MONOGRAFÍA FUNDAMENTOS TEÓRICOS, MATEMÁTICOS Y ESTADO DEL ARTE DE LAS TEORÍAS BASE DE LA ENCRIPTACIÓN DE DATOS

ANDRÉS FELIPEVÁSQUEZ HENAO

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD DE INGENIERIAS
PROGRAMA INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
PEREIRA
2011

MONOGRAFÍA FUNDAMENTOS TEÓRICOS, MATEMÁTICOS Y ESTADO DEL ARTE DE LAS TEORÍAS BASE DE LA ENCRIPTACIÓN DE DATOS

ANDRÉS FELIPE VÁSQUEZ HENAO

MONOGRAFIA

DIRECTOR DE PROYECTO OMAR IVAN TREJOS BURITICA

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD DE INGENIERIAS
PROGRAMA INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
PEREIRA
2011

Nota de aceptación:

_____
_____
_____
_____
_____
_____

_____
Firma del presidente del jurado

_____
Firma del jurado

_____
Firma del jurado

Pereira 19 Enero de 2011

Anhelando que de esta pequeña semilla de humilde conocimiento, germine algún día una secoya de sabiduría que sirva como cimiento al futuro de la ciencia y la tecnología para el bien de la humanidad…

AGRADECIMIENTOS

Puede que la portada de este documento tenga como autor a una sola persona, pero es imposible decir que un proyecto como este lo puede desarrollar uno solo, siempre existen personas alrededor que con su tiempo, favores, consejos y correcciones hacen posible que lleguemos a una meta y un resultado que satisface las expectativas de aquellos que siempre están pendientes de nuestras victorias y derrotas y por ende esto hace que uno mismo se sienta feliz de cumplir una nueva meta y superar uno de los muchos obstáculos que se presentan en el camino de la vida.

Agradezco primeramente a Dios, quien me hizo más de un milagrito para poder desarrollar este proyecto completamente, a mis padres que en ningún momento me han dado la espalda y me han apoyado en todas las decisiones que tomo a diario, a mis hermanos que tuvieron la paciencia y el entendimiento en los momentos que estuve tan ocupado y además me colaboraron en lo que más estuvo a su alcance, al resto de mi familia que siempre me apoya y esta a la expectativa de mis logros, a mi pareja que tuvo que entender que debía administrar muy bien mi tiempo y me ayudo demasiado, a mis amigos y compañeros que me brindaron su apoyo incondicional y me facilitaron muchas herramientas e información que se encuentra plasmada en este documento y por ultimo quiero dar un agradecimiento especial al director, asesor y colaborador que hizo que todo esto fuera posible, el ingeniero Omar Iván Trejos Buriticá de quien aprendí demasiado, siempre me guió y acompaño en todo el camino y me dio la oportunidad de desarrollar mi proyecto de grado para optar al título de Ingeniero de Sistemas y Computación.

CONTENIDO

pág.

LISTA DE TABLAS

pág.

LISTA DE CUADROS

pág.

LISTA DE FIGURAS

pág.

INTRODUCCIÓN

Desde sus inicios, el hombre desarrolló maneras de comunicarse con los demás y desde entonces, surge la necesidad que parte de su información sea sólo conocida por las personas a quienes está destinada. Esta necesidad de poder enviar mensajes de modo que únicamente fueran entendidos por los destinatarios, hizo que cada civilización, utilizara códigos secretos y métodos de ocultar la información que se considera valiosa, de modo que no sea descifrable por aquellas personas que puedan interceptar estos mensajes.

En Egipto, por ejemplo, la criptografía se encuentra en jeroglíficos no estándares tallados en monumentos. Sin embargo, no se piensa que sean intentos serios de comunicación secreta. En la Grecia clásica, los espartanos transmitían sus mensajes utilizando el la escitala, método que consistía en enrollar en un bastón una cinta de papel y sobre la cual se escribía el mensaje a ser transmitido, posteriormente se desenrollaba y así se mandaba a su destino, lugar en el cual se enrollaba en otro bastón idéntico al primero para así poder entender el mensaje original. En la época del imperio romano en el siglo I a.C., Julio César empleó un código secreto que consistía en sustituir cada letra del mensaje por otra que en el alfabeto estuviese desplazada tres lugares, este método se conoce como “cifrado de Cesar”. Con características similares a este cifrado, siglos más tarde, se generaron otros cifrados como el “cifrado de vigenère” (1586), “Cifrado de Beaufort” (1710) y el “cifrado de Vernam” (1917) el cual empleaba un alfabeto binario.

En el siglo XVI Leon Battista Alberti presenta un manuscrito en el que describe un disco cifrador con el que es posible cifrar textos sin que exista una correspondencia única entre el alfabeto del mensaje y el alfabeto de cifrado, con este sistema, cada letra del texto claro podía ser cifrada con un carácter distinto dependiendo esto de una clave secreta. En el siglo XIX comienzan a desarrollarse diversos sistemas de cifrado con las características poli alfabéticas propuestas por Alberti, que consiste en tener mensajes cifrados con alfabetos que pueden ser diferentes al usado por el mensaje original e incluso de tamaños diferentes. Durante las dos guerras mundiales, la criptología (cuya importancia militar es fácil de comprender) adquirió un alto grado de perfección. Se destacan: 1. el uso de la máquina Enigma, la cual era utilizada por el partido nazi para cifrar la mayor parte de sus comunicaciones secretas, esta consistía de un banco de rotores montados sobre un eje, en cuyos perímetros había 26 contactos eléctricos, uno por cada letra del alfabeto inglés y una de sus principales fortalezas era la dependencia de la clave como tal, contando con el buen numero de más de 3,283,883,513,796,974,198,700,882,069,882,752,878,379,955,261,095,623,685,444,055, 315,226,006,433,616,627,409,666,933,182,371,1,802,769,920,000,000,000 claves posibles; también destacó la máquina Hagelin, en las que a través de ruedas con piñones realiza una cifra similar a la utilizada por el sistema de Beaufort, se destaco y fue muy

conocida la maquina japonés purple que fue clave en el conocido ataque a Pearl Harbor el 7 de diciembre de 1941.

Hasta la Segunda Guerra Mundial, todos los cifrados eran de clave secreta o simétricos, a finales de los años 40, el pionero en la Teoría de la Información, C.E. Shannon, sugirió el uso de una mezcla de transposiciones y sustituciones, lo cual se conoce como cifrado de productos. En los años setentas, IBM retomó la sugerencia de Shannon y desarrolló un sistema al que bautizó como LUCIFER. En 1973 el NBS (Nacional Boreal of Standards) organizó un concurso solicitando un algoritmo de encriptación de datos, teniendo como respuesta de IBM el sistema ya mencionado. En 1974, después de analizar las propuestas y determinar que ninguna parecía adecuada, hizo una segunda petición. IBM presentó otro criptosistema, todo lo anterior fue el preámbulo al nacimiento del DES (Data Encryption Standard), que tras ciertas modificaciones por parte de la NSA, fue adoptado y publicado por la Federal Information Processing Standard en 1977 (actualmente el FIPS 46-3).

DES fue el primer estándar internacional de cifrado y la publicación de sus especificaciones, por la NBS, estimuló una explosión del interés público y académico por la criptografía. Es de interés mencionar que desde su nacimiento DES fue muy criticado; es más, muchos consideraban que sus días estaban contados y sin embargo, no sale de la norma hasta julio de 1998. Con relación a triple-DES, se puede decir que sustituye a DES en 1998 (FIPS PUB 46-3). En la actualidad el criptosistema DES, pese a que ya no es la norma, aún sigue siendo utilizado en algunos medios públicos y privados, como por ejemplo, en el área legal. La última versión de DES, Triple-DES, se utiliza en las áreas financieras y públicas, pero su sucesor y que actualmente está en uso como estándar es el AES (Advanced Encryption Standard) el cual es probablemente un algoritmo de cifrado de los más utilizados hoy en día pues es seguro y eficiente, de hecho es el estándar de cifrado elegido por el Instituto Nacional de estándares y tecnología como estándar de proceso de información federal; en 2001 apareció AES basado en el algoritmo RINJDAEL, que fue el algoritmo ganador del concurso convocado por, el Instituto Nacional de Estándares y Tecnología de EEUU (NIST) pues cumplía las características requeridas perfectamente, garantizando la seguridad hasta los siguientes 20 años.

Aunque se ha publicado más información sobre el criptoanálisis de DES que de cualquier otro cifrado de bloque, el ataque más práctico a la fecha sigue siendo el de fuerza bruta el cual prueba cada una de las posibles claves; además de este método están dentro de los más importantes ataques: balance tiempo–memoria, que es un ataque en que se realiza trabajo de computo previo; diferencial, en el que para romper las 16 rondas completas, requiere de 247 textos planos escogidos y trabaja con computo sobre las cajas de sustitución; lineal, este ataque usa aproximaciones lineales para describir la acción de un bloque de cifrado.

El criptoanálisis es la ciencia opuesta a la criptografía (quizás no es muy afortunado hablar de ciencias opuestas, sino más bien de ciencias complementarias), ya que si ésta trata principalmente de crear y analizar criptosistemas seguros, la primera intenta romper esos sistemas, demostrando su vulnerabilidad: dicho de otra forma, trata de descifrar los criptogramas. El término descifrar siempre va acompañado de discusiones de carácter técnico, aunque se asumirá que descifrar es conseguir el texto en claro a partir de un criptograma, sin entrar en polémicas de reversibilidad y solidez de criptosistemas, para entender bien esta ciencia se describe de manera básica el funcionamiento de un método de criptoanálisis el cual es el generador auto-shrinking que trabaja con cifrados de flujo compuestos de algoritmos generadores de secuencia cifrante, siguiendo esta premisa, se ha emprendido el criptoanálisis de este generador mediante variaciones a la técnica del Guess and Determine. Se ha llevado a cabo una personalización de dicha técnica, mediante lo que se consideran suposiciones bien definidas para romper este generador de secuencia cifrante.

Otra ciencia complementaria al cifrado que se debe mencionar es la estenografía, la cual no consiste en modificar la información sino únicamente ocultarla mediante medios gráficos, de sonidos, de camuflaje entre otros, el principal motor de esta ciencia radica en la necesidad de transmitir información oculta, pero que no se sepa dónde va oculta, lo cual es una desventaja de la criptografía, ya que la información va oculta pero se sabe en donde esta oculta, por ejemplo se puede robar un documento cifrado y tratar de sacarle la información así sea por métodos de fuerza bruta y es posible que se logre o no, pero en el caso de la estenografía es más difícil aun tratar de robar un documento e intentar descifrarlo o buscarle información que puede o no que la tenga.

Todos los métodos y ciencias mencionadas desembocan en aplicaciones de seguridad en redes; la seguridad en las comunicaciones es un tema de vanguardia desde sus inicios. La criptografía aplicada a la seguridad de la información es una herramienta muy importante para asegurar su integridad, confiabilidad y confidencialidad. Por eso, el estudio de las bases teóricas y de las implementaciones prácticas para el intercambio de información segura está adquiriendo un gran auge. Las técnicas criptográficas y de criptoanálisis son temas que constantemente aparecen en el desarrollo de nuevos algoritmos para asegurar dicha confidencialidad. Por estos motivos nacen protocolos, normas, estándares y servicios que buscan protegerse a las debilidades y posibles ataques que sufre la información, desde sus sistemas de comunicaciones basados tanto en software como en hardware, en el ultimo capitulo de este proyecto no se pretende dar más que una idea general del tema: un desarrollo amplio y detallado del mismo necesitaría no solo un capitulo, sino un libro completo.

1. TEORÍA DE LA INFORMACIÓN

1.1. INTRODUCCIÓN

En este proceso comunicativo intervienen básicamente 5 componentes:

- ✓ **Fuente:** componente de naturaleza humana o mecánica que determina el tipo de mensaje que se transmitirá y su grado de complejidad.
- ✓ **Codificador:** Recurso técnico que transforma el mensaje originado por la fuente de información en señales apropiadas según el canal a utilizar.
- ✓ **Canal:** Medio generalmente físico que transporta las señales en el espacio (cumple funciones de mediación y transporte).
- ✓ **Decodificador:** Recurso técnico que transforma las señales recibidas a una forma entendible por el Destino.
- ✓ **Destino:** componente terminal del proceso de comunicación, al cual está dirigido el mensaje.

La fuente y el destino son entidades sobre las cuales poco es posible hacer, pues se constituyen en entidades independientes.

En este capítulo se limitara a estudiar el tipo de fuentes que existen y las propiedades de cada una de ellas. Enfocándose en las tres componentes intermedias, pues es allí donde se lleva a cabo la manipulación de los datos que contienen la información que fluye entre las dos terminales del proceso de comunicación. Es el objetivo de este capítulo es cambiar la percepción sobre lo que se entiende por información, pasando de una visión cualitativa e intuitiva a una visión cuantitativa y analítica de la misma.

Para llegar a una medida de la información, se pasara la idea intuitiva que se tiene de ésta. En primer lugar se presentara que la cantidad de información proporcionada por cierto dato es menor cuanto más se espera ese dato. Si una persona comenta el tiempo que hace en Bogotá, al decir "lluvioso" la información obtenida de este comentario es casi nula, ya es lo que se estaba esperando con mayor probabilidad. Si por el contrario dice que es "soleado", se recibe más información, ya que la probabilidad de que esto ocurra es menor (Ver Tabla 1).

Es posible considerar las informaciones acerca del tiempo en Bogotá como datos que se reciben de cierta variable aleatoria, que cada vez que se interroga, arroja

como respuesta el tiempo que hace en Bogotá. Esta variable aleatoria se convierte en una fuente de información.

La idea de la probabilidad de ocurrencia de cierto evento que arroja información queda ahora modelada por una variable aleatoria y su conjunto de mensajes y probabilidades asociadas. Para el caso anterior, (ver Tabla 1).

La ganancia de información que se experimenta, al recibir un mensaje, se puede entender como la reducción de incertidumbre sobre el estado de la fuente que se experimenta tras recibir el mensaje.

Tabla 1. Estado del tiempo y sus probabilidades asociadas

TIEMPO	PROBABILIDAD
Lluvioso	0.70
Soleado	0.30

1.2. FUENTES DE INFORMACIÓN

Para hablar de información hay que empezar por definir lo que es una fuente de información.

Una fuente de información puede ser modelada como una variable aleatoria; al recibir un mensaje de esa fuente, se obtiene una cantidad de información, que depende sólo de la probabilidad de emisión de ese mensaje; además, la cantidad de información es una función creciente con la inversa de la probabilidad (cuanta menor probabilidad, mayor cantidad de información se recibe, como se observo con en el ejemplo anterior “El clima en Bogotá”).

Desde el punto de vista matemático una fuente de información es un elemento que entrega una señal, y una señal es una función de una o más variables que contiene información acerca de la naturaleza o comportamiento de algún fenómeno. Es decir, se considerara como una señal tanto al fenómeno físico que transporta la información como a la función matemática que representa a ese fenómeno. Cualquiera de las dos formas sirve como soporte a la información.

Las fuentes de información se clasifican basándose en el tipo de señal que entregan y las variables independientes que determinan su estado, en este caso el

tiempo es la variable independiente. De este modo se tienen los siguientes tipos de fuentes:

- ✓ **Fuentes de tiempo continuo:** la función está definida para cualquier valor de la variable independiente.
- ✓ **Fuentes de tiempo discreto:** la función sólo está definida para un conjunto contable de instantes de tiempo.

Pero se pueden clasificar también según el rango de valores que cubren las señales. En este caso los tipos de fuentes de información son:

- ✓ **Fuentes continuas o de amplitud continua:** el valor de la función toma un rango continuo de valores.
- ✓ **Fuentes discretas o de amplitud discreta:** el valor de la función sólo toma un conjunto finito de valores. A cada uno de estos valores lo llamamos símbolo. El conjunto de todos los símbolos se suele llamar alfabeto. La elección del alfabeto es, en cierto modo arbitrario, ya que es posible agrupar varios símbolos para crear otros.

Estas dos clasificaciones son ortogonales. Es decir, existen fuentes continuas de tiempo continuo, fuentes continuas de tiempo discreto, fuentes discretas de tiempo continuo y fuentes discretas de tiempo discreto. Aunque en la práctica sólo se encuentran dos tipos: las llamadas Fuentes Analógicas, que son fuentes continuas de tiempo continuo; y las llamadas Fuentes Digitales, que son fuentes discretas de tiempo discreto.

Las fuentes digitales se clasifican según la relación que tenga un símbolo con los símbolos que han precedido su aparición. Así, las fuentes de información también se clasifican en:

- ✓ Fuentes sin memoria: los símbolos son estadísticamente independientes entre sí. Los símbolos que hayan aparecido hasta el momento no van a condicionar al símbolo presente ni a posteriores.
- ✓ Fuentes con memoria: la aparición de los símbolos no es estadísticamente independiente. Es decir, si han aparecido M-1 símbolos, el símbolo M-ésimo está condicionado por los anteriores.

1.3. MODELO MATEMÁTICO DE LA TEORÍA DE LA INFORMACIÓN

El modelo matemático que sustenta lo que hoy se llama teoría de la información nace con los trabajados de Claude Shannon 1948 y de Norbert Wiener en 1949, con estos trabajos se establecieron los fundamentos de la moderna teoría estadística de la comunicación. Ambos hombres investigaban la manera de extraer información a partir de un fondo ruidoso y ambos aplicaron conceptos estadísticos al problema.

Wiener trato el caso en el cual, las señales portadoras de información están, en todo o en parte, fuera del control del diseñador y todo el procedimiento debe realizarse en el terminal receptor. El planteo el problema así: "dado un conjunto de posibles señales, no escogidas por nosotros, mas el inevitable ruidos, como es posible hacer el mejor estimativo de los valores presentes y futuros de la señal recibida?". Las soluciones óptimas a este problema dieron origen a la denominada teoría de la detección.

El trabajo de Shannon trato, por su parte, con el caso en el cual el procesamiento de la señal puede darse en ambos extremos: Transmisor y receptor. Shannon planteo el problema de esta manera: “dado un conjunto de posibles mensajes que puede generar una fuente, no escogidos por nosotros, cual es la mejor forma de representar tales mensajes de tal manera que pueda transportar óptimamente la información sobre un sistema sujeto a las limitaciones físicas inherentes a los sistemas reales?”

Para tratar este problema de forma adecuada, es necesario concentrarse más en la información en sí que en las señales que la transporta. Para esto Shannon trato de hallar la repuesta a las siguientes preguntas:

- ✓ ¿En qué forma precisa es restringida la transmisión por las limitaciones físicas del medio? (Atenuación, Ancho de banda y Ruido).
- ✓ ¿Existe un sistema ideal de comunicación y, si es así, cuáles son sus características?
- ✓ ¿Cómo pueden compararse los sistemas de comunicación existentes con el sistema ideal y como puede ser mejorado su comportamiento?

La teoría de la información es pues un cuerpo matemático que trata de responder a las preguntas anteriores a través de la definición precisa de tres conceptos básicos: la medida de la información, la capacidad de un canal de comunicación

para transferir información, y la codificación como un medio para utilizar los canales al límite de su capacidad. Estos tres conceptos están enlazados en el que se ha dado en llamar el teorema fundamental de la teoría de la información, expresado así por C. Shannon y W. Wearver en su libro “The mathematical theory of comunication”:

"*Dada la fuente de información y un canal de comunicación, existe una técnica de codificación tal que la información puede ser transmitida sobre el canal a cualquier velocidad inferior a la capacidad del canal y con una frecuencia arbitrariamente pequeña de errores a pesar de la presencia de ruido."*

El aspecto más sorprendente y asombroso de este teorema lo constituye la posibilidad de transmisión libre de errores sobre un canal ruidoso por medio del uso de la codificación, pero advierte que la velocidad de transmisión se ve afectada por el ruido, lo cual es lógico pues en presencia de ruido se hace necesaria una codificación redundante de los datos.

En esencia la codificación se usa para acoplar la fuente y el canal con miras a lograr la máxima eficiencia en la transmisión de información, en forma análoga al acople de impedancias que busca la máxima transferencia de potencia.

A continuación se expondrán, cada uno de los conceptos de la teoría de la información:

Shannon y Weaver relacionan la cantidad de información de un mensaje con el grado de incertidumbre que este representaba para el receptor. Esto a su vez, se relaciona con la probabilidad de recibir un mensaje dado entre un conjunto de mensajes posibles, a más improbable el mensaje, mayor información con lleva para el usuario. Alternativamente en el lado transmisor, la medida de la información es una indicación de la libertad de escogencia ejercida por la fuente al seleccionar el mensaje.

Shannon postulo entonces que la medida de la información de un mensaje era función de la probabilidad de su aparición.

Para un mensaje **A** con probabilidad de aparición p_a, se tiene que:

(1)

$I_a = f(P_a)$

Un razonamiento intuitivo sugiere que la función f() cumple las siguientes propiedades:

(2)

$f(P_a) \geq 0$ Donde $0 \leq P_a \leq 1$

(3)

$f(P_a) \geq f(P_b)$ Para $P_a < P_b$

(4)

$\lim\_ f(P_a) = 0$

$P_a \rightarrow 1$

Además si dos mensajes A y B son recibidos, la cantidad de información transportada por el mensaje compuesto C=AB es I_c, para esta situación se pueden presentar los siguientes casos:

(5)

$I_c = I_a + I_b = f(P_a) + f(P_b)$ Si A y B son independientes

(6)

$I_c = f(P(A \cap B))) = f(P(A)) + f(P(A \mid B))$

Si A y B no son independientes, en este caso el orden en el cual los mensajes fueron recibidos determina la cantidad de información del mensaje compuesto.

Además, si los mensajes proceden de la misma fuente y son estadísticamente independientes, se tiene que:

(7)

$P_c = P_a P_b$

De (5) y (7) se tiene que:

(8)

$f(P_a P_b) = f(P_a) + f(P_b)$

Existe una y solo una función que satisface las condiciones dadas por (2), (3), (4) y (7) y tal función es el Logaritmo. Por lo tanto Shannon definió la información transportada por un mensaje A como:

(9)

$I_a = - \log_b P_a$

Especificar la base “b” equivale a seleccionar la unidad de información. La convención es tomar b=2 con la que la unidad correspondiente viene a ser el bit, palabra originada de la contracción de bynary digit propuesta por J. W. Tukey.

Por tanto:

(10)

$I_a = - \log_2 (P_a)$ bits

Esta escogencia de la base se debió a que la elección más simple es la que se presenta cuando se tiene que seleccionar uno de dos mensajes equiprobables ($P_a=P_b=1/2$) con lo que se obtiene:

(11)

$I_a = - \log_2 (1/2) = 1$ bit

De aquí en adelante el 2 estará implícito en la expresión logarítmica.

Una medida de la información promedia producida por una fuente viene dada por la entropía de la fuente. Suponiendo una fuente discreta cuyos símbolos son estadísticamente independientes. Si la fuente puede reproducir m símbolo con posibilidad p_1 a p_m cuando el j-esimo símbolo es transmitido este conlleva $I_j=\text{Log}(1/p_j)$ bits de información. Si se transmiten N símbolos (N>>1), el j-esimo símbolo se presenta alrededor de Npj veces y la información total transportada por los N símbolos será:

(12)

$$It = \sum_{j=1}^{m} \text{NPj Ij}$$

La información promedio por símbolo determina entonces la entropía de la fuente discreta, que se define como:

(13)

$$H \sum_{j=1}^{m} \mathrm{Pj\ Ij} = \sum_{j=1}^{m} \mathrm{Pj} \log(1/\mathrm{Pj})\ \text{bits / simbolo}$$

En resumen la entropía de una fuente discreta significa simplemente que aunque no se sea capaz de decir cuál será el siguiente símbolo producido por la fuente, es posible esperar, en promedio H bits de información por símbolo o NH bits en un mensaje de N símbolos si N es grande.

Por un razonamiento sencillo puede demostrarse que la entropía de una fuente es máxima cuando los m símbolos que es capaz de producir son equiprobables. En tal caso dicha entropía venda dada por:

(14)

$H = H_{max} = - \log (1/m) = \log (m)$

De lo anterior se deduce que para una fuente binaria para la cual los dos símbolos no son equiprobables, la información promedio por símbolo (la entropía de la fuente) es inferior a un bit.

Para describir completamente a una fuente discreta de información, no basta con definirle su entropía sino que además es necesario determinar su rata de entropía o rata de información en bits por segundo (bps). Esta se define como:

(15)

$R = H / t_{prom}$ bps

Donde t_{prom} es la duración promedio por símbolo. Es decir:

(16)

$$t\ prom = 1/N \sum_{j=1}^{m} \mathrm{NPj\ tj} \ = \sum_{j=1}^{m} \mathrm{Pj\ tj}$$

Aquí tj es la duración del j-esimo símbolo.

1.4. CANALES DE COMUNICACIÓN

El canal es medio de transmisión que va entre la fuente y el destino del mensaje y en el cual, la señal está sujeta a tres fenómenos físicos: La atenuación, la distorsión y el ruido.

La señal se atenúa por qué parte de su energía se convierte en calor durante la transmisión. La señal se distorsiona debido a que el medio de transmisión contiene elementos que almacenan energía en los que un cambio en la energía almacenada requiere una cantidad definida de tiempo.

La velocidad a la que el medio puede cambiar su energía almacenada se determina a través de su respuesta de frecuencia la cual puede expresarse en términos del ancho de banda del sistema.

La señal se contamina con ruido durante su tránsito entre la fuente y el destino debido a interferencias externa y el ruido térmico producido por el movimiento aleatorio de átomos y electrones que conforman la materia del medio de transmisión. La relación entre la potencia de la señal y la potencia del ruido, expresada en decibelios, se denomina la relación señal a ruido. Esta relación resume la incidencia que sobre la señal tienen los fenómenos antes descritos de la atenuación y el ruido, y debe en un sistema bien concebido, mantenerse lo mas alta posible.

Como lo demostró Nyquist en su publicación de 1928, un canal de comunicaciones de ancho de banda W puede transmitir hasta 2W muestras independientes de la señal por segundo. Suponiendo que cada muestra corresponde a uno de m niveles posibles de señal. De aquí resulta que el número de bits por muestra es Log (m), por lo que el canal puede transmitir 2W Log (m) bps. Esta cantidad se define como la capacidad del canal y se denota por C. Es decir:

- ✓ C = 2W log (m) bps

Shannon analizo la capacidad de un canal ruidoso y encontró que esta podía ser expresada como:

- ✓ $C = 2W \log (1 + S/N)$ bps (Ley de Hartley – Shannon)

Donde S/N es la relación señal a ruido que se calcula por la expresión:

- ✓ $S/N = 10 \log_{10} (P_s/P_r)$

Donde P_s y P_r son la potencia de señal y la potencia de ruido respectivamente.

La capacidad es entonces la medida de la cantidad de información que un canal puede transferir por unidad de tiempo. Por ejemplo, un módem de 33.600 baudios puede transmitir 33.600 dígitos binarios por segundo, su velocidad de transmisión de símbolos es k = 33600 sb/s.

La ley de Hartley - Shannon también dice que para una rata de información dada, es posible reducir la potencia de la señal siempre que incrementemos el ancho de banda en la cantidad apropiada y viceversa. Esto lleva al importante y útil resultado de que es posible la compresión del ancho de banda.

La velocidad de transmisión de símbolos del canal k, medida en sb/s, está directamente relacionada con la capacidad del canal y la codificación empleada para representar los mensajes emitidos por la fuente. Es posible concluir que la velocidad promedio a la que se transmite la información de una fuente por un canal, medida en bits/s. Depende de:

- ✓ La entropía de la fuente H(S). Es decir, de la información transmitida en promedio con cada mensaje, medida en bits/msj.
- ✓ La longitud media n del código empleado para transmitir los mensajes, medida en sb/msj.

1.5. CODIFICACIÓN

1.5.1. Introducción

Un código permite acoplar la fuente y el canal para lograr una transmisión de información optima.

Del teorema fundamental de la teoría de la información se concluye que la salida de una fuente de mensajes puede ser codificada de tal forma que el número medio de bits por palabra de código sea igual a H, la entropía de la fuente.

Se define entonces la eficiencia de un código como la relación (expresada en porcentaje) de la entropía H de la fuente y el número medio de bits por palabra de código.

La codificación de una fuente se realiza mediante la representación cada mensaje de la fuente por medio de una secuencia de símbolos de un alfabeto. Un alfabeto es un conjunto finito de símbolos, $A = \{a_1, a_2,..., a_n\}$. El alfabeto se elige de forma que sus símbolos se puedan transmitir eficiente y correctamente a través del canal.

En sistemas de comunicación digital usamos un alfabeto binario. Es decir:

- $A = \{0, 1\}$

El conjunto de secuencias binarias asignadas a los mensajes de la fuente se denomina código, y cada elemento es una palabra código.

Un ejemplo es el código ASCII La fuente es el conjunto de caracteres ASCII {26 letras minúsculas, 26 letras mayúsculas, 10 dígitos numéricos y diversos caracteres especiales y signos de puntuación}. Las palabras código son secuencias de 7 símbolos binarios (Ejemplo: carácter('a')=1000011).

Se presentara otro ejemplo. Dada la fuente S = {x1, x2, x3, x4} y el alfabeto A = {0, 1}, un posible código es el que se presenta:

Xi:	x1	x2	x3	x4
C (xi):	0	11	00	01

Un código se puede considerar como una aplicación inyectiva que asocia a cada mensaje una tupla de símbolos del alfabeto, ha de ser inyectiva para que a cada mensaje le corresponda una palabra distinta.

Según el número de símbolos del alfabeto, los códigos se clasifican en binarios, ternarios o en general, n-arios. Por ejemplo, el código ASCII es binario. El código Morse es ternario, pues su alfabeto consiste en A = {·,-, pausa}.

La longitud de una palabra código Ci es el número de símbolos que contiene (orden de la tupla), se denota como L (Ci).

1.5.2. Propiedades de los Códigos

Los códigos poseen propiedades que los hacen particularmente diferentes. Tales propiedades son:

Longitud Media: cada palabra de código asignada a cada símbolo del alfabeto fuente tiene una longitud L_k. A partir de aquí se define la longitud media de un código como:

$$\mathrm{Lm} = \mathrm{Lprom} = \sum_{k=1}^{n} \mathrm{Pk}\,\mathrm{Ik}$$

Donde Pi es la probabilidad del mensaje Ci

La longitud media representa el número medio de bits por símbolo del alfabeto fuente que se utilizan en el proceso de codificación, también se denomina tasa del código (Rate).

Eficiencia: A partir del concepto de longitud media la eficiencia de un código se define como:

$$n = L_{min} / L$$

Siendo:

$$L_{min} < L$$

Para calcular L_{min} es necesario tener en cuenta el primer teorema de Shannon o teorema de la codificación de la fuente:

Dada una fuente discreta de entropía H, la longitud media de la palabra de código está acotada inferiormente por H. Teniendo esto en cuenta L_{min} se fija como el valor de la entropía con lo que la eficiencia puede escribirse como:

$$n = H/L$$

Redundancia: se denomina redundancia de un código a la información superflua o innecesaria para interpretar el significado de los datos originales. Se define como:

$$\sigma = 1 - n$$

1.5.3. Clasificación de los Códigos

Los códigos se pueden clasificar de varias formas dependiendo de ciertas características presentes en su formación.

- **Códigos de longitud fija**

En la codificación de longitud fija, se asignan palabras de código de longitud iguales a cada símbolo en un alfabeto A sin tener en cuenta sus probabilidades. Si el alfabeto tiene M símbolos diferentes (o bloques de símbolos), entonces la longitud de las palabras de código en bits es el entero más pequeño mayor o igual que $\log_2 M$.

Dos esquemas de codificación de longitud fija comúnmente usados son los códigos naturales y los códigos Gray. Puede demostrarse que la codificación de longitud fija sólo es óptima cuando:

- El número de símbolos es igual a una potencia de dos.
- Todos los símbolos son equiprobables.

Sólo entonces podría la entropía de la fuente ser igual a la longitud promedio de las palabras código que es igual a la longitud de cada palabra código en el caso de la codificación de longitud fija. A menudo, algunos símbolos son más probables que otros en este caso sería más ventajoso usar una codificación diferente a la longitud fija para aprovechar esta característica.

Es posible concluir que un código de longitud fija cumple con la siguiente expresión:

$$L(C_i) = L(C_j) \text{ para todo } i,j <= n$$

Por ejemplo, el código ASCII es un código uniforme de longitud de palabra 7.

Para un código uniforme, L_m es la longitud constante de palabra.

- **Códigos de longitud variable o no uniforme**

El método más simple de compresión sin pérdidas consiste en reducir únicamente la redundancia de la codificación. Esta redundancia está normalmente presente en cualquier codificación binaria. Dicha redundancia se puede eliminar construyendo un código de longitud variable. Esta clase de códigos pretende que a cada símbolo del alfabeto fuente le corresponda una palabra de código de longitud mínima según algún criterio de minimización dado.

Es posible incrementar la velocidad de transmisión de los mensajes de una fuente, usando un código no uniforme tal que los mensajes más probables se codifiquen con las palabras más cortas. Así, la mayor parte de las veces se estará transmitiendo palabras cortas. Equivale a disminuir la longitud media L_m, pues las longitudes $L(C_i)$ pequeñas corresponderán a los mayores factores de peso P_i. A menor longitud media, mayor Bit Rate (menor tamaño de archivo o mayor velocidad de transmisión de información), y por tanto, mayor relación de compresión.

Los códigos variables son la clave de la compresión y cabe anotar que es la idea que capturaba el código Morse. El código Morse es un código no uniforme, pues contiene palabras de distinta longitud.

Por ejemplo, C('E') = “·” y C('Z') =” - - ··”

La clave para comprimir es usar un código con la mínima longitud media posible, teniendo en cuenta que existe un límite teórico (H<=Lm, donde H es la entropía de la fuente).

- **Códigos de traducción única**

Al recibir la secuencia de símbolos correspondiente a una secuencia de mensajes, ésta puede decodificarse sin ambigüedad, esta clase de códigos también son llamados unívocamente decodificable.

- Ejemplo de código no unívocamente decodificable

Xi:	x1	x2	x3	x4
C (xi):	1	10	11	01

Porque existe al menos una secuencia de símbolos que admite más de una decodificación: 1011 puede decodificarse como x2x3, x1x4x1 o x2x1x1.

- Ejemplo de código unívocamente decodificable

Xi:	x1	x2
C (xi):	0	01

El símbolo 1 hace de delimitador. Por ejemplo, la secuencia 0010001 sólo admite la decodificación x1x2x1x1x2.

- **Código instantáneo**

Es posible decodificar al terminar de recibir cada palabra, sin necesidad de esperar a las palabras que le suceden Esto ocurre si y sólo si ninguna palabra es prefijo de otra (código prefijo).

- Ejemplo de código no instantáneo

Xi:	x1	x2	x3	x4
C (xi):	0	01	110	1110

La secuencia 0110 sólo admite una decodificación: x1x3, pero hay que esperar al final de la segunda palabra para poder decodificar la primera.

- Ejemplo de código instantáneo

Xi:	x1	x2	x3	x4
C (xi):	0	10	110	1110

Ejemplo, suponiendo que se emite la secuencia de mensajes x1x3x4, codificada como 01101110. Es Posible decodificar la secuencia 01101110 al final de cada palabra.

Teorema 1. Todo código C uniforme es instantáneo. El reciproco no es cierto.

- Ejemplo de código instantáneo y no uniforme

Xi:	x1	x2	x3	x4
C (xi):	0	10	110	1110

Teorema 2.- Todo código C instantáneo es de traducción única. El reciproco no es cierto.

- Ejemplo de código de traducción única y no instantáneo

Xi:	x1	x2
C (xi):	0	01

- **Código óptimo**

Dada una fuente S y un alfabeto A, decimos que un código C es óptimo si y sólo si C es instantáneo y no existe otro código instantáneo con menor longitud media.

Los códigos óptimos no son únicos.

Antes de estudiar cómo construir códigos óptimos, se estudiara cómo construir códigos instantáneos y una condición necesaria y suficiente para la existencia de estos.

Teorema 3.- (Desigualdad de Kraft-McMillan) Sea un código binario C con m palabras de longitudes n1, n2,... nm. Si C es instantáneo, entonces verifica necesariamente:

$$K(C) = \sum_{\mathrm{i}=1}^{m} 2^{-\mathrm{ni}} \leq 1$$

- Ejemplo de un código instantáneo C

Xi:	x1	x2	x1	x2
C (xi):	0	01	110	1111
ni:	1	2	3	4

Por ser instantáneo, debe verificar la desigualdad de Kraft-McMillan. En efecto:

$K(C)= 2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} = 15/16 <= 1$

Teorema 4.- Dado un conjunto de enteros n1, n2, ... nm, que satisfacen la desigualdad de Kraft-McMillan, siempre es posible encontrar un código instantáneo C cuyas longitudes de palabra son n1, n2, ... nm. El teorema establece una condición suficiente para la existencia de un código instantáneo con unas longitudes de palabra dadas. Tanto este teorema, como el anterior, son también válidos para códigos n-arios en general.

1.5.4. Tipos de codificación

Existen diferentes tipos de codificación y cada una tiene aplicaciones específicas. A continuación se describirán cada uno de los diferentes tipos de codificación.

- **Codificación en la fuente**

El objetivo de la codificación es obtener una representación eficiente de los símbolos del alfabeto fuente. Para que la codificación sea eficiente es necesario tener un conocimiento de las probabilidades de cada uno de los símbolos del alfabeto fuente.

El dispositivo que realiza esta tarea es el codificador de la fuente. Este codificador debe cumplir el requisito de que cada palabra de código debe decodificarse de forma única, de forma que la secuencia original sea reconstruida perfectamente a partir de la secuencia codificada.

- **Codificación del canal**

En ocasiones se producen diferencias entre las secuencias de datos enviadas a través de un canal y las secuencias de datos recibidas debidas a la existencia de ruido en el canal. A estas diferencias se les denomina errores. Por ello es necesario realizar una codificación a la entrada del canal, cuyo objetivo es que el receptor sea capaz de detectar y corregir los errores producidos en los datos durante su transmisión por el canal.

La codificación del canal consiste en introducir redundancia, de forma que sea posible reconstruir la secuencia de datos original de la forma más fiable posible.

Aun aplicando una codificación del canal redundante, los errores pueden aparecer es allí donde se hace necesario aplicar las llamadas técnicas de detección y corrección de errores. Se presentara en qué consisten cada una de estas técnicas:

- Detección de Errores o Corrección Hacia Atrás (ARQ - Automatic Repeat Request): cuando el receptor detecta un error solicita al emisor la repetición del bloque de datos transmitido. El emisor retransmitirá los datos tantas veces como sea necesario hasta que los datos se reciban sin errores.

- Corrección de Errores o Corrección Hacia Delante (FEC - Forward Error Correction): se basa en el uso de códigos autocorrectores que permiten la corrección de errores en el receptor.

1.6. DIFUSIÓN PERFECTA EN CRIPTOGRAFÍA

Según Shannon, la criptografía se debe basar en dos principios, confusión y difusión que, trabajando en conjunto, puedan proveer de la seguridad deseada. La confusión se fundamenta en la modificación de los símbolos del mensaje original, mientras que la difusión procura ocultar las estadísticas que puedan aparecer en ese mensaje.
La difusión, el objeto de este estudio, consiste en cambiar el orden de los símbolos del mensaje original, pero sin modificarlos; se puede pensar en ello como una permutación de los símbolos. Como es bien sabido, para un mensaje de *n* símbolos, hay *n!* permutaciones posibles; la meta es lograr codificar todas las permutaciones de la forma más eficiente posible, hablando en términos informáticos se trata de ocupar la menor cantidad de bits posibles.
Con el sistema binario de numeración, para poder asignar un número único a *m* elementos se precisan | *log2 (m)* | bits (| *...* | indica entero superior), así pues para lograr codificar *n!* elementos, se precisa de | *log2(n!)* | bits. A continuación se muestra una correspondencia que recoge los bits necesarios para almacenar las permutaciones de forma ideal:

Nº	**Bits necesarios**	**Nº**	**Bits necesarios**
1	0	**17**	48,33760331
2	1	**18**	52,50752831
3	2,584962501	**19**	56,75545583
4	4,584962501	**20**	61,07738392

5	6,906890596	**21**	65,46970134
6	9,491853096	**22**	69,92913296
7	12,29920802	**23**	74,45269492
8	15,29920802	**24**	79,03765742
9	18,46913302	**25**	83,68151361
10	21,79106111	**26**	88,38195333
11	25,25049273	**27**	93,13684083
12	28,83545523	**28**	97,94419575
13	32,53589495	**29**	102,8021767
14	36,34324987	**30**	107,7090673
15	40,25014047	**31**	112,6632637
16	44,25014047	**32**	117,6632637

Así pues, según lo anterior, con 118 bits, debería poder construirse todas las permutaciones de 32 elementos y aun así sobrarían combinaciones, la forma más evidente de conseguirlo es asociar a cada número natural una permutación única.
Se enfrenta a un problema que trata de conseguir que una función biyectiva entre un conjunto de los números naturales (incluyendo el 0) y el conjunto de todas las permutaciones que se pueden lograr con N elementos, es decir, si se llama *Sn* al grupo de permutaciones de *n* elementos, lograr una función *f* tal que a cada valor del conjunto *0..(n!-1)* le asocie un elemento distinto de *Sn*, y una función *f-1* que a cada elemento de *Sn* le asocie un número natural del conjunto *0..(n!-1)*.

De conseguir esto, se tiene garantizado que se podrá almacenar las permutaciones en el menor espacio posible, se podría construir permutaciones a partir de números al azar o provenientes de algún sitio, como por ejemplo subclaves, y utilizarlo como un generador de difusión perfecta en el que son posibles todas las permutaciones y no se producen colisiones entre ellas.
Para abordar la construcción se va a utilizar de un conjunto de datos más manejable, con menos elementos. El objetivo es lograr asociar de una forma lógica y extensible una permutación distinta a cada número de 0 a 5. Al decir extensible, se quiere decir que el método sirva para cualquier tamaño del conjunto de datos de entrada.

Supóngase que los datos de entrada son *“ABC”* ahora véase todas las posibles permutaciones:

Permutación	**Nº asociado**
AB	0
BA	1

Permutación	**Nº asociado**
ABC	0,0
BAC	0,1
ACB	1,0
BCA	1,1
CAB	2,0
CBA	2,1

A cada permutación se le ha asociado un número que será clave en el proceso. El proceso de construcción es el siguiente: si se tuviese un sólo elemento *"A"* no hay más que una permutación.

Imagínese que se tiene dos elementos:

AB, ahora hay dos permutaciones (*AB y BA)*; la permutación AB llevará el número asociado 0 y la permutación BA llevará el número asociado 1. En otras palabras, al introducir un nuevo elemento *B se* empieza a introducirlo desde el final, a cada paso a la izquierda que se mueva, se incrementa en una unidad ese valor. Para insertar *C* se repetiría el procedimiento sobre una pareja (*AB o BA*) ya existente. El nuevo elemento podrá ir a parar a la derecha del todo (0), entre ambos (1) y al principio (2). Este número se agrega a la izquierda del número que tenía anteriormente la pareja.

Véase más despacio, supóngase que se tiene *BA*, correspondiente al número asociado 1; se va a introducir el elemento *C* y véase las distintas posibilidades:

_B_A_(1):

- _B_A**C** (**0**,1)
- _B**C**A_ (**1**,1)
- **C**B_A_ (**2**,1)

Si se quisiera realizar la misma operación con *4* elementos, habría *4* lugares posibles a donde ir el elemento *D*, y se le asociarían números del *0* al *3*. Se ve fácilmente que para introducir el elemento *n-ésimo* hace falta número de *0* a *(n-1)*. Así, cualquier permutación de *n* elementos quedará definida de forma unívoca como una secuencia de números de la siguiente forma:

$$(X_{0...n-1}, \dots, X_{0..3}, X_{0...2}, X_{0...1})$$

Donde el valor de $X_{0...k}$ es un valor comprendido entre *0* y *k*.

Ahora llega el último paso, codificar esa secuencia en un sólo número. Para ello, se utilizará algo parecido a los sistemas de numeración:

$$X_{0...n-1}\cdot(n-1)! + \dots + X_{0...3}\cdot 3! + X_{0...2}\cdot 2! + X_{0...1}\cdot 1! = N < n!$$

Se puede demostrar que si todas las $X_{0...k}$ adquieren su valor máximo *k*, $N = n!-1$, y todo $N < n!$ puede ser escrito de forma única. De la misma forma, dado un $N < n!$ se puede obtener la secuencia de números, según la división $D = d \cdot c + r$, tomando como divisor *(n-1)!*, después *(n-2)!* y así sucesivamente:

$$N = X_{0..n-1} \cdot (n-1)! + R1 \qquad \textbf{(1}$$
$$R1 = X_{0..n-2} \cdot (n-2)! + R2 \qquad \textbf{(2}$$
$$\dots$$
$$Rn\text{-}2 = X_{0..1} \cdot 1! \qquad \textbf{(n-1}$$

Si se sustituye hacia arriba, se tiene:

$$N = X_{0...n-1}\cdot(n-1)! + \dots + X_{0...3}\cdot 3! + X_{0...2}\cdot 2! + X_{0...1}\cdot 1!$$

En la última ecuación no hay resto porque $1! = 1$ y $R_{n-2} = X_{0...1}$.

Se ha conseguido, por tanto, crear una función que es capaz de asociar a cada número entero una permutación, y a cada permutación un número entero, de forma unívoca. Se establece de esta forma una biyección entre ambos grupos.

Véase un ejemplo aplicado a la criptografía, se dispone de un criptosistema que procesa los mensajes en bloques de 8 elementos. Se desea crear una capa de difusión en la que todas las permutaciones sean posibles. Según lo visto, deberá

ser posible codificar todas las **40320** permutaciones en los números desde 0 hasta **40319**, y para ello se emplearán 16 bits de almacenamiento.

1) Dado el mensaje ***ABCDEFGH*** si se obtiene ***DGFAHCBE***, esta permutación corresponde a la secuencia ***(3,5,4,0,3,1,0)*** y el número de esta permutación es:

$$\boldsymbol{N} = 3 \cdot 7! + 5 \cdot 6! + 4 \cdot 5! + 0 \cdot 4! + 3 \cdot 3! + 1 \cdot 2! + 0 = \boldsymbol{19220}$$

2) Dado el número **34759**, véase que permutación es:

Dividendo	**Divisor**	**Cociente**	**Resto**
34759	5040	**6**	4519
4519	720	**6**	199
199	120	**1**	79
79	24	**3**	7
7	6	**1**	1
1	2	**0**	1
1	1	**1**	0

Corresponde a la secuencia ***(6,6,1,3,1,0,1)*** y es la permutación ***GHBEADFC***.

La capa de difusión de este hipotético criptosistema podría tomar varios bits de la clave, formar un número, y con él crear una configuración distinta para cambiar la difusión; con esto se pretende que el criptosistema sea mucho más resistente a los ataques. Además este algoritmo se podrá utilizar en otras aplicaciones.

2. HISTORIA ANTIGUA A INICIOS DE LA MODERNA

2.1. INTRODUCCIÓN

La criptografía es la disciplina que se encarga del estudio de códigos secretos o llamados también códigos cifrados (en griego *kriptos* significa secreto y *gráhos*, escritura), el arte de enmascarar los mensajes con signos convencionales, que sólo cobran sentido a la luz de una clave secreta, nació con la escritura.

Es una disciplina muy antigua, sus orígenes se remontan al nacimiento de nuestra civilización. En sus orígenes, su único objetivo era el proteger la confidencialidad de informaciones militares y políticas, su rastro se encuentra ya en las tablas cuneiformes, y los papiros demuestran que los primeros egipcios, hebreos, babilonios y asirios conocieron y aplicaron sus inescrutables técnicas.

Entre el Antiguo Egipto e Internet, los criptogramas han protagonizado buena parte de los grandes episodios históricos y un sinfín de anécdotas. Existen mensajes cifrados entre los 64 artículos del Kamasutra, el manual erótico hindú del Vatsyayana, abundan en los textos diplomáticos, pueblan las órdenes militares en tiempos de guerra y, por supuesto, son la esencia de la actividad de los espías. Sin embargo, en la actualidad es una ciencia interesante no sólo en esos campos, sino para cualquier otro que esté interesado en la confidencialidad de unos determinados datos, que alcanzan hoy su máxima expresión gracias al desarrollo de los sistemas informáticos y de las redes mundiales de comunicación.

Los jeroglíficos egipcios, la escítala espartana, el cifrado mesopotámico, el cifrario de César, el sistema de Polybios, la cifra del Kama-Sutra, entre los que se verán en este capítulo son ejemplos de sustitución (cada una de las letras del mensaje original tiene una correspondencia fija en el mensaje cifrado) y de sustitución (las letras simplemente se cambian de sitio o se transponen, por tanto las letras son las mismas en el mensaje original y en el cifrado. En términos de pasatiempos se dice que las letras se trasponen o se anagraman).

En principio, parece muy difícil descubrir el mensaje cifrado por cualquiera de estos procedimientos, pero una persona inteligente y observadora puede descifrar el secreto sin demasiada dificultad.

2.2. CIFRADO EGIPCIO

Las primeras manifestaciones del cifrado, tuvieron lugar en la cultura egipcia, el siguiente es considerado como el primer ejemplo documentado de la escritura cifrada:

Cerca de 1900 a.C.

La historia acontece en una villa egipcia cerca del río Nilo llamada MenetKhufu. Khnumhotep II era un arquitecto del faraón Amenemhet II. Él construía algunos monumentos para el faraón, los cuales necesitaban ser documentados. No es preciso decir que estas informaciones escritas en pastillas de arcilla, no eran para caer en el dominio público. El escriba de Khnumhotep II tuvo la idea de sustituir algunas palabras o tramos de texto de estas pastillas. Si el documento fuera robado, el ladrón no encontraría el camino que lo llevaría al tesoro moriría de hambre, perdido en las catacumbas de la pirámide.

TIPOS DE ESCRITURA

Cuestiones militares, religiosas y comerciales impulsaron desde tiempos remotos el uso de escrituras secretas. Ya los antiguos egipcios usaron métodos criptográficos. Por ejemplo, los sacerdotes egipcios utilizaron la escritura hierática (jeroglífica) que era claramente incomprensible para el resto de la población y que hacia parte de los tres métodos de escritura más conocidos desarrollados por los egipcios.

Jeroglífica o jeroglífica monumental

Usada en inscripciones de monumentos y decoración. Es el tipo de escritura más antiguo y más complejo. Se empleó desde el 3100 a.C., fecha aproximada del primer papiro conocido. Era un tipo de escritura sagrada, llamada "escritura de la palabra de dios", y como tal se empleaba en sarcófagos, tumbas, monumentos y esculturas, y se representaba con gran detalle. La palabra jeroglífico deriva del griego "tahieroglyphica" que significa "las letras talladas en piedra" y se debe a la asociación de los jeroglíficos con las inscripciones monumentales. A pesar de que algunas personas, además de los escribas, la sabían leer y escribir, la escritura

jeroglífica era la más desconocida y estaba reservada a muy pocos. Se escribía en cualquier sentido (excepto de abajo a arriba) y en líneas o columnas.

Hierática

Surgió como escritura abreviada de la jeroglífica cursiva. Lógicamente el sistema jeroglífico no era apropiado para escrituras rápidas y esto motivó el nacimiento de la escritura hierática, mucho más fluida y estilizada. Era más sencilla que la jeroglífica por lo que se podía emplear en textos religiosos y es la más utilizada sobre papiro. El término proviene del griego "hieratika", que significa sacerdotal. La escritura hierática puede siempre transcribirse en jeroglíficos, si bien el resultado no es el mismo que cuando se compone un texto originariamente en jeroglífica. La jeroglífica cursiva desapareció en torno al año 1000 a.C. mientras que la hierática se utilizó en textos religiosos hasta fines de la civilización egipcia. También se empleaba en textos científicos y obras literarias. Era un tipo de escritura muy útil en papiros y ostracas. El texto se escribía en tinta negra con una caña afilada. La tinta roja se empleaba como remarcación de determinados apartados. A pesar de que la hierática temprana se escribía, como la jeroglífica, en líneas o columnas indistintamente, a partir de la XII dinastía, los textos aparecen sólo en líneas y siempre de derecha a izquierda.

Demótica

El término demótico proviene del griego "demotika" ("popular") y se refiere a los asuntos diarios. El término fue utilizado por primera vez por Herodoto. Representa una evolución de la lengua hablada, y es un estado intermedio entre el egipcio y el posterior copto. Es una forma abreviada de la hierática, de trazo rápido y sencillo, con mayores ligaduras en los signos y esquematizando los grupos con enlaces. En esta escritura, es ya difícil reconocer los signos jeroglíficos originales. Se empleaba en asuntos cotidianos, transacciones comerciales, e incluso en algunas inscripciones en piedra, como la piedra de Roseta, donde se la denomina "escritura de los libros”. Su uso comenzó aproximadamente en torno a la XXVI dinastía y se mantuvo hasta el siglo III d.C. Desde la época ptolemaica se empleó también en textos científicos y religiosos, convirtiéndose en la escritura oficial. Se empleó en las regiones del norte desde el año 700 a.C y en todo el país desde el 600 a.C. Se escribía en líneas de derecha a izquierda. El demótico se corresponde aproximadamente a la lengua hablada de los siglos VII y VI a.C. Realmente,

aunque el demótico se sirvió del egipcio Nuevo y estaba basado en el hierático, los jeroglíficos son irreconocibles.

2.3. CIFRADO MESOPOTÁMICO

Época 1500 a.C.

Figura 1. Pastilla Mesopotámica

Fuente: Historia de la criptología, disponible en: *http://serdis.dis.ulpgc.es/~ii-cript/PAGINA%20WEB%20CLASICA/CRIPTPLOGIA/HISTORIA%20DE%20CRIPTOLOGIA%20-%20ANTIGUA.htm* [online]

La criptografía de la Mesopotamia ultrapasó la egipcia, llegando a un nivel bastante moderno. El primer registro del uso de la criptografía en esta región está en una fórmula para hacer esmaltes para cerámica. La pastilla que contiene la fórmula tiene sólo cerca de 8 cm x 5 cm y fue hallado a los márgenes del río Tigre. Usaba símbolos especiales que pueden tener varios significados diferentes, según Kahn.

En esta época, mercaderes asirios usaban "intaglios", que son piezas planas de piedra con símbolos entallados para su identificación. El moderno comercio con firmas digitales estaba inventado.

Esta también fue la época en que culturas como la de Egipto, China, India y de la Mesopotamia desarrollaron la esteganografía:

- Tatuajes con mensajes en la cabeza de esclavos. Infelizmente era preciso esperar el cabello crecer. El descifrado era hecho en el barbero.

- Lacras en la madera de placas de cera. Las lacras eran escondidas con cera nueva. Para descifrar, bastaba derretir la cera.
- Mensajes dentro del estómago de animales de caza... y también de humanos.

2.4. CIFRADO DE LOS ESCRIBAS HEBREOS

En la Criptografía (de "*kriptos"*, oculto) no se trata de esconder el mensaje, sino de imposibilitar su interpretación a quien no conozca el código para su descifrado.

Escribas hebreos, escribiendo el libro de Jeremías, usaron la cifra de sustitución simple por el alfabeto reverso conocida como ATBASH. Las cifras más conocidas de la época junto con este son, el ALBAM y el ATBAH, las llamadas cifras hebraicas.

En la polémica y famosa novela "El Código Da Vinci" de Dan Brown, se nos presenta uno de los códigos más antiguos, utilizado por los hebreos en sus escrituras, el código del Atbash, en el que cada letra de su alef-beit (alfabeto), es sustituida por la que ocupa el mismo lugar pero en sentido inverso; un esquema de este código con caracteres latinos, es el siguiente:

A= primera letra del abecedario.

Z=ultima letra del abecedario.

Por lo tanto **A=Z.**

Tabla 2. Igualdades de cifrado escribas hebreos

A=Z	B=Y	C=X	D=W	E=V	F=U	G=T	H=S	I=R	J=Q	K=P	L=O	M=N
N=M	O=L	P=K	Q=J	R=I	S=H	T=G	U=F	V=E	W=D	X=C	Y=B	Z=A

Por último para cifrar, se hace corresponder la letra superior con su correspondiente inferior, siendo esta última la que figura en el texto cifrado.

Vease un ejemplo:

El mensaje **FIRMA LA PAZ**

Cuadro 1. Ejemplo 01 cifrado escribas hebreos

F	I	R	M	A
U	R	I	N	Z

Y así sucesivamente se convierte en **URINZ AZ KZA.**

Estos sistemas que utilizan los mismos símbolos intercambiándolos entre sí, utilizan técnicas llamadas de trasposición; cuando se sustituyen los símbolos o letras por otros se dice que se utilizan técnicas de sustitución. Estos dos conjuntos de técnicas conforman la base de la criptografía clásica.

Época 400 a.C.

Textos griegos antiguos, de Enéas el Táctico, Políbio y otros describen varios métodos de ocultar mensajes, pero ningún de ellos parece haber sido efectivamente utilizado (Glikman). Aun así es interesante conocer los relojes d'agua, un método curioso y muy singular.

El nombre real de Enéas el Táctico (AeneasTacticus) supuestamente sería Enéas de Stymphalus. Él fue un científico militar y criptógrafo griego. Inventó un sistema ótico de comunicación semejante al telégrafo: los relojes de agua, los cuales son sistemas de comunicación a la distancia. Cada integrante del sistema poseía jarros exactamente iguales, contiendo la misma cantidad de agua. Cada jarrón poseía un agujero del mismo diámetro en todos los jarrones, el cual permanecía sellado. Dentro del jarro había un bastión con diversos mensajes inscritos.

Cuando uno de los integrantes quería hacer contacto con otro, hacía una señal con fuego y humo. Cuando el otro respondía, ambos abrían simultáneamente el agujero del jarro. Con la ayuda de una segunda señal con fuego, ambos cerraban el orificio simultáneamente. De esta forma, la superficie del agua en el jarro apuntaba para el mensaje deseado.

2.5. LA ESCITALA ESPARTANA (475 a.C)

El primer caso claro de uso de métodos criptográficos se dio durante la guerra entre Atenas y Esparta. El historiador griego Plutarco, describe *la escítala* de la

siguiente manera: “La escitala era un palo o bastón en el cual se enrollaba en espiral una tira de cuero.

Sobre esa tira se escribía el mensaje en columnas paralelas al eje del palo. La tira desenrollada mostraba un texto sin relación aparente con el texto inicial, pero que podía leerse volviendo a enrollar la tira sobre un palo del mismo diámetro que el primero”.

Con este sistema los gobernantes de Esparta transmitieron, con eficacia, sus instrucciones secretas a los generales de su ejército, durante las campañas militares.

Lógicamente, este procedimiento suponía que tanto el emisor como el receptor del mensaje dispusieran de un palo o bastón con las mismas características físicas: grosor y longitud.

Figura 2. Escitala espartana

Fuente: http://farm4.static.flickr.com/3647/3509160604_50ae17ea3b_b.jpg [online]

2.6. EL CIFRARIO DE JULIO CÉSAR (50 a.C)

Este método fue empleado en los tiempos de la Roma Imperial. El algoritmo de César, llamado así porque es el procedimiento que empleaba **Julio César** para enviar mensajes secretos a sus legiones, es uno de los algoritmos criptográficos más simples. Es un algoritmo de **sustitución**, su cifrado consistía simplemente en sustituir una letra por la situada tres lugares más allá en el

alfabeto esto es la A se transformaba en D, la B en E y así sucesivamente hasta que la Z se convertía en C.

Tabla 3. Equivalencia alfabeto original – Cifrado Julio Cesar

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Alfabeto original: Arriba

Alfabeto cifrado: Abajo

Por ejemplo: El mensaje **FIRMA LA PAZ**

Ejemplo 01 Cifrado Julio Cesar

F I R M A

I L U P D

Y así sucesivamente se convierte en **ILUPD OD SDC.**

Nota: **"**Hoy en día, cualquier alfabeto que esté codificado con el alfabeto desplazado pero en su orden se llama "cifrado de César", aun cuando la letra inicial sea diferente de la D"1.

2.6.1. Tratamiento matemático:

Si se asigna a cada letra un número (A =00, B =01, C=02,.....Z=25), y consideramos un alfabeto de 26 letras, la transformación criptográfica en términos matemáticos se puede explicar bajo la siguiente fórmula de congruencias:

Cº (M + 3) (mód 26) M, corresponde a la letra del mensaje original C, es la letra correspondiente a M pero en el mensaje cifrado.

Obsérvese que este algoritmo ni siquiera posee clave, puesto que la transformación siempre es la misma. Obviamente, para descifrar basta con restar 3 al número de orden de las letras del criptograma.

Ejemplo:

Asumiendo un alfabeto de 26 símbolos como el siguiente:

Tabla 4. Numeración letras del alfabeto

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Se va a cifrar el siguiente mensaje: **PAZ**

Se puede hacer manualmente o utilizando la fórmula anteriormente dada:

1. Reemplazar **M** por el valor de la primera letra, en este caso P equivale a 15.
2. Realizar la operación indicada: **C = (15 + 3) (mód 26) = 18.**
3. Corresponder el número obtenido con la letra, en nuestro caso la S.
4. Realizar la operación con las letras restantes.

Así se obtiene las siguientes correspondencias:

Cuadro 2. Ejemplo 01 Cifrado Julio Cesar

M	C
P	S
A	D
Z	C

Por tanto el Mensaje Codificado es: **SDC**.

