Fundamentos de encriptación y seguridad de la información

Ing. Diego Fernando Veloz Chérrez
Ing. Marcela Estefanía Mora Campana
Ing. Fabricio Javier Santacruz Sulca

compAs
Grupo de capacitación e investigación pedagógica

Fundamentos de encriptación y seguridad de la información

Ing. Diego Fernando Veloz Chérrez
Ing. Marcela Estefanía Mora Campana
Ing. Fabricio Javier Santacruz Sulca

Este libro ha sido debidamente examinado y valorado en la modalidad doble par ciego con fin de garantizar la calidad científica del mismo.

© Publicaciones Editorial Grupo Compás
Guayaquil - Ecuador
compasacademico@icloud.com
https://repositorio.grupocompas.com

Veloz, D., Mora, M., Santacruz, F. (2023) Fundamentos de encriptación y seguridad de la información.. Editorial Grupo Compás

© Ing. Diego Fernando Veloz Chérrez
Ing. Marcela Estefanía Mora Campana
Ing. Fabricio Javier Santacruz Sulca

ISBN: 978-9942-33-670-5

Contenido

PREFACIO

Estimado lector, la finalidad de este libro es reunir información sobre criptografía y criptoanálisis que sean útiles para orientarlo en esta área. La intención es brindarle una herramienta que recopila información de varios autores y casos de estudio propuestos para que se logren entender de mejor manera los conceptos. El temario aborda una amplia variedad de temas entre los que se destacan diferentes técnicas criptográficas y sus usos, las cuales tienen como propósito específico prevenir las vulnerabilidades de seguridad más comunes en sistemas informáticos.

La importancia de escribir otro libro más sobre encriptación se debe a la importancia de estas técnicas que son consideradas como una de las medidas más elementales para la prevención de ataques a los datos que son transmitidos, almacenados y distribuidos por infraestructuras tecnológicas. Por lo tanto, la intención de los autores es recopilar información relevante que ayude a entender de una manera clara esta temática. Así mismo, ha sido escrito con el uso de terminología que pueda ser entendible tanto para personas que inician su camino en esta área de la seguridad como para estudiantes de carreras técnicas como Telecomunicaciones, Redes y Desarrollo de software que deseen complementar su conocimiento en fundamentos de encriptación. Sin embargo, en un mundo con dependencia tecnológica, estos conocimientos deberían ser comprendidos por todos los usuarios de tecnología con acceso a infraestructuras de comunicación.

Esperamos cumplir con las expectativas de conocimiento que cada uno de ustedes tengan.

Los autores

INTRODUCCIÓN

La seguridad en la transmisión de información ha tomado parte fundamental en la historia de la humanidad, desde la época prehistórica hasta la actualidad, la seguridad de la información juega un papel muy importante en las comunicaciones, ya sea del ejército, del gobierno, de los periodistas, la necesidad de poder enviar información de forma segura hace que nazca la criptología, y con ella, la criptografía.

La criptografía es una herramienta muy útil cuando se desea tener seguridad informática; puede ser también entendida como un medio para garantizar las propiedades de confidencialidad, integridad y disponibilidad de los recursos de un sistema. Se considera la criptografía como el nombre genérico con el que se designan dos disciplinas opuestas y a la vez complementarias, es importante tener claros los conceptos básicos que están detrás de los sistemas criptográficos modernos. Estos conceptos van desde entender qué es la criptografía, cómo está clasificada, entender el funcionamiento básico de algunos sistemas de cifrado y conocer cómo se forman los documentos digitales como firmas y sobres digitales.

1.1 HISTORIA DE LA CRIPTOGRAFÍA

La criptografía surge aproximadamente en los orígenes del hombre, desde que este aprendió a comunicarse o incluso podemos mencionar que es tan antigua como la escritura. Por este motivo tuvo que encontrar medios que le permitieran asegurar parte de sus comunicaciones, o mejor dicho necesitaba mantener sus mensajes con cierta confidencialidad. El principio fundamental de la criptografía, es mantener una comunicación entre dos personas de forma que sea incomprensible por el resto.

Criptografía Antigua

La criptografía antigua abarca desde los orígenes del hombre hasta el año 500 D.C. durante esta época se empezó a desarrollar la escritura, primero con imágenes que representaban cierta

acción u objetos, después los símbolos para poder escribir. Un caso de ello son las pinturas rupestres que se han encontrado en muchas partes del mundo y que datan de hace 14.000 años aproximadamente. Con el paso del tiempo, y la evolución de la humanidad, se empezaron a desarrollar la escritura, primero representando imágenes y signos con significados, como los jeroglíficos de los egipcios, en los cuales en cierto modo se puede apreciar lo que es la criptografía, ya que su lenguaje no es fácil de entender en la actualidad y requiere un análisis de patrones, y además, sobre su propia escritura, los egipcios empezaron a practicar la criptografía aplicando cifrado a su ya de por si complicado lenguaje. Pero aún no hay mucha información al respecto. (Patricia, 2009)

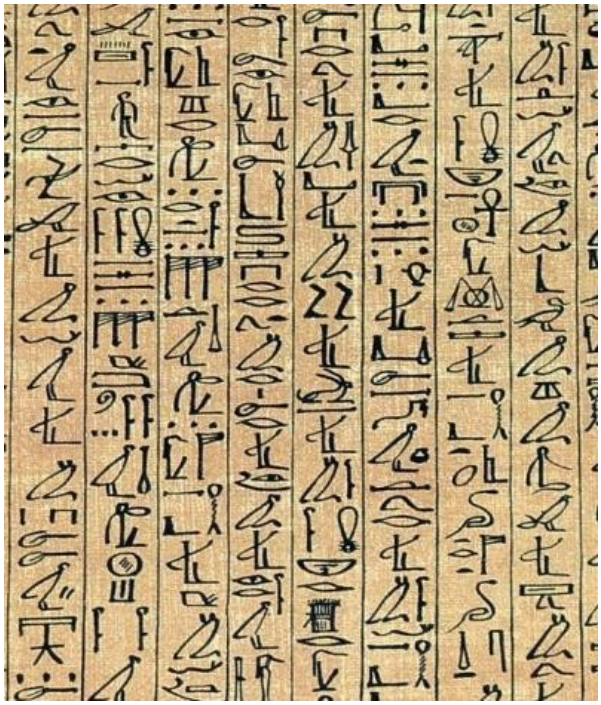


Fig. 1 Criptografía Antigua

Criptografía Medieval

Fue probablemente el análisis textual del Corán, de motivación religiosa, lo que llevó a la invención de la técnica del análisis de frecuencias para romper los cifrados por sustitución mono alfabéticos, en algún momento alrededor del año 1000. Fue el avance criptoanalítico más importante hasta la Segunda Guerra Mundial. Esencialmente, todos los cifrados quedaron vulnerables a esta técnica criptoanalítica hasta la invención del cifrado poli alfabético por Leon Battista Alberti (1465), y muchos lo siguieron siendo desde entonces. (Ventura, 2011)

Criptografía moderna

Con la llegada de los computadores y la posibilidad de cálculo masivo de alta velocidad, la criptografía entró en crisis, Claude Shannon en los cincuentas escribió su famoso artículo A Comunicación Theory of Secrecy Systems, que es uno de los pilares de la teoría de la información. En base a eso se creó la criptología moderna con principios distintos y mucho más formales que la tradicional.

Fin del secreto. Los métodos detallados para encriptar deben ser públicos, conocidos por todos. Mientras más gente los pueda revisar, mejor. La dificultad debe estar en un problema matemático que no tenga solución conocida, por ejemplo la factorización de grandes números.

Secreto perfecto. Cualquier criptograma se puede romper, en teoría por fuerza bruta, es decir probando todas las combinaciones posibles. Existe un solo método para lograr un criptograma inmune al análisis masivo, se llama "one time pad"1 pero no es práctico porque requiere de claves del mismo porte o mayores que el propio mensaje.

Seguridad física. A diferencia de la teórica esta se podría desencriptar2 si se tuviese una enorme cantidad de tiempo (miles de siglos por ejemplo) y de capacidad de cálculo, con el avance de los computadores esta seguridad va bajando a medida que pasa el tiempo: lo que era físicamente seguro hace 10 años.

En los setentas Define y Hellman inventaron las claves públicas, que solucionaron el problema de seguridad al enviar una clave por canal inseguro. Cada usuario tiene una clave privada que conserva para sí y una pública, que cualquiera puede ver, para encriptar o descifrar se necesita tener ambas claves. (Miret, 2013)

Desde entonces hasta hoy ha habido un crecimiento espectacular de la tecnología criptográfica, si bien la mayor parte de estos avances se mantenían y se siguen manteniendo, según algunos en secreto. Financiadas fundamentalmente por la NSA (Agencia Nacional de Seguridad de los EE.UU.), la mayor parte de las investigaciones hasta hace relativamente poco tiempo han sido tratadas como secretos militares. Sin embargo, en los últimos años, investigaciones serias llevadas a cabo en universidades de todo el mundo han logrado que la Criptografía sea una ciencia al alcance de todos, y que se convierta en la piedra angular de asuntos tan importantes como el comercio electrónico, la telefonía móvil, o las nuevas plataformas de distribución de contenidos multimedia. La experiencia ha demostrado que la única manera de tener buenos algoritmos es

[1] One time pad llamado Relleno de un solo uso, es un tipo de algoritmos de cifrado por el que el texto en claro se combina con una clave aleatoria o «libreta» igual de larga que el texto en claro y que sólo se utiliza una vez.

