    AC0A794F    E9A06DC8    1DDB0E44    8DB448C2
t = 59: 9EF775C3    F03D3782    EB029E53    E9A06DC8    1DDB0E44
t = 60: 36254B13    9EF775C3    BC0F4DE0    EB029E53    E9A06DC8
t = 61: 4080D4DC    36254B13    E7BDDD70    BC0F4DE0    EB029E53
t = 62: 2BFAF7A8    4080D4DC    CD8952C4    E7BDDD70    BC0F4DE0
t = 63: 513F9CA0    2BFAF7A8    10203537    CD8952C4    E7BDDD70
t = 64: E5895C81    513F9CA0    0AFEBDEA    10203537    CD8952C4
t = 65: 1037D2D5    E5895C81    144FE728    0AFEBDEA    10203537
t = 66: 14A82DA9    1037D2D5    79625720    144FE728    0AFEBDEA
t = 67: 6D17C9FD    14A82DA9    440DF4B5    79625720    144FE728
t = 68: 2C7B07BD    6D17C9FD    452A0B6A    440DF4B5    79625720
t = 69: FDF6EFFF    2C7B07BD    5B45F27F    452A0B6A    440DF4B5
t = 70: 112B96E3    FDF6EFFF    4B1EC1EF    5B45F27F    452A0B6A
t = 71: 84065712    112B96E3    FF7DBBFF    4B1EC1EF    5B45F27F
t = 72: AB89FB71    84065712    C44AE5B8    FF7DBBFF    4B1EC1EF
t = 73: C5210E35    AB89FB71    A10195C4    C44AE5B8    FF7DBBFF
t = 74: 352D9F4B    C5210E35    6AE27EDC    A10195C4    C44AE5B8
t = 75: 1A0E0E0A    352D9F4B    7148438D    6AE27EDC    A10195C4
t = 76: D0D47349    1A0E0E0A    CD4B67D2    7148438D    6AE27EDC
t = 77: AD38620D    D0D47349    86838382    CD4B67D2    7148438D
t = 78: D3AD7C25    AD38620D    74351CD2    86838382    CD4B67D2
t = 79: 8CE34517    D3AD7C25    6B4E1883    74351CD2    86838382.
```

Block 1 has been processed. The values of $\{H_i\}$ are

$H_0 = 67452301 + 8CE34517 = F4286818$

$H_1 = EFCDAB89 + D3AD7C25 = C37B27AE$

$H_2 = 98BADCFE + 6B4E1883 = 0408F581$

$H_3 = 10325476 + 74351CD2 = 84677148$

$H_4 = C3D2E1F0 + 86838382 = 4A566572.$

Start processing block 2. The words of block 2 are

```
W[0] = 00000000
W[1] = 00000000
W[2] = 00000000
W[3] = 00000000
W[4] = 00000000
W[5] = 00000000
```

```
W[6]  = 00000000
W[7]  = 00000000
W[8]  = 00000000
W[9]  = 00000000
W[10] = 00000000
W[11] = 00000000
W[12] = 00000000
W[13] = 00000000
W[14] = 00000000
W[15] = 000001C0.
```

The hex values of A,B,C,D,E after pass t of the for "t = 0 to 79" loop (step (d) of Section 7 or step (c) of Section 8) are

	A	B	C	D	E
t = 0:	2DF257E9	F4286818	B0DEC9EB	0408F581	84677148
t = 1:	4D3DC58F	2DF257E9	3D0A1A06	B0DEC9EB	0408F581
t = 2:	C352BB05	4D3DC58F	4B7C95FA	3D0A1A06	B0DEC9EB
t = 3:	EEF743C6	C352BB05	D34F7163	4B7C95FA	3D0A1A06
t = 4:	41E34277	EEF743C6	70D4AEC1	D34F7163	4B7C95FA
t = 5:	5443915C	41E34277	BBBDD0F1	70D4AEC1	D34F7163
t = 6:	E7FA0377	5443915C	D078D09D	BBBDD0F1	70D4AEC1
t = 7:	C6946813	E7FA0377	1510E457	D078D09D	BBBDD0F1
t = 8:	FDDE1DE1	C6946813	F9FE80DD	1510E457	D078D09D
t = 9:	B8538ACA	FDDE1DE1	F1A51A04	F9FE80DD	1510E457
t = 10:	6BA94F63	B8538ACA	7F778778	F1A51A04	F9FE80DD
t = 11:	43A2792F	6BA94F63	AE14E2B2	7F778778	F1A51A04
t = 12:	FECD7BBF	43A2792F	DAEA53D8	AE14E2B2	7F778778
t = 13:	A2604CA8	FECD7BBF	D0E89E4B	DAEA53D8	AE14E2B2
t = 14:	258B0BAA	A2604CA8	FFB35EEF	D0E89E4B	DAEA53D8
t = 15:	D9772360	258B0BAA	2898132A	FFB35EEF	D0E89E4B
t = 16:	5507DB6E	D9772360	8962C2EA	2898132A	FFB35EEF
t = 17:	A51B58BC	5507DB6E	365DC8D8	8962C2EA	2898132A
t = 18:	C2EB709F	A51B58BC	9541F6DB	365DC8D8	8962C2EA
t = 19:	D8992153	C2EB709F	2946D62F	9541F6DB	365DC8D8
t = 20:	37482F5F	D8992153	F0BADC27	2946D62F	9541F6DB
t = 21:	EE8700BD	37482F5F	F6264854	F0BADC27	2946D62F
t = 22:	9AD594B9	EE8700BD	CDD20BD7	F6264854	F0BADC27
t = 23:	8FBAA5B9	9AD594B9	7BA1C02F	CDD20BD7	F6264854
t = 24:	88FB5867	8FBAA5B9	66B5652E	7BA1C02F	CDD20BD7
t = 25:	EEC50521	88FB5867	63EEA96E	66B5652E	7BA1C02F
t = 26:	50BCE434	EEC50521	E23ED619	63EEA96E	66B5652E
t = 27:	5C416DAF	50BCE434	7BB14148	E23ED619	63EEA96E
t = 28:	2429BE5F	5C416DAF	142F390D	7BB14148	E23ED619
t = 29:	0A2FB108	2429BE5F	D7105B6B	142F390D	7BB14148
t = 30:	17986223	0A2FB108	C90A6F97	D7105B6B	142F390D
t = 31:	8A4AF384	17986223	028BEC42	C90A6F97	D7105B6B
t = 32:	6B629993	8A4AF384	C5E61888	028BEC42	C90A6F97
t = 33:	F15F04F3	6B629993	2292BCE1	C5E61888	028BEC42
t = 34:	295CC25B	F15F04F3	DAD8A664	2292BCE1	C5E61888

t = 35:	696DA404	295CC25B	FC57C13C	DAD8A664	2292BCE1
t = 36:	CEF5AE12	696DA404	CA573096	FC57C13C	DAD8A664
t = 37:	87D5B80C	CEF5AE12	1A5B6901	CA573096	FC57C13C
t = 38:	84E2A5F2	87D5B80C	B3BD6B84	1A5B6901	CA573096
t = 39:	03BB6310	84E2A5F2	21F56E03	B3BD6B84	1A5B6901
t = 40:	C2D8F75F	03BB6310	A138A97C	21F56E03	B3BD6B84
t = 41:	BFB25768	C2D8F75F	00EED8C4	A138A97C	21F56E03
t = 42:	28589152	BFB25768	F0B63DD7	00EED8C4	A138A97C
t = 43:	EC1D3D61	28589152	2FEC95DA	F0B63DD7	00EED8C4
t = 44:	3CAED7AF	EC1D3D61	8A162454	2FEC95DA	F0B63DD7
t = 45:	C3D033EA	3CAED7AF	7B074F58	8A162454	2FEC95DA
t = 46:	7316056A	C3D033EA	CF2BB5EB	7B074F58	8A162454
t = 47:	46F93B68	7316056A	B0F40CFA	CF2BB5EB	7B074F58
t = 48:	DC8E7F26	46F93B68	9CC5815A	B0F40CFA	CF2BB5EB
t = 49:	850D411C	DC8E7F26	11BE4EDA	9CC5815A	B0F40CFA
t = 50:	7E4672C0	850D411C	B7239FC9	11BE4EDA	9CC5815A
t = 51:	89FBD41D	7E4672C0	21435047	B7239FC9	11BE4EDA
t = 52:	1797E228	89FBD41D	1F919CB0	21435047	B7239FC9
t = 53:	431D65BC	1797E228	627EF507	1F919CB0	21435047
t = 54:	2BDBB8CB	431D65BC	05E5F88A	627EF507	1F919CB0
t = 55:	6DA72E7F	2BDBB8CB	10C7596F	05E5F88A	627EF507
t = 56:	A8495A9B	6DA72E7F	CAF6EE32	10C7596F	05E5F88A
t = 57:	E785655A	A8495A9B	DB69CB9F	CAF6EE32	10C7596F
t = 58:	5B086C42	E785655A	EA1256A6	DB69CB9F	CAF6EE32
t = 59:	A65818F7	5B086C42	B9E15956	EA1256A6	DB69CB9F
t = 60:	7AAB101B	A65818F7	96C21B10	B9E15956	EA1256A6
t = 61:	93614C9C	7AAB101B	E996063D	96C21B10	B9E15956
t = 62:	F66D9BF4	93614C9C	DEAAC406	E996063D	96C21B10
t = 63:	D504902B	F66D9BF4	24D85327	DEAAC406	E996063D
t = 64:	60A9DA62	D504902B	3D9B66FD	24D85327	DEAAC406
t = 65:	8B687819	60A9DA62	F541240A	3D9B66FD	24D85327
t = 66:	083E90C3	8B687819	982A7698	F541240A	3D9B66FD
t = 67:	F6226BBF	083E90C3	62DA1E06	982A7698	F541240A
t = 68:	76C0563B	F6226BBF	C20FA430	62DA1E06	982A7698
t = 69:	989DD165	76C0563B	FD889AEF	C20FA430	62DA1E06
t = 70:	8B2C7573	989DD165	DDB0158E	FD889AEF	C20FA430
t = 71:	AE1B8E7B	8B2C7573	66277459	DDB0158E	FD889AEF
t = 72:	CA1840DE	AE1B8E7B	E2CB1D5C	66277459	DDB0158E
t = 73:	16F3BABB	CA1840DE	EB86E39E	E2CB1D5C	66277459
t = 74:	D28D83AD	16F3BABB	B2861037	EB86E39E	E2CB1D5C
t = 75:	6BC02DFE	D28D83AD	C5BCEEAE	B2861037	EB86E39E
t = 76:	D3A6E275	6BC02DFE	74A360EB	C5BCEEAE	B2861037
t = 77:	DA955482	D3A6E275	9AF00B7F	74A360EB	C5BCEEAE
t = 78:	58C0AAC0	DA955482	74E9B89D	9AF00B7F	74A360EB
t = 79:	906FD62C	58C0AAC0	B6A55520	74E9B89D	9AF00B7F.

Block 2 has been processed. The values of $\{H_i\}$ are

H_0 = F4286818 + 906FD62C = 84983E44

H_1 = C37B27AE + 58C0AAC0 = 1C3BD26E

H_2 = 0408F581 + B6A55520 = BAAE4AA1

H_3 84677148 + 74E9B89D = F95129E5

H_4 4A566572 + 9AF00B7F = E54670F1.

Message digest = 84983E44 1C3BD26E BAAE4AA1 F95129E5 E54670F1

APPENDIX C. A THIRD SAMPLE MESSAGE AND ITS MESSAGE DIGEST

This appendix is for informational purposes only and is not required to meet the standard.

Let the message be the binary-coded form of the ASCII string which consists of 1,000,000 repetitions of "a".

Message digest = `34AA973C D4C4DAA4 F61EEB2B DBAD2731 6534016F`

Federal Information

Processing Standards Publication 197

November 26, 2001

Announcing the

ADVANCED ENCRYPTION STANDARD (AES)

Federal Information Processing Standards Publications (FIPS PUBS) are issued by the National Institute of Standards and Technology (NIST) after approval by the Secretary of Commerce pursuant to Section 5131 of the Information Technology Management Reform Act of 1996 (Public Law 104-106) and the Computer Security Act of 1987 (Public Law 100-235).

1. **Name of Standard.** Advanced Encryption Standard (AES) (FIPS PUB 197).

2. **Category of Standard.** Computer Security Standard, Cryptography.

3. **Explanation.** The Advanced Encryption Standard (AES) specifies a FIPS-approved cryptographic algorithm that can be used to protect electronic data. The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. Encryption converts data to an unintelligible form called ciphertext; decrypting the ciphertext converts the data back into its original form, called plaintext.

The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits.

4. **Approving Authority.** Secretary of Commerce.

5. **Maintenance Agency.** Department of Commerce, National Institute of Standards and Technology, Information Technology Laboratory (ITL).

6. **Applicability.** This standard may be used by Federal departments and agencies when an agency determines that sensitive (unclassified) information (as defined in P. L. 100-235) requires cryptographic protection.

Other FIPS-approved cryptographic algorithms may be used in addition to, or in lieu of, this standard. Federal agencies or departments that use cryptographic devices for protecting classified information can use those devices for protecting sensitive (unclassified) information in lieu of this standard.

In addition, this standard may be adopted and used by non-Federal Government organizations. Such use is encouraged when it provides the desired security for commercial and private organizations.

7. **Specifications** Federal Information Processing Standard (FIPS) 197, Advanced Encryption Standard (AES) (affixed).

8. **Implementations.** The algorithm specified in this standard may be implemented in software, firmware, hardware, or any combination thereof. The specific implementation may depend on several factors such as the application, the environment, the technology used, etc. The algorithm shall be used in conjunction with a FIPS approved or NIST recommended mode of operation. Object Identifiers (OIDs) and any associated parameters for AES used in these modes are available at the Computer Security Objects Register (CSOR), located at http://csrc.nist.gov/csor/ [2].

Implementations of the algorithm that are tested by an accredited laboratory and validated will be considered as complying with this standard. Since cryptographic security depends on many factors besides the correct implementation of an encryption algorithm, Federal Government employees, and others, should also refer to NIST Special Publication 800-21, *Guideline for Implementing Cryptography in the Federal Government*, for additional information and guidance (NIST SP 800-21 is available at http://csrc.nist.gov/publications/).

9. **Implementation Schedule.** This standard becomes effective on May 26, 2002.

10. **Patents.** Implementations of the algorithm specified in this standard may be covered by U.S. and foreign patents.

11. **Export Control.** Certain cryptographic devices and technical data regarding them are subject to Federal export controls. Exports of cryptographic modules implementing this standard and technical data regarding them must comply with these Federal regulations and be licensed by the Bureau of Export Administration of the U.S. Department of Commerce. Applicable Federal government export controls are specified in Title 15, Code of Federal Regulations (CFR) Part 740.17; Title 15, CFR Part 742; and Title 15, CFR Part 774, Category 5, Part 2.

12. **Qualifications.** NIST will continue to follow developments in the analysis of the AES algorithm. As with its other cryptographic algorithm standards, NIST will formally reevaluate this standard every five years.

Both this standard and possible threats reducing the security provided through the use of this standard will undergo review by NIST as appropriate, taking into account newly available analysis and technology. In addition, the awareness of any breakthrough in technology or any mathematical weakness of the algorithm will cause NIST to reevaluate this standard and provide necessary revisions.

13. **Waiver Procedure.** Under certain exceptional circumstances, the heads of Federal agencies, or their delegates, may approve waivers to Federal Information Processing Standards (FIPS). The heads of such agencies may redelegate such authority only to a senior official designated pursuant to Section 3506(b) of Title 44, U.S. Code. Waivers shall be granted only when compliance with this standard would

a. adversely affect the accomplishment of the mission of an operator of Federal computer system or

b. cause a major adverse financial impact on the operator that is not offset by government-wide savings.

Agency heads may act upon a written waiver request containing the information detailed above. Agency heads may also act without a written waiver request when they determine that conditions for meeting the standard cannot be met. Agency heads may approve waivers only by a written decision that explains the basis on which the agency head made the required finding(s). A copy of each such decision, with procurement sensitive or classified portions clearly identified, shall be sent to: National Institute of Standards and Technology; ATTN: FIPS Waiver Decision, Information Technology Laboratory, 100 Bureau Drive, Stop 8900, Gaithersburg, MD 20899-8900.

In addition, notice of each waiver granted and each delegation of authority to approve waivers shall be sent promptly to the Committee on Government Operations of the House of Representatives and the Committee on Government Affairs of the Senate and shall be published promptly in the Federal Register.

When the determination on a waiver applies to the procurement of equipment and/or services, a notice of the waiver determination must be published in the Commerce Business Daily as a part of the notice of solicitation for offers of an acquisition or, if the waiver determination is made after that notice is published, by amendment to such notice.

A copy of the waiver, any supporting documents, the document approving the waiver and any supporting and accompanying documents, with such deletions as the agency is authorized and decides to make under Section 552(b) of Title 5, U.S. Code, shall be part of the procurement documentation and retained by the agency.

14. Where to obtain copies. This publication is available electronically by accessing http://csrc.nist.gov/publications/. A list of other available computer security publications, including ordering information, can be obtained from NIST Publications List 91, which is available at the same web site. Alternatively, copies of NIST computer security publications are available from: National Technical Information Service (NTIS), 5285 Port Royal Road, Springfield, VA 22161.

Appendix A - Key Expansion Examples

This appendix shows the development of the key schedule for various key sizes. Note that multi-byte values are presented using the notation described in Sec. 3. The intermediate values produced during the development of the key schedule (see Sec. 5.2) are given in the following table (all values are in hexadecimal format, with the exception of the index column (i)).

A.1 Expansion of a 128-bit Cipher Key

This section contains the key expansion of the following cipher key:

```
Cipher Key = 2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c
```

for $Nk = 4$, which results in

$w_0 = 2b7e1516$ $w_1 = 28aed2a6$ $w_2 = abf71588$ $w_3 = 09cf4f3c$

i (dec)	temp	After RotWord()	After SubWord()	Rcon[i/Nk]	After XOR with Rcon	w[i-Nk]	w[i] = temp XOR w[i-Nk]
4	09cf4f3c	cf4f3c09	8a84eb01	01000000	8b84eb01	2b7e1516	a0fafe17
5	a0fafe17					28aed2a6	88542cb1
6	88542cb1					abf71588	23a33939
7	23a33939					09cf4f3c	2a6c7605
8	2a6c7605	6c76052a	50386be5	02000000	52386be5	a0fafe17	f2c295f2
9	f2c295f2					88542cb1	7a96b943
10	7a96b943					23a33939	5935807a
11	5935807a					2a6c7605	7359f67f
12	7359f67f	59f67f73	cb42d28f	04000000	cf42d28f	f2c295f2	3d80477d
13	3d80477d					7a96b943	4716fe3e
14	4716fe3e					5935807a	1e237e44
15	1e237e44					7359f67f	6d7a883b
16	6d7a883b	7a883b6d	dac4e23c	08000000	d2c4e23c	3d80477d	ef44a541
17	ef44a541					4716fe3e	a8525b7f
18	a8525b7f					1e237e44	b671253b
19	b671253b					6d7a883b	db0bad00
20	db0bad00	0bad00db	2b9563b9	10000000	3b9563b9	ef44a541	d4d1c6f8
21	d4d1c6f8					a8525b7f	7c839d87
22	7c839d87					b671253b	caf2b8bc
23	caf2b8bc					db0bad00	11f915bc

24	11f915bc	f915bc11	99596582	20000000	b9596582	d4d1c6f8	6d88a37a
25	6d88a37a					7c839d87	110b3efd
26	110b3efd					caf2b8bc	dbf98641
27	dbf98641					11f915bc	ca0093fd
28	ca0093fd	0093fdca	63dc5474	40000000	23dc5474	6d88a37a	4e54f70e
29	4e54f70e					110b3efd	5f5fc9f3
30	5f5fc9f3					dbf98641	84a64fb2
31	84a64fb2					ca0093fd	4ea6dc4f
32	4ea6dc4f	a6dc4f4e	2486842f	80000000	a486842f	4e54f70e	ead27321
33	ead27321					5f5fc9f3	b58dbad2
34	b58dbad2					84a64fb2	312bf560
35	312bf560					4ea6dc4f	7f8d292f
36	7f8d292f	8d292f7f	5da515d2	1b000000	46a515d2	ead27321	ac7766f3
37	ac7766f3					b58dbad2	19fadc21
38	19fadc21					312bf560	28d12941
39	28d12941					7f8d292f	575c006e
40	575c006e	5c006e57	4a639f5b	36000000	7c639f5b	ac7766f3	d014f9a8
41	d014f9a8					19fadc21	c9ee2589
42	c9ee2589					28d12941	e13f0cc8
43	e13f0cc8					575c006e	b6630ca6

A.2 Expansion of a 192-bit Cipher Key

This section contains the key expansion of the following cipher key:

```
Cipher Key =    8e 73 b0 f7 da 0e 64 52 c8 10 f3 2b
                80 90 79 e5 62 f8 ea d2 52 2c 6b 7b
```

for $Nk = 6$, which results in

$w_0 = 8e73b0f7$ $w_1 = da0e6452$ $w_2 = c810f32b$ $w_3 = 809079e5$

$w_4 = 62f8ead2$ $w_5 = 522c6b7b$

i (dec)	temp	After RotWord()	After SubWord()	Rcon[i/Nk]	After XOR with Rcon	w[i-Nk]	w[i] = temp XOR w[i-Nk]
6	522c6b7b	2c6b7b52	717f2100	01000000	707f2100	8e73b0f7	fe0c91f7
7	fe0c91f7					da0e6452	2402f5a5
8	2402f5a5					c810f32b	ec12068e

9	ec12068e					809079e5	6c827f6b
10	6c827f6b					62f8ead2	0e7a95b9
11	0e7a95b9					522c6b7b	5c56fec2
12	5c56fec2	56fec25c	b1bb254a	02000000	b3bb254a	fe0c91f7	4db7b4bd
13	4db7b4bd					2402f5a5	69b54118
14	69b54118					ec12068e	85a74796
15	85a74796					6c827f6b	e92538fd
16	e92538fd					0e7a95b9	e75fad44
17	e75fad44					5c56fec2	bb095386
18	bb095386	095386bb	01ed44ea	04000000	05ed44ea	4db7b4bd	485af057
19	485af057					69b54118	21efb14f
20	21efb14f					85a74796	a448f6d9
21	a448f6d9					e92538fd	4d6dce24
22	4d6dce24					e75fad44	aa326360
23	aa326360					bb095386	113b30e6
24	113b30e6	3b30e611	e2048e82	08000000	ea048e82	485af057	a25e7ed5
25	a25e7ed5					21efb14f	83b1cf9a
26	83b1cf9a					a448f6d9	27f93943
27	27f93943					4d6dce24	6a94f767
28	6a94f767					aa326360	c0a69407
29	c0a69407					113b30e6	d19da4e1
30	d19da4e1	9da4e1d1	5e49f83e	10000000	4e49f83e	a25e7ed5	ec1786eb
31	ec1786eb					83b1cf9a	6fa64971
32	6fa64971					27f93943	485f7032
33	485f7032					6a94f767	22cb8755
34	22cb8755					c0a69407	e26d1352
35	e26d1352					d19da4e1	33f0b7b3
36	33f0b7b3	f0b7b333	8ca96dc3	20000000	aca96dc3	ec1786eb	40beeb28
37	40beeb28					6fa64971	2f18a259
38	2f18a259					485f7032	6747d26b
39	6747d26b					22cb8755	458c553e
40	458c553e					e26d1352	a7e1466c
41	a7e1466c					33f0b7b3	9411f1df
42	9411f1df	11f1df94	82a19e22	40000000	c2a19e22	40beeb28	821f750a
43	821f750a					2f18a259	ad07d753

44	ad07d753					6747d26b	ca400538
45	ca400538					458c553e	8fcc5006
46	8fcc5006					a7e1466c	282d166a
47	282d166a					9411f1df	bc3ce7b5
48	bc3ce7b5	3ce7b5bc	eb94d565	80000000	6b94d565	821f750a	e98ba06f
49	e98ba06f					ad07d753	448c773c
50	448c773c					ca400538	8ecc7204
51	8ecc7204					8fcc5006	01002202

A.3 Expansion of a 256-bit Cipher Key

This section contains the key expansion of the following cipher key:

```
Cipher Key =    60 3d eb 10 15 ca 71 be 2b 73 ae f0 85 7d 77 81
                1f 35 2c 07 3b 61 08 d7 2d 98 10 a3 09 14 df f4
```

for $Nk = 8$, which results in

$w_0 = 603deb10$ $w_1 = 15ca71be$ $w_2 = 2b73aef0$ $w_3 = 857d7781$

$w_4 = 1f352c07$ $w_5 = 3b6108d7$ $w_6 = 2d9810a3$ $w_7 = 0914dff4$

i (dec)	temp	After RotWord()	After SubWord()	Rcon[i/Nk]	After XOR with Rcon	w[i-Nk]	w[i] = temp XOR w[i-Nk]
8	0914dff4	14dff409	fa9ebf01	01000000	fb9ebf01	603deb10	9ba35411
9	9ba35411					15ca71be	8e6925af
10	8e6925af					2b73aef0	a51a8b5f
11	a51a8b5f					857d7781	2067fcde
12	2067fcde		b785b01d			1f352c07	a8b09c1a
13	a8b09c1a					3b6108d7	93d194cd
14	93d194cd					2d9810a3	be49846e
15	be49846e					0914dff4	b75d5b9a
16	b75d5b9a	5d5b9ab7	4c39b8a9	02000000	4e39b8a9	9ba35411	d59aecb8
17	d59aecb8					8e6925af	5bf3c917
18	5bf3c917					a51a8b5f	fee94248
19	fee94248					2067fcde	de8ebe96
20	de8ebe96		1d19ae90			a8b09c1a	b5a9328a
21	b5a9328a					93d194cd	2678a647
22	2678a647					be49846e	98312229

23	98312229					b75d5b9a	2f6c79b3
24	2f6c79b3	6c79b32f	50b66d15	04000000	54b66d15	d59aecb8	812c81ad
25	812c81ad					5bf3c917	dadf48ba
26	dadf48ba					fee94248	24360af2
27	24360af2					de8ebe96	fab8b464
28	fab8b464		2d6c8d43			b5a9328a	98c5bfc9
29	98c5bfc9					2678a647	bebd198e
30	bebd198e					98312229	268c3ba7
31	268c3ba7					2f6c79b3	09e04214
32	09e04214	e0421409	e12cfa01	08000000	e92cfa01	812c81ad	68007bac
33	68007bac					dadf48ba	b2df3316
34	b2df3316					24360af2	96e939e4
35	96e939e4					fab8b464	6c518d80
36	6c518d80		50d15dcd			98c5bfc9	c814e204
37	c814e204					bebd198e	76a9fb8a
38	76a9fb8a					268c3ba7	5025c02d
39	5025c02d					09e04214	59c58239
40	59c58239	c5823959	a61312cb	10000000	b61312cb	68007bac	de136967
41	de136967					b2df3316	6ccc5a71
42	6ccc5a71					96e939e4	fa256395
43	fa256395					6c518d80	9674ee15
44	9674ee15		90922859			c814e204	5886ca5d
45	5886ca5d					76a9fb8a	2e2f31d7
46	2e2f31d7					5025c02d	7e0af1fa
47	7e0af1fa					59c58239	27cf73c3
48	27cf73c3	cf73c327	8a8f2ecc	20000000	aa8f2ecc	de136967	749c47ab
49	749c47ab					6ccc5a71	18501dda
50	18501dda					fa256395	e2757e4f
51	e2757e4f					9674ee15	7401905a
52	7401905a		927c60be			5886ca5d	cafaaae3
53	cafaaae3					2e2f31d7	e4d59b34
54	e4d59b34					7e0af1fa	9adf6ace
55	9adf6ace					27cf73c3	bd10190d
56	bd10190d	10190dbd	cad4d77a	40000000	8ad4d77a	749c47ab	fe4890d1
57	fe4890d1					18501dda	e6188d0b

58	e6188d0b					e2757e4f	046df344
59	046df344					7401905a	706c631e

Appendix B – Cipher Example

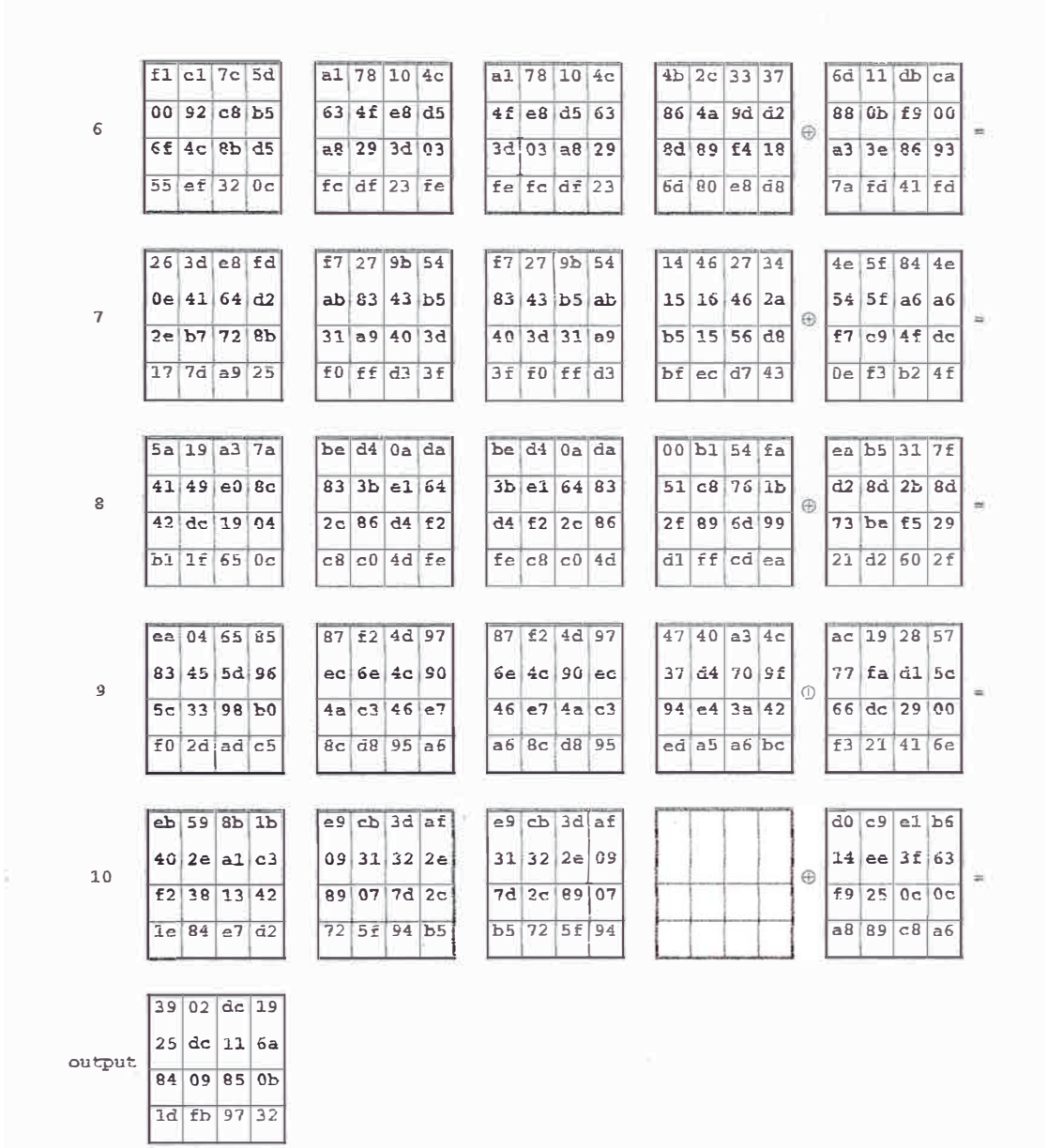
The following diagram shows the values in the State array as the Cipher progresses for a block length and a Cipher Key length of 16 bytes each (i.e., $Nb = 4$ and $Nk = 4$).