2.7. EL MÉTODO DE POLYBIOS

Polybio fue un historiador griego nacido en el año 200 a.C. en el año 168 a.C. fue llevado como rehén a Roma. Allí escribió una historia de Roma. No obstante, no se dedicó exclusivamente a la historia. Tuvo tiempo de inventar un código

conocido como “Tablero de Políbio”, que fue un sistema que acabó siendo adoptado muy a menudo como método criptográfico.

Colocó las letras del alfabeto en una red cuadrada de 5x5. El sistema de cifrado consistía en hacer corresponder a cada letra del alfabeto un par de letras que indicaban la fila y la columna, en la cual aquella se encontraba.

Tabla 5. Tabla de Polybio con letras

	A	B	C	D	E
A	A	B	C	D	E
B	F	G	H	I,J	K
C	L	M	N,Ñ	O	P
D	Q	R	S	T	U
E	V	W	X	Y	Z

Cada letra tiene una coordenada en letras por así decirlo:

La **M** seria **CB,** la **O** seria **CD y** así sucesivamente con todo el alfabeto.

Por ejemplo el texto: **DESEAMOS LA PAZ**

Cuadro 3. Ejemplo 01 cifrado de polybio

D	**E**	**S**	**E**	**A**	**M**	**O**	**S**
AD	AE	DC	AE	AA	CB	CD	DC

Y así sucesivamente se convierte en: **ADAEDCAEAACBCDDC CAAA CEAAEE**

Si en el tablero de Polybios se introduce números, resulta una variante sumamente interesante:

Tabla 6. Tabla de Polybio con números

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I,J	K
3	L	M	N,Ñ	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Cada letra viene representada por dos números, el de su fila y el de su columna.

Así, K = 25, w = 52, mientras que la letras N y Ñ tienen una misma representación.

Por ejemplo el texto: **DESEAMOS LA PAZ**

Cuadro 4. Ejemplo 02 cifrado de Polybio.

D	**E**	**S**	**E**	**A**	**M**	**O**	**S**
14	15	43	15	11	32	34	43

Y así sucesivamente se convierte en: **1415431511323443 3111 351155**

Polybios sugería usar este sistema como método de transmisión de mensajes a larga distancia (usando antorchas o fogatas, por ejemplo). Pero, sin duda, el gran éxito de su método reside en la conversión de letras en números, la reducción en el número de caracteres finales, y la división de una unidad en dos partes manipulables separadamente. Lo que ha servido de base para otros sistemas de cifrado, es el caso del sistema Playfair.

2.8. LA CIFRA DEL KAMA-SUTRA

Época 400 d.C

Una de las descripciones más antiguas de encriptación por sustitución está en el Kama-Sutra, un texto escrito el siglo IV d.C. por el sabio hindú Vatsyayana, sin embargo basado en manuscritos datados de más de 800 años. El Kama-Sutra recomienda que las mujeres estudien 64 artes, incluyendo la culinaria, la forma de vestir, masaje y la preparación de perfumes. La lista también incluye algunos artes

menos obvios como prestidigitación, ajedrez, encuadernación de libros y carpintería. En la lista, la de número 45 es la mlecchita-vikalpa, el arte de la escritura secreta, indicada para ayudar las mujeres a esconder los detalles de sus relaciones.

La técnica recomendada es la de formar pares aleatorios con las letras del alfabeto, (cada quien puede formar los pares de la manera y orden que desee) y después sustituir cada letra del texto original por su correspondiente pareja:

Cuadro 5. Ejemplo 01 Kamasutra

A	D	H	I	K	M	O	R	S	U	W	Y	Z
V	X	B	G	J	C	Q	L	N	E	F	P	T

En este caso la **A** se convertiría en **V**, la **B** en **H**, la **C** en **M**, la **D** en **X** y así sucesivamente.

Por ejemplo para cifrar "**ENCONTRÉMONOS A MEDIA NOCHE**" con la anterior combinación de pares,

Cuadro 6. Ejemplo 02 Kamasutra

E	**N**	**C**	**O**	**N**	**T**	**R**	**E**	**M**	**O**	**N**	**O**	**S**
U	S	M	Q	S	Z	L	U	C	Q	S	Q	N

Y así sucesivamente se obtiene: **USMQSZLUCQSQN V CUXGVSQMBU.**

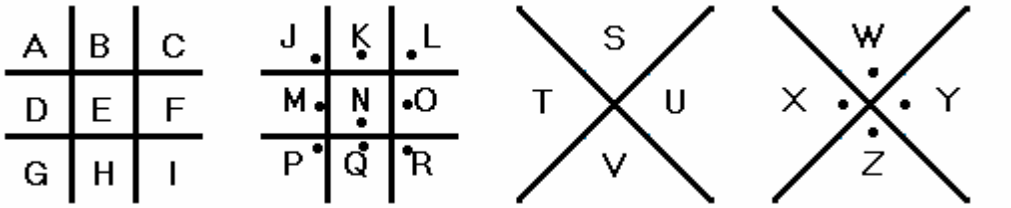
2.9. LA CIFRA PIGPEN

Este sistema de cifrado fue utilizado por los masones en el siglo XVIII, para preservar la privacidad de sus archivos.

En 1533, Heinrich Cornelius Agrippa von Nettelsheim publica el *De occultaphilosophia*, en Colonia, en Alemania. En el libro 3, capítulo 30, describe su cifra de sustitución mono alfabética, hoy conocida como Cifra Pig Pen. La traducción literal del nombre es Cerdo en el Chiquero y viene del hecho de que cada una de las letras (los cerdos) es colocada en una "casa" (el chiquero).

Las letras del alfabeto son dispuestas en dos grupos de 9 y dos grupos de 4 letras, separadas por líneas, como es mostrado abajo.

Figura 3. Cifra PIG PEN

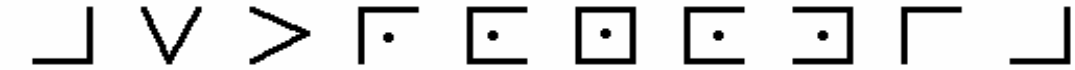


Fuente: http://upload.wikimedia.org/wikipedia/commons/thumb/3/36/Pigpen_cipher_key.svg/230px-Pigpen_cipher_key.svg.png [online]

De acuerdo con el diagrama, cada letra del alfabeto es sustituida por las líneas que la envuelven.

Por ejemplo, ASTRONOMIA se cifraría como:

Cuadro 7. Ejemplo 01 cifra PIG PEN

A	S	T	R	O	N	O	M	I	A
									

2.10. LA CIFRA PLAYFAIR-WHEATSTONE

2.10.1. Introducción

El primer registró la descripción de la cifra de Playfair estaba en un documento firmado por Wheatstone encendido 26 de marzo1854. Sin embargo, el esquema vino eventual ser sabido por el nombre del amigo de Wheatstone Señor Playfair, que lo popularizó. Fue rechazado por la oficina extranjera británica cuando fue desarrollado debido a su complejidad percibida. Cuando Wheatstone ofreció demostrar que tres fuera de cuatro muchachos en una escuela próxima podrían

aprender utilizarla en 15 minutos, la secretaria inferior de la oficina extranjera respondió, “que es muy posible, pero usted podría nunca enseñarla a los agregados.”

Fue utilizado para los propósitos tácticos cerca Británico fuerzas en Segunda guerra de Boer y adentro Primera Guerra Mundial y para el mismo propósito por Australianos y Alemanes durante Segunda Guerra Mundial. Esto era porque Playfair es razonablemente rápido utilizar y no requiere ningún equipo especial. Un panorama típico para el uso de Playfair sería proteger secretos importantes pero no críticos durante combate real. Para el momento en que los criptoanalistas enemigos podrían romper el mensaje, la información les era inútil.

Playfair es utilizado no más por las fuerzas militares debido al advenimiento de los dispositivos digitales del cifrado. Playfair ahora se mira como inseguro para cualquier propósito porque las computadoras hand-held modernas podrían romper fácilmente la cifra dentro de segundos.

La primera solución publicada de la cifra de Playfair fue descrita en un folleto de 19 páginas por el teniente José O. Mauborgne, publicado adentro 1914.

2.10.2. Uso de Playfair

Este sistema de cifrado sustituye cada par de letras del texto llano por otro par de letras en el texto cifrado. Para codificar y transmitir un mensaje, el emisor y el receptor deben acordar una palabra clave. Para este ejemplo vamos a utilizar la palabra **CHARLES**.

A continuación, se construye una matriz cuadrada de tamaño 5x5, comenzando en la primera línea, de arriba hacia abajo, con las letras de la palabra clave, (Si la palabra clave, posee letras repetidas, estas se colocan una sola vez; por ejemplo, si la palabra fuese “**PLAYFAIR**”, en la tabla ubicamos “**P L A Y F I R**”). Los espacios restantes se llenan con las letras del abecedario en orden, únicamente omitiendo aquellas letras que ya se han colocado por estar en la palabra clave.

Cuadro 8. Ejemplo 01 cifra playfair

C	**H**	**A**	**R**	**L**
E	**S**	B	D	F
G	I/J	K	M	N
O	P	Q	T	U
V	W	X	Y	Z

Alfabeto utilizado: A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z.

Obsérvese que no se coloca la **Ñ**, por el origen anglosajón de este sistema de cifrado, así como la I y la J se colocan juntas; en castellano, se tendría que colocar juntas **Ñ** y **W,** por ser ésta una letra poco habitual en el castellano.

A continuación se divide el mensaje en pares de letras (dígrafos). Las dos letras de cada dígrafo han de ser diferentes, lo que se consigue insertando una x adicional entre dos letras iguales y añadiendo una x al final si la última letra del texto llano queda desemparejada.

Ejemplo de ello lo tenemos en el siguiente mensaje en inglés, por su origen es anglosajón, (Esto no evita cifrar texto en otros idiomas):

Texto llano:

MEET ME AT HAMMERSMITH BRIGDE TONIGHT

Texto dividido en dígrafos inicialmente:

ME-ET-ME-AT-HA-**MM**-ER-SM-IT-HB-RI-GD-ET-ON-IG-HT

Nótese que desde el dígrafo “mm” ya se incumple una de las reglas para formar los dígrafos en este método de cifrado, en donde las dos letras de cada dígrafo han de ser diferentes: esto lo corregimos insertando la x adicional, “mx” y de ahí en adelante se reorganizan los dígrafos y así sucesivamente si vuelve a ocurrir en los nuevos órdenes.

Texto llano en dígrafos:

ME-ET-ME-AT-HA-MX-ME-RS-MI-TH-BR-ID-GE-TO-NI-GH-T**X**

Se puede observar que al final hay una “x” también, esto es por la segunda regla para formar los dígrafos en donde se añade una x al final si la última letra del texto llano queda desemparejada.

A continuación puede empezar el cifrado:

Todos los dígrafos resultantes son de una de las siguientes tres clases al buscar sus letras en la tabla ya definida con la palabra clave:

i. Las dos letras están en la misma fila.
ii. Las dos letras están en la misma columna.
iii. Las dos letras están en filas y columnas diferentes.

- ✓ Si el dígrafo es de clase (**i**), cada letra del dígrafo es sustituida por la que queda a la derecha de cada una de ellas; la última letra de cada fila será sustituida por la primera.

  Dígrafo: HA.

  Fila de la tabla:

C	H	**A**	**R**	L

  Cifrado: Sustituyo la H por la A y la A por la R.

  Dígrafo resultante: AR.

- ✓ Si el dígrafo es de clase (**ii**), cada letra del dígrafo es sustituida por la que queda debajo de cada una de ellas; la última letra de cada columna será sustituida por la primera

  Dígrafo: CO.

  Columna de la tabla:

C
E
G
O
V

  Cifrado: Sustituyo la C por la E y la O por la V.

  Dígrafo resultante: EV.

✓ Si el dígrafo es de clase (**iii**), la regla de sustitución es la siguiente:

- Observe en la fila de la primera letra del dígrafo, a qué columna corresponde la segunda letra del dígrafo. La letra que en la fila está en la misma columna sustituirá a la primera.

  Dígrafo: ME

Cuadro 9. Ejemplo 02 cifra playfair

C	H	A	R	L
E	S	B	D	F
G	I/J	K	**M**	N
O	P	Q	T	U
V	W	X	Y	Z

Cifrado: Se sustituye la M por la G.

- Para la segunda letra procederemos de manera similar. Observaremos en la fila de la segunda letra, hasta localizar la columna de la primera. La letra situada en esa misma columna sustituirá a la primera.

Dígrafo: ME.

Cuadro 10. Ejemplo 03 cifra playfair

C	H	A	R	L
E	S	B	D	F
G	I/J	K	**M**	N
O	P	Q	T	U
V	W	X	Y	Z

Cifrado: Se sustituye la E por la D

Dígrafo resultante: GD.

Ahora se cifra el mensaje completo:

Cuadro 11. Ejemplo 04 cifra playfair

Text	ME	ET	ME	AT	HA	MX	ME	RS	MI	TH	BR	ID	GE	TO	NI	GH	TX
	GD	DO	GD	RQ	AR	KY	GD	HD	NK	PR	DA	MS	OG	UP	GK	IC	QY

o cifrado:

GDDOGDRQARKYGDHDNKPRDAMSOGUPGKICQY

El receptor, que conoce la palabra clave, puede descifrar el texto invirtiendo el proceso con cada dígrafo del texto cifrado. En el caso (i) sustituirá cada letra por la situada a su izquierda y en el caso (ii) por la situada encima. En el caso (iii) hará el intercambio mirando la fila del dígrafo cifrado; así, PR se descifra como TH.

Este sistema de cifrado no es inexpugnable, pues buscando los dígrafos más frecuentes en el mensaje cifrado, se puede suponer que corresponden a los dígrafos más frecuentes en el inglés: TH, HE, AN, IN, ER, RE, ES. En cualquier idioma, es descifrable mediante el análisis estadístico.

2.11. CIFRADO DE VIGENÉRE

2.11.1. Introducción

El francés Blaise de Vigenérefue un diplomático francés nacido en 1523, en el siglo XVI, desarrolló la teoría de la criptología polialfabética, por esta razón su nombre ha acabado asociado con uno de los métodos famosos de sustitución polialfabética. Lo que hoy se denomina “tablero de Vigenére”.

Consiste en una disposición de letras que contiene en orden los 26 alfabetos de César, es decir, se utilizan no uno, sino 26 alfabetos cifrados, cada uno de ellos comenzando en la letra siguiente del anterior. La naturaleza polialfabética es lo que le da su fuerza, pero también hace que sea mucho más complicada de usar. Además, para proteger más el cifrado suele introducirse una palabra **clave**, que consiste en una palabra o texto que se repite a largo de todo el mensaje a cifrar, como se verá más adelante.

La cifra resulta invalida para el análisis de frecuencia, pues una misma letra que aparezca varias veces en el texto cifrado puede representar en cada ocasión una letra diferente del texto llano y a su vez una letra que aparezca varias veces en el texto llano puede estar representada por diferentes letras en el texto cifrado, particularidad que hizo tan fuerte en su época este método.

2.11.2. Uso del tablero de Vigenére

Tabla 7. Tablero de Vigenere

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

i. Para cifrar se procede de la siguiente manera:
ii. Se busca una palabra clave fácil de recordar.
iii. Se escribe la palabra debajo del texto en claro, repitiéndose tantas veces como sea necesario.

iv. Cada letra del texto en claro se codifica con el alfabeto de la tabla marcado por la letra inferior, o sea, la letra de la clave que corresponde.

Ejemplo:

Clave = **AZUL**

Texto: **EL EJÉRCITO ESTÁ PREPARADO**

Proceso: Se escribe le clave debajo del texto a cifrar.

ELEJERCITOESTAPREPARADO

AZULAZULAZULAZULAZULAZU

Cada letra de la parte superior se relacionara en el tablero con la de la parte inferior, donde la superior se ubica en la primera fila y la inferior en la primera columna de la siguiente manera:

Por ejemplo, para el caso de la segunda E, la cual se cifraría como la Y, o la primera R, la cual se cifraría como la Q, o la I, la cual se cifraría como la T, o la primera T, la cual se cifraría como la T igualmente por estar en la misma fila que la letra de abajo que en este caso es la A y así sucesivamente, muy similar al método de ubicar coordenadas en un plano cartesiano.

Cuadro 12. Ejemplo 01 cifrado Vigenére

A	B	C	D	**E**	F	G	H	**I**	J	K	L	M	N	O	P	Q	**R**	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Por último, cada una de las letras del mensaje se transforma en otra.

El mensaje cifrado es:

Cuadro 13. Ejemplo 02 cifrado Vigenére

E	L	E	J	E	R	C	I	T	O	E	S	T	A	P	R	E	P	A	R	A	D	O
E	**K**	**Y**	**U**	**E**	**Q**	**W**	**T**	**T**	**N**	**Y**	**D**	**T**	**Z**	**J**	**C**	**E**	**O**	**U**	**C**	**A**	**C**	**I**

Es decir: **EK YUEQWTTN YDTZ JCEOUCACI**

El receptor, que conoce la palabra clave, puede descifrar el texto, colocando de nuevo la palabra clave debajo del texto cifrado, pero en este caso lo único que se hace es ubicar la letra de encima sobre la línea correspondiente a la letra de

abajo, la cual ubicamos en la primera columna, al obtener esto, observamos en que columna quedo, respecto a las letras de la primera fila.

Cuadro 14. Ejemplo 03 cifrado Vigenére

E	**K**	**Y**	**U**	**E**	**Q**	**W**	**T**	**T**	**N**	**Y**	**D**	**T**	**Z**	**J**	**C**	**E**	**O**	**U**	**C**	**A**	**C**	**I**
A	Z	U	L	A	Z	U	L	A	Z	U	L	A	Z	U	L	A	Z	U	L	A	Z	U
E	L	E	J	E	R	C	I	T	O	E	S	T	A	P	R	E	P	A	R	A	D	O

Así se tiene por ejemplo en el caso de la K, que está en la fila de la Z, en la columna de la L, por tal la letra original es esta L, o en el caso de la W, que está en la fila de la U, en la columna de la C, por tal la letra original es la C, o en el caso de la primera C, que está en la fila de la L, en la columna de la R, por tal esta es la letra original y así sucesivamente hasta obtener el mensaje completo.

Cuadro 15. Ejemplo 04 cifrado Vigenére

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	**C**	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	**W**	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	**K**	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

A pesar de que el cifrado es mucho más sólido que el cifrado César, aun así se puede romper fácilmente. Cuando los mensajes son mucho más largos que la palabra clave, es posible identificar el largo de la palabra clave y utilizar, para cada secuencia de palabra clave, el método de cálculo de la frecuencia con que aparecen las letras, y determinar así los caracteres de las palabras claves una a la vez.

Para evitar este problema, una solución es utilizar una palabra clave que sea casi igual de larga como el texto, a fin de evitar un estudio estadístico del texto cifrado. El problema con este tipo de método es la longitud de la clave de cifrado (cuanto más largo el texto a ser cifrado, más grande deberá ser la clave) que impide su memorización e implica una probabilidad mucho más grande de errores en la clave (un solo error hace que el texto sea imposible de leer).

Las investigaciones de Blaise de Vigenére, así como los métodos utilizados en su época están recogidos en su libro *Traicté des Chiffres*, publicado en 1586. Curiosamente un sistema tan avanzado fue ignorado durante casi dos siglos.

Seguramente las razones para no utilizar la cifra de Vigenére son varias: el uso extendido, por parte de los criptógrafos, de las cifras monoalfabéticas, añadiendo homófonos y sobretodo la dificultad de utilizar las cifras polialfabéticas.

3. HISTORIA MODERNA

3.1. INTRODUCCIÓN

A lo largo de la historia militar, siempre ha sido de gran importancia el conseguir una buena red de comunicaciones entre las fuerzas armadas, sobre todo cuando se comenzó a realizar ataques organizados. Pero siempre ha existido un gran riesgo; si el enemigo llega a interceptar la comunicación, tendrá una información de primera mano sobre las intenciones del oponente, que puede darle una ventaja enorme. La solución está en transformar el mensaje en un uno indescifrable para todo el mundo excepto para el destinatario.

A lo largo de la historia, se pueden distinguir tres formas de codificación de la información:

- Métodos manuales

- Métodos mecánicos

- Métodos electrónicos

Los métodos manuales fueron los primeros utilizados, y se basan en la necesidad de una persona que se encargue de codificar/decodificar el mensaje. Como ejemplo, uno de los primero métodos de cifrado usados, el sistema denominado *"Cifrado del César" (Descrito en el capitulo anterior).* Este es un sistema muy simple, pero que supone un primer paso hacia codificaciones más complejas. Este tipo de codificación, consistente en la sustitución de una letra por otra, se denomina código de sustitución. Pierde utilidad a medida que crece la longitud del mensaje, dado que se puede descifrar por análisis estadístico de la presencia de cada símbolo, a partir del conocimiento de la presencia de una letra concreta en un idioma (Es preferible conocer el idioma en que está escrito el texto, pero no imprescindible), este método se explica en el capitulo siguiente.

Desde el siglo XIX y hasta la Segunda Guerra Mundial una de las figuras más importantes fue la del holandés Auguste Kerckhoffs con sus reglas, las cuales marcaron la pauta de un sistema de cifrado seguro. Pero es en el siglo XX cuando la historia de la criptografía vuelve a presentar importantes avances. Estos códigos pasados fueron evolucionando en complejidad a lo largo de la historia, sin

embargo el número de operaciones necesarias para la codificación/decodificación de un mensaje, hacían imposible pasar de un cierto nivel de dificultad.

En especial durante las dos contiendas bélicas que marcaron al siglo: la Gran Guerra y la Segunda Guerra Mundial, la criptografía usa una nueva herramienta que permitirá conseguir mejores y más seguras cifras: las máquinas de cálculo.

La aparición de las calculadoras mecánicas en el siglo XIX proporcionó una nueva forma para implementar los códigos mucho más rápidos y potentes. Estas máquinas realizaban operaciones matemáticas por métodos mecánicos de forma automática, eliminando la necesidad de una persona que conozca el código.

La máquina Enigma fue el máximo exponente de este tipo de dispositivos: una máquina de rotores que automatizaba considerablemente los cálculos que era necesario realizar para las operaciones de cifrado y descifrado de mensajes. Para vencer al ingenio alemán, fue necesario el concurso de los mejores matemáticos de la época y un gran esfuerzo computacional. No en vano, los mayores avances tanto en el campo de la criptografía como en el del cripto-análisis no empezaron hasta entonces, pues después de la guerra aparecieron los primeros sistemas digitales electrónicos a base de válvulas de vacío en un principio y de transistores después, los que asumieron las tareas de criptografía hasta hoy en día.

3.2. REGLAS DE KERCHOFFS

Kerchoffs escribió en el siglo XIX la criptografía militar, en la que estableció unas normas que todo criptosistema debía cumplir con el objetivo de evitar ser violado en un ataque criptoanalista.

Estas normas siguen siendo importantes hoy en día e inicialmente fueron las siguientes:

- ✓ No debe existir ninguna forma de recuperar mediante el criptograma el texto inicial o la clave.
- ✓ Todo sistema criptográfico ha de estar compuesto por dos tipos de información:
    - Pública: como es la serie de algoritmos que lo definen.
    - Privada: como es la clave. En los sistemas asimétricos parte de la clave es también información pública.
- ✓ La clave escogida debe ser fácil de recordar y modificar.

- ✓ El criptograma ha de poder ser transmitido usando los medios de comunicación habituales.
- ✓ La complejidad del proceso de recuperación del texto original debe corresponderse con el beneficio obtenido.

Posteriormente se conocieron otras reglas en cuanto al uso y manejo del criptosistema como tal, de entre las cuales se destaca este grupo traducidas del francés:

- ✓ El sistema debe ser matemáticamente, indescifrable.
- ✓ El mensaje cifrado no debe ser secreto, debe poder caer en las manos del enemigo sin inconveniente alguno.
- ✓ Su clave debe ser comunicable y reconfigurable sin la ayuda de notas escritas y de acuerdo a la voluntad de los correspondientes.
- ✓ Debe ser aplicable a la correspondencia telegráfica.
- ✓ Debe ser portable y su uso y función no deben requerir de la ayuda de varias personas;
- ✓ Finalmente, es necesario, dado las circunstancias que ordenan su uso, que el sistema sea fácil de utilizar, sin requerir de tensión mental ni el conocimiento de una serie larga de reglas para su correcto uso.

3.2.1. Tipos de ataque

Kerchoffs definió sus reglas basándose en los distintos tipos de ataque que un criptosistema puede sufrir. Los distintos ataques se clasifican en función de la información conocida por el criptoanalista.

1) Ataque sólo con texto cifrado:

En este caso el criptoanalista sólo posee el criptograma, sin conocer nada sobre su contenido, aunque en la práctica es muy fácil deducir parte del contenido del mensaje, como puede ser la cabecera del documento, etc. Los modernos criptosistemas son totalmente efectivos contra este tipo de ataque, pero a pesar de todo el criptoanalista puede obtener una cantidad de datos estadísticos considerable, ya que generalmente dichas partes siguen una estructura

predefinida y siempre idéntica, y es por ello que en el diseño de los modernos criptosistemas este ataque se sigue teniendo en cuenta.

2) **Ataque con texto original conocido**

En este caso el criptoanalista conoce, o puede predecir, el texto correspondiente a ciertas partes del criptograma. A partir de esto el criptoanalista ha de conseguir descifrar el resto del criptograma, bien sea deduciendo la clave empleada a la hora de cifrar o aprovechando las palabras de las que se tiene una correspondencia. Este ataque es especialmente efectivo contra sistemas de cifrado simétrico por bloques, ya que en éstos cada palabra conserva su longitud inicial en el criptograma.

3) **Ataque con texto original escogido**

En este caso el criptoanalista puede obtener el texto cifrado correspondiente a un texto claro escogido por él, entendiendo que no conoce la clave empleada en el proceso de cifrado. Este ataque dado que permite comparar la transformación de diversos textos planos es bastante efectivo como para sistemas simétricos, como para especialmente algunos asimétricos.

4) **Ataque con texto cifrado escogido**

En este caso el criptoanalista puede obtener el texto claro correspondiente a un criptograma de su elección. Al igual que en el caso anterior, se entiende que el criptoanalista tampoco conoce la clave empleada. Este es el caso menos común de todos.

Existen otros dos tipos de ataque, pero éstos no son dependientes de la información conocida por el enemigo, sino del modo de actuar de éste.

5) **Ataque con intermediario.**

Este ataque se basa simplemente en la intrusión del criptoanalista en el momento en el que las dos partes que quieren intercambiar la información están intercambiándose las claves. El criptoanalista ha de aprovechar este momento para hacer llegar a ambas partes su propia clave. Así éste recibe los mensajes primero, y luego los envía al destinatario legítimo, haciendo creer a ambas partes que el proceso de comunicación es seguro.

Este método pone de manifiesto uno de los grandes inconvenientes del sistema simétrico, el intercambio de claves, que ha de ser realizado a través de un medio

seguro, de lo contrario el enemigo puede hacerse fácilmente con éstas, pudiendo así cifrar y descifrar sin problema con absoluta transparencia tanto para emisor como para receptor.

6) Ataque de prueba y ensayo

Éste es el método más simple de todos. Consiste en probar cada una de las posibles claves, hasta dar con la que permite descifrar el criptograma. Éste método es lento si es una sola máquina la que realiza las pruebas, pero con la aparición de Internet, y con ello la posibilidad de poner a trabajar multitud de equipos en paralelo, este sistema ha conseguido que muchos de los sistemas se hayan dejado de emplear, por el temor a que una organización suficientemente poderosa pueda descifrar los criptogramas.

3.3. CIFRADO DE VERNAM

3.3.1. Introducción

Para evitar el inconveniente del sistema de Vigenére, en 1917 el ingeniero americano Vernam, propone un sistema de sustitución polialfabético de gran importancia en la criptografía, pues es el único que se demuestra matemáticamente perfecto. En él se puede abordar cuestiones probabilísticas, en concreto sobre la generación de números aleatorios.

Conocido el tamaño, “k”, del mensaje que se desea cifrar, se considera una sucesión finita “k” de variables aleatorias independientes e idénticamente distribuidas según una distribución equiprobable sobre Z_m, que será la clave usada de tamaño “K” también. Entonces se realiza una suma módulo “m” entre cada letra del mensaje con cada letra de la sucesión de claves.

Las comunicaciones militares requerían velocidad y simplicidad, ya que podían enviar y recibir cientos de mensajes cada día, por lo que los mandos militares se mostraban reticentes o negativos a adoptar la cifra polialfabética, a causa de su complejidad y buscaron formas intermedias, que fueran más difíciles de descifrar que las monoalfabéticas pero más sencillas que una cifra polialfabética.

Así surgió la cifra de sustitución homofónica. En ella cada letra es reemplazada por una variedad de sustitutos, y el número de sustitutos potenciales es

proporcional a la frecuencia de la letra. Por ejemplo, la letra "*a"* supone el 8% de todas las letras del inglés escrito, de manera que se asignaría ocho símbolos para representarla. Cada vez que apareciese una "*a"* en el texto llano sería reemplazada en el texto cifrado por uno de los ocho símbolos elegido al azar, de forma que al final de la codificación cada símbolo constituiría aproximadamente el 1% del texto codificado.

3.3.2. Cifrado

Primero que todo debe conocerse el equivalente en binario de las letras del alfabeto, supóngase un alfabeto de en binario de la A a la P, para representarlo en 4 dígitos de la siguiente manera:

Tabla 8. Tabla de equivalente binario a letras alfabeto

0	0	0	0	0	**A**
1	0	0	0	1	**B**
2	0	0	1	0	**C**
3	0	0	1	1	**D**
4	0	1	0	0	**E**
5	0	1	0	1	**F**
6	0	1	1	0	**G**
7	0	1	1	1	**H**
8	1	0	0	0	**I**
9	1	0	0	1	**J**
10	1	0	1	0	**K**
11	1	0	1	1	**L**
12	1	1	0	0	**M**
13	1	1	0	1	**N**
14	1	1	1	0	**O**
15	1	1	1	1	**P**

A partir de esta tabla pueden formarse distintas palabras como por ejemplo:

AMAR, AMOR, PAPA, MAMA, ENANO, PODIA…

O pequeñas frases como:

MI MAMA BEBE CAFÉ CON LECHE EN LA NOCHE, AHÍ NO HAY CAFÉ NI LECHE…

Para el cifrado debe de tomarse la palabra o frase a cifrar, por ejemplo ENANO y se saca su equivalente en binario:

Cuadro 16. Ejemplo 01 cifrado Vernam

E	**N**	**A**	**N**	**O**
0100	1101	0000	1101	1110

De igual manera se define una palabra clave de igual tamaño que la palabra a cifrar, para este caso la palabra NOCHE y de igual manera se obtiene su equivalente en binario.

Cuadro 17. Ejemplo 02 cifrado Vernam

N	**O**	**C**	**H**	**E**
1101	1110	0010	0111	0100

Teniendo ambas cifras se suma cada digito entre si siguiendo las siguientes reglas de la suma de binarios:

- 0+1 = 1
- 1+0 = 1
- 0+0 = 0
- 1+1 = 0

Al sumar se obtiene:

Cuadro 18. Ejemplo 03 cifrado Vernam

0	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**1**	**1**	**0**
1	**1**	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**1**	**1**	**0**	**1**	**0**	**0**
1	0	0	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	0

Al separarlos de a cuatro dígitos de nuevo y sacar su equivalente decimal se obtiene:

Cuadro 19. Ejemplo 04 cifrado Vernam

1001	**0011**	**0010**	**1010**	**1010**
9	3	2	10	10

Texto cifrado= 9321010

Esto es lo que hizo tan práctico y perfecto este método, por la facilidad de enviar números en vez de las palabras, además de que su descifrado no el complejo:

3.3.3. Descifrado

Para descifrar mediante este método, basta con realizar la misma operación del cifrado, solo que en vez del texto a transmitir se utiliza el texto ya cifrado:

Texto cifrado= 9321010

Se obtiene su equivalente en binario:

Cuadro 20. Ejemplo 05 cifrado Vernam

9	**3**	**2**	**10**	**10**
1001	0011	0010	1010	1010

Al igual que con la palabra clave de nuevo:

Cuadro 21. Ejemplo 06 cifrado Vernam

N	**O**	**C**	**H**	**E**
1101	1110	0010	0111	0100

A partir de esto se realiza la suma entre ambos, bajo las mismas reglas que para el cifrado:

Cuadro 22. Ejemplo 07 cifrado Vernam

1	**0**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**
1	**1**	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**1**	**1**	**0**	**1**	**0**	**0**
0	1	0	0	1	1	0	1	0	0	0	0	1	1	0	1	1	1	1	0

Nuevamente se divide en bloques de 4 dígitos y se obtiene su equivalente en letra:

Cuadro 23. Ejemplo 08 cifrado Vernam

0100	**1101**	**0000**	**1101**	**1110**
E	N	A	N	O

Y de esta manera se obtiene el mensaje original de nuevo.

3.4. CIFRADO DE BEAUFORT

3.4.1. Cifrado

En 1710, *Giovanni Sestri*, basado en el método de cifra de Vigenére, propone un algoritmo simétrico que sirve tanto para cifrar como para descifrar. El invento del cifrador en cuestión finalmente se le atribuye al inglés Sir *Francis Beaufort*, amigo de Sestri, y recibe precisamente el nombre de cifrador de Beaufort.

Como se va observar, la tabla es muy similar a la de Vigenére, donde la diferencia radica en que se invierte el orden de las letras del alfabeto y luego se les aplica un desplazamiento hacia la derecha de ($k_i + 1$) posiciones.

Para el cifrado se utiliza de igual manera que la de Vigenére.

Tabla 9. Tabla cifrado de Beaufort

		0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
		A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
A	0	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B
B	1	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C
C	2	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D
D	3	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E
E	4	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F
F	5	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G
G	6	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H
H	7	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I
I	8	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J
J	9	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K
K	10	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L
L	11	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M
M	12	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N
N	13	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O
O	14	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P
P	15	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q
Q	16	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R
R	17	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S
S	18	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T
T	19	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U
U	20	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V
V	21	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W
W	22	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X
X	23	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y
Y	24	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z
Z	25	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A

Ejemplo:

Usando la Tabla de Beaufort con clave “ULTIMÁTUM”, cifrar el mensaje “ESTO ES LA GUERRA CABALLEROS”.

K = ULTIMATUM

M = ESTO ES LA GUERRA CABALLEROS

Cuadro 24. Ejemplo 01 de Beaufort

E	**S**	**T**	**O**	**E**	**S**	**L**	**A**	**G**	**U**	**E**	**R**	**R**	**A**	**C**	**A**	**B**	**A**	**L**	**L**	**E**	**R**	**O**	**S**
U	L	T	I	M	A	T	U	M	U	L	T	I	M	A	T	U	M	U	L	T	I	M	A

Solución:

Siguiendo la tabla de Beaufort se obtiene:

Cuadro 25. Ejemplo 02 de Beaufort

E	**S**	**T**	**O**	**E**	**S**	**L**	**A**	**G**	**U**	**E**	**R**	**R**	**A**	**C**	**A**	**B**	**A**	**L**	**L**	**E**	**R**	**O**	**S**
Q	T	A	U	I	I	I	U	G	A	H	C	R	M	X	T	T	M	J	A	P	R	Y	I

C = QTAU II IU GAHCRM XTTMJAPRYI

3.4.2. Descifrado

Por la operación de sustitución empleada, es posible que, a diferencia del de Vigenére, un carácter se cifre con su valor en claro con una clave distinta de la letra A, como es el caso en el ejemplo anterior de la novena letra la (*G*), que se cifra en claro, con la clave (M), respectivamente. La operación de descifrado, es la misma que la de cifrado, con la excepción que en vez de texto claro “M” se cuenta con el texto cifrado “C”.

Cuadro 26. Ejemplo Descifrado 01 de Beaufort

		0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
		A	**B**	**C**	**D**	E	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	S	T	**U**	**V**	**W**	**X**	**Y**	**Z**
A	0	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B
B	1	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C
C	2	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D
D	3	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E
E	4	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F
F	5	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G
G	6	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H
H	7	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I
I	8	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J
J	9	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K
K	10	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L
L	11	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M
M	12	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N
N	13	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O
O	14	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P
P	15	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q
Q	16	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R
R	17	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S
S	18	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T
T	19	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U
U	20	U	T	S	R	**Q**	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V
V	21	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W
W	22	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X
X	23	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y
Y	24	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z
Z	25	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A

Y así sucesivamente para cada letra del mensaje cifrado hasta obtener el mensaje original:

Cuadro 27. Ejemplo descifrado 02 de Beaufort

Q	**T**	**A**	**U**	**I**	**I**	**I**	**U**	**G**	**A**	**H**	**C**	**R**	**M**	**X**	**T**	**T**	**M**	**J**	**A**	**P**	**R**	**Y**	**I**
U	L	T	I	M	A	T	U	M	U	L	T	I	M	A	T	U	M	U	L	T	I	M	A
E	S	T	O	E	S	L	A	G	U	E	R	R	A	C	A	B	A	L	L	E	R	O	S

A igual resultado se llega al utilizar la Tabla de Beaufort para descifrar.

De la misma forma que en el método de Vigenére, debe posicionarse en la fila correspondiente al carácter de la clave y buscar la retícula en la que aparezca el elemento cifrado, su proyección a la fila superior del texto en claro entrega el carácter del mensaje. Se puede observar que en este caso, la fila de desplazamiento 0 (letra *A*) no se corresponde con la del texto en claro como sucedía con Vigenére.

3.4.3. Variante de Beaufort (Criptosistema de Beaufort)

Existe otro método más simple matemático y rápido creado a partir de Beaufort; la sustitución empleada en este cifrador sigue la siguiente expresión:

$\mathbf{C_i} = E_{ki}(M_i) = (k_i - M_i) \bmod n$

De otra manera:

$\mathbf{C_i} = K_i \oplus (-M_i) \Rightarrow (\oplus$ (**resultado suma**) en módulo n)

Donde:

K_i = Cada letra de la clave.

M_i = Cada letra del texto a cifrar.

n = Cantidad de letras del alfabeto utilizado.

$E_{ki}(M_i)$ = Resultado de la resta de K_i menos M_i aplicando mod n.

C_i = Cada letra del mensaje cifrado.

Por ejemplo, para las tres primeras palabras del ejemplo anterior (ESTO ES LA) con los valores para cada letra según el alfabeto:

Cuadro 28. Ejemplo 01 Variación Beaufort

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12

Cuadro 28. Continuación

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

$\mathbf{k_1}$ = U; $\mathbf{C_1}$ = E ⇒ (20-4) mod 26 = 16 ⇒ $\mathbf{M_1}$ = Q

$\mathbf{k_2}$ = L; $\mathbf{C_2}$ = S ⇒ (11-18) mod 26 = 7 ⇒ $\mathbf{M_2}$ = H

$\mathbf{k_3}$ = T; $\mathbf{C_3}$ = T ⇒ (19-19) mod 26 = 0 ⇒ $\mathbf{M_3}$ = A

$\mathbf{k_4}$ = I; $\mathbf{C_4}$ = O ⇒ (8-14) mod 26 = 6 ⇒ $\mathbf{M_4}$ = G

$\mathbf{k_5}$ = M; $\mathbf{C_5}$ = E ⇒ (12-4) mod 26 = 8 ⇒ $\mathbf{M_5}$ = I

$\mathbf{k_6}$ = A; $\mathbf{C_6}$ = S ⇒ (0-18) mod 26 = 18 ⇒ $\mathbf{M_6}$ = S

$\mathbf{k_7}$ = T; $\mathbf{C_7}$ = L ⇒ (19-11) mod 26 = 8 ⇒ $\mathbf{M_7}$ = I

$\mathbf{k_8}$ = U; $\mathbf{C_8}$ = A ⇒ (20-0) mod 26 = 20 ⇒ $\mathbf{M_8}$ = U

La particularidad de este método es que tanto para cifrar como para descifrar se utiliza la misma fórmula o base.

3.5. DISCO DE ALBERTI

3.5.1. Introducción

Leon Battista Alberti es conocido como “El Padre de la Criptología Occidental”, en parte porque desarrollo la sustitución polialfabética. La sustitución polialfabética es una técnica que permite que diferentes símbolos cifrados puedan representar el mismo símbolo del texto claro. Esto dificulta la interpretación del texto cifrado por la aplicación del análisis de frecuencia.

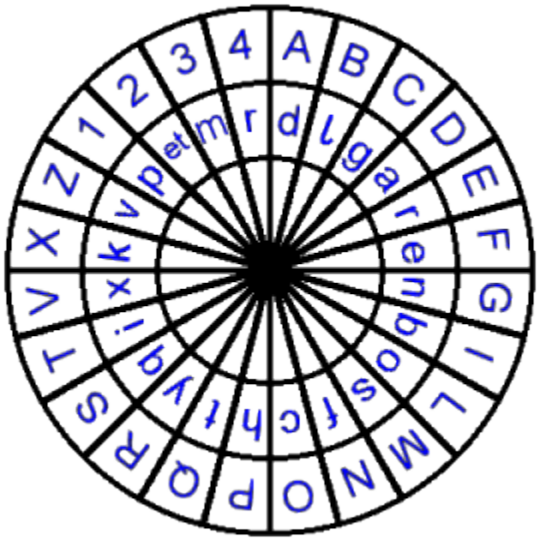
Para desarrollar esta técnica, Alberti estudio los métodos para quebrar cifras de la época y elaboró una cifra que podría anular estos métodos. La cifra de Alberti es una de las cifras polialfabéticas más elaboradas que no obtuvo el éxito merecido, siendo uno de los motivos para la decisión del autor de mantenerla secreta. Su famoso tratado “De Componendis Cyphris” solamente fue publicado en Venecia un siglo más tarde como parte de un otro, también de su autoría, el “oposcoli morali”, y pasó casi que desapercibido.

Alberti completó este descubrimiento, determinante en la historia de la criptología, con una otra invención notable: el código de recifrado. Él compuso una tabla con todas las combinaciones posibles de las cifras 1, 2, 3, 4, de 11 a 4444. De esta forma, obtuvo 336 grupos que utilizaba como un pequeño repertorio.

3.5.2. Descripción y uso

Alberti sugirió el uso de un disco compuesto por dos anillos concéntricos en su exterior los 20 caracteres del latín, estos son, los mismos del alfabeto castellano excepto las letras H, J, Ñ, K, U, W e Y, y se incluyen los números 1, 2, 3 y 4 para códigos especiales. Por otra, en el disco interior aparecen todos los caracteres del latín además del signo (& o et) y las letras H, K e Y. Al ser 24 los caracteres representados en cada disco, es posible definir hasta 24 sustituciones diferentes; es decir, dependiendo de la posición del disco interior la cantidad máxima de alfabetos de cifrado es igual a 24. Luego, para cifrar un mensaje, se repasa letra a letra el texto en claro del disco exterior y se sustituye cada una de ellas por la letra correspondiente del disco interior. Las 20 letras mayúsculas están en orden alfabética y las 24 minúsculas están fuera de orden. Letras minúsculas fuera de orden es una norma fundamental pues, si estuvieran en orden, la cifra sería sólo una generalización del Código de César.

Figura 4. Disco de alberti



Fuente: http://4.bp.blogspot.com/_gQzrdyDD4wE/SCq8o4fspNI/AAAAAAAAABE/yFSomnKjQv8/s320/Discoalberti.gif [online]

Fijada una letra minúscula como índice (por ejemplo la letra m), se debe ajustar el disco móvil interno y escribir, como primera letra del criptograma, la letra mayúscula que ha de escogerse para corresponder a la n (en el ejemplo, la letra G). A continuación, algunas letras son cifradas. Los números 1, 2, 3, y 4 sirven de nulos.

La innovación que supone este sistema consiste en que el alfabeto de sustitución puede ser cambiado durante el proceso de cifrado, por ejemplo cada k caracteres, simplemente girando el disco interior y por tanto utilizando otro alfabeto de sustitución. Una manera de hacerlo es cuando se escribe una nueva letra mayúscula para indicar claramente al destinatario de que hubo un cambio en el alfabeto cifrante. Se ajusta la nueva letra escogida para que coincida con la letra índice (por ejemplo, Z con m) y nuevamente algunas letras son cifradas. Para aumentar la seguridad (la letra mayúscula constituye una ayuda no sólo para el destinatario pero también para el "enemigo"), Alberti sugirió usar uno de los cuatro números para indicar la alteración de la lista cifrante. La letra minúscula correspondiente al número será la nueva llave. Decididamente se hace una cifra mucho más elaborada y claramente superior a las que la siguieron en el tiempo, inclusive la famosa Tabla de Vigenére.

Una vez establecida la correspondencia entre caracteres de ambos discos, se sustituye el texto en claro del disco exterior por cada una de las letras correspondientes del disco interior, cambiando al abecedario correspondiente (prefijado por los comunicantes) cada x palabras, habiendo sido x también prefijada por los comunicantes.

Ejemplo 1: Cifrar con el disco de Alberti, en la posición de la figura 4 el texto.

Texto a cifrar = FIRMA LA PAZ.

Texto cifrado = EBYSD OD HDV.

F	**I**	**R**	**M**	**A**	**L**	**A**	**P**	**A**	**Z**
E	B	Y	S	D	O	D	H	D	V

Ejemplo 2: Cifrar con el disco de Alberti, en la posición donde coinciden el numero “2” para el disco exterior y con la letra “s” para el interior (de acuerdo al orden de las letras de la figura 4), el siguiente texto.

Texto a cifrar = EL BATALLON ESTA LISTO.

Texto cifrado = IP THRHPPOM ISRH PVARR.

Cuadro 29. Ejemplo 01 disco de alberti

E	**L**	**B**	**A**	**T**	**A**	**L**	**L**	**O**	**N**	**E**	**S**	**T**	**A**	**L**	**I**	**S**	**T**	**O**
I	P	T	H	R	H	P	P	O	M	I	S	R	H	P	V	A	R	R

Ejemplo 3: Cifrar con el disco de Alberti, tomando como posición inicial la que se presenta en la figura 4 y girando el interior 4 posiciones en el sentido de las manecillas del reloj, cada 5 letras.

Texto a cifrar = ATACAMOS EN LA TARDE

Separación cada 5 letras = ATACA--MOS EN-- LA TAR—DE

Cuadro 30. Ejemplo 02 disco de alberti

A	**T**	**A**	**C**	**A**	**M**	**O**	**S**	**E**	**N**	**L**	**A**	**T**	**A**	**R**	**D**	**E**
D	I	D	G	A	E	B	C	D	N	D	I	O	I	N	Q	I

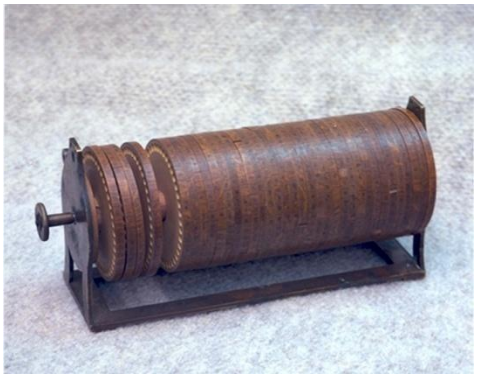
Texto cifrado = DIDGAEBC DN DI OINQI

Para descifrar el mensaje se utilizan las mismas instrucciones, la única diferencia es que para esta ocasión se reemplaza la letra del disco interior, con su correspondiente en el exterior y así obtener el mensaje original.

3.6. RUEDA DE JEFFERSON

Este dispositivo fue inventado por el archifamoso Thomas Jefferson (1743-1826), redactor de la declaración de independencia de Estados Unidos, aunque el primero en fabricarla en serie fue Ettiene Bazeries en 1891.

Figura 5. Rueda de Jefferson



Fuente: http://upload.wikimedia.org/wikipedia/commons/4/40/Jefferson%27s_disk_cipher.jpg [online]

La Rueda de Jefferson fue una de las primeras máquinas criptográficas elaborada en 1790 por Thomas Jefferson antes de que él se convirtiera en Presidente de los Estados Unidos. Consistía de 26 ruedas de madera, cada una de ellas, grabada con las letras del alfabeto en orden desordenado sobre su periferia, colocados sobre un eje común para formar un cilindro. El orden de las letras en cada rueda, era único y por lo tanto no había dos ruedas que fueran idénticos.

Para cifrar un texto plano, las ruedas fueron colocadas en el eje en un orden particular. Cada una de las letras del texto plano luego eran colocadas en ruedas consecutivas, una letra en cada rueda. Las ruedas eran rotadas a esa posición en la que el texto aparecía en una fila. Cualquiera de las otras filas, podría ser utilizado como sistema de cifrado de texto. Para descifrar el mensaje, el receptor tendría que organizar el texto cifrado en una fila en la rueda de Jefferson. Después el receptor tiene que examinar todas las otras filas y ver cuál es la que tiene más

sentido. Esta era la fila de texto plano. Es muy poco probable que más una fila tenían sentido. Una de las principales desventajas de este método de cifrado es que la máquina tenía que ser previamente distribuida a los posibles destinatarios del mensaje. En una época en que ir de punto A al punto B fácilmente podría tomar un mes, el proceso llevaba mucho tiempo. Así que Jefferson lo descarto y más tarde comenzó a utilizar sistemas de cifrado más convenientes.

Ejemplo:

- ✓ Supóngase una rueda de Jefferson de 10 discos y que cada uno es extendido en una tira, en la cual se puede ver todas las letras en desorden.
- ✓ A partir de estas tiras, se busca formar un mensaje, ubicando 1 letra por cada tira.

Cuadro 31. Ejemplo 01 Rueda de Jefferson

C	Y	J	A	V	M	M	A	B	N
Z	Z	S	B	W	N	L	Z	Q	O
A	J	Q	C	X	L	K	B	R	P
Y	H	H	D	E	O	J	Y	C	Q
X	S	T	E	Y	K	I	C	P	R
B	***R***	***P***	***F***	***C***	***Z***	***H***	***X***	***S***	***S***
W	E	E	G	U	J	G	D	D	T
L	Q	O	H	H	X	F	W	O	U
V	F	A	I	T	I	E	E	T	V
M	U	N	J	I	W	D	U	E	W
K	G	F	K	Z	H	C	F	M	X
U	C	G	L	S	Y	B	T	U	Y
F	**I**	**R**	**M**	**A**	**L**	**A**	**P**	**A**	**Z**
N	T	U	N	J	P	N	S	N	A
G	D	M	O	B	Q	O	G	V	B
T	V	L	P	P	A	P	R	F	C
J	B	B	Q	K	R	Q	H	Z	D
E	P	V	R	Q	B	R	Q	W	E
O	W	K	S	O	S	S	I	G	F
S	A	C	T	G	C	T	V	L	G
D	O	W	U	F	T	U	J	X	H
H	N	I	V	L	D	V	O	H	I
P	M	D	W	M	U	W	K	K	J
I	L	Z	X	R	E	X	N	Y	K
Q	***K***	***Y***	***Y***	***D***	***V***	***Y***	***L***	***I***	***L***
R	X	X	Z	N	F	Z	M	J	M

Utilícese la Frase = FIRMA LA PAZ.

Uno de los principios de este método se basa en que el texto cifrado puede tomarse de cualquier otra línea del cilindro; para este caso, dos posibles textos cifrados para el ejemplo serian:

1. BRPFCZHXSS
2. QKYYDVVYLIL

Cuadro 32. Ejemplo 02 Rueda de Jefferson

F	I	R	M	A	L	A	P	A	Z
B	***R***	***P***	***F***	***C***	***Z***	***H***	***X***	***S***	***S***
Q	***K***	***Y***	***Y***	***D***	***V***	***Y***	***L***	***I***	***L***

Así, de esta manera podría seguirse escogiendo cualquiera de las filas o líneas de la rueda e igualmente seria un cifrado valido para este método, con la gran ventaja de que un análisis estadístico se hace extremadamente difícil.

En el temprano siglo 19, un criptógrafo francés, Etienne Bazeries re-inventó un dispositivo similar llamado el Cilindro Bazeries con 20 a 30 discos. Pronto, en 1922, el ejército de los EE.UU. adopto un dispositivo similar cuyo nombre en código es M-94. Ambicionado por el Coronel Parker Hitt y aplicado por el Mayor Joseph Mauborgne. Hecho de aluminio y con 25 discos. Las letras del disco, 17 leían "ARMYOFTHEUS" a lo largo de su periferia. Este dispositivo mantuvo en servicio hasta 1943. La Marina de los EE.UU. también hizo uso de este dispositivo bajo el nombre de código CSP488.

3.7. LA CIFRA ADFGVX

3.7.1. Cifrado

Este sistema de cifrado fue introducido por los alemanes en marzo de 1918, poco antes de la gran ofensiva que lanzaron sobre París. Afortunadamente para los franceses, en la Oficina de Cifras trabajaba un criptoanalista llamado Georges Painvin que dedicó todos sus esfuerzos a descifrarla; perdió 15 kilos en el esfuerzo, pero al conseguirlo permitió que los alemanes perdieran el factor sorpresa y fueran finalmente derrotados.

En la cifra ADFGVX hay sustitución y trasposición. La codificación comienza dibujando una cuadrícula de 6x6, llenando los 36 cuadrados con una disposición aleatoria de las letras y los diez dígitos. Cada fila y cada columna de la cuadrícula se identifican con una de las seis letras ADFGVX.

La cifra se nombra después de las seis letras posibles usadas en el texto cifrado: A, D, F, G, V y X. Estas letras fueron elegidas deliberadamente porque suenan muy diferentes de uno a cuando están transmitidas vía Código Morse. La intención era reducir la posibilidad de error del operador.

La disposición de la cuadrícula es parte de la clave, de modo que debe ser conocida por el emisor y por el receptor.

Tabla 10. Tabla de la cifra ADFGVX

	A	**D**	**F**	**G**	**V**	**X**
A	8	P	3	D	1	N
D	L	T	4	O	A	H
F	7	K	B	C	5	Z
G	J	U	6	W	G	M
V	X	S	V	I	R	2
X	9	E	Y	0	F	Q

La primera fase del cifrado consiste en sustituir cada letra del texto llano por la pareja fila-columna que corresponden a su posición en la cuadrícula acordada.

Ejemplo:

Texto: ATACAMOS POR LA TARDE

Cifrado fase 1:

Cuadro 33. Ejemplo 01 de la cifra ADFGVX

A	**T**	**A**	**C**	**A**	**M**	**O**	**S**	**P**	**O**	**R**	**L**	**A**	**T**	**A**	**R**	**D**	**E**
DV	DD	DV	FG	DV	GX	DG	VD	AD	DG	VV	DA	DV	DD	DV	VV	AG	XD

Hasta aquí se trata de una cifra de sustitución mono alfabética y bastaría un análisis estadístico de frecuencias para descifrarla. Sin embargo, la segunda fase consiste en una trasposición, lo que dificulta mucho más el criptoanálisis. La

trasposición depende de una clave; para el ejemplo en desarrollo será la palabra PACO, la cual debe ser compartida por el emisor y el receptor.

Hay que tener en cuenta que el doble del número de letras del texto ha de ser divisible por el número de letras de la palabra clave. Si ello no fuese así, se puede añadir un cierto número de ceros al final del texto inicial para que se cumpla lo indicado.

La trasposición se lleva a cabo de la siguiente manera: Se construye una cuadrícula con tantas columnas como letras tenga la palabra clave, y se escribe el texto cifrado en la fase 1 en las filas de la cuadrícula de la siguiente manera:

Cuadro 34. Ejemplo clave de la cifra ADFGVX

P	A	C	O
D	V	D	D
D	V	F	G
D	V	G	X
D	G	V	D
A	D	D	G
V	V	D	A
D	V	D	D
D	V	V	V
A	G	X	D

A continuación, se cambia el orden de las columnas, poniendo las letras de la palabra clave en orden alfabético de la siguiente manera:

Cuadro 35. Ejemplo 02 de la cifra ADFGVX

A	C	O	P
V	D	D	D
V	F	G	D
V	G	X	D
G	V	D	D
D	D	G	A
V	D	A	V
V	D	D	D
V	V	V	D
G	X	D	A

Para completar la segunda fase de cifrado, después de la trasposición se leen las letras de la cuadricula, pero de acuerdo al orden de las columnas, es decir:

Texto cifrado final:

Cuadro 36. Ejemplo 03 de la cifra ADFGVX

A	**T**	**A**	**C**	**A**	**M**	**O**	**S**	**P**	**O**	**R**	**L**	**A**	**T**	**A**	**R**	**D**	**E**
VV	VG	DV	VV	GD	FG	VD	DD	VX	DG	XD	GA	DV	DD	DD	DA	VD	DA

El texto cifrado final se transmite en código Morse y el receptor invierte el proceso de codificación para obtener el texto original.

Todo el texto cifrado se compone con sólo 6 letras A, D, F, G, V y X. La razón de su elección se basa en que sus equivalentes en el código Morse son muy diferentes y ello reduce al mínimo el número de errores durante la transmisión del mensaje, que se hacía mediante una emisora de radio y un aparato telegráfico.

3.7.2. Descifrado

Para descifrar el mensaje el receptor debía aplicar un proceso de codificación para obtener el texto original.

Ejemplo:

1. Descifrar el siguiente mensaje, en cifra ADFGVX con la palabra clave LISBOA:

   DXDFVXDGVDDDGFXXGGGDDADAFVDGVAVFVDXDDXGXDX

Organizado:

D	**X**	**D**	**F**	**V**	**X**	**D**	**G**	**V**	**D**	**D**	**D**	**G**	**F**	**X**	**X**	**G**	**G**	**G**	**D**	**D**

Continúa:

A	**D**	**A**	**F**	**V**	**D**	**G**	**V**	**A**	**V**	**F**	**V**	**D**	**X**	**D**	**D**	**X**	**G**	**X**	**D**	**X**

A partir de lo anterior ya se sabe que la palabra clave “LISBOA” tiene 6 letras y el texto cifrado posee 42 letras, por tal al dividirlos obtenemos que la cuadricula debe ser de 6 columnas por 7 filas mas la fila de la clave para una tabla de 6 x 8.

Para el llenado de la tabla debe tenerse en cuenta que las letras de la palabra clave para la primera fila debe colocarse en orden alfabético y las letras del texto cifrado deben colocarse de manera consecutiva llenando columna por columna:

Cuadro 37. Ejemplo 04 de la cifra ADFGVX

A	**B**	**I**	**L**	**O**	**S**
D	G	X	A	V	D
X	V	X	D	A	D
D	D	G	A	V	X
F	D	G	F	F	G
V	D	G	V	V	X
X	G	D	D	D	D
D	F	D	G	X	X

Posteriormente se organizan las columnas para formar la palabra clave original:

Cuadro 38. Ejemplo 05 de la cifra ADFGVX

L	**I**	**S**	**B**	**O**	**A**
A	X	D	G	V	D
D	X	D	V	A	X
A	G	X	D	V	D
F	G	G	D	F	F
V	G	X	D	V	V
D	D	D	G	D	X
G	D	X	F	X	D

Ahora debe de leerse cada letra pero en el orden de las filas:

AXDGVDDXDVAXAGXDVDVGGDFFVGXDVVDDDGDXGDXFXD

Y se procede a organizar por parejas:

AX	DG	VD	DX	DV	AX	AG	XD	VD	FG	GD

FF	VG	XD	VV	DD	DG	DX	GD	XF	XD

En este punto solo queda tomar la tabla acordada y se ubica la letra que corresponde a la pareja fila-columna tomada:

Cuadro 39. Ejemplo 06 de la cifra ADFGVX

	A	**D**	**F**	**G**	**V**	**X**
A	8	P	3	D	1	N
D	L	T	4	O	A	H
F	7	K	B	C	5	Z
G	J	U	6	W	G	M
V	X	S	V	I	R	2
X	9	E	Y	0	F	Q

Se ubican las letras del mensaje original:

Cuadro 40. Ejemplo 07 de la cifra ADFGVX

AX	**DG**	**VD**	**DX**	**DV**	**AX**	**AG**	**XD**	**VD**	**FG**	**GD**
N	O	S	H	A	N	D	E	S	C	U

FF	**VG**	**XD**	**VV**	**DD**	**DG**	**DX**	**GD**	**XF**	**XD**
B	I	E	R	T	O	H	U	Y	E

Texto Original: NOS HAN DESCUBIERTO HUYE

Cabe mencionar que cualquier error en el texto cifrado hace que el mensaje original no tenga sentido.

3.8. CRIPTOSISTEMA DE HILL

3.8.1. Introducción

Los cifrados monográficos, en los que se sustituye un carácter por otro de una forma preestablecida, son vulnerables al análisis de frecuencia de aparición de las letras. Para evitarlo se desarrollaron esquemas basados en cifrar bloques de letras de una cierta longitud fija, o sea, cifrado poligráfico.

Este sistema está basado en el álgebra lineal y ha sido importante en la historia de la criptografía. Fue Inventado por Lester S. Hill en 1929, y fue el primer sistema criptográfico polialfabético que era práctico para trabajar con más de tres símbolos simultáneamente.

Suponiendo que se trabaja con un alfabeto de 26 caracteres.

Las letras se numeran en orden alfabético de forma tal que A=0, B=1,…, Z=25:

Cabe destacar que también se utilizan alfabetos en los que se incluye el espacio como un carácter.

Un cifrado de Hill se obtiene al transformar bloques de n caracteres del texto original, a un texto cifrado a través de la relación

$C = (A \cdot P) \pmod{26}$

- Donde A es una matriz n x n que debe ser invertible módulo 26 (Debe recordarse que el 26 proviene de que solo se están utilizando las letras del abecedario anglosajón de la A a la Z que son justamente 26) es decir que el m.c.d del determinante de A y 26 sea 1; dicho de otro modo, que el determinante de A no divida a 26 ni sea un múltiplo de 26.
- C es la matriz columna resultante del cifrado de P (Un bloque de n caracteres).