[2] Desencriptar.- denota el paso inverso a cifrar, que sería descifrar .

que estos sean accesibles, para que puedan ser sometidos al escrutinio de toda la comunidad científica. (Lucerna López, 2010)

1.2 CONCEPTOS BASICOS

1.2.1 CRIPTOGRAFÍA

Proviene en un sentido etimológico del griego Kriptos=ocultar, Graphos=escritura, y su definición es: "Arte de escribir con clave secreta o de un modo enigmático".

La criptografía es la ciencia que se encarga de aplicar técnicas de protección de información ante terceros basándose en la ocultación o la descomposición de la misma. Actualmente ha alcanzado mayor auge en áreas de Informática, Telemática y en las Matemáticas, en donde se emplean algoritmos para poder cifrar información. De hecho, con la invención de las computadoras, y luego de la redes de computadoras, se hizo necesario la seguridad de transmisión de datos de un lugar a otro. Lo que originalmente nació solo para proteger comunicaciones científicas y militares, pronto alcanzo a todos los medios civiles por el boom de la computadora y el internet, en donde se necesita establecer comunicaciones seguras y que no puedan ser interceptadas. (Dávila, 2011)

La aplicación de la criptografía es para aumentar la seguridad de un sistema informático en l a actualidad. Se debe tener en cuenta que seguridad no solo se refiere a la protección de datos a nivel lógico, se deben de tener en cuenta varios factores al momento de enviar un mensaje por ejemplo:

Alguien que pretende mandar información confidencial aplica técnicas criptográficas para poder "esconder" el mensaje (se llama cifrar o encriptar), manda el mensaje por una línea de comunicación que se supone insegura y después solo el receptor autorizado pueda leer el mensaje "escondido" (se llama descifrar o descencriptar)

Fig. 2 Envió de Información

1.2.2 OBJETIVOS DE LA CRIPTOGRAFÍA

Seguridad física: En este nivel se toma en cuenta la protección que tiene un sistema informático a nivel físico. Por ejemplo, medidas contra incendios, políticas de backup, el acceso que tiene un usuario a un sistema informático a nivel físico, y con ello al nivel de acceso que tiene a la información que alberga.

Seguridad de la información: en este punto se presta atención a la preservación de la información ante terceros, y ante personal no autorizado. En este punto se emplea la criptografía para proteger la información tanto en sistemas aislados como en sistemas interconectados

Seguridad del canal de comunicación: Un canal de comunicación se considera inseguro debido a que los enlaces que generan esta fuera del alcance de un usuario, ya que pertenecen a terceros. En este punto es donde se toma en consideración el uso de la criptografía ya que es imposible saber si el canal está siendo escuchado o intervenido.

Problemas de autentificación: el uso de canales inseguros de plantea otro problema, que es el asegurarse de que la información que enviamos o recibimos, sea realmente de quien creemos que viene. Entonces en este punto se hace necesario el uso de certificados digitales.

Problemas de suplantación: la suplantación de identidad es un problema que afecta tanto a sistemas aislados como a sistemas interconectados, ya que un usuario no autorizado puede acceder a un sistema desde dentro o desde fuera de esta. En este punto se requiere del uso de una contraseña para tener acceso al sistema

No repudio: en este punto, el no repudio va de la mano con la autentificación del usurario. Cuando se recibe un mensaje, no solo se hace necesario identificar quien lo envió, sino que también el emisor asuma las responsabilidades derivadas de la información que acaba de enviar. En este punto es necesario impedir que el emisor pueda negar su autoría sobre:

La criptografía se divide en dos grandes ramas

la criptografía de clave privada o simétrica

la criptografía de clave pública o asimétrica,

DES pertenece al primer grupo y RSA al segundo. (Angel, 2003)

1.2.3 CRIPTOANÁLISIS

El criptoanálisis consiste en comprometer la seguridad de un criptosistema. Esto se puede hacer descifrando un mensaje sin conocer la llave, o bien obteniendo a partir de uno o más criptogramas la clave que ha sido empleada en su codificación. No se considera criptoanálisis el descubrimiento de un algoritmo secreto de cifrado; hemos de suponer por el contrario que los algoritmos siempre son conocidos. (MASTER, 2004)

Fig. 3 Maquina Enigma (Spencer Platt).

El mecanismo que se emplean para obtenerlos es indiferente, y puede ser resultado de escuchar un canal de comunicaciones, o de la posibilidad de que el objeto de nuestro ataque responda con un criptograma cuando se envía un mensaje. Obviamente, cuanto mayor sea la cantidad de pares, más probabilidades de éxito tendrá el criptoanálisis. (EcuRed, 2011)

1.2.4 CRIPTOSISTEMA

Se define como una cuaterna de elementos formados por un conjunto finito llamando alfabeto, a partir del cual, y utilizando ciertas normas sintácticas y semánticas podremos emitir un mensaje en claro (plaintext) u obtener el texto en claro correspondiente a un mensaje cifrado (ciphertext).

1.2.5 ESTRUCTURA DE CRIPTOSISTEMA

Definiremos un criptosistema como una quíntupla de variables (M, C, K, E, D), donde:

m: representa el conjunto de todos los mensajes sin cifrar (lo que se denomina texto claro, o plaintext) que pueden ser enviados.

C: representa el conjunto de todos los posibles mensajes cifrados, o criptogramas.

K: representa el conjunto de claves que se pueden emplear en el criptosistema.

E: es el conjunto de transformaciones de cifrado o familia de funciones que se aplica a cada elemento de M para obtener un elemento de C. Existe una transformación diferente E_k para cada valor posible de la clave k.

D: es el conjunto de transformaciones de descifrado, análogo a E. (Lucerna López, 2010)

Todo criptosistema ha de cumplir la siguiente condición:

$$\mathbf{Dk(Ek(m)) = m}$$

Es decir, que si tenemos un mensaje m, lo ciframos empleando la clave k y luego lo desciframos empleando la misma clave, obtenemos de nuevo el mensaje original m, ver fig. 4.

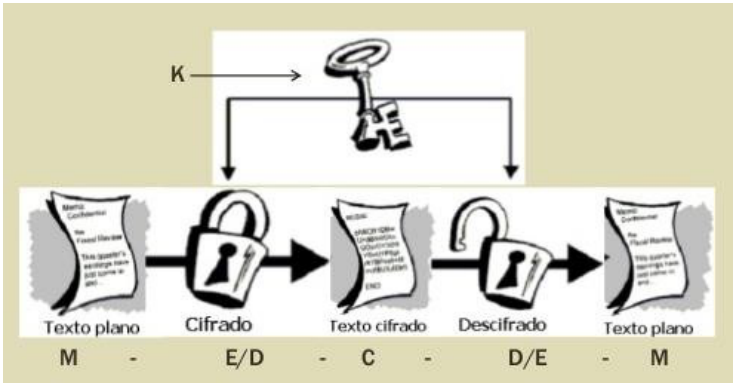


Fig. 4 Ejemplo básico de cifrado

1.2.6 ESTRUCTURA DEL CRIPTOSISTEMA

Existen dos tipos fundamentales de criptosistemas utilizados para cifrar datos e información digital y ser enviados posteriormente después por medios de transmisión libre:

Criptosistemas simétricos o de clave privada: Son aquellos 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 nos lleva preguntarnos como transmitir la clave de forma segura.

Criptosistemas asimétricos o de llave pública: que emplean una doble clave (K_p se conoce como clave privada) y kP se conoce como clave pública. Una de ellas sirve para la transformación E de cifrado y la otra para la transformación D de descifrado.

En muchos casos son intercambiables, esto es, si empleamos una para cifrar la otra sirve para descifrar y viceversa. Estos criptosistemas deben cumplir además que el conocimiento de la clave pública kP 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. (Lucerna López, 2010)

1.2.7 ESTEGANOGRAFÍA

Del griego steganos (oculto) y graphos (escritura), la esteganografía se puede definir como la ocultación de información en un canal encubierto con el propósito de prevenir la detección de un mensaje oculto.

La esteganografía estudia el conjunto de técnicas cuyo fin es insertar información sensible dentro de otro fichero. A este fichero se le denomina fichero contenedor3. De esta forma, se consigue que la información pase inadvertida a terceros, de tal forma que sólo sea recuperada por un usuario legítimo que conozca un determinado algoritmo de extracción de la misma. (Inteco, 2001)

Se utilizan dos elementos, el mensaje (la información que queremos ocultar) y el camuflaje o tapadera (el elemento que hará de tapadera), una función estego4 y quizás algún tipo de clave necesaria para descifrarla.

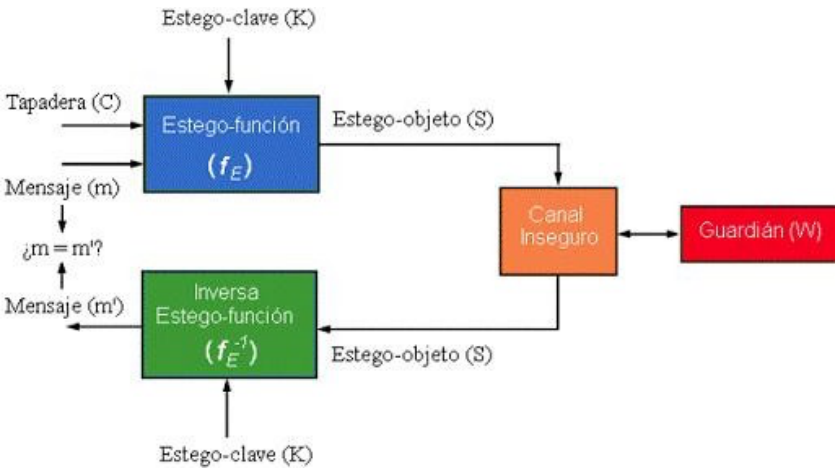


Fig. 5 Ejemplo básico de Esteganografía.