```
Input =        32 43 f6 a8 88 5a 30 8d 31 31 98 a2 e0 37 07 34

Cipher Key = 2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c
```

The Round Key values are taken from the Key Expansion example in Appendix A.

Round Number	Start of Round	After SubBytes	After ShiftRows	After MixColumns	Round Key Value
input	32 88 31 e0 43 5a 31 37 f6 30 98 07 a8 8d a2 34			⊕	2b 28 ab 09 7e ae f7 cf 15 d2 15 4f 16 a6 88 3c =
1	19 a0 9a e9 3d f4 c6 f8 e3 e2 8d 48 be 2b 2a 08	d4 e0 b8 1e 27 bf b4 41 11 98 5d 52 ae f1 e5 30	d4 e0 b8 1e bf b4 41 27 5d 52 11 98 30 ae f1 e5	04 e0 48 28 66 cb f8 06 81 19 d3 26 e5 9a 7a 4c	⊕ a0 88 23 2a fa 54 a3 6c fe 2c 39 76 17 b1 39 05 =
2	a4 68 6b 02 9c 9f 5b 6a 7f 35 ea 50 f2 2b 43 49	49 45 7f 77 de db 39 02 d2 96 87 53 89 f1 1a 3b	49 45 7f 77 db 39 02 de 87 53 d2 96 3b 89 f1 1a	58 1b db 1b 4d 4b e7 6b ca 5a ca b0 f1 ac a8 e5	⊕ f2 7a 59 73 c2 96 35 59 95 b9 80 f6 f2 43 7a 7f =
3	aa 61 82 68 8f dd d2 32 5f e3 4a 46 03 ef d2 9a	ac ef 13 45 73 c1 b5 23 cf 11 d6 5a 7b df b5 b8	ac ef 13 45 c1 b5 23 73 d6 5a cf 11 b8 7b df b5	75 20 53 bb ec 0b c0 25 09 63 cf d0 93 33 7c dc	⊕ 3d 47 1e 6d 80 16 23 7a 47 fe 7e 88 7d 3e 44 3b =
4	48 67 4d d6 6c 1d e3 5f 4e 9d b1 58 ee 0d 38 e7	52 85 e3 f6 50 a4 11 cf 2f 5e c8 6a 28 d7 07 94	52 85 e3 f6 a4 11 cf 50 c8 6a 2f 5e 94 28 d7 07	0f 60 6f 5e d6 31 c0 b3 da 38 10 13 a9 bf 6b 01	⊕ ef a8 b6 db 44 52 71 0b a5 5b 25 ad 41 7f 3b 00 =
5	e0 c8 d9 85 92 63 b1 b8 7f 63 35 be e8 c0 50 01	e1 e8 35 97 4f fb c8 6c d2 fb 96 ae 9b ba 53 7c	e1 e8 35 97 fb c8 6c 4f 96 ae d2 fb 7c 9b ba 53	25 bd b6 4c d1 11 3a 4c a9 d1 33 c0 ad 68 8e b0	⊕ d4 7c ca 11 d1 83 f2 f9 c6 9d b9 15 f8 87 bc bc =


6
f1 c1 7c 5d
00 92 c8 b5
6f 4c 8b d5
55 ef 32 0c
a1 78 10 4c
63 4f e8 d5
a8 29 3d 03
fc df 23 fe
a1 78 10 4c
4f e8 d5 63
3d 03 a8 29
fe fc df 23
4b 2c 33 37
86 4a 9d d2
8d 89 f4 18
6d 80 e8 d8
6d 11 db ca
88 0b f9 00
a3 3e 86 93
7a fd 41 fd
7
26 3d e8 fd
0e 41 64 d2
2e b7 72 8b
17 7d a9 25
f7 27 9b 54
ab 83 43 b5
31 a9 40 3d
f0 ff d3 3f
f7 27 9b 54
83 43 b5 ab
40 3d 31 a9
3f f0 ff d3
14 46 27 34
15 16 46 2a
b5 15 56 d8
bf ec d7 43
4e 5f 84 4e
54 5f a6 a6
f7 c9 4f dc
0e f3 b2 4f
8
5a 19 a3 7a
41 49 e0 8c
42 dc 19 04
b1 1f 65 0c
be d4 0a da
83 3b e1 64
2c 86 d4 f2
c8 c0 4d fe
be d4 0a da
3b e1 64 83
d4 f2 2c 86
fe c8 c0 4d
00 b1 54 fa
51 c8 76 1b
2f 89 6d 99
d1 ff cd ea
ea b5 31 7f
d2 8d 2b 8d
73 be f5 29
21 d2 60 2f
9
ea 04 65 85
83 45 5d 96
5c 33 98 b0
f0 2d ad c5
87 f2 4d 97
ec 6e 4c 90
4a c3 46 e7
8c d8 95 a6
87 f2 4d 97
6e 4c 90 ec
46 e7 4a c3
a6 8c d8 95
47 40 a3 4c
37 d4 70 9f
94 e4 3a 42
ed a5 a6 bc
ac 19 28 57
77 fa d1 5c
66 dc 29 00
f3 21 41 6e
10
eb 59 8b 1b
40 2e a1 c3
f2 38 13 42
1e 84 e7 d2
e9 cb 3d af
09 31 32 2e
89 07 7d 2c
72 5f 94 b5
e9 cb 3d af
31 32 2e 09
7d 2c 89 07
b5 72 5f 94
d0 c9 e1 b6
14 ee 3f 63
f9 25 0c 0c
a8 89 c8 a6
output
39 02 dc 19
25 dc 11 6a
84 09 85 0b
1d fb 97 32

Appendix C – Example Vectors

This appendix contains example vectors, including intermediate values – for all three AES key lengths (Nk = 4, 6, and 8), for the Cipher, Inverse Cipher, and Equivalent Inverse Cipher that are described in Sec. 5.1, 5.3, and 5.3.5, respectively. Additional examples may be found at [1] and [5].

All vectors are in hexadecimal notation, with each pair of characters giving a byte value in which the left character of each pair provides the bit pattern for the 4 bit group containing the higher numbered bits using the notation explained in Sec. 3.2, while the right character provides the bit pattern for the lower-numbered bits. The array index for all bytes (groups of two hexadecimal digits) within these test vectors starts at zero and increases from left to right.

```
Legend for CIPHER (ENCRYPT) (round number r = 0 to 10, 12 or 14):

   input:   cipher input
   start:   state at start of round[r]
   s box:   state after SubBytes()
   s row:   state after ShiftRows()
   m col:   state after MixColumns()
   k sch:   key schedule value for round[r]
   output:  cipher output


Legend for INVERSE CIPHER (DECRYPT) (round number r = 0 to 10, 12 or 14):
   iinput:  inverse cipher input
   istart:  state at start of round[r]
   is box:  state after InvSubBytes()
   is row:  state after InvShiftRows()
   ik sch:  key schedule value for round[r]
   ik add:  state after AddRoundKey()
   ioutput: inverse cipher output


Legend for EQUIVALENT INVERSE CIPHER (DECRYPT) (round number r   0 to 10, 12
   or 14):

   iinput:  inverse cipher input
   istart:  state at start of round[r]
   is box:  state after InvSubBytes()
   is row:  state after InvShiftRows()
   im col:  state after InvMixColumns()
   ik sch:  key schedule value for round[r]
   ioutput: inverse cipher output
```

C.1 AES-128 (Nk=4, Nr=10)

```
PLAINTEXT:          00112233445566778899aabbccddeeff
KEY:                000102030405060708090a0b0c0d0e0f

CIPHER (ENCRYPT):
```

```
round[ 0].input     00112233445566778899aabbccddeeff
round[ 0].k_sch     000102030405060708090a0b0c0d0e0f
round[ 1].start     00102030405060708090a0b0c0d0e0f0
round[ 1].s_box     63cab7040953d051cd60e0e7ba70e18c
round[ 1].s_row     6353e08c0960e104cd70b751bacad0e7
round[ 1].m_col     5f72641557f5bc92f7be3b291db9f91a
round[ 1].k_sch     d6aa74fdd2af72fadaa678f1d6ab76fe
round[ 2].start     89d810e8855ace682d1843d8cb128fe4
round[ 2].s_box     a761ca9b97be8b45d8ad1a611fc97369
round[ 2].s_row     a7be1a6997ad739bd8c9ca451f618b61
round[ 2].m_col     ff87968431d86a51645151fa773ad009
round[ 2].k_sch     b692cf0b643dbdf1be9bc5006830b3fe
round[ 3].start     4915598f55e5d7a0daca94fa1f0a63f7
round[ 3].s_box     3b59cb73fcd90ee05774222dc067fb68
round[ 3].s_row     3bd92268fc74fb735767cbe0c0590e2d
round[ 3].m_col     4c9c1e66f771f0762c3f868e534df256
round[ 3].k_sch     b6ff744ed2c2c9bf6c590cbf0469bf41
round[ 4].start     fa636a2825b339c940668a3157244d17
round[ 4].s_box     2dfb02343f6d12dd09337ec75b36e3f0
round[ 4].s_row     2d6d7ef03f33e334093602dd5bfb12c7
round[ 4].m_col     6385b79ffc538df997be478e7547d691
round[ 4].k_sch     47f7f7bc95353e03f96c32bcfd058dfd
round[ 5].start     247240236966b3fa6ed2753288425b6c
round[ 5].s_box     36400926f9336d2d9fb59d23c42c3950
round[ 5].s_row     36339d50f9b539269f2c092dc4406d23
round[ 5].m_col     f4bcd45432e554d075f1d6c51dd03b3c
round[ 5].k_sch     3caaa3e8a99f9deb50f3af57adf622aa
round[ 6].start     c81677bc9b7ac93b25027992b0261996
round[ 6].s_box     e847f56514dadde23f77b64fe7f7d490
round[ 6].s_row     e8dab6901477d4653ff7f5e2e747dd4f
round[ 6].m_col     9816ee7400f87f556b2c049c8e5ad036
round[ 6].k_sch     5e390f7df7a69296a7553dc10aa31f6b
round[ 7].start     c62fe109f75eedc3cc79395d84f9cf5d
round[ 7].s_box     b415f8016858552e4bb6124c5f998a4c
round[ 7].s_row     b458124c68b68a014b99f82e5f15554c
round[ 7].m_col     c57e1c159a9bd286f05f4be098c63439
round[ 7].k_sch     14f9701ae35fe28c440adf4d4ea9c026
round[ 8].start     d1876c0f79c4300ab45594add66ff41f
round[ 8].s_box     3e175076b61c04678dfc2295f6a8bfc0
round[ 8].s_row     3e1c22c0b6fcbf768da85067f6170495
round[ 8].m_col     baa03de7a1f9b56ed5512cba5f414d23
round[ 8].k_sch     47438735a41c65b9e016baf4aebf7ad2
round[ 9].start     fde3bad205e5d0d73547964ef1fe37f1
round[ 9].s_box     5411f4b56bd9700e96a0902fa1bb9aa1
round[ 9].s_row     54d990a16ba09ab596bbf40ea111702f
round[ 9].m_col     e9f74eec023020f61bf2ccf2353c21c7
round[ 9].k_sch     549932d1f08557681093ed9cbe2c974e
round[10].start     bd6e7c3df2b5779e0b61216e8b10b689
round[10].s_box     7a9f102789d5f50b2beffd9f3dca4ea7
round[10].s_row     7ad5fda789ef4e272bca100b3d9ff59f
round[10].k_sch     13111d7fe3944a17f307a78b4d2b30c5
round[10].output    69c4e0d86a7b0430d8cdb78070b4c55a