3.8.2. Cifrado

Se elije un entero “d” que determina bloques de “d” elementos que son tratados como un vector de “d” dimensiones. Se elije de forma aleatoria una matriz de “d × d” elementos los cuales serán la clave a utilizar y que deberá ser pactada por el emisor con el receptor.

Ejemplo:

- ✓ Se pacta la matriz de 2 x 2: $\begin{bmatrix} 11 & 8 \\ 3 & 2 \end{bmatrix}$ la cual será la clave.
- ✓ Esto significa que del texto original deberá tomarse de a 2 letras.

Los elementos de la matriz de (d × d) para este caso, deben ser enteros entre 0 y 25, además la matriz M debe ser invertible en $\mathbb{Z}_{26}^{n}$.

Para la encriptación, el texto es dividido en bloques de d elementos los cuales se multiplican por la matriz M= (d × d). Todas las operaciones aritméticas se realizan en la forma modulo 26, es decir que 26=0, 27=1, 28=2 y así sucesivamente.

Dado un mensaje a encriptar se debe tomar bloques de "d" caracteres del mensaje a enviar y aplicar: $M \times P_i = C$, donde C es el código cifrado para el mensaje P_i.

Ejemplo:

Se desea cifrar la palabra CODIGO y se pacta la siguiente matriz como la clave:

$$A = \begin{bmatrix} 5 & 17 & 20 \\ 9 & 23 & 3 \\ 2 & 11 & 13 \end{bmatrix}$$

Antes que nada se debe verificar que la matriz elegida sea invertible en modulo 26. Hay una forma relativamente sencilla de averiguar esto a través del cálculo del determinante. Si el determinante de la matriz es 0 o tiene factores comunes con el módulo (en el caso de 26 los factores son 2 y 13), entonces la matriz no puede utilizarse. Al ser 2 uno de los factores de 26 muchas matrices no podrán utilizarse, en resumen (no servirán todas en las que su determinante sea 0, un múltiplo de 2 o un múltiplo de 13).

Para ver si es invertible, se calcula el determinante de A:

$$A = \begin{bmatrix} 5 & 17 & 20 \\ 9 & 23 & 3 \\ 2 & 11 & 13 \end{bmatrix} = 5(23*13 - 3*11) - 17(9*13 - 3*2) + 20(9*11 - 23*2)$$

Det A = 1215 – 1734 + 1060 = 503 mod 26 = 9

La matriz A es invertible en modulo 26, por lo que si sirve de clave.

Para encriptar la palabra con esta matriz de 3 x 3, se debe de dividir la palabra en bloques de 3 caracteres, obteniendo así:

- ✓ P_1=”COD”
- ✓ P_2=”IGO”

Posteriormente se obtiene su equivalente en número según el alfabeto definido:

- ✓ P_1=“2, 14, 3”
- ✓ P_2=”8, 6, 14”

Y se procede a realizar la operación de cifrado:

Primer Bloque:

$$A \cdot P_1 = \begin{bmatrix} 5 & 17 & 20 \\ 9 & 23 & 3 \\ 2 & 11 & 13 \end{bmatrix} \begin{bmatrix} 2 \\ 14 \\ 3 \end{bmatrix} = \begin{bmatrix} 5x2 + 17x14 + 20x3 \\ 9x2 + 23x14 + 3x3 \\ 2x2 + 11x14 + 13x3 \end{bmatrix} = \begin{bmatrix} 308 \\ 349 \\ 197 \end{bmatrix} \text{ mod } 26 = \begin{bmatrix} 22 \\ 11 \\ 15 \end{bmatrix}$$

El primer bloque “COD” se codifica entonces como “WLP”.

Segundo Bloque:

$$A \cdot P_2 = \begin{bmatrix} 5 & 17 & 20 \\ 9 & 23 & 3 \\ 2 & 11 & 13 \end{bmatrix} \begin{bmatrix} 8 \\ 6 \\ 14 \end{bmatrix} = \begin{bmatrix} 5x8 + 17x6 + 20x14 \\ 9x8 + 23x6 + 3x14 \\ 2x8 + 11x6 + 13x14 \end{bmatrix} = \begin{bmatrix} 422 \\ 252 \\ 264 \end{bmatrix} \text{ mod } 26 = \begin{bmatrix} 6 \\ 18 \\ 4 \end{bmatrix}$$

El segundo bloque “IGO” se codifica entonces como “GSE”.

Al final se obtiene “WLPGSE”.

Se observa que las dos "O" se codificaran de forma diferente.

Cuadro 41. Ejemplo 01 de HILL

C	**O**	**D**	**I**	**G**	**O**
W	L	P	G	S	E

Para desencriptar el método es idéntico al anterior pero usando la matriz inversa de la usada para encriptar.

3.9. LA MÁQUINA ENIGMA

3.9.1. Historia (El advenimiento de Enigma)

Enigma es hija del desarrollo técnico del siglo XX. Frente a todo la anterior, “papel y lápiz”, Enigma significa la mecanización aplicada al mundo de la criptografía.

En 1918 el ingeniero eléctrico e inventor alemán Arthur Scherbius, estaba molesto con los ineficaces métodos manuales de cifrado y siendo el conocedor de la debilidad que había mostrado la ciencia criptográfica alemana antes y durante la I

Guerra Mundial, ideó un sistema de cifra que magnificara las ventajas de la técnica, gran fiabilidad, facilidad de uso, robustez y que proveyera comunicaciones absolutamente seguras.

Fundada una empresa con un socio, Richard Ritter, patentó un aparato. Este resultó ser una caja compacta de 34 x 28 x 15 cm, pero con el considerable peso de 12 kilos, y un coste bastante elevado, el equivalente a más de 30.000 euros de hoy en día. Por tal, no tuvo el éxito esperado; a pesar de ser muy avanzada para su época, por lo que el invento pasó a ser ofrecido a los militares.

En un principio, también el ejército alemán rechazó el invento, pero tras sucesivas modificaciones el artefacto fue aceptado en 1923 cuando las Fuerzas Armadas se dieron cuenta de lo que le había pasado a raíz de la publicación de unos documentos ingleses donde describían como habían roto los sistemas de cifrado alemanes, por esta razón, frenéticamente buscaron una solución, y la encontraron en Enigma. Gracias a ésta decisión, las fuerzas armadas alemanas dejaron atrás el engorroso sistema de libros de claves para centralizar todo el cifrado/descifrado de sus comunicaciones en un aparato del tamaño y apariencia de una máquina de escribir.

A partir de 1925, se encargó la fabricación en serie de Enigma para todas las ramas de las Fuerzas Armadas, comenzando por la más entusiasta del invento, la Marina, (el Ejército comenzó su uso intensivo en 1929) así como para ministerios y organizaciones gubernamentales, como telégrafos, ferrocarriles, las altas jerarquías nazis, etc. Según se iba introduciendo la Enigma, Alemania se fue blindando en cuanto a comunicaciones se refiere, como constataba que el desciframiento que habitualmente se hacía de los mensajes alemanes por parte de otros países cayeron en picada hasta cero. Estos países, después de unos intensos estudios de la nueva cifra, llegaron a la conclusión de que Alemania tenía las comunicaciones más invulnerables del mundo, y aparcaron resignados el asunto. Scherbius no llegó a conocer el inmenso éxito de su invento (y su posterior desciframiento) puesto que murió en 1929 en un accidente en una carrera de caballos.

3.9.2. Funcionamiento básico de Enigma

La máquina Enigma no es en realidad muy complicada, pero el ingenioso uso

conjunto de las partes que la componían, hizo de ella durante mucho tiempo un invulnerable sistema de cifrado.

Elementalmente, Enigma consistía en un teclado de máquina de escribir con las correspondientes letras del alfabeto, un dispositivo llamado modificador, que realizaba la codificación, y un tablero de luces donde se indicaba el texto ya cifrado. De tal manera que el operador tecleaba una letra del texto y veía como se iluminaba otra letra del tablero de luces, que era la equivalente en el texto cifrado y así con todas las letras del texto. El modificador (también llamado rotor) consiste en una ruleta o cilindro con 26 cables o conexiones, conectado al teclado por 26 puntos, uno por cada letra, y que cada cable daba una serie de vueltas hasta salir por otro punto conectado al tablero de luces, iluminando una letra. Dependiendo de la posición del modificador (según por donde entra la pulsación de la tecla). Enigma ofrecía 26 posibles claves, ya que el modificador podía ser girado a mano por el operador, puesto que tenía grabado los números del 1 al 26 en una ruleta dentada. Así, poniendo el modificador en una posición dada, el operador elegía la clave con la que iba a cifrar el mensaje (posición que debía saber el receptor, pero el uso real de Enigma se explica más adelante).

Así descrita, Enigma era muy débil, puesto que no hacía más que un cifrado de sustitución monoalfabético, es decir, sustituir una letra por otra, según el recorrido del cable del modificador, siendo fácil de descifrar probando una a una las 26 posibles claves. Así que añadió otra característica, que era que el modificador girara un veintiseisavo (1/26) de vuelta después de cada pulsación, de tal manera que cambiaba las posiciones de los cables en los puntos de entrada del teclado y en los de salida al tablero de luces cada letra del mensaje. Con esto consiguió que Enigma hiciera un cifrado de sustitución polialfabético.

Las 26 posiciones del modificador hacen referencia al siguiente alfabeto:

Tabla 11. Equivalencia numeración letras

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**
01	02	03	04	05	06	07	08	09	10	11	12	13

N	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
14	15	16	17	18	19	20	21	22	23	24	25	26

De acuerdo al funcionamiento anterior, supóngase 1 disco con las se divisiones, el cual se desenrolla como una tira:

Estado Inicial:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Pulsación 1:

Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Pulsación 2:

Y	**Z**	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X

Pulsación 3:

X	**Y**	**Z**	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W

Pulsación 4:

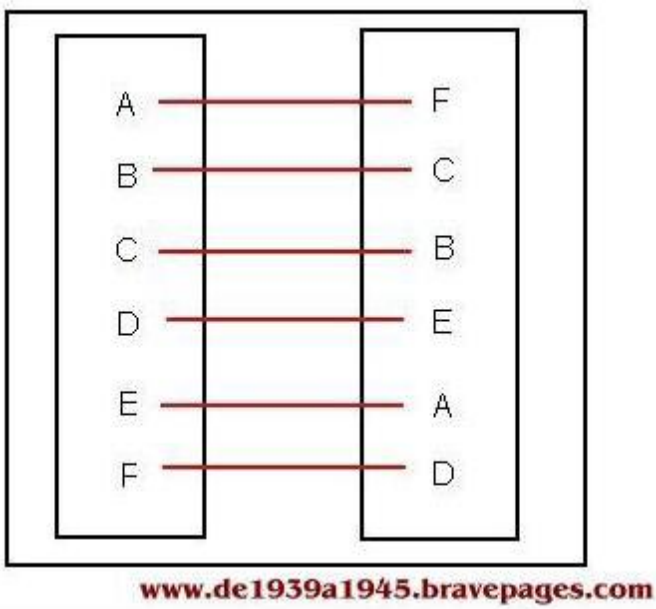
W	**X**	**Y**	**Z**	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V

Y así sucesivamente, cada vez que se presiona una tecla, el disco o modificador, da un veintiseisavo (1/26) de giro.

3.9.3. Funcionamiento ideal de Enigma

De acuerdo al sistema anterior con un solo disco, con cada pulsación y el correspondiente giro del modificador, se cambiaba de alfabeto de cifrado y así hasta 26 alfabetos para cifrar, de tal manera que, por ejemplo, teclear AA daba FT (sin el giro, habría dado FF), momento que comenzaba el cifrado como en la primera pulsación. Al proporcionar 26 claves (alfabetos) que se iban turnando a cada pulsación automáticamente, aumentó la seguridad, pero aun seguía siendo un sistema débil, así que añadió otros 2 modificadores conectados uno a otro, de tal manera que el segundo modificador sólo daba un giro cuando el primero había completado sus 26 giros, y el tercero daba un giro cuando el segundo daba sus correspondientes 26 giros. Es algo parecido al funcionamiento del segundero, minutero y aguja horaria de un reloj.

Figura 6. Esquema de dos cilindros



Fuente: www.de1939a1945.bravepages.com [online]

De tal manera que ahora Enigma tenía a su disposición 26 x 26 x 26 = **17.576** claves que iban cambiando continuamente con cada pulsación. Esto ya brindaba una seguridad moderada.

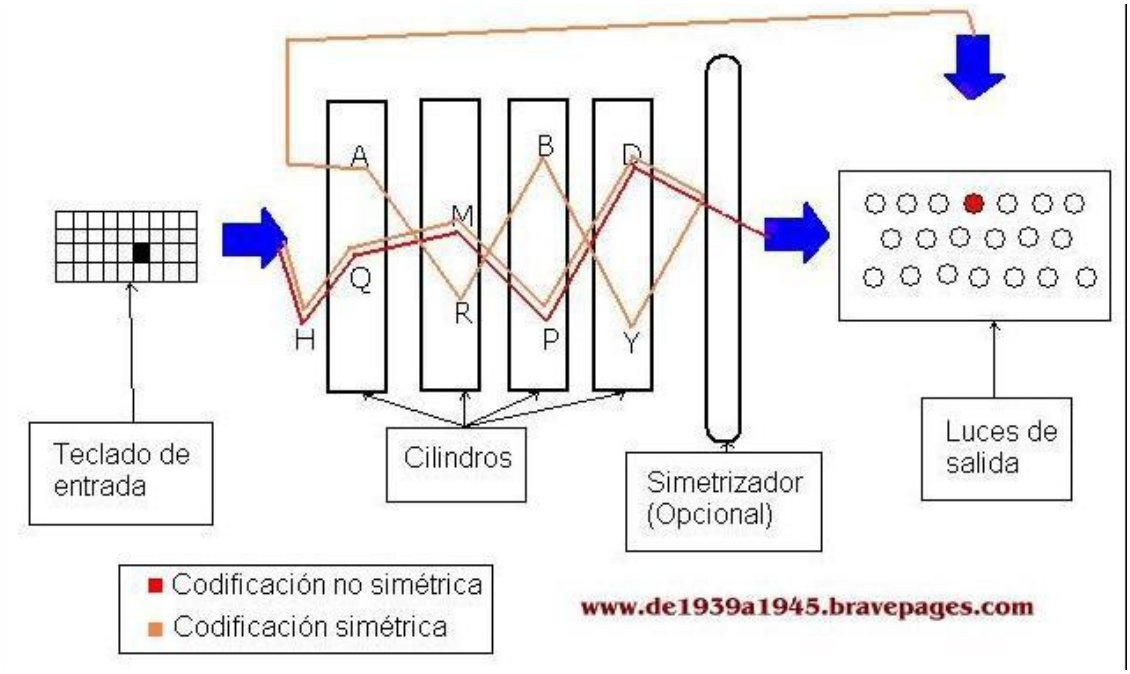
En su avance hacia la definitiva Enigma, Scherbius hizo que los modificadores tuvieran el alfabeto en orden variable (Sin embargo la numeración si conservo su orden del 01 al 26, para efectos de manejo de las claves), llegando a conocerse 5 juegos de cilindros con alfabetos en orden aleatorio, además permitió que fueran intercambiables, es decir, el modificador número uno podía ponerse en el hueco número 3, el número 3 en el hueco 2 y el número 2 en el hueco 1 (Era necesario que tanto emisor como receptor usaran el mismo juego de discos, 3 en este caso). Esto aumentaba el número de claves en un factor de 6, las posibles posiciones de los modificadores, a saber: (1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1) de tal manera que el número de claves aumentó a 17.576 x 6 = **105.456**.

En su búsqueda de la máquina de cifrado perfecta, Scherbius añadió otros dos elementos, (podía haber seguido añadiendo modificadores, pero eso habría hecho a Enigma demasiado voluminosa y pesada).

Uno fue el elemento reflector o simetrizador, que es en realidad un modificador, pero que no gira, se limita a recibir la señal que ha pasado por los tres

modificadores y reenviarla hacia el tablero de luces pasando otra vez por los tres modificadores pero por otro camino distinto, de tal manera que llegue la misma señal o letra que recibió. Esto parece que no sirve de mucho, porque no añade más claves a la máquina, pero con ello se conseguía un sistema simétrico, en el que el camino de codificación realizaba también la decodificación (De otra manera, hubieran sido necesarios un código para cifrar y otro para descifrar. Así, sólo era necesario introducir el mensaje cifrado con la misma clave inicial y este aparecía descifrado en los indicadores luminosos).

Figura 7. Esquema de funcionamiento de Enigma



Fuente: http://www.de1939a1945.com/imagenes/at_enigmafuncionamiento.jpg [online]

El otro elemento es un tablero de clavijas, donde se podían intercambiar mediante cables, pares de letras antes de entrar en los modificadores. Los operadores de Enigma disponían de 6 cables (luego fueron mas) para conectar 6 pares de letras de entre las 26 posibles, de tal manera que por ejemplo se conectaba un cable de la D a la H, así, a la hora de codificar el mensaje, el operador tecleaba la D del mensaje a cifrar, pero en realidad, esa D seguía el camino por los modificadores que le hubiera correspondido a la H.

Teniendo en cuenta las maneras de conectar intercambiando seis pares de letras entre 26, esto producía **100.391.791.500** combinaciones posibles, es decir, mas claves, que multiplicadas por **105.456** daba la curiosa e inmensa cifra de:

3,283,883,513,796,974,198,700,882,069,882,752,878,379,955,261,095,623,685,444,055,315,226,006,433,616,627,409,666,933,182,371,154,802,769,920,000,000,000

Aproximadamente **3.284 x 10^38** claves posibles en Enigma, algo fuera del alcance humano para descifrar. En ese momento Enigma se convirtió en el sistema indescifrable que Scherbius soñaba.

3.9.4. Operación de Enigma

Conociéndose ya las partes y el funcionamiento de Enigma como tal, a partir de la descripción anterior, se deduce que la "clave", que tanto emisor como receptor debe conocer, está compuesta por:

- ✓ El orden de los modificadores en los huecos.
- ✓ La posición inicial de estos (de 1 a 26 que se coloca con la ruleta).
- ✓ Las conexiones del clavijero.

Mensualmente, se distribuía entre los operadores de Enigma un libro de claves, con una clave para cada día del mes (según avanzaba la guerra, fue distribuido en periodos más cortos), de tal manera que el operador, en un día dado, leía algo como:

- 3 - 1 - 2
- 23-24-8
- A/X,F/Q,K/U,S/E,P/I,M/L

Lo que le indicaba que debía hacer lo siguiente:

- ✓ Poner el tercer (3) modificador en el primer hueco y girarlo hasta la posición (23).

- ✓ Poner el primer (1) modificador en el segundo hueco y girarlo hasta la posición (24).
- ✓ Poner el segundo (2) modificador en el tercer hueco y girarlo hasta la posición (8).
- ✓ Así mismo debía conectar los seis cables en el clavijero con los pares de letras indicados; la A con la X, la F con la Q, la K con la U, la S con la E, la P con la I y la M con la L.

Y con esta disposición de la máquina, comenzaba el envío de mensajes.

Figura 8. Máquina enigma con un juego de 5 cilindros sin colocar.

Sólo se empleaban 3 en la comunicación.

Fuente: http://www.de1939a1945.com/imagenes/at_enigmacilindros.jpg

El receptor debía asimismo colocar la máquina en la misma disposición según el libro de códigos, y aquí es donde juega su papel el reflector o simetrizador. Simplemente teclearía el mensaje cifrado recibido, y el tablero de luces le daría el mensaje original sin codificar. Obviamente eso tenía una gran importancia en el

tiempo de operación en tiempos de guerra, obviando la necesidad de lápiz y papel para el descifrado del mensaje.

3.9.5. Ultimo refuerzo de seguridad

Los alemanes se dieron cuenta que funcionando así, generarían un sinfín de mensajes con la misma clave (ya adelantaban que en periodos de operaciones de guerra, el tráfico de comunicaciones iba a ser inmenso) durante todo un día, y esto es un asidero para los criptoanalistas. Desde el principio de la historia de la criptografía, un número significativo de mensajes codificados con la misma clave ha dado a los descifradores un punto de partida para romper la clave. Conscientes de ellos, emitieron una serie de órdenes sobre cómo se debía utilizar Enigma. Lo que no fueron conscientes es que al señalar una serie de normas estrictas, aunque al principio pudieran parecer sensatas, estaban proporcionando otros tipos de asideros a los que los criptoanalistas se iba a agarrar cual clavo ardiendo, que fueron el principio del fin de Enigma.

Estas normas de uso eran principalmente:

- No se podía conectar una letra con su inmediata anterior o posterior en el clavijero.
- Un modificador no podía estar más de un día en el mismo hueco. Una letra no podía ser codificada como si misma (una B nunca sería codificada como B), esto lo hacía el diseño del cableado interno de los modificadores.

Pero la norma más importante fue el concepto de “clave de mensaje”. Como se comenta arriba, se quería evitar un intenso tráfico de mensajes con la misma clave. Por eso decidieron que cada mensaje que se enviara debería tener su propia clave.

Esto plantea un problema obvio, ¿cómo podía saber el receptor, la clave con la que había sido codificado el mensaje que había recibido?

La solución fue operando de la siguiente manera:

El emisor organizaba a Enigma según la clave del día indicada por el libro de códigos, de acuerdo a los modificadores, su orden y las conexiones de los cables.

Con esta configuración, escribía tres letras elegidas al azar, por ejemplo **ABC**, y las tecleaba, obteniendo como salida, por ejemplo **TYU** (La salida varía de acuerdo a la organización de Enigma) en el tablero de luces. Luego, giraba los modificadores desde su posición inicial (La indicada para ese día por el libro) a la posición de esas tres letras, en este caso **1 - 2 - 3** (A=1, B=2 y C=3). De tal manera que el mensaje transmitido era **TYU**, que era la codificación de **ABC** según la clave del día. La clave podría ser cualquier otra, por ejemplo en el caso de digitar las tres letras iníciales **DFI** los números serían (**4 - 6 - 9**), se transmiten estas tres letras en esa configuración de Enigma, se giran los modificadores desde su posición inicial a las nuevas posiciones (**4 - 6 - 9**), pero se deja su orden en los huecos y los cables igual, y entonces se codifica el mensaje a enviar.

El receptor, que tenía la máquina dispuesta con la clave del día, recibía la transmisión y se fijaba en las primeras tres letras. Las tecleaba (**TYU** para este ejemplo) y veía **ABC**. Entonces giraba los modificadores a esa disposición, **1 - 2 - 3**, y tecleaba el resto del mensaje, obteniendo el original.

Como ultima norma, los alemanes obligaban a teclear dos veces seguidas las tres letras de la clave de mensaje, para evitar errores por interferencias en la transmisión o de los operadores. Así que realmente el emisor, con la clave del día, tecleaba **ABCABC**, obteniendo como salida por ejemplo, **TYUPLO** (como se explicó antes, con el giro de los modificadores, se codifica cada letra de distinta manera, así se repita), y luego orientaba los modificadores a **1 - 2 - 3** y codificaba el mensaje.

El receptor recibía el mensaje, tecleaba las seis primeras letras, **TYUPLO**, y veía **ABCABC**, con lo que ya sabía la clave de mensaje.

Ejemplo:

Supóngase la organización de 3 modificadores con su reflector (1 - 15 - 8):

Cuadro 42. Ejemplo 01 maquina enigma

01	**Y**		15	**B**		08	**N**	**A**
02	**Z**		16	**Q**		09	**O**	**B**
03	**J**		17	**R**		10	**P**	**C**
04	**H**		18	**C**		11	**Q**	**D**
05	**S**		19	**P**		12	**R**	**E**
06	**R**		20	**S**		13	**S**	**F**
07	**E**		21	**D**		14	**T**	**G**
08	**Q**		22	**O**		15	**U**	**H**
09	**F**		23	**T**		16	**V**	**I**
10	**U**		24	**E**		17	**W**	**J**
11	**G**		25	**M**		18	**X**	**K**
12	**C**		26	**U**		19	**Y**	**L**
13	**I**		01	**A**		20	**Z**	**M**
14	**T**		02	**N**		21	**A**	**N**
15	**D**		03	**V**		22	**B**	**O**
16	**V**		04	**F**		23	**C**	**P**
17	**B**		05	**Z**		24	**D**	**Q**
18	**P**		06	**W**		25	**E**	**R**
19	**W**		07	**G**		26	**F**	**S**
20	**A**		08	**L**		01	**G**	**T**
21	**O**		09	**X**		02	**H**	**U**
22	**N**		10	**H**		03	**I**	**V**
23	**M**		11	**K**		04	**J**	**W**
24	**L**		12	**Y**		05	**K**	**X**
25	**K**		13	**I**		06	**L**	**Y**
26	**X**		14	**J**		07	**M**	**Z**

Con una correspondencia lineal entre cada letra de cada disco, dos ejemplo para este caso:

- **X = J = M** es decir, se teclea **X** y como salida al reflector se obtiene **M** y partiendo de la **M** del reflector **Z = A = I** como salida final**.**
- **V = F = C** es decir, se teclea **V** y como salida al reflector se obtiene **C** y partiendo de la **C** del reflector **P = R = J** como salida final**.**

Y así para todas las letras.

Recordando la correspondencia numérica según el alfabeto utilizado:

Tabla 12. Equivalencia numeración letras

A	B	C	D	E	F	G	H	I	J	K	L	M
01	02	03	04	05	06	07	08	09	10	11	12	13

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
14	15	16	17	18	19	20	21	22	23	24	25	26

Supóngase que se teclea la combinación **CHK (3 - 8 - 11)**, su transmisión seria de la siguiente manera:

C ⟶ **U** ⟶ **Y** (En el disco reflector se devuelve por el camino de la Y).

L ⟶ **I** ⟶ **K** Obteniendo como salida la letra K.

El primer modificador da un veintiseisavo (1/26) de giro y se transmite la segunda letra:

H ⟶ **P** ⟶ **R** (En el disco reflector se devuelve por el camino de la R).

E ⟶ **W** ⟶ **B** Obteniendo como salida la letra B.

El primer modificador da un veintiseisavo (1/26) de giro y se transmite la tercera letra:

K ⟶ **B** ⟶ **N** (En el disco reflector se devuelve por el camino de la N).

A ⟶ **N** ⟶ **C** Obteniendo como salida la letra C.

Al final se obtiene **KBC** y esto es lo que se transmite, (Recuerde que la clave se enviaba dos veces, es decir, se tecleaba, CHKCHK).

Al teclear la clave se giran los discos a la posición indicada (3 - 8 - 11) y se continúa la transmisión del mensaje.

El receptor recibía **KBC…** al inicio del mensaje cifrado, lo ingresa a Enigma y obtiene **CHK,** es decir (3 - 8 - 11), giraba los modificadores a esta posición y continuaba con el descifrado del mensaje.

Esta norma de seguridad, junto a las anteriores, dio lo que necesitaban los resignados criptoanalistas para animarse a desentrañar a Enigma.

3.9.6. Una maquina aun más segura

Hasta ahora se ha descrito a Enigma como una máquina única, exactamente igual para todos los operadores, que la hacían funcionar de igual manera, como si fuera una superred de comunicaciones que abarcaba todo. Pero en la realidad, ni todo el mundo usaba la misma máquina, ni la operaba de igual manera, además que tanto la propia máquina como su uso fue variando a lo largo del tiempo.

Por ejemplo, el Afrika Korps (Fuerza militar alemana enviada al norte de África en 1941) mantenía sus propios métodos en su propia red. El frente oriental era distinto del frente occidental. Las máquinas eran distintas en algunos casos. Variaban el número de modificadores, el número de cables para conexiones en el clavijero, incluso en algunos casos el cableado interno de los modificadores.

Otro ejemplo fue que a los operadores se les proporcionó 5 modificadores para cambiar entre los tres huecos, 8 ó 10 cables para el tablero de clavijas, era configurable el número de giros que debía hacer un modificador para que girara el siguiente, usando un anillo para modificar la posición de la uña de arrastre (ya no era una norma fija los 26 giros que se habían definido inicialmente, para un giro del siguiente), en un momento dado se suprimió la repetición de la clave de mensaje y otras modificaciones.

Esto supuso un aumento considerable de las configuraciones posibles, ofreciendo muchas más claves, y amargando la vida a los criptoanalistas aliados. Señalar que la Enigma Naval, conocida en algunos lugares como “**Máquina N**” fue la más segura y la que más costo desentrañar. Esto fue así porque la Enigma Naval tuvo unas características físicas diferentes, como que incorporaba 4 modificadores, el reflector también era configurable (al igual que en un modificador, se le podía colocar en 26 posiciones), había más cables para el tablero de clavijas, se disponía de 8 modificadores para intercambiar y además, sus operadores la usaron de una manera más cuidadosa, eficiente y con una metodología propia.

3.10. OTRAS MAQUINAS DE CIFRADO

3.10.1. La maquina Hagelin

La máquina Hagelin fue inventada por el criptólogo sueco *Boris Hagelin*, quien adquirió en 1927 la fábrica de máquinas de cifrar de *Arvid G. Damm*, otro inventor sueco que no tuvo la suerte de sacar un producto competitivo en el mercado. Entre los años veinte y los treinta, Hagelin diseña diversas máquinas (B-21, B-211, C-36, C-48, etc.) en las que a través de ruedas con piñones realiza una cifra similar a la utilizada por el sistema de Beaufort ya visto al inicio de este capitulo.

La particularidad de estas máquinas que luego hicieron millonario a Hagelin, estaba en una periodicidad muy alta puesto que el número de dientes de las diferentes ruedas eran primos entre sí. Para seis ruedas estos valores eran 26, 25, 23, 21, 19 y 17, de forma que el período era igual a su producto, un valor que supera los 100 millones. La ecuación matemática que representa al cifrado de Hagelin es:

$Ci = Eki(M_j) = (k_i - M_j) \bmod n$

Donde:

K_i = Cada letra de la clave.

M_i = Cada letra del texto a cifrar.

n = Cantidad de letras del alfabeto utilizado.

E_{ki} (M_i) = Resultado de la resta de K_i menos M_i aplicando mod n.

C_i = Cada letra del mensaje cifrado.

3.10.2. La máquina japonesa: *"Purple"*

Los japoneses obtuvieron una Enigma de los alemanes y, fieles a sus costumbres, la adaptaron y modificaron para sus propios fines. Esta era similar al modelo alemán, pero con 4 cilindros. Los Estados Unidos se encontraron con un problema

similar al británico, al tener que romper las claves japonesas para así averiguar sus intenciones.

La costumbre japonesa de repetir la frase *"Tengo el honor de informar a su excelencia..."* al comienzo de cada comunicación, así como la transmisión de mensajes en un sistema ya conocido y en *"Purple"* facilitó en gran manera la construcción de una réplica operativa para 1940.

Gracias a ello, los americanos descifraron en Diciembre de 1941 un mensaje transmitido a la embajada japonesa en E.E.U.U: La declaración de guerra. Sin embargo, el que fuese un documento tan importante hizo necesario que lo transcribiesen oficiales de la embajada y que no eran expertos en ello. Esto provocó que la declaración de Guerra se produjese tras el ataque a Pearl Harbour y no antes.

El conocimiento de las claves japonesas durante toda la guerra fue una importante ventaja para E.E.U.U. Como ejemplo, la victoria de Midway fue en parte posible gracias a que los americanos conocían de antemano la intención de atacar el archipiélago.

3.10.3. Maquina de Lorenz

La Lorenz **SZ 40** y la **SZ 42** (*Schlüsselzusatz*, que significa "cifrado adjunto") eran máquinas alemanas de cifrado utilizadas durante la Segunda Guerra Mundial en circuitos de teletipo (Dispositivo telegráfico de transmisión de datos). Criptógrafos británicos, que se refirieron al tráfico alemán de datos de teletipo cifrados como "Fish", denominaron al aparato y su tráfico como "Tunny" (Atunes, Atún). Mientras la bien conocida Enigma fue usada generalmente por unidades de combate, la Máquina de Lorenz fue usada para comunicaciones de alto nivel. El ingenio en sí plasmado en una maquina, tenía las medidas de 51cm × 46cm × 46cm, y funcionó como dispositivo adjunto a las máquinas de cifrado de ese entonces, principalmente Enigma.

Esta máquina podría decirse que fue la que abrió camino a las primeras computadoras, las cuales fueron creadas con la finalidad de romper los mensajes de Enigma y Lorenz, unas de las más conocidas, las máquinas **Colossus** fueron los primeros dispositivos calculadores electrónicos usados por los británicos para leer las comunicaciones cifradas alemanas durante la Segunda Guerra Mundial.

Colossus fue uno de los primeros computadores digitales, las máquinas Colossus se usaron para descifrar los mensajes cifrados, que se interceptaban de las comunicaciones de la Alemania Nazi, usando la máquina Lorenz **SZ40 - SZ42**.

Criptógrafos británicos en Bletchley Park dedujeron la operación de la máquina en enero de 1942 sin haber visto nunca una máquina de Lorenz. Ese fue posible debido a un error cometido por un operador alemán. El 30 de agosto de 1941 un mensaje de 4000 letras fue transmitido, pero el mensaje no fue recibido correctamente del otro lado por lo cual el receptor envió un mensaje descifrado solicitando la retransmisión del mensaje original. Esto permitió a los "rompedores del código" saber que era lo que estaba ocurriendo. El mensaje fue retransmitido con la misma configuración de clave (HQIBPEXEZMUG) una práctica prohibida. Por otra parte, la segunda vez el operador realizó pequeñas alteraciones al mensaje, como utilizar abreviaciones. A partir de estos dos textos cifrados se logro recuperar ambos mensajes en texto plano y el código de cifrado. A partir del código de cifrado, la estructura completa de la máquina fue reconstruida por el matemático W. T. Tutte.

3.11. CRIPTOGRAFÍA CUÁNTICA

3.11.1. Introducción

La historia documenta diferentes métodos de protección que permiten ocultar mensajes secretos dentro de textos aparentemente banales (esteganografía), o bien que pasaran desapercibidos al enemigo (tintas invisibles o micropuntos en periódicos o libros) hasta llegar a la criptografía, que permite cifrar el contenido de un mensaje haciéndolo ilegible para todos, salvo el propietario de la clave de descifrado (la cifra de Cesar, por ejemplo). En general, la criptografía intenta resolver dos problemas que si bien son distintos, están ligados el uno con el otro.

El primer objetivo que aborda la criptografía es el cifrado de los mensajes enviados para ocultar la información que se desea transmitir. Ya en la antigua roma Julio Cesar utilizaba sistemas primitivos de cifrado para evitar que sus enemigos pudiesen enterarse de sus planes si capturaban a un mensajero.

Desde tiempos primitivos se han ensayado muchos métodos y protocolos, hasta llegar a los actuales (DES, AES, RSA etc.). Como un ejemplo simple de cifrado, se puede citar la “cinta aleatoria”, totalmente seguro e inviolable, que consiste en generar como clave una lista aleatoria de ceros y unos, que se suma al contenido del mensaje (previamente convertido al binario). El receptor tiene en su poder una cinta idéntica, que sirve de “llave” para leer el mensaje y deducir el texto original (simplemente, restando la clave al texto recibido). A pesar de ser tan elemental, este sistema es ciento por ciento inviolable, aunque se disponga del ordenador más potente y de todo el tiempo del mundo, siempre y cuando la cinta con la clave se utilice una sola vez (de hecho, este sistema en ingles se conoce como “one-time pad”).

Si se sigue investigando sobre el cifrado, es porque el sistema anterior a pesar de ser tan seguro y sencillo tiene alguna fisura, si no sería usado por todo el mundo.

En efecto, hay una: se debe tener un medio seguro de hacer llegar la clave al destinatario. Todos medios de comunicación (correo, teléfono, etc.) pueden ser interceptados, por lo que se tienen los mismos problemas que al principio. Si se dispusiera de un canal seguro para hacer llegar la clave, bien se podría haber usado para enviar directamente el mensaje sin cifrar, y fin del problema.

En definitiva, no existe forma segura de poner en conocimiento del destinatario el valor de la cinta, es decir, de la clave. Por este motivo, se han desarrollado protocolos y algoritmos que permitan compartir secretos a través de canales públicos. Actualmente el algoritmo más usado se basa en criptografía de clave pública (el famoso RSA).

Este tipo de algoritmos basan su funcionamiento en la utilización de un juego de dos claves, una pública, conocida por todos y que se utiliza para cifrar la información a transmitir, y otra privada, que solo es conocida por su propietario, con la que se descifra el mensaje para recuperar la información secreta.

Esto es posible gracias a ciertas funciones matemáticas que tienen la particularidad de ser muy sencillas en un sentido, pero sumamente complicadas en el contrario. Se puede entender mejor con un ejemplo: es muy simple elevar un número al cuadrado, por ejemplo 5. Pero resulta muy complicado extraer mentalmente la raíz cuadrada de 5. RSA basa su fortaleza en la gran dificultad que representa el obtener los factores primos que componen un número “grande” (cientos de dígitos). Utilizando los ordenadores más potentes, se tardarían millones de años en factorizar una clave de este tipo.

3.11.2. Entrada a los estados cuánticos

Así como un ordenador tradicional utiliza la presencia o ausencia de corriente eléctrica para representar dos estados (cero o uno), un ordenador cuántico se basa en las propiedades cuánticas de la materia para almacenar información sirviéndose, por ejemplo, de dos estados diferentes de un átomo o de dos polarizaciones distintas de un fotón. La gran diferencia es que, debido a principios de la física cuántica, además de los dos estados que representan un 1 y un 0, respectivamente, el átomo puede encontrarse en una superposición de ambos, esto es, se encuentra en estado 0 y 1 a la vez. De esta forma, un sistema cuántico de dos estados (llamado qubit) se encuentra en una superposición de los dos estados lógicos 0 y 1. Por lo tanto, un qubit sirve para codificar un 0, un 1 ó ¡0 y 1 al mismo tiempo! Si se dispone de varios qubits se podrían codificar simultáneamente cantidades de información impresionantes.

Estas características, además de ofrecer grandes capacidades de almacenamiento y velocidades de proceso, permiten el abordar problemas clásicos que son irresolubles en la práctica usando métodos tradicionales de una forma totalmente nueva. De hecho, a mediados de los noventas los laboratorios de AT&T Bell invento un par de algoritmos ejecutables en ordenadores cuánticos, uno de ellos capaz de factorizar números grandes en un tiempo insignificante frente a los ordenadores clásicos, y otro que puede buscar en una lista a velocidades muy superiores a las actuales.

Estas dos soluciones vienen a ser la herramienta capaz de demoler el andamiaje sobre el que se ha montado el RSA, dado que si se lograra construir un ordenador cuántico capaz de ejecutar estos algoritmos, ya no harían falta millones de años de procesamiento para quebrar una clave, por lo que habría que buscar métodos diferentes de cifrado. Piénsese que el algoritmo cuántico de factorización permite romper DES o cualquier otro algoritmo de cifrado de clave secreta en un tiempo que es raíz cuadrada del que se tardaría con un ordenador clásico. En otras palabras, todos los secretos guardados con claves de hasta 64 bits, hoy en día consideradas invulnerables, dejarían de serlo. Por ejemplo, si para atacar el DES por fuerza bruta hay que probar 256 claves con un ordenador convencional, el algoritmo de la AT&T lo haría en solo 16 intentos. Afortunadamente, todavía no se ha construido un ordenador de este tipo, ni se espera hacerlo en las próximas dos o tres décadas.

Heisenberg postulo en el siglo pasado que es imposible conocer de manera simultánea dos variables complementarias de una partícula, como son su posición y velocidad. Dicho postulado se conoce como “Principio de Incertidumbre”. Antes de seguir, también se debe recordar que la luz (o los fotones que la componen) pueden polarizarse en una de cuatro direcciones posibles: vertical (que se representará con “|”), horizontal (“—“), diagonal a la izquierda (“\”) o diagonal a la derecha (“/”). Estas cuatro polarizaciones forman dos bases perpendiculares entre sí: | y --, y / y \, respectivamente.

El principio de incertidumbre de Heisenberg impide que se pueda saber en cuál de las cuatro posibles polarizaciones se encuentra un fotón. Para conocerla, se debería utilizar un filtro, que se podría imaginar como una ranura en una lámina, que tuviera la orientación de la polarización que se quisiese comprobar, por ejemplo, vertical (|). Es evidente que los fotones con la misma polarización pasarán, mientras que los polarizados horizontalmente, y por lo tanto, perpendiculares al filtro, no pasarán. Sorprendentemente, y debido a principios físicos bien demostrados ¡la mitad de los polarizados diagonalmente pasarán y serán reorientados verticalmente! Por lo tanto, si se envía un fotón y pasa a través del filtro, no puede saberse a ciencia cierta si poseía polarización vertical o diagonal, tanto \ como /. Igualmente, si no pasa, no puede afirmarse que estuviera polarizado horizontal o diagonalmente. En ambos casos, un fotón con polarización diagonal podría pasar o no con igual probabilidad.

Ahora bien, ¿cómo puede servir de este fenómeno cuántico para encriptar información confidencial? Al igual que en el caso de la “cinta aleatoria” que se veía al principio de la nota, una de las formas de polarización de los fotones enviados representa el “0” y la otra el “1”. De esta manera, en la base rectangular, que se llamará (+), un 1 vendría representado por una polarización |, mientras que un 0, por --; mientras que en la base diagonal (x), el 1 sería la polarización / y el 0, \. En estas condiciones, para enviar un mensaje binario, el espía “A” envía fotones a “B” con la polarización adecuada, cambiando aleatoriamente de una base a otra. Ahora supóngase que un espía “E” intercepta la comunicación entre “A” y “B”, utilizando un filtro cualquiera (por ejemplo, |) y mide su polarización. Debido al principio de incertidumbre mencionado, jamás podrá saber si los fotones del mensaje pertenecían a la base ortogonal “+” o “x”. La comunicación es segura. Pero evidentemente, “B” debería tener la misma dificultad que “E” para poder descifrar el mensaje, dado que las leyes físicas son las mismas para ambos.

Por supuesto, se supone que “A” y “B” no han acordado de antemano que bases utilizaran para enviar cada fotón, dado que si dispusiesen de una forma segura de

hacerse llegar esa información, la utilizarían para enviar el mensaje y no necesitarían de la criptografía. Tampoco pueden utilizar otro sistema de cifrado como RSA, ya que como se vio, la criptografía cuántica lo hace vulnerable.

3.11.3. Una Solución

La solución a este dilema fue propuesta en 1984 por Charles Bennet y Gilles Brassard, quienes idearon el siguiente método para hacer llegar el mensaje al destinatario sin necesidad de recurrir a otros canales de distribución. En esencia, consta de tres pasos:

Primero: “A” le manda a “B” una cadena de “0” y “1” totalmente aleatoria, utilizando una elección entre bases + y x también al azar.

Segundo: “B” mide la polarización de esos fotones mediante un filtro, utilizando también de forma totalmente al azar las bases + y x. Dado que desconoce que bases uso “A” para codificar el mensaje, en promedio 1 de cada 4 bits que recibe “B” será erróneo.

Tercero: “A” se comunica con “B”, utilizando para ello cualquier canal disponible, aunque sea completamente inseguro (y que probablemente “E” esta escuchando) y le dice que base de polarización utilizo para enviar cada fotón (+ o x) aunque no le dice que polarización concreta (|, --, \, /). “B” le responde con una lista de los casos en que acertó con la polarización correcta (y por lo tanto recibió el 0 o el 1 sin error). Luego, ambos eliminan de la lista los bits que se recibieron con las bases erróneas, con lo que obtienen una secuencia aleatoria más corta que la original, pero que es la misma para ambos. Esta secuencia constituye una “cinta aleatoria” 100% segura.

Véase ahora que paso con el intruso “E”, si quiere interceptar el mensaje transigido entre “A” y “B”. Si ha tenido la suerte de escuchar el mensaje, no ha podido obtener nada en concreto de el, ya que ignora las polarizaciones concretas que utilizo “A” para codificar el mensaje (no se olvide que “A” le dice a “B” que bases utilizo, pero no las polarizaciones empleadas). Además, la sola presencia de “E” en la línea será detectada, dado que si mide la polarización de un fotón con el filtro adecuado, lo alterara. Si bien esta alteración impide que “A” y “B” se comuniquen, ya que este cambio puede hacer que se obtengan distintos bits a pesar de utilizar las mismas bases ortogonales.

Es decir, hace falta un método para detectar si “E” está presente en la línea. Y es tan sencillo como que “B” le cuente a “A” utilizando cualquier canal cuáles son los primeros, 50 bits (por ejemplo) de su clave aleatoria. Si coinciden con los de “A”, entonces saben que “E” no les espió, ni el canal tiene ruido y utilizan con seguridad el resto de los bits generados. Si no coinciden, ya saben que “E” está presente, o utilizaron un canal muy ruidoso y por lo tanto deben desechar la clave entera.

En los años transcurridos desde 1984 no se han producido demasiados avances teóricos en el ámbito de la criptografía cuántica. Pero si se han logrado avances en la implementación practica de un sistema que aproveche dichos postulados.

El principal obstáculo fue al principio la manipulación de fotones individuales. Cualquier fuente de luz (sol, láser, LED) emite enormes cantidades de ellos, por lo que emitir y detectar fotones individuales fue un desafió que se pudo resolver recién en 1989, cuando el mismo Bennet consiguió la primer transmisión de señales cuánticas a una distancia de 32 centímetros. Si bien la distancia fue pequeña, implico la resolución de innumerables problemas prácticos.

En 1995, investigadores de la Universidad de Ginebra utilizando un canal de fibra óptica lograron el envió de fotones individuales a una distancia de 23 kilómetros. Luego, el laboratorio de Los Álamos logro superar los 50 kilómetros. Se han logrado enlaces aéreos (sin soporte físico) de hasta 1,6 kilómetros. Queda mucho camino por recorrer, pero la transmisión de claves mediante este sistema es una opción perfectamente viable.

Si bien existen obstáculos por superar, algunos importantes como lograr formas seguras de saber que “A” realmente se está comunicando con “B” y no con “E”, que pasa si se interrumpe la transmisión, etc., se ha demostrado teóricamente que la transmisión cuántica de información es inviolable.

Como escribe Singh en su excelente libro "The Code Book", si los ingenieros consiguen hacer funcionar la criptografía cuántica a través de largas distancias, la evolución de los algoritmos de cifrado se detendrá. La búsqueda de la privacidad habrá llegado a su fin. La tecnología garantizaría las comunicaciones seguras para gobiernos, militares, empresas y particulares. La única cuestión pendiente sería si los gobiernos permitirían a los ciudadanos usarla. ¿Cómo regularán los Estados la criptografía cuántica, enriqueciendo la Era de la Información, pero sin proteger al mismo tiempo las actividades criminales?

Seguramente no será mañana, pero se puede contar que en un futuro no muy lejano las comunicaciones entre personas serán totalmente inviolables, haciendo realidad el sueño de Cesar.

3.11.4. Algoritmo BB84 Para Criptografía Cuántica

El esquema de codificación BB84 fue el primer codificador cuántico de información clásica en ser propuesto de forma tal que el receptor, legitimo o ilegitimo, pueda recuperar con 100% de confiabilidad. Esta es la base sobre la cual están fundados la mayoría de los protocolos cuánticos.

1) La fuente de luz, generalmente un LED (Light emitting diode) o láser, es filtrada para producir un rayo polarizado en ráfagas cortas y con muy baja intensidad. La polarización en cada ráfaga es entonces modulado por el emisor “A” de forma aleatoria en uno de los cuatro estados (horizontal, vertical, circular-izquierdo o circular-derecho).

2) El receptor, “B”, mide las polarizaciones de los fotones en una secuencia de bases aleatoria (rectilíneo o circular).

3) “B” le dice públicamente al emisor que secuencia de bases utilizo.

4) “A” le dice al receptor públicamente cuales bases fueron elegidas correctamente.

5) “A” y “B” descartan todas las observaciones en las que no se eligió? la base correcta.

6) Las observaciones son interpretadas usando un esquema binario por ejemplo: horizontal o circular-izquierdo es 0, vertical o circular-derecho es 1.

Este protocolo se complica con la presencia de ruido, el que puede ocurrir en forma aleatoria o ser introducido por una escucha. Con la existencia de ruido las polarizaciones observadas por el receptor pueden no coincidir con las emitidas por el emisor. Para lidiar con esta posibilidad, “A” y “B” deben asegurarse que poseen la misma cadena de bits. Esto se realiza usando una búsqueda binaria con verificación de paridad para aislar las diferencias. Con el descarte del último bit de cada comparación, la discusión pública de la paridad se vuelve inofensiva. En el protocolo de Bennett de 1991 este proceso es:

1) “A” y “B” acuerdan una permutación aleatoria de las posiciones de los bits en sus cadenas, para distribuir aleatoriamente la posición de los errores.

2) Las cadenas se parten en bloques de longitud k, con k elegido de forma tal que la probabilidad de múltiples errores por bloque sea muy baja.

3) Por cada bloque, “A” y “B” computan y anuncian públicamente las paridades. Luego el último bit de cada bloque es descartado.

4) Para cada bloque en el que difirieron las paridades calculadas, “A” y “B” usan una búsqueda binaria con log(k) iteraciones para localizar y corregir el error en el bloque.

5) Para contemplar múltiples errores que aún no han sido detectados, los pasos 1 al 4 son repetidos con tamaños de bloque cada vez más grandes.

6) Para determinar si aún quedan errores, “A” y “B” repiten un chequeo aleatorio:

   a. “A” y “B” acuerdan públicamente una muestra de la mitad de las posiciones en sus cadenas de bits.

   b. Públicamente comparan las paridades y descartan un bit. Si las cadenas difieren, las paridades van a discrepar con probabilidad ?.

   c. Si hay discrepancias, “A” y “B” utilizan una búsqueda binaria para encontrarlas y eliminarlas.

7) Si no hay desacuerdos después de “l” iteraciones, se concluye que sus cadenas coinciden con una probabilidad de error de 2-l.

3.11.5. 2010 Criptografía cuántica, crackeada

La criptografía cuántica sigue siendo imposible de crackear en teoría, el problema es la implementación.

Feihu Xu, Bing Qi y Hoi-Kwong Lo de la Universidad de Toronto dicen haber craqueado uno de los sistemas criptográficos comerciales.

El tema es así: las pruebas que dicen que la criptografía cuántica es imposible de crackear se basan en implementaciones perfectas que en la realidad no es posible

realizar, siempre habrá *ruido* que introducirá errores. Por ello es que los dispositivos de criptografía deben tolerar cierto ruido. Varias pruebas dicen que si el error se mantiene por debajo del 20%, el mensaje sigue siendo seguro. Sin embargo esas pruebas sólo consideran el ruido ambiente, no consideran errores introducidos en la preparación del estado inicial (el que envía el emisor) y allí es donde Xu, Qi y Lo hicieron su truco. Ellos afirman que debido a ese error extra un tercero puede interceptar algunos bits cuánticos, leerlos y volverlos a enviar de una manera que incrementa el error en sólo 19,7%, lo cual será tolerado ya que está por debajo del 20% permitido.

Este problema es fácil de resolver, es cuestión de considerar también el error introducido por el emisor en el cálculo de tolerancia. Pero el problema fue destapado: de la teoría a la práctica habrá siempre cosas que asumir y en esas asunciones es donde está la debilidad.

3.11.6. Conclusión

El método criptográfico clásico One Time Pad es 100% seguro, es imposible de crackear, el problema es que para utilizarlo y que sea seguro, debe haber algún modo de distribuir las claves a utilizar, las cuales deben ser del mismo largo que el mensaje (para 100% de seguridad). Para hacer eso existen métodos de distribución de claves cuánticos que son 100% seguros, imposibles de crackear, el problema es que para implementarlo debe haber algún modo de realizar un dispositivo sin errores. Para hacer eso se utilizan técnicas de corrección de errores, y se demuestra que se puede tolerar hasta un 20% de error ambiente y sigue siendo 100% seguro, el problema son los otros errores que no se tienen en cuenta. Ahí es donde entra el crack.

4. CRIPTOANALISIS

4.1. INTRODUCCION

El criptoanálisis es la ciencia opuesta a la criptografía (quizás no es muy afortunado hablar de ciencias opuestas, sino más bien de ciencias complementarias), ya que si ésta trata principalmente de crear y analizar criptosistemas seguros, la primera intenta romper esos sistemas, demostrando su vulnerabilidad: dicho de otra forma, trata de descifrar los criptogramas. El término descifrar siempre va acompañado de discusiones de carácter técnico, aunque se asumirá que descifrar es conseguir el texto en claro a partir de un criptograma, sin entrar en polémicas de reversibilidad y solidez de criptosistemas.

En el análisis para establecer las posibles debilidades de un sistema de cifrado, se han de asumir las denominadas condiciones del peor caso:

- El criptoanalista tiene acceso completo al algoritmo de encriptación.
- El criptoanalista tiene una cantidad considerable de texto cifrado.
- El criptoanalista conoce el texto en claro de parte de ese texto cifrado.

También se asume generalmente el Principio de Kerckhoffs, que establece que la seguridad del cifrado ha de residir exclusivamente en el secreto de la clave, y no en el mecanismo de cifrado.

Aunque para validar la robustez de un criptosistema normalmente se suponen todas las condiciones del peor caso, existen ataques más específicos, en los que no se cumplen todas estas condiciones. Cuando el método de ataque consiste simplemente en probar todas y cada una de las posibles claves del espacio de claves hasta encontrar la correcta, se encuentra un ataque de fuerza bruta o ataque exhaustivo. Si el atacante conoce el algoritmo de cifrado y sólo tiene acceso al criptograma, se plantea un ataque sólo al criptograma; un caso más favorable para el criptoanalista se produce cuando el ataque cumple todas las condiciones del peor caso; en este caso, el criptoanálisis se denomina de texto en claro conocido. Si además el atacante puede cifrar una cantidad indeterminada de texto en claro al ataque se le denomina de texto en claro escogido; este es el caso habitual de los ataques contra el sistema de verificación de usuarios utilizado por Unix, donde un intruso consigue la tabla de contraseñas (generalmente /etc/passwd) y se limita a realizar cifrados de textos en claro de su elección y a

comparar los resultados con las claves cifradas (a este ataque también se le llama de diccionario, debido a que el atacante suele utilizar un fichero `diccionario' con los textos en claro que va a utilizar). El caso más favorable para un analista se produce cuando puede obtener el texto en claro correspondiente a criptogramas de su elección; en este caso el ataque se denomina de texto cifrado escogido.

Cualquier algoritmo de cifrado, para ser considerado seguro, ha de soportar todos estos ataques y otros no citados; sin embargo, en la criptografía, como en cualquier aspecto de la seguridad, informática o no, no se debe olvidar un factor muy importante: las personas. El sistema más robusto caerá fácilmente si se tortura al emisor o al receptor hasta que desvelen el contenido del mensaje, o si se le ofrece a uno de ellos una gran cantidad de dinero; este tipo de ataques (sobornos, amenazas, extorsión, tortura...) se consideran siempre los más efectivos.

4.2. CRIPTOANÁLISIS DEL GENERADOR AUTO-SHRINKING: UNA PROPUESTA PRÁCTICA

La criptografía de clave secreta se divide en dos grandes apartados: procedimientos de cifrado en flujo y procedimientos de cifrado en bloque. Los sistemas de cifrado en flujo son los más rápidos dentro de los métodos de cifrado, de ahí que en la actualidad se utilicen en numerosas aplicaciones prácticas.

Prueba de ello son, por ejemplo, los algoritmos A5 (en su doble versión A5/1 y A5/2) que se utilizan en telefonía móvil GSM, el algoritmo E0 usado en especificaciones de Bluetooth ó el algoritmo RC4 utilizado en Microsoft Word y Excel.

Un sistema de cifrado en flujo está compuesto por un algoritmo o generador de secuencia cifrante (conocido públicamente) y una clave de cifrado (conocida únicamente por los dos comunicantes). La clave corresponde normalmente al estado inicial del generador de secuencia. En el momento de su inicialización el generador toma la clave como semilla, posteriormente se ejecuta el algoritmo mediante un clock a la velocidad que necesite la aplicación y así se genera la secuencia cifrante.

Para cifrar, el emisor realiza una operación XOR, bit a bit, entre la secuencia cifrante y el texto claro, dando origen al texto cifrado que es el que se va a enviar

por el canal de información. Para descifrar, el receptor genera la misma secuencia cifrante que suma bit a bit con el texto cifrado recibido y recupera así el texto claro original.

Muchos algoritmos de cifrado en flujo están basados en Registros de Desplazamiento con Realimentación Lineal, en inglés Linear Feedback Shift Registers (LFSRs), cuyas secuencias de salida, las *PN*-secuencias, se combinan entre sí mediante algún procedimiento o función no lineal. Entre los generadores de secuencias pseudoaleatorias más conocidos y mejor estudiados con aplicaciones criptográficas se puede señalar: generadores combinacionales, filtros no lineales, generadores controlados por uno o varios relojes, generadores multi-velocidad, generadores con decimación irregular, etc. Todas estas estructuras producen a su salida secuencias cifrantes con una alta complejidad lineal, períodos muy largos y buenas propiedades estadísticas.

Los procedimientos de cifrado de flujo se utilizan para dar seguridad criptográfica a sistemas de comunicaciones con requerimientos de velocidad y sincronismo. Uno de estos procedimientos de cifrado de amplia difusión es el generador auto-shrinking. La Unión Europea, a través del Proyecto Stork, propuso a la comunidad científica internacional romper este generador mediante alguna técnica criptoanalítica que mejorase el ataque TMTO.

Siguiendo esta premisa, se ha emprendido el criptoanálisis de este generador mediante variaciones a la técnica del Guess and Determine. Se ha llevado a cabo una personalización de dicha técnica, mediante lo que se consideran suposiciones bien definidas para romper este generador de secuencia cifrante.

Este trabajo está dividido en secciones y organizado de la siguiente manera. En la sección 2, se describe el generador auto-shrinking con sus correspondientes características y propiedades de pseudoaleatoriedad. La sección 3 está dedicada tanto a la técnica de criptoanálisis propuesta como a los correspondientes resultados de implementación. En la sección 4, el ataque desarrollado se compara con otras técnicas criptoanalíticas en uso, mientras que en la sección 5 se presentan las conclusiones y futuros trabajos a realizar.

4.3. GENERADOR AUTO-SHRINKING

El generador auto-shrinking fue diseñado por Meier y Staffelbach para su uso en

aplicaciones criptográficas. Este generador es de fácil implementación y consiste en un único LFSR de L etapas y polinomio de realimentación primitivo de grado L.

Se decima de forma irregular dando origen a la secuencia *auto-shrunken*, notada $\{z_n\}$, o secuencia de salida del generador. La regla de decimación es xtremadamente simple, se consideran pares (S_{2i}, S_{2i+1}) $(i = 0, 1, 2, \ldots)$ de bits consecutivos de $\{s_n\}$ tales que:

• Si $S_{2i} = 1$, entonces $Z_j = S_{2i+1}$.

• Si $S_{2i} = 0$, entonces S_{2i+1} se descarta.

Es decir, si el primer bit del par considerado es un 1, entonces el segundo bit se inserta en la secuencia de salida. Por el contrario, si el primer bit del par considerado es un 0, entonces el segundo bit se rechaza. De esta manera, se van eliminando determinados bits de la secuencia $\{S_n\}$ y los que quedan constituyen la secuencia $\{Z_n\}$ o secuencia auto-shrunken. La clave de este generador es el estado inicial del LFSR, aunque los autores recomiendan que el polinomio de realimentación forme también parte de la misma. De acuerdo con, el período, la complejidad lineal y las propiedades estadísticas de la secuencia $\{Z_n\}$ son muy adecuados para su aplicación en criptografía.

Asimismo y según, el polinomio característico $P(x)$ de la secuencia auto-shrunken $\{z_n\}$ viene dado por:

$$P(x) = (x + 1)^p, \qquad (1)$$

Donde el valor de p está acotado por:

$$2^{L-2} < p < 2^{L-1}. \qquad (2)$$

Esto implica una relación de recurrencia lineal de la forma:

$$(E + 1)^p z_n = 0, \qquad (3)$$

Donde E es el operador desplazamiento que actúa sobre los términos de la secuencia $\{z_n\}$, es decir:

$$E Z_n = Z_{n+1}. \qquad (4)$$

La ecuación (3) representa una ecuación en diferencias lineal con coeficientes binarios y constantes cuyo polinomio característico $P(x)$ dado por la ecuación (1) tiene una única raíz $\lambda = 1$ con multiplicidad p.

El período de esta secuencia auto-shrunken $\{z_n\}$ se denota por T y viene dado por:

$$T = 2^{L-1}. \quad (5)$$

La secuencia $\{s_n\}$, generada por el LFSR, puede considerarse formada por dos secuencias diferentes $\{c_n\}$ y $\{b_n\}$ que responden a los bits de subíndices pares e impares, respectivamente, de la secuencia $\{s_n\}$. Es decir:

$$c_i = S_{2i} \; \forall \; i \geq 0 \quad (6)$$

$$b_i = S_{2i+1} \; \forall \; i \geq 0 \quad (7)$$

A su vez, estas dos secuencias corresponden a la misma *PN*-secuencia $\{S_n\}$ generada por el LFSR pero desplazadas entre sí una distancia de 2^{L-1} bits. La existencia de dicho desplazamiento permite expresar una secuencia en función de la otra. Así se ve que:

$$bi = \sum_{j=0}^{L-1} \mathrm{h(x)Cj} \; (0 \leq \mathrm{i} \; \leq \mathrm{L} - 1) \quad (8)$$

Donde el polinomio $h(x)$ sirve para expresar unos coeficientes en función de los otros. A partir de esta relación, se puede armar una matriz de coeficientes H, que es la que usaremos para resolver el sistema de ecuaciones.