En la fig. 5. El camuflaje (con el mensaje oculto en su interior) viajará por un canal previsiblemente inseguro (una página web, o el camino entre dos reinos), pudiendo ser interceptado por algún otro receptor (llamado habitualmente guardián). Y el objetivo es que el guardián sea incapaz de, o bien darse cuenta de que el camuflaje tiene un mensaje oculto, o en caso de darse cuenta, que no pueda obtener el mensaje.

Una vez que llega a su destino, el receptor objetivo aplicará la estego-función y la estego-clave (si es que había), separando camuflaje de mensaje, y obteniendo así la información buscada.

1.2.8 TÉCNICAS DE ESTEGANOGRAFÍA

Existen muchos otros tipos de Esteganografía:

Esteganografía pura: No existe estego-clave, y por tanto, se presupone que el guardián es incapaz de reconocer una información oculta mediante estego de un mensaje normal. Se aplica

[3] Fichero contenedor.- Son gráficos, documentos, programas ejecutables, etc.
[4] Estego.- Algún tipo de técnica para que el elemento oculto pase desapercibido dentro del camuflaje.

entonces seguridad basada en la oscuridad, y el mejor ejemplo es el mini reto hacking que proponía hace un par de semanas, donde la imagen tenía oculta otra imagen que cualquiera podía obtener abriendo el camuflaje (la primera imagen) con un descompresor cualquiera.

Esteganografía de clave secreta: En este caso, la estego-función depende de una clave que deben conocer tanto el emisor como el receptor. Un ejemplo sencillo sería un texto oculto bajo otro texto generado aleatoriamente a partir de un capítulo de un libro que los dos (emisor y receptor) conozcan y una contraseña que dictará el cómo descifrar el mensaje camuflado.

Atendiendo ya a los canales utilizados, podemos hablar de:

Esteganografía en texto: Ofuscar información en texto, de manera que algunas (o todas) las palabras del mismo, sirvan para descifrar el mensaje. Hay varias técnicas, pero la mayoría se basan en aplicar algún algoritmo que permita elegir diferentes estados.

Ejemplo

Un 0 o un 1 según diferentes palabras (coche = 0, gato=1), en un orden específico. Así, cuando en el texto aparezca por primera vez coche, entendemos que habrá un 0, y si en vez de coche apareciera gato, sería un 1. Seguiremos leyendo y poco a poco obtendremos el resto del mensaje oculto

Spammimic (EN), una herramienta muy antigua encargada de codificar mensajes en supuestos mensajes de spam (los típicos mensajes de spam que vemos en foros y blogs). Simplemente dices el mensaje a ocultar, y spammimic genera el texto de spam, que el emisor colocaría en una fuente conocida, y el receptor recuperaría para hacer el proceso contrario. Cualquier guardián que viera el mensaje solo vería un spam más, sin darse cuenta de que hay información oculta.

Esteganografía en sistemas operativos y ficheros: Consiste en ofuscar información apoyándose en las restricciones propias del canal utilizado.

Ejemplo

Ofuscar información utilizando ADS en NTFS. En NTFS un fichero está construido por varios flujos (uno para los permisos del archivo, otro para datos del usuario).

Herramientas como LADS o crucial ADS permiten recopilar este tipo de archivos.

Esteganografía en formato de ficheros: Ofuscamos información en las limitaciones propias del fichero, o de los elementos de control de los sistemas encargados de leer el fichero.

Ejemplo

La Técnica de Final de Fichero (EOF), que aprovecha el poco control a la hora de leer un fichero para ocultar información al final del mismo. La mayoría de sistemas solo leen la información necesaria, por lo que se puede agregar contenido adicional que pasará desapercibido.

Esteganografía hardware: Aprovechar, de nuevo, las limitaciones (o fallos de seguridad) de un elemento físico para ofuscar información dentro de él.

Ejemplo

El uso de malware en BIOS (BadBIOS) que sacudió internet entera hace relativamente poco, para disgusto de los amantes de la conspiración mundial. Y el BadUSB que básicamente sigue el mismo principio.

Esteganografía en tecnologías web: Aprovechar la propia arquitectura del lenguaje de maquetado para ofuscar información. Como bien sabe, HTML no distingue entre mayúsculas y minúsculas, por lo que para el navegador
 sería lo mismo que
, que
 o que
. Y ahí tenemos varias alternativas distintas que funcionan de la misma manera.

Esteganografía en protocolos de comunicación: Semejante a las técnicas aplicadas en el formato de ficheros, pero con el formato de los protocolos

Ejemplo

Oculta información en cabeceras UDP.

Esteganografía en contenido multimedia: Tanto en imágenes como en sonido o vídeo. Las técnicas aplicadas en esteganografía de imágenes son muy parecidas a las aplicadas en sonido (basadas normalmente en ocultar información de poco peso en elementos muy pequeños del sistema de archivos multimedia, por ejemplo a nivel de pixeles en imágenes). En vídeo encontramos técnicas cruzadas de los dos anteriores, con algunas nuevas en las que interfiere movimientos específicos (guardar información cuando una zona cambie bruscamente de tonalidad) o en ejes críticos (subidas de volumen o espacios que cumplan x particularidad). (Iglesias, 2015)

La Esteganografía en contraste con la criptografía, donde el enemigo permite ser detectado, interceptado y modificado sus mensajes si se logra violar el sistema de seguridad de la encriptación de dicho mensaje en lugar de solo tratar de escribir de manera que no sea fácilmente descifrable. EL acierto de la esteganografía radica justamente en ocultar los mensajes dentro otros mensajes de manera que no exista enemigo que detecte que hay un segundo mensaje y que se halla oculto. (Ardita , Caratti , Do Cabo, & Giusto , 1998)

1.2.9 SEGURIDAD

El concepto de seguridad en la información es mucho más amplio que la simple protección de los datos a nivel lógico. Para proporcionar una seguridad se debe tener en cuenta múltiples factores, tanto internos como externos. Se caracterizar el sistema que va a albergar la información para poder identificar las amenazas.

CLASIFICACION

Seguridad física.- Consiste en la "aplicación de barreras físicas y procedimientos de control, como medidas de prevención y contramedidas ante amenazas a los recursos e información confidencial". (HUERTA, 2002)

Seguridad de la información.- Es la preservación de su confidencialidad, integridad y disponibilidad, así como de los sistemas implicados en su tratamiento, dentro de una organización, frente a observadores no autorizados. Así pues, estos tres términos constituyen la base sobre la que se cimienta todo el edificio de la seguridad de la información. Para garantizar que la seguridad de la información es gestionada correctamente, se debe hacer uso de un proceso sistemático, documentado y conocido por toda la organización, desde un enfoque de riesgo empresarial. (Spohr, 2007)

Seguridad del canal de comunicación.- Aunque los usuarios de las computadoras que son extremos de una comunicación puedan estar tranquilos en cuanto a la seguridad de estas computadoras, la red de comunicación siempre es un punto de desconfianza. Los canales de comunicación rara vez se consideran seguros. Debido a que en la mayoría de los casos escapan a nuestro control, ya que pertenecen a terceros, resulta imposible asegurarse totalmente de que no están siendo escuchados o intervenidos. (Stallings, 1997)

Problemas de autentificación.- Debido a los problemas del canal de comunicación, es necesario asegurarse de que la información que recibimos en la computadora viene de quien realmente

creemos que viene, y que además no ha sido alterada. Para esto se suele emplear criptografía asimétrica en conjunción con funciones resumen (hash).

Problemas de suplantación.- En las redes existe el problema añadido de que cualquier usuario autorizado puede acceder al sistema desde fuera, en lo cual se confiara en sistemas fiables para garantizar que los usuarios no están siendo suplantados por intrusos. Para conseguir esto normalmente se emplean mecanismos basados en contraseñas.

No repudio.- Cuando se recibe un mensaje no solo es necesario poder identificar de forma unívoca al remitente, sino que este asuma todas las responsabilidades derivadas de la información que haya podido enviar. En este sentido es fundamental impedir que el emisor pueda repudiar un mensaje, es decir, negar su autoría sobre él.

Anonimato.- Es el concepto opuesto al del no repudio. En determinadas aplicaciones, como puede ser un proceso electoral o la simple denuncia de violaciones de los derechos humanos en entornos dictatoriales, es crucial garantizar el anonimato del ciudadano para poder preservar su intimidad y su libertad.

Autenticación

El concepto de autentificación viene asociado a la comprobación del origen e integridad de la información. En general, y debido a los diferentes tipos de situaciones que podemos encontrar en un sistema informático, distinguiremos tres tipos de autentificación:

Autentificación de mensaje. Queremos garantizar la procedencia de un mensaje conocido, de forma que podamos asegurarnos 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 frente al sistema de un dispositivo concreto. Este dispositivo puede estar solo o tratarse de una llave electrónica que sustituye a la contraseña para identificar a un usuario. (Lucerna López, 2010)

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.