INVERSE CIPHER (DECRYPT):
round[ 0].iinput    69c4e0d86a7b0430d8cdb78070b4c55a
round[ 0].ik_sch    13111d7fe3944a17f307a78b4d2b30c5
round[ 1].istart    7ad5fda789ef4e272bca100b3d9ff59f
```

```
round[ 1].is_row   7a9f102789d5f50b2beffd9f3dca4ea7
round[ 1].is_box   bd6e7c3df2b5779e0b61216e8b10b689
round[ 1].ik_sch   549932d1f08557681093ed9cbe2c974e
round[ 1].ik_add   e9f74eec023020f61bf2ccf2353c21c7
round[ 2].istart   54d990a16ba09ab596bbf40ea111702f
round[ 2].is_row   5411f4b56bd9700e96a0902fa1bb9aa1
round[ 2].is_box   fde3bad205e5d0d73547964ef1fe37f1
round[ 2].ik_sch   47438735a41c65b9e016baf4aebf7ad2
round[ 2].ik_add   baa03de7a1f9b56ed5512cba5f414d23
round[ 3].istart   3e1c22c0b6fcbf768da85067f6170495
round[ 3].is_row   3ei75076b61c04678dfc2295f6a8bfc0
round[ 3].is_box   d1876c0f79c4300ab45594add66ff41f
round[ 3].ik_sch   14f9701ae35fe28c440adf4d4ea9c026
round[ 3].ik_add   c57e1c159a9bd286f05f4be098c63439
round[ 4].istart   b458124c68b68a014b99f82e5f15554c
round[ 4].is_row   b415f8016858552e4bb6124c5f998a4c
round[ 4].is_box   c62fe109f75eedc3cc79395d84f9cf5d
round[ 4].ik_sch   5e390f7df7a69296a7553dc10aa31f6b
round[ 4].ik_add   9816ee7400f87f556b2c049c8e5ad036
round[ 5].istart   e8dab6901477d4653ff7f5e2e747dd4f
round[ 5].is_row   e847f56514dadde23f77b64fe7f7d490
round[ 5].is_box   c81677bc9b7ac93b25027992b0261996
round[ 5].ik_sch   3caaa3e8a99f9deb50f3af57adf622aa
round[ 5].ik_add   f4bcd45432e554d075f1d6c51dd03b3c
round[ 6].istart   36339d50f9b539269f2c092dc4406d23
round[ 6].is_row   36400926f9336d2d9fb59d23c42c3950
round[ 6].is_box   247240236966b3fa6ed2753288425b6c
round[ 6].ik_sch   47f7f7bc95353e03f96c32bcfd058dfd
round[ 6].ik_add   6385b79ffc538df997be478e7547d691
round[ 7].istart   2d6d7ef03f33e334093602dd5bfb12c7
round[ 7].is_row   2dfb02343f6d12dd09337ec75b36e3f0
round[ 7].is_box   fa636a2825b339c940668a3157244d17
round[ 7].ik_sch   b6ff744ed2c2c9bf6c590cbf0469bf41
round[ 7].ik_add   4c9c1e66f771f0762c3f868e534df256
round[ 8].istart   3bd92268fc74fb735767cbe0c0590e2d
round[ 8].is_row   3b59cb73fcd90ee05774222dc067fb68
round[ 8].is_box   4915598f55e5d7a0daca94fa1f0a63f7
round[ 8].ik_sch   b692cf0b643dbdf1be9bc5006830b3fe
round[ 8].ik_add   ff87968431d86a51645151fa773ad009
round[ 9].istart   a7bela6997ad739bd8c9ca451f618b61
round[ 9].is_row   a761ca9b97be8b45d8ad1a611fc97369
round[ 9].is_box   89d810e8855ace682d1843d8cb128fe4
round[ 9].ik_sch   d6aa74fdd2af72fadaa678f1d6ab76fe
round[ 9].ik_add   5f72641557f5bc92f7be3b291db9f91a
round[10].istart   6353e08c0960e104cd70b751bacad0e7
round[10].is_row   63cab7040953d051cd60e0e7ba70e18c
round[10].is_box   000102030405060708090a0b0c0d0e0f0
round[10].ik_sch   00010203040506070809 0a0b0c0d0e0f
round[10].ioutput  00112233445566778899aabbccddeeff

EQUIVALENT INVERSE CIPHER (DECRYPT):
round[ 0].iinput   69c4e0d86a7b0430d8cdb78070b4c55a
round[ 0].ik_sch   13111d7fe3944a17f307a78b4d2b30c5
round[ 1].istart   7ad5fda789ef4e272bca100b3d9ff59f
round[ 1].is_box   bdb52189f261b63d0b107c9e8b6e776e
round[ 1].is_row   bd6e7c3df2b5779e0b61216e8b10b689
round[ 1].im_col   4773b91ff72f354361cb018ea1e6cf2c
```

```
round[ 1].ik_sch   13aa29be9c8faff6f770f58000f7bf03
round[ 2].istart   54d990a16ba09ab596bbf40ea111702f
round[ 2].is_box   fde596f1054737d235febad7f1e3d04e
round[ 2].is_row   fde3bad205e5d0d73547964ef1fe37f1
round[ 2].im_col   2d7e86a339d9393ee6570a1101904e16
round[ 2].ik_sch   1362a4638f2586486bff5a76f7874a83
round[ 3].istart   3e1c22c0b6fcbf768da85067f6170495
round[ 3].is_box   d1c4941f7955f40fb46f6c0ad68730ad
round[ 3].is_row   d1876c0f79c4300ab45594add66ff41f
round[ 3].im_col   39daee38f4f1a82aaf432410c36d45b9
round[ 3].ik_sch   8d82fc749c47222be4dadc3e9c7810f5
round[ 4].istart   b458124c68b68a014b99f82e5f15554c
round[ 4].is_box   c65e395df779cf09ccf9e1c3842fed5d
round[ 4].is_row   c62fe109f75eedc3cc79395d84f9cf5d
round[ 4].im_col   9a39bf1d05b20a3a476a0bf79fe51184
round[ 4].ik_sch   72e3098d11c5de5f789dfe1578a2cccb
round[ 5].istart   e8dab6901477d4653ff7f5e2e747dd4f
round[ 5].is_box   c87a79969b0219bc2526773bb016c992
round[ 5].is_row   c81677bc9b7ac93b25027992b0261996
round[ 5].im_col   18f78d779a93eef4f6742967c47f5ffd
round[ 5].ik_sch   2ec410276326d7d26958204a003f32de
round[ 6].istart   36339d50f9b539269f2c092dc4406d23
round[ 6].is_box   2466756c69d25b236e4240fa8872b332
round[ 6].is_row   247240236966b3fa6ed2753288425b6c
round[ 6].im_col   85cf8bf472d124c10348f545329c0053
round[ 6].ik_sch   a8a2f5044de2c7f50a7ef79869671294
round[ 7].istart   2d6d7ef03f33e334093602dd5bfb12c7
round[ 7].is_box   fab38a1725664d2840246ac957633931
round[ 7].is_row   fa636a2825b339c940668a3157244d17
round[ 7].im_col   fc1fc1f91934c98210fbfb8da340eb21
round[ 7].ik_sch   c7c6e391e54032f1479c306d6319e50c
round[ 8].istart   3bd92268fc74fb735767cbe0c0590e2d
round[ 8].is_box   49e594f755ca638fda0a59a01f15d7fa
round[ 8].is_row   4915598f55e5d7a0daca94fa1f0a63f7
round[ 8].im_col   076518f0b52ba2fb7a15c8d93be45e00
round[ 8].ik_sch   a0db02992286d160a2dc029c2485d561
round[ 9].istart   a7be1a6997ad739bd8c9ca451f618b61
round[ 9].is_box   895a43e485188fe82d121068cbd8ced8
round[ 9].is_row   89d810e8855ace682d1843d8cb128fe4
round[ 9].im_col   ef053f7c8b3d32fd4d2a64ad3c93071a
round[ 9].ik_sch   8c56dff0825dd3f9805ad3fc8659d7fd
round[10].istart   6353e08c0960e104cd70b751bacad0e7
round[10].is_box   0050a0f04090e03080d02070c01060b0
round[10].is_row   00102030405060708090a0b0c0d0e0f0
round[10].ik_sch   000102030405060708090a0b0c0d0e0f
round[10].ioutput  00112233445566778899aabbccddeeff
```

C.2 AES-192 (Nk=6, Nr=12)

```
PLAINTEXT:  00112233445566778899aabbccddeeff
KEY:        000102030405060708090a0b0c0d0e0f1011121314151617

CIPHER (ENCRYPT):
round[ 0].input    00112233445566778899aabbccddeeff
round[ 0].k_sch    000102030405060708090a0b0c0d0e0f
round[ 1].start    00102030405060708090a0b0c0d0e0f0
```