Esta matriz de coeficientes se construye para los sucesivos valores de bi expresados en función de los coeficientes Ci específicos.

$$H = \begin{bmatrix} b0 \\ b1 \\ b2 \\ b3 \\ . \\ . \\ . \\ bL-1 \end{bmatrix} \quad (9)$$

4.3.1. Ataque propuesto

El ataque consiste en generar o crear una serie de suposiciones para los $L/2+1$ bits iníciales de C_i y, a partir de ellos, determinar los restantes bits de C_i a la vez que los L bits de b_i. Juntos forman la condición inicial de la secuencia $\{S_n\}$. Romper

el generador es equivalente a encontrar la condición inicial de la secuencia $\{S_n\}$ o, en su defecto, un estado inicial que continúe la secuencia lógica de bits $\{S_n\}$.

Para ello, se arman bloques de B bits de secuencia interceptada (suposiciones) y, para cada bloque, se comprueba si se rompe o no el criptosistema. En efecto, se va desplazando dentro de estos B bits haciendo las comprobaciones pertinentes hasta encontrar la solución correcta.

Dado que es posible expresar los coeficientes de acuerdo con las ecuaciones (8) y (9), tenemos la posibilidad de plantear un sistema de ecuaciones lineales. Entonces y a partir de esta matriz H y del polinomio característico del LFSR, se plantea el sistema de ecuaciones lineales con coeficientes binarios que habrá que resolver. No todas las suposiciones son aptas para la resolución del sistema, por lo que hay que seleccionarlas adecuadamente. Esta selección de suposiciones es lo que hace del criptoanálisis, un ataque efectivo y más eficaz que el que utiliza la técnica de Guess and Determine.

Se define a continuación los parámetros que caracterizan el ataque criptoanalítico:

- T_s: Tiempo que se tarda en obtener todas las suposiciones posibles o las que se usaran en el ataque. Este parámetro no influye en el tiempo de procesado del criptoanálisis y depende del generador auto-shrinking a criptoanalizar (longitud y polinomio característico del LFSR).
- *Cantidad de Suposiciones*: Número de suposiciones. Para cada suposición, se conocen $L/2+1$ bits de secuencia $\{c_n\}$, de un total de L bits. El ataque consiste en probar cada una de estas suposiciones hasta romper el sistema. Es decir, hasta calcular el resto de los L-$L/2+1$ bits de $\{c_n\}$ y los L bits de $\{b_n\}$ con el fin de poder seguir generando bits de la secuencia $\{z_n\}$.
- N: Número de bloques en los que se divide el período de secuencia de salida $\{z_n\}$ o secuencia auto-shrunken.
- B: Longitud del bloque en el que se divide la secuencia $\{z_n\}$, es decir la cantidad de bits de secuencia interceptada que se necesita para romper el generador.
- *Porcentaje de aciertos*: Número total de aciertos sobre el total de bloques estudiados de longitud B bits por bloque.
- *ECU*: Cantidad de bits supuestamente conocidos de c. Este valor es igual a $L/2+1$ para todo polinomio de L etapas a considerar.

La idea básica consiste en analizar, para valores pequeños de L, el número de aciertos sobre todo un período de la secuencia de salida, extrapolando luego estos resultados a generadores auto-shrinking en un rango de aplicación criptográfica. Este análisis intensivo puede llevarse a efecto hasta valores de $L \approx 20$, con longitudes mayores el período de la secuencia generada se hace relativamente grande.

Las pruebas hasta el momento se han realizado en Matlab, simulando diferentes generadores auto-shrinking con registros de desplazamiento de longitud L, donde L varía en el intervalo 10 - 20 etapas. A su vez, para cada una de estas longitudes se han simulado LFSRs con diferentes polinomios de realimentación.

4.3.2. Método propuesto

Para realizar este ataque habrá que plantear, a partir de una suposición y una cantidad de secuencia interceptada, un sistema booleano de ecuaciones lineales y, posteriormente, resolverlo. La resolución se lleva a cabo aplicando sencillamente el método de Gauss. Claramente, el sistema planteado no siempre va a tener solución. A su vez hay que hacer notar que, en cada suposición considerada, sólo los bits c_i que toman el valor 1 aportan una ecuación al sistema. Por el contrario, los bits c_i con valor 0 no añaden nuevas ecuaciones.

A partir del programa desarrollado también es posible expresar los coeficientes c_i en función de los b_i y viceversa. Esta doble vía nos permite calcular una mayor cantidad de incógnitas y no sólo obtenerlas a partir de la resolución del sistema.

4.3.3. Tiempo, Memoria y Serie Cifrante

El tiempo de procesamiento máximo del ataque criptoanalítico es el que tarda el programa en procesar todas las suposiciones sobre todos los bits de cada bloque B. Ahora se exponen algunos resultados numéricos.

Para generadores con $L = 20$, bloques de secuencia interceptada de valor $B = 120$ bits y *Cantidad de Suposiciones* = 34, se ha alcanzado un éxito de rotura del criptosistema del 82%. Por tanto, la cantidad de bits necesarios para lograr este porcentaje de éxitos será del orden de $O(2^{0.33*L})$. Dicho orden podría disminuir si

se agregara alguna suposición adicional. A partir de los ejemplos programados puede también apreciarse que, con una cantidad de secuencia interceptada entre 100 - 200 bits, se obtiene un éxito de rotura del 93 %. Luego, para *L* = 20, la cantidad promedio de bits necesarios para romper el generador estaría en torno a los 150 bits.

Es importante destacar también que, para bloques de *B* = 120 bits, aunque haya un 18% de casos que no puedan resolverse sólo el 2% de los mismos son consecutivos. Luego, si bien hay ya garantizado un 82% de éxitos, se tendrá un 98% de éxitos al analizar el bloque siguiente.

Cabe la misma disquisición para bloques de *B* = 100 bits, donde sólo el 6% de los bloques no resueltos son consecutivos, con lo cual si bien existe ya un 75% de éxitos, se tendrá un 94% de éxitos de rotura con el bloque siguiente.

El tiempo de procesamiento hay que computarlo considerando el número de suposiciones que se utilizan en el ataque y la cantidad de bits que componen el bloque de secuencia interceptada. Se tendría así el número total de intentos para romper el criptosistema.

Hay que tener presente que solo son necesarios *L* bits de secuencia interceptada para plantear el sistema de ecuaciones y romper el generador. El problema es que no sería realista considerar que el ataque comienza precisamente en el bit de secuencia interceptada a partir del cual el sistema es capaz de romper el generador. Por ello, hay que tomar *B* bits de secuencia interceptada y desplazarse dentro del bloque hasta encontrar la solución correcta o estado inicial del LFSR.

Los tiempos de procesamiento deben ser aún evaluados convenientemente. De todas maneras, la forma en que están desarrollados los programas permite un procesamiento en paralelo, lo cual también disminuirá drásticamente los tiempos de procesamiento.

El ataque en paralelo supondría, entre otras cosas, tener una CPU por cada suposición. De este modo el tiempo de procesamiento disminuiría en un orden que sería directamente proporcional al número de suposiciones. Otro ejemplo de paralelismo estaría dado si se consideraran tantas CPU´s como bits hay en el bloque de secuencia interceptada. En este caso, el tiempo de procesamiento estaría dado por el tiempo que tarda el programa en procesar los *L* bits de secuencia interceptada que necesita para resolver el sistema de ecuaciones. La cantidad de memoria en uso es muy baja, y responde a la cantidad de secuencia

interceptada necesaria para plantear el sistema de ecuaciones y a la tabla de suposiciones que se tiene que tener pre-computada.

4.3.4. Pre-Procesamiento

No es necesario para algunos criptoanálisis, realizar un pre-procesamiento con tiempos elevados. Los datos a pre-procesar son las suposiciones que se van a necesitar tener calculadas antes de correr el programa.

De hecho es relativamente fácil encontrar estos supuestos que luego se utilizan para la determinación de los coeficientes c_i y b_i $(0 \leq i \leq L\text{-}1)$.

Calcular las suposiciones, en realidad, implica recorrer el universo de posibles soluciones al sistema de ecuaciones. Como se ha dicho anteriormente, existen suposiciones con un número X de 0´s en los bits supuestos de c_i, lo que supone ($ECU\text{-}X$) ecuaciones válidas para resolver. A su vez, de estas ecuaciones no todas permitirán obtener una solución. Por otro lado, el lugar donde se encuentren los 0´s dentro de la suposición también será un factor a considerar en la resolución del sistema.

En resumen, hallar las suposiciones correctas implica hallar la mayor cantidad posible de ecuaciones que resuelvan el sistema.

El método práctico utilizado en este criptoanálisis, para hallar las suposiciones, consiste en tomar en principio aquellas que tienen solo 1 cero y corroborar que se resuelve el sistema de ecuaciones. Luego en lugar de seleccionar esas suposiciones , se consideran las que se encuentran a continuación, en general tienen más de un cero, aumentándose la cantidad de ceros, en función del grado L, del polinomio en cuestión.

Debe ser tenida en cuenta la resolución del sistema de ecuaciones. En muchos casos un cero en determinado bit de la suposición, genera que el sistema no pueda ser resuelto. Dichos bits deben ser analizados, para el posterior descarte de la suposición, de ser necesario, en cada generador Auto-Shrinking.

4.3.5. Programas

Todos los programas están realizados en Matlab. Consisten en un programa o función principal que requiere de otros programas o funciones secundarias. Dichas funciones son llamadas para realizar tareas específicas tales como:

- Separar en bloques de longitud estipulada los bits que componen la serie cifrante y que se consideran los datos de entrada.
- Introducir como datos todos los supuestos. Éstos son valores del estado inicial de la secuencia $\{c_n\}$. Dicha secuencia se genera a partir del LFSR del generador auto-shrinking.
- Plantear el sistema de ecuaciones necesario a partir de la secuencia cifrante correspondiente. El sistema de ecuaciones se plantea ya que es conocida la expresión de los coeficientes $b_i = F(c_i)$.
- Resolver el sistema de ecuaciones mediante método de Gauss. Es posible que con los datos propuestos no se pueda resolver, en ese caso habrá que considerar otra suposición. Si se logra resolver el sistema, entonces como salida produce la condición inicial de $\{c_n\}$ de L bits.
- Verificar si cada uno de los datos c_i y b_i hallados son correctos. Esta comprobación se lleva a cabo generando a partir de los resultados obtenidos más secuencia de salida y comparándola con la secuencia interceptada.

Véase la siguiente secuencia de tareas que están programadas. Todos los programas corren en Matlab.

1) Armar el bloque de datos de longitud fija

2) Introducir todos los supuestos

3) Desplazar dentro de los bits de secuencia cifrante

4) Plantear el sistema de ecuaciones

5) Completar término independiente con bit correspondiente de secuencia cifrante

6) Resolver el sistema de ecuaciones

7) Hallar *L* valores de c y b_{ii}

8) SI

El generador está roto

ENTONCES

El sistema está roto

9) SI NO

Volver al paso 3).

Existen otras funciones o tareas llevadas a cabo por programas secundarios, que no se corren en tiempo real como ocurre con los anteriores, pero que son necesarios para el criptoanálisis. Las tareas que realizan se pueden agrupar en:

- Determinar *F*, función que define $b_i = F(c_i)$.
- Armar y guardar la matriz de coeficientes *H*.
- Generar secuencia cifrante para determinar suposiciones válidas.

Los programas son de índole general, pudiendo ser aplicados en la resolución de un generador auto-shrinking cualquiera sea el polinomio generador del mismo. De hecho, estos programas han evaluado satisfactoriamente polinomios generadores que van desde grado 10 al grado 20. A partir del análisis realizado con registros o polinomios de grados relativamente bajos, [10 - 20], es posible extrapolar los resultados a LFSRs de mayor número de etapas.

4.4. OTROS CRIPTOANÁLISIS

En esta sección se muestran algunos criptoanálisis efectuados sobre este generador. En el marco del proyecto STORK, Strategic Road Map for Crypto, se plantean una serie de problemas abiertos en criptografía. Entre ellos se encuentra el criptoanálisis del generador auto-shrinking logrando órdenes de complejidad menores a los obtenidos aplicando la Técnica de Time-Memory Trade- Off.

A continuación se ven algunas propiedades tanto de la técnica de TMTO, como de otra que ha logrado mejores resultados y que se conoce como Técnica de Guess-and-Determine.

4.4.1. Time Memory Trade Off

Existen trabajos publicados sobre técnicas criptoanalíticas aplicadas al generador auto-shrinking. En general, los parámetros que se tienen en cuenta son la complejidad de la fase pre-computacional, la complejidad de memoria, la complejidad en tiempo y la cantidad de serie cifrante o secuencia interceptada que se requiere para su rotura.

Aplicando esta técnica de TMTO se logran órdenes de $O(2^{0.5*L})$ para la complejidad en tiempo, $O(2^{0.5*L})$ para la complejidad de memoria, $O(2^{0.25*L})$ para la cantidad de secuencia interceptada necesaria con una pre-computación de $O(2^{0.75*L})$. Estos órdenes son elevados y un criptoanálisis en tiempo real para un generador auto-shrinking en un rango de aplicación criptográfica sería sumamente complejo por los recursos que requeriría.

Esta Técnica de TMTO es ampliamente conocida y es aplicada actualmente para el criptoanálisis del algoritmo A5/1 utilizado en comunicaciones cifradas de telefonía celular, en la banda de GSM.

4.4.2. Guess and Determine

Es conocida en la Literatura una técnica de criptoanálisis, denominda Guess-and-Determine, y que ha sido aplicada sobre el generador auto-shrinking por B. Zhang and D. Feng (Científicos de la Academia China de Ciencias). Si bien el criptoanálisis que logran es aparentemente muy eficaz, su puesta en práctica resulta compleja. En efecto, se muestra un ejemplo de criptoanálisis de un generador auto-shrinking de longitud $L = 40$, aunque no resulta tan clara la necesidad que tienen en lo que se refiere a complejidad de memoria ni a la implementación de los supuestos. Más aún, no aparece comentado el porcentaje de aciertos que logran para distintos polinomios ni la cantidad exacta de secuencia interceptada que han utilizado. Logran resolver en forma confiable el generador auto-shrinking con una complejidad de tiempo de $O(2^{0.556*L})$, complejidad de memoria $O(L^2)$, y con sólo $O(2^{0.161*L})$ de bits de secuencia interceptada.

4.4.3. Conclusiones y Trabajo Futuro

Romper un generador de aplicación criptográfica no es tarea fácil, de todos modos aquí se presenta una técnica Criptoanalítica con parámetros muy claros y bien definidos para su resolución. Si bien todavía es prematuro hablar de tiempos de procesamiento, pues habría que optimizar la programación, si se ha logrado sintetizar una Técnica Criptoanalítica eficaz en la rotura de este generador. Se ha logrado asimismo encontrar un método de aplicación general que es independiente del polinomio generador o de su grado, y que consigue romper el Generador Auto-Shrinking. Para su implementación, sólo hay que variar en un programa el vector o array que representa el polinomio de realimentación de R_1 .

Este capítulo se refiere a una propuesta práctica para un criptoanálisis efectivo y pensando en un criptoanalista sin grandes recursos. Concretamente, se ha logrado órdenes de complejidad que están en $O(2^{0.33L})$ en lo que se refiere a secuencia interceptada, $O(L)$ en cuanto a cantidad de memoria consumida y cuya complejidad pre-computacional es sumamente baja, únicamente lo requerido para hallar los supuestos.

5. ESTEGANOGRAFIA

5.1. HISTORIA

Del griego steganoz (*steganos*, encubierto -con el sentido de oculto-) y graptoz (*graphos*, escritura) nace el término esteganografía: el arte de escribir de forma oculta.

Aunque κρυπτός (*criptos*, oculto) y steganoz (*steganos*, encubierto) puedan parecer en un principio términos equivalentes, o al menos similares, son cosas completamente distintas. La criptografía es el arte de escribir de forma enigmática (según la Real Academia Española), mientras que la esteganografía es el arte de escribir de forma oculta. Puede que sigan pareciendo similares, pero las connotaciones toman mucho valor al analizarlo detenidamente: la criptografía tiene su fuerza en la imposibilidad de comprender el mensaje, mientras que la esteganografía la tiene en el desconocimiento de que el mensaje siquiera existe.

Aplicado al campo informático, se puede dar los siguientes ejemplos: se podría robar un mensaje cifrado con relativa facilidad, pero aún sabiendo que contiene información importante se sería incapaz de obtener información alguna de él (si la criptografía ha cumplido con su cometido). Respecto a la esteganografía, se puede capturar el tráfico completo de un individuo y tratar de analizarlo completamente (y el "ruido de fondo" hoy en día es mucho), sin tener la certeza de que haya o no un mensaje oculto. Se espera que quede claro desde un principio que, en contra de lo que algunas personas dicen, la esteganografía NO es un tipo de criptografía: son técnicas distintas e independientes, si bien pueden complementarse entre ellas (y de hecho lo suelen hacer).

5.2. MÉTODOS

5.2.1. Métodos clásicos

La esteganografía da sus primeros pasos en la antigua Grecia. Se cuenta en "*Les Històries d'Heròdot*" que Demeratus quería comunicar a la ciudad de Esparta que Xerxes tenía planes para invadir Grecia. Para evitar ser capturado por espionaje en los controles, escribió sus mensajes en tablas que luego fueron cubiertas con

cera, de forma que parecían no haber sido usadas. Ésta es posiblemente una de las primeras manifestaciones en la historia de mensajes esteganografiados.

Otro método usado durante siglos consistía en tatuar al mensajero (generalmente un esclavo) un mensaje en la cabeza afeitada para después dejarle crecer el pelo y enviar así el mensaje oculto.

5.2.2. Cifrado nulo (Null Cipher)

El método de escritura de meta-información en un texto es usado desde hace siglos, y sigue siendo usado hoy en día. Esto es debido a que se trata posiblemente de uno de los métodos más sencillos de ocultar información.

Consiste en escribir un texto aparentemente inofensivo donde, mediante algún mecanismo conocido por el legítimo receptor de la información (actualmente se habla de algoritmos y claves), subyace la información realmente importante.

Vease un ejemplo de un mensaje real enviado por un espía alemán durante la Segunda Guerra Mundial:

- *Apparently neutral's protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue affects pretext for embargo on by products, ejecting suets and vegetable oils.*

Si de este inocente texto se extrae la segunda letra de cada palabra, se obtiene este otro mensaje:

- *Pershing sails from NY June 1*

Aquí se ve lo fácil que es esconder información en textos, así como comprender la necesidad de gran cantidad de información (ruido) para ocultar la auténtica información de forma que no llame la atención.

5.2.3. Tinta invisible

Aunque el método de escritura con tinta invisible es usado desde la edad media, es en la Segunda Guerra Mundial cuando adquiere una importancia capital. Fue usado muy activamente por la resistencia en los campos de prisioneros nazis.

Generalmente se usa de la siguiente forma: en primer lugar se escribe una carta completamente normal, y después se escribe, entre las líneas de esa carta, otro texto donde está la información importante. Era habitual el uso de vinagre, zumos de frutas u orina, aunque hoy en día existen compuestos químicos específicos que sirven igualmente y no desprenden olores tan fuertes (que serían fácilmente detectados por un perro entrenado). Al calentar el papel, la escritura oculta se hace visible.

5.2.4. Micropuntos

La tecnología de los micropuntos fue inventada por los alemanes durante la Segunda Guerra Mundial y fue usada de forma muy activa durante la época de la guerra fría.

La técnica se basa en esconder puntos minúsculos en fotografías, tan pequeños que para el ojo humano -e incluso para instrumentos ópticos básicos como lupas- resultan invisibles, pero que forman un patrón de información significativa.

Debido a la naturaleza analógica de esta técnica, resultaba fácilmente detectable para los servicios de inteligencia, si bien advertir la presencia de mensajes esteganografiados no siempre significa que puedan ser legibles. Aún así, descubrir la presencia de un mensaje esteganografiado se considera un fracaso de la esteganografía que lo soporta, pues la imposibilidad de comprender su contenido conforma su capa de cifrado.

5.2.5. Técnicas actuales

Actualmente la esteganografía está irremisiblemente ligada a los ordenadores, que le han proporcionado el medio necesario para ser efectiva, y del que durante siglos no pudo disponer. Así mismo, está íntimamente ligada a la criptología en general y a la criptografía en particular.

Hoy en día se usan multitud de técnicas esteganográficas, pero todas se basan en los mismos principios de ocultación de información. Este punto, al ser el eje central del documento, se verá en detalle más adelante.

Cabe destacar que gracias a los ordenadores personales, y al software libre en gran medida, técnicas antes reservadas a unos pocos están ahora al alcance de cualquiera... hasta de las peores personas.

Todo esto viene por la campaña de acoso y derribo que ciertos sectores norteamericanos emprendieron contra la criptografía y esteganografía en general, especialmente contra PGP y su creador Philip Zimmermann, al publicarse que esta clase de técnicas fueron supuestamente utilizadas por la organización terrorista AlQaeda para transmitir información previa a los atentados del 11 de Septiembre en Nueva York.

Señores: no hay medios culpables, sino personas culpables. No culpen a PGP de un atentado, no menos que a Samuel Colt por inventar el revólver (eso sin entrar a analizar los usos positivos de PGP frente a los dudosos usos positivos del revólver...) o a Albert Einstein por contribuir al desarrollo de la bomba de fisión.

5.3. PUBLICACIONES

Son realmente pocas las publicaciones que existen sobre esteganografía, en comparación por ejemplo con la criptografía. Si se busca publicaciones en castellano, el número desciende muchísimo más, hasta casi el cero.

Al contrario que la criptografía, que existe de una manera importante desde antes del desarrollo del computador y fue desarrollada especialmente en la Segunda Guerra Mundial, la esteganografía, pese a haber sufrido un desarrollo similar durante ese mismo periodo de tiempo, nunca dispuso de medios que la permitieran tomarse en cuenta hasta la aparición de la moderna ciencia de la computación. Es por eso que las publicaciones referentes al tema son tan escasas, y generalmente se encuentran englobadas como una parte de un texto criptográfico.

Vease algunas de las principales publicaciones relacionadas con el tema por orden cronológico:

- ✓ **Hypnerotomachia poliphili (1499) - Anónimo**

Es un libro publicado en 1499 por Aldus Manutius. El libro versa sobre

conocimiento general, tratando temas como arquitectura, ingeniería, paisajes, creación de jardines, pintura, escultura... de todo menos esteganografía, criptografía o códigos de forma alguna.

¿Qué es pues lo que hace tan significativo este libro? Contiene muchísimos datos escondidos en su interior. El más famoso, y el considerado como primer texto esteganografiado de forma escrita se obtiene tomando la primera letra de cada uno de sus capítulos, formando:

Poliam frater Franciscus Columna peramavit

Que significa "*El Padre Francesco Colonna ama apasionadamente a Polia*".

Por cierto, Francesco Colonna aún vivía cuando el libro fue publicado.

- ✓ **Steganographia (1499) - Tritheim Johannes Heidenberg**

Se trata de la publicación más notoria de Tritheim. En Steganographia se incluye un sistema de esteganografía bastante avanzado, pero la temática general del libro (magia y métodos de aprendizaje acelerados) hizo que nunca se tomara demasiado en serio el texto.

No obstante, en este libro se trataba un método de envío de mensajes sin caracteres ni mensajero.

Se puede encontrar un profundo análisis de este libro en:

- http://www.esotericarchives.com/tritheim/stegano.htm.

- ✓ **Polygraphiæ (1516) - T ritheim Johannes Heidenberg**

Segunda publicación relacionada con este tema de Tritheim. En Polygraphiæ vuelve a tratar el tema de la escritura y su significado, aunque no es un libro muy importante en el campo de la esteganografía.

No obstante, junto a Steganographia -del mismo autor-, sentará las bases de lo que será Schola Steganographica, el considerado como primer libro de esteganografía y criptografía de la historia, y uno de los más importantes.

- ✓ **Schola Steganographica (1665) - Gasparis Schott**

Este libro es uno de los más importantes en la historia de la criptografía y la esteganografía.

En él se discuten los conocimientos de la época respecto a la escritura oculta o cifrada, algunos de ellos tratados anteriormente, pero que en este libro toman un profundo giro en el enfoque de estudio: Schott se aleja de lo esotérico y lo mágico para enfocar la esteganografía y la criptografía desde el punto de vista de la técnica y la ciencia.

En el Whipple Science Museum de Cambridge se conserva una copia completa e intacta (la de la fotografía) que fue publicada junto a otra obra del mismo autor (Technica curiosa).

Este texto trata el uso de micrografía y escritura oculta en los mensajes enviados por palomas durante la guerra franco-prusiana (1870-1871).

Según el texto, las técnicas de ocultación de información fueron decisivas en el desenlace de la contienda.

- ✓ **La cryptographie militaire (1883) - Auguste Kerckhoffs**

Es un estudio del uso de técnicas criptográficas, esteganográficas y en general de ocultación de información en el ejército francés del siglo XIX.

Cualquier estudioso de la criptografía ha oído al menos hablar de los Principios de Kerckhoffs. Pues estos principios fueron enunciados en Enero de 1883 durante la redacción de este libro.

- ✓ **Le filigrane (1907) - Charles-Moïse Briquet**

Esta publicación es un diccionario histórico de las marcas de agua, usadas a lo largo de la historia para autentificar todo tipo de documentos. Hoy en día las marcas de agua siguen usándose en la expedición de papel moneda.

- ✓ **The Codebreakers: The story of secret writing (1967) - David Kahn - [KAHN67] - Actualizado 1996**

The Codebreakers es un libro de obligada lectura para los aficionados a la criptografía, esteganografía y ocultación de información en general.

Se trata de “La Biblia” de los códigos, formando un texto de referencia histórica incomparable. El autor barre un periodo de tiempo descomunal, desde los primeros jeroglíficos del 3000 A.C. hasta la época de publicación de la primera edición del libro (1967).

El hecho de que este libro fuera publicado antes de la existencia de la moderna criptografía computacional no hace sino aumentar el valor intrínseco del texto, pues muestra las entrañas de la teoría de códigos. El libro también muestra episodios históricos importantes relacionados con la ocultación de información, como el Telegrama Zimmermann de la I Guerra Mundial o el funcionamiento y ruptura de las máquinas Enigma de la II Guerra Mundial.

El libro fue revisado en 1996 por el propio autor para contemplar la moderna criptografía de clave pública y la influencia de la informática en la ocultación de información. Aún así, no son muchos los cambios entre ediciones, y se podría decir que el libro sigue siendo algo que pertenece a 1967.

5.4. BASES DE LA ESTEGANOGRAFIA

El desarrollo de la informática e Internet ha supuesto el marco perfecto para que la esteganografía alcance su mayoría de edad. Los avances en computación proporcionan medios para calcular rápidamente los cambios necesarios en la ocultación de un mensaje, e Internet proporciona los medios necesarios para transportar grandes cantidades de información a cualquier punto del planeta.

La esteganografía actual se basa en esconder datos binarios en la maraña de bits que supone un fichero.

Los bits que componen el mensaje a ocultar se introducen (bien sea añadiéndolos, o realizando operaciones aritméticas con los originales) en el fichero ya existente, procurando que el fichero resultante después de realizar los cambios parezca el original.

¿Cómo se logra que el fichero resultante no parezca haber sido modificado? Depende de qué tipo de fichero se esté modificando. Prácticamente cualquier tipo de fichero es bueno para ocultar datos en su interior, pero hay algunos (imágenes y sonido principalmente) que resultan ideales para este cometido, por motivos que más adelante se comentaran. Así mismo existen ciertos programas especializados en ocultación de información en sectores de disco no usados.

Sea cual sea el tipo de información que se quiera esteganografíar, y sea cual sea el medio en que se quiera hacer, hay ciertas reglas básicas:

- ✓ Toda información (texto ASCII, hexadecimal, código morse...) que se quiera introducir, debe ser primero convertida a binario. Si bien cualquier base numérica es válida, la comodidad trabajando con binario es mucho mayor.
- ✓ Nunca hay que permitir que un supuesto atacante obtenga el fichero original (anterior a la modificación), pues permitiría, mediante comparación, establecer pautas de cambios en la información. Esto podría llevar en última instancia a desentrañar el mensaje oculto.
- ✓ Las cabeceras de los ficheros -salvo excepciones- NO deben ser modificadas.
- ✓ No transmitir la clave o algoritmo esteganográfico por un medio inseguro.

Aunque la esteganografía computacional clásica consiste en la modificación binaria del fichero que sirve de canal, existen ciertas técnicas para casos particulares de ficheros que también son válidas (aunque complicadas de hacer “a mano”, con lo cual dependemos de algún tipo de software... cosa que no se quiere). Un ejemplo de estas técnicas es la adición de mensajes ocultos a los ficheros de sonido mediante superposición de capas de sonidos que no resultan audibles para el oído humano, pero que sí contienen información.

Así mismo, también he encontrado documentación de técnicas basadas en ocultación de mensajes en ficheros de imagen creados con potentes programas de tratamiento gráfico (como Gimp o Photoshop) mediante el uso de capas transparentes donde se alojaba la información. Al igual que en el caso anteriormente citado, no se considera esta técnica segura en absoluto.

Así pues, se tratará únicamente la esteganografía al nivel del bit (mediante editores hexadecimales).

5.5. FICHEROS INTERPRETADOS

Antes de ver los tipos más comunes de ficheros interpretados usados en esteganografía, se va a echar un vistazo a algunos detalles previos cuya comprensión es muy conveniente para entender la esteganografía.

5.5.1. ¿Qué es un fichero interpretado?

Entre los ficheros que se encuentran en el disco duro se encuentra principalmente dos grupos: ficheros interpretados y ficheros ejecutables. Los ficheros ejecutables son los que mandan instrucciones al procesador (a través del sistema operativo y su núcleo o kernel), mientras que los ficheros interpretados son aquellos usados para leer o escribir información, pero siempre mediante algún software.

Para entender esto más fácilmente, se puede pensar en situaciones cotidianas. Al abrir una fotografía PNG ó JPG con el software de imagen favorito (Gimp, Photoshop...) se está leyendo un fichero interpretado (la fotografía) mediante un software que previamente se ejecuta (mediante el fichero ejecutable). En este proceso, en primer lugar se ejecuta el fichero ejecutable que, a través del SO, manda instrucciones al procesador, y en segundo lugar ese software ya arrancado interpreta la información contenida en una imagen y envía las instrucciones necesarias para poder ver en pantalla esa información ya procesada.

Existen unos ficheros denominados “scripts” que en determinados casos, pese a ser ficheros interpretados, son capaces de ejecutar código en el procesador (aunque siempre mediante un intérprete). Es el caso de, por ejemplo, programas escritos en bash scripting (ejecutados por bash, sh...) o en perl (ejecutados por el intérprete de perl). Aunque hay quien los considera ejecutables y quien no, porque en cualquier caso no son muy útiles con fines esteganográficos, debido a que en su mayoría se trata de ficheros de texto plano.

5.5.2. Estructura de un fichero

Antes de empezar a modificar ficheros es importante entender la estructura básica de un fichero.

Aunque los usuarios de sistemas Windows no lo sepan, un fichero es lo que es debido a una información que se encuentra embebida en su interior y que se llama cabecera. Aunque en S.O como Unix y compatibles (Linux, xBSD...), un fichero BMP es reconocido como tal aunque su nombre sea “foto.pepe”; mientras que en los sistemas Windows simplemente se verifica la extensión del fichero, de forma que carta.bmp se tratará de abrir con el visor de imágenes predeterminado aún tratándose de texto.

Además de la cabecera, el fichero contiene la información propiamente dicha (y en algunos casos otros campos que no vienen al caso). Debido a que cada tipo de fichero tiene una estructura distinta, se pueden dar casos anecdóticos, como por ejemplo que un fichero pueda ser a la vez de dos tipos sin que éstos interfieran entre sí. Un ejemplo es un fichero gif-zip. ¿Cómo es posible esto? Pues por las especificaciones de cada uno de los ficheros: un fichero gif define en su cabecera el tamaño del mismo, de forma que cualquier byte posterior al supuesto fin de la información es ignorado, y un fichero zip almacena la información en unas tablas de forma que cualquier byte anterior al inicio de éstas es ignorado. Así, si se define una cabecera gif de, pongase, 5000 bytes, y en el byte 5001 y posteriores se ponen las tablas zip, se tendrá un fichero que es gif y zip a la vez.

Por norma general, NO se va a tocar nunca la cabecera del fichero. Siempre se habrá de fijar en el cuerpo del mismo para introducir los datos. La forma en que se introducirán los datos, así como el sitio donde se hará, depende del tipo de fichero.

5.5.3. Ficheros de imagen

Las imágenes es de lejos el tipo de fichero más utilizado para esteganografíar mensajes ocultos.

Conviene puntualizar que hay distintos tipos de ficheros de imagen:

- ✓ **Windows BitMaP (BMP):** Es el formato gráfico más simple, y aunque teóricamente es capaz de realizar compresión de imagen, en la práctica jamás se usa. Consiste simplemente en una cabecera y los valores de cada pixel de la imagen (ocupando cada pixel 4, 8, 16, 24 ó 32 bits según la calidad del color) empezando de abajo hacia arriba y de izquierda a derecha. Su principal ventaja es la sencillez (es el formato más indicado

para realizar esteganografía). Su mayor inconveniente es el inmenso tamaño que ocupan.

- ✓ **PC Paintbrush (PCX):** Este tipo de fichero es una evolución del mapa de bits tradicional. En PCX se usa el algoritmo de compresión RLE. Mediante RLE cuando dos o más pixels consecutivos tienen el mismo color, el algoritmo guarda la información del color y el número de pixels que lo usan (para la posterior visualización).

  El criterio para almacenar el número de pixels usando el color es el siguiente: si el byte es menor o igual que 192, corresponde a un único pixel, pero si es superior a 192 el número de pixels repetidos lo dan los seis bits menos significativos del byte (se pone a cero los dos bits más significativos) y el color lo da el byte siguiente.

  Como ventajas tiene la sencillez del algoritmo (aunque, como ya se verá, este formato de compresión hace mucho más tedioso la realización de esteganografía), y como inconveniente, la escasa compresión que se obtiene en fotografías (que tienen mayor variedad de pixels), que al fin y al cabo es el mayor uso de imagenes.

- ✓ **Graphics Image Format (GIF):** Es uno de los mejores formatos de compresión (sobre todo para imágenes con grandes áreas de un mismo color), además de ser la opción más sencilla para animaciones vectoriales (flash y otros métodos más caros y complejos aparte...). El formato GIF89a además soporta transparencias y entrelazado. Usa el algoritmo de compresión LZW (usado en compresión de ficheros también), mucho más complejo que RLE. Su principal punto débil es la limitación a 256 colores (8 bits) de la paleta de color, lo cual lo hace desaconsejable para cualquier tipo de fotografía o imagen realista.

  Su principal ventaja es la enorme compresión (cosa que nos complica sobremanera la esteganografía) y la capacidad de uso de transparencias y entrelazado, mientras que su mayor defecto es la escasa paleta de colores.

- ✓ **Joint Photographic Experts Group (JPEG):** Este fichero es, con diferencia, el más popular. El algoritmo de compresión de JPEG se basa en un defecto del ojo humano que impide la completa visualización de la paleta de 24 bits, por lo que elimina la información que el ojo humano no es capaz

de procesar. Esto nos da una importante reducción de tamaño, pero -muy importante este algoritmo SÍ tiene pérdida de información en el proceso de compresión. Dependiendo del factor de compresión la pérdida de imagen puede ser visible o no al ojo humano.

Una variante del JPEG original es el JPEG progresivo, que realiza entrelazado de datos para visualizarlo en primer lugar con baja calidad e ir aumentando la misma en varias pasadas.

La principal ventaja que tiene JPEG es su calidad a la hora de representar fotografías (con su paleta de 16 bits y su alta compresión), y su principal desventaja es la pérdida de calidad e información con grandes ratios de compresión.

JPEG es sin duda el más usado en esteganografía, pero eso no significa que sea el más sencillo.

- ✓ **Tagged Image File Format (TIFF):** TIFF es un formato usado en imágenes de altísima resolución y calidad, principalmente en trabajos de imprenta o fotografía profesional. Se trata básicamente de un mapa de bits preparado para el estándar CMYK, y preparado para el uso de muchos estándares y formatos de compresión diversos, que pueden ser usados en la misma imagen.

  La ventaja de este formato es la enorme calidad obtenida, y su principal desventaja el tamaño que ocupa. Debido a lo específico de este tipo de fichero, no es prácticamente usado para esteganografía.

- ✓ **Portable Network Graphics (PNG):** El formato PNG nace debido a los problemas de la patente del algoritmo LZW (Lempel-Ziv-Welch), y con la intención de sustituir a GIF como estándar. PNG cubre prácticamente todas las características de GIF, con un mejor algoritmo de compresión, sin pérdida de información y con una paleta de color muy superior a los 256 bits de GIF (16 bits).

  Además, se trata del único formato comprimido que incorpora la información del canal alpha, logrando una altísima calidad en el uso de capas y transparencias.

PNG además es uno de los primeros ficheros de imagen en contener información acerca del fichero en forma de metadatos de texto.

A pesar de todas sus ventajas (principalmente el tratarse de un formato libre), tiene un defecto: no permite el uso de animaciones (al contrario que GIF). La organización W3C (autores de PNG) ha creado el formato MNG para animaciones.

Aunque son muchas las formas de representar una imagen en un ordenador, todas tienen un denominador común: tienen que representar colores mediante bits, bien sea cada punto, vectores o tablas.

La principal arma es la calidad de la imagen, por lo que según la profundidad de color será más o menos sencilla la realización de esteganografía. Con 4 y 8 bits (16 y 256 colores respectivamente) la variedad no es muy alta, por lo que la diferencia entre colores contiguos es poca.

En profundidades de color más comunes hoy día, como 16, 24 y 32 bits (65.535, 16.777.216 y 4.294.967.296 de colores respectivamente) la cosa es muy diferente.

Si bien con 16 bits la diferencia entre colores no es tanta, modificar un punto en una imagen de tamaño medio (200*200 = 40.000 puntos) y con una diferencia de color de 1 entre 65.535... Nadie podrá distinguir este punto. Con 24 y 32 bits ya la diferencia es muchísimo mayor.

Una manera de hacerlo es cambiando el color de tres puntos en la imagen de forma que sean los colores inmediatamente contiguos en una paleta de más de 16 millones de colores. El ojo humano no podría distinguirlo nunca.

También es posible representar los ceros restando uno a la cifra original y los unos dejándola inalterada, o los ceros restando uno y los unos sumando uno, aunque lo más recomendable es alterar lo mínimo posible el fichero original.

En ficheros con compresión es más difícil introducir cambios sin que el resultado se altere, debido a que se debe respetar la estructura del algoritmo de compresión, cosa bastante complicada. Aún así, con algo de práctica y paciencia es perfectamente posible introducir mensajes esteganografiados en imágenes JPEG por ejemplo.

Ya en Internet, mediante correo electrónico o una página web, es muy fácil distribuir un documento con información esteganografiada.

5.5.4. Ficheros de sonido

Los ficheros de sonido son también utilizados a menudo en técnicas esteganográficas.

Por ejemplo los tipos de ficheros de sonido más comunes:

- ✓ **Waveform Audio File Format (WAV):** Se trata del formato de sonido desarrollado por Microsoft para su sistema operativo Windows. Se compone de una cabecera de 43 bytes y un conjunto arbitrario de bytes que contienen las muestras una tras otra, sin ningún tipo de compresión y con cuantificación uniforme. Al ser un formato realmente sencillo, es muy útil para realizar tratamiento digital de sonido.

  Su principal ventaja es la sencillez del formato, y su mayor inconveniente la cantidad de espacio requerido (una muestra de 10 segundos en calidad de CD -PCM, 44 KHz, 16 bit, estéreo ocupa 1,6 Mb).

- ✓ **Motion Picture Experts Group - Audio Layer 3 (MP3):** Sin duda el más famoso de todos los formatos de audio. Este formato utiliza un algoritmo de compresión con pérdida de información, basándose en las limitaciones del oído humano: somos sensibles a las frecuencias medias, pero poco perceptivos con las altas o bajas; y además percibimos mal los sonidos bajos que suenan a la vez que sonidos muy fuertes (efecto de ocultamiento). Así pues, eliminando los sonidos que no oímos, logra un ratio de compresión de hasta 12:1.

  Su principal ventaja, sin duda, es la gran compresión y la poca pérdida de calidad que tiene, y su principal inconveniente es la pérdida inevitable de información.

- ✓ **OGG Vorbis (OGG):** Este formato se desarrolla para constituir una alternativa totalmente libre a mp3. Aunque es un formato aún muy joven (aún no hace dos años de la finalización de la versión 1.0), tiene mucho que decir, pues tiene un sistema de compresión similar al de mp3, pero logrando mayores ratios de compresión (ocupa menos) y una mayor calidad de sonido.

El principal obstáculo para la implantación de ogg es la “estandarización” de mp3: reproductores portátiles, equipos de audio, reproductores DVD domésticos.

La principal ventaja de ogg es la mayor compresión y calidad con respecto a mp3, así como ser 100% libre de patentes. Su principal inconveniente, como en mp3, es la pérdida de información en el proceso de compresión.

La representación de la información en los ficheros de sonido se realiza generalmente mediante el registro consecutivo de las muestras que componen el sonido. Al igual que ocurría con los ficheros de imagen, el sonido es representado siempre en forma de bits, y cada vez con una calidad mayor, de forma que para los experimentos esteganográficos se puede jugar con los bits menos significativos de la información.

En el sonido ocurre una cosa muy curiosa: da prácticamente igual que la calidad sea alta o baja, pues en calidades altas la cantidad de bits usados para representar el sonido es muy alta, y un cambio ínfimo no influye en el resultado; mientras que en los de calidad baja, aunque la modificación sea más llamativa, al tratarse de un sonido de baja calidad (por ejemplo calidad de radio), cualquier modificación puede pasar inadvertida como ruido de fondo.

La forma de introducir los datos es la misma que ya se ha visto en el caso de ficheros de imagen.

5.6. FICHEROS EJECUTABLES

¿Esconder datos en ficheros ejecutables? ¿Es eso posible? SÍ, aunque se trata de una técnica compleja que no siempre da buenos resultados.

Antes de entrar al meollo de la cuestión, conviene hacer una anotación: si bien para comprender esta parte no se necesitara conocimientos específicos, si se quiere realizar de forma seria esta técnica se debe tener ciertos conocimientos sobre virus informáticos e ingeniería inversa (desensamblado, crackeo...), así como también de programación en lenguaje ensamblador (AT&T e Intel, según se trabaje con ficheros ELF o Win32 PE).

Para entender cómo funciona esta técnica se tiene que echar un vistazo al modo de trabajo de los virus que infectan ejecutables. Este tipo de virus hoy día no son

muy comunes, pues han sido desplazados por los omnipresentes gusanos, pero hace unos cuantos años era el tipo de virus más común. Según su técnica de infección se podía distinguir dos tipos de virus:

El primer tipo inyecta su código al final del ejecutable e introduce en algún lugar libre del mismo una instrucción de salto incondicional (JMP) hacia el área que contiene el código del virus. El principal efecto es el aumento de tamaño del fichero con respecto a la versión sin infectar (dada la inyección de código).

El segundo tipo, mucho más complejo de programar y mucho más raro de encontrar, busca áreas libres en el ejecutable (debido al alineamiento) y distribuye su código en esos "huecos" del fichero, uniendo todo el código mediante diversas instrucciones de salto. Los efectos de este complejo método de infección son principalmente que el fichero sigue ocupando el mismo espacio en disco y que las compañías antivirus se muerden las uñas a la altura del codo para crear rutinas de desinfección para estos bichos.

En este caso no se quiere infectar nada, pero sí se piensa que no estaría mal poder ocultar información en esos huecos libres que se sabe que existen.

- ✓ ¿Por qué existen esos huecos?

Dependiendo del compilador que se use, éste puede usar un alineamiento de entre 4 y 16 bytes.

Eso da un posible espacio de huecos libres de entre 0 y 15 bytes para utilizar como se quiera. La mayoría de los compiladores de C (principal lenguaje de programación) trabajan con un alineamiento de 16 bytes.

- ✓ ¿Cómo se reconoce esos huecos libres?

Cada compilador tiene una forma de rellenar esos huecos. Por ejemplo, los compiladores de C de Microsoft suelen usar la instrucción debug (INT 3) que en forma hexadecimal es 0xCC (CC); los compiladores de C de Borland suelen usar la instrucción NOP que en forma hexadecimal es 0x90 (90); y otros compiladores como GCC (Gnu C Compiler) usan diversas instrucciones, lo que entre otras cosas dificulta la infección del ejecutable.

- ✓ ¿Se puede escribir en esos huecos sin más?

NO, y este es uno de los principales motivos por el que esta técnica es tan compleja. Si antes de esos huecos se encuentra un salto incondicional, o una instrucción comparativa que ejecute dos o más saltos condicionales dependiendo

del resultado de evaluar la expresión, o cualquier otra cosa con la condición de que el programa NO continúe en ese hilo de ejecución de forma lineal entonces sí se puede usar esos huecos al antojo. En caso de que el programa continúe su hilo de ejecución de forma normal, NO se podrán usar.

Esto diferencia la técnica esteganográfica en ejecutables de la infección por virus.

A un virus le interesa ejecutar código, enviar instrucciones al procesador, de forma que los huecos que valen (los menos) al virus no le valen, y viceversa. Se quiere que se modifique NO sea enviado al procesador como código a ejecutar, porque en el mejor de los casos el programa hará cosas raras, y en el peor (la mayoría) el ejecutable quedará inservible con lo cual no se está cumpliendo los preceptos de la correcta esteganografía.

Así pues, el que quiera saber más de esteganografía en ejecutables primero debe empaparse de ingeniería inversa. Al fin y al cabo ya se sabe cómo encontrar los bytes libres, y ya se sabe cómo introducir información.

Ahora el jarro de agua fría: la esteganografía en ejecutables tiene varios problemas serios.

A pesar de la complejidad de la esteganografía sobre ejecutables, no es mejor que sobre ficheros interpretados a decir verdad es peor. ¿Por qué? Porque cualquiera con conocimientos en ingeniería inversa puede ver que el código del programa llegado a determinado punto no ejecuta instrucciones de un área que contiene instrucciones (muy probablemente instrucciones sin sentido, al tratarse de información), y deducir que ese área originalmente contenía instrucciones vacías.

Además existe otro problema: si el software usado realiza comprobaciones MD5, el ejecutable modificado será considerado corrupto, y por tanto no válido.

Cualquier pretensión de pasar información inadvertida se pierde.

Por último, las compañías antivirus en sus productos modernos incluyen sistemas heurísticos de detección de virus, mediante los cuales reconocen patrones de conducta, no virus concretos. La información esteganografiada en un ejecutable se parece bastante a un virus, por lo que quizá sea detectada como un virus.

Como conclusión se puede decir que la esteganografía en ficheros ejecutables no es práctica, pues implica tener conocimientos avanzados de ramas de la seguridad informática muy complejas (virus, lenguaje ensamblador, ingeniería

inversa), necesita mucho tiempo para llevarse a cabo, y conlleva bastantes problemas prácticos en su ejecución.

5.7. ESTEGANOGRAFIA AVANZADA

La esteganografía es un arte complejo y con muchos matices. Sin llegar aún a la combinación de esteganografía y criptografía, es posible el uso de determinadas técnicas avanzadas que permiten aumentar la eficacia de una información oculta mediante esteganografía. Véase algunas:

- ✓ **Uso de múltiples claves**

Esta técnica es heredada directamente de la criptografía, pero con distinta forma de aplicación. Consiste en usar distintas codificaciones para cada porción arbitraria del mensaje a ocultar. Así, una frase de cinco palabras puede tener una clave de codificación para cada una de las palabras: en la primera se resta una unidad en los ceros y se suma una unidad en los unos, en la segunda se realiza lo mismo pero invirtiendo el orden de los bits, en la se tercera realiza el XOR de los bits. Naturalmente la clave ha de ser conocida por el destinatario.

- ✓ **Esteganografía en capas**

Mediante esteganografía en capas se establece una relación lineal entre los elementos ocultos. Así, la codificación de la segunda palabra o letra de un mensaje depende de la primera (puede depender del último valor de la cifra, del último valor modificado, de la posición).

Así se establece un orden estricto de decodificación que impide obtener completamente el mensaje sin la primera parte, con lo cual únicamente se debe comunicar la clave para obtener esta parte y la pauta a seguir para encadenar los fragmentos.

- ✓ **Adición de ruido**

Aunque en un mensaje esteganografiado TODO el fichero es considerado ruido, se puede añadir ruido en el proceso de esteganografiado. Así, además de modificar los bits necesarios para inyectar un mensaje, se puede modificar unos cuantos bits aleatorios del mensaje de forma que aún teniendo el fichero original, un posible atacante deba conocer el sistema de codificación usado.

- ✓ **Uso de distintas magnitudes**

Aunque lo habitual es variar en 1 bit el byte del mensaje original, nada impide variarlo en más bits.

Así, se puede establecer claves complejas, como por ejemplo: ocultar una frase de cinco palabras, y al ocultar la primera de las palabras se suma 1 bit en la codificación de la primera letra, 2 bits en la codificación de la segunda, 3 bits en la tercera, hasta que vuelva a aparecer una modificación de 1 bit, que significará el inicio de otra palabra.

Mientras se trabaje con ficheros que usen mucha información (imágenes de 24 bits o más por ejemplo) no se notará que se varié la escala en 1 ó 10 unidades, y proporciona un tipo de clave más compleja.

- ✓ **Otras técnicas**

Existen muchas otras técnicas esteganográficas que permiten aumentar la complejidad y seguridad, tantas como se quiera idear. Lo ideal es que cada uno idee y use su propia técnica.

5.8. ESTEGANOGRAFÍA Y CRIPTOGRAFÍA

Como ya se comentó al principio del presente documento, la esteganografía hoy día está íntimamente ligada a la criptografía. Teniendo unos conocimientos básicos de criptografía y esteganografía se puede ocultar los datos con un grado de seguridad sorprendentemente alto.

Mediante técnicas criptográficas como las usadas por PGP (Pretty Good Privacy) se puede hacer los datos completamente ilegibles, pero un fichero cifrado con PGP tiene una estructura muy específica que lo hace reconocible de inmediato. En ciertos países el gobierno controla a la población hasta el extremo de controlar la información que emiten a la red (por ejemplo en China), por lo que cualquier dato cifrado sería interceptado de inmediato. Se puede comparar la criptografía a tener alarma en casa: la seguridad aumenta muchísimo, pero todos los que vean las medidas de seguridad sabrán que se tienen cosas importantes que guardar (por norma general se cifran únicamente los contenidos muy importantes).

Mediante técnicas esteganográficas se puede hacer que cualquier información pase inadvertida (subyaciendo en información inofensiva), pero la seguridad intrínseca de la esteganografía para datos importantes no es mucha.

Mediante la combinación de estas dos técnicas se establece dos capas en la seguridad de la información: la primera capa, más externa, es la esteganográfica; y la segunda, interna, la criptográfica.

Cada una de las capas tiene un cometido en esta peculiar simbiosis: la capa criptográfica se encarga de la seguridad de los datos (pues aunque la esteganografía sea un medio de proteger datos, no es comparable al cifrado) mientras que la capa esteganográfica protege la integridad de la capa criptográfica.

Aunque nada impide realizar los cifrados a mano, para usar algoritmos y claves complejas es casi imprescindible el uso de software especializado. Se recomienda cualquier implementación de openPGP, especialmente GnuPG que es software libre, y cuyo código fuente puede ser revisado por cualquiera que desconfíe.

Si se cifran los datos de forma que devuelva una armadura ASCII, ya se tendrá todos los caracteres que componen el mensaje cifrado. Es muy recomendable que la clave que se use para la realización de criptografía orientada a esteganografía use el algoritmo DH/DSS (Diffie-Hellman / Digital Standard Signature), pues el tamaño de salida de datos es mucho menor. Con la armadura ASCII del mensaje solamente se necesita un poco de paciencia para esteganografíar los datos.

5.9. ATAQUES A LA ESTEGANOGRAFÍA

No son muchas las investigaciones publicadas acerca de métodos de ataque a la esteganografía, aunque es de esperar que debido al auge que ha experimentado hoy en día gracias a técnicas como las marcas de agua, se publiquen más investigaciones sobre debilidades y formas de aprovecharlas.

Los dos métodos más usados para detectar y atacar la esteganografía son:

- ✓ **Ataque visual**

Consiste básicamente en buscar de forma manual diferencias entre el fichero original y el esteganografiado, siempre y cuando dispongamos del fichero original.

En caso de no disponer de él, se pueden buscar irregularidades en el fichero esteganografiado para tratar de encontrar signos de la existencia de datos ocultos, pero difícilmente se podrá obtener información útil más allá de la existencia de los mismos.

Esta técnica es la más rudimentaria a la hora de realizar análisis esteganográfico y no tiene mucha utilidad real.

- ✓ **Ataque estadístico**

Este tipo de ataque se basa en el mismo concepto que el criptoanálisis diferencial del que hablé en Criptosistemas Informáticos.

El concepto de este tipo de ataques se basa en la comparación de la frecuencia de distribución de colores de un fichero potencialmente esteganografiado con la frecuencia que se podría esperar en teoría de un fichero esteganografiado.

Aunque los resultados son bastante buenos (ha demostrado en muchas ocasiones ser efectiva), esta técnica es extremadamente lenta.

Si la creación de mensajes esteganografiados a mano ya es complicada, el ataque estadístico lo es mucho más, por lo que se automatiza mediante diversos programas. Pero a nuestro favor se tiene que los programas especializados en detección y ruptura de mensajes esteganografiados suelen buscar pautas de mensajes ocultos con algún tipo de software especializado en la creación de los mismos. Si se esteganografía los mensajes a mano, la posibilidad de que la información sea recuperada ilícitamente es ínfima.

Uno de los programas más famosos para detectar la presencia de esteganografía en ficheros de imagen JPG es Stegdetect (http://www.outguess.org/detection.php).

Este software, mediante su módulo Stegbreak, realiza ataques de diccionario a los principales sistemas públicos de esteganografía.

En conclusión dada la naturaleza de la esteganografía, resulta extremadamente resistente a cualquier tipo de ataque.

Los programas que aseguran detectar y recuperar datos ocultos mediante esteganografía lo hacen siempre buscando mensajes ocultos con programas de dominio público. Si se esteganografía los propios mensajes a mano, si se es programador, se desarrolla una herramienta privada, las posibilidades de que los datos sean comprometidos son prácticamente cero.

5.9.1. Software Esteganográfico

Actualmente existen muchos programas especializados en ocultación de mensajes de texto mediante esteganografía, en la mayoría de los casos en archivos de imagen o sonido.

En primer lugar, NO se considera buena idea el uso de ningún tipo de software para ocultar datos importantes, puesto que en la esteganografía no existen las claves como en la criptografía, sino que la clave es la técnica usada para esteganografíar. Si se usa un software que tiene su propia técnica, se está permitiendo implícitamente que cualquiera con ese software pueda recuperar los datos.

Como mucho, ciertos programas incorporan la opción de establecer una contraseña y añadir algún cifrado sencillo a los datos, nada que un ataque de diccionario no pueda solucionar.

En segundo lugar, al igual que ocurre con la inmensa mayoría de los programas de cifrado, existe la posibilidad (sí, se es un paranoico) de la existencia de backdoors que comprometan totalmente la información contenida. Por ello, solamente se considera fiable al 100% el software libre, aunque en caso concreto, el que el código sea público facilita aún más a un atacante el obtener la información.

6. COMPLEJIDAD DE LOS ALGORITMOS

6.1. INTRODUCCIÓN

Cuando se quiere diseñar una solución a un problema de cifrado se crea un algoritmo que se pretende enfrentar a un posible atacante esperando que éste sea incapaz de resolverlo. Pero, ¿bajo qué circunstancias se puede considerar que un problema es intratable? Evidentemente, se quiere que el fisgón se enfrente a unos requerimientos de computación que no pueda asumir. La cuestión es cómo modelizar y cuantificar la capacidad de cálculo necesaria para abordar un problema.

La resolución práctica de un problema exige por una parte un algoritmo o método de resolución y por otra un programa o codificación de aquel en un ordenador real. Ambos componentes tienen su importancia; pero la del algoritmo es absolutamente esencial, mientras que la codificación puede muchas veces pasar a nivel de anécdota.

Para efectos prácticos, debe preocupar los recursos físicos necesarios para que un programa se ejecute. Aunque puede haber muchos parámetros, los más usuales son el tiempo de ejecución y la cantidad de memoria (espacio). Ocurre con frecuencia que ambos parámetros están fijados por otras razones y se plantea la pregunta inversa: ¿cuál es el tamaño del mayor problema que puedo resolver en T segundos y/o con M bytes de memoria? En lo que sigue es mejor centrarse casi siempre en el parámetro tiempo de ejecución, si bien las ideas desarrolladas son fácilmente aplicables a otro tipo de recursos.

Para cada problema se determinara una medida N de su tamaño (por número de datos) y se intentara hallar respuestas en función de dicho N. El concepto exacto que mide N depende de la naturaleza del problema. Así, para un vector se suele utilizar como N su longitud; para una matriz, el número de elementos que la componen; para un grafo, puede ser el número de nodos (a veces es más importante considerar el número de arcos, dependiendo del tipo de problema a resolver); en un fichero se suele usar el número de registros, etc. Es imposible dar una regla general, pues cada problema tiene su propia lógica de coste.

6.2. TIEMPO DE EJECUCIÓN

Una medida que suele ser útil conocer es el tiempo de ejecución de un programa

en función de N, lo que se denominara T(N). Esta función se puede medir físicamente (ejecutando el programa, reloj en mano), o calcularse sobre el código contando instrucciones a ejecutar y multiplicando por el tiempo requerido por cada instrucción. Así, un trozo sencillo de programa como

- S1; for (int i= 0; i < N; i++) S2;

  Requiere: T(N)= t1 + t2*N

Siendo t1 el tiempo que lleve ejecutar la serie "S1" de sentencias, y t2 el que lleve la serie "S2".

Prácticamente todos los programas reales incluyen alguna sentencia condicional, haciendo que las sentencias efectivamente ejecutadas dependan de los datos concretos que se le presenten. Esto hace que más que un valor T(N) se deba hablar de un rango de valores

- Tmin(N) <= T(N) <= Tmax(N)

Los extremos son habitualmente conocidos como "caso peor" y "caso mejor". Entre ambos se hallara algún "caso promedio" o más frecuente.

Cualquier fórmula T(N) incluye referencias al parámetro N y a una serie de constantes "Ti" que dependen de factores externos al algoritmo como pueden ser la calidad del código generado por el compilador y la velocidad de ejecución de instrucciones del ordenador que lo ejecuta. Dado que es fácil cambiar de compilador y que la potencia de los ordenadores crece a un ritmo vertiginoso (en la actualidad, se duplica anualmente), se intentará analizar los algoritmos con algún nivel de independencia de estos factores; es decir, se buscará estimaciones generales ampliamente válidas.

6.3. ASÍNTOTAS

Por una parte se necesita analizar la potencia de los algoritmos independientemente de la potencia de la máquina que los ejecute e incluso de la habilidad del programador que los codifique. Por otra, este análisis interesa especialmente cuando el algoritmo se aplica a problemas grandes. Casi siempre los problemas pequeños se pueden resolver de cualquier forma, apareciendo las limitaciones al atacar problemas grandes. No debe olvidarse que cualquier técnica de ingeniería, si funciona, acaba aplicándose al problema más grande que sea

posible: las tecnologías de éxito, antes o después, acaban llevándose al límite de sus posibilidades.

Las consideraciones anteriores llevan a estudiar el comportamiento de un algoritmo cuando se fuerza el tamaño del problema al que se aplica. Matemáticamente hablando, cuando N tiende a infinito. Es decir, su comportamiento asintótico.

Sean "g(n)" diferentes funciones que determinan el uso de recursos. Habrá funciones "g" de todos los colores. Lo que se va a intentar es identificar "familias" de funciones, usando como criterio de agrupación su comportamiento asintótico.

A un conjunto de funciones que comparten un mismo comportamiento asintótico se le denominará un orden de complejidad'. Habitualmente estos conjuntos se denominan O, existiendo una infinidad de ellos.

Para cada uno de estos conjuntos se suele identificar un miembro f(n) que se utiliza como representante de la clase, hablándose del conjunto de funciones "g" que son del orden de "f(n)", denotándose como:

- g IN O (f(n))

Con frecuencia es posible encontrarse con que no es necesario conocer el comportamiento exacto, sino que basta conocer una cota superior, es decir, alguna función que se comporte "aún peor".

La definición matemática de estos conjuntos debe ser muy cuidadosa para involucrar ambos aspectos: identificación de una familia y posible utilización como cota superior de otras funciones menos malas:

Dícese que el conjunto O (f(n)) es el de las funciones de orden de f(n), que se define como:

- O(f(n))= { g: INTEGER -> $REAL^+$ tales que
  existen las constantes k y N_0 tales que
  para todo $N > N_0$, $g(N) <= k*f(N)$ }

En palabras, O (f(n)) está formado por aquellas funciones g(n) que crecen a un ritmo menor o igual que el de f(n).

De las funciones "g" que forman este conjunto O (f(n)) se dice que "están dominadas asintóticamente" por "f", en el sentido de que para N suficientemente grande, y salvo una constante multiplicativa "k", f(n) es una cota superior de g(n).