1.3 SISTEMAS CRIPTOGRAFICOS

1.3.1 CONCEPTOS PREVIOS

Para poder hablar de los sistemas criptográficos debemos tener una noción del vocabulario que utilizaremos.

Texto Plano.- Es el texto que queremos proteger mediante el uso de técnicas criptográficas.

Criptograma.- Al texto una vez que ha sido transformado mediante alguna técnica criptográfica. Este texto resulta ilegible a no ser que se conozca la clave para volver a recuperar el "texto plano original**".**

Encriptación.- Al proceso que transforma un texto plano en un criptograma.

Desencriptación.- Al proceso que recupera el texto plano de un criptograma.

1.3.2 CLASIFICACIÓN

Los sistemas criptográficos los podemos clasificar según como encripten los diferentes símbolos del alfabeto que estén utilizando.

Monoalfabeticos.- Son aquellos en los que cada símbolo se encripta siempre con el mismo símbolo, la encriptación de cada símbolo es independiente del mensaje. Este tipo de sistemas son inseguros ya que se pueden romper mediante "análisis estadísticos de frecuencias".

Polialfabeticos.- Son aquellos en los que cada símbolo del alfabeto no se encripta con el mismo símbolo, la encriptación depende del mensaje. El primer requisito para que un sistema criptográfico sea seguro es que sea polialfabético. (Litwak & Escalante, 2004)

Numero de claves usadas.- Si tanto el emisor como el receptor usan la misma clave, el sistema se denomina simétrico, de clave única, de clave secreta o cifrado convencional. En cambio, si el emisor y el receptor usan cada uno claves diferentes, el sistema se denomina asimétrico, de dos claves o cifrado de clave pública

Tipo de operación utilizado para trasformar el texto claro en texto cifrado.-Todos los algoritmos de cifrado se basan en dos principios: sustitución y transposición. Lo fundamental es que todas las operaciones sean inversas (descifrar).

Forma de procesar el texto claro.- Un cifrado de bloques procesa un bloque de elementos cada vez, produciendo un bloque de salida por cada bloque de entrada. (UNAM, 2004)

En la fig. 6 Cuando se quiere mandar información confidencial se aplican técnicas criptográficas para poder así "esconder" el mensaje (cifrar o encriptar), luego se manda el mensaje por una supuesta línea de comunicación insegura y después sólo el receptor autorizado puede leer el mensaje "escondido" (descifrar o descencriptar).

Fig. 6 Comunicación Insegura

En el caso de la llamada "criptografía fuerte", es muy segura y es prácticamente imposible desencriptar mensajes encriptados con este tipo de criptografía.

La controversia surge ya que de este tipo de criptografía podría ser utilizado por organizaciones criminales para asegurar sus comunicaciones y de esta forma poder cometer sus actividades criminales de una forma más segura. Es por ello que hay interesados en que este tipo de criptografía no se utilice argumentando lo anterior, pero hay otro argumento también contundente que es el derecho a la intimidad.

Básicamente la criptografía se divide en dos ramas:

Criptografía de Clave Privada o Simétrica.

Criptografía de Clave Asimétrica o Pública.

Ahora bien, independientemente de si son simétricos o asimétricos, es si tiene o no estructura de grupo.

Un sistema criptográfico tiene estructura de grupo si $\mathrm{V}\ K1, K2 \in K\ 3\ k3 \in K$ tal que para cualquier $m \in M$ se verifique que:

$$E(E(m, K2), K1) = E(m, K3)$$

Es importante que un sistema criptográfico no tenga estructura de grupo, ya que si ciframos un mensaje con una clave K1 y luego ciframos un mensaje con otra clave K2 entonces aumentamos la seguridad del sistema. Mientras que si el sistema hubiera tenido estructura de grupo no

habríamos hecho nada, ya que existiría una clave K3 que realizaría la misma operación, con lo cual no habríamos incrementado la seguridad del sistema.

Son muchas las técnicas criptográficas existentes y cada cierto aparecen nuevos logros y algoritmos buscando la mayor simplicidad sin minimizar la robustez de los sistemas, además existe cierta controversia que tipo de criptografía se debe utilizar

1.4 GESTION DE CLAVES

Todas las técnicas criptográficas dependen en última instancia de una o varias claves, por lo que su gestión es de vital importancia. Esta tarea incluye básicamente:

La generación de las claves de forma que cumplan una serie de requisitos. Este proceso es dependiente del algoritmo en el que se va utilizar la clave en cuestión, aunque generalmente se emplea una fuente generadora de números pseudo-aleatorios como base para la creación de la clave (la clave debe ser lo más aleatoria posible).

Registro. Las claves se han de vincular a la entidad que las usará.

Su distribución a todas las entidades que las puedan necesitar.

Su protección contra la revelación o sustitución no autorizadas.

El suministro de mecanismos para informar a las entidades que las conocen en caso de que la seguridad de dichas claves haya sido comprometida (revocación).

El tipo de método empleado para llevar a cabo la gestión de las claves es diferente se gún el tipo de criptografía utilizada (simétrica o asimétrica).

Todas las claves tienen un tiempo determinado de vida, el criptoperiodo, para evitar que las técnicas de criptoanálisis tengan el suficiente tiempo e información para “romper” el algoritmo criptográfico asociado. (Cavalli, Guanca, & Juncos, 2002).

La Gestión de Claves se encuentra conformada por varios requerimiento para proteger la seguridad del ente o empresa, la generación de claves que cumpla una serie de acuerdos diferenciando según el tipo de criptografía utilizada.

1.5 DISTRIBUCIÓN SIMÉTRICA DE CLAVES SIMÉTRICAS

Estos métodos suelen tener tres tipos de claves:

De sesión, empleadas para encriptar los datos de usuario y que son actualizadas frecuentemente,

De encriptación de claves, para proteger las claves de sesión,

Maestras, que son distribuidas manualmente y se utilizan para proteger las de encriptación de claves cuando son intercambiadas.

La distribución manual de las claves maestras es el princi pal inconveniente de la gestión mediante técnicas simétricas, por lo que la mayoría de los sistemas utilizan métodos asimétricos para llevar a cabo dicha distribución.

"Este tipo de cifrado es muy fácil de usar. Además, es muy útil para el cifrado de archivos de datos personales, ya que sólo se requiere de una clave. Esta forma de cifrado también se puede utilizar para ayudar a prevenir riesgos en la seguridad"

1.6 ACUERDO DE DISTRIBUCION

Este método, creado por W. Diffie y M. Hellman, permite que dos entidades acuerden una clave simétrica sin necesidad de comunicación previa. El algoritmo propuesto se basa en emplear una función unidireccional con trampa (trapdoor one-way function).

Una función $y = f(x)$ se denomina unidireccional con trampa si:

A cada valor de x le corresponde una y.

Dado un valor de x es fácil hallar y.

Dado un valor de y es fácil calcular x si se dispone de una cierta información (clave) y difícil sin el conocimiento de dicha información.

La función empleada por Diffie-Hellman es la exponencial discreta, cuya inversa, el algoritmo discreto es difícil de calcular.

1.7 DISTRIBUCION ASIMETRICA DE CLAVES SIMETRICAS

Como ya se ha comentado en capítulos anteriores, el servicio de confidencialidad se puede proporcionar utilizando técnicas simétricas o asimétricas, pero estas últimas suponen una mayor carga computacional. Por ello, los métodos que se emplean comúnmente son una combinación de ambas categorías criptográficas.

Generalmente, el proceso se realiza en los siguientes pasos:

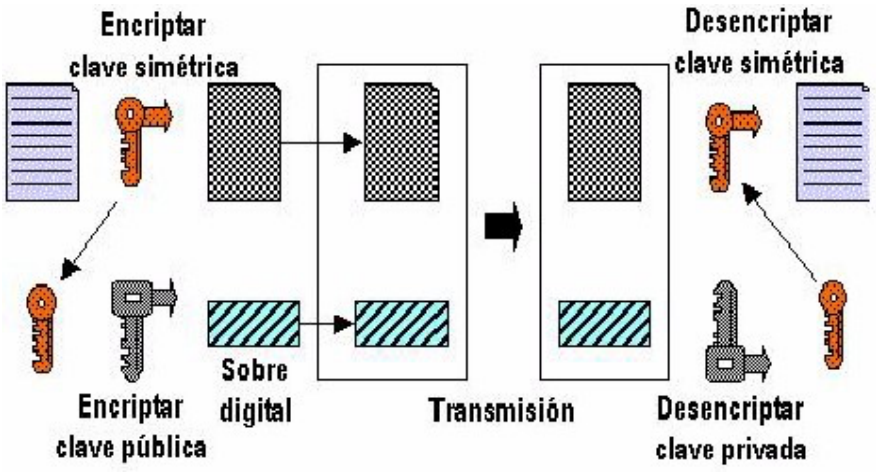


Fig. 7 Procesos Criptográficos

En la Fig.7 vemos los pasos que se deben seguir para realizar el proceso criptográfico.

Se encripta el texto en claro con una clave simétrica para obtener el texto cifrado.

La clave simétrica sé encripta con la clave pública del receptor obteniendo así el denominado sobre digital.

Se envía el texto cifrado y el sobre digital.

En recepción, se desencriptar la clave simétrica con la clave privada del receptor.