```
round[ 1].s_box     63cab7040953d051cd60e0e7ba70e18c
round[ 1].s_row     6353e08c0960e104cd70b751bacad0e7
round[ 1].m_col     5f72641557f5bc92f7be3b291db9f91a
round[ 1].k_sch     101112131415161758 46f2f95c43f4fe
round[ 2].start     4f63760643e0aa85aff8c9d041fa0de4
round[ 2].s_box     84fb386f1ae1ac977941dd70832dd769
round[ 2].s_row     84e1dd691a41d76f792d389783fbac70
round[ 2].m_col     9f487f794f955f662afc86abd7f1ab29
round[ 2].k_sch     544afef55847f0fa4856e2e95c43f4fe
round[ 3].start     cb02818c17d2af9c62aa64428bb25fd7
round[ 3].s_box     1f770c64f0b579deaaac432c3d37cf0e
round[ 3].s_row     1fb5430ef0accf64aa370cde3d77792c
round[ 3].m_col     b7a53ecbbf9d75a0c40efc79b674cc11
round[ 3].k_sch     40f949b31cbabd4d48f043b810b7b342
round[ 4].start     f75c7778a327c8ed8cfebfc1a6c37f53
round[ 4].s_box     684af5bc0acce85564bb0878242ed2ed
round[ 4].s_row     68cc08ed0abbd2bc642ef555244ae878
round[ 4].m_col     7a1e98bdacb6d1141a6944dd06eb2d3e
round[ 4].k_sch     58e151ab04a2a5557effb5416245080c
round[ 5].start     22ffc916a81474416496f19c64ae2532
round[ 5].s_box     9316dd47c2fa92834390a1de43e43f23
round[ 5].s_row     93faa123c2903f4743e4dd83431692de
round[ 5].m_col     aaa755b34cffe57cef6f98e1f01c13e6
round[ 5].k_sch     2ab54bb43a02f8f662e3a95d66410c08
round[ 6].start     80121e0776fd1d8a8d8c31bc965d1fee
round[ 6].s_box     cdc972c53854a47e5d64c765904cc028
round[ 6].s_row     cd54c7283864c0c55d4c727e90c9a465
round[ 6].m_col     921f748fd96e937d622d7725ba8ba50c
round[ 6].k_sch     f501857297448d7ebdf1c6ca87f33e3c
round[ 7].start     671ef1fd4e2a1e03dfdcb1ef3d789b30
round[ 7].s_box     8572a1542fe5727b9e86c8df27bc1404
round[ 7].s_row     85e5c8042f8614549ebca17b277272df
round[ 7].m_col     e913e7b18f507d4b227ef652758acbcc
round[ 7].k_sch     e510976183519b6934157c9ea351f1e0
round[ 8].start     0c0370d00c01e622166b8accd6db3a2c
round[ 8].s_box     fe7b5170fe7c8e93477f7e4bf6b98071
round[ 8].s_row     fe7c7e71fe7f807047b95193f67b8e4b
round[ 8].m_col     6cf5edf996eb0a069c4ef21cbfc25762
round[ 8].k_sch     1ea0372a995309167c439e77ff12051e
round[ 9].start     7255dad30fb80310e00d6c6b40d0527c
round[ 9].s_box     40fc5766766c7bcae1d7507f09700010
round[ 9].s_row     406c501076d70066e17057ca09fc7b7f
round[ 9].m_col     7478bcdce8a50b81d4327a9009188262
round[ 9].k_sch     dd7e0e887e2fff68608fc842f9dcc154
round[10].start     a906b254968af4e9b4bdb2d2f0c44336
round[10].s_box     d36f3720907ebf1e8d7a37b58c1c1a05
round[10].s_row     d37e3705907a1a208d1c371e8c6fbfb5
round[10].m_col     0d73cc2d8f6abe8b0cf2dd9bb83d422e
round[10].k_sch     859f5f237a8d5a3dc0c02952beefd63a
round[11].start     88ec930ef5e7e4b6cc32f4c906d29414
round[11].s_box     c4cedcabe694694e4b23bfdd6fb522fa
round[11].s_row     c494bffae62322ab4bb5dc4e6fce69dd
round[11].m_col     71d720933b6d677dc00b8f28238e0fb7
round[11].k_sch     de601e7827bcdf2ca223800fd8aeda32
round[12].start     afb73eeb1cd1b85162280f27fb20d585
round[12].s_box     79a9b2e99c3e6cd1aa3476cc0fb70397
round[12].s_row     793e76979c3403e9aab7b2d10fa96ccc
```

```
round[12].k_sch    a4970a331a78dc09c418c271e3a41d5d
round[12].output   dda97ca4864cdfa06eaf70a0ec0d7191

INVERSE CIPHER (DECRYPT):
round[ 0].iinput   dda97ca4864cdfe06eaf70a0ec0d7191
round[ 0].ik_sch   a4970a331a78dc09c418c271e3a41d5d
round[ 1].istart   793e76979c3403e9aab7b2d10fa96ccc
round[ 1].is_row   79a9b2e99c3e6cd1aa3476cc0fb70397
round[ 1].is_box   afb73eeb1cd1b85162280f27fb20d585
round[ 1].ik_sch   de601e7827bcdf2ca223800fd8aeda32
round[ 1].ik_add   71d720933b6d677dc00b8f28238e0fb7
round[ 2].istart   c494bffae62322ab4bb5dc4e6fce69dd
round[ 2].is_row   c4cedcabe694694e4b23bfdd6fb522fa
round[ 2].is_box   88ec930ef5e7e4b6cc32f4c906d29414
round[ 2].ik_sch   859f5f237a8d5a3dc0c02952beefd63a
round[ 2].ik_add   0d73cc2d8f6abe8b0cf2dd9bb83d422e
round[ 3].istart   d37e3705907a1a208d1c371e8c6fbfb5
round[ 3].is_row   d36f3720907ebf1e8d7a37b58c1c1a05
round[ 3].is_box   a906b254968af4e9b4bdb2d2f0c44336
round[ 3].ik_sch   dd7e0e887e2fff68608fc842f9dcc154
round[ 3].ik_add   7478bcdce8a50b81d4327a9009188262
round[ 4].istart   406c501076d70066e17057ca09fc7b7f
round[ 4].is_row   40fc5766766c7bcae1d7507f09700010
round[ 4].is_box   7255dad30fb80310e00d6c6b40d0527c
round[ 4].ik_sch   1ea0372a995309167c439e77ff12051e
round[ 4].ik_add   6cf5edf996eb0a069c4ef21cbfc25762
round[ 5].istart   fe7c7e71fe7f807047b95193f67b8e4b
round[ 5].is_row   fe7b5170fe7c8e93477f7e4bf6b98071
round[ 5].is_box   0c0370d00c01e622166b8accd6db3a2c
round[ 5].ik_sch   e5109761835l9b6934157c9ea351f1e0
round[ 5].ik_add   e913e7b18f507d4b227ef652758acbcc
round[ 6].istart   85e5c8042f8614549ebca17b277272df
round[ 6].is_row   8572a1542fe5727b9e86c8df27bc1404
round[ 6].is_box   671ef1fd4e2a1e03dfdcb1ef3d789b30
round[ 6].ik_sch   f501857297448d7ebdf1c6ca87f33e3c
round[ 6].ik_add   921f748fd96e937d622d7725ba8ba50c
round[ 7].istart   cd54c7283864c0c55d4c727e90c9a465
round[ 7].is_row   cdc972c53854a47e5d64c765904cc028
round[ 7].is_box   80121e0776fd1d8a8d8c31bc965d1fee
round[ 7].ik_sch   2ab54bb43a02f8f662e3a95d66410c08
round[ 7].ik_add   aaa755b34cffe57cef6f98e1f01c13e6
round[ 8].istart   93faa123c2903f4743e4dd83431692de
round[ 8].is_row   9316dd47c2fa92834390a1de43e43f23
round[ 8].is_box   22ffc916a81474416496f19c64ae2532
round[ 8].ik_sch   58e151ab04a2a5557effb5416245080c
round[ 8].ik_add   7a1e98bdacb6d1141a6944dd06eb2d3e
round[ 9].istart   68cc08ed0abbd2bc642ef555244ae878
round[ 9].is_row   684af5bc0acce85564bb0878242ed2ed
round[ 9].is_box   f75c7778a327c8ed8cfebfc1a6c37f53
round[ 9].ik_sch   40f949b31cbabd4d48f043b810b7b342
round[ 9].ik_add   b7a53ecbbf9d75a0c40efc79b674cc11
round[10].istart   1fb5430ef0accf64aa370cde3d77792c
round[10].is_row   1f770c64f0b579deaaac432c3d37cf0e
round[10].is_box   cb02818c17d2af9c62aa64428bb25fd7
round[10].ik_sch   544afef55847f0fa4856e2e95c43f4fe
round[10].ik_add   9f487f794f955f662afc86abd7f1ab29
round[11].istart   84e1dd691a41d76f792d389783fbac70
```

```
round[11].is_row   84fb386f1ae1ac977941dd70832dd769
round[11].is_box   4f63760643e0aa85aff8c9d041fa0de4
round[11].ik_sch   101112131415161758462f95c43f4fe
round[11].ik_add   5f72641557f5bc92f7be3b291db9f91a
round[12].istart   6353e08c0960e104cd70b751bacad0e7
round[12].is_row   63cab7040953d051cd60e0e7ba70e18c
round[12].is_box   000102030405060708090a0b0c0d0e0f0
round[12].ik_sch   00010203040506070809a0b0c0d0e0f
round[12].ioutput  00112233445566778899aabbccddeeff