6.4. ÓRDENES DE COMPLEJIDAD

Se dice que O (f(n)) define un "orden de complejidad". Se escoge como representante de este orden a la función f(n) más sencilla del mismo. Así se tendrá:

- O (1) orden constante
- O (log n) orden logarítmico
- O (n) orden lineal
- O (n log n)
- O (n^2) orden cuadrático
- O (n^a) orden polinomial (a > 2)
- O (a^n) orden exponencial (a > 2)
- O (n!) orden factorial

Es más, se puede identificar una jerarquía de órdenes de complejidad que coincide con el orden anterior; jerarquía en el sentido de que cada orden de complejidad superior tiene a los inferiores como subconjuntos. Si un algoritmo A se puede demostrar de un cierto orden O, es cierto que también pertenece a todos los órdenes superiores (la relación de orden cota superior de' es transitiva); pero en la práctica lo útil es encontrar la "menor cota superior", es decir el menor orden de complejidad que lo cubra.

6.5. IMPACTO PRÁCTICO

Para captar la importancia relativa de los órdenes de complejidad conviene echar algunas cuentas.

Sea un problema que se sabe resolver con algoritmos de diferentes complejidades. Para compararlos entre sí, supóngase que todos ellos requieren 1 hora de ordenador para resolver un problema de tamaño N=100.

¿Qué ocurre si se dispone del doble de tiempo? Nótese que esto es lo mismo que disponer del mismo tiempo en un ordenador el doble de potente, y que el ritmo actual de progreso del hardware es exactamente ese:

"duplicación anual del número de instrucciones por segundo".

¿Qué ocurre si se quiere resolver un problema de tamaño 2n?

O(f(n))	N= 100	t= 2h	N= 200
log n	1 h	10000	1.15 h
n	1 h	200	2 h
n log n	1 h	199	2.30 h
n^2	1 h	141	4 h
n^3	1 h	126	8 h
2^n	1 h	101	10^{30} h

Los algoritmos de complejidad O(n) y O(n log n) son los que muestran un comportamiento más "natural": prácticamente a doble de tiempo, doble de datos procesables.

Los algoritmos de complejidad logarítmica son un descubrimiento fenomenal, pues en el doble de tiempo permiten atacar problemas notablemente mayores, y para resolver un problema el doble de grande sólo hace falta un poco más de tiempo (ni mucho menos el doble).

Los algoritmos de tipo polinómico no son una maravilla, y se enfrentan con dificultad a problemas de tamaño creciente. La práctica viene a decir que son el límite de lo "tratable".

Sobre la tratabilidad de los algoritmos de complejidad polinómica habría mucho que hablar, y a veces semejante calificativo es puro eufemismo. Mientras complejidades del orden $O(n^2)$ y $O(n^3)$ suelen ser efectivamente abordables, prácticamente nadie acepta algoritmos de orden $O(n^{100})$, por muy polinómicos que sean. La frontera es imprecisa.

Cualquier algoritmo por encima de una complejidad polinómica se dice "intratable" y sólo será aplicable a problemas ridículamente pequeños.

A la vista de lo anterior se comprende que los programadores busquen algoritmos de complejidad lineal. Es un golpe de suerte encontrar algo de complejidad logarítmica. Si se encuentran soluciones polinomiales, se puede vivir con ellas; pero ante soluciones de complejidad exponencial, más vale seguir buscando.

Más sin embargo de acuerdo a lo anterior:

- Si un programa se va a ejecutar muy pocas veces, los costes de codificación y depuración son los que más importan, relegando la complejidad a un papel secundario.
- Si a un programa se le prevé larga vida, hay que pensar que le tocará mantenerlo a otra persona y, por tanto, conviene tener en cuenta su legibilidad, incluso a costa de la complejidad de los algoritmos empleados.
- Si se puede garantizar que un programa sólo va a trabajar sobre datos pequeños (valores bajos de N), el orden de complejidad del algoritmo que se use suele ser irrelevante, pudiendo llegar a ser incluso contraproducente.

  Por ejemplo, si se dispone de dos algoritmos para el mismo problema, con tiempos de ejecución respectivos:

Algoritmo	**Tiempo**	**Complejidad**
f	100 n	O (n)
g	n^2	O (n^2)

  Asintóticamente, "f" es mejor algoritmo que "g"; pero esto es cierto a partir de N > 100.

Si el problema no va a tratar jamás problemas de tamaño mayor que 100, es mejor solución usar el algoritmo "g".

El ejemplo anterior muestra que las constantes que aparecen en las fórmulas para T(n), y que desaparecen al calcular las funciones de complejidad, pueden ser decisivas desde el punto de vista de ingeniería.

Pueden darse incluso ejemplos más dramáticos:

Algoritmo	**Tiempo**	**Complejidad**
f	n	O (n)
g	100 n	O (n)

Aún siendo dos algoritmos con idéntico comportamiento asintótico, es obvio que el algoritmo "f" es siempre 100 veces más rápido que el "g" y candidato primero a ser utilizado.

- Usualmente un programa de baja complejidad en cuanto a tiempo de ejecución, suele conllevar un alto consumo de memoria; y viceversa. A

veces hay que sopesar ambos factores, quedándose en algún punto de compromiso.

- En problemas de cálculo numérico hay que tener en cuenta más factores que su complejidad pura y dura, o incluso que su tiempo de ejecución: queda por considerar la precisión del cálculo, el máximo error introducido en cálculos intermedios, la estabilidad del algoritmo, etc. etc.

6.6. PROPIEDADES DE LOS CONJUNTOS O(F)

No se entrará en muchas profundidades, ni en demostraciones, que se pueden hallar en los libros especializados. No obstante, algo hay que saber de cómo se trabaja con los conjuntos O( ) para poder evaluar los algoritmos con los que se pueda encontrar.

Para simplificar la notación, usaremos O(f) para decir O(f(n))

Las primeras reglas sólo expresan matemáticamente el concepto de jerarquía de órdenes de complejidad:

A. La relación de orden definida por
f < g <=> f(n) IN O (g)
Es reflexiva: f(n) IN O(f)
Y transitiva: f(n) IN O(g) y g(n) IN O(h) => f(n) IN O(h)

B. f IN O (g) y g IN O (f) <=> O (f) = O (g)

Las siguientes propiedades se pueden utilizar como reglas para el cálculo de órdenes de complejidad. Toda la maquinaria matemática para el cálculo de límites se puede aplicar directamente:

C. $\text{Lim}_{(n->inf)}$ f(n)/g(n) = 0 => f IN O (g)
=> g NOT_IN O (f)
=> O (f) es subconjunto de O (g)

D. $\text{Lim}_{(n->inf)}$ f(n)/g(n) = k => f IN O (g)
=> g IN O (f)
=> O (f) = O (g)

E. $\text{Lim}_{(n->inf)}$ f(n)/g(n)= INF => f NOT_IN O (g)
=> g IN O (f)
=> O (f) es superconjunto de O(g)

Las que siguen son reglas habituales en el cálculo de límites:

F. Si f, g IN O(h) => f+g IN O(h)
G. Sea k una constante, f(n) IN O(g) => k*f(n) IN O(g)
H. Si f IN O(h1) y g IN O(h2) => f+g IN O(h1+h2)
I. Si f IN O(h1) y g IN O(h2) => f*g IN O(h1*h2)
J. Sean los reales 0 < a < b => $O(n^a)$ es subconjunto de $O(n^b)$
K. Sea P(n) un polinomio de grado k => P(n) IN $O(n^k)$
L. Sean los reales a, b > 1 => $O(\log_a) = O(\log_b)$

La regla [L] permite olvidar la base en la que se calculan los logaritmos en expresiones de complejidad.

La combinación de las reglas [K, G] es probablemente la más usada, permitiendo de un plumazo olvidar todos los componentes de un polinomio, menos su grado.

Por último, la regla [H] es la básica para analizar el concepto de secuencia en un programa: la composición secuencial de dos trozos de programa es de orden de complejidad el de la suma de sus partes.

6.7. REGLAS PRÁCTICAS

Aunque no existe una receta que siempre funcione para calcular la complejidad de un algoritmo, si es posible tratar sistemáticamente una gran cantidad de ellos, basándose en que suelen estar bien estructurados y siguen pautas uniformes.

Los algoritmos bien estructurados combinan las sentencias de alguna de las formas siguientes

1. Sentencias sencillas
2. Secuencia (;)
3. Decisión (if)
4. Bucles
5. Llamadas a procedimientos

Sentencias sencillas

Se refiere a las sentencias de asignación, entrada/salida, etc. siempre y cuando no trabajen sobre variables estructuradas cuyo tamaño esté relacionado con el

tamaño N del problema. La inmensa mayoría de las sentencias de un algoritmo requieren un tiempo constante de ejecución, siendo su complejidad O (1).

Secuencia (;)

La complejidad de una serie de elementos de un programa es del orden de la suma de las complejidades individuales, aplicándose las operaciones arriba expuestas.

Decisión (if)

La condición suele ser de O (1), complejidad a sumar con la peor posible, bien en la rama THEN, o bien en la rama ELSE. En decisiones múltiples (ELSE IF, SWITCH CASE), se tomara la peor de las ramas.

Bucles

En los bucles con contador explícito, se puede distinguir dos casos, que el tamaño N forme parte de los límites o que no. Si el bucle se realiza un número fijo de veces, independiente de N, entonces la repetición sólo introduce una constante multiplicativa que puede absorberse.

Ej: *for (int i= 0; i < K; i++) { algo_de_O(1) } => K * O(1) = O(1)*

Si el tamaño N aparece como límite de iteraciones:
Ej: *for (int i= 0; i < N; i++) { algo_de_O(1) } => N * O(1) = O(n)*

Ej:
```
for (int i= 0; i < N; i++)
    {
     for (int j= 0; j < N; j++)
        {
         algo_de_O(1)
        }
    }
```

Se tendrá N * N * O(1) = $O(n^2)$

Ej:
```
for (int i= 0; i < N; i++)
    {
      for (int j= 0; j < i; j++)
        {
          algo_de_O(1)
        }
     }
```

El bucle exterior se realiza N veces, mientras que el interior se realiza 1, 2, 3, ... N veces respectivamente. En total, 1 + 2 + 3 + ... + N = N*(1+N)/2 -> $O(n^2)$

A veces aparecen bucles multiplicativos, donde la evolución de la variable de control no es lineal (como en los casos anteriores)

Ej:
```
c= 1;
while (c < N)
      {
       algo_de_O(1)
       c= 2*c;
      }
```

El valor inicial de "c" es 1, siendo "2^k" al cabo de "k" iteraciones. El número de iteraciones es tal que:

$2^k >= N => k= eis (\log_2 (N))$ [el entero inmediato superior]

Y, por tanto, la complejidad del bucle es O (log n).

Ej:
```
c = N;
while (c > 1)
      {
        algo_de_O(1)
        c= c / 2;
      }
```

Un razonamiento análogo conlleva a $\log_2$ (N) iteraciones y, por tanto, a un orden O(log n) de complejidad.

Ej:
```
for (int i= 0; i < N; i++)
     {
      c= i;
      while (c > 0)
            {
             algo_de_O(1)
             c= c/2;
            }
     }
```

Se tiene un bucle interno de orden O (log n) que se ejecuta N veces, luego el conjunto es de orden O (n log n).

6.8. LLAMADAS A PROCEDIMIENTOS

La complejidad de llamar a un procedimiento viene dada por la complejidad del contenido del procedimiento en sí. El coste de llamar no es sino una constante que se puede obviar inmediatamente dentro de los análisis asintóticos.

El cálculo de la complejidad asociada a un procedimiento puede complicarse notablemente si se trata de procedimientos recursivos. Es fácil que se tenga que aplicar técnicas propias de la matemática discreta, tema que queda fuera de los límites de esta nota técnica.

Ejemplo: evaluación de un polinomio

Se va a aplicar lo explicado hasta ahora a un problema de fácil especificación:
Diseñar un programa para evaluar un polinomio P(x) de grado N;

```
class Polinomio
{
  private double[ ] coeficientes;

  Polinomio (double [ ] coeficientes)
   {
     this.coeficientes= new double [coeficientes.length];
     System.arraycopy (coeficientes, 0, this.coeficientes, 0,
                              coeficientes.length);
   }

  double evalua_1 (double x)
      {
       double resultado= 0.0;
       for (int termino= 0; termino < coeficientes.length; termino++)
           {
             double xn= 1.0;
             for (int j= 0; j < termino; j++)
             xn*= x;                          // x elevado a n
             resultado+= coeficientes[termino] * xn;
           }
       return resultado;
      }
}
```

Como medida del tamaño se tomará para N el grado del polinomio, que es el número de coeficientes en C. Así pues, el bucle más exterior (1) se ejecuta N veces. El bucle interior (2) se ejecuta, respectivamente:

- 1 + 2 + 3 + ... + N veces = N*(1+N)/2 => $O(n^2)$

Intuitivamente, sin embargo, este problema debería ser menos complejo, pues repugna al sentido común que sea de una complejidad tan elevada. Se puede ser más inteligente a la hora de evaluar la potencia x^n:

```
double evalua_2 (double x)
{
   double resultado= 0.0;
   for (int termino= 0; termino < coeficientes.length; termino++)
        {
          resultado+= coeficientes[termino] * potencia(x, termino);
        }
   return resultado;
}

private double potencia (double x, int n)
{
   if (n == 0)
      return 1.0;   // si es potencia impar
   if (n%2 == 1)
     return x * potencia(x, n-1);   // si es potencia par
   double t= potencia(x, n/2);
   return t*t;
 }
```

El análisis de la función Potencia es delicado, pues si el exponente es par, el problema tiene una evolución logarítmica; mientras que si es impar, su evolución es lineal. No obstante, como si "j" es impar entonces "j-1" es par, el caso peor es que en la mitad de los casos se tenga "j" impar y en la otra mitad sea par. El caso mejor, por contra, es que siempre sea "j" par.

Un ejemplo de caso peor sería x^{31}, que implica la siguiente serie para j:

- 31 30 15 14 7 6 3 2 1

Cuyo número de términos se puede acotar superiormente por 2 * eis ($\log_2(j)$), donde eis (r) es el entero inmediatamente superior (este cálculo responde al razonamiento de que en el caso mejor se visitará eis ($\log_2(j)$) valores pares de "j"; y en el caso peor se puede encontrar con otros tantos números impares entremezclados).

Por tanto, la complejidad de Potencia es de orden O (log n).

Insertada la función Potencia en la función EvaluaPolinomio, la complejidad compuesta es del orden O(n log n), al multiplicarse por N un subalgoritmo de O(log n).

Así y todo, esto sigue resultando extravagante y excesivamente costoso. En efecto, basta reconsiderar el algoritmo almacenando las potencias de "X" ya calculadas para mejorarlo sensiblemente:

```
double evalua_3 (double x)
{
   double xn= 1.0;
   double resultado= coeficientes [0];
   for (int termino= 1; termino < coeficientes.length; termino++)
       {
         xn*= x;
         resultado+= coeficientes[termino] * xn;
       }
   return resultado;
}
```

Que queda en un algoritmo de O(n).

Habiendo N coeficientes C distintos, es imposible encontrar ningún algoritmo de un orden inferior de complejidad.

En cambio, si es posible encontrar otros algoritmos de idéntica complejidad:

```
double evalua_4 (double x)
{
   double resultado= 0.0;
   for (int termino= coeficientes.length-1; termino >= 0; termino--)
       {
         resultado= resultado * x +
         coeficientes[termino];
       }
   return resultado;
}
```

No obstante ser ambos algoritmos de idéntico orden de complejidad, cabe resaltar que sus tiempos de ejecución serán notablemente distintos. En efecto, mientras el último algoritmo ejecuta N multiplicaciones y N sumas, el penúltimo requiere 2N multiplicaciones y N sumas. Si, como es frecuente, el tiempo de ejecución es notablemente superior para realizar una multiplicación, cabe razonar que el último algoritmo ejecutará en la mitad de tiempo que el anterior.

6.9. PROBLEMAS P, NP Y NP-COMPLETOS

Hasta aquí se ha venido hablando de algoritmos. Cuando se enfrenta a un problema concreto, habrá una serie de algoritmos aplicables. Se suele decir que el orden de complejidad de un problema es el del mejor algoritmo que se conozca para resolverlo. Así se clasifican los problemas, y los estudios sobre algoritmos se aplican a la realidad.

Estos estudios han llevado a la constatación de que existen problemas muy difíciles, problemas que desafían la utilización de los ordenadores para resolverlos. En lo que sigue se mencionan las clases de problemas que hoy por hoy se escapan a un tratamiento informático.

Clase P

Los algoritmos de complejidad polinómica se dice que son tratables en el sentido de que suelen ser abordables en la práctica. Los problemas para los que se conocen algoritmos con esta complejidad se dice que forman la clase P. Aquellos problemas para los que la mejor solución que se conoce es de complejidad superior a la polinómica, se dice que son problemas intratables. Sería muy interesante encontrar alguna solución polinómica (o mejor) que permitiera abordarlos.

Clase NP

Algunos de estos problemas intratables pueden caracterizarse por el curioso hecho de que puede aplicarse un algoritmo polinómico para comprobar si una posible solución es válida o no. Esta característica lleva a un método de resolución no determinista consistente en aplicar heurísticos para obtener soluciones hipotéticas que se van desestimando (o aceptando) a ritmo polinómico. Los problemas de esta clase se denominan NP (la N de no-deterministas y la P de polinómicos).

Clase NP-completos

Se conoce una amplia variedad de problemas de tipo NP, de los cuales destacan algunos de ellos de extrema complejidad. Gráficamente se puede decir que algunos problemas se hallan en la "frontera externa" de la clase NP. Son problemas NP, y son los peores problemas posibles de clase NP. Estos problemas se caracterizan por ser todos "iguales" en el sentido de que si se descubriera una solución P para alguno de ellos, esta solución sería fácilmente aplicable a todos ellos. Actualmente hay un premio de prestigio equivalente al Nobel reservado para el que descubra semejante solución (y se duda seriamente de que alguien lo consiga).

Es más, si se descubriera una solución para los problemas NP-completos, esta sería aplicable a todos los problemas NP y, por tanto, la clase NP desaparecería del mundo científico al carecerse de problemas de ese tipo. Realmente, tras años de búsqueda exhaustiva de dicha solución, es hecho ampliamente aceptado que no debe existir, aunque nadie ha demostrado, todavía, la imposibilidad de su existencia.

6.10. CONCLUSIONES

En esta sección se ha contemplado únicamente aquellos problemas para los que existe una solución algorítmica (el programa finaliza siempre, aunque necesite un número astronómico de operaciones elementales), y se ha dejado a un lado deliberadamente aquellos problemas para los cuales no existen algoritmos cuya finalización esté garantizada (problemas no–decidibles y semidecicibles), ya que en principio escapan al propósito de este documento.

El hecho de que no se conozca un algoritmo eficiente para resolver un problema no quiere decir que éste no exista, y por eso es tan importante la Teoría de Algoritmos para la Criptografía. Si, por ejemplo, se lograra descubrir un método eficiente capaz de resolver logaritmos discretos, algunos de los algoritmos asimétricos más populares en la actualidad dejarían de ser seguros. De hecho, la continua reducción del tiempo de ejecución necesario para resolver ciertos problemas, propiciada por la aparición de algoritmos más eficientes, junto con el avance de las prestaciones del hardware disponible, obliga con relativa frecuencia a actualizar las previsiones sobre la seguridad de muchos sistemas criptográficos.

7. DES, AES y RSA INICIO DE UNA NUEVA ERA

7.1. INTRODUCCIÓN

El crecimiento explosivo de la informática y la convergencia entre ordenadores y comunicaciones, hacen pensar que los algoritmos de cifrado corresponderán al código ejecutable más popular en el futuro. Tiene aplicaciones en los negocios por Internet, correo electrónico, dinero electrónico (tarjetas de crédito, tarjetas inteligentes), llaveros electrónicos, chips de seguridad contra robos de automóviles, registros públicos. Todas esas aplicaciones requieren el cifrado de cierta información. Más aún, en un mundo donde todos los ordenadores están interconectados entre sí y todo puede ser accedido por todos, los sistemas operativos del futuro cifrarán todos los ficheros que guardan en el disco duro de manera automática para una mayor seguridad y eficacia.

Existen dos tipos fundamentales de criptosistemas:

Criptosistemas simétricos o de clave privada: Son aquéllos que emplean la misma clave **k** tanto para cifrar como para descifrar. Presentan el inconveniente de que, para ser empleados en comunicaciones, la clave **k** debe estar tanto en el emisor como en el receptor, lo cual plantea la situación de elegir cómo transmitir la clave de forma segura.

Criptosistemas asimétricos o de clave pública: Estos emplean una doble clave (k_p, k_P), k_p se conoce como clave privada y k_P se conoce como clave pública. Una de ellas sirve para la transformación de cifrado y la otra para la transformación de descifrado. En muchos casos son intercambiables, esto es, si se emplea una para cifrar la otra sirve para descifrar y viceversa. Estos criptosistemas deben cumplir además que el conocimiento de la clave pública k_p no permita calcular la clave privada k_P. Ofrecen un abanico superior de posibilidades, pudiendo emplearse para establecer comunicaciones seguras por canales inseguros puesto que únicamente viaja por el canal la clave pública, o para llevar a cabo autentificaciones.

En la práctica se emplea una combinación de estos dos tipos de criptosistemas, puesto que los segundos presentan el inconveniente de ser computacionalmente mucho más costosos que los primeros. En el mundo real se codifican los mensajes (largos) mediante algoritmos simétricos, que suelen ser muy eficientes,

y luego se hace uso de la criptografía asimétrica para codificar las claves simétricas (cortas).

Los algoritmos de cifrado simétrico más modernos son una combinación de los cifradores de substitución y de transposición o permutación, que se corresponden con los conceptos de confusión y difusión introducidos por Claude Shannon. El principal objetivo de la confusión es esconder la relación entre el texto claro, el texto cifrado y la clave y, el de la difusión trata de propagar la información real en el mensaje cifrado. La aplicación sucesiva de un cierto número de tales criptosistemas simples puede dar origen a un sistema criptográfico iterado (cifrador producto) suficientemente seguro. La mayoría de los criptosistemas se basan en diferentes capas de substituciones y permutaciones, estructura denominada SPN (Substitution-Permutation Networks), redes de substitución – permutación y, en particular algoritmos como el DES y el AES.

En 1978 Rivest, Shamir y Adleman descubrieron el primer esquema de encriptación de llave pública y un procedimiento de firmas que ahora se conoce como RSA, este método se basa en un problema matemático que provee infalibilidad (inflexibilidad para resolverlo), se refiere al problema de factorización de enteros largos, lo cual hace que su fuerte no sea el algoritmo como tal, sino sus bases matemáticas.

El conocimiento de la estructura interna de los mecanismos de encriptación hace posible elegir el más adecuado a las necesidades de la aplicación a utilizar, el objetivo de este capítulo es analizar 3 de los métodos más comunes y utilizados en la seguridad de la información, a fin de poner de manifiesto sus ventajas y debilidades y abrir la mente a nuevos mecanismos y algoritmos de cifrado de datos.

7.2. CIFRADOR DE BLOQUE TIPO FEISTEL

En criptografía, el Cifrado de Feistel ó red de Feistel es un método de cifrado en bloque con una estructura particular. Debe su nombre al criptógrafo de IBM Horst Feistel. También es conocida comúnmente como Red de Feistel. Un gran número de algoritmos de cifrado por bloques lo utilizan, siendo el más conocido el algoritmo Data Encryption Standard (***DES***), el cual se describirá en detalle en el siguiente punto. Las redes de Feistel presentan la ventaja de ser reversibles por lo

que las operaciones de cifrado y descifrado son idénticas, requiriendo únicamente invertir el orden de las subclaves utilizadas.

El primer algoritmo basado en las redes de Feistel fue el algoritmo Lucifer, diseñado al amparo de IBM por Horst Feistel y Don Coppersmith a principios de la década del 1970, aunque la popularidad para este esquema llegó cuando el Gobierno Federal de los Estados Unidos adoptó el algoritmo ***DES*** como estándar para el cifrado de las comunicaciones gubernamentales. Este algoritmo derivaba del algoritmo Lucifer y también está constituido por una red de Feistel.

Este algoritmo se denomina simétrico por rondas, es decir, realiza siempre las mismas operaciones un número determinado de veces (denominadas rondas):

Este método viene regido por las siguientes condiciones:

- Dado un bloque de N bits (típico 64) éste se dividirá en dos mitades.
- Existirá una función unidireccional F (muy difícil de invertir).
- Se realizan operaciones con la clave k_i sólo con una mitad del bloque, y se permutan en cada vuelta las dos mitades, operación que se repite durante n vueltas.

Ejemplo básico:

Para el siguiente ejemplo el algoritmo usará bloques de tamaño 8 caracteres, cada bloque se dividirá en dos, tendrá dos vueltas y en cada vuelta realizará una operación de sustitución sencilla S y una permutación P sobre la 1ª mitad.

Sustitución: $S_i = (M_i + 1) \bmod 27$

Permutación: $P_i = \prod_{3241}$ (El carácter 1º pasa a la 4ª posición en el criptograma, el 4º a la 3ª, el 2º a la 2ª y el 3º a la 1ª).

Por ejemplo: ABCD, luego de la permutación, CBDA.

Se habrá de utilizar el mensaje:

M = STAR WARS, LA MISIÓN CONTINÚA

Primero que todo se divide en grupos de 4 letras el mensaje:

Cuadro 43. Ejemplo 01 de Feistel

BLOQUE 1		BLOQUE 2		BLOQUE 3	
STAR	WARS	**LAMI**	SION	**CONT**	INUA

Para la función de sustitución se tendrá en cuenta la siguiente correspondencia:

Tabla 13. Equivalencia numeración letras

A	**B**	**C**	**D**	**E**	**F**	**J**	**H**	**I**	**J**	**K**	**L**	**M**	**N**
01	02	03	04	05	06	07	08	09	10	11	12	13	14

Ñ	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
15	16	17	18	19	20	21	22	23	24	25	26	27

Primera vuelta:

Sustitución: $S_i = (M_i + 1) \bmod 27$

STAR: M_1 (S=20), M_2 (T=21), M_3 (A=01), M_4 (R=19).

S_1: (20+1) mod 27 = 21 = T

S_2: (21+1) mod 27 = 22 = U

S_3: (01+1) mod 27 = 02 = B

S_4: (19+1) mod 27 = 20 = S

LAMI: M_1 (L=12), M_2 (A=01), M_3 (M=13), M_4 (I=09).

S_1: (12+1) mod 27 = 13 = M

S_2: (01+1) mod 27 = 02 = B

S_3: (13+1) mod 27 = 14 = N

S_4: (09+1) mod 27 = 10 = J

CONT: M_1 (C=03), M_2 (O=16), M_3 (N=14), M_4 (T=21).

S_1: (03+1) mod 27 = 04 = D

S_2: (16+1) mod 27 = 17 = P

S_3: (14+1) mod 27 = 15 = Ñ

S_4: (21+1) mod 27 = 22 = U

Cuadro 44. Ejemplo Permutación.

$P_i = \Pi_{3241}$

	BLOQUE 1		**BLOQUE 2**		**BLOQUE 3**	
M1:	STAR	WARS	LAMI	SION	CONT	INUA
S1:	TUBS	WARS	MBNJ	SION	DPÑU	INUA
P1:	BUST	WARS	NBJM	SION	ÑPUD	INUA

	BLOQUE 1		**BLOQUE 2**		**BLOQUE 3**	
M2:	WARS	STAR	SION	LAMI	INUA	CONT
S2:	XBST	TUBS	TJPÑ	MBNJ	JÑVB	DPÑU
P2:	SBTX	BUST	PJÑT	NBJM	VÑBJ	ÑPUD

Nótese que se intercambian los bloques y posteriormente se aplica el mismo procedimiento de la primera vuelta a aquellos bloques que no habían sido cambiados, de esta manera quedan procesados todos los bloques y a su vez quedan intercambiados entre ellos, todo esto con apenas 2 vueltas ó iteraciones.

Mensaje cifrado: SBTX BUST PJÑT NBJM VÑBJ ÑPUD

Aunque parezca exagerado ó increíble, el ***DES*** hace prácticamente lo mismo trabajando con bits y con funciones un poco más “complejas”, además de que para cada bloque de 64 bits lo realiza 16 veces, 16 vueltas ó 16 iteraciones, como mejor lo entienda.

7.3. ALGORITMO DE CIFRADO DES

7.3.1. Introducción

Es el algoritmo simétrico más extendido mundialmente. Se basa en el algoritmo LUCIFER, que había sido desarrollado por IBM a principios de los setenta, DES (*Data Encryption Standard*, estándar de cifrado de datos) es un algoritmo desarrollado originalmente por IBM a requerimiento del NBS (National 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 EE.UU. y posteriormente modificado y adoptado por el gobierno de EE.UU. en 1977 como estándar de cifrado de todas las informaciones sensibles no clasificadas. Posteriormente, en 1980, el NIST estandarizó los diferentes modos de operación del algoritmo. Es el más estudiado y utilizado de los algoritmos de clave simétrica.

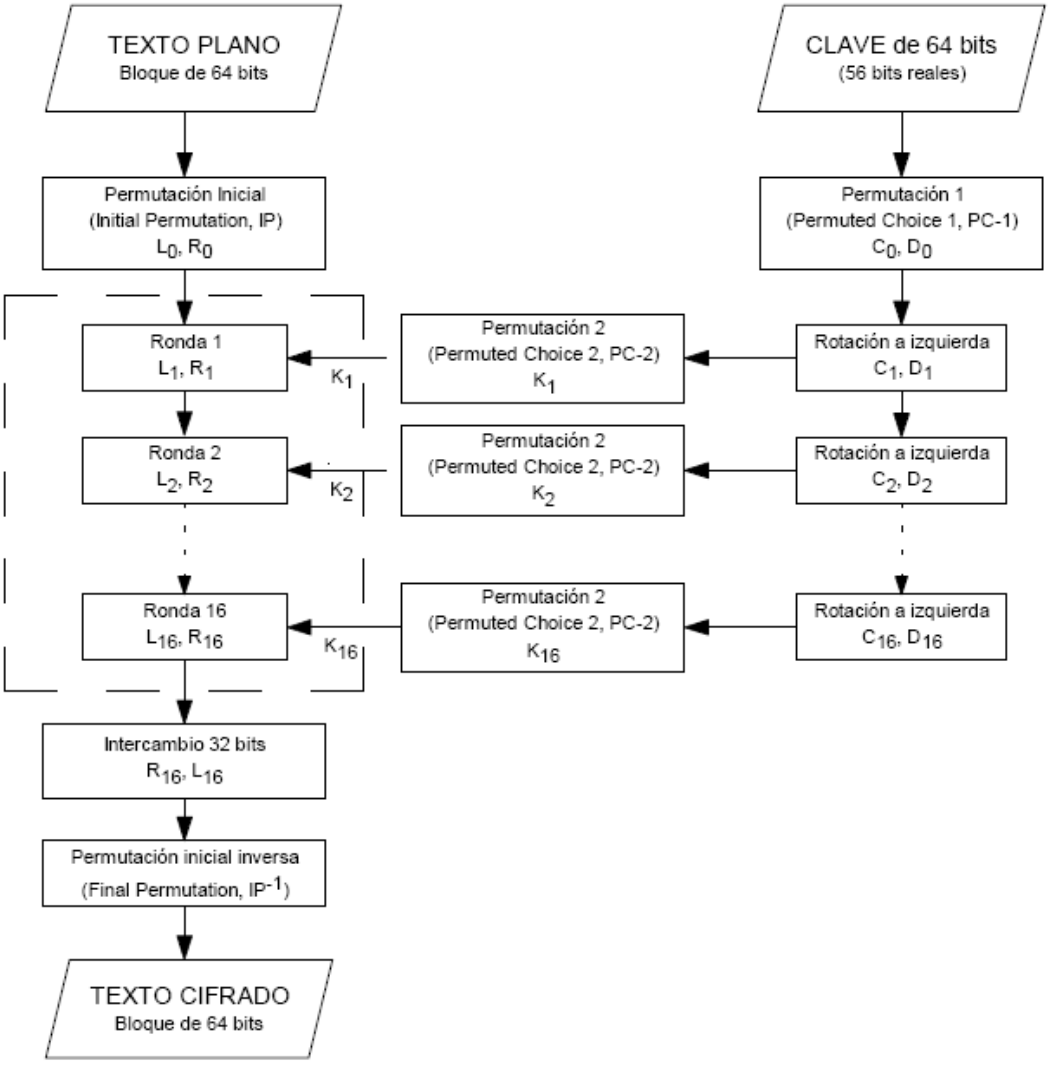
En realidad la NSA lo diseñó para ser implementado por hardware, creyendo que los detalles iban a ser mantenidos en secreto, pero la Oficina Nacional de Estandarización publicó su especificación con suficiente detalle como para que cualquiera pudiera implementarlo por software. No fue casualidad que el siguiente algoritmo adoptado (Skipjack) fuera mantenido en secreto.

En 1987 la NSA se opone a que se siga manteniendo como estándar pero, por motivos económicos, finalmente se renueva.

A mediados de 1997, se demostró que un ataque por la fuerza bruta a DES era viable, debido a la escasa longitud que emplea en su clave. En el mismo año tras 4 meses de cálculo se descifra un mensaje cifrado con el DES. Hoy el record es de 23 horas. Es sustituido a partir del 2000 por el AES (Algoritmo que se estudiará más adelante). No obstante, el algoritmo aún no ha demostrado ninguna debilidad grave desde el punto de vista teórico, por lo que su estudio sigue siendo plenamente interesante1.

Un breve resumen del esquema de funcionamiento del algoritmo ***DES*** que posteriormente se describirá en detalle, es el siguiente:

Figura 9. Esquema general del algoritmo DES



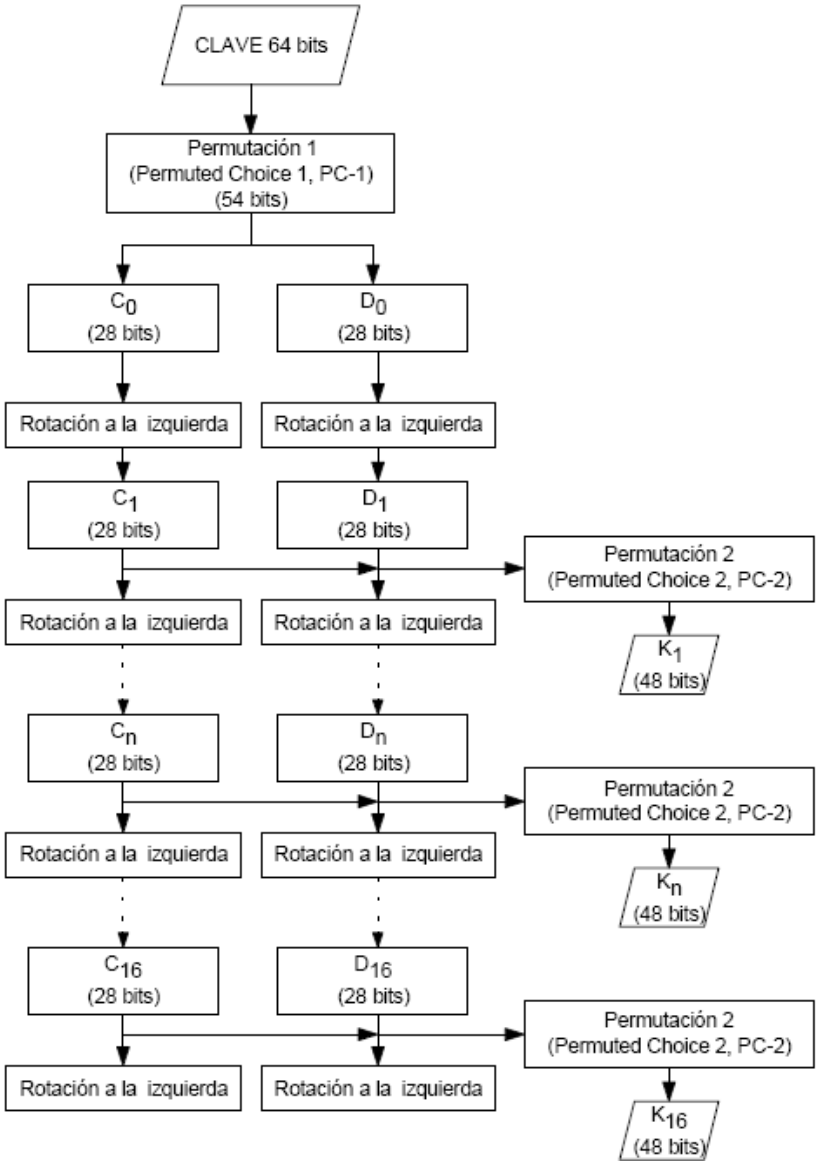
Fuente: Jorge Sánchez Arriazu, descripción del algoritmo DES, p. 2

Como puede verse en el esquema (figura 9), Data Encryption Standard (DES), es un algoritmo de cifrado en bloque simétrico, de longitud fija, el cual consiste de dos permutaciones, 16 vueltas en donde el mensaje de 64 bits es dividido en dos bloques de 32 bits, en este punto se aplica una Red de Feistel de 16 rondas, después de usar la primer permutación llamada P1, es cifrado 16 veces utilizando cada vez una subclave, la cual se genera 16 veces en un proceso paralelo y por ultimo realizar un ultima permutación que me entregara el texto cifrado.

En el proceso para descifrar se utiliza el mismo algoritmo con las subclaves en orden inverso, dando como consecuencia, la simetría del algoritmo.

En cada una de las 16 iteraciones se emplea un valor, Ki, obtenido a partir de la clave de 56 bits y distinto en cada iteración.

Figura 10. Calculo de las subclaves K_i.



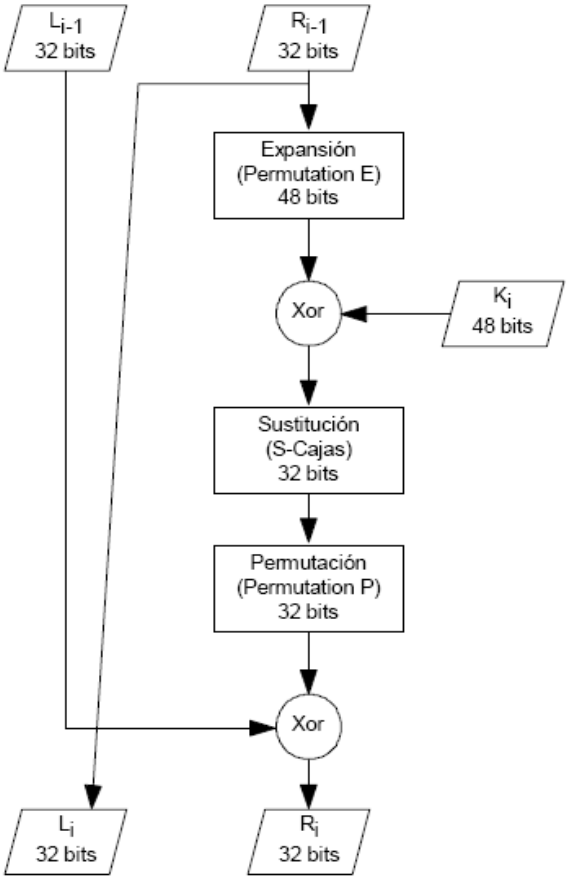
Fuente: Jorge Sánchez Arriazu, descripción del algoritmo DES, p. 3

Se realiza una permutación inicial (PC-1) sobre la clave, y luego la clave obtenida se divide en dos mitades de 28 bits, cada una de las cuales se rota a izquierda un número de bits determinado que no siempre es el mismo. K_i se deriva de la elección permutada (PC-2) de 48 de los 56 bits de estas dos mitades rotadas. Todo este proceso se explicara de manera detallada más adelante.

Para descifrar basta con usar el mismo algoritmo empleando las Ki en orden inverso.

La función f de la red de Feistel se compone de una permutación de expansión (E), que convierte el bloque correspondiente de 32 bits en uno de 48. Después realiza una or-exclusiva con el valor Ki, también de 48 bits, aplica ocho S-Cajas de 6*4 bits, y efectúa una nueva permutación (P).

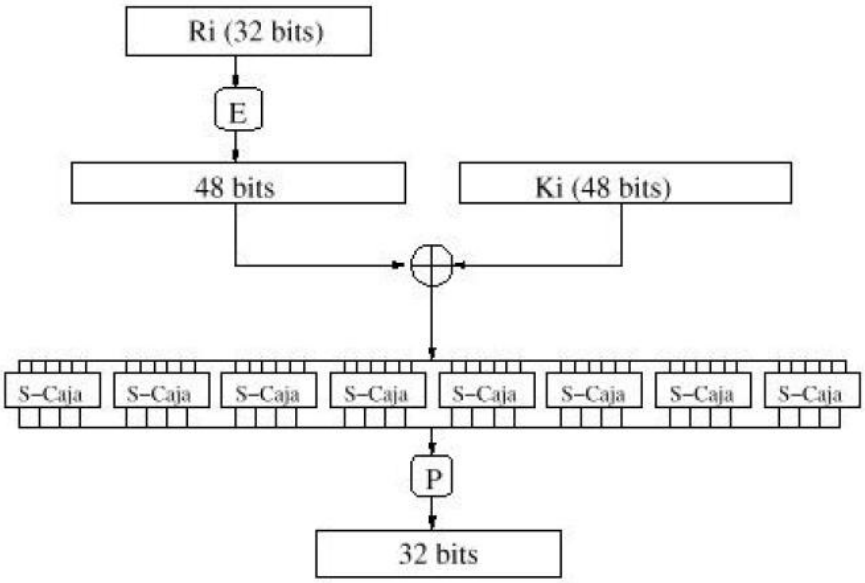
Figura 11. Ronda de la función f en el DES.



Fuente: Jorge Sánchez Arriazu, descripción del algoritmo DES, p. 4

Como puede verse en la figura 11, hay un bloque llamado “Sustitución (S-Cajas) 32 bits”, en general, una S-Caja ó Caja S lo que hace es tomar un número *m* de bits de entrada y los transforma en *n* bits de salida a través de una correspondencia de bits establecida.

Figura 12. Ronda de las cajas S.



Fuente: Jorge Sánchez Arriazu, descripción del algoritmo DES, p. 6

En el algoritmo del ***DES***, la caja-S, maneja 8 cajas-S internas, cada una con diferente codificación; en resumen, lo que se hace es recibir 48 bits como entrada, repartirlos en grupos de 6 bits para cada una de las 8 cajas-S internas, las cuales entregan como salida 4 bits mediante una pequeña correspondencia entre los bits externos e internos de cada grupo de 6 bits (Explicados en detalle más adelante), al juntar las salidas de las 8 cajas-S, se obtienen 32 bits que son entregados como salida. En las cajas-S se logra la fortaleza del algoritmo, ya que es una función unidireccional y no lineal.

7.3.2. Descripción paso a paso de algoritmo DES

7.3.2.1. Permutación inicial P1

DES, es un algoritmo de cifrado en bloque simétrico, para este caso el bloque de tamaño fijo es de 64 bits, como se menciono antes, el algoritmo cuenta con 2 tipos de permutaciones principales (P1, $P1^{-1}$), la primera P1 está dada por la siguiente tabla:

Tabla 14. Tabla base permutación P1:

01	**02**	03	**04**	05	**06**	07	**08**
09	**10**	11	**12**	13	**14**	15	**16**
17	**18**	19	**20**	21	**22**	23	**24**
25	**26**	27	**28**	29	**30**	31	**32**
33	**34**	35	**36**	37	**38**	39	**40**
41	**42**	43	**44**	45	**46**	47	**48**
49	**50**	51	**52**	53	**54**	55	**56**
57	**58**	59	**60**	61	**62**	63	**64**

Para obtener la tabla de permutación P1, las columnas de la tabla base, se convertirán en las filas de la siguiente pero leyéndolas de abajo hacia arriba, además, lo que comienza a hacer interesante esta tabla, es que las primeras filas están conformadas por las columnas sombreadas en la tabla anterior y las segundas cuatro filas, son las columnas restantes, como se muestra a continuación:

Tabla 15. Tabla permutación P1:

Li	**58**	**50**	**42**	**34**	**26**	**18**	**10**	**02**
	60	**52**	**44**	**36**	**28**	**20**	**12**	**04**
	62	**54**	**46**	**38**	**30**	**22**	**14**	**06**
	64	**56**	**48**	**40**	**32**	**24**	**16**	**08**
Ri	57	49	41	33	25	17	09	01
	59	51	43	35	27	19	11	03
	61	53	45	37	29	21	13	05
	63	55	47	39	31	23	15	07

En cada parte del algoritmo, básicamente se trabaja bloques de 32 bits c/u, por tal motivo la primera iteración viene dada por la tabla anterior.

Nótese como en la parte superior quedan los números pares y en la parte inferior los impares y al separar cada sub-tabla y leer las columnas de arriba abajo, cada número aumenta en 2 unidades respecto al anterior.

Después de recibir un bloque de entrada (parte del texto a cifrar) de 64 bits, el primer paso consiste en aplicar a este la permutación P_1, teniendo como resultado un orden de salida que se identifica leyendo la tabla de izquierda a derecha, en

donde el bit número 58 en el mensaje de entrada, después de la permutación, ocupara la posición 1 en el de salida y así sucesivamente.

Después de esta primera permutación P_1, los 64 bits de la salida se dividen en 2 sub-bloques de 32 bits cada uno (Left - Right) (Li - Ri) respectivamente, en donde Li serán los primeros 32 bits del bloque de salida y Ri serán 32 restantes.

Ejemplo:

Supóngase que se tiene el siguiente mensaje:

"*Necesito los números de Sannden y Tamo los cocineros chinos de ayer*".

8 caracteres = 8 bytes = 64 bits, por tal motivo debe dividirse el mensaje en grupos de 8 letras:

Cuadro 45. Ejemplo 01 DES

N	e	c	e	s	i	t	o
l	o	s	n	u	m	e	r
o	s	d	e	s	a	n	n
d	**e**	**n**	**y**	**t**	**a**	**m**	**o**
l	o	s	c	o	c	i	n
e	r	o	s	c	h	i	n
o	s	d	e	a	y	e	r

Si un bloque formado por un mensaje, no tiene la longitud de 64 bits, se rellena utilizando ceros al final.

Vamos a utilizar el octeto del medio, es decir el cuarto octeto (*denytamo*) para explicar todo el proceso de cifrado del DES.

En la siguiente tabla se muestra el carácter (en orden alfabético), su notación decimal y su equivalente en binario (que es con el que trabaja el DES).

Cuadro 46. Equivalencia decimal, carácter, binario

Carácter	Decimal	Binario
a	97	01100001
D	68	01000100
e	101	01100101
m	109	01101101
n	110	01101110
o	111	01101111
t	116	01110100
y	121	01111001

De esta manera, la permutación es de la siguiente manera:

Cuadro 47. Ejemplo 02 DES

Tabla inicial:

D	0	**1**	0	**0**	0	**1**	0	**0**
E	0	**1**	1	**0**	0	**1**	0	**1**
N	0	**1**	1	**0**	1	**1**	1	**0**
Y	0	**1**	1	**1**	1	**0**	0	**1**
T	0	**1**	1	**1**	0	**1**	0	**0**
A	0	**1**	1	**0**	0	**0**	0	**1**
M	0	**1**	1	**0**	1	**1**	0	**1**
O	0	**1**	1	**0**	1	**1**	1	**1**

Permutación P1:

1	**1**	**1**	**1**	**1**	**1**	**1**	**1**
0	**0**	**0**	**1**	**1**	**0**	**0**	**0**
1	**1**	**0**	**1**	**0**	**1**	**1**	**1**
1	**1**	**1**	**0**	**1**	**0**	**1**	**0**
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	0
1	1	0	0	1	1	0	0
1	0	0	0	0	1	0	0

Se muestra en la tabla de permutación P1 la parte superior sombreada y la inferior no, para resaltar los dos sub-bloques L0 y R0, dando como resultado:

L0: 11111111 00011000 11010111 11101010

R0: 00000000 11111110 11001100 10000100

A partir de este punto se aplicaran los siguientes pasos hasta el numeral **7.3.2.10,** al finalizar este, nuevamente se retorna al paso que hay a continuación y así de esta manera, hasta completar 16 vueltas o rondas.

7.3.2.2. Permutación E

En esta parte se trabajará con el sub-bloque R0 (32 bits); esta segunda parte viene dada por la permutación E, la cual es la encargada de expandir el bloque de 32 bits a 48 bits, como se muestra a continuación:

Tabla 16. Tabla base permutación E

01	02	03	**04**
05	06	07	**08**
09	10	11	**12**
13	14	15	**16**
17	18	19	**20**
21	22	23	**24**
25	26	27	**28**
29	30	31	**32**

La parte sombreada muestra los bits que se habrán de duplicar, este es el método con el que se expandirá a 48 bits. Se añade de a una columna a cada extremo de la tabla base, los bits se ubicaran colocando el bit 01 duplicado en la última posición y el bit 32 duplicado en la primera, luego de esto los cuadros en blanco se rellenaran con los bits duplicados restantes que corresponden a las columnas de 01 y del 32 respectivamente:

Tabla 17. Tabla Permutación E:

32	01	02	03	04	05
04	05	06	07	08	09
08	09	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	**01**

Ahora se procede a aplicar la permutación E al sub-bloque R0:

R0 = 0000 0000 1111 1110 1100 1100 1000 0100

Cuadro 48. Ejemplo 03 DES

Tabla Inicial:

0	0	0	0
0	0	0	0
1	1	1	1
1	1	1	0
1	1	0	0
1	1	0	0
1	0	0	0
0	1	0	0

Permutación E:

0	0	0	0	0	0
0	0	0	0	0	1
0	1	1	1	1	1
1	1	1	1	0	1
0	1	1	0	0	1
0	1	1	0	0	1
0	1	0	0	0	0
0	0	1	0	0	**0**

El resultado de la permutación E(R0) (48 bits) es:

E(R0) = 000000 000001 011111 111101 011001 011001 010000 001000

Con esto finaliza la segunda parte del algoritmo, ahora se procede al cálculo de las subclaves.

7.3.2.3. Generación de la subclave Ki

La clave Ki tiene un valor inicial de 64 bits, este valor es de longitud fija para todo tipo de clave que se desee ingresar.

Tabla 18. Tabla base generación subclave Ki

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

Se elimina la última columna, la de los múltiplos de 8, es decir, se eliminará el octavo bit (el menos significativo) de cada octeto que representa una carácter de la clave, quedando así una tabla de 56 bits:

Tabla 19. Tabla subclave de 56 bits

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

Las columnas de parte sombreada representan las filas de la primera mitad (28 bits) de la permutación PC_1, las cuales son leídas de abajo hacia arriba. Las columnas que no están sombreadas representan los 28 bits restantes de la segunda mitad, leídos igualmente de abajo hacia arriba, pero comenzando por el último bit de la tabla, de la siguiente manera:

Tabla 20. Tabla permutación PC_1

Ci	**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**
Di	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

La tabla de la Permutación PC_1, se utiliza para realizar la permutación inicial en la generación de la subclave Ki (Se utiliza una sola vez). Una vez realizada la permutación, los 56 bits se dividen en dos sub-bloques (Ci - Di) de 28 bits c/u.

Para continuar con el ejemplo de cifrado del DES, se va a utilizar la clave K= “Santiago”.

En la siguiente tabla se muestra el carácter (en orden alfabético), su notación decimal y su equivalente en binario (que es con el que trabaja el DES).

Cuadro 49. Ejemplo 04 DES

Carácter	Decimal	Binario
a	97	01100001
g	103	01100111
i	105	01101001
n	110	01101110
o	111	01101111
S	83	01010011
t	116	01110100

De esta manera, la permutación es de la siguiente manera:

Cuadro 50. Ejemplo 05 DES

Tabla inicial:

S	0	1	0	1	0	0	1	**1**
A	0	1	1	0	0	0	0	**1**
N	0	1	1	0	1	1	1	**0**
T	0	1	1	1	0	1	0	**0**
I	0	1	1	0	1	0	0	**1**
A	0	1	1	0	0	0	0	**1**
G	0	1	1	0	0	1	1	**1**
O	0	1	1	0	1	1	1	**1**

Tabla 56 bits:

0	**1**	**0**	1	0	0	1
0	**1**	**1**	0	0	0	0
0	**1**	**1**	0	1	1	1
0	**1**	**1**	1	0	1	0
0	**1**	**1**	**0**	1	0	0
0	**1**	**1**	**0**	0	0	0
0	**1**	**1**	**0**	0	1	1
0	**1**	**1**	**0**	1	1	1

Permutación PC_1:

0	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**1**	**1**	**1**	**1**	**1**
1	**1**	**1**	**1**	**1**	**1**	**1**
1	**1**	**0**	**0**	**0**	**0**	**0**
1	1	0	0	0	1	0
1	1	1	0	0	1	1
0	0	1	0	0	1	0
1	0	0	1	0	0	1

Sub-bloques iniciales:

C0: 0000000 0111111 1111111 1100000

D0: 1100010 1110011 0010010 1001001

A estos bloques aun les falta aplicarles el desplazamiento de bits, que se explica a continuación.

7.3.2.4. Desplazamiento LS

Otro paso que añade complejidad al DES es el desplazamiento a la izquierda de manera circular, de 1 o 2 bits; este desplazamiento viene regido por la siguiente

tabla, cuya correspondencia está dada, por el número de bits a desplazar, de acuerdo a la vuelta o ronda en que se encuentre la subclave; que como ya se menciono son 16 en total:

Tabla 21. Tabla LS_i

Vuelta	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16
No. bits	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Como se aplica un desplazamiento de 28 bits en total en cada bloque de clave al finalizar las 16 vueltas, entonces $D_{16} = D_0$ y $C_{16} = C_0$.

Al tener los dos sub-bloques C0 y D0, se procede a aplicar el desplazamiento LS_1, se marca el primer bit, para entender el desplazamiento:

C0: **0**000000 0111111 1111111 1100000

LS_1(C0)= 0000000 1111111 1111111 100000**0**

D0: **1**100010 1110011 0010010 1001001

LS_1(D0)= 1000101 1100110 0100101 001001**1**

Al aplicar el desplazamiento se obtiene como resultado C1 y D1:

C1: 0000000 1111111 1111111 1000000

D1: 1000101 1100110 0100101 0010011

Con este par de sub-bloques se aplicara la permutación PC2, explicada a continuación.

7.3.2.5. Permutación PC2

La permutación PC2 se conoce como permutación de compresión, dada por las operaciones de concatenar y permutar Ci y Di, se va a comprimir de 56 a 48 bits para obtener la clave Ki, la cual posteriormente será utilizada en la función: **(f (R_{i-1}, K_i)).**

El orden de concatenar C_i y D_i, es utilizando primero los bits de Ci y luego los de Di, la tabla PC2 es una tabla de 8 x 6, dando como resultado 8 bloques de 6 bits:

Tabla 22. Tabla (Ci, Di)

01	02	03	04	05	06	07
08	**09**	10	11	12	13	14
15	16	17	**18**	19	20	21
22	23	24	**25**	26	27	28
29	30	31	32	33	34	**35**
36	37	**38**	39	40	41	42
43	44	45	46	47	48	49
50	51	52	53	**54**	55	56

Para obtener la tabla de permutación PC2 se eliminan los bits 9,18, 22, 25, 35, 38, 43, 54, de esta manera queda una tabla de 48 bits:

Tabla 23. Tabla Permutación PC2

14	17	11	24	01	05
03	28	15	06	21	10
23	19	12	04	26	08
16	07	27	20	13	02
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

En el camino de añadir más complejidad al DES se genera esta tabla; nótese que en este caso en particular, no se sigue ningún orden para generar la permutación PC2, este es el resultado de organizar los bits de manera aleatoria (Orden especifico definido por los creadores del DES, uno de los puntos base que hace tan fuerte este algoritmo).

Cabe destacar que este orden obtenido es fijo para el DES, es decir, esto significa que la permutación PC2 **NO** cambia en ningún momento.

Ejemplo:

Continuando con el ejemplo, se muestra la manera de aplicar la permutación PC2 a los sub-bloques C1 y D1, para obtener la clave K1:

C1: 0000000 1111111 1111111 1000000

D1: 1000101 1100110 0100101 0010011

Concatenados: (C1, D1)= Los 56 bits de entrada inicial a la Permutación PC2.

0000000 1111111 1111111 1000000 1000101 1100110 0100101 0010011

Cuadro 51. Tabla (C1, D1)

0	0	0	0	0	0	0
1	**1**	1	1	1	1	1
1	1	1	**1**	1	1	1
1	0	0	**0**	0	0	0
1	0	0	0	1	0	**1**
1	1	**0**	0	1	1	0
0	1	0	0	1	0	1
0	0	1	0	**0**	1	1

Se muestran resaltados los bits que se han de eliminar para comprimir a 48 bits, posteriormente se ubican los bits restantes en la tabla de permutación PC2:

Cuadro 52. Permutación PC2

1	1	1	0	0	0
0	0	1	0	1	1
0	1	1	0	0	1
1	0	0	1	1	0
1	1	0	1	1	1
0	1	0	0	1	0
1	1	0	1	0	0
0	0	0	1	1	0

El resultado de haber realizado la permutación PC2 es la generación de la clave K_1 siendo: **$K_1 = PC2\ (C_1, D_1)$.**

K1: 111000 001011 011001 100110 110111 010010 110100 000110

En estas condiciones, la clave Ki (48 bits), está definida por las ecuaciones:

Ci = *LS* (Ci-1) **Di =** *LS* (Di -1)

Ki = PC2 (Ci , Di)

Las operaciones *LS*(i) y PC2, se repiten 15 veces más, para así obtener las 16 subclaves de cifrado totales, pero antes de volver a realizar estas operaciones, hay que seguir tres pasos más.

7.3.2.6. La función (f (R_{i-1}, K_i))

Al tener la clave K1 y la expansión de (R0), el siguiente paso es la función $f(R_{i-1}, K_i)$, la cual consta de tres procesos (Suma OR exclusivo, ocho funciones no lineales, Permutación P), siendo las ocho funciones lineales (Cajas-S) la mayor virtud del algoritmo. A continuación se explican los procesos.

7.3.2.7. Suma OR exclusivo

Con la clave K1 y E(R0), se procede a realizar la suma OR exclusiva, con esto, se tiene un 50% de certeza que el siguiente bit sea 1 o 0, con lo cual aumenta la dificultad de poder descifrar el mensaje, se hace mas fuerte el DES:

La operación OR exclusiva está definida en la siguiente tabla:

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Retomando la clave K1 y la expansión de R0, se procede a realizar la suma OR exclusiva entre ambos:

K1: 111000 001011 011001 100110 110111 010010 110100 000110

E(R0) = 000000 000001 011111 111101 011001 011001 010000 001000

K1 ⊕ E(R0): 111000 001010 000110 011011 101110 001011 100100 001110

7.3.2.8. Funciones no lineales (Cajas-S)

En criptografía, una Caja-S (**S-Box)** es un componente básico de los algoritmos de cifrado de clave simétrica. En los cifradores por bloques son usadas a menudo para oscurecer la relación existente entre texto plano y texto cifrado (la propiedad confusión de Shannon). En muchos casos, por ejemplo el DES, las Cajas-S son elegidas cuidadosamente para ser resistentes al criptoanálisis.

En general, una Caja-S toma un número *m* de bits de entrada y lo transforma en *n* bits de salida. El DES usa tablas prefijadas o predefinidas, las cuales fueron objeto de intensivo estudio durante años con la intención de localizar una puerta trasera, una vulnerabilidad conocida sólo por sus desarrolladores, implantada en el cifrador. Otras investigaciones han demostrado que incluso una leve modificación en una Caja-S podría haber debilitado significativamente al DES.

Para el caso del DES, dada una entrada de 6 bits, se obtiene una salida de 4 bits, la cual se encuentra a través de una correspondencia de los dos bits externos y le los 4 internos. Por ejemplo, una entrada "**1**11100**0**" tiene como bits externos "**10**" y "1100" como bits internos.

Cada uno de los 6 bits están diferenciados por la correspondencia donde los bits externos darán la fila y los internos la columna.

Posibles combinaciones de bits externos:

00= 0 **01=** 1

10= 2 **11=** 3

Posibles combinaciones de bits internos:

0000= 00	**0001=** 01	**0010=** 02	**0011=** 03
0100= 04	**0101=** 05	**0110=** 06	**0111=** 07

1000= 08	**1001=** 09	**1010=** 10	**1011=** 11
1100= 12	**1101=** 13	**1110=** 14	**1111=** 15

Esta equivalencia en binario es la misma utilizada para la salida de cada Caja-S.

Por ejemplo de la cadena de bits "**0**11011**1**", da como resultado la fila 01 con la columna 1101, es decir, la fila 1, columna 13, en su respectiva Caja-S.

Se habla de "su respectiva Caja-S", porque así como son 8 grupos de 6 bits, hay 8 Cajas-S diferentes para cada uno de estos grupos respectivamente.