Sé desencriptar el texto cifrado con la clave simétrica obtenida en el paso anterior para obtener el texto claro original.

2 CRIPTOGRAFIA SIMETRICA

Son el conjunto de métodos que nos permiten una comunicación segura entre las partes que lo componen, siempre y cuando previamente se haya intercambiado una clave que la llamaremos **clave simétrica**, es decir la simetría hace referencia a que las partes usan la misma llave para cifrar como para descifrar, es por ello que la robustez del algoritmo recae en mantener el secreto de la misma, la rapidez y facilidad de implementación es la principal característica de este método de encriptación.

A este tipo de criptografía se la conoce como criptografía de clave privada o de llave privada.

Fig. 8 Tipos de cifrado

Esta criptografía de llave privada puede ser clasificada en tres grandes grupos:

Criptografía simétrica de lluvia.

Criptografía simétrica de bloques.

Criptografía simétrica de resumen.

Esta criptografía ha sido la más utilizada a lo largo de toda la historia, esta se puede implementar en distintos dispositivos, manuales, mecánicos, eléctricos así también como en algoritmos

programables en cualquier tipo de computadora, la idea general es aplicar diferentes funciones al mensaje a cifrar de forma que solo conociendo la llave o clave esta se pueda descifrar.

Los sistemas implementados con este tipo de encriptación son más rápidos que los de la clave pública y resultan apropiados para el cifrado de grandes volúmenes de datos.

La desventaja que estos sistemas presentan es que el emisor y el receptor comparten la clave, por lo cual se hace inseguro el envío y recepción de la clave ya que de cualquier manera que esta se envié, es posible que alguien la intercepte. (Sevilla, 2008)

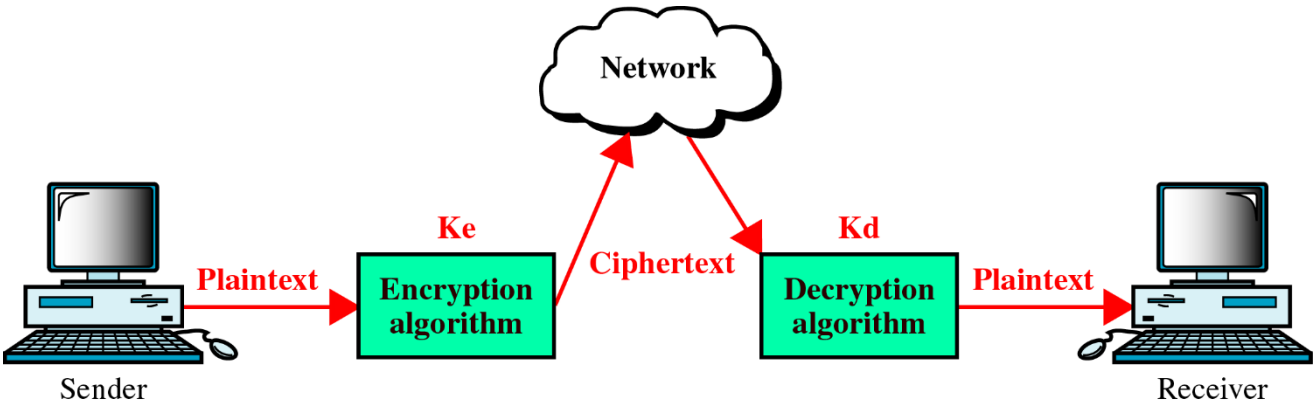


Fig. 9 Esquema de encriptación

Este sistema de encriptación se utiliza para encriptar el cuerpo de los mensajes en el correo electrónico o los datos intercambiados en las comunicaciones digitales

2.1 CIFRADO DE FLUJO

El emisor A ,con una clave secreta y un algoritmo determinístico (RKG),genera una secuencia binaria(s) cuyos elementos se suman módulo 2 con los correspondientes bits de texto plano m, dando lugar a los bits del texto cifrado c , esta secuencia (c)es la que se envía a través de un medio de trasmisión ,en el receptor B con la misma clave proporcionada y el mismo algoritmo determinístico ,genera la misma secuencia cifrante (s),que se suma módulo 2 con la secuencia cifrada (c)dando lugar a los bits del texto plano.

Los tamaños de las claves oscilan entre 120 y 250 bits (Noelia Litwak, 2004)

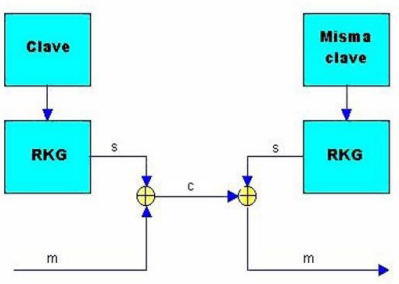


Fig. 10 Algoritmo de cifrado de flujo

2.2 CIFRADO DE BLOQUE

El cifrado de bloque está constituido por cuatro elementos:

Transformación inicial por permutación.

Una función criptográfica débil (no compleja) iterada n veces o vueltas.

Transformación final para que las operaciones de encriptación y desencriptación sean simétricas.

El uso de un algoritmo de expansión de claves que tiene como objeto verter la clave de usuario, normalmente de longitud limitada entre 32 y 256 bits, en un conjunto de subclaves que puedan estar constituidas por varios cientos de bits en total.

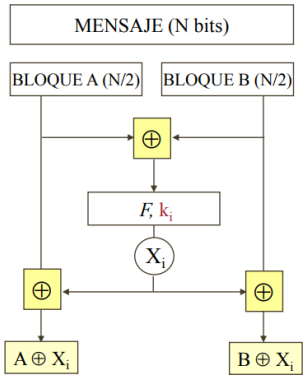


Fig. 11 Algoritmo de cifrado de bloque

Cada bloque se divide en dos partes iguales que se combinan con XOR.

El resultado se cifra usando una clave y una función de un solo sentido (muy difícil de invertir sin conocer la clave)

El cifrado se combina de nuevo mediante XOR con los dos bloques originales.

2.3 CIFRADO DE CESAR

Este sistema de encriptación fue utilizado por Julio Cesar y es un caso muy particular de una serie de sistemas de encriptación conocidos como “VIGENERE”. Este sistema de encriptación es completamente inseguro, pero útil para explicar los fundamentos de los métodos criptográficos,

suponiendo q tenemos un alfabeto con N elementos .el método consiste en asignar un número a cada símbolo del alfabeto A=0, B=1,….Z=N-1.

Para realizar la encriptación a cada letra le sumaremos un número, menor estrictamente, que N modulo N y le haremos corresponder a cada letra asociada.

A→0 y A+4=4 →4≈4 mod 26, 4→D

Si para encriptar sumamos 4 mod N, entonces para desencriptar restamos 3 mod N , siendo N el número de símbolos de nuestro alfabeto.

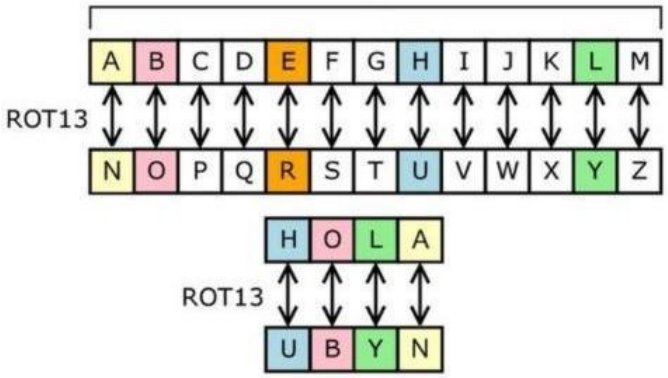


Fig. 12 Cifrado de desplazamiento (Cesar)

Este sistema es simétrico, las claves para encriptar y desencriptar deben mantenerse en secreto, ya que conocer una de ellas implica conocer la otra. Este sistema es Mono alfabético, sensible ataques por análisis estadístico de frecuencias, además posee una estructura de grupo, ya que si primero desplazamos m unidades cada letra y luego lo hacemos N unidades es como si hubiésemos desplazado M+N unidades desde el principio. (Noelia Litwak, 2004)

2.4 CIFRADO DE FEISTEL

Se denominan así a los criptosistemas en los que el bloque de datos se fracciona en dos mitades y en cada vuelta de encriptación se trabaja alternativamente con una de las dos mitades.

Dado un bloque de N bits (64 por lo general), este se dividirá en dos mitades.

Existirá una función indirecta F (difícil de invertir).