EQUIVALENT INVERSE CIPHER (DECRYPT):
round[ 0].iinput   dda97ca4864cdfe06eaf70a0ec0d7191
round[ 0].ik_sch   a4970a331a78dc09c418c271e3a41d5d
round[ 1].istart   793e76979c3403e9aab7b2d10fa96ccc
round[ 1].is_box   afd10f851c28d5eb62203e51fbb7b827
round[ 1].is_row   afb73eeb1cd1b85162280f27fb20d585
round[ 1].im_col   122a02f7242ac8e20605afce51cc7264
round[ 1].ik_sch   d6bebd0dc209ea494db073803e021bb9
round[ 2].istart   c494bffae62322ab4bb5dc4e6fce69dd
round[ 2].is_box   88e7f414f532940eccd293b606ece4c9
round[ 2].is_row   88ec930ef5e7e4b6cc32f4c906d29414
round[ 2].im_col   5cc7aecce3c872194ae5ef8309a933c7
round[ 2].ik_sch   8fb999c973b26839c7f9d89d85c68c72
round[ 3].istart   d37e3705907a1a208d1c371e8c6fbfb5
round[ 3].is_box   a98ab23696bd4354b4c4b2e9f006f4d2
round[ 3].is_row   a906b254968af4e9b4bdb2d2f0c44336
round[ 3].im_col   b7113ed134e85489b20866b51d4b2c3b
round[ 3].ik_sch   f77d6ec1423f54ef5378317f14b75744
round[ 4].istart   406c501076d70066e17057ca09fc7b7f
round[ 4].is_box   72b86c7c0f0d52d3e0d0da104055036b
round[ 4].is_row   7255dad30fb80310e00d6c6b40d0527c
round[ 4].im_col   ef3b1be1b9b0e64bdcb79f1e0a707fbb
round[ 4].ik_sch   1147659047cf663b9b0ece8dfc0bf1f0
round[ 5].istart   fe7c7e71fe7f807047b95193f67b8e4b
round[ 5].is_box   0c018a2c0c6b3ad016db7022d603e6cc
round[ 5].is_row   0c0370d00c01e622166b8accd6db3a2c
round[ 5].im_col   592460b248832b2952e0b83192304 8f1
round[ 5].ik_sch   dcc1a8b667053f7dcc5c194ab5423a2e
round[ 6].istart   85e5c8042f8614549ebca17b277272df
round[ 6].is_box   672ab1304edc9bfddf78f1033d1e1eef
round[ 6].is_row   671ef1fd4e2a1e03dfdcb1ef3d789b30
round[ 6].im_col   0b8a7783417ae3a1f9492dc0c641a7ce
round[ 6].ik_sch   c6deb0ab791e2364a4055fbe568803ab
round[ 7].istart   cd54c7283864c0c55d4c727e90c9a465
round[ 7].is_box   80fd31ee768c1f078d5d1e8a96121dbc
round[ 7].is_row   80121e0776fd1d8a8d8c31bc965d1fee
round[ 7].im_col   4ee1ddf9301d6352c9ad769ef8d20515
round[ 7].ik_sch   dd1b7cdaf28d5c158a49ab1dbbc497cb
round[ 8].istart   93faa123c2903f4743e4dd83431692de
round[ 8].is_box   2214f132a896251664aec94164ff749c
round[ 8].is_row   22ffc916a81474416496f19c64ae2532
round[ 8].im_col   1008ffe53b36ee6af27b42549b8a7bb7
round[ 8].ik_sch   78c4f708318d3cd69655b701bfc093cf
round[ 9].istart   68cc08ed0abbd2bc642ef555244ae878
round[ 9].is_box   f727bf53a3fe7f788cc377eda65cc8c1
round[ 9].is_row   f75c7778a327c8ed8cfebfc1a6c37f53
round[ 9].im_col   7f69ac1ed939ebaac8ece3cb12e159e3
```

```
round[ 9].ik_sch   60dcef10299524ce62dbef152f9620cf
round[10].istart   1fb5430ef0accf64aa370cde3d77792c
round[10].is_box   cbd264d717aa5f8c62b2819c8b02af42
round[10].is_row   cb02818c17d2af9c62aa64428bb25fd7
round[10].im_col   cfaf16b2570c18b52e7fef50cab267ae
round[10].ik_sch   4b4ecbdb4d4dcfda5752d7c74949cbde
round[11].istart   84e1dd691a41d76f792d389783fbac70
round[11].is_box   4fe0c9e443f80d06affa76854163aad0
round[11].is_row   4f63760643e0aa85aff8c9d041fa0de4
round[11].im_col   794cf891177bfd1d8a327086f3831b39
round[11].ik_sch   1a1f181d1e1b1c194742c7d74949cbde
round[12].istart   6353e08c0960e104cd70b751bacad0e7
round[12].is_box   0050a0f04090e03080d02070c01060b0
round[12].is_row   00102030405060708090a0b0c0d0e0f0
round[12].ik_sch   00010203040506070809a0b0c0d0e0f
round[12].ioutput  00112233445566778899aabbccddeeff
```

C.3 AES-256 (*Nk*=8, *Nr*=14)

```
PLAINTEXT:  00112233445566778899aabbccddeeff
KEY:        000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f

CIPHER (ENCRYPT):
round[ 0].input    00112233445566778899aabbccddeeff
round[ 0].k_sch    000102030405060708090a0b0c0d0e0f
round[ 1].start    00102030405060708090a0b0c0d0e0f0
round[ 1].s_box    63cab7040953d051cd60e0e7ba70e18c
round[ 1].s_row    6353e08c0960e104cd70b751bacad0e7
round[ 1].m_col    5f72641557f5bc92f7be3b291db9f91a
round[ 1].k_sch    101112131415161718191a1b1c1d1e1f
round[ 2].start    4f63760643e0aa85efa7213201a4e705
round[ 2].s_box    84fb386f1ae1ac97df5cfd237c49946b
round[ 2].s_row    84e1fd6b1a5c946fdf4938977cfbac23
round[ 2].m_col    bd2a395d2b6ac438d192443e615da195
round[ 2].k_sch    a573c29fa176c498a97fce93a572c09c
round[ 3].start    1859fbc28a1c00a078ed8aadc42f6109
round[ 3].s_box    adcb0f257e9c63e0bc557e951c15ef01
round[ 3].s_row    ad9c7e017e55ef25bc150fe01ccb6395
round[ 3].m_col    810dce0cc9db8172b3678c1e88a1b5bd
round[ 3].k_sch    1651a8cd0244beda1a5da4c10640bade
round[ 4].start    975c66c1cb9f3fa8a93a28df8ee10f63
round[ 4].s_box    884a33781fdb75c2d380349e19f876fb
round[ 4].s_row    88db34fb1f807678d3f833c2194a759e
round[ 4].m_col    b2822d81abe6fb275faf103a078c0033
round[ 4].k_sch    ae87dff00ff11b68a68ed5fb03fc1567
round[ 5].start    1c05f271a417e04ff921c5c104701554
round[ 5].s_box    9c6b89a349f0e18499fda678f2515920
round[ 5].s_row    9cf0a62049fd59a399518984f26be178
round[ 5].m_col    aeb65ba974e0f822d73f567bdb64c877
round[ 5].k_sch    6de1f1486fa54f9275f8eb5373b8518d
round[ 6].start    c357aae11b45b7b0a2c7bd28a8dc99fa
round[ 6].s_box    2e5bacf8af6ea9e73ac67a34c286ee2d
round[ 6].s_row    2e6e7a2dafc6eef83a86ace7c25ba934
round[ 6].m_col    b951c33c02e9bd29ae25cdb1efa08cc7
round[ 6].k_sch    c656827fc9a799176f294cec6cd5598b
round[ 7].start    7f074143cb4e243ec10c815d8375d54c
round[ 7].s_box    d2c5831a1f2f36b278fe0c4cec9d0329
```

```
round[ 7].s_row     d22f0c291ffe031a789d83b2ecc5364c
round[ 7].m_col     ebb19e1c3ee7c9e87d7535e9ed6b9144
round[ 7].k_sch     3de23a75524775e727bf9eb45407cf39
round[ 8].start     d653a4696ca0bc0f5acaab5db96c5e7d
round[ 8].s_box     f6ed49f950e06576be74624c565058ff
round[ 8].s_row     f6e062ff507458f9be50497656ed654c
round[ 8].m_col     5174c8669da98435a8b3e62ca974a5ea
round[ 8].k_sch     0bdc905fc27b0948ad5245a4c1871c2f
round[ 9].start     5aa858395fd28d7d05e1a38868f3b9c5
round[ 9].s_box     bec26a12cfb55dff6bf80ac4450d56a6
round[ 9].s_row     beb50aa6cff856126b0d6aff45c25dc4
round[ 9].m_col     0f77ee31d2ccadc05430a83f4ef96ac3
round[ 9].k_sch     45f5a66017b2d387300d4d33640a820a
round[10].start     4a824851c57e7e47643de50c2af3e8c9
round[10].s_box     d61352d1a6f3f3a04327d9fee50d9bdd
round[10].s_row     d6f3d9dda6279bd1430d52a0e513f3fe
round[10].m_col     bd86f0ea748fc4f4630f11c1e9331233
round[10].k_sch     7ccff71cbeb4fe5413e6bbf0d261a7df
round[11].start     c14907f6ca3b3aa070e9aa313b52b5ec
round[11].s_box     783bc54274e280e0511eacc7e200d5ce
round[11].s_row     78e2acce741ed5425100c5e0e23b80c7
round[11].m_col     af8690415d6e1dd387e5fbedd5c89013
round[11].k_sch     f01afafee7a82979d7a5644ab3afe640
round[12].start     5f9c6abfbac634aa50409fa766677653
round[12].s_box     cfde0208f4b418ac5309db5c338538ed
round[12].s_row     cfb4dbedf4093808538502ac33de185c
round[12].m_col     7427fae4d8a695269ce83d315be0392b
round[12].k_sch     2541fe719bf500258813bbd55a721c0a
round[13].start     516604954353950314fb86e401922521
round[13].s_box     d133f22a1aed2a7bfa0f44697c4f3ffd
round[13].s_row     d1ed44fd1a0f3f2afa4ff27b7c332a69
round[13].m_col     2c21a820306f154ab712c75eee0da04f
round[13].k_sch     4e5a6699a9f24fe07e572baacdf8cdea
round[14].start     627bceb9999d5aaac945ecf423f56da5
round[14].s_box     aa218b56ee5ebeacdd6ecebf26e63c06
round[14].s_row     aa5ece06ee6e3c56dde68bac2621bebf
round[14].k_sch     24fc79ccbf0979e9371ac23c6d68de36
round[14].output    8ea2b7ca516745bfeafc49904b496089

INVERSE CIPHER (DECRYPT):
round[ 0].iinput    8ea2b7ca516745bfeafc49904b496089
round[ 0].ik_sch    24fc79ccbf0979e9371ac23c6d68de36
round[ 1].istart    aa5ece06ee6e3c56dde68bac2621bebf
round[ 1].is_row    aa218b56ee5ebeacdd6ecebf26e63c06
round[ 1].is_box    627bceb9999d5aaac945ecf423f56da5
round[ 1].ik_sch    4e5a6699a9f24fe07e572baacdf8cdea
round[ 1].ik_add    2c21a820306f154ab712c75eee0da04f
round[ 2].istart    d1ed44fd1a0f3f2afa4ff27b7c332a69
round[ 2].is_row    d133f22a1aed2a7bfa0f44697c4f3ffd
round[ 2].is_box    516604954353950314fb86e401922521
round[ 2].ik_sch    2541fe719bf500258813bbd55a721c0a
round[ 2].ik_add    7427fae4d8a695269ce83d315be0392b
round[ 3].istart    cfb4dbedf4093808538502ac33de185c
round[ 3].is_row    cfde0208f4b418ac5309db5c338538ed
round[ 3].is_box    5f9c6abfbac634aa50409fa766677653
round[ 3].ik_sch    f01afafee7a82979d7a5644ab3afe640
round[ 3].ik_add    af8690415d6e1dd387e5fbedd5c89013
```