Continuando al mismo tiempo con el ejemplo, la cadena de bits obtenida de la suma OR exclusiva, se subdivide en 8 bloques de 6 bits:

Cuadro 53. K1 XOR E (R0)

S1	S2	S3	S4	S5	S6	S7	S8
111000**0**	**0**01010**0**	**0**00110**0**	**0**11011	**1**01110	**0**01011	**1**00100	**0**01110

De acuerdo a lo anterior, la correspondencia en cada Caja-S es la siguiente:

Cuadro 54. Ejemplo correspondencia Cajas-S

CAJA	FILA	COLUMNA
S1	2	12
S2	0	05
S3	0	03
S4	1	13
S5	2	07
S6	1	05
S7	2	02
S8	0	07

Las 8 Cajas-S del DES son las siguientes:

Tabla 24. Caja S1

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	**3**	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Tabla 25. Caja S2

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	15	1	8	14	6	**11**	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

Tabla 26. Caja S3

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	10	0	9	**14**	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

Tabla 27. CajaS4

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	**10**	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Tabla 28. Caja S5

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	**8**	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

Tabla 29. Caja S6

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	**12**	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

Tabla 30. CajaS7

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	**11**	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

Tabla 31. Caja S8

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	13	2	8	4	6	15	11	**1**	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Nótese que en cada Caja-S ya se encuentra sombreado el resultado de cada una respecto al ejemplo en curso, por tal motivo, la salida para la cadena de bits K1 $\oplus$ E(R0): 111000 001010 000110 011011 101110 001011 100100 001110, es la siguiente:

Cuadro 55. Ejemplo Salida Cajas-S

CAJA	SALIDA	BINARIO
S1	03	0011
S2	11	1011
S3	14	1110
S4	10	1010
S5	08	1000
S6	12	1100
S7	11	1011
S8	01	0001

Nueva cadena **K** $\oplus$ **E(R0):** 0011 1011 1110 1010 1000 1100 1011 0001 (32 bits). Aun falta el último paso en la función (*f* (R_{i-1}, K_i)), el cual es la permutación P.

7.3.2.9. Permutación P

Este es el último paso en la función (*f* (R_{i-1}, K_i)), es una permutación cuya salida se sumará con el sub-bloque L0 dando origen al bloque R1.

Tabla 32. Permutación P

Tabla base:

01	02	03	04
05	06	07	08
09	10	11	12
13	14	15	16
17	18	19	20
21	22	23	24
25	26	27	28
29	30	31	32

Permutación P:

16	07	20	21
29	12	28	17
01	15	23	26
05	18	31	10
02	08	24	14
32	27	03	09
19	13	30	06
22	11	04	25

Tal como en el caso de la permutación PC2, no se sigue ningún orden para generar la permutación P, este es el resultado de organizar los bits de manera aleatoria (Orden especifico definido por los creadores del DES, uno de los puntos base que hace tan fuerte este algoritmo).

Continuando con el ejemplo en desarrollo, donde en este punto se tiene la salida de las Cajas-S:

K $\oplus$ E(R0): 0011 1011 1110 1010 1000 1100 1011 0001

Cuadro 56. Ejemplo permutación P

Tabla base:

0	0	1	1
1	0	1	1
1	1	1	0
1	0	1	0
1	0	0	0
1	1	0	0
1	0	1	1
0	0	0	1

Permutación P:

0	1	0	1
0	0	1	1
0	1	0	0
1	0	0	1
0	1	0	0
1	1	1	1
0	1	0	0
1	1	1	1

***f* (R_0 , K_1) =** 0101 0011 0100 1001 0100 1111 0100 1111

7.3.2.10. Suma Li$\oplus$ Ri

El registro de bits obtenido por la función *f* (R_0 , K_1), es sumado con un OR Exclusivo con el registro de bits de L0, dando como resultado la entrada para el siguiente sub-bloque de Ri (R1).

Continuando con el ejemplo en ejecución se realiza la suma de OR Exclusivo entre: *f* (R_{i-1} , K_i) $\oplus$ L0.

***F* (R_0 , K_1) =** 0101 0011 0100 1001 0100 1111 0100 1111

L0= 1111 1111 0001 1000 1101 0111 1110 1010

R1= 1010 1100 0101 0001 1001 1000 1010 0101

En este punto ya se tiene el sub-bloque R1 para la siguiente vuelta o ronda del DES, pero aun falta obtener a L1, el cual como en el paso anterior hace una suma de OR Exclusivo con la salida de la siguiente ronda.

El registro de bits realizado en la permutación inicial P1, que origino al sub-bloque R0, es la entrada para el siguiente sub-bloque de Li (L1), el proceso se repite 15 veces, siendo el último proceso la permutación $P1^{-1}$.

R0= 0000 0000 1111 1110 1100 1100 1000 0100.

Este bloque se convierte en el nuevo L1, dando como resultado final de esta primera vuelta o ronda, a los siguientes:

L1: 0000 0000 1111 1110 1100 1100 1000 0100

R1: 1010 1100 0101 0001 1001 1000 1010 0101

En estas condiciones, el cifrado DES está definido por las ecuaciones:

$$\mathbf{Li =} \text{ Ri-1}$$

$$\mathbf{Ri =} \text{ Li-1} \oplus f(\text{ Ri-1 , Ki })$$

Hasta este punto se ha descrito una ronda completa del DES, debe retornarse al numeral **7.3.2.2** y así de esta manera, hasta completar 16 vueltas o rondas y tener al final los sub-bloques L16 y R16.

7.3.2.11. Permutación inversa $P1^{-1}$

La tabla de la permutación inversa ($P1^{-1}$) se define por la siguiente correspondencia, siendo la salida final de esta, bloque de mensaje o texto ya cifrado:

Tabla 33. Tabla base inversa P1

01	**02**	**03**	**04**	**05**	**06**	**07**	**08**
09	**10**	**11**	**12**	**13**	**14**	**15**	**16**
17	**18**	**19**	**20**	**21**	**22**	**23**	**24**
25	**26**	**27**	**28**	**29**	**30**	**31**	**32**
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

Para obtener la tabla de permutación $P1^{-1}$, las filas de la tabla base, se convertirán en las columnas de la siguiente leyéndolas de izquierda a derecha, pero ubicándolas en la segunda tabla de abajo hacia arriba, además, lo que hace interesante a esta tabla, es que las filas en blanco y las sombreadas se turnan para conformar las columnas de la siguiente tabla, es decir, la primera fila en blanco, es la primera columna de $P1^{-1}$, seguido, la primera fila sombreada es la segunda columna, la segunda blanca, la tercer columna, la segunda sombreada, la cuarta y así sucesivamente hasta completar a $P1^{-1}$.

Tabla 34. Tabla Permutación $P1^{-1}$

40	**08**	48	**16**	56	**24**	64	**32**
39	**07**	47	**15**	55	**23**	63	**31**
38	**06**	46	**14**	54	**22**	62	**30**
37	**05**	45	**13**	56	**21**	61	**29**
36	**04**	44	**12**	52	**20**	60	**28**
35	**03**	43	**11**	51	**19**	59	**27**
34	**02**	42	**10**	50	**18**	58	**26**
33	**01**	41	**09**	49	**17**	57	**25**

Continuando con el ejemplo utilizado en todo el proceso, si se toman los valores que se tienen de L1 y R1 y se cruzan, suponiendo que son los valores de L16 y R16, se puede continuar con el ejemplo del procedimiento.

R1 = L16: 1010 1100 0101 0001 1001 1000 1010 0101

L1 = R16: 0000 0000 1111 1110 1100 1100 1000 0100

Se concatenan L16 y R16 y se divide en grupos de 8 bits para obtener:

10101100 01010001 10011000 10100101
00000000 11111110 11001100 10000100

Cuadro 57. Ejemplo permutación inversa P1

Tabla base:

1	0	1	0	1	1	0	0
0	1	0	1	0	0	0	1
1	0	0	1	1	0	0	0
1	0	1	0	0	1	0	1
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	0
1	1	0	0	1	1	0	0
1	0	0	0	0	1	0	0

Permutación $P1^{-1}$:

0	0	0	1	0	0	0	1
0	0	1	0	0	0	0	0
0	1	1	0	1	0	1	1
0	1	1	0	1	1	0	0
0	0	1	1	0	1	0	0
0	1	1	0	0	0	0	1
0	0	1	1	1	0	0	0
0	1	1	0	1	1	1	1

La salida de esta permutación $P1^{-1}$ es la siguiente:

00010001 00100000 01101011 01101100
00110100 01100001 00111000 01101111

Al convertir cada octeto a su equivalente decimal y este a su vez convertirlo a carácter se obtiene lo siguiente:

Tabla 35. Equivalencia binario, decimal, caracter

Binario	Decimal	Carácter
00010001	17	◄
00100000	32	(Espacio)
01111011	123	{
01101100	108	L
00110100	52	4
01100001	97	A
00111000	56	8
01101111	111	O

Al final de algoritmo se tiene:

Bloque inicial: “denytamo”

Bloque cifrado: ◄(spacio){l4a8o

Como muestra del resultado final, cabe mencionar que si el mensaje es mayor de 64 bits, se utilizan los bloques necesarios para dividir el mensaje original, si un bloque formado por un mensaje, no tiene la longitud de 64 bits, se rellena utilizando ceros al final.

7.3.3. Descifrado mediante DES

El algoritmo descrito en el punto anterior, igualmente se utiliza para obtener el mensaje original a partir del mensaje cifrado, esto es, empezando con la entrada del bloque de mensaje cifrado de 64 bits, aplicando el procedimiento descrito anteriormente, realizando las 16 vueltas hasta obtener el mensaje en claro.

La diferencia respecto al método de cifrado y que permite el descifrado, esta dada por las siguientes condiciones:

Como se va a descifrar, se deben de tomar los sub-bloques en sentido contrario, de esta manera se puede pensar que se está manipulando desde los sub-bloques L16 y R16 hasta llegar a L0 y R0.

También las sub-claves Ki se tomaran en sentido inverso, en donde K16 será K1, K15 = K2, y así sucesivamente hasta tener K1 = K16.

Además, los desplazamientos para el cálculo de las subclaves de descifrado son los mismos de la tabla LS_i pero ahora se toman hacia la derecha, puesto que los desplazamientos coinciden y al final se obtendrá de igual manera: D0 = D16 y C0 = C16.

De esta manera se comprueba que el DES es un algoritmo de cifrado simétrico.

7.3.4. Claves débiles en el DES

El algoritmo DES presenta algunas claves débiles y semidebiles de acuerdo a su

estructura y funcionamiento. En general, todos aquellos valores de la clave que conducen a una secuencia inadecuada de Ki son poco recomendables.

DES tiene unas pocas claves débiles y semi-débiles. Son las claves que causan que el modo de cifrado de DES funcione de manera idéntica al modo de descifrado de DES.

Durante una operación, la clave secreta de 56 bits (Después de eliminar el bit menos significativo de cada carácter) se descompone en 16 sub-claves de acuerdo con el procedimiento de clave de DES; cada una corresponde a una de las dieciséis rondas de DES. Las claves débiles de DES son aquellas que producen 16 subclaves idénticas. Esto ocurre cuando los bits de la clave son:

- Todo ceros.
- Todo unos.
- La primera mitad ceros y la segunda unos.
- La primera mitad unos y la segunda ceros.

Dado que todas las subclaves son idénticas y que DES es una red de Feistel (el cifrado es autoreversible), significa que cifrar dos veces con una de estas claves, produce el texto plano original.

DES tiene también claves semi-débiles. Vienen en pares K_1 and K_2 y poseen la siguiente propiedad: E_{K1} (E_{K2}(M)) = M, donde E_{Ki}(M) es el algoritmo de cifrado cifrando el mensaje M con la clave K.

Existen seis pares de claves semidébiles. Las claves débiles y semi-débiles no son fallos totales de DES. Existen 2^{56} (7.21×10^{16}, alrededor de 72 cuadrillones) de posibles claves para DES, de las cuales cuatro son débiles y doce semidébiles. Esta es una fracción tan pequeña del espacio total de claves que los usuarios no deben preocuparse, ya que dichas claves pueden ser detectadas al ser generadas.

7.3.5. Triple DES

Desarrollado por IBM en el año 1978. (3DES o TDES), es un tipo de algoritmo que

realiza un triple cifrado DES, esto lo hace muchísimo más seguro que el cifrado DES simple.

El Triple DES, no es un cifrado múltiple, ya que todas las subclases no son independientes, esto es porque DES tiene la propiedad matemática de no ser un grupo; esto implica que si se cifra el mismo bloque dos veces, con dos llaves distintas, se aumenta el tamaño efectivo de la llave.

La variante más simple del Triple DES funciona de la siguiente manera:

$$C= E^{K3}_{DES} (D^{K2}_{DES}(E^{K1}_{DES}(M)))$$

Donde *M* es el mensaje a cifrar y k1, k2 y k3 las respectivas claves DES. En la variante 3TDES las tres claves son diferentes; en la variante 2TDES, la primera y tercera clave son iguales.

TDES nació por la inseguridad que traía una clave de 56 bits. Las claves de 56 bits son posibles de descifrar utilizando un ataque de fuerza bruta. El TDES agranda el largo de la llave, sin necesidad de cambiar de algoritmo cifrador.

El método de cifrado TDES desaparece progresivamente, siendo reemplazado por el algoritmo AES que es considerado mucho más rápido (hasta 6 veces más rápido). De todas maneras, algunas tarjetas de créditos y otros métodos de pago electrónico, todavía tienen como estándar el algoritmo Triple DES.

7.4. ALGORITMO AES, 20 AÑOS DE SEGURIDAD

7.4.1. Introducción e historia

El AES (Advanced Encryption Standard) es probablemente un algoritmo de cifrado de los más utilizados hoy en día pues es seguro y eficiente, de hecho es el estándar de cifrado elegido por el Instituto Nacional de estándares y tecnología como estándar de proceso de información federal.

Se puede empezar por ver el resumen básico de la evolución del estándar de cifrado en los últimos años. En 1999 DES, cuya longitud de clave era un tanto corta y podía ser un tanto comprometida, fue sustituido por TDES o TDEA, que triplicaba la clave.

Finalmente en 2001 apareció AES basado en el algoritmo RINJDAEL, que fue el algoritmo ganador del concurso convocado por, el Instituto Nacional de Estándares y Tecnología de EEUU (NIST) pues cumplía las características requeridas perfectamente, garantizando la seguridad hasta los siguientes 20 años.

En el año 1997 NIST, emprende un proceso abierto para la selección de un nuevo algoritmo de cifrado (AES), que sustituya al antiguo estándar de cifrado (DES).

Este algoritmo es útil no sólo para proteger la información del Gobierno EE. UU, sino que también se espera que sea utilizado masivamente por el sector privado, y que sea adoptado por el resto de países, incluso Europa, ya que es un algoritmo público. Las características principales del nuevo algoritmo recaen en la: rapidez, seguridad, eficiencia y facilidad de implementación.

En 1997, el Instituto Nacional de Estándares y Tecnología de EEUU (NIST) inicia los primeros pasos para la consolidación de un Estándar de Cifrado Avanzado (AES), que pueda permitir proteger los datos confidenciales del gobierno, así como la información sensible de los ciudadanos.

El objetivo principal era desarrollar una especificación que permitiese encontrar un algoritmo de cifrado que sustituya al obsoleto DES (inseguro actualmente debido a su pequeña clave de 56 bits que permite ataques por fuerza bruta en tiempos razonables, con máquinas específicas).

Se aceptaron quince candidatos durante la Primera Conferencia de Candidato que fueron:

1.	CAST-256	Entust Technologies, Inc. (C. Adams).
2.	CRYPTION	Future Systems, Inc. (Chae Hoon Lim).
3.	DEAL	L. Knudsen, R. Outerbridge.
4.	DFC	CNRS-Ecole Normale Superiere (S. Vaudenay).
5.	E2 NTT	Nippon Telegraph and Telephone Corporation (M. Kanda).
6.	FROG	TecApro International S.A. (D. Georgoudis, Leroux, Chaves).
7.	HPC	R. Schoeppel.
8.	LOKI97	L. Brown, J. Pieprzyk, J.Seberry.
9.	MAGENTA	Deutshe Telekom AG (K. Huber).
10.	MARS	IBM (Nevenko Zunic).
11.	RC6	RSA Laboratories (Rivest, M. Robshaw, Sidney, Yin).
12.	***Rijndael***	***Joan Daemen, Vicent Rijmen.***
13.	SAFER+	Cylink Corporation (L. Chen).
14.	SERPENT	R. Anderson, E. Biham, L. Knudsen.

15. TWOFISH B.Schneier, J.Kelsey, D.Whiting, D.Wagner, C. Hall, N. Ferguson.

La segunda conferencia de Candidatos a AES, se celebró para discutir los resultados de las numerosas pruebas y criptoanálisis realizados por la comunidad criptográfica mundial sobre los quince candidatos iníciales.

Basándose en estos comentarios y análisis, NIST seleccionó cinco candidatos finalistas.

Los cinco algoritmos seleccionados para la siguiente ronda serían:

1. MARS.
2. RC6.
3. RIJNDAEL.
4. SERPENT.
5. TWOFISH.

Los asistentes presentaron nuevos documentos de evaluación y criptoanálisis de los últimos cinco candidatos.

Por fin, el 2 de octubre de 2000, el NIST anunció el algoritmo ganador: Rijndael propuesto por los belgas Vicent Rijmen y Joan Daemen.

Los motivos para seleccionar a “RIJNDAEL”, fue su buena combinación de seguridad-velocidad-eficiencia, sencillez y flexibilidad.

7.4.2. Descripción básica del AES

7.4.2.1. Campos finitos GF(2^8)

Esta sección es muy importante para poder entender muchos términos que se mencionaran más adelante en el funcionamiento del AES, ya que AES trabaja como elementos básicos a los bytes (conjunto de 8 bits), AES ve a los bytes como elementos del campo finito GF(2^8).

En álgebra abstracta, un cuerpo finito, campo finito o campo de Galois (llamado así por Évariste Galois) es un cuerpo que contiene un número finito de elementos.

Los cuerpos finitos son importantes en teoría de números, geometría algebraica, teoría de Galois, y Criptografía.

Dado que todo cuerpo de característica **0** contiene a los racionales y es por lo tanto infinito, todos los campos finitos tienen característica prima, y por lo tanto, su tamaño (o cardinalidad) es de la forma $\mathbf{p^n}$, para **p** primo y **n > 0** entero. No es en general cierto, sin embargo, que todo cuerpo de característica prima sea finito.

Si $\mathbf{q = p^n}$ es una potencia de un primo, existe exactamente un campo con q elementos, denotado por **Fq** o **GF(q)**. Se puede construir como sigue: encuéntrese un polinomio irreducible **f(x)** de grado **n** con coeficientes en **Fp**, y **defínase Fq = Fp[X] / < f (T) >**, donde **Fp[X]** denota el anillo de todos los polinomios con coeficientes en **Fp**, **< f(x) >** denota el ideal generado por **f(x)**, y la barra diagonal indica el anillo cociente (definido de forma similar al grupo cociente). El polinomio **f(x)** se puede hallar factorizando $\mathbf{X^q - X}$ sobre **Fp**. El campo **Fq** contiene a (una copia de) **Fp** como subcampo.

En este algoritmo todos los bytes se interpretan como elementos de un cuerpo finito. Concretamente, se representan mediante campos de Galois que se representan como **GF(k)**. ("Galois Field").

Los campos de Galois son muy interesantes en Criptografía, gracias a que existe un inverso aditivo y multiplicativo que permite cifrar y descifrar en el mismo cuerpo **Zk**, eliminando así los problemas de redondeo o truncamiento de valores si tales operaciones de cifrado y descifrado se hubiesen realizado en aritmética real.

En el caso del AES, interesa utilizar una aritmética en módulo **p** sobre polinomios de grado **m**, siendo **p** un número primo. Este campo de Galois queda representado como: $GF(p^m)$, en donde los elementos de $GF(p^m)$ se representan como polinomios con coeficientes en **Zp** de grado menor que **m**, es decir:

$$GF(p^m) = \{ \lambda_0 + \lambda_1 x + \lambda_2 x^2 + ... + \lambda_{m-1} x^{m-1}; \lambda_0, \lambda_1, \lambda_2, ..., \lambda_{m-1} \in Zp\}$$

Cada elemento de $\mathbf{GF(p^m)}$ es un resto módulo **p(x)**, donde **p(x)** es un polinomio irreducible de grado **m**, esto es, que no puede ser factorizado en polinomios de grado menor que **m**.

En el caso del algoritmo Rijndael, son interesantes los campos del tipo $\mathbf{GF(2^m)}$ puesto que los coeficientes en este caso serán los restos del módulo **2**, es decir, **0** y **1**, lo que permite una representación binaria. Por lo tanto, cada elemento del campo se representa con **m** bits y el número de elementos será $\mathbf{2^m}$.

Por ejemplo, para el campo GF(2^3) sus elementos son:

- 0,1
- $x^2 + x + 1$

Que son precisamente todos los restos de un polinomio de grado **m - 1** = **2**.

En el caso del algoritmo Rijndael, se definen operaciones a nivel de byte, en el campo GF(2^8).

Un byte **b** consiste de los bits b_7 b_6 b_5 b_4 b3 b_2 b_1 b_0; si se considera como un polinomio con coeficientes en {0, 1}, se tiene el polinomio:

$$b_7 x^7 + b_6 x^6 + b_5 x^5 + b_4 x^4 + b_4 x^3 + b_2 x^2 + b_1 x^1 + b_0 x^0.$$

Por tal motivo AES utiliza parejas de caracteres en hexadecimal, ya que cada uno necesita únicamente de 4 bits para su representación. Por ejemplo, un byte con el valor hexadecimal "**23**", en binario **0010 0011**, corresponde con el polinomio:

- ✓ $x^5 + x^1 + 1$.

Nótese como ayuda, la siguiente correspondencia de hexadecimal a binario:

Tabla 36. Tabla Hexadecimal a Binario

HEXADECIMAL	BINARIO
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

7.4.2.2. El anillo GF(2^8) [x] / (x^4+1)

Otra estructura que usa AES es la del anillo GF(2^8) [x] / (x^4+1), otra de las operaciones básicas de AES es multiplicar un polinomio de tercer grado con coeficientes en GF(2^8) , por una constante. Es decir, un elemento de GF(2^8) [x] / (x^4+1), por un polinomio constante, el resultado se reduce módulo (x^4+1) para obtener un polinomio de grado 3. El objetivo de lo anterior es poder definir multiplicación entre columnas de 4 bytes por un elemento constante también de 4 bytes. En el proceso de descifrado se aplica la operación inversa y aunque el polinomio (x^4+1) no es irreducible, en este caso particular la constante si tiene inverso en el anillo, por lo tanto no hay problema.

Sea a(x) un polinomio tal que $a_0 + a_1x + a_2x^2 + a_3x^3 \epsilon$ GF(2^8) [x] / (x^4+1), donde $a_i \epsilon$ GF(2^8), y b(x) otro polinomio igual, entonces a(x)b(x) = c(x) donde c(x) tiene la misma forma, particularmente:

$$a(x)b(x) = (a_0 + a_1x + a_2x^2 + a_3x^3)(b_0 + b_1x + b_2x^2 + b_3x^3)$$

$$= a_0 b_0 + (a_1 b_0 + a_0 b_1)x + (a_2 b_0 + a_1 b_1 + a_0 b_2)x^2$$

$$+ (a_3 b_0 + a_2 b_1 + a_1 b_2 + a_0 b_3)x^3$$

$$+ (a_3 b_1 + a_2 b_2 + a_1 b_3)x^4$$

$$+ (a_3 b_2 + a_2 b_3)x^5 + a_3 b_3 x^6$$

El siguiente paso es aplicar módulo (x^4 + 1) a todo el polinomio, que es aplicar módulo a cada uno de sus términos, entonces $x^i \bmod (x^4+1) = x^{i \bmod 4}$, es decir el resultado de a(x)b(x) queda como:

$$a(x)b(x) = (a_0 b_0 + a_3 b_1 + a_2 b_2 + a_1 b_3)$$

$$+(a_1 b_0 + a_0 b_1 + a_3 b_2 + a_2 b_3)x$$

$$+(a_2 b_0 + a_1 b_1 + a_0 b_2 + a_3 b_3)x^2$$

$$+(a_3 b_0 + a_2 b_1 + a_1 b_2 + a_0 b_3)x^3$$

$$= c_0 + c_1 x^1 + c_2 x^2 + c_3 x^3$$

Finalmente de manera matricial el anterior producto puede ser visto como:

$$\begin{bmatrix} c0 \\ c1 \\ c2 \\ c3 \end{bmatrix} = \begin{bmatrix} a0 & a3 & a2 & a1 \\ a1 & a0 & a3 & a2 \\ a2 & a1 & a0 & a3 \\ a3 & a2 & a1 & a0 \end{bmatrix} \cdot \begin{bmatrix} b0 \\ b1 \\ b2 \\ b3 \end{bmatrix}$$

7.4.2.3. Características básicas

Entre los principales requisitos del AES se destacan los siguientes tres:

- ✓ Ser un algoritmo de cifrado en bloque simétrico, es decir, el algoritmo utiliza un método criptográfico que usa una misma clave para cifrar y para descifrar mensajes. Las dos partes que se comunican han de ponerse de acuerdo sobre la clave a usar, de esta manera una vez que ambas partes tienen acceso a esta clave, el remitente cifra un mensaje usándola, lo envía al destinatario, y éste lo descifra con la misma clave.
- ✓ Estar diseñado de manera que se pueda aumentar la longitud de clave según las necesidades, de manera que las claves podrán variar siempre y cuando el tamaño sea múltiplo de 4 bytes. Las longitudes de clave utilizadas por defecto serán 128 bits, 192 bits y 256 bits.
- ✓ A su vez está diseñado para que el tamaño de los bloques de datos también sea variable dentro del mismo rango y con las mismas restricciones, es decir, bloques de datos cuyo tamaño sea múltiplo de 4 bytes.

La estructura del algoritmo Rijndael (AES) está formada por un conjunto de rondas, entendiendo por rondas un conjunto de iteraciones de cuatro funciones matemáticas diferentes e invertibles.

Por tal motivo, el algoritmo se basa en aplicar un número de rondas determinado a un texto en claro para producir una información cifrada. La información generada por cada función es un resultado intermedio, que se conoce como ***Estado***, o si se prefiere ***Estado Intermedio***.

El algoritmo representa el ***Estado*** como un matriz rectangular de bytes, que posee cuatro filas y **Nb** columnas. Siendo el número de columnas **Nb** en función del tamaño del bloque:

Nb = tamaño del bloque utilizado en bits /32.

De esta manera, si se tiene un bloque con 128 bits se tendría una matriz de cuatro filas y (**Nb** = 128 / 32) = 4 columnas quedando una matriz de forma:

Tabla 37. Matriz de Estado

A00	A01	A02	A03
A10	A11	A12	A13
A20	A21	A22	A23
A30	A31	A32	A33

La clave del sistema se representa con una estructura análoga a la del ***Estado***, es decir, se representa mediante una matriz rectangular de bytes de cuatro filas y **Nk** columnas.

Siendo el número de columnas **Nk** en función del tamaño de la clave:

Nk = tamaño de la clave en bits /32.

De esta manera, al igual que ocurría con el tamaño del bloque, si se tiene una clave con 128 bits se dispondría de una matriz de cuatro filas y (**Nk** = 128 / 32) = 4 columnas, quedando una matriz de forma:

Tabla 38. Matriz de la clave

K00	K01	K02	K03
K10	K11	K12	K13
K20	K21	K22	K23
K30	K31	K32	K33

El número de iteraciones o vueltas de las cuatro transformaciones sobre la información, o mejor dicho sobre la matriz de Estado Intermedio depende de la versión del algoritmo que se utilice. Esta explicación se da en base al algoritmo en que los autores definen que para tamaños de bloques y claves entre 128 y 256 bits (con incrementos de 32 bits) el número de vueltas **Nr** es determinado por la siguiente expresión:

Nr = máx (Nk, Nb) + 6.

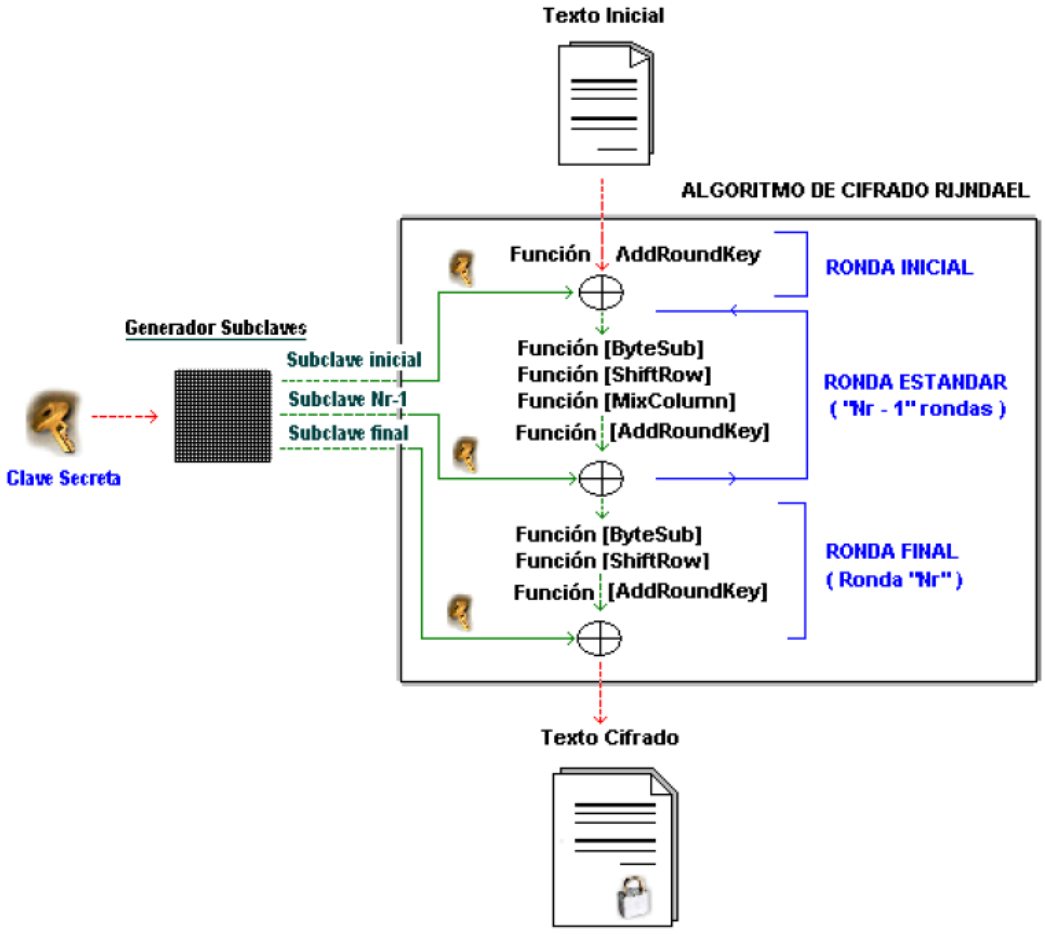
De manera informativa se muestra cómo quedarían las rondas para cada uno de los posibles resultados, siendo la primera columna el tamaño de la clave y la primera fila el tamaño del bloque:

Tabla 39. Correspondencia numero de rondas según Clave/Bloque

Clave/Bloque	Nb= 4(128 bits)	Nb= 6(192 bits)	Nb= 8(256bits)
Nk= 4(128 bits)	10	12	14
Nk= 6(193 bits)	12	12	14
Nk= 8(256 bits)	14	14	14

Teniendo ya una visión general del algoritmo, se puede profundizar más en la manera como son cifrados los datos; un esquema general de cifrado del AES es el siguiente:

Figura 13. Cifrado del algoritmo AES



Fuente: http://www.coriol.com/blog/images/2008/05/rijndael1.jpg [online]

Como se había mencionado y se puede ver en el gráfico, el proceso de cifrado consiste en la aplicación de cuatro funciones matemáticas invertibles sobre la información que se desea cifrar. Las transformaciones se repiten en su orden en cada ronda o vuelta definida con anterioridad.

Las cuatro funciones matemáticas son:

- **a.** ByteSub o sustitución de bytes.
- **b.** ShiftRow o etapa de desplazamiento de filas.
- **c.** MixColumn, o etapas de mezcla de columnas.
- **d.** AddRoundKey o etapa de adición de clave.
- ✓ KeySchedule o expansión de clave (Función matemática de la clave).

En esencia, la información a cifrar se va transformando en la matriz de ***Estado*** la cual se introduce al cifrador, y sufre una primera transformación, en la ronda inicial, que consiste en una operación or-exclusiva (AddRoundKey) entre una Subclave generada y la matriz de Estado. A continuación, a la matriz de ***Estado*** resultante se le aplican cuatro transformaciones invertibles, repitiéndose este proceso **Nr - 1** veces, en lo que se conoce como ***Ronda Estándar***. Finalmente, se le aplica una última ronda o vuelta a la matriz de Estado resultante de las **Nr - 1** rondas anteriores, aplicando las funciones ***ByteSub, ShiftRow*** y ***AddRoundKey*** en este orden. El resultado de la ronda final produce el bloque cifrado deseado.

7.4.3. Descripción del proceso de cifrado

Para iniciar la explicación paso a paso del AES primero que todo, recordar que puede trabajar con bloques de datos de 128, 192, 256 bits y longitudes de claves de 128, 192, 256 bits. Además AES tiene 10, 12 ó 14 vueltas respectivamente. Cada vuelta consiste en la aplicación de una ronda estándar, que consiste en cuatro transformaciones básicas. La última ronda es especial y consiste de tres operaciones básicas, añadiendo siempre una ronda inicial. Por otro lado, se tiene el programa de claves o extensión de la clave.

Cabe recordar que AES interpreta el bloque de entrada de 128 bits, como una matriz 4 x 4 de entradas de bytes. Si el bloque es de 192 bits se agregan dos columnas más, y si es de 256 se agregan cuatro columnas más.

De esta manera el bloque de datos, suponiendo que se dispone de un bloque de 128 bits, organizados en una matriz, es de la siguiente manera:

Tabla 40. Matriz para 128 bits

A00	A01	A02	A03
A10	A11	A12	A13
A20	A21	A22	A23
A30	A31	A32	A33

Para una mejor comprensión se va a pasar a exponer un ejemplo explicativo en el que se vea cómo actúa cada una de estas funciones sobre una clave y un bloque de datos, teniendo en cuenta que la ejecución de las funciones es la siguiente:

- AddRoundKey
- ByteSub
- ShiftRows
- MixColumns
- AddRoundKey

Ejemplo. Se aplica el algoritmo criptográfico AES con los siguientes datos:

Bloque de datos: 32 43 F6 A8 88 5A 30 8D 31 31 98 A2 E0 37 07 34

Cuadro 58. Matriz de Estado

32	88	31	E0
43	5ª	31	37
F6	30	98	07
A8	8D	A2	34

Clave inicial: 2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 C7 4F 3C.

Cuadro 59. Matriz Clave

2B	28	AB	09
7E	AE	F7	CF
15	D2	15	4F
16	A6	88	3C

7.4.3.1. Función AddRoundKey

En esta función, etapa o tratamiento se procede a realizar un XOR byte a byte entre la matriz de ***Estado*** o matriz intermedia y la matriz de la ***clave*** o subclave, dependiendo de la ronda en la que se encuentre.

Tabla 41. Función AddRoundKey

Matriz Estado:

A00	A01	A02	A03
A10	A11	A12	A13
A20	A21	A22	A23
A30	A31	A32	A33

$\oplus$

Matriz de clave:

K00	K01	K02	K03
K10	K11	K12	K13
K20	K21	K22	K23
K30	K31	K32	K33

A21 $\oplus$ K21

De esta manera toma la matriz $[A_{ij}]$ y $[K_{ij}]$ y aplicando el XOR da como resultado la matriz $[A_{ij} \oplus K_{ij}]$.

Una vez acaba la función MixColumns, se vuelve a a proceder con esta función de AddRoundKey, creándose de nuevo un estado intermedio 1 pero en la siguiente ronda.

Aplicando el ejemplo en curso se tiene lo siguiente:

Cuadro 60. Ejemplo AddRoundKey

Matriz Estado:

32	88	31	E0
43	5A	31	37
F6	30	98	07
A8	8D	A2	34

$\oplus$

Matriz de clave:

2B	28	AB	09
7E	AE	F7	CF
15	D2	15	4F
16	A6	88	3C

Se tiene la siguiente tabla de equivalencia de hexadecimal a binario:

Tabla 42. Hexadecimal a Binario

HEXADECIMAL	BINARIO
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

32 xor 2B = 0011 0010
$\oplus$
0010 1011

0001 1001 **= 19**

Para cada elemento del bloque de entrada se realiza una operación XOR con su equivalente en posición del bloque de la clave.

Cuadro 61. Nueva Matriz Estado intermedio 1

19	A0	9A	E9
3D	F4	C6	F8
E3	E2	8D	48
BE	2B	2A	08

7.4.3.2. Cálculo de las subclaves

Basándose en el principio de la criptografía moderna, mediante el cual se establece que la seguridad de un algoritmo sólo debe depender de la clave utilizada, se utilizan diferentes subclaves Ki tanto en el cifrado como en el descifrado para que el resultado del algoritmo dependa de una información externa al sistema: la clave del usuario.

Estas subclaves (RoundKeys) se derivan de la clave principal K mediante el uso de dos funciones: una de expansión y otra de selección.

Siendo ***n*** el número de rondas que aplique el algoritmo, el número total de bits para subclaves que se necesitan para todas las rondas es igual al tamaño del bloque utilizado multiplicado por (n + 1). También se puede ver como **(4 (n + 1) * Nb)** bytes. Es decir, por ejemplo, para un tamaño de bloque de 128 bits y 10 vueltas, se necesitan 1408 bits de subclaves: (128 * 11 = 1408 bits) que en forma matricial son 44 columnas de 4 filas, (este punto se explica en detalle en el numeral **6.4.3.6**).

Por tanto, el número de claves que se generan depende del número de rondas empleadas (**Nr** = máx (Nk, Nb) + 6.).

7.4.3.3. Función ByteSub

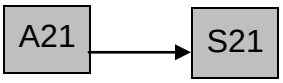
En esta función o etapa se procede a realizar una sustitución no lineal que se aplica a cada byte de la matriz de ***Estado*** de forma independiente, generando un nuevo byte, es decir, a cada elemento de la matriz de ***Estado*** se le sustituye por otro byte que depende del primero mencionado. Esta sustitución se lleva a cabo utilizando unas tablas o matrices (Cajas-S) invertibles.

Tabla 43. Función ByteSub

A00	A01	A02	A03
A11	A12	A13	A10
A20	A21	A22	A23
A33	A30	A31	A32

→

S00	S01	S02	S03
S11	S12	S13	S10
S20	S21	S22	S23
S33	S30	S31	S32



Esta sustitución tiene 2 apartados o transformaciones:

1. Cada byte es considerado como un elemento del GF(2^8) que genera el polinomio irreducible m(x) = x8 + x4 + x3 + x + 1 y sustituido por su inversa multiplicativa. El valor cero en este caso no varía pues no tiene recíproco.

Tabla 44. Inversos multiplicativos en hexadecimal

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	8D	F6	CB	52	7B	D1	E8	4F	29	C0	B0	E1	E5	C7
1	74	B4	AA	4B	99	2B	60	5F	58	3F	FD	CC	FF	40	EE	B2
2	3A	6E	5A	F1	55	4D	A8	C9	C1	0A	98	15	30	44	A2	C2
3	3C	45	92	6C	F3	39	66	42	F2	35	20	6F	77	BB	59	19
4	1D	FE	37	67	2D	31	F5	69	A7	64	AB	13	54	25	E9	09
5	ED	5C	05	CA	4C	24	87	BF	18	3E	22	F0	51	EC	61	17
6	16	5E	AF	D3	49	A6	36	43	F4	47	91	DF	33	93	21	3B
7	79	B7	97	85	10	B5	BA	3C	B6	70	D0	06	A1	FA	81	82
8	83	7E	7F	80	96	73	BE	56	9B	9E	95	D9	F7	02	B9	A4
9	DE	6A	32	6D	D8	8A	84	72	2A	14	9F	88	F9	DC	89	9A
A	FB	7C	2E	C3	8F	B8	65	48	26	C8	12	4A	CE	E7	D2	62
B	0C	E0	1F	EF	11	75	78	71	A5	8E	76	3D	BD	BC	86	57
C	0B	28	2F	A3	DA	D4	E4	0F	A9	27	53	04	1B	FC	AC	E6
D	7A	07	AE	63	C5	DB	E2	EA	94	8B	C4	D5	9D	F8	90	6B
E	B1	0D	D6	EB	C6	0E	CF	AD	08	4E	D7	E3	5D	50	1E	B3
F	5B	23	38	34	68	46	03	8C	DD	9C	7D	A0	CD	1A	41	1C

2. Ahora la transformación lineal que sigue al inverso multiplicativo se aplica bit a bit y sigue la siguiente regla:
   Transformación lineal en GF(2^8) ⟶ GF(2^8)

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

   En términos de bits queda como:

   $\mathbf{b_0'} = b_0$ XOR b_4 XOR b_5 XOR b_6 XOR b_7 XOR c_0
   $\mathbf{b_1'} = b_1$ XOR b_5 XOR b_6 XOR b_7 XOR b_0 XOR c_1
   $\mathbf{b_2'} = b_2$ XOR b_6 XOR b_7 XOR b_0 XOR b_1 XOR c_2
   $\mathbf{b_3'} = b_3$ XOR b_7 XOR b_0 XOR b_1 XOR b_2 XOR c_3

$\mathbf{b_4'} = b_4 \text{ XOR } b_0 \text{ XOR } b_1 \text{ XOR } b_2 \text{ XOR } b_3 \text{ XOR } c_4$
$\mathbf{b_5'} = b_5 \text{ XOR } b_1 \text{ XOR } b_2 \text{ XOR } b_3 \text{ XOR } b_4 \text{ XOR } c_5$
$\mathbf{b_6'} = b_6 \text{ XOR } b_2 \text{ XOR } b_3 \text{ XOR } b_4 \text{ XOR } b_5 \text{ XOR } c_6$
$\mathbf{b_7'} = b_7 \text{ XOR } b_3 \text{ XOR } b_4 \text{ XOR } b_5 \text{ XOR } b_6 \text{ XOR } c_7$

En representación matricial:

$$\begin{bmatrix} b0' \\ b1' \\ b2' \\ b4' \\ b4' \\ b5' \\ b6' \\ b7' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b0 \\ b1 \\ b2 \\ b3 \\ b4 \\ b5 \\ b6 \\ b7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

El resultado final de estas dos transformaciones se puede expresar en una tabla de sustitución denominada S-Box (Caja-S) aplicada a cada byte, sabiendo que este está expresado en forma hexadecimal, de manera que el elemento más a la izquierda deberá situarse en la columna y el elemento más a la derecha en la fila, siendo el resultado final el lugar donde coincidan o se unan.

Tabla 45. Sustitución Caja-S AES

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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	FF	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	A6	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	0F	B0	54	BB	16

Ejemplo rápido demostrativo:

Por ejemplo la aplicación de Bytesub de “7b” sigue los siguientes pasos:

$7B = 0111\ 1011 = X^6 + X^5 + X^4 + X^3 + X^1 + X^0$

$7B^{-1} = 00000110 = 06$, el cual se obtiene de cruzar la fila 7 con la columna B en la matriz de los inversos multiplicativos.

Ahora aplicando la transformación lineal donde se reemplaza en la matriz con:

Cuadro 62. Ejemplo 01 ByteSub

b0	b1	b2	b3	b4	b5	b6	b7
0	0	0	0	0	1	1	0

b7	b6	b5	b4	b3	b2	b1	b0
0	1	1	0	0	0	0	0

Y finalmente se tiene 0010 0001 = 21, que puede obtenerse directamente en la intersección de la fila 7 columna B de la tabla de Caja-S del AES y de esta manera evitarse todo tipo de cálculo matemático a la hora de examinar el algoritmo.

Continuando con el ejemplo en curso para esta etapa:

Cuadro 63. Matriz de estado

19	A0	9ª	E9
3D	F4	C6	F8
E3	E2	8D	48
BE	2B	2ª	08

Para cada elemento se realiza una operación equivalente a buscar en la tabla denominada Caja-S AES, los dos dígitos que forman cada elemento y se obtiene el byte a sustituir:

19 = Fila 1, Columna 9 = **D4**

3D = Fila 3, Columna D = **27**

Cuadro 64. Nueva Matriz de Estado intermedio 2

D4	E0	B8	1E
27	BF	B4	41
11	98	5D	52
AE	F1	E5	30

7.4.3.4. Función ShiftRow

En esta etapa se procede a realizar un desplazamiento a la izquierda cíclicamente de las filas que conforman la matriz de ***Estado*** actual, es decir, rotar los bytes de las filas de la matriz de estado resultante de la transformación anterior de la siguiente manera:

- *Fila 0:* 0 posiciones, no se desplaza.
- *Fila 1:* 1 byte a la izquierda.
- *Fila 2:* 2 bytes a la izquierda.
- *Fila 3:* 3 bytes a la izquierda.
- *Fila n:* n bytes a la izquierda.

Cada fila se desplaza un número de posiciones diferentes, este número de vueltas o rotaciones dependerá del tamaño del bloque **Nb**, explicado en la estructura del algoritmo.

De forma gráfica, un bloque de 128 bits quedaría de la siguiente manera:

Tabla 46. Función ShiftRow

Matriz Estado S:

Fila 0	A00	A01	A02	A03
Fila 1	A10	A11	A12	A13
Fila 2	A20	A21	A22	A23
Fila 3	A30	A31	A32	A33

Matriz Estado S':

Fila 0	A00	A01	A02	A03
Fila 1	A11	A12	A13	A10
Fila 2	A22	A23	A20	A21
Fila 3	A33	A30	A31	A32

Se resalta la primera posición de cada fila de la matriz de **Estado S** para observar como quedan organizadas en la matriz **Estado S'**.

Continuando con el ejemplo en curso para esta etapa:

Estado intermedio 3:

Dependiendo del número de fila en el que se encuentre se realiza 1, 2 o n desplazamientos circulares hacia la izquierda.

Cuadro 65. Ejemplo 01 ShiftRow

Matriz Estado S:

D4	E0	B8	1E
27	BF	B4	41
11	98	5D	52
AE	F1	E5	30

Matriz Estado S':

D4	E0	B8	1E
BF	B4	41	27
5D	63	11	98
30	AE	F1	E5

De igual manera, se resalta la primera posición de cada fila de la matriz de **Estado S** para observar como quedan organizadas en la matriz **Estado S'**.

7.4.3.5. Función MixColumns

En esta etapa se trabaja con los bytes de una misma columna de la matriz de ***Estado*** resultante de la transformación anterior. Las columnas son tratadas como polinomios, cuyos coeficientes pertenecen a $GF(2^8)$.

La transformación consiste en multiplicar las columnas en módulo $(x^4 + 1)$ por el polinomio **c(x)** = 03 x^3 + 01 x^2 + 01 x + 02. De forma matemática se expresa: **S' (x)** = c(x) $\bigoplus$ S(x). Donde **S'(x)** representa la matriz de ***Estado*** resultante de esta etapa, S(x) representa la matriz de ***Estado*** de entrada.

Para comprender más fácil esta fórmula, se puede expresar de manera matricial y grafica de la siguiente manera:

Forma matricial:

Entonces A' = A **.** c(x), como, se vio en el numeral (6.4.2.2), este producto se puede representar por la siguiente matriz.

$$\begin{bmatrix} A'0c \\ A'1c \\ A'2c \\ A'3c \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} A0c \\ A1c \\ A2c \\ A3c \end{bmatrix}$$

De forma grafica, en un bloque de 128 bits:

MixColumns toma cada columna A, y la manda a otra columna A', que se obtiene al multiplicar A por una cadena de bytes constante.

Tabla 47. Función MixColumns

A00	A0c	A02	A03
A10	A1c	A12	A13
A20	A2c	A22	A23
A30	A3c	A32	A33

A00	A0c ’	A02	A03
A11	A1c ’	A13	A10
A22	A2c ’	A20	A21
A33	A3c ’	A31	A32

Continuando con el ejemplo en curso para esta etapa:

Cuadro 66. Matriz de estado

D4	E0	B8	1E
BF	B4	41	27
5D	63	11	98
30	AE	F1	E5

Estado intermedio 4:

Se realiza una operación S’(x) = c(x) XOR S(x) donde c(x) = 03 x^3 + 01 x^2 + 01 x^1 + 02, S’ y S serán la matriz resultado y la matriz de entrada respectivamente.

Para la primera columna: D4 BF 5D 30

S’(0)**=** (02(D4) + 03(BF) 01(5D) + 01(30)) x^0

+ (01(D4) + 02(BF) 03(5D) + 01(30)) x^1

+ (01(D4) + 01(BF) 02(5D) + 03(30)) x^2

+ (03(D4) + 01(BF) 01(5D) + 02(30)) x^3 = D4 + BF x^1 + 5D x^2 + 30 x^3

Cuadro 67. Nueva matriz de Estado (Ya aplicada MixColumns)

D4	E0	B8	1E
BF	B4	41	27
5D	63	11	98
30	AE	F1	E5

Terminada esta etapa concluye la primera ronda del algoritmo, el cual para este caso consta de 10 rondas como ya se definió, a partir de este punto se aplica la siguiente expansión a la clave para obtener el nuevo bloque de ***clave*** que junto con la anterior matriz de ***Estado*** resultante, se convierten en las entradas de la nueva ronda, en la función ***Addroundkey.***

7.4.3.6. Función KeySchedule

Aunque ***Rijndael*** está diseñado para manejar muchos casos de longitudes de claves y de bloques, finalmente el algoritmo AES definido en el estándar determina que solo se permiten los casos de bloques de 128 bits, y longitudes de claves de 128, 192 y 256 bits respectivamente.

Por otra parte la longitud de 128 bits, garantiza la seguridad del sistema hasta después del año 2030, por lo que la siguiente explicación se desarrolla en base al caso de claves de 128 bits, aunque pueden ser extendidos fácilmente a los casos restantes.

La clave se expande a una matriz de 4 filas y **Nb**(**Nr**+1) columnas: Siendo **Nb** (Tamaño del bloque a cifrar) y **Nr** (Numero de rondas a aplicar).

Tabla 48. Matriz de claves

K00	K01	K02	K03	…
K10	K11	K12	K13	…
K20	K21	K22	K23	…
K30	K31	K32	K33	…

Cada columna de la matriz de claves se denominará w[i] partiendo de W0; para entender mejor la manera en que opera KeySchedule se trabaja a la par con la clave inicial dada como ejemplo:

Clave inicial: 2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C

Partiendo de la clave dada, se obtienen los valores para las columnas w0, w1, w2 y w3, quedando de la siguiente manera:

- **w0**= 2B 7E 15 16
- **w1**= 28 AE D2 A6
- **w2**= AB F7 15 88
- **w3**= 09 CF 4F 3C

Cuadro 68. Matriz de claves del ejemplo

w0	w1	w2	w3
2B	28	AB	09
7E	AE	F7	CF
15	D2	15	4F
16	A6	88	3C

Como se trabaja con bloques de 128 bits y una clave del mismo tamaño, se calcula el número de columnas w[i] que se necesita:

Primero el número de rondas **(Nr** = máx (Nk, Nb) + 6), donde:

- **Nk**= (128 bits / 8 bits) / 4 filas = 4
- **Nb** = (128 bits / 8 bits) / 4 filas = 4
- Por tal, **Nr** = máx (4, 4) + 6 = 10 rondas

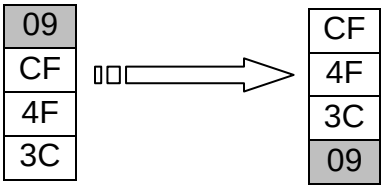
Segundo se calcula el número de columnas w[i] requeridos = (Tamaño del bloque de clave * (Nr + 1) / 8 bits de cada pareja / 4 columnas):

- w[i] requeridos = 128 bits * (10 rondas + 1) = 128 * (11) = 1408 bits
- 1408 bits / 8 bits = 176 bits
- 176 bits / 4 columnas = **44** columnas w[i]

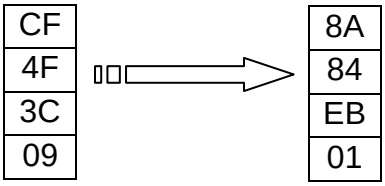
Teniendo ya el dato de 44 columnas a hallar, se debe mencionar que para las w[i] donde "**i**" es múltiplo de 4 se aplica una función en especial, de resto simplemente son sumas XOR.

En la primera ronda se pasará a calcular w[4], como ya se mencionó, al al ser múltiplo de 4 se sigue un procedimiento especial aplicando una serie de operaciones:

1. **RotByte,** es una rotación circular hacia arriba de una posición de la columna temporal: en este caso es la columna formada por w[3].



2. **SubByte**, es la sustitución de byte, de acuerdo con la tabla ya conocida Caja-S AES.



3. **Temp XOR Rcon [i/Nk]** se aplica un XOR entre el Temp, es decir la columna que sale de las anteriores operaciones, y el Rcon [i/Nk], en este caso resulta:

   Donde Rcon se obtiene de acuerdo a la siguiente forma:

   Rcon [i] = (RC[i],00,00,00);
   RC[i] = x**.** RC[i-1] = x^{i-1}

   Es decir, desde el punto de vista de polinomios y resultado hexadecimal:

   RC[1] = 01
   RC[2] = x^1 = 02
   RC[3] = x^2 = 04
   RC[4] = x^3 = 08
   RC[5] = x^4 = 10
   RC[6] = x^5 = 20
   RC[7] = x^6 = 40

RC[8] = x^7 = 80
RC[9] = $x^8 = x^4 + x^3 + x^1 + 1$ = 1B
RC[10] = $x^9 = x^5 + x^4 + x^2 + x$ = 36

Rcon [4/4] = (01,00,00,00)
Temp $\oplus$ Rcon [4/4] = Nueva columna temporal

8A		01	=	8B
84	$\oplus$	00	=	84
EB		00	=	EB
01		00	=	01

4. Por último se tiene otro XOR de la nueva columna temporal con W[i – 4], en este caso w[0].

Temp $\oplus$ w[i-4] =

8B		2B	=	A0
84	$\oplus$	7E	=	FA
EB		15	=	FE
01		16	=	17

De esta manera, finalizada esta ronda se obtiene w[4] y la clave queda de la siguiente manera:

Cuadro 69. Matriz de la clave actual

w0	w1	w2	w3	w4
2B	28	AB	09	A0
7E	AE	F7	CF	FA
15	D2	15	4F	FE
16	A6	88	3C	17

Las siguientes rondas hasta llegar a calcular w[8], no siguen el procedimiento especial explicado hasta ahora (Únicamente cuando "**i**" es múltiplo de 4) y simplemente se realiza la última operación en la que se realiza un XOR del temp con w[i-4]. De esta forma se obtiene w[5],w[6],w[7].

w[5] = w[4] temp XOR w[1]

w[6] = w[5] temp XOR w[2]

w[7] = w[6] temp XOR w[3]

Tras estas 3 rondas la clave tendrá la forma:

Cuadro 70. Matriz de la clave actual

w0	w1	w2	w3	w4	w5	w6	w7
2B	28	AB	09	A0	88	23	2A
7E	AE	F7	CF	FA	54	A3	6C
15	D2	15	4F	FE	2C	39	76
16	A6	88	3C	17	B1	39	05

En la ronda en la que se pasa a calcular w[8] ocurre como en w[4], con el procedimiento especial nuevamente; todas estas operaciones hasta rellenar las **44** columnas de las que está compuesta la clave en este caso.

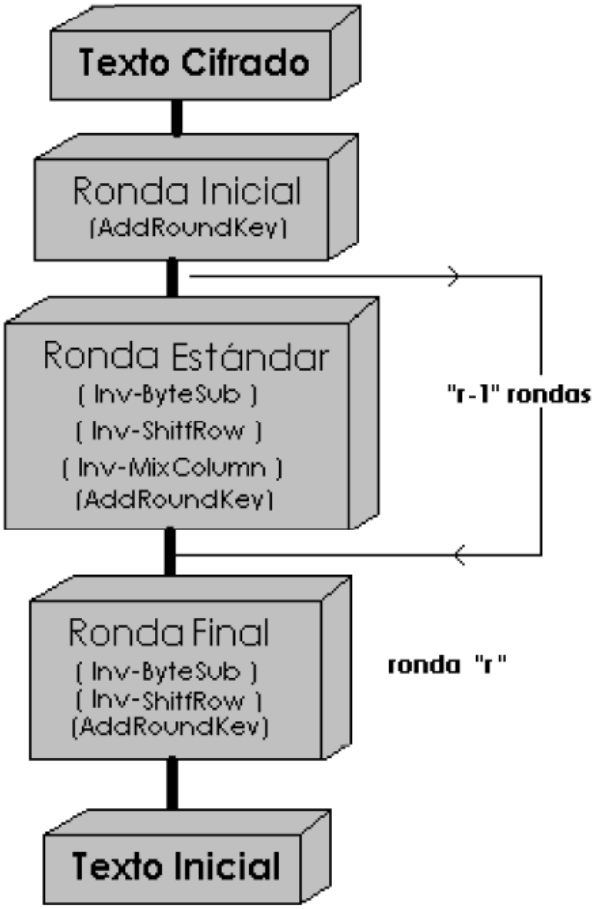
7.4.4. Descifrado con el AES

El proceso de descifrado es muy similar al cifrado, sólo hay que hacer el proceso inverso, es decir, invertir el orden de todas las operaciones realizadas, y hacer las transformaciones inversas.

Cabe destacar que en este proceso, las subclaves utilizadas, van desde la última generada en el proceso de cifrado hasta la primera (que corresponderá con bytes de la clave elegida para cifrar).

De forma gráfica:

Figura 14. Descifrado del AES



Fuente: Criptografía de clave privada

Como se mencionó en los apartados anteriores, las funciones matemáticas utilizadas son invertibles, de esta forma a la hora de descifrar serán las inversas de las funciones utilizadas.

La función ***AddRoundKey*** desempeña la misma función salvo que en vez de empezar en la primera ronda empezará por la última y acabará por la primera.
La función ***Inv-ByteSub*** realiza la aplicación inversa de la correspondiente Caja-S AES correspondiente a cada byte de la matriz de estado. De esta manera se obtiene otra tabla con los valores inversos a los valores de la Caja-S AES.

Tabla 49. Tabla inversa Caja-S AES

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	B3	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

La función ***Inv-ShiftRow*** es inversa a la función ***ShiftRow*** en la que en vez de desplazar las filas de la matriz hacia la izquierda se desplazan a la derecha, el mismo número de posiciones que se desplazó con anterioridad.

Por último la función ***Inv-MixColumns*** es la inversa de la función ***MixColumns*** en la que se debe operar sobre los bytes de una misma columna, considerando las columnas como polinomios con coeficientes en GF(2^8), que son multiplicados por el polinomio d(x) = 0B x^3 + 0D x^2 + 09 x^1 + 0E x^0 , que es el inverso de c(x).

De forma matemática: S(x) = d(x) XOR S'(x)

Donde S(x) representa la matriz de estado resultante de esta etapa, y S'(x) representa la matriz de ***Estado*** de entrada.

Esta fórmula para comprenderla más fácilmente se puede expresar en forma matricial de la siguiente manera.

$$\begin{bmatrix} S'0c \\ S'1c \\ S'2c \\ S'3c \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \cdot \begin{bmatrix} S0c \\ S1c \\ S2c \\ S3c \end{bmatrix}$$

Y con esta última etapa, se realiza el número de rondas necesario para obtener de nuevo el mensaje original, de igual manera que como se utilizaron para el cifrado.

7.5. RSA CIFRADO MEDIANTE PRIMOS

7.5.1. Introducción

El método de cifrado de datos conocido como algoritmo RSA, por los nombres de sus inventores (Rivest, Shamir y Adleman), es un sistema criptográfico de clave pública. El algoritmo fue descrito en 1977 por Ron **R**ivest, Adi **S**hamir y Len **A**dleman, del Instituto Tecnológico de Massachusetts (MIT); Clifford Cocks, un matemático británico que trabajaba para la agencia de inteligencia británica GCHQ, había descrito un sistema equivalente en un documento interno en 1973. Debido al elevado coste de las computadoras necesarias para implementarlo en la época, su idea no trascendió. Su descubrimiento, sin embargo, no fue revelado hasta 1997 ya que era confidencial, por lo que **R**ivest, **S**hamir y **A**dleman desarrollaron **RSA** de forma independiente.

El algoritmo fue patentado por el MIT en 1983 en Estados Unidos con el número ***4.405.829***. Esta patente expiró el 21 de septiembre de 2000. Como el algoritmo fue publicado antes de patentar la aplicación, esto impidió que se pudiera patentar en otros lugares del mundo. Dado que Cocks trabajó en un organismo gubernamental, una patente en Estados Unidos tampoco hubiera sido posible.

En la actualidad, RSA es el primer y más utilizado algoritmo de este tipo y es válido tanto para cifrar como para firmar digitalmente.

La seguridad de este algoritmo radica en el problema de la factorización de números enteros. Los mensajes enviados se representan mediante números, y el funcionamiento se basa en el producto, conocido, de dos números primos grandes elegidos al azar y mantenidos en secreto. Actualmente estos primos son del orden de 10^{200}, y se prevé que su tamaño aumente con el aumento de la capacidad de cálculo de los ordenadores.

Como en todo sistema de clave pública, cada usuario posee dos claves de cifrado: una pública y otra privada. Cuando se quiere enviar un mensaje, el emisor busca la clave pública del receptor, cifra su mensaje con esa clave, y una vez que el mensaje cifrado llega al receptor, este se ocupa de descifrarlo usando su clave privada.

Se cree que RSA será seguro mientras no se conozcan formas rápidas de descomponer un número grande en producto de primos. La computación cuántica podría proveer de una solución a este problema de factorización.

7.5.2. Las bases matemáticas que hacen fuerte al RSA

En los algoritmos ya descritos, el **DES** y el **AES**, se notó que su fortaleza radica en la mezcla de la complejidad del algoritmo y sus bases matemáticas, pero principalmente en el primero. El **RSA** como se pudo ver, es un algoritmo bastante sencillo, ya que su fortaleza radica en las bases matemáticas que posee, las cuales se rigen por los siguientes:

- ✓ Anillo de enteros.
- ✓ División de enteros. A = b*(cociente) + residuo.
- ✓ Múltiplos y divisores de enteros.
- ✓ Máximo común divisor.
- ✓ Mínimo común múltiplo.
- ✓ Números primos y números primos entre sí. (Lo que hace fuerte al RSA).
- ✓ Descomposición en factores primos.
- ✓ ***Algoritmo de Euclides.***
- ✓ ***Algoritmo extendido de Euclides.***
- ✓ ***Función multiplicativa de Euler***.

Todas las anteriores son bases matemáticas que la mayoría conocen, ya que son bases que vienen desde la primaria y la secundaria; es posible que las 3 últimas no sean de mucha familiaridad para muchos, por lo cual son las seleccionadas para ser definidas y explicadas a continuación.

7.5.2.1. Algoritmo de Euclides

Este es un método para calcular el máximo común divisor de dos números cualesquiera, sin necesidad de descomponerlos en factores primos. Esto es muy importante desde el punto de vista de la computación, porque descomponer un número en factores primos, según se sabe, es un algoritmo de coste exponencial en el tamaño del número, es decir, el tiempo que cuesta ejecutarlo crece exponencialmente con dicho tamaño. Sin embargo, el costo del algoritmo de Euclides es un tiempo lineal con el tamaño.
El algoritmo usa la siguiente proposición:

Proposición: Dados dos enteros positivos **a** y **b**, si se divide el mayor entre el menor, lo cual da un cociente **q** y un resto **r**, es decir, **a** = **bq** + **r**, se cumple que mcd(**a**, **b**) = mcd(**b**, **r**).

Demostración: Llámese **d** = mcd(**a**, **b**) y **d'** = mcd(**b**; **r**). Entonces **d** es por definición divisor de **b**; por otra parte, **r** = **a** - **bq**, y como **d** divide a **a** y también a **bq** (porque divide a **b**) entonces **d** es también divisor de **r**.

Por otra parte, por definición de **d'** se sabe que **d'** es divisor de **b** y de **r**. Se tienen entonces dos divisores de **b** y de **r**, siendo **d'** por definición el mayor de todos. Así pues, deberá ser **d** ≤ **d'**.

Por otra parte, **d'** es por definición divisor de **b**, y como **a** = **bq** + **r** y **d'** es divisor de **b** y de **r**, lo es de **a**. Se tiene entonces que **d'** es divisor de **b** y de **a** y también que **d** es divisor de **b** y de **a** y es el máximo de todos ellos. Por tanto, deberá ser **d'** ≤ **d**. El único modo de que simultáneamente **d** ≤ **d'** y **d'** ≤ **d** es que **d'** = **d**, como se quería probar.

El algoritmo de Euclides se basa en la anterior proposición: dados **a** y **b**, se calcula la división entera entre ellos, obteniendo un cociente **q** y un resto **r**. Entonces se calcula de nuevo la división entera entre **b** y **r**, obteniendo un nuevo cociente y un nuevo resto. El proceso se itera hasta que el resto sea 0, en cuyo caso el último divisor **b** es el máximo común divisor.

Es fácil ver que la sucesión de los restos es estrictamente decreciente, y como se trata de una sucesión de números positivos, llegará a valer 0, con lo que el algoritmo efectivamente acaba.

Ejemplo: sean **a** = 643 y **b** = 412. Entonces, escribiendo en cada línea a = q**b** + *r*,

1) 643 = 1 *x* **412** + *231* (Al dividir 643 entre 412)
2) 412 = 1 *x* **231** + *181* (Al dividir 412 entre 231)
3) 231 = 1 *x* **181** + *50* (Al dividir 231 entre 181)
4) 181 = 3 *x* **50** + *31*
5) 50 = 1 *x* **31** + *19*
6) 31 = 1 *x* **19** + *12*
7) 19 = 1 *x* **12** + *7*
8) 12 = 1 *x* **7** + *5*
9) 7 = 1 *x* **5** + *2*
10) 5 = 2 *x* **2** + *1*
11) 2 = 2 *x* **1** + *0*

Por tanto, el mcd de 643 y 412 es el último divisor **b**, o sea, **1**. En este caso particular se ha encontrado que 643 y 412 son primos entre sí. En general, obsérvese que en cada paso el algoritmo sustituye cada dividendo por el último resto obtenido, y cada divisor por el penúltimo resto, calculando entonces de nuevo la división para generar los actuales cociente y resto.

Cabe destacar que esto no vale para los dos primeros pasos, en los que aún no se tienen dos restos anteriores.

7.5.2.2. Algoritmo extendido de Euclides

Una variación interesante del algoritmo de Euclides que es muy útil para el RSA, es el algoritmo extendido. Es posible probar que el máximo común divisor de dos enteros **d** = mcd(**a**, **b**) (y en general cualquier otro divisor) se puede escribir siempre como combinación lineal de ambos, o sea, existen dos enteros ƛ y µ tales que **d** = ƛ **a** + µ **b**. A esta igualdad se la llama "Identidad de Bezout".

Para encontrar dichos enteros basta recorrer los pasos del algoritmo de Euclides visto antes, al revés, realizando en cada paso solo las sustituciones hacia atrás que sean pertinentes. En el paso i-ésimo se debe sustituir en el despeje del resto correspondiente a dicho paso; debe hacerse en uno de sus dos sumandos cada vez, y usando para ello la igualdad del paso (i – 2), excepto en el primero, en que se usa la igualdad del paso inmediatamente anterior. En el ejemplo visto antes, se

comienza en la línea que contiene el mcd (línea 10) y se despeja (en este caso vale 1):

1 = 5 - 2 *x* 2

Ahora, se sustituye en el segundo sumando usando la línea 9
1 = 5 - 2 *x* (7 - 1 *x* 5) = 5 - 2 *x* 7 + 2 *x* 5 = 3 *x* 5 - 2 *x* 7

Se sustituye en el primer sumando, usando la línea 8
1 = 3 *x* (12 - 1 *x* 7) - 2 *x* 7 = 3 *x* 12 - 3 *x* 7 - 2 *x* 7 = 3 *x* 12 - 5 *x* 7

Se sustituye en el segundo sumando usando la línea 7
1 = 3 *x* 12 - 5 *x* (19 - 1 *x* 12) = 3 *x* 12 - 5 *x* 19 + 5 *x* 12 = 8 *x* 12 - 5 *x* 19

Se sustituye en el primer sumando usando la línea 6
1 = 8 *x* (31 - 1 *x* 19) - 5 *x* 19 = 8 *x* 31 - 8 *x* 19 - 5 *x* 19 = 8 *x* 31 - 13 *x* 19

Se sustituye en el segundo sumando usando la línea 5
1 = 8 *x* 31 - 13 *x* (50 - 1 *x* 31) = 8 *x* 31 - 13 *x* 50 + 13 *x* 31 = 21 *x* 31 - 13 *x* 50

Se sustituye en el primer sumando usando la línea 4
1 = 21 *x* (181 - 3 *x* 50) - 13 *x* 50 = 21 *x* 181 - 63 *x* 50 - 13 *x* 50 = 21 *x* 181 - 76 *x* 50

Se sustituye en el segundo sumando usando la línea 3
1 = 21 *x* 181 - 76 *x* (231 - 1 *x* 181) = 21 *x* 181 - 76 *x* 231 + 76 *x* 181 = 97 *x* 181 - 76 *x* 231

Se sustituye en el primer sumando usando la línea 2
1 = 97 *x* (412 - 1 *x* 231) - 76 *x* 231 = 97 *x* 412 - 97 *x* 231 - 76 *x* 231 = 97 *x* 412 - 173 *x* 231

Se sustituye en el segundo sumando usando la línea 1
1 = 97 *x* 412 - 173 *x* (643 - 412) = 97 *x* 412 - 173 *x* 643 + 173 *x* 412 = 270 *x* 412 - 173 *x* 643

De donde se ha hallado que 1 = ʎ *x* 412 + μ *x* 643 con **ʎ** = 270 y **μ** = -173.

7.5.2.3. Función multiplicativa de Euler

El siguiente concepto es muy importante en el desarrollo de RSA, es la definición y propiedades de la llamada *función multiplicativa de Euler*. Informalmente, si se tiene un entero **p**, éste será divisible por algunos, aunque no todos, los enteros anteriores a él (en el caso extremo de que fuese primo, por ninguno). O sea, algunos de estos enteros serán primos relativos con **p** y otros no. La función de Euler dice, para cada **p**, cuántos lo son.

Definición: Se define la función multiplicativa de Euler, $\phi(\mathbf{r})$, como el número de números enteros mayores o iguales que 1 y menores que **r** que son primos relativos con él, es decir:

$\phi(\mathbf{r}) = \#\{s / s >= 1 \text{ y } s < \mathbf{r} \text{ y } mcd(s, \mathbf{r}) = 1\}$
Donde # denota el cardinal del conjunto.

Véase ahora como, dado un número **r**, se puede calcular el valor de $\phi(\mathbf{r})$:

1. Si **r** es primo, $\phi(\mathbf{r}) = r - 1$
2. Si **r** es el producto de dos números primos entre sí, o sea, $\mathbf{r} = \mathbf{pq}$ con $mcd(\mathbf{p}, \mathbf{q}) = 1$, entonces $\phi(\mathbf{r}) = \phi(p)\ \phi(q)$
3. Si $\mathbf{r} = \mathbf{p}^k$ con **p** primo, entonces $\phi(\mathbf{r}) = \mathbf{p}^k - \mathbf{p}^{k-1} = \mathbf{p}^k(\mathbf{p} - 1)$

7.5.3. Descripción del cifrado

El algoritmo RSA puede ser explicado de la siguiente manera:

Tenemos dos sujetos **A** y **B** que quieren comunicarse, pero solo lo pueden hacer a través de un medio en el que cualquier otro sujeto **C**, puede coger ese mensaje y leerlo sin problema. ¿De qué manera pueden enviar un mensaje que no pueda ser leído por otra persona ajena a **A** y **B**?

- ✓ La explicación es muy sencilla:

A envía una caja fuerte con un candado abierto, a **B**. Luego de esto **B** mete su mensaje en la caja y cierra esta con el candado que venía en ella, de modo que cuando la caja es enviada a **A** de regreso, este puede abrirla, ya que es él quien posee la llave de dicho candado.

Técnicamente, **A** envía a **B** un «mensaje llano» *M* en forma de un número *m* menor que otro número *n*, mediante un protocolo reversible conocido como *padding scheme* («patrón de relleno»). A continuación genera el «mensaje cifrado» *c* mediante la siguiente operación:

$$c = m^e \pmod n,$$

Donde *e* es la clave pública de **B**.

Ahora **A** descifra el mensaje en clave *c* mediante la operación inversa dada por:

$$m = c^d \pmod n$$

Donde *d* es la clave privada que solo **A** conoce.

Como se notará a continuación, RSA es un algoritmo bastante sencillo, y es posible que no se note su fortaleza, pero sus principios matemáticos, basados en números primos son los que hacen que este pequeño pueda ser una fortaleza.

7.5.3.1. Construcción de la clave

Inicialmente es necesario generar aleatoriamente dos números primos suficientemente grandes (64 bits como mínimo), a los que se llamaran p y q.

A continuación se calcula **n** como producto de **p** y **q**:

$$n = p * q$$

Se calcula ϕ:

$$\phi(n) = (p-1)(q-1),$$

Donde ϕ es la función de Euler.

Se calcula un número natural "**e**" de manera que MCD (e, ϕ(n)) = 1, es decir, "**e**" debe ser primo relativo de ϕ(n). Es lo mismo que buscar un número impar por el que dividir ϕ(n) de cero como resto.

Mediante el algoritmo extendido de Euclides se calcula d = inverso de "**e**" en Zn:

$$d: e.d \pmod{\phi(n)} = 1$$

Puede calcularse:

$$d = ((Y * \phi(n)) + 1) / e$$

Para Y = 1,2,3,... hasta encontrar un **d** entero.

Ya se tiene la clave publica que son el par de números (e, n), y se tiene la clave privada que son el par de números (d, n).

La dificultad de descifrado de RSA radica en la dificultad del cálculo de ϕ(n), por ello se debe tomar **p** y **q** lo suficientemente grandes.

7.5.3.2. Como se envía un mensaje cifrado

La persona que envía el mensaje, ha de conocer la parte pública de la clave del destinatario.

Supóngase que se tiene el siguiente mensaje: $x_1, x_2, x_3, \ldots, x_n$.

Se procede a hacer $x^e \bmod n = y$.

En otras palabras la función de cifrado es $C = M^e \bmod n$.

7.5.3.3. Recepción del mensaje

Teniendo a **y** se hace $y^d \bmod n$ ¿=? X.

En Zn: $y = x^e$

$Y^d = x^{ed}$ $\quad\quad$ $d = e^{-1}$ en Z ϕ(n)

E * d = 1 en Z ϕ(n)

Luego: $x^{ed} = x^{1+k\phi(n)} = x * (x^{\phi(n)})^k = x$

Sabiendo que $x^{\phi(n)} = 1$ en Z ϕ(n) según el teorema de Euler.

En resumen la función de descifrado es $M = C^d \bmod n$.

7.5.4. Ejemplo de cifrado RSA

Aquí se tienen dos ejemplos de cifrado/descifrado con RSA. Los parámetros usados aquí son pequeños y orientativos con respecto a los que maneja el algoritmo en realidad:

Primer ejemplo:

1) Se seleccionan dos números primos: **p** = 7; **q** = 17
2) Se calcula **n** = **p** x **q**; 17 x 7 = 119
3) Se calcula ϕ(n) = (p-1) x (q-1) = 6 x 16 = 96
4) Se selecciona **e** tal que 1 < **e** < ϕ(n) y mcd (ϕ(n), **e**) = 1;
1< **e** < 96 y mcd (96, **e**) = 1, **e** = 5 (Este es uno de los posibles).
5) Se realiza el cálculo de **d** tal que: **d** x **e** mod ϕ(n) = 1;
d x 5 mod(96) = 1, **d** = 77

- ✓ La clave publica es {e, n} = {5, 119}.
- ✓ La clave privada es {d, n} = {77, 119}.

Mensaje M = 19.

Cifrado **C** = $M^e \bmod n$;
$M^e = 19^5 = 2476099$; **C** = 2476099 mod 119
C = 66, Mensaje cifrado.

Descifrado **M** = $C^d \bmod n$;
$C^d = 66^{77} = 1{,}2731x\ e{+}140$; **M** = (1,2731x e+140) mod 119
M = 19, Mensaje original.

Segundo Ejemplo:

Para este ejemplo se ha seleccionado **p** = 3 y **q** = 11, dando **n** = 33 y **ϕ(n)** = 20, un valor adecuado de **e** es **e** = 7, puesto que 7 y 20 no tienen factores comunes, con estas selecciones, **d** puede encontrarse resolviendo la ecuación 7 x **d** (mod 20) = 1, que produce **d** = 3.

El texto cifrado **C**, de un mensaje de texto normal **M**, se da por la regla ***C*** = M^7 (mod 33). El texto cifrado lo descifra el receptor de acuerdo con la regla **M** = C^3 (mod 33).

La palabra que se desea cifrar es "CASCO".

Para este ejemplo utilícese la siguiente equivalencia:

Cuadro 71. Equivalencia numérica letras

A	B	C	D	E	F	G	H	I	J	K	L	M	N
01	02	03	04	05	06	07	08	09	10	11	12	13	14

Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
15	16	17	18	19	20	21	22	23	24	25	26	27

EMISOR:

CASCO = 03 01 20 03 16 = Mensaje a cifrar.

Mensaje cifrado **C** = M^e mod n

M^e = {03^7, 01^7, 20^7, 03^7, 16^7} = {2187, 1, 1280000000, 2187, 268435456}
M^e *mod n* = {2187 mod (33), 1 mod(33), 1280000000 mod(33), 2187 mod(33), 268435456 mod(33)}

M^e *mod n* = {09, 01, 26, 09, 25} = Mensaje cifrado = ***IAYIX***

RECEPTOR:

IAYIX = 09 01 26 09 25 = Mensaje a descifrar

Descifrado **M** = C^d mod n.

C^d = {09^3, 01^3, 26^3, 09^3, 25^3} = {729, 1, 17576, 729, 15625}

C^d *mod n* = {729 mod(33), 1 mod(33), 17576 mod(33), 729 mod(33), 15625 mod(33)}

C^d *mod n* = {03, 01, 20, 03, 16} = Mensaje Original = ***CASCO***

7.6. COMPARACION COMPLEJIDAD DES, AES, RSA

Con el cifrado Triple DES, la variabilidad de la permutación inicial aumenta la complejidad, la longitud de las claves es de 112 bits, todo en conjunto redunda en el incremento de la seguridad del criptosistema hibrido.

Si con un equipo que lleve a cabo 250 millones de claves por segundo y se tarda un segundo para romper el cifrado DES por medio de fuerza bruta se requerirían 2,285 billones de años para 'quebrar' un cifrado con Triple DES; con el crecimiento de la tecnología es importante incrementar el tamaño de las llaves.

7.6.1. Complejidad de la ruptura del DES vs AES vs RSA

En DES: Es un algoritmo de complejidad polinomial, lo cual permite que sea computacionalmente muy tratable, pero esto se convierte en arma de doble filo, puesto que los computadores pueden ser utilizados para descubrir rápidamente criptosistemas ingenuos, deberían utilizarse algoritmos de cifrado, que no tengan debilidades estadísticas y matemáticas, y que sean computacionalmente imposibles de descubrir con el fin de hacer que el intento de quebrantarlos consuma un tiempo prohibitivo. Al mismo tiempo, la complejidad computacional del cifrado y descifrado debería ser razonable ya que representa recargos de procesamiento que incrementan los retardos de comunicación.

Un algoritmo que se cree proporciona un compromiso razonable entre estas exigencias es la norma de Encriptacion Estandar de Datos (DES, Data Encryption Standard).

El *criptoanálisis diferencial* fue descubierto a finales de los 80 por Eli Biham y Adi Shamir, aunque era conocido anteriormente tanto por la NSA como por IBM y mantenido en secreto. Para romper las 16 rondas completas, el criptoanálisis diferencial requiere 2^{47} textos planos escogidos. DES fue diseñado para ser resistente al CD.

El *criptoanálisis lineal* fue descubierto por Mitsuru Matsui, y necesita 2^{43} textos planos conocidos (Matsui, 1993); el método fue implementado (Matsui, 1994), y fue el primer criptoanálisis experimental de DES que se dio a conocer. No hay evidencias de que DES fuese adaptado para ser resistente a este tipo de ataque. Una generalización del CL — el *criptoanálisis lineal múltiple* — se propuso en 1994 Kaliski and Robshaw), y fue mejorada por Biryukov y otros (2004); su análisis sugiere que se podrían utilizar múltiples aproximaciones lineales para reducir los requisitos de datos del ataque en al menos un factor de 4 (es decir, 2^{41} en lugar de 2^{43}). Una reducción similar en la complejidad de datos puede obtenerse con una variante del criptoanálisis lineal de textos planos escogidos (Knudsen y Mathiassen, 2000). Junod (2001) realizó varios experimentos para determinar la complejidad real del criptoanálisis lineal, y descubrió que era algo más rápido de lo predicho, requiriendo un tiempo equivalente a 2^{39}–2^{41} comprobaciones en DES.

El *ataque mejorado de Davies*: mientras que el análisis lineal y diferencial son técnicas generales y pueden aplicarse a multitud de esquemas diferentes, el ataque de Davies es una técnica especializada para DES. Propuesta por vez primera por Davies en los 80, y mejorada por Biham and Biryukov (1997). La forma más potente del ataque requiere 2^{50} textos planos conocidos, tiene una complejidad computacional de 2^{50}, y tiene un 51% de probabilidad de éxito.

Existen también ataques pensados para versiones del algoritmo con menos rondas, es decir versiones de DES con menos de dieciséis rondas. Dichos análisis ofrecen una perspectiva sobre cuantas rondas son necesarias para conseguir seguridad, y cuánto «margen de seguridad» proporciona la versión completa. El criptoanálisis diferencial-lineal fue propuesto por Langford y Hellman en 1994, y combina criptoanálisis diferencial y lineal en un mismo ataque. Una versión mejorada del ataque puede romper un DES de 9 rondas con $2^{15.8}$ textos planos conocidos y tiene una complejidad temporal de $2^{29.2}$ (Biham y otros, 2002).

En AES: Es un algoritmo de complejidad polinomial, lo que permite que sea computacionalmente tratable, pero esto también permite que se le pueda atacar. Los primeros ataques conocidos para este algoritmo de Encripción fueron publicados en 1998, el primer ataque únicamente logra romper la seguridad

teóricamente de el cifrador para 6 ciclos, mas tarde Gilbert y Miner en el año 2000 publicaron un ataque para el cifrador cuando usa 7 ciclos. Cabe recordar que Rijndael usa 10, 12 o 14 ciclos.

Otras técnicas usadas, como, La técnica de la suma parcial redujo considerablemente la complejidad del ataque a los 6 ciclos, técnica que es posible usar también con 7 y 8 ciclos. En algunos casos donde se puede usar textos conocidos el factor de trabajo se ve reducido. Todos estos ataques se han basado en los trabajos y en las técnicas del ataque Cuadrado y únicamente se han trabajado con tamaño de bloque de 128-bits.

La técnica del ataque Cuadrado puede ser usada en ataques contra el algoritmo de Rijndael con 6, 7 y 8 ciclos y pretende encontrar la llave, utilizando 256 textos encriptados, que solamente se diferencien en un byte, después aplicar MixColumn en el primer ciclo y luego intercambiar el orden de las transformaciones MixColumn y AddRoundKey en el ultimo ciclo, con este cambio y un análisis empírico los investigadores Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Mike Stay, David Wagner y Doug Whiting, se dieron cuenta que el texto encriptado depende de cuatro bytes de la RoundKey del ultimo ciclo, y de esta manera reducir el número considerable de llaves posibles para un texto encriptado. Sin embargo en esta técnica se puede ver que es necesaria una gran cantidad de textos encriptados y un gran procesamiento para su análisis. Además a pesar de que se tiene la misma estructura para tamaños de bloques mayores de 128, su criptoanálisis debe ser tratado de diferente manera.

En RSA: El tamaño de las llaves son números primos lo suficientemente grandes y fuertes ante los ataques a este cifrado. En RSA es encontrar el numero de operaciones para realizar las factorizaciones en promedio es de e^{350}. El uso de llaves más grandes en RSA es de 10^{220} a 10^{230} dígitos.

Llaves de 1024 bits en RSA (llave privada y publica): se estima que se requieren de más de 70 años con las computadoras más potentes y un presupuesto de mas de 100 millones de dólares para poder quebrar este algoritmo.

En la tabla se muestra de acuerdo a la longitud de las claves en cifrados simétricos el tiempo para una búsqueda exhaustiva de claves para romper el código.

Tabla 50. Longitud de claves en cifrados simétricos

Algoritmo	Tamaño de la clave (bits)	Complejidad	Tamaño de bloque (bits)	No. de etapas	Tiempo necesario a 10^6 cifrados/microsegundos
DES	56	$2^{56} \approx 7.2x10^{16}$	64	16	10.01 hr.
Triple DES	112 168	2^{112} $2^{168} \approx 3.7x10^{50}$	64	48	$5.9x10^{30}$ años.
AES	128 192 256	$2^{128} \approx 3.4x10^{38}$ 2^{192} 2^{256}	128	10 12 14	$5.4x10^{18}$ años.

En la siguiente tabla se muestra por el método de fuerza bruta el tiempo para encontrar la factorización de n, en el cifrado asimétrico RSA.

Tabla 51. Longitud de los números primos del cifrado asimétrico RSA

Tamaño de la clave (dígitos)	No. de Operaciones	Tiempo
50	1.4 x 1010	3.9 hr
75	9.0 x 1012	104 días
100	2.3 x 1015	74 años
200	1.2 x 1023	3.8×10^9 años
300	1.5 x 1029	4.9×10^{13} años
500	1.3 x 1039	4.2×10^{25} años

El nivel de seguridad en bits de RSA comparado con los criptosistemas Triple DES, AES se muestra en la siguiente tabla:

Tabla 52. Nivel de seguridad de RSA con Triple DES y AES.

	Nivel de seguridad (bits)			
Criptosistema	Triple DES (112)	AES (128)	AES (192)	AES (256)
RSA	2048	3072	8192	15360

Complejidad Computacional: Es esencialmente el tiempo de ejecución de un algoritmo, en función del “tamaño" de los datos de entrada. La palabra “tiempo" no

hace referencia al tiempo físico, que dependerá en buena medida del desarrollo de los ordenadores, sino a la cantidad de operaciones bit, entendiendo como operación bit una operación básica entre dos bits.

Esta cantidad de operaciones se mide, en algoritmos deterministas, en el peor caso posible, es decir, para una distribución de datos que hace el algoritmo lo más largo posible.

El estudio de la complejidad de un algoritmo dirá cómo de eficaz es en la práctica dicho algoritmo.

Tabla 53. Complejidad ciclomatica del AES

Algoritmo	Complejidad
Ronda de agregación de llave	1
Expansión de la llave	13
Mezcla de columnas	4
Suma de puntos	1
Generador del polinomio	3
Multiplicación de polinomio	5
Generación de la caja S	5
Cifrado de datos	1
Descifrado de datos	1
Inversión del cifrado	9

Tabla 54. Complejidad Ciclomatica del RSA

Algoritmo	Complejidad
Calculo de los valores primos	12
Cifrado de datos	5
Descifrado de datos	4

Las tablas muestran que el nivel de complejidad al realizar la generación de parámetros de RSA es mayor a la de los otros dos criptosistemas debido a que seleccionar números primos grandes de una operación compleja, y la aritmética del cifrado y el descifrado también lo es. Para AES la complejidad se encuentra en la elaboración de las tablas de sustitución, la mezcla de las columnas y el intercambio de filas, todo el trabajo de matrices; sin embargo, dicha complejidad se realiza una sola vez; en la actualidad los algoritmos funcionales incluyen las tablas como constantes lo cual facilita su uso y el proceso de cifrar y descifrar

ocurre realizando transformaciones en los bits, a diferencia de los sistemas asimétricos que trabajan con vectores y puntos para el cifrado.

7.6.2. Demostración complejidad RSA

Se dice que un algoritmo de complejidad f (n) es polinomial si existe d >= 0 tal que f (n) = $O(n^d)$. En caso contrario, se dice que el algoritmo es exponencial.

Se entiende que un algoritmo de complejidad polinomial es eficiente, mientras que uno de complejidad exponencial es ineficiente.

Para un número M se tiene que su longitud al escribirlo en base *a* es $[\log_a M] + 1$. Como $\log_a M = (\ln M / \ln a)$, en realidad no importa, en orden al cálculo de la complejidad, si se escribe el dato de entrada, M, en base 2 (bits) o en otra base.

Se denotará: long n = $[\log_2 n] + 1$.

En RSA las operaciones de cifrado y descifrado son iguales y consisten en calcular a^b mod n para enteros n, a y b.

Problema: Calcular a^b mod m.

Primera idea: Ir haciendo las multiplicaciones

- ✓ a^i mod m = (a^{i-1} mod m) (a mod m), para 2 <= i <= b.

Inconveniente: Su complejidad es O(bM(n)). Por tanto exponencial en long b:

- ✓ O(bM(n)) = O(2(long b)M(n)

Existe un algoritmo mucho más rápido que este, se llama algoritmo de potenciación modular o de cuadrados sucesivos.

Base: Se hace la división de b entre 2: b = 2b' + b_0, b_0 <= 1. Entonces:

- ✓ $a^b = (a^2)^{b'} a^{b0}$, y, a^{b0} = a o a^{b0} = 1.

Inicio: Se escribe b en forma binaria, b = $b_s b_{s-1} \ldots b_0$
r := 1; x := a mod n.

Iteración: Para 0 <= i <= s:

Si $b_i = 1$ entonces r := rx mod n.
x := x2 mod n.

Salida: La salida del algoritmo es r.

Obsérvese que se hacen s + 1 = long *b* etapas. En cada una hay que calcular un cuadrado y, en el peor de los casos, un producto, junto con sus reducciones módulo m (división euclídea), por lo que la complejidad es:

- ✓ O((long b)M(m)) = O((long b)(longm)2)

Lo que convierte el algoritmo en muy eficiente.

8. SEGURIDAD EN LAS REDES

8.1. INTRODUCCIÓN

Este ultimo capitulo pretende ser una pequeña muestra de las aplicaciones criptográficas más importantes en la actualidad. En los capítulos previos se han estudiado de forma detallada los principios básicos de la Criptografía en sus aspectos teóricos y matemáticos. En el capitulo actual se describe el uso de estos conceptos en varias aplicaciones que permiten realizar comunicaciones seguras a través de las nuevas redes de transmisión. No se pretende dar más que una idea general del tema: un desarrollo amplio y detallado del mismo necesitaría no solo un capitulo, sino un libro completo.

En 1976 se da un grandioso desarrollo de esta ciencia cuando Diffie y Hellman publicaron el articulo “New Directions in Cryptography” donde se introdujo el concepto revolucionario de criptografía de llave pública y un nuevo método muy ingenioso para el intercambio de la llave (o clave). La infalibilidad de este procedimiento se basa en el problema matemático del algoritmo discreto. En 1978 Rivest, Shamir y Adleman descubrieron el primer esquema de encriptación de llave pública y un procedimiento de firmas que ahora se conoce como **RSA**, este método se basa en otro problema matemático que también provee infalibilidad (inflexibilidad para resolverlo), se refiere al problema de factorización de enteros largos.

Luego vinieron las firmas digitales, afianzando las teorías de encriptación de llave pública, en 1991se adopto el primer estándar para firmas digitales llamado ISO/IEC 9796 basado en el esquema RSA. Por tal motivo, en primer lugar se exponen, de forma general, las necesidades de seguridad de las redes de comunicación y los mecanismos ideados para satisfacerlas. Posteriormente, se citan las recomendaciones dadas por distintos organismos internacionales para la normalización e inclusión de los elementos de seguridad en los equipos y sistemas, también se hace referencia a los algoritmos Hash y a las firmas digitales. Finalmente, se describen diseños concretos para aplicaciones específicas, como servicios de mensajería, redes bancarias y de transacciones comerciales, etc.

8.2. PRINCIPIOS DE SEGURIDAD EN REDES DE COMUNICACIONES

La creciente expansión de las redes de comunicaciones ha hecho necesarios la adopción y el desarrollo de herramientas de seguridad que protejan tanto los datos transmitidos como el acceso a los elementos de la red de los posibles ataques que puedan sufrir. Son especialmente delicadas las redes informáticas, debido a ciertas características en su estructura, como es, por ejemplo, la ausencia de conexión física directa entre el emisor y el receptor, que no asegura la inmediata transmisión entre ambos, ni siquiera que esta llegue a producirse. Para que el emisor tenga certeza de que los datos transmitidos alcanzan su destino, son necesarios mensajes de confirmación de reparto o de lectura enviados por el receptor.

Los ataques contra un sistema de comunicaciones se pueden agrupar en cuatro categorías:

- **Interrupción:** algún elemento del sistema es puesto fuera de servicio.
- **Interceptación:** alguien no autorizado accede a cierta información o a cualquier elemento de la red.
- **Modificación:** alguien no autorizado, tras haber accedido a un mensaje o a cualquier elemento de la red, altera su contenido.
- **Fabricación:** alguien no autorizado, falsificando su identidad, inserta información en el sistema.

El ataque por interceptación se considera “pasivo”, mientras que los otros tres tipos se clasifican como ataques “activos”. Para poder detectar y prevenir estos ataques, las redes deben incorporar mecanismos que proporcionan servicios de seguridad. Estos servicios pueden resumirse en:

Confidencialidad:
Solo las personas o maquinas autorizadas pueden acceder a la información transmitida a través de una red de comunicaciones o al contenido de la información guardada en un sistema informático. En algunos casos, no solo hay que proteger el contenido de los mensajes (*confidencialidad del mensaje*), sino también las identidades del emisor y del receptor (*confidencialidad del tráfico de mensajes).*

Integridad:
Ninguna persona no autorizada ha de poder modificar la información transmitida o

almacenada. El mensaje debe llegar a su destino sin haber sufrido alteración alguna en su contenido o en el orden de la recepción de sus unidades si se compone de varios bloques.

Autenticación:
El origen de un mensaje ha de estar perfectamente identificado.

No –repudio:
Debe quedar constatado si un usuario envía o recibe algún mensaje. De esta manera, ni el emisor del mensaje ni el receptor del mismo pueden negar que se haya efectuado la transmisión.

Control de acceso:
Solo los usuarios autorizados debidamente identificados pueden obtener permiso de acceso a los recursos del sistema.

Disponibilidad:
El sistema no debe permitir que usuarios no autorizados dejen fuera de funcionamiento elementos de la red e impidan, así, las comunicaciones.

8.2.1. Aplicaciones de seguridad en redes de comunicación.

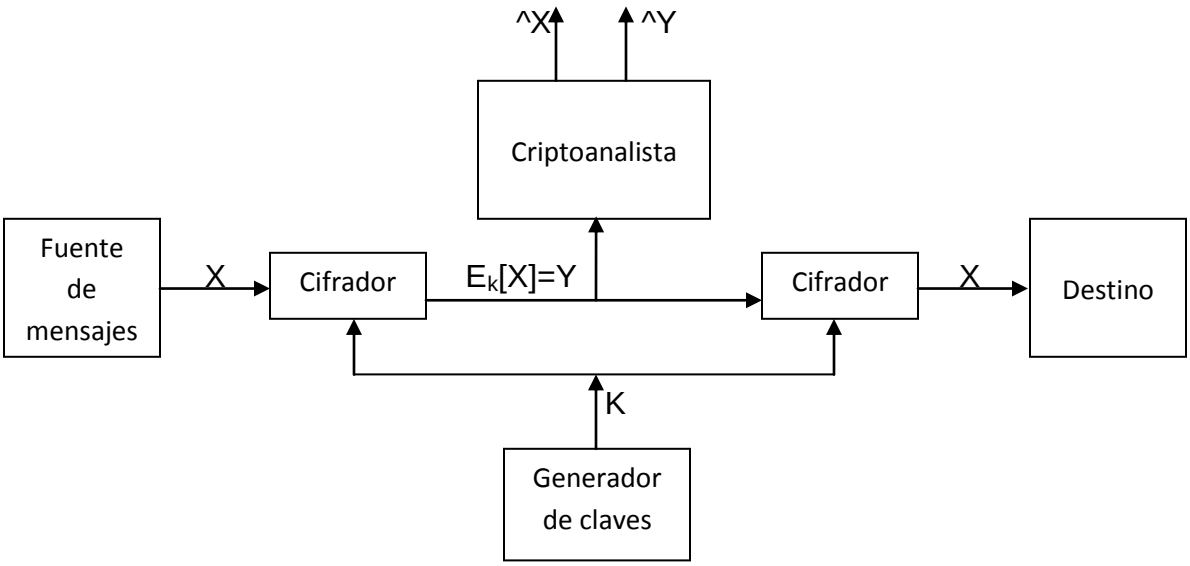
Tanto los sistemas convencionales o simétricos de cifrado como los más novedosos, basados en algoritmos de clave pública, ofrecen soluciones para los distintos servicios de seguridad.

En una red de paquetes conmutados, la confidencialidad de la información transmitida se puede conseguir mediante un cifrado simétrico que debe realizarse bien a nivel de enlace o bien extremo a extremo. En el esquema de la Figura 15 se muestra la colocación de los elementos de seguridad para el caso de un cifrado simétrico en una red de conmutación de paquetes.

En el caso de un cifrado a nivel de enlaces se necesitan equipos de cifrado en cada nodo de la red, tanto en su entrada como en su salida. El número de elementos de cifrado es muy alto, como se puede apreciar en la Figura 15, pero permite asegurar la confidencialidad del tráfico de mensajes. Sin embargo, la seguridad en los nodos de conmutación puede disminuir, ya que el mensaje a de ser descifrado para leer la dirección de destino que aparece en la cabecera del

paquete y en una red pública no se tiene ningún control sobre los nodos. Otro punto a tener en cuenta en este caso es la necesidad de una clave única para cada par de nodos, clave que ha de ser diferente para cada enlace entre nodos; este hecho, que supone una ventaja al tener que transmitir la clave a solo dos equipos, es a la vez, un inconveniente debido al elevado número de claves que resultan ser necesarias.

Figura 15. Esquema de criptosistema simétrico



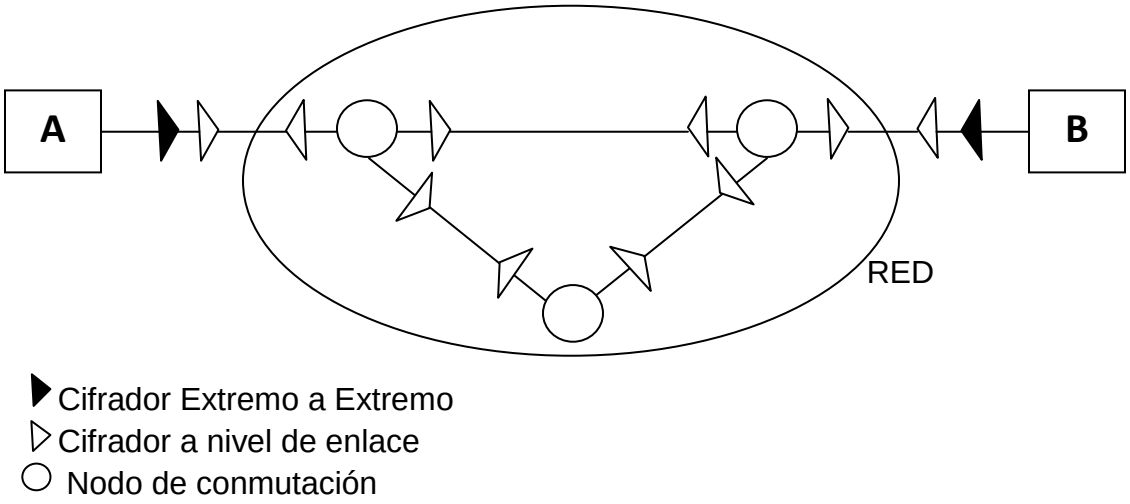
X= Mensaje en claro
Y= Mensaje Cifrado
K= Clave
$E_K[*]$= Operación de cifrado de mensajes con una clave K
^X= Clave Estimada
^Y= Mensaje Estimado

Cuando se realiza un cifrado extremo a extremo son los dos equipos terminales los encargados de ejecutar el proceso correspondiente. Solo se pueden cifrar los datos transmitidos; las cabeceras de los paquetes han de quedar sin modificar para que la red pueda encaminaros hasta su destino. En este caso los nodos son transparentes al cifrado y la confidencialidad del mensaje queda asegurada, aunque no ocurre lo mismo con la confidencialidad del tráfico. Cuando se cifra extremo a extremo se puede proporcionar cierto grado de autenticación que no es posible cuando se cifran en los enlaces; si un usuario es capaz de descifrar con su

clave simétrica la información que ha recibido puede estar seguro de que ha sido cifrada con la clave correcta. Para conseguir un buen grado de seguridad es conveniente realizar ambos tipos de cifrado.

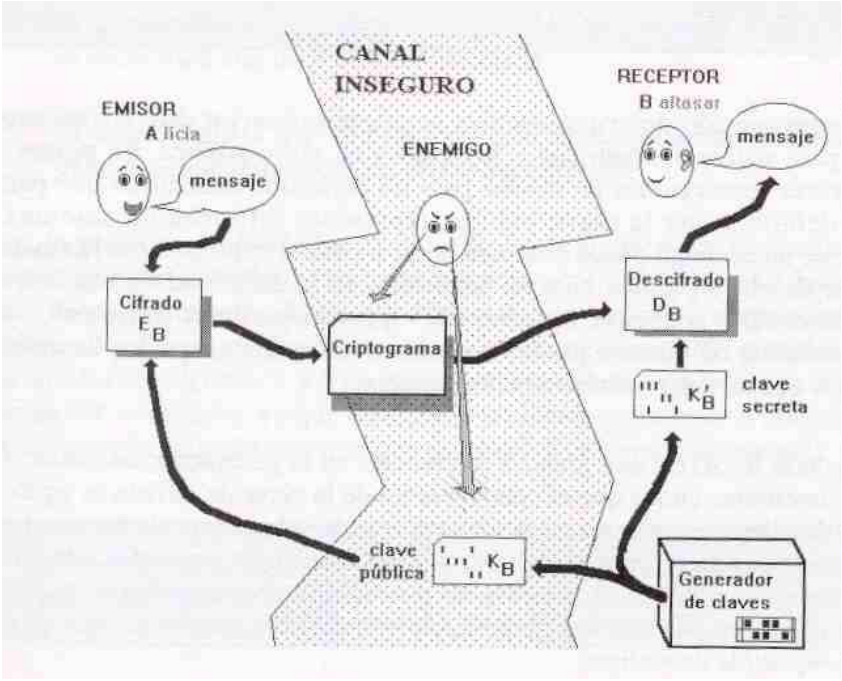
Uno de los problemas más importantes que presenta la utilización de cifradores simétricos en las redes de comunicaciones es la distribución de claves, pero a pesar de los inconvenientes de este tipo de cifrado que se han descrito, los algoritmos de cifrado en bloque y de cifrado en flujo son los únicos que permiten alcanzar altas velocidades al ser implementados con un programa de ordenador o en un circuito electrónico para usarlos en redes de comunicaciones.

Figura 16. Cifrado simétrico en una red de conmutación de paquetes.



Los criptosistemas de clave pública (Figura 16) resuelven de una forma cómoda algunos de los inconvenientes de los sistemas anteriores, principalmente autenticación y firma digital usando la clave privada de **A***;* pero, aunque proporciona confidencialidad a partir de la clave publica **B**, la baja velocidad obtenida con los algoritmos dificulta su utilización para el envió de mensajes cifrados de gran longitud. Si es útil, sin embargo, para el envió de claves de forma segura a los nodos de la red que emplean cifradores simétricos.

Figura 17. Esquema de criptosistema con clave pública



Fuente:http://www.monografias.com/trabajos76/utilidad-aritmetica-modular-criptografia/image011.gif [online]

La presentación de ambos tipos de cifrado queda reflejada en la tabla 55.

Tabla 55. Comparación entre cifrado simétrico y asimétrico.

CIFRADO SIMETRICO	CIFRADO CON CLAVE PUBLICA
• Confidencialidad • Cierto grado de autenticación • Sin firma digital • Alta velocidad	• Confidencialidad • Autenticación total • Firma Digital • Baja Velocidad

8.3. NORMATIVAS SOBRE SEGURIDAD

Las normativas sobre seguridad empezaron a desarrollarse a finales de los años setenta cuando surgió la necesidad de proteger ciertas comunicaciones que no pertenecían a los restringidos ambientes militares y diplomáticos, hasta entonces únicos usuarios de comunicaciones seguras. Bancos y multinacionales fueron los promotores y primeros beneficiarios de la normalización de los métodos de protección de la información y las comunicaciones. Dicha normalización, además de permitir una comunicación cifrada entre distintas entidades distintos equipos, incrementa el nivel de seguridad de los algoritmos al hacerlos públicos, lo que facilita su estudio y la posibilidad de analizarlos detalladamente.

Las organizaciones internacionales dedicadas a la elaboración de normativas han generado varios documentos relativos a los distintos aspectos de seguridad. Generalmente, en estas normas se tratan aspectos muy genéricos de la seguridad, como la situación de los elementos de seguridad en los distintos puntos de la red, sin entrar en detalles de algoritmos concretos. Así ocurre, por ejemplo, en la norma ISO 7498.2 y en distintas recomendaciones de las series X.2xx, X.4xx, X.5xx, X.7xx y X.8xx dadas por la Unión Internacional de Telecomunicaciones (ITU) y los trabajos realizados por el subcomité 27 (SC27) dependiente de la Organización Internacional de Estándares (ISO) y de la comisión Electrotécnica Internacional (IEC).

Por otra parte, el Instituto Nacional Americano de Estándares (ANSI), en sus series X3 Y X9, describe con detalle el cifrado, la integridad, la autenticación y la administración de claves. En la serie X12 está contemplada la seguridad para el intercambio de datos electrónicos (EDI). Muchas de las normas ANSI han sido posteriormente modificadas y adoptadas por ISO.

Existen, además, otros grupos y otras comisiones, como son el Instituto Europeo de Estándares de Telecomunicaciones (ETSI) o la Asociación Europea de Fabricantes de Ordenadores (ECMA), que están trabajando en cuestiones de normativa de seguridad, aunque sus resultados aún no son definitivos.

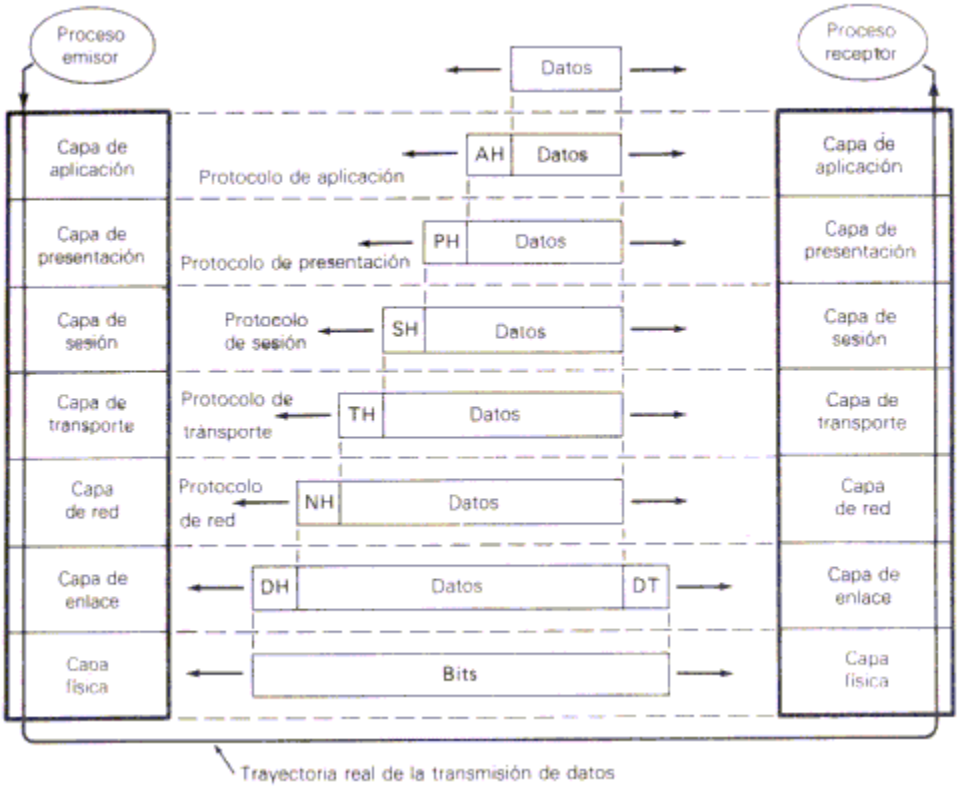
8.3.1. Organización de Estándares Internacionales (ISO).

Los servicios y mecanismos de seguridad están siendo incorporados al modelo de capas para la interconexión de sistemas abiertos (OSI), definido en la norma ISO

7498, según la norma ISO 7498.2, redactada en 1989. Sin embargo, la mayor parte de los trabajos han sido desarrollados para el nivel de aplicación. Ello indica que la seguridad es responsabilidad última del usuario, aunque la red pueda proporcionar algunos servicios adicionales.

Los niveles del modelo OSI se pueden agrupar en dos bloques (Figura 18): uno de ellos incluiría los niveles 1-3 (físico, enlace, red), los encargados de las comunicaciones entre redes, y en el otro grupo estarían los niveles 4-7 (transporte, sesión, presentación, aplicación), que relacionan los sistemas terminales a través de la red. Esta división resulta útil desde el punto de vista de la seguridad.

Figura 18. Niveles OSI y seguridad



Fuente: http://www.frm.utn.edu.ar/comunicaciones/images/osi8.gif [online]

Según ISO 7498.2, pueden existir servicios de confidencialidad en cualquiera de las tres capas **1-3**. En el nivel **1** se puede realizar el cifrado de todo el trafico del canal, evitando así que puedan distinguirse los bloques de información, los bloques de relleno y las direcciones de emisor y receptor; con ello se ofrece el servicio de confidencialidad en el trafico de mensajes. En la capa **2** pueden

cifrarse los campos de información de los bloques, al igual que en la capa **3**. En esta última pueden existir otros servicios de seguridad, como los servicios de gestión de acceso seguro, control de acceso, autenticación de entidades y servicio de integridad. Las comunicaciones seguras entre equipos terminales de datos a través d una red de paquetes conmutados harían uso de ellos.

Para que el usuario pueda acceder a la seguridad que ofrecen los niveles **1-3** se necesitan procedimientos de gestión de acceso que pasen parámetros "passwords" entre las capas **3** y **4**.

En el segundo grupo, que abarca las capas **4-7**, aparecen duplicados algunos de los servicios de seguridad ya completados en la capas **1-3**. Esta duplicación es solo aparente, ya que un servicio ofrecido por dos capas distintas no conlleva las mismas prestaciones. En el nivel de red, la seguridad se aplica a un tráfico multiplexado; es decir, no ofrece servicios de seguridad específicos para cada usuario ni un acceso restringido a la red. Además, la confirmación de recepción o reparto que envía la capa de red no indica el estado de los datos, que si queda reflejado en las confirmaciones procedentes de la capa de aplicación.

En la capa **5** no está completado ningún tipo de seguridad y en la capa **6** solamente confidencialidades. El servicio de no-repudio sólo está previsto en el nivel de aplicación.

8.3.2. Unión Internacional de Telecomunicaciones (ITU)

La ITU ha adoptado el modelo OSI con los servicios y protocolos definidos en las recomendaciones de la serie X.2xx. Referentes al tema de seguridad hay dos recomendaciones concretas cuya última versión data de mediados del año 1994: la recomendación X.273, en la que se describe el protocolo de seguridad para el nivel de red, y la recomendación X.274, donde se define el protocolo de seguridad para el nivel de transporte.

En el nivel de aplicación destacan las definiciones del sistema de mensajería que aparecen en la serie X.4xx y el directorio X.5xx. En ambos casos los documentos definen de forma amplia los procedimientos para proporcionar seguridad a distintos elementos y funciones, aunque solo se hace una breve mención sobre os algoritmos.

En la serie X.7xx, referente a la gestión del modelo OSI, aparecen algunas recomendaciones relativas a la seguridad como la X.736, la X.737, la X.740 y X.741. Aunque las recomendaciones X.736 y X.740 fueron redactadas en 1992, en 1995 sufrieron algunas modificaciones y se redactaron las otras dos. En ellas se tratan los temas de alarmas, tests, auditorias y atributos para control de acceso.

La recomendación X.800 data de 1991 y describe la arquitectura de seguridad para OSI. Durante 1995 se publicaron el resto de recomendaciones de la serie X.8xx, hasta un total de doce, y en ellas se tratan de forma genérica los distintos servicios de seguridad confidencialidad, integridad, control de acceso y autenticación en cada una de las capas del modelo OSI:

- **X.800:** Security architecture for Open Systems Interconnection for CCITT applications.
- **X.802:** Lower layers security model.
- **X.803:** Upper layers security model.
- **X.810:** Security frameworks for open systems: Overview.
- **X.811:** Security frameworks for open systems: Authentication frameworks.
- **X.812:** Security frameworks for open systems: Access control frameworks.
- **X.814:** Security frameworks for open systems: Confidentiality frameworks.
- **X.815:** Security frameworks for open systems: Integrity frameworks.
- **X.816:** Security frameworks for open systems: Security audit and alarms framework.
- **X.830:** Generic upper layers security: Overview, models and notation.
- **X.831:** Generic upper layers security: Security Exchange Service Element (SESE) service definition.
- **X.832:** Generic upper layers security: Security Exchange Service Element (SESE) Protocol specification.
- **X.833:** Generic upper layers security: Protecting transfer syntax specification.

8.3.3. Subcomité 27 (SC27)

Como se mencionó anteriormente, el SC27 realiza un trabajo de normativa sobre

la (seguridad genérica): es decir, sin referencias a algoritmos criptográficos concretos. En el SC27 existen tres grupos de trabajo (WG1, WG2 Y WG3), cada uno de los cuales está encargado de generar la normativa correspondiente sobre distintos aspectos de la seguridad.

8.3.3.1. WG1

Es el responsable de identificar las necesidades criptográficas, de definir de forma general los servicios de seguridad y de sugerir la línea maestra para el uso y gestión de sistema seguros relacionados con la tecnología de la información. Algunas de las normas generadas son:

- ISO/IEC 9979: *Procedures for the registration of cryptographic algorithms.*
- ISO/IEC DIS 11770-1: *Key management – Part 1: Key management framework.*
- ISO/IEC WD 14516-1 y 14516-2: *Guidelines on the use and management of Trusted Third-Party services.*

8.3.3.2. WG2

Se encuentra de la definición de técnicas y mecanismos necesarios para ofrecer los distintos servicios de seguridad en las redes de comunicaciones. Algunas de las normas redactadas por este son:

- ISO/IEC 9796: *Digital signature scheme giving message recovery.*
- ISO/IEC 9797: *Data integrity using a cryptographic check function employing a block cipher algorithm.*
- ISO/IEC 9798-1, 9798-2, 9798.3, 9798-4 Y 9798-5: *Entity authentication.*
- ISO/IEC CD 10118-1 Y 10118-2: *Hash functions.*
- ISO/IEC DIS 11770-1: *Key management – Part 1: Key management framework.*
- ISO/IEC DIS 1177O-2 y 11770-3: *Key management – Mechanisms using symmetric techniques.*

- ISO/IEC DIS 11770-2 y 11770-3: *Key management – mechanisms using asymmetric techniques.*
- ISO/IEC CD 13338-1, 13338-2 Y 13338-3: *Non-repudiation.*
- ISO/IEC DC 14888-1, 14888-2 Y 14888-3: *Digital signatures.*

8.3.3.3. WG3

Se dedica a establecer criterios de evaluación de la seguridad como los dados en:

- **ISO/IEC WD:** Evaluation criteria for Information Technology Systems.

La mayor parte de estas normas han aparecido durante 1994, 1995 y 1996, y está prevista una ampliación para muchas de ellas en los próximos años. El texto redactado coincide en muchos casos con el de las recomendaciones dadas por ITU.

8.4. PROTOCOLOS DE COMUNICACIÓN SEGURA

La Criptografía cubre hoy en día objetivos distintos, a veces muy alejados, del tradicional y más conocido de la transmisión secreta de información. Quizás la aplicación más antigua de la Criptografía sea precisamente la de establecer canales de comunicaciones seguros entre dos puntos. Desde un soldado galopando a través de territorio enemigo hasta un haz láser, pasando por un hilo telegráfico, el ser humano ha empleado infinidad de medios para poder enviar sus mensajes, cada uno de ellos con sus propias peculiaridades. Pero si hay una característica que se puede considerar común a todos los canales de comunicaciones, es la ausencia de control que se posee sobre el mismo desde el interlocutor **A** hasta el **B**. En general, se debe considerar que los mensajes son depositados en un medio ajeno al emisor y receptor, el cual es usualmente hostil, y que los medios que se apliquen para su protección deben ser válidos en los casos más desfavorables.

Un mensaje liberado en un medio hostil se enfrenta principalmente a dos peligros:

Acceso por agentes no autorizados: En un medio sobre el que no se puede ejercer ningún control, esta posibilidad debe tomarse muy en serio. Tanto que en lugar de suponer que el enemigo puede acceder al mensaje, se debe dar por

hecho que va a hacerlo. Por lo tanto, los sistemas de protección deben centrarse en garantizar que el mensaje resulte ininteligible al atacante.

Alteraciones en el mensaje: Este problema puede llegar a ser mucho peor que el anterior, ya que si se recibe un mensaje que ha sido modificado y se da por bueno, las consecuencias para la comunicación pueden ser catastróficas. En este sentido, las alteraciones pueden aplicarse tanto sobre el mensaje propiamente dicho, como sobre la información acerca de su verdadera procedencia.

La Criptografía, como ya se ha visto en anteriores capítulos, proporciona mecanismos fiables para evitar los dos peligros que se acaban de mencionar. En general, cada una de las aplicaciones concretas que necesiten de estas técnicas poseerá unas características específicas, por lo que en cada caso habrá una combinación de algoritmos criptográficos que permitirá proporcionar al sistema el nivel de seguridad necesario.

Estas combinaciones de algoritmos se estructurarán finalmente en forma de protocolos, para proporcionar métodos de comunicación segura normalizados.

Protocolo Criptográfico es un protocolo (es decir un conjunto bien definido de etapas, implicando a dos o más partes y acordado por ellas, designado para realizar una tarea específica) que utiliza como herramienta algún algoritmo criptográfico.

Existe una amplia variedad de protocolos criptográficos, que dan respuesta a diferentes objetivos. Se trata de un tema muy amplio y en rápido crecimiento. A continuación se citan algunos de estos tipos de protocolos, mas sin embargo no se desarrollan en profundidad:

1. **Protocolos de Autentificación de Usuario:** Permiten garantizar que el remitente de un mensaje o el usuario con el que se establece comunicación es realmente quien pretende ser.
2. **Protocolos de Autentificación del Mensaje:** Garantizan que el mensaje enviado no ha sido substituido por otro ni alterado (integridad del mensaje).
3. **Distribución de claves:** Un problema importante en Criptógrafa de clave privada es el de la creación y transporte de las claves a utilizar por cada par de usuarios. En cuanto a las claves de un sistema de clave pública; la problemática de su distribución es distinta (no es necesario el secreto), demandando protocolos específicos.

4. **Protocolos para Compartir Secretos:** Su objetivo es distribuir un cierto secreto (por ejemplo la clave para abrir una caja fuerte), entre un conjunto P de participantes, de forma que ciertos subconjuntos prefijados de P puedan, uniendo sus participaciones, recuperar dicho secreto.
5. **Pruebas de Conocimiento Cero:** Permiten a un individuo convencer a otro de que posee una cierta información, sin revelarle nada sobre el contenido de la misma.
6. **Transacciones Electrónicas Seguras:** Permiten realizar de manera electrónica segura las operaciones bancarias habituales, firma electrónica de contratos, etc.
7. **Compromiso de bit:** Permiten a una parte **A** comprometerse con una elección (un bit o más generalmente una serie de bits) sin revelar tal elección hasta un momento posterior. El protocolo garantiza a otra parte **B** que **A** no cambia su elección.
8. **Transferencias Transcordadas:** Permiten a una parte **A** enviar a otra **B** un mensaje o secreto entre dos posibles. **A** no conoce cuál de los dos ha recibido realmente **B**.
9. **Elecciones Electrónicas:** Permiten realizar un proceso electoral electrónicamente, garantizando la deseable privacidad de cada votante y la imposibilidad de fraude.
10. **Jugar al Poker por Internet:** Posibilita a dos personas, físicamente separadas, mantener una partida de Poker (o similar: cara o cruz, chinos, etc), comunicándose por correo electrónico, teléfono, etc, garantizando la imposibilidad de hacer trampa.

Sin embargo no se puede dejar de lado el conjunto de protocolos más general, grande, conocido y de gran impacto en todas las redes de comunicaciones, como lo son los protocolos TCP/IP, SSL, TSL y el IPsec, sin olvidar que más atrás ya se mencionó las directrices del modelo OSI.

8.4.1. Protocolos TCP/IP

El conjunto básico de protocolos sobre los que se construye la red Internet se conoce popularmente como TCP/IP, agrupación de los nombres de dos de los elementos más importantes, pero no los únicos, de la familia: TCP (Transmission Control Protocol) e IP (Internet Protocol).

El modelo de comunicaciones sobre el que se basa Internet se estructura en forma de capas apiladas, de manera que cada una de ellas se comunica con las capas inmediatamente superior e inferior, logrando diversos niveles de abstracción, que permiten intercambiar información de forma transparente entre ordenadores. La consecuencia más importante de este enfoque es que dos dispositivos cualesquiera, que pueden estar conectados a Internet por medios totalmente distintos, fibra óptica, cable de cobre, láser, ondas electromagnéticas, entre otros, y separados por multitud de enlaces diferentes, satélite, cables submarinos, redes inalámbricas, etc; pueden conectarse entre ellos simplemente con que dispongan de una implementación de TCP/IP.

A diferencia del modelo OSI, que consta de siete capas, denominadas aplicación, presentación, sesión, transporte, red, enlace y física, los protocolos TCP/IP se organizan únicamente en cinco (Figura 20). Aunque la correspondencia no es exacta, se puede decir que básicamente los tres niveles superiores del modelo OSI se agrupan en el nivel de aplicación de TCP/IP. Se comenta brevemente cada uno de ellos:

Capa Física: Describe las características físicas de la comunicación, como son el medio empleado, los voltajes necesarios, la modulación empleada, etc.

Capa de Enlace: Indica cómo los paquetes de información viajan a través del medio físico, indicando qué campos de bits se añaden a éstos para que puedan ser reconocidos satisfactoriamente en destino. Ejemplos de protocolos de enlace: Ethernet, 802.11 WiFi, Token Ring, etc.

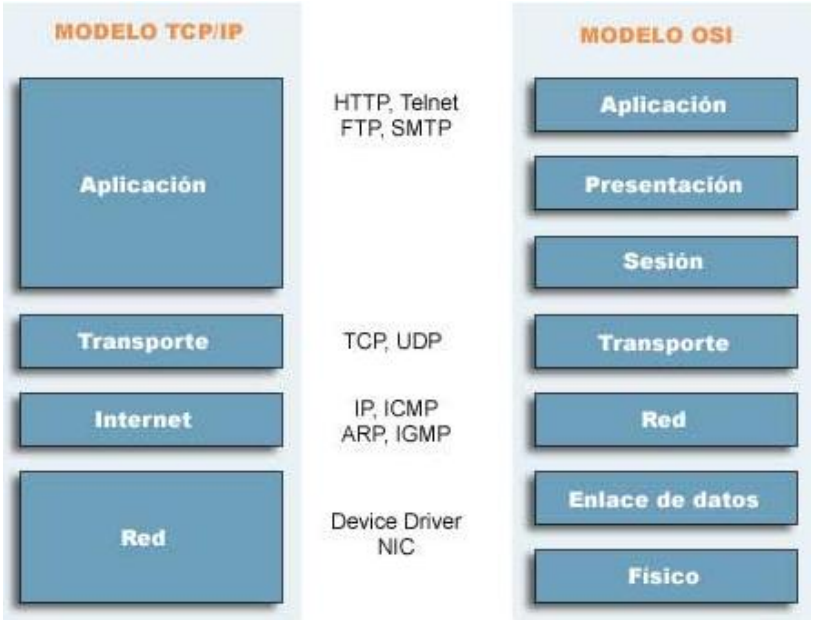
Capa de Red: En ella se ubica el protocolo IP, cuyo propósito consiste en hacer llegar los paquetes a su destino a través de una única red. Existen algunos protocolos de mayor nivel, como ICMP o IGMP, que aunque se construyen sobre IP, también pertenecen a la capa de red, a diferencia de lo que ocurre en el modelo OSI.

Capa de Transporte: Su propósito es garantizar que los paquetes han llegado a su destino, y en el orden correcto. El protocolo más importante en este nivel es TCP, pero existen otros como UDP, DCCP o RTP.

Capa de Aplicación: Esta es la capa a la que acceden de forma directa la mayoría de las aplicaciones que usan Internet. En ella se reciben los datos, que son pasados a las capas inferiores para que sean enviados a su destino. A este

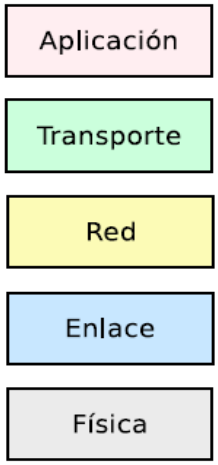
nivel pertenecen protocolos tales como HTTP, FTP, SSH, HTTPS, IMAP, DNS, SMTP, IRC, etc.

Figura 19. Correspondencia capas TCP/IP y OSI



Fuente: http://4.bp.blogspot.com/_s3LSKwWfOjs/SxAqxrD_2OI/AAAAAAAAAHs/gFZ5wvIZXSk/s1600/tcp_ip_osi.jpg [online]

Figura 20. Esquema de protocolos del modelo TCP/IP



Fuente: http://gridra.files.wordpress.com/2008/09/tcp-ip3.jpg

En la práctica, se puede encontrar protocolos encaminados a obtener comunicaciones seguras en prácticamente todos los niveles de este esquema. Los distintos protocolos de comunicación segura pueden ser utilizados para construir las denominadas redes privadas virtuales. Una red privada virtual, en inglés VPN (Virtual Private Network) es una red de comunicaciones privada construida sobre una red pública. Hacia los usuarios se comporta como si se tratara de una red interna, ofreciendo acceso únicamente a aquellos que estén autorizados, y resultando inaccesible para los demás, cuando en realidad todas las conexiones se realizan a través de Internet.

8.4.2. Protocolo SSL

El protocolo SSL (Secure Sockets Layer), desarrollado originalmente por la empresa Netscape, permite establecer conexiones seguras a través de Internet, de forma sencilla y transparente. Se sitúa en la capa de aplicación (Figura 20), directamente sobre el protocolo TCP, y aunque puede proporcionar seguridad a cualquier aplicación que corra sobre TCP, se usa principalmente para proporcionar seguridad a los protocolos HTTP (web), SMTP (email) y NNTP (news), dando lugar en el primero de los casos a los servidores web seguros, cuya URL comienza por el prefijo https://. Su fundamento consiste en interponer una fase de codificación de los mensajes antes de enviarlos a través de la red. Una vez que se ha establecido la comunicación, cuando una aplicación quiere enviar información a otra computadora, la capa SSL la recoge y la codifica, para luego enviarla a su destino a través de la red. Análogamente, el módulo SSL del otro ordenador se encarga de decodificar los mensajes y se los pasa como texto claro a la aplicación destinataria.

SSL también incorpora un mecanismo de autentificación que permite garantizar la identidad de los interlocutores. Típicamente, ya que este protocolo se diseñó originalmente para establecer comunicaciones web, el único que suele autentificarse es el servidor, aunque también puede realizarse una autentificación mutua.

Una comunicación a través de SSL implica tres fases fundamentalmente:

- ✓ Establecimiento de la conexión y negociación de los algoritmos criptográficos que van a usarse en la comunicación, a partir del conjunto de algoritmos soportados por cada uno de los interlocutores.

- ✓ Intercambio de claves, empleando algún mecanismo de clave pública, y autentificación de los interlocutores a partir de sus certificados digitales (Se explica más adelante).
- ✓ Cifrado simétrico del tráfico.

Una de las ventajas de emplear un protocolo de comunicaciones en lugar de un algoritmo o algoritmos concretos, es que ninguna de las fases del protocolo queda atada a ningún algoritmo, por lo que si en el futuro aparecen algoritmos mejores, o alguno de los que se emplean en un momento dado quedara comprometido, el cambio se puede hacer sin modificar el protocolo. En la actualidad, las implementaciones típicas de SSL soportan algoritmos como RSA, Diffie-Hellman o DSA para la parte asimétrica; RC2, RC4, IDEA, DES, Triple DES o AES para la simétrica (DES, AES y RSA capítulo 6), y como funciones resumen.

Las ventajas de SSL son evidentes, ya que liberan a las aplicaciones de llevar a cabo las operaciones criptográficas antes de enviar la información, y su transparencia permite usarlo de manera inmediata sin modificar apenas los programas ya existentes.

Desde hace tiempo los principales navegadores de Internet incorporan un módulo SSL, que se activa de forma automática cuando es necesario. Hasta diciembre de 1999, debido a las restricciones de exportación de material criptográfico existentes en los EE.UU, la mayoría de los navegadores incorporaban un nivel de seguridad bastante pobre (claves simétricas de 40 bits), por lo que conviene comprobar qué nivel de seguridad se está empleando cada vez que se haga una conexión.

Existen implementaciones de SSL que permiten construir los denominados túneles SSL, que permiten dirigir cualquier conexión a un puerto TCP a través de una conexión SSL previa, de forma transparente para las aplicaciones que se conectan.

8.4.3. Protocolo TLS

TLS (descrito en el documento RFC 2246, Disponible en internet: <http://www.ietf.org/rfc/rfc2246.txt>) es un protocolo basado en la versión 3.0 de SSL, si bien con una serie de mejoras que lo hacen incompatible con este último.

Una de las ventajas que proporciona sobre SSL es que puede ser iniciado a partir de una conexión TCP ya existente, lo cual permite seguir trabajando con los mismos puertos que los protocolos no cifrados. Mientras que SSL es un protocolo incompatible con TCP, lo cual significa que no podemos establecer una conexión de un cliente TCP a un servidor SSL ni al revés, y por tanto es necesario diferenciarlos utilizando distintos números de puerto (80 para un servidor web normal y 443 para un servidor web sobre SSL), con TLS puede establecerse la conexión normalmente a través de TCP y el puerto 80, y luego activar sobre el mismo el protocolo TLS.

En este protocolo se emplea una serie de medidas de seguridad adicionales, encaminadas a protegerlo de distintos tipos de ataque, en especial de los de intermediario:

- ✓ Uso de funciones **MAC** (Massage Authentication Codes) en lugar de funciones **MDC** (Modification Detection Codes) ó **MAC** (Massage Authentication Codes) únicamente.
- ✓ Numeración secuencial de todos los campos que componen la comunicación, e incorporación de esta información al cálculo de los MAC.
- ✓ Protección frente a ataques que intentan forzar el empleo de versiones antiguas menos seguras del protocolo o cifrados más débiles.
- ✓ El mensaje que finaliza la fase de establecimiento de la conexión incorpora una signatura (hash) de todos los datos intercambiados por ambos interlocutores.

Si bien el método usado con más frecuencia para establecer conexiones seguras a través de Internet sigue siendo SSL, cabe esperar que con el tiempo sea paulatinamente reemplazado por TLS, y que este último se convierta en el estándar de seguridad para las comunicaciones cifradas en Internet.

8.4.4. Protocolos IPsec

IPsec es un estándar que proporciona cifrado y autentificación a los paquetes IP, trabajando en la capa de red. En lugar de tratarse de un único protocolo, IPsec es en realidad un conjunto de protocolos, definidos en diversos RFCs (principalmente en el 2401), encaminados a proporcionar autentificación, confidencialidad e integridad a las comunicaciones IP. Su carácter obligatorio dentro del estándar IPv6 (recuérdese que en IPv4, la versión más empleada en la actualidad de este

protocolo, es opcional) hará con seguridad que la popularidad de IPsec crezca al mismo ritmo que la implantación de la nueva versión del protocolo IP.

IPsec puede ser utilizado para proteger una o más rutas entre un par de ordenadores, un par de pasarelas de seguridad, ordenadores que hacen de intermediarios entre otros, y que implementan los protocolos IPsec.

En función del tipo de ruta que se proteja, se distinguen dos modos de operación:

Modo túnel: Se realiza entre dos pasarelas de seguridad, de forma que éstas se encargan de crear una ruta segura entre dos ordenadores conectados a ellas, a través de la cual viajan los paquetes. De este modo se puede disponer dentro de una red local de un ordenador que desempeñe las labores de pasarela, al que las computadoras de la propia red envíen los paquetes, para que éste les aplique los protocolos IPsec antes de remitirlos al destinatario o a su pasarela de seguridad asociada. Este modo permite interconectar de forma segura ordenadores que no incorporen IPsec, con la única condición de que existan pasarelas de seguridad en las redes locales de cada uno de ellos.

Modo transporte: En este caso los cálculos criptográficos relativos a los protocolos IPsec se realizan en cada extremo de la comunicación.
Básicamente, IPsec se compone a su vez de dos protocolos, cada uno de los cuales añade una serie de campos, o modifica los ya existentes, a los paquetes IP: Cabecera de autentificación IP, abreviado como AH (IP Authenticacion Header), diseñado para proporcionar integridad, autentificación del origen de los paquetes, y un mecanismo opcional para evitar ataques por repetición de paquetes.

Protocolo de encapsulamiento de carga de seguridad, o ESP (Encapsulating Security Payload) que, además de proveer integridad, autentificación y protección contra repeticiones, permite cifrar el contenido de los paquetes.

Debido a que algunos de los servicios que IPsec proporciona necesitan de la distribución e intercambio de las claves necesarias para cifrar, autentificar y verificar la integridad de los paquetes, es necesario que éste trabaje en consonancia con un conjunto externo de mecanismos que permita llevar a cabo esta tarea, tales como los protocolos IKE, SKIP o Kerberos.

8.5. CERTIFICADOS Y FIRMAS DIGITALES

Cuando se establece una comunicación de cualquier tipo es necesario poder asegurar que los mensajes no han sufrido alteraciones, es decir, que la información recibida coincide exactamente con la enviada. En muchos casos, existe el requerimiento adicional de conocer la identidad del emisor, sea éste una persona o algún tipo de dispositivo, para evitar que sea suplantado por un impostor. Se denomina en general autentificación (o autenticación) a las operaciones consistentes en verificar tanto la identidad del emisor como la integridad de los mensajes que de él se reciben.

Independientemente de que la operación de autentificación se lleve a cabo sobre el contenido de una comunicación o sobre los propios interlocutores, ésta puede realizarse en el mismo momento, de forma interactiva, como cuando se introduce una contraseña para acceder a un sistema, o dejarse pospuesta para ser realizada posteriormente fuera de línea, como cuando se firma digitalmente un mensaje, en cuyo caso la firma puede ser verificada tantas veces como se desee, una vez finalizada la comunicación.

Por autentificación se entiende cualquier método que permita comprobar de manera segura alguna característica sobre un objeto. Dicha característica puede ser su origen, su integridad, su identidad, etc. Se considera tres grandes tipos dentro de los métodos de autentificación:

Autentificación de mensaje: Se quiere garantizar la procedencia de un mensaje conocido, de forma que se pueda asegurar de que no es una falsificación. Este mecanismo se conoce habitualmente como firma digital.
Autentificación de usuario mediante contraseña: En este caso se trata de garantizar la presencia de un usuario legal en el sistema. El usuario deberá poseer una contraseña secreta que le permita identificarse.

Autentificación de dispositivo: Se trata de garantizar la presencia de un dispositivo valido. Este dispositivo puede estar solo o tratarse de una llave electrónica que sustituye a la contraseña para identificar a un usuario.

Se puede notar que la autentificación de usuario por medio de alguna característica biométrica, como pueden ser las huellas digitales, la retina, el iris, la voz, etc. puede reducirse a un problema de autentificación de dispositivo, solo que el dispositivo en este caso es el propio usuario.

8.5.1. Funciones Hash

Una herramienta fundamental en la criptografía, son las funciones hash, son usadas principalmente para resolver el problema de la integridad de los mensajes, así como la autenticidad de mensajes y de su origen. Una función hash es también ampliamente usada para la firma digital, ya que los documentos a firmar son en general demasiado grandes, la función hash les asocia una cadena de longitud 160 bits por ejemplo, que los hace más manejables para el propósito de firma digital.

La función hash efectúa básicamente lo siguiente: un mensaje de longitud arbitraria lo transforma de forma "única" a un mensaje de longitud constante.

¿Cómo hace esto? La idea general es la siguiente:
La función hash toma como entrada una cadena de longitud arbitraria, dígase 5259 bits, luego divide éste mensaje en pedazos iguales, puede ser de 160 bits, como en este caso y en general el mensaje original no será un múltiplo de 160, entonces para completar un número entero de pedazos de 160 bits, al último se le agrega un relleno, de puros ceros por ejemplo. En este caso en 5259 caben 32 pedazos de 160 bits y sobran 139, entonces se agregarán 21 ceros más.

El mensaje toma la forma $X = X_1, X_2, X_3,\ldots,X_t$ donde cada X_i tiene igual longitud (160 bits para este caso).

Posteriormente se asocia un valor constante a un vector inicial IV y (H_0=IV).

Ahora se obtiene H_1 que es el resultado de combinar H_0 con X_1 usando una función de compresión f:

- ✓ $H_1 = f(H_0,X_1)$

Posteriormente se obtiene H_2, combinando H_1 y X_2 con f:

- ✓ $H_2 = f(H_1,X_2)$

Se hace lo mismo para obtener H_3:

- ✓ $H_3 = f(H_2,X_3)$

Hasta llegar a H_t:

- ✓ $H_t = f(H_{t-1}, X_t)$

Entonces el valor hash será $h(M) = H_t$

Por medio de algún método determinado, (no se profundiza en ellos), lo que se hace es tomar el mensaje partirlo en pedazos de longitud constante y combinar con este método pedazo por pedazo hasta obtener un mensaje único de longitud fija.

Las funciones hash (o primitivas hash) pueden operar como: **MDC** (Modification Detection Codes) ó **MAC** (Massage Authentication Codes).

Los MDC sirven para resolver el problema de la integridad de la información, al mensaje se le aplica un **MDC** (una función hash) y se manda junto con el propio mensaje, al recibirlo el receptor aplica la función hash al mensaje y comprueba que sea igual al hash que se envió antes.

Es decir, se aplica un hash al mensaje **M** y se envía con el mensaje (**M**, *h*(**M**)), cuando se recibe se le aplica una vez más el hash (ya que **M** es público) obteniendo *h*'(**M**), si *h*(**M**)=*h*'(**M**), entonces se acepta que el mensaje sea transmitido sin alteración.

Los MAC sirven para autenticar el origen de los mensajes (junto con la integridad), un MAC. Es decir, se combina el mensaje **M** con una clave privada K y se les aplica un hash *h*(**M**,K), si al llegar a su destino *h*(**M**, K) se comprueba de integridad de la clave privada K, entonces se demuestra que el origen es solo el que tiene la misma clave K, probando así la autenticidad del origen del mensaje.

Las propiedades que deben de tener las primitivas hash son:

1. Resistencia a la preimagen, significa que dada cualquier imagen, es computacionalmente imposible encontrar un mensaje *x* tal que *h*(*x*)=*y*. Otra forma como se conoce esta propiedad es que *h* sea de un solo sentido.

2. Resistencia a una segunda preimagen, significa que dado *x*, es computacionalmente imposible encontrar una *x*' tal que *h*(*x*)=*h*(*x*'). Otra

forma de conocer esta propiedad es que *h* sea resistente a una colisión suave.

3. Resistencia a colisión significa que es computacionalmente imposible encontrar dos mensajes diferentes *x*, *x*' tal que *h*(*x*)=*h*(*x*'). Esta propiedad también se conoce como resistencia a colisión fuerte.

Para ilustrar la necesidad de estas propiedades se muestra los siguientes ejemplos:

Considérese un esquema de firma digital con apéndice, entonces la firma se aplica a *h*(*x*), en este caso *h* debe ser un **MDC** con resistencia a una 2° preimagen, ya que de lo contrario un atacante C que conozca la firma sobre *h*(*x*), puede encontrar otro mensaje *x*' tal que *h*(*x*) = *h*(*x*') y reclamar que la firma es del documento *x*'.

Si el atacante C puede hacer que el usuario firme un mensaje, entonces el atacante puede encontrar una colisión (*x*, *x*') (en lugar de lo más difícil que es encontrar una segunda preimagen de *x*) y hacer firmar al usuario a *x* diciendo que firmo *x*'. En este caso es necesaria la propiedad de resistencia a colisión.

Por último si (*e*,*n*) es la clave pública **RSA** de A, C puede elegir aleatoriamente un "***y***" y calcular $z = y^e \bmod n$, y reclamar que "***y***" es la firma de *z*, si C puede encontrar una preimagen *x* tal que *z* = *h*(*x*), donde *x* es importante para A. Esto es evitable si *h* es resistente a preimagen.

Las funciones hash más conocidas son las siguientes: las que se crean a partir de un block cipher (Método de cifrado en bloque) como DES, MD5, SHA-1 y RIPEMD 160.

Actualmente se ha podido encontrar debilidades en las funciones hash que tienen como salida una cadena de 128 bits, por lo que se ha recomendado usar salidas de 160 bits. Así mismo se han encontrado ataques a MD5 y SHA-0 (antecesora de SHA-1), esto ha dado lugar que se dirija la atención sobre la función has RIPEMD-160.

El ataque más conocido (a fuerza bruta) a una función hash es conocido como "birthday attack" y se basa en la siguiente paradoja, si hay 23 personas en un local existe una probabilidad de al menos 1/2, de que existan dos personas con el

mismo cumpleaños. Aunque parezca muy difícil esa posibilidad se puede mostrar que en general al recorrer la raíz cuadrada del número de un conjunto de datos, se tiene la probabilidad de al menos ½ de encontrar dos iguales.

Al aplicar esto a una función hash, es necesario recorrer entonces la raíz cuadrada de 2160 mensajes para poder encontrar dos con el mismo hash, o sea encontrar una colisión. Por lo tanto una función hash con salida 2160 tiene una complejidad de 280, y una función de 128 bits de salida tiene una complejidad de 264, por lo que es recomendable usar actualmente salida de 160 bits (48 dígitos).

La criptografía simétrica, es claramente insuficiente para llevar a cabo comunicaciones seguras a través de canales inseguros (Internet), debido a que los dos interlocutores necesitan compartir una clave secreta llamada "clave de sesión". Dicha clave debe ser transmitida en algún momento desde un extremo a otro del canal de comunicación de forma segura, ya que de ella depende la protección de toda la información que se transmita a lo largo de esa sesión en particular. Se necesita, pues, un canal seguro para poder crear otro canal seguro.

La criptografía asimétrica ofrece una salida al problema, proporcionando ese canal seguro de comunicación que va a permitir a los participantes intercambiar las claves de sesión. Y ésa no es la única ventaja, ya que los algoritmos asimétricos ofrecen mecanismos fiables para que ambos interlocutores se puedan identificar frente al otro de manera segura. La razón por la que no se emplean algoritmos asimétricos todo el tiempo es porque, entre otras ventajas, los criptosistemas simétricos resultan mucho más eficaces y rápidos.

8.5.2. Firmas Digitales

Una firma digital es una secuencia de bits que se añade a una pieza de información cualquiera, y que permite garantizar su autenticidad de forma independiente del proceso de transmisión, tantas veces como se desee. Presenta una analogía directa con la firma manuscrita, y para que sea equiparable a esta última debe cumplir las siguientes propiedades:
Va ligada indisolublemente al mensaje: Una firma digital válida para un documento no puede ser válida para otro distinto.

Sólo puede ser generada por su legítimo titular: Al igual que cada persona tiene una forma diferente de escribir, y que la escritura de dos personas diferentes

puede ser distinguida mediante análisis caligráficos, una firma digital sólo puede ser construida por la persona o personas a quienes legalmente corresponde.

Es públicamente verificable: Cualquiera puede comprobar su autenticidad en cualquier momento, de forma sencilla.

La forma más extendida de calcular firmas digitales consiste en emplear una combinación de cifrado asimétrico y funciones resumen. El esquema de funcionamiento queda ilustrado en la Figura 21.

Se sabe que un mensaje **m** puede ser autentificado codificando con la llave privada $\mathbf{K_p}$ el resultado de aplicarle una función resumen, EK_p (r(m)). Esa información adicional (que se denomina firma o signatura del mensaje **m**) solo puede ser generada por el poseedor de la clave privada $\mathbf{K_p}$. Cualquiera que tenga la llave publica correspondiente está en condiciones de decodificar y verificar la firma. Para que sea segura, la función resumen r(x) debe cumplir ciertas características:

- ✓ r(m) es de longitud fija, independientemente de la longitud de m.
- ✓ Dado m, es fácil calcular r(m).
- ✓ Dado r(m), es computacionalmente intratable recuperar m.
- ✓ Dado m, es computacionalmente intratable obtener un m' tal que r(m) = r(m').

8.5.3. Longitud Adecuada para una signatura

Para decidir cuál debe ser la longitud apropiada de una signatura, vease primero el siguiente ejemplo: ¿Cual es la cantidad **n** de personas que hay que poner en una habitación para que la probabilidad **P** de que el cumpleaños de una de ellas sea el mismo día que el mío supere el 50%? Se sabe que cuando **n** = 1, **P** = 1/365. Cuando **n** = 2, la probabilidad de que ningún cumpleaños coincida con el mío es el producto de la probabilidad de que no coincida el primero, por la probabilidad de que no coincida el segundo, luego:

$$\mathbf{P} = 1 - (364/365) \cdot (364/365)$$

En el caso general,

$$\mathbf{P} = 1 - (364/365)^n$$

Para que **P** > 0,5, **n** debe ser al menos igual a 253. Sin embargo, ¿cuál sería la cantidad de gente necesaria para que la probabilidad **Q** de que dos personas cualesquiera tengan el mismo cumpleaños supere el 50%? Las dos primeras personas (o sea, cuando **n** = 2) tienen una probabilidad (364/365) de no compartir el cumpleaños; una tercera, suponiendo que las dos primeras no lo comparten, tiene una probabilidad (363/365) de no compartirlo con las otras dos, por lo que se tiene ((364·363) / (365·365)), y así sucesivamente.

En el caso general queda:

$$Q = 1 – (364 \cdot 363 \ldots (365 - n + 1) / (365^{n)} \text{ con } n \geq 2$$

Si se hacen los cálculos, se verá que **Q** > 0,5 si **n** > 22, una cantidad sorprendentemente mucho menor que 253.
La consecuencia de este ejemplo, conocido como la paradoja del cumpleaños, es que aunque resulte muy difícil dado un **m** calcular un **m'** tal que r(**m**) = r(**m'**), es considerablemente menos costoso generar muchos valores aleatoriamente, y posteriormente buscar entre ellos una pareja cualquiera (**m**, **m'**), tal que r(**m**) = r(**m'**).

En el caso de una firma de 64 bits, se necesita 2^{64} mensajes dado un **m** para obtener el **m'**, pero bastara con generar aproximadamente 2^{32} mensajes aleatorios para que aparezcan dos con la misma signatura, en general, si la primera cantidad es muy grande, la segunda cantidad es aproximadamente su raíz cuadrada. El primer ataque llevaría 600.000 años con una computadora que generara un millón de mensajes por segundo, mientras que el segundo necesitaría apenas una hora.

Se debe añadir a la lista de condiciones sobre las funciones resumen lo siguiente:

- ✓ Debe ser difícil encontrar dos mensajes aleatorios, **m** y **m'**, tales que r(**m**) = r(**m'**).

Hoy por hoy se recomienda emplear signaturas de al menos 128 bits, siendo 160 bits el valor más usado.

8.5.4. Certificados Digitales

Un certificado digital es esencialmente una clave pública y un identificador, firmados digitalmente por una autoridad de certificación, y su utilidad es demostrar que una clave pública pertenece a un usuario concreto. Evidentemente, la citada autoridad de certificación debe encargarse de verificar previamente que la clave pública es auténtica. En España, por ejemplo, la Fábrica Nacional de Moneda y Timbre actúa como autoridad certificadora, firmando las claves públicas de los ciudadanos y generando los certificados digitales correspondientes. Cualquier entidad que disponga de la clave pública de la FNMT estará en condiciones de verificar sus certificados digitales, otorgando la confianza correspondiente a las claves públicas asociadas a los mismos.

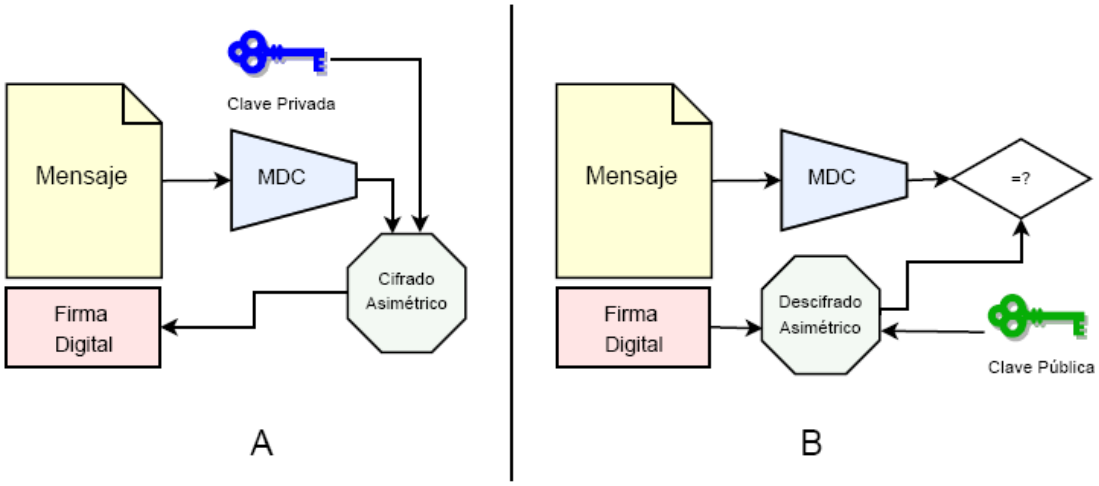
8.5.5. Certificados X.509

El formato de certificados X.509 (Recomendación X.509 de CCITT:“The Directory - Autentication Framework”. 1988) es uno de los más comunes y extendidos en la actualidad.

El estándar X.509 sólo define la sintaxis de los certificados, por lo que no está atado a ningún algoritmo en particular, y contempla los siguientes campos:

- ✓ Versión.
- ✓ Número de serie.
- ✓ Identificador del algoritmo empleado para la firma digital.
- ✓ Nombre del certificador.
- ✓ Periodo de validez.
- ✓ Nombre del sujeto.
- ✓ Clave pública del sujeto.
- ✓ Identificador único del certificador.
- ✓ Identificador único del sujeto.
- ✓ Extensiones.
- ✓ Firma digital de todo lo anterior generada por el certificador.

Figura 21. Esquema de firma digital



Fuente: http://www.rioja2.com/diario/medios/img/2010/07/firma02_33009.jpg

8.5.6. Certificados de revocación

Cuando una clave pública pierde su validez por destrucción o robo de la clave privada correspondiente por ejemplo, es necesario anularla. Para ello se emplean los denominados certificados de revocación que no son más que un mensaje que identifica a la clave pública que se desea anular, firmada por la clave privada correspondiente.

De esta forma se garantiza que una clave pública únicamente puede ser revocada por su legítimo propietario si la clave privada resulta comprometida, al atacante no le interesará revocarla, ya que entonces el material robado perdería su valor. Como puede verse, para revocar una clave pública es necesario estar en posesión de la privada, por lo que si se pierde esta última, jamás se podrá hacer la revocación.

Para evitar estos problemas, conviene seguir una serie de pautas:

- ✓ Generar los pares de claves con un período limitado de validez. De esta forma, si no se puede revocar una clave, expirará por sí misma.
- ✓ Generar el certificado de revocación junto con el propio par de claves, y almacenarlo en lugar seguro.

- ✓ Algunos protocolos permiten nombrar revocadores para la clave pública, que podrán generar un certificado de revocación empleando únicamente sus propias claves privadas.

Si una clave ha sido anulada por alguna causa, las autoridades que la hubieran certificado deben cancelar todos sus certificados asociados. Esto hace que todas las autoridades dispongan de listas de revocación de certificados (CRL), que actualizan periódicamente, además de un servicio de consulta de las mismas.

Dicho servicio suele permitir tanto la consulta de la validez de un certificado concreto, como la descarga total o parcial de la CRL correspondiente.

8.6. SEGURIDAD EN EL CORREO ELECTRONICO

El correo electrónico es quizá la aplicación distribuida de mayor difusión y con unas perspectivas de uso en claro aumento, que hacen necesario el desarrollo de servicios de aumentación y confidencialidad. Hasta ahora han aparecido varios esquemas que intentan dar solución a esta demanda, pero hay dos de ellos que han tenido una relación especial y su uso está muy difundido: Prettry Good Privacy (**PGP**) y Privacy Enhanced Mail (**PEM**).

8.6.1. Prettry Good Privacy (PGP)

Ha sido diseñado por Phil Zimmermann para proporcionar autentificación y confidencialidad tanto en el envió de mensajes por correo electrónico como en la protección de datos almacenados. Existe una versión de libre uso, disponible en internet, que incluye el código fuente y que puede correr en multitud de plataformas con distintos sistemas operativos.

Utiliza los algoritmos **RSA** y MD5 para autenticar los mensajes, y **RSA** e IDEA para proporcionarles confidencialidad. Se va a ver brevemente como se realizan estas funciones suponiendo que un emisor **A** quiere enviar un mensaje de forma segura, utilizando PGP, a un receptor **B**.

Para autenticar un mensaje que **A** envía a **B**, se genera un código hash de 128 bits de ese mensaje mediante el MD5. El código hash se cifra con **RSA**, utilizando la clave privada de **A**, y se añade al mensaje. El receptor **B** descifra el mensaje y

recupera el código hash con la clave publica de **A** utilizando **RSA**. Hecho esto, **B** genera el código hash a partir del mensaje y lo compara con el código hash descifrado. Si coinciden, el mensaje es autentico.

Para proporcionar confidencialidad a un mensaje transmitido o almacenado. **A** genera, en primer lugar, un mensaje y una clave de sesión aleatoria de 128 bits. El mensaje se cifra con esta clave mediante el algoritmo IDEA. Posteriormente la clave de sesión se cifra con **RSA** utilizando la clave pública de **B** y se añade al mensaje. **B** descifra el mensaje mediante **RSA** con su clave privada y, así, recupera la clave de sesión que es utilizada por el algoritmo IDEA para descifrar la parte de datos del mensaje.

Se puede utilizar el algoritmo **RSA** con distintas longitudes de clave (384, 512, 1.024…. bits), según el nivel de seguridad requerido.

8.6.2. Privacy Enhanced Mail (PEM).

Se trata de un borrador para una norma internet que trata de dar seguridad a los servicios de correo electrónico. Se puede utilizar con el protocolo de transferencia de correo simple para internet (**SMTP**) o con cualquier otro esquema de mensajería, como X.400. Están contemplados servicios de autenticación y confidencialidad, así como la gestión de claves mediante certificados. Proporciona seguridad extremo a extremo, creando una cabecera encapsulada que se pone al principio del texto del mensaje. Esta cabecera contiene los parámetros de seguridad que incluyen integridad de mensajes y que pueden ser firmados, proporcionando así autentificación del origen, y no-repudio del origen.

La autentificación se puede hacer mediante algoritmos simétricos, **DES-ECB** con MD2 o MD5, o asimétricos, **RSA** con MD2 o MD5. Este último método proporciona además firma digital. Para realizar la gestión de clave también se pude elegir entre los mismos algoritmos simétricos o asimétricos. La confidencialidad de los datos se consigue mediante el algoritmo **DES** en modo **CBC.**

8.6.3. Seguridad en el directorio X.500

El directorio X.500 es una base de datos de comunicaciones distribuida y fácilmente accesible. Está diseñada para guardar nombres, direcciones, rutas y en

general, la información necesaria para localizar y establecer comunicación con una persona o con un recurso.

Este directorio juega un importante papel en el tema de la seguridad, concretamente en criptografía de clave pública, ya que es el lugar idóneo para guardar los certificados emitidos por una autoridad de certificación. Un certificado es un documento público que garantiza la identidad de un usuario asociando su nombre a su clave pública. Con independencia de esta función, el directorio X.500 también necesita seguridad por sí mismo, ya que ciertos accesos al directorio deben estar protegidos; sobre todo, en las cuestiones relacionadas con su mantenimiento.

Como ya se ha dicho, se trata de una base de datos distribuida que mantiene los datos en la base del directorio (**DIB**). Aunque una persona puede tener acceso a él, está especialmente pensado para que accedan de aplicaciones.

Las operaciones que se pueden realizar en el directorio son las de lectura, comparación, búsqueda, selección en listas y modificación de datos. Evidentemente, esta última operación debe estar controlada por procedimientos de identificación de los usuarios del directorio.

Cualquier usuario tiene acceso al directorio a través de un agente de usuario del directorio (**DUA**), que es un proceso de aplicación **OSI** encargado de asignar las interfaces locales como procedimientos y protocolos para usar el directorio. Los **DUA** aceden al **DIB** por los agentes del servicio del directorio (**DSA**), que son aplicaciones **OSI**. Existen dos protocolos que permiten la relación entre los distintos elementos: el **DSP** y el **DAP**. El primero de ellos permite la comunicación de dos **DSA** entre sí. El **DAP** controla la comunicación entre un **DUA** y un **DSA**.

8.7. ARQUITECTURAS DE SEGURIDAD: KERBEROS

Las actuales redes de ordenadores proporcionan numerosos servicios a un gran número de usuarios, y necesitan una arquitectura cuidadosamente diseñada para dar seguridad a las operaciones que en ellas se realizan.

La arquitectura de seguridad crea un entorno de trabajo completo, dentro del cual se realiza distintos servicios de seguridad. Deben dar respuesta adecuada a cada una de las aplicaciones que requieren seguridad, y cambiar su esquema

dependiendo de la acción solicitada (suministrar claves, autorizar o identificar a una persona, etc.) y según el tipo de aplicación que requiera el servicio. Por ejemplo, no es lo mismo autenticar correo electrónico, aplicación en la que las demoras de tiempo no tienen demasiada importancia, que realizar la autentificación en un servidor que este siendo consultado frecuentemente.

Además de dar la respuesta a los requisitos de seguridad de la red informática, hay otras características importantes a la hora de diseñar un sistema seguro. Es interesante implementar una arquitectura en la que el servidor este distribuido de forma que cuando un elemento o grupo de elementos del sistema se queden inoperativos, esto no afecte a la totalidad y deje a la red fuera de servicio. También se debe procurar que las operaciones para la autenticación sean prácticamente transparentes para el usuario que se habrá de limitar a introducir su identidad escribiendo un <<Password>> o introduciendo su **PIN** por algún otro medio (tarjeta magnética, tarjeta inteligente, bio-**PIN**, etc.). Además, el sistema ha de dar servicio a un gran número de clientes, y la ampliación de este número no debe producir ningún problema.

Todo esto conduce a la idea de utilizar una arquitectura distribuida como en el ejemplo que se ve a continuación.

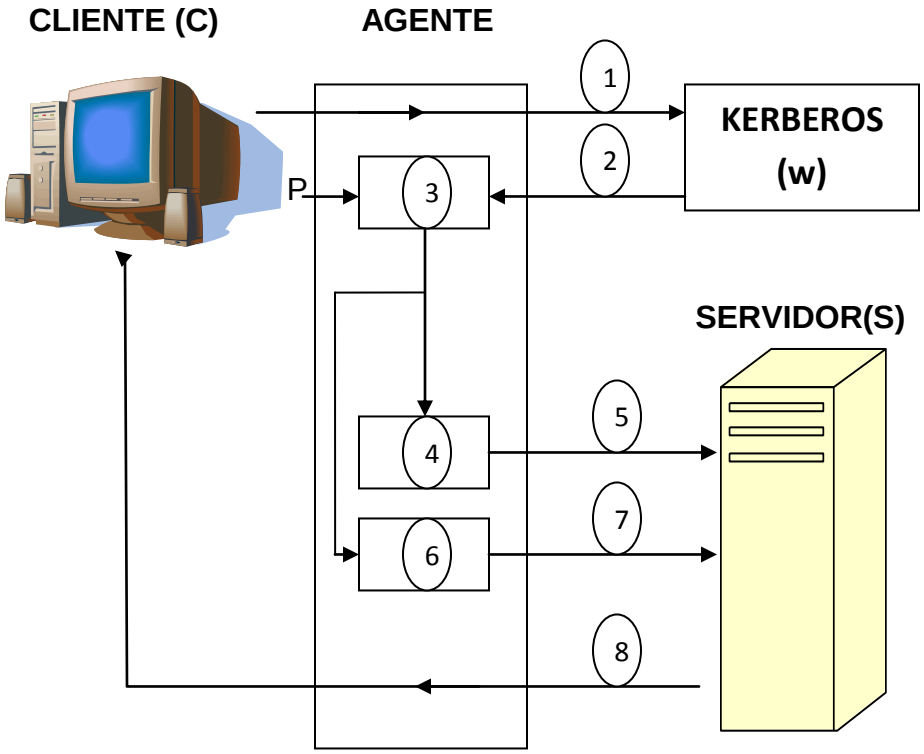
8.7.1. Kerberos

Es un servicio de autenticación con una arquitectura cliente/servidor distribuida, diseñado en el instituto tecnológico de Massachusetts (**MTI**). Se han realizado varias versiones, siendo la versión **4** la más difundida, aunque con algunas deficiencias de funcionamiento que corrige la versión **5**.

A continuación se va a ver, de forma muy resumida, el funcionamiento del sistema con la ayuda de la Figura 22. El algoritmo de cifrado simétrico utilizado es el **DES**. Un cliente (**C**) desea acceder a un sistema protegido, un servidor (**S**), y para ello ha de solicitar permiso a Kerberos (**W**) a través de un agente, paso (1). La petición ha de contener la identidad de **C** la identidad de **S** y la hora de la petición. Si Kerberos aprueba la petición envía a **C** varios parámetros, paso (2), cifrados con **DES** bajo una clave simétrica compartida por **C** y por **W**. estos parámetros son: la clave de sesión que va a ser utilizada por **C** y **S,** la de **S,** un sello de tiempo, el eco de la hora de la petición y el *ticket* solicitado, cuya información está cifrada por

una clave simétrica compartida solo por **W** y **S** de forma que solo **S** puede entenderla.

Figura 22. Sistema de autenticación de Kerberos.



Fuente: http://2.bp.blogspot.com/_glLQO-Nu-xA/TFA15AxbSgI/AAAAAAAABqc/2K1u5esjfuE/s400/sistemas9.jpg

En el paso (3), **C** descifra la información recibida de **W** con la clave secreta compartida por ambos y que ha sido calculada mediante una función unidireccional a partir de un <<Password>>, **P** de **C**; tras este paso, se ha desechar dicha clave.

Una vez descifrada la información se comprueba que si son correctas la identidad de **S** y la hora. Si es así, se valida el *ticket*, paso (4), que está cifrado con la clave simétrica compartida por **S** y **W,** y se envía a **S,** paso (5). Para que **S** tenga certeza de que el *ticket* es válido, se cifra con la clave de la sesión que está incluida en el propio *ticket*, paso (6), y se envía a **S,** paso (7). En **S** se descifran el *ticket* y su versión cifrada para autenticar a **C**, se comprueba su validez por lo

sellos de tiempo y se comparan ambas copias descifradas del *ticket.* Si coinciden, **S** envía a **C** el permiso de acceso, paso (8).

8.8. APLICACIONES BANCARIAS Y FINANCIERAS

En la mayor parte de estas aplicaciones es mucho más importante identificar perfectamente al emisor y al receptor que proteger los datos de información transmitidos durante la comunicación. Así pues, se han de cuidar mucho más los servicios de autenticación que los de confidencialidad. Esto, unido a que usualmente los mensaje enviados son cortos, hace que en este tipo de aplicaciones tenga mayor interés la utilización de algoritmos de clave pública y en algunos casos, algoritmos de cifrado en bloque.

8.8.1. Seguridad en EDI

La aplicación conocida como Intercambio Electrónico de Datos (EDI) también tiene sus servicios de seguridad bastantes definidos y normalizados. Su utilización en transacciones comerciales así lo requiere, especialmente la cuestión relacionada con los servicios de autenticación.

ITU ha elaborado dos normas: las recomendaciones F.435 y X. 435, que permiten utilizar EDI en el sistema de mensajería X.400. La recomendación de F.435 define los servicios de mensajes EDI, y la X. 435 los sistemas de mensajes EDI; es decir, como operan los servicios de escritos en la norma F.435.

Los mensajes EDI (EDIM) constan de una cabecera y de un cuerpo que, juntos, forman el campo de contenido de un mensaje X.400. En la cabecera aparecen numerosos campos relacionados con identificación y el tratamiento que ha de tener el mensaje, pero solo algunos de estos campos están relacionados con temas de seguridad. En el cuerpo del mensaje pueden aparecer intercambios del tipo EDIFACT (norma ISO 9735) o del tipo ANSI X12.

Se llama notificación EDI (EDIN) a un mensaje enlazado al mensaje original, a través de referencias cruzadas, con el que tiene una serie de campos comunes seguidos de campos que indican el tipo de notificación. Las notificaciones EDI permiten confirmar la recepción del mensaje en el nivel de aplicación y no solo su reparto.

F.435. y X. 435 introducen elementos de seguridad adicionales a los ya presentes en X.400, fundamentalmente para garantizar la autenticidad y evitar el repudio tanto de los mensajes como de la notificaciones. La persona que origina el mensaje EDI puede solicitar los servicios de justificación/no-repudio de la notificación EDI activando los bits adecuados en el campo de petición de notificación que hay en la cabecera del mensaje.

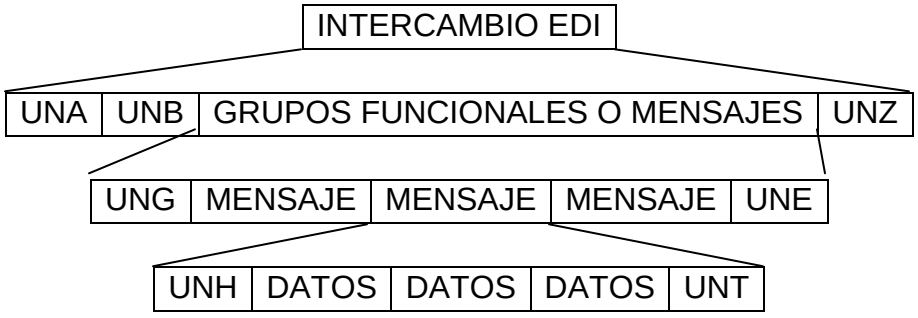
8.8.2. EDIFACT

Aunque la X.400 se utiliza para proteger un intercambio de datos EDI completo, mensajes más notificación, no es la única infraestructura que puede ser utilizada. Los intercambios están compuestos por mensajes y grupos funcionales, grupos de mensajes similares, todos con el mismo destino, pero no todos necesitan protección. La X. 435 no tiene capacidad para realizar una seguridad selectiva.

Por este motivo, para proteger los intercambios, y sus subunidades, se ha pensado en construir la protección dentro de la misma estructura EDI utilizando unos indicadores. Para regular este tipo de operaciones han aparecido varias normas, entre las que destacan la serie ANSI X12 y, muy especialmente, EDIFACT.

La estructura de un mensaje EDIFACT puede verse en la Figura 23. En ella se ve que cada parte tiene unos delimitadores de cabecera (UNA, UNB, UNG, UNH) y de final (UNZ, UNE, UNT). Los delimitadores de la cabecera identifican y especifican el intercambio y el grupo funcional o mensaje, mientras que los delimitadores de final se utilizan para realizar chequeos.

Figura 23. Estructura de un mensaje **EDI**



El programa TEDIS, de la comisión europea, ha elaborado una propuesta sobre los servicios de la seguridad en EDIFACT, principalmente para controlar la autenticidad en los mensajes y en los niveles de intercambio basada en la criptología de clave asimétrica.

8.8.3. ISO 8730

Se trata de una de las normas más antiguas y más extendidas en temas de seguridad para transferencias interbancarias. La norma **ISO 8730,** basada en la norma **ANSI X9.9,** utiliza cifrado de clave simétrica (**DES**) para autenticar las transferencias. Los mecanismos de distribución de claves están regulados por la norma **ISO 8732**.

En la norma **ISO 8730** existen varias opciones para tratar los datos formateados a la hora de calcular el MAC. Entre estas opciones esta la posibilidad de considerar los datos como texto o como un fichero binario. Si el mensaje se trata como texto, se puede dejar como esta o editarlo, suprimiéndole los caracteres superfluos, que hacen más lenta la transmisión y le quitan transparencia; también, en el caso de tratar los datos como texto, se puede seleccionar o decidir que campos se incluyen.

Según **ISO 8730,** varios de los campos que integran una transferencia interbancaria deben de estar incluidos en el mensaje de autenticación. Entre ellos están:

- ✓ **MAC:** código de autenticación del mensaje, que consta de ocho dígitos hexadecimales.
- ✓ **DMC:** fecha en que se calculo el MAC.
- ✓ **IDA:** identificador para que el receptor utilice la clave de autentificación.
- ✓ **MID:** identificador de mensaje. Se trata de un número generado por el emisor a partir de **DMC** e **IDA** para protección contra la duplicación o la pérdida del mensaje.
- ✓ Elementos específicos en el texto del mensaje, como valor de la transacción, entidades participantes, beneficiarios, etc.

Los distintos campos tienen sus propios delimitadores, que marcan el inicio y el final de cada uno de ellos.

8.8.4. SWIFT (Society for Worldwide Interbank Financial Telecommunication)

Ofrece un servicio Para transferencia de pagos con varios mecanismos de seguridad, como son el cifrado de líneas interurbanas, la protección de acceso a la red mediante el empleo de códigos y la posibilidad de cifrado en las conexiones usuario-red alquiladas. Los usuarios de **SWIFT** suelen utilizar productos específicos, suministrados por esta entidad, que añaden a sus terminales para permitir la identificación del operador.

Las comunicaciones se realizan sobre una red de paquetes conmutados tipo **X.25,** y las transacciones financieras pueden tener varios protocolos, formatos y otras medidas de seguridad, como las autenticaciones. **SWIFT** ha mejorado la seguridad básica de la red añadiendo algunos elementos de seguridad de usuario, como el intercambio seguro de claves y el control de acceso con tarjetas inteligentes.

Para realizar operaciones de gran volumen-pago de pensiones, nominas, etc.-, información de gestión de riesgos, protección del estado de cuentas y otras, **SWIFT** ha creado la transferencia de ficheros interbancaria (**IFT**) que opera sobre **X.400.** Para la transferencia de ficheros se utiliza un protocolo, el **pIFT,** dentro de los mensajes **X.400,** que incluye una cabecera de seguridad con un <<token>> (basado en **X.509**).

Los servicios de seguridad incluidos son: integridad del contenido del **p IFT,** autenticación del mensaje original y confidencialidad. Se pueden usar varios algoritmos de clave simétrica o asimétrica, para lo que el emisor genera un <<token>> que contiene un identificador de algoritmo seleccionado. Son funciones de seguridad extremo a extremo, y la elección de los algoritmos o la gestión de las claves no tienen por qué depender de **SWIFT.**

También se están desarrollando productos **EDI** tanto para **X.25** como para **X.400,** y, posiblemente, el formato **EDIFACT** en los mensajes valla reemplazado paulatinamente al formato original diseñado por **SWIFT**.

8.8.5. ETEBAC 5

Es un protocolo diseñado por el *Comité Francais d`Organisation et de Normalisation bancaires* (**CFONB**) para la utilización en la banca francesa, y que permite realizar de forma segura operaciones de cierta envergadura entre instituciones financieras y sus clientes.

ETEBAC 5 utiliza un protocolo de transferencia de ficheros llamado PeSIT y opera sobre una línea X.25. Acepta criptografía con clave simétrica o asimétrica, **DES** y **RSA,** respectivamente.

Entre los servicios de seguridad ofrecidos por **ETEBAC 5** están la autenticación reciproca entre el banco y el cliente, la integridad de los datos con un MAC, el no-repudio reciproco y la confidencialidad de la transferencia con **DES** (es un servicio opcional) **ETEBAC 5** depende de una autoridad de certificación, tipo X.509, que proporcione certificados a los usuarios en los que vaya incluida su clave pública.

Se distinguen dos niveles de operación asociados con la transferencia de ficheros, cada uno con una clave diferente. La tarea de un nivel es proteger el contenido de los ficheros intercambiados, mientras que el otro nivel se encarga de la seguridad del establecimiento real de la transmisión.

ETEBAC 5 contiene muchos elementos que proporcionan una gran flexibilidad y seguridad a los servicios ofrecidos. Por ejemplo, no solo se firma el contenido del fichero, sino también el MAC del identificador del fichero. **ETEBAC 5** ha sido el primer estándar especializado en incluir **RSA** como criptografía de clave pública.

9. CONCLUSIONES

Para proporcionar una base solida y clara de la teoría de la información se hace necesario el conocimiento de los conceptos que la involucran para contar con una valiosa herramienta en el momento de explorar técnicas de cifrado de datos.

Adquirir información histórica para conocer las raíces de la criptografía aumenta el entendimiento de la ciencia como tal y permite realizar una abstracción más allá de los métodos matemáticos para crear un algoritmo de cifrado nuevo y/o modificar uno existente.

Estudiar algoritmos de cifrado simples que requieren poca fundamentación matemática, permite familiarizarse más a fondo con los términos y los algoritmos antiguos más famosos, además se puede llegar a la conclusión de que no es de vital importancia tener avanzados conocimientos matemáticos para inmiscuirse en este mundo de métodos, ideas e imaginación que exigió al hombre al máximo para ocultar su información, mundo del cual podemos hacer parte si se tiene un objetivo definido y una buena imaginación.

Es difícil no apasionarse con este tema cuando se conoce su historia y sus aportes en ella, por tal motivo es buena idea para el investigador profundizar en algoritmos cada vez más complejos y con más bases matemáticas que a su vez preparen la mente con el ánimo de perfilarse a algoritmos de alta complejidad.

Con toda ciencia viene su opuesta o su complementaria, pero en el caso del criptoanálisis puede decirse que es la ciencia enemiga, ya que al repasar sus bases matemáticas e históricas se deduce su prioridad de desenmascarar los secretos de los documentos cifrados haciendo que cada día la criptografía se esfuerce cada vez mas y mas sacando lo mejor de sus creadores con el ánimo de evitar ser derrotados tanto por ataques de fuerza bruta como por herramientas criptoanalíticas como el generador Auto Shrinking.

En la ciencia de la esteganografía su éxito radica en la selección deliberada del medio en el que se desea camuflar la información, sus fundamentos teóricos, históricos y matemáticos permiten ver que no es complementaria o dependiente de la criptografía ni muchos menos podrá reemplazarla, pero pueden colaborarse entre sí, no hay barrera ya que se puede llegar hasta donde la imaginación lo permita.

Las aplicaciones en redes necesitan de algoritmos de cifrado muy seguros para el momento de transmitir información de manera segura, unos son fuertes en su estructura, sus métodos y sus rondas como el DES y el AES, otros son fuertes por sus bases matemáticas a pesar de no tener un sistema robusto como el RSA, lastimosamente el único de estos que aun se considera lo bastante seguro para estar como estándar internacional es el AES ya que los otros dos tienen tiempos de hasta un día en ser rotos mediante ataques de fuerza bruta.

En las redes actuales donde los volúmenes de información son extremadamente altos, se hace necesaria la creación de herramientas que complementen a la criptografía, como los protocolos y los estándares que permiten el uso confiable de herramientas de la red como los correos electrónicos, a pesar de tener mucha dependencia de la clave de cifrado y esta deba transmitirse por el mismo medio, se han desarrollado métodos y herramientas como los algoritmos hash y las firmas digitales que permiten que entidades como los bancos confíen en la criptografía para transmitir su información.

La seguridad en la red es un tema demasiado delicado ya que cualquier error en un algoritmo o algún cambio que no haya sido previamente probado y planificado puede afectar de manera exponencial la seguridad y pasar de ser demasiado seguro a ser lo bastante vulnerable como para ser roto en corto tiempo o permita que se filtre información por fuera de sus destinos.

REFERENCIAS BIBLIOGRÁFICAS

Criptografía [en línea]. INIXA seguridad y comunicación [citado 20 Junio 2010]. Disponible en internet: <http://www.inixa.com/seguridad_criptografia.php>

JESÚS SILVA, El almacenamiento de información en las computadoras, Agosto 27 de 2001 [en línea] Copyright 2000 CLAVE EMPRESARIAL COM, S.A. DE C.V. [Citado 10 Julio 2010]. Disponible en internet: <http://www.claveempresarial.com/principiantes/notas/nota010827b.shtml>

La Criptografía como elemento de la seguridad informática [en línea] Biblioteca virtual en salud de cuba [citado 25 Junio 2010]. Disponible en internet: <http://www.bvs.sld.cu/revistas/aci/vol11_6_03/aci11603.htm>

PINO CABALLERO Gil. Introducción a la criptografía. Editorial Ra-Ma Año 2002 – p. 133 – ISBN: 84-7897-520-9

Data Encryption [en línea] Federal Information Processing Standards. Standard (FIPS 46-2). [Citado 20 julio 2010]. Disponible en internet: <http://www.itl.nist.gov/fipspubs/fip46-2.htm>

POE Edgar Allan. El Escarabajo de Oro. Editorial Altaya. Año 1994 – publicado por primera vez en junio de 1843, en Philadelphia.

ÁNGULO Carlos Alberto, OCAMPO Sandra Milena, BLANDON Luis Hernando, Una mirada a la esteganografía. En: Revista Scientia et Technica, Universidad Tecnologica de Pereira, Pereira Diciembre de 2007

A study on encryption algorithms and modes for disk encryption [en linea] Universidad tecnologica de pereira – Biblioteca virtual de la IEEE [citado 26 mayo 2010] Disponible en internet: <http://ieeexplore.ieee.org/search/srchabstract.jsp?tp=&arnumber=5166897&queryText%3D%28encryption%29%26openedRefinements%3D*%26ranges%3D2008_2010_Publication_Year%26pageNumber%3D2>

"Request for Comments on Candidate Algorithms for the Advanced Encryption Standard (AES)", Federal Register, Volume 63, Number 177, pp. 49091-49093, Sept 14, 1998.

RSA97 Data Security Inc., "Goverment Encription Standard Takes a Fall." RSA Data Press Relace, Junio 17 1997.

J. Daemen, "Cipher and Hash Function Design Strategies Based on Linear and Diferential Cryptanalysis," Doctoral Dissertartion, Marzo 1995, K.U.Leuven.

Criptosistemas de clave secreta [en linea], ibiblio Biblioteca Virtual [Citado 20 agosto 2010]. Disponible en línea: <http://www.ibiblio.org/pub/linux/docs/LuCaS/Manuales-LuCAS/doc-unixsec/unixsec-html/node308.html>

Criptosistemas de clave pública [en linea] ibiblio Biblioteca Virtual [citado 22 agosto 2010], Disponible en internet: <http://www.ibiblio.org/pub/linux/docs/LuCaS/Manuales-LuCAS/doc-unixsec/unixsec-html/node309.html>

Amenazas a la seguridad informática [en linea], VirusProt El Sitio Líder en Seguridad Informática [citado 25 agosto 2010], Disponible en línea: <http://www.virusprot.com/Amzsolup.html>

LEON BATTISTA Alberti, Un tratado en cifras, transporte. En: A. Zaccagnini. Advertencia cerca David Kahn, Galimberti, Torino 1997.

Criptosistema de Hill [en linea], Textos científicos [citado 26 septiembre 2010]. Disponible en internet: <www.textoscientificos.com/criptografia/hill>

ANGEL ANGEL José de Jesús, Antecedentes matemáticos, AES - Advanced Encryption Standard, versión 2005, Disponible en internet: <http://computacion.cs.cinvestav.mx/~jjangel/aes/AES_v2005_jjaa.pdf>

LOPEZ LOPEZ Eddie Estuardo, Análisis de métodos de encriptación y seguridad en redes, Trabajo de grado Ingeniero de sistemas y computación, Guatemala, Universidad Francisco Marroquin, Facultad de Ingeniería de sistemas Informática y ciencias de la computación, 2001, 63 p.

Breve Historia de La Criptografia Clasica [en línea], José Luis Tábara – Scribd [citado en 27 mayo 2010], Disponible en internet: <http://www.scribd.com/doc/16660111/Breve-Historia-de-La-Criptografia-Clasica>

DOMÍNGUEZ ESPINOZA Edgar Uriel y PACHECO GÓMEZ Leonardo, Algoritmos de criptografía clásica, Trabajo de grado Ingeniero de sistemas y computación, Ciudad de México. Facultad de Ingeniería, Universidad nacional autónoma de México, 2007 13 p.

La criptografía Clásica.pdf [en línea], Santiago Fernández – Universidad del gobierno vasco [citado 02 mayo 2010], disponible en línea <http://www.hezkuntza.ejgv.euskadi.net/r43-573/es/contenidos/informacion/dia6_sigma/es_sigma/adjuntos/sigma_24/9_Criptografia_clasica.pdf>

LUCENA LÓPEZ Manuel José, Seguridad en Redes de Computadores En: Criptografía y Seguridad en Computadores, tercera edición, mayo de 2003, p 181.

LUCENA LÓPEZ Manuel José, Aplicaciones Criptográficas En: Criptografía y Seguridad en Computadores, cuarta edición, junio 2008, p 221.

ARRIAZU Jorge Sánchez, Descripción del algoritmo DES [en línea], tierradelazaro [citado 12 de junio 2010], Disponible en internet: <http://www.tierradelazaro.com/public/libros/des.pdf>

Criptografía Clásica [en línea], Universidad autónoma de Madrid [citado 09 julio 2010], Disponible en internet: <http://www.uam.es/proyectosinv/estalmat/Actividades%20Estalmat-ICM2006/Sistemas%20de%20cifrado.pdf>

Protocolos criptográficos [en línea], Juan Tena Ayuso – criptored, [citado 10 julio 2010]. Disponible en internet: <http://www.criptored.upm.es/guiateoria/gt_m023c.htm>

Criptografía clásica, Cifrado por bloques [en línea], Mygnet la comunidad de la programación, [citado 23 agosto 2010], Disponible en internet: <http://www.mygnet.net/articulos/seguridad/criptografia_de_llave_privada_parte_3.837>

GUZMAN SALAS Juan Manuel, Implementación en un FPGA del algoritmo de encriptación doble ronda como una solución al teorema LR, Trabajo de grado Maestro en Ciencias en tecnología de Computo, México DF, Instituto Politécnico Nacional Centro de Innovación y Desarrollo Tecnológico en Computo, 2007 9-12 p

PARA Christof y PELZL Jan, The Data Encryption Standard (DES) and Alternatives En: Understanding Cryptography, New York, Springer, 2010, p 55

PARA Christof y PELZL Jan, The Advanced Encryption Standard (AES), Op. cit. p 87.

PARA Christof y PELZL Jan, The RSA Cryptosystem, Op. cit. p 173

Capítulo 1, La complejidad de los algoritmos [en línea], Lenguajes y ciencias de la computación, Universidad de Malaga, [citado 05 enero 2011], Disponible en internet <http://www.lcc.uma.es/~av/Libro/CAP1.pdf>

José A. Mañas, Análisis de Algoritmos: Complejidad, Noviembre, 1997 [en línea], Departamento de ingeniería de sistemas Telemáticos, E.T.S Ingenieros de telecomunicación, Universidad Politécnica de Madrid [citado 06 enero 2011], Disponible en internet <http://www.lab.dit.upm.es/~lprg/material/apuntes/o/index.html>

Difusión perfecta en criptografía [en línea], Kriptopolis [citado 09 enero 2011], Disponible en internet <http://www.kriptopolis.org/docs/El_Difusor.pdf>

Criptografía cuántica [en línea], Diviertenet [citado 11 enero 2011], Disponible en internet < http://www.diviertenet.com/foro/programacion/2747-criptografia-cuantica.html>

Criptografía Cuántica - Principio y algoritmos [en línea], Textos científicos [citado 12 enero 2011], Disponible en internet < http://www.textoscientificos.com/criptografia/criptoquantica>