Esta realiza operaciones con la clave KI solo con la mitad del bloque, y se permutan en cada vuelta as dos mitades, esta operación se repite durante N vueltas (Noelia Litwak, 2004)

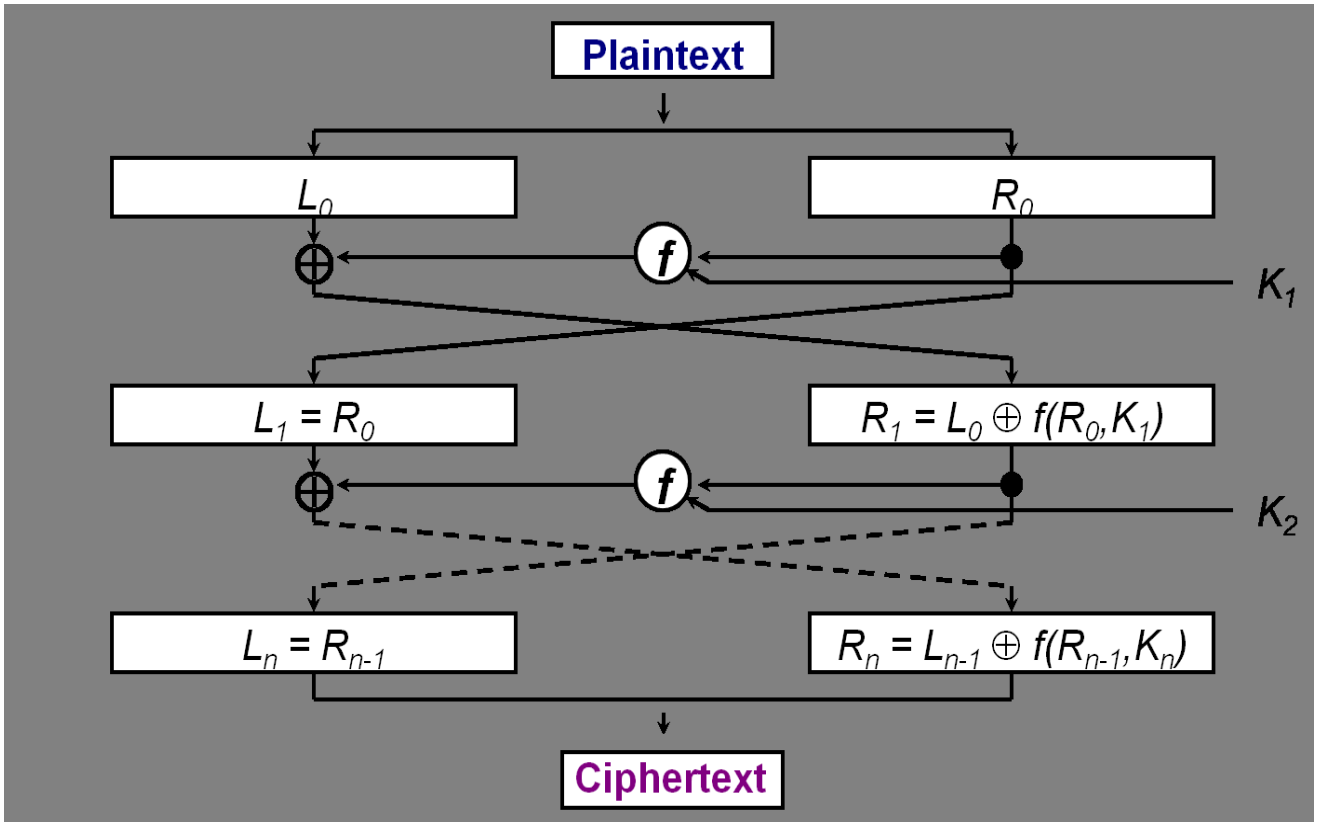


Fig. 13 Algoritmo de Feistel

2.4.1 DES "DATA ENCRYPTION STANDARDAR

2.4.2 HISTORIA

El primer sistema criptográfico del que se tiene constancia es la Escítala. Este sistema data del siglo V a.c. y era usado en Esparta. Los primeros sistemas de cifrado estuvieron ligados a campañas militares dadas la necesidad de evitar que el enemigo obtuviese los movimientos de las tropas al interceptar mensajes. El cifrado de César es un cifrado de sustitución monoalfabética. Ya en el siglo XV es inventado un sistema de sustitución polialfabética por León Battista Alberti.

En el siglo XX, a consecuencia de las dos guerras mundiales, la criptografía sufre un gran avance. En el año 1920 comenzó a usarse la máquina enigma. Su fama se debe a su uso por parte del ejército Alemán. Con los avances de la tecnología, entre otros, se desarrolla el algoritmo DES (Data Encryption Standard) es un algoritmo de cifrado desarrollado por la NSA a petición del gobierno de EEUU bajo la presión de las empresas por la necesidad de un método para proteger sus comunicaciones. DES fue escogido como FIPS (Federal Information processing Standard) en el año 1976 y su uso se extendió por todo el mundo. Hoy en día DES es considerado inseguro dada

su clave de 56 bits, insuficiente frente al poder computacional actual. En su variante Triple DES el algoritmo se cree seguro. (Corrales Sánchez, Cilleruelo Rodríguez, & Cilleruelo Rodríguez, s.f.)

2.4.3 DESCRIPCIÓN DEL ALGORITMO

DES(Data Encryption Standard, estándar de cifrado de datos) es un algoritmo desarrollado originalmente por IBM a requerimiento del NBS (Oficina Nacional de Estandarización, en la actualidad denominado NIST, 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. El nombre original del algoritmo, tal como lo denominó IBM, era Lucifer.

Trabajaba sobre bloques de 128 bits, teniendo la clave igual longitud. Se basaba en operaciones lógicas booleanas y podía ser implementado fácilmente, tanto en software como en hardware. Tras las modificaciones introducidas por el NBS, consistentes básicamente en la reducción de la longitud de clave y de los bloques, DES cifra bloques de 64 bits, mediante permutación y sustitución y usando una clave de 64 bits, de los que 8 son de paridad (esto es, en realidad usa 56 bits), produciendo así 64 bits cifrados. (Sánchez Arriazu, 1999)

2.4.4 PRINCIPIOS DEL DISEÑO

DES cifra bloques de 64 bits, mediante permutación y sustitución y usando una clave de 64 bits, de los que 8 son de paridad (esto es, en realidad usa 56 bits), produciendo así 64 bits cifrados.

DES tiene 19 etapas diferentes.

La primera etapa es una transposición, una permutación inicial (IP) del texto plano de 64 bits, independientemente de la clave. La última etapa es otra transposición (IP-1), exactamente la inversa de la primera. La penúltima etapa intercambia los 32 bits de la izquierda y los 32 de la derecha. Las 16 etapas restantes son una Red de Feistel de 16 rondas.

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.

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. Ki sederiva de la elección permutada (PC-2) de 48 de los 56 bits de estas dos mitades rotadas. 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).

Para descifrar basta con usar el mismo algoritmo empleando las Ki en orden inverso. Está descrito oficialmente en FIPS PUB 46. (SEVILLA, 2008)

2.4.5 PROPIEDADES

BlockSize Obtiene o establece el tamaño del bloque de la operación criptográfica en bits.

FeedbackSize Obtiene o establece el tamaño de respuesta de la operación criptográfica en bits.

Obtiene o establece el vector de inicialización (IV) del algoritmo simétrico.

Key Obtiene o establece la clave secreta del algoritmo simétrico.

KeySize Obtiene o establece el tamaño de la clave secreta usada por el algoritmo simétrico en bits.

LegalBlockSizes Obtiene los tamaños de bloque, en bits, admitidos por el algoritmo simétrico.

LegalKeySizesObtiene los tamaños de clave, en bits, admitidos por el algoritmo simétrico.

ModeObtiene o establece el modo de funcionamiento del algoritmo simétrico.

Padding Obtiene o establece el modo de relleno usado en el algoritmo simétrico.

2.4.6 MODO DE OPERACIÓN

ECB-Electronic Code Book Mode:

ECB ha sido estandarizado por el NIST (U.S. National Institute for Standards and Technology). Este modo de cifrado es el más simple de todos, pues se limita a partir el mensaje en bloques y cifrarlos por separado.

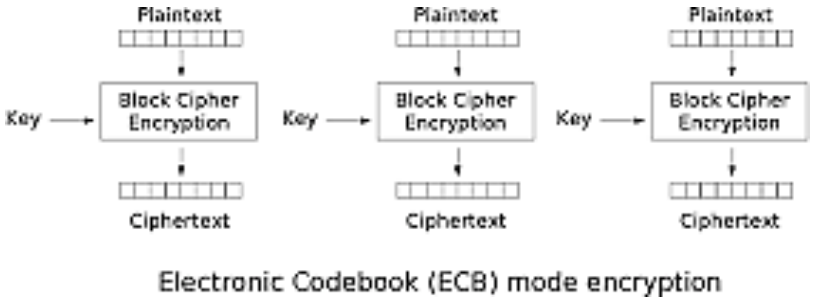


Fig. 14 Cifrado de bloque por separado

Entre las ventajas de este método destaca la posibilidad de romper el mensaje en bloques y cifrarlos en paralelo o el acceso aleatorio a diferentes bloques.

Sin embargo, las desventajas de este modo de cifrado son enormes, por lo que se usa cada vez menos. El hecho de cifrar los bloques por separado implica que cuando se cifre un bloque con cierto valor, siempre se obtendrá el mismo resultado. Esto hace posible los ataques de diccionario.

Además, cuando se cifran varios bloques y se envían por un canal inseguro, es posible que un adversario elimine ciertos bloques sin ser detectado, o que capture algunos bloques y los reenvíe más adelante.

DES al ir cambiando constantemente por las diferentes entidades de estandarización, este trabaja sobre bloques de 128 bits con su clave simétrica de misma longitud y a la vez puede utilizarse con bloques de 64 bits mediante permutación y sustitución y usando una clave de 64 bits, de los que 8 son de paridad lo son 56 bits los cuales nos ofrecen 64 bits DES crea una función de feistel mas permutaciones de expansión y transposiciones las cuales mediante una clave nos permite cifrar bloques mediante 16 iteraciones

2.4.7 TIPOS DE ATAQUES Y VULNERABILIDADES

Existen tres ataques conocidos que pueden romper las dieciséis rondas completas de DES con menos complejidad que un ataque por fuerza bruta: el criptoanálisis diferencial (CD), el

criptoanálisis lineal (CL) y el ataque de Davies. De todas maneras, estos ataques son sólo teóricos y no es posible llevarlos a la práctica.

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 247 textos planos escogidos. DES fue diseñado para ser resistente al CD. El criptoanálisis lineal fue descubierto por Mitsuru Matsui, y necesita 243 textos planos escogidos (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, 241 en lugar de 243).

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 239 – 241 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 y Biryukov (1997). La forma Data Encryption Standard - DES más potente del ataque requiere 250 textos planos escogidos 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 cuántas rondas son necesarias para conseguir seguridad, y cuánto «margen de seguridad» proporciona la versión completa (DES, s.f.)

DES al ser atacado mediante criptoanálisis lineal de textos planos escogidos determinaron la complejidad real del criptoanálisis lineal, y descubrió que era algo más rápido de lo predicho, requiriendo un tiempo equivalente a 239–241 comprobaciones en DES.DES más potente del ataque requiere 250 textos planos escogidos y tiene un 51% de probabilidad de éxito ,este ha tenido que crear nuevos algoritmos los cuales le permitan tener un mayor nivel de cifrado creando así criptosistemas los cuales dan origen al des doble y triple los cuales son realizar encriptación

sobre encriptación con diferentes claves de seguridad por nivel doble y triple los cuales aumentan su complejidad y la dificultad de obtener las llaves de cada nivel.

2.4.8 DES

Des doble y des triple

Un posible modo de incrementar de modo efectivo el tamaño de la clave de DES sería realizar una doble operación de cifrado sobre el texto plano. Tomando el texto plano, podríamos obtener el criptograma a través de la doble transformación. Y ahora para el atacante la tarea es enormemente mayor: encontrar ambas claves que juntamente llegan a nuestra transformación. ¿Aumentaría eso la seguridad de nuestro criptosistema?

En realidad, para muchos criptosistemas, aplicar dos veces el proceso de cifrado con dos claves diferentes es equivalente a realizar el proceso de cifrado con una tercera clave diferente. ¿Es así para DES?: Dadas las claves, existe una clave. Esta cuestión se plantea frecuentemente de la siguiente forma: ¿Es DES un grupo?, o también ¿Es DES una transformación cerrada bajo la operación de composición?

La respuesta es que NO. Afortunadamente no es así, en el caso de DES: la composición de dos cifrados DES no es equivalente a un cifrado DES con una tercera clave en principio distinta a las otras dos.

Existe una forma de ataque que anula esa posibilidad. Para levarla a cabo basta con tener la herramienta que rompe DES y una enorme capacidad de memoria. El procedimiento para atacar esta cifra es el siguiente:

1. Supóngase que el atacante ha interceptado el mensaje plano a transmitir, ^{W} , y el criptograma.

2. Computa y almacena en memoria todos los valores para todos los posibles valores

3. Computa y almacena en memoria todos los valores para todos los posibles valores

4. Compara ambas listas en busca de encontrar cadenas de texto iguales.

Entre las claves que han conducido a estas cadenas coincidentes están las dos claves buscadas.

Por tanto, el atacante ha de probar todas las claves para el paso n. 2, y ha de probar todas las claves para el paso n. 3. Por tanto, si es el número total de claves posibles, al realizar el proceso

de cifrado dos veces deberá realizar una búsqueda de claves, y no de como habíamos esperado al principio.

A este ataque se le ha dado el nombre “meet–in–the–middle attack”, o ataque de “encuentro–en–la–mitad”.

Triple des

Doble–DES queda por tanto descartado como criptosistema seguro una vez se ha logrado comprometer el criptosistema DES. ¿Qué ocurre en el caso de Triple–DES? En ese caso, el algoritmo de ataque meet–in–the–middle no funciona correctamente.

Existen dos formas habituales de aplicar Triple–DES:

1. Con tres claves y, aplicamos la transformación

2. Con dos claves, aplicamos la transformación. En este caso, evidentemente, cuando el algoritmo Triple–DES se reduce al DES tradicional.

Ambas formas dan gozan de una seguridad equivalente a la de un criptosistema de longitud de clave igual a 112 bits. Ambas formas son resistentes, como ya se ha dicho, al ataque meet–in–the–middle.

Existen, sin embargo, otras formas de Data Encryption Standard - DES ataque definidas, de factura similar a la presentada en este algoritmo, pero requieren una cantidad de memoria tal que la hacen impracticable. Queda indicada, por tanto, la posible debilidad, que no supone hoy por hoy una amenaza viable. Rivest tiene presentada otra vía para reforzar DES. Tomando tres claves, realizamos la transformación: Modificar el texto plano con la operación XOR con la clave; al resultado aplicar la transformación DES con la clave, y operar el resultado obtenido de nuevo con el operador XOR y ahora la clave. Este método se conoce como DESX y se ha mostrado también suficientemente seguro.

2.5 AES "ADVANCED ENCRYPTION SATANDARD"

2.5.1 HISTORIA

Advanced Encrytion Estándar (AES) es un cifrado para datos electrónicos, este a sido adoptado a nivel mundial por su gran fiabilidad y seguridad. Este cifrado es el medio aceptado para cifrar la

información digital, incluyendo datos financieros, de telecomunicaciones, gubernamentales y militares.

En octubre de 2000 el instituto internacional de estándar y tecnología “NIST” anunciaba oficialmente la adopción del algoritmo Rijndael como nuevo Estándar Avanzado de Cifrado (AES) para su empleo en aplicaciones criptográficas no militares, culminando así un proceso de más de tres años, encaminado a proporcionar a la comunidad internacional un nuevo algoritmo de cifrado potente, eficiente, y fácil de implementar. DES tenıa por fin un sucesor.

AES se convirtió rápidamente en un estándar universal para todas las plataformas, gracias a su alto nivel de seguridad y rendimiento, el cual lo hace adecuado para la encriptación mediante software. (software, 2018)

2.5.2 CONCEPTOS RIJNDAEL

Es un acrónimo formado por los nombres de sus dos autores, los belgas Joan Daemen y Vincent Rijmen. Su interés radica en que todo el proceso de selección, revisión y estudio tanto de este algoritmo como de los restantes candidatos, se ha efectuado de forma pública y abierta, por lo que, prácticamente por primera vez, toda la comunidad criptográfica mundial ha participado en su análisis, lo cual ´ convierte a Rijndael en un algoritmo perfectamente digno de la confianza de todos. (Pousa., 2013)

2.5.3 OPERACIONES BÁSICAS DEL AES

AES toma como elemento básico al byte (8 bits) y ve a los bytes como elementos del campo finito de Galois o GF(28), toda operación del algoritmo está basada en operaciones sobre este campo finito, rotaciones de bytes y operaciones de suma módulo 2.

El AES permite un bloque de cifrado de tamaño variable de 128, 192, 256 bits asociado a claves de los mismos tamaños. Es decir

3.4 x 10^38 posibles claves para 128 bits

6.2 x 10^57 posibles claves para 192 bits

1.1 x 10^77 posibles claves para 256 bits

Asumiendo que una máquina recupera una clave DES en un segundo (probando 7.2 x 10^16 claves por segundo), tardaría aproximadamente 149000 de billones de años para romper una llave 128 bits AES. (software, 2018)

2.5.4 CRITERIOS Y FUNDAMENTOS DE DISEÑO PARA EL AES

AES toma como elemento básico al byte (8 bits) y ve a los bytes como elementos del campo finito de Galois o GF(28), toda operación del algoritmo está basada en operaciones sobre este campo finito, rotaciones de bytes y operaciones de suma módulo 2.

El AES permite un bloque de cifrado de tamaño variable de 128, 192, 256 bits asociado a claves de los mismos tamaños. Es decir

3.4 x 10^38 posibles claves para 128 bits

6.2 x 10^57 posibles claves para 192 bits

1.1 x 10^77 posibles claves para 256 bits

Asumiendo que una máquina recupera una clave DES en un segundo (probando 7.2 x 10^16 claves por segundo), tardaría aproximadamente 149000 de billones de años para romper una llave 128 bits AES. (software, 2018)

2.5.5 DESCRIPCIÓN DEL ALGORITMO

Como ya se ha dicho, el AES es un cifrado en bloque de criptografía simétrica, es decir, trabaja cifrando y descifrando bloque a bloque, utilizando la misma clave privada para ambos procesos. En el estándar, el algoritmo Rijndael divide los datos de entrada en bloques de 4 palabras de 32 bits, es decir, $4 \times 32 = 128$ bits. Es necesario decir que el algoritmo Rijndael puede trabajar también con bloques mayores de 192 y 256 bits, pero no vienen contemplados en el estándar.

Al bloque de datos se le conoce como Estado. En cuanto a la longitud de clave, el estándar trabaja con longitudes de N k palabras de 32 bits, donde N k = 4, 6 ´o 8. Es decir, que el algoritmo trabaja con longitudes de clave de 128 (4×32), 192 (6×32) ´o 256 (8×32) bits. (Ocon, 2013)

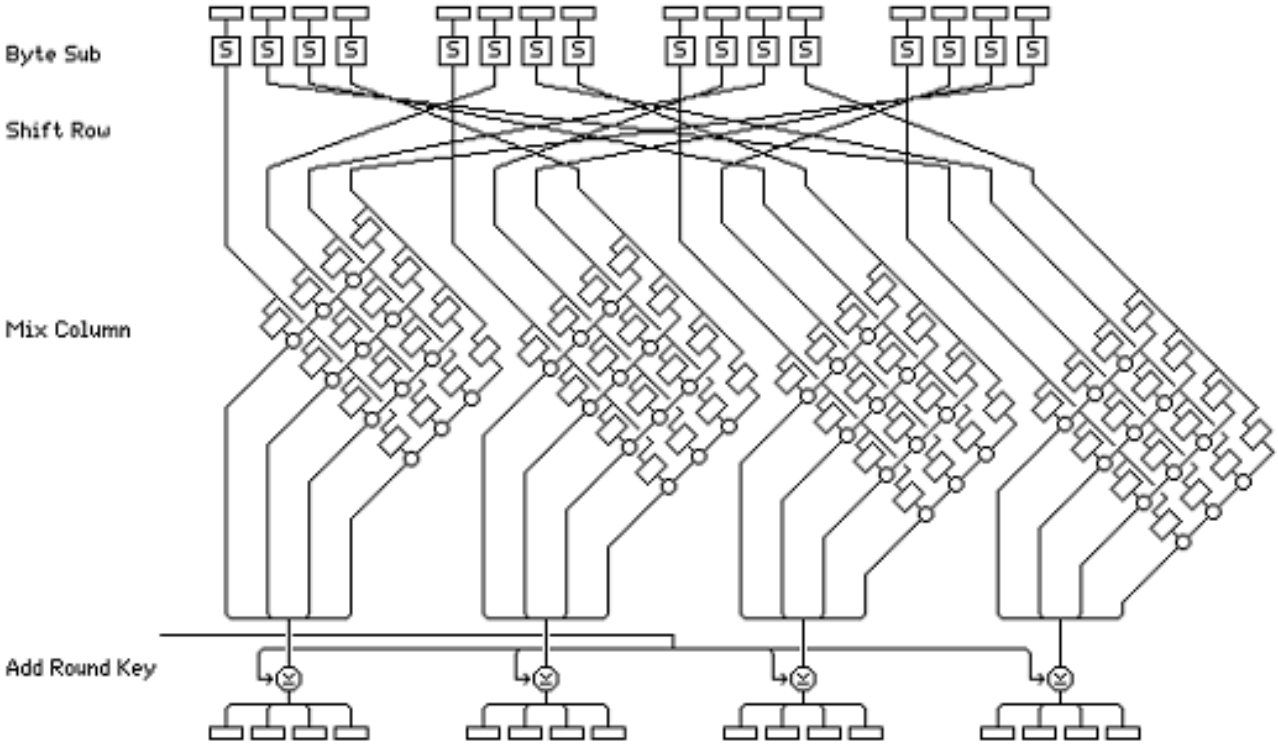


Fig. 15 Cifrado AES

2.5.6 ALGORITMO EQUIVALENTE PARA EL DESCIFRADO

Para definir el proceso de cifrado del AES, vamos a asumir que la longitud de clave escogida es de 128 bits, ya que la longitud de clave no afecta a las diversas operaciones que realiza el cifrado. Con el objetivo de facilitar el entendimiento de las operaciones realizadas por el cifrado, se realizaran pequeños ejemplos. Básicamente, el cifrado aplica al Estado cuatro operaciones durante un número determinado de rondas. Dicho número de rondas (Nr) viene definido por la longitud de clave utilizada, siendo Nr = 10 para una longitud de clave de 128 bits, Nr = 12 para 192 bits y Nr = 14 para 256 bits. Las cuatro operaciones realizadas en el cifrado son denominadas:

SubBytes.

ShiftRows.

MixColumns.

AddRoundKey.

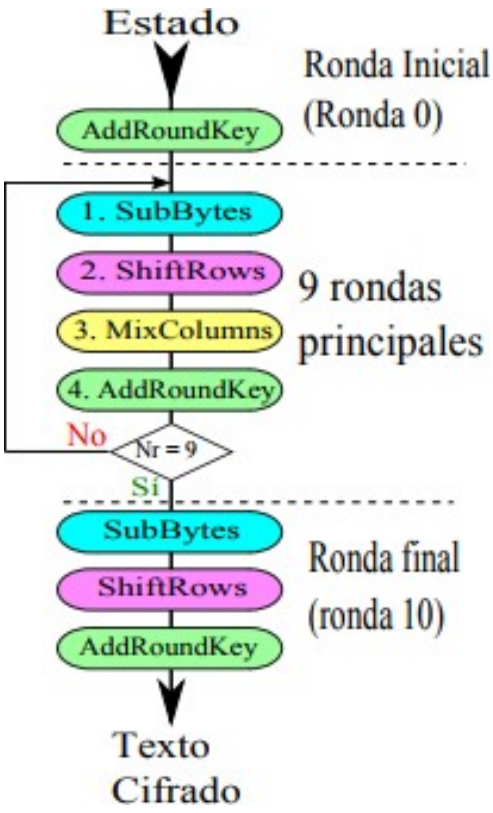


Fig. 16 Algoritmo de encriptación AES

Como vemos, en primer lugar, se realiza una ronda inicial en la que únicamente se aplica una operación AddRoundKey.

Posteriormente, se realizan las nueve rondas principales, en las que se aplican las cuatro operaciones del cifrado en este orden: SubBytes, ShitfRows, MixColumns y AddRoundKey. Por último, se realiza la ronda final, en la que se aplican las operaciones SubBytes, ShiftRows y AddRoundKey, obteniendo así nuestro texto cifrado. Cabe destacar que en cada ronda se utilizan diferentes subclaves, derivadas de la clave original. De modo que en la ronda inicial se utiliza la clave original (si la clave es de 128 bits) y en la ronda final se utiliza la subclave número 10.

2.5.7 IMPLEMENTACIÓN EN MICROS DE 8 BITS ("SMART CARDS") Y 32 BITS (PCS)

Es el corazón de Twofish. La palabra de entrada X es dividida en 4 bytes,cada uno es pasado por su propio S-Box que tomando 8 bits de entrada produce 8 de salida. Los 4 resultados son interpretados como un vector de longitud 4 sobre un campo GF (28) y multiplicado por una matriz de 4 por 4 MDS.

En este proceso contamos con una función adicional llamada h que toma 2 palabras X de 32 bits y una lista L= (L0,..., Lk-1) de palabras de 32 bits y produce una palabra de salida. Esta operación se realiza en k pasos, donde en cada uno los 4 bytes de entrada son pasados a través de un S-Box fijo y XOR a dos con un byte derivado de la lista. Finalmente estos bytes son pasados por un S-Box nuevamente y multiplicados por una matriz MDS como en la función g de modo que se reutilizan las primitivas. Podemos definir ahora los S-Boxes de la función g como g(X) = h(X, S) esto es para i = 0, 3. Las S-Boxes dependientes de las claves Si son formadas mediante un mapeo de Xi a Yi en la función h y donde L es igual al vector S derivado de las claves. Finalmente arribamos a las claves

expandidas Kj usando la función h
38

2.5.8 MODOS DE OPERACIÓN

SubBytes

La operación SubBytes consiste en una sustitución no lineal de bytes. Dicha sustitución se realiza aplicando la fórmula:

$$S'_{i,j} = M \cdot S_{i,j}^{-1} + C$$

$$\text{Donde } M = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}, C = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

Fig. 17 Sustitución no lineal

S^{-1}= Multiplicativo inverso del byte a transformar, con el bit menos significativo arriba.

S' = Transformación SubBytes, con el bit menos significativo arriba.

Existe una tabla de sustitución fija, llamada S-box, que aplica estas operaciones y permite realizar la operación SubBytes mediante un simple vistazo

Tabla 1 Sustitución Fija

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1x	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2x	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3x	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4x	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5x	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6x	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7x	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8x	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9x	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
Ax	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
Bx	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
Cx	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
Dx	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
Ex	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
Fx	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

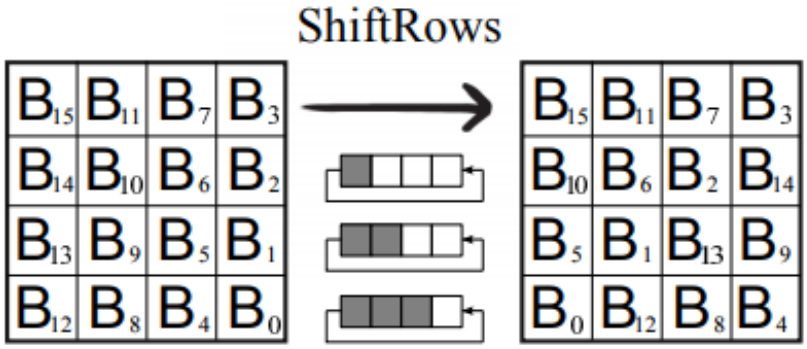
Entonces, si necesitamos realizar la operación SubBytes al número {19}, no tenemos más que mirar la tabla, fijándonos en la fila 1x y la columna x9, obteniendo así que la transformación de {19} es {D4}.

ShiftRows

La operación ShiftRows consiste en una rotación cíclica hacia la izquierda de las filas de la notación matricial del Estado, de manera que la primera fila permanece igual, la segunda fila se rota hacia la izquierda una posición, la tercera fila se rota hacia la izquierda dos posiciones y, por

último, la cuarta fila se rota hacia la izquierda