```
round[ 4].istart   78e2acce741ed5425100c5e0e23b80c7
round[ 4].is_row   783bc54274e280e0511eacc7e200d5ce
round[ 4].is_box   c14907f6ca3b3aa070e9aa313b52b5ec
round[ 4].ik_sch   7ccff71cbeb4fe5413e6bbf0d261a7df
round[ 4].ik_add   bd86f0ea748fc4f4630f11c1e9331233
round[ 5].istart   d6f3d9dda6279bd1430d52a0e513f3fe
round[ 5].is_row   d61352d1a6f3f3a04327d9fee50d9bdd
round[ 5].is_box   4a824851c57e7e47643de50c2af3e8c9
round[ 5].ik_sch   45f5a66017b2d387300d4d33640a820a
round[ 5].ik_add   0f77ee31d2ccadc05430a83f4ef96ac3
round[ 6].istart   beb50aa6cff856126b0d6aff45c25dc4
round[ 6].is_row   bec26a12cfb55dff6bf80ac4450d56a6
round[ 6].is_box   5aa858395fd28d7d05e1a38868f3b9c5
round[ 6].ik_sch   0bdc905fc27b0948ad5245a4c1871c2f
round[ 6].ik_add   5174c8669da98435a8b3e62ca974a5ea
round[ 7].istart   f6e062ff507458f9be50497656ed654c
round[ 7].is_row   f6ed49f950e06576be74624c565058ff
round[ 7].is_box   d653a4696ca0bc0f5acaab5db96c5e7d
round[ 7].ik_sch   3de23a75524775e727bf9eb45407cf39
round[ 7].ik_add   ebb19e1c3ee7c9e87d7535e9ed6b9144
round[ 8].istart   d22f0c291ffe031a789d83b2ecc5364c
round[ 8].is_row   d2c5831a1f2f36b278fe0c4cec9d0329
round[ 8].is_box   7f074143cb4e243ec10c815d8375d54c
round[ 8].ik_sch   c656827fc9a799176f294cec6cd5598b
round[ 8].ik_add   b951c33c02e9bd29ae25cdb1efa08cc7
round[ 9].istart   2e6e7a2dafc6eef83a86ace7c25ba934
round[ 9].is_row   2e5bacf8af6ea9e73ac67a34c286ee2d
round[ 9].is_box   c357aae11b45b7b0a2c7bd28a8dc99fa
round[ 9].ik_sch   6de1f1486fa54f9275f8eb5373b8518d
round[ 9].ik_add   aeb65ba974e0f822d73f567bdb64c877
round[10].istart   9cf0a62049fd59a399518984f26be178
round[10].is_row   9c6b89a349f0e18499fda678f2515920
round[10].is_box   1c05f271a417e04ff921c5c104701554
round[10].ik_sch   ae87dff00ff11b68a68ed5fb03fc1567
round[10].ik_add   b2822d81abe6fb275faf103a078c0033
round[11].istart   88db34fb1f807678d3f833c2194a759e
round[11].is_row   884a33781fdb75c2d380349e19f876fb
round[11].is_box   975c66c1cb9f3fa8a93a28df8ee10f63
round[11].ik_sch   1651a8cd0244beda1a5da4c10640bade
round[11].ik_add   810dce0cc9db8172b3678c1e88a1b5bd
round[12].istart   ad9c7e017e55ef25bc150fe01ccb6395
round[12].is_row   adcb0f257e9c63e0bc557e951c15ef01
round[12].is_box   1859fbc28a1c00a078ed8aadc42f6109
round[12].ik_sch   a573c29fa176c498a97fce93a572c09c
round[12].ik_add   bd2a395d2b6ac438d192443e615da195
round[13].istart   84e1fd6b1a5c946fdf4938977cfbac23
round[13].is_row   84fb386f1ae1ac97df5cfd237c49946b
round[13].is_box   4f63760643e0aa85efa7213201a4e705
round[13].ik_sch   101112131415161718191a1b1c1d1e1f
round[13].ik_add   5f72641557f5bc92f7be3b291db9f91a
round[14].istart   6353e08c0960e104cd70b751bacad0e7
round[14].is_row   63cab7040953d051cd60e0e7ba70e18c
round[14].is_box   00102030405060708090a0b0c0d0e0f0
round[14].ik_sch   000102030405060708090a0b0c0d0e0f
round[14].ioutput  00112233445566778899aabbccddeeff
```

EQUIVALENT INVERSE CIPHER (DECRYPT):

```
round[ 0].iinput   8ea2b7ca516745bfeafc49904b496089
round[ 0].ik_sch   24fc79ccbf0979e9371ac23c6d68de36
round[ 1].istart   aa5ece06ee6e3c56dde68bac2621bebf
round[ 1].is_box   629deca599456db9c9f5ceaa237b5af4
round[ 1].is_row   627bceb9999d5aaac945ecf423f56da5
round[ 1].im_col   e51c9502a5c1950506a61024596b2b07
round[ 1].ik_sch   34f1d1ffbfceaa2ffce9e25f2558016e
round[ 2].istart   d1ed44fd1a0f3f2afa4ff27b7c332a69
round[ 2].is_box   5153862143fb259514 20403016695e4
round[ 2].is_row   5166049543539503 14fb86e401922521
round[ 2].im_col   91a29306cc450d0226f4b5eaef5efed8
round[ 2].ik_sch   5e1648eb384c350a7571b746dc80e684
round[ 3].istart   cfb4dbedf4093808538502ac33de185c
round[ 3].is_box   5fc69f53ba4076bf50676aaa669c34a7
round[ 3].is_row   5f9c6abfbac634aa5040 fa766677653
round[ 3].im_col   b041a94eff21ae9212278d903b8a63f6
round[ 3].ik_sch   c8a305808b3f7bd043274870d9b1e331
round[ 4].istart   78e2acce741ed5425100c5e0e23b80c7
round[ 4].is_box   c13baaeccae9b5f6705207a03b493a31
round[ 4].is_row   c14907f6ca3b3aa070e9aa313b52b5ec
round[ 4].im_col   638357cec07de6300e30d0ec4ce2a23c
round[ 4].ik_sch   b5708e13665a7de14d3d824ca9f151c2
round[ 5].istart   d6f3d9dda6279bd1430d52a0e513f3fe
round[ 5].is_box   4a7ee5c9c53de85164f348472a827e0c
round[ 5].is_row   4a824851c57e7e47643de50c2af3e8c9
round[ 5].im_col   ca6f71058c642842a315595fdf54f685
round[ 5].ik_sch   74da7ba3439c7e50c81833a09a96ab41
round[ 6].istart   beb50aa6cff856126b0d6aff45c25dc4
round[ 6].is_box   5ad2a3c55fe1b93905f3587d68a88d88
round[ 6].is_row   5aa858395fd28d7d05e1a38868f3b9c5
round[ 6].im_col   ca46f5ea835eab0b9537b6dbb221b6c2
round[ 6].ik_sch   3ca69715d32af3f22b67ffade4ccd38e
round[ 7].istart   f6e062ff507458f9be50497656ed654c
round[ 7].is_box   d6a0ab7d6cca5e695a6ca40fb953bc5d
round[ 7].is_row   d653a4696ca0bc0f5acaab5db96c5e7d
round[ 7].im_col   2a70c8da28b806e9f319ce42be4baaad
round[ 7].ik_sch   f85fc4f3374605f38b844df0528e98e1
round[ 8].istart   d22f0c291ffe031a789d83b2ecc5364c
round[ 8].is_box   7f4e814ccb0cd543c175413e8307245d
round[ 8].is_row   7f074143cb4e243ec10c815d8375d54c
round[ 8].im_col   f0073ab7404a8a1fc2cba0b80df08517
round[ 8].ik_sch   de69409aef8c64e7f84d0c5fcfab2c23
round[ 9].istart   2e6e7a2dafc6eef83a86ace7c25ba934
round[ 9].is_box   c345bdfa1bc799e1a2dcaab0a857b728
round[ 9].is_row   c357aae11b45b7b0a2c7bd28a8dc99fa
round[ 9].im_col   3225fe3686e498a32593c1872b613469
round[ 9].ik_sch   aed55816cf19c100bcc24803d90ad511
round[10].istart   9cf0a62049fd59a399518984f26be178
round[10].is_box   1c17c554a4211571f970f24f0405e0c1
round[10].is_row   1c05f271a417e04ff921c5c104701554
round[10].im_col   9d1d5c462e655205c4395b7a2eac55e2
round[10].ik_sch   15c668bd31e5247d17c168b837e6207c
round[11].istart   88db34fb1f807678d3f833c2194a759e
round[11].is_box   979f2863cb3a0fc1a9e166a88e5c3fdf
round[11].is_row   975c66c1cb9f3fa8a93a28df8ee10f63
round[11].im_col   d24bfb0e1f997633cfce86e37903fe87
round[11].ik_sch   7fd7850f61cc991673db890365c89d12
```

```
round[12].istart   ad9c7e017e55ef25bc150fe01ccb6395
round[12].is_box   181c8a098aed61c2782ffba0c45900ad
round[12].is_row   1859fbc28a1c00a078ed8aadc42f6109
round[12].im_col   aec9bda23e7fd8aff96d74525cdce4e7
round[12].ik_sch   2a2840c924234cc026244cc5202748c4
round[13].istart   84e1fd6b1a5c946fdf4938977cfbac23
round[13].is_box   4fe0210543a7e706efa476850163aa32
round[13].is_row   4f63760643e0aa85efa7213201a4e705
round[13].im_col   794cf891177bfd1ddf67a744acd9c4f6
round[13].ik_sch   1a1f181d1e1b1c191217101516131411
round[14].istart   6353e08c0960e104cd70b751bacad0e7
round[14].is_box   0050a0f04090e03080d02070c01060b0
round[14].is_row   00102030405060708090a0b0c0d0e0f0
round[14].ik_sch   000102030405060708090a0b0c0d0e0f
round[14].ioutput  00112233445566778899aabbccddeeff
```

Appendix D - References

AES page available via http://www.nist.gov/CryptoToolkit.[4]

[2] Computer Security Objects Register (CSOR): http://csrc.nist.gov/csor/.

[3] J. Daemen and V. Rijmen, *AES Proposal: Rijndael*, AES Algorithm Submission, September 3, 1999, available at [1].

[4] J. Daemen and V. Rijmen, *The block cipher Rijndael*, Smart Card research and Applications, LNCS 1820, Springer-Verlag, pp. 288-296.

[5] B. Gladman's AES related home page http://fp.gladman.plus.com/cryptography_technology/.

[6] A. Lee, NIST Special Publication 800-21, *Guideline for Implementing Cryptography in the Federal Government*, National Institute of Standards and Technology, November 1999.

[7] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, New York, 1997, p. 81-83.

[8] J. Nechvatal, et. al., *Report on the Development of the Advanced Encryption Standard (AES)*, National Institute of Standards and Technology, October 2, 2000, available at [1].

[4] A complete set of documentation from the AES development effort – including announcements, public comments, analysis papers, conference proceedings, etc. – is available from this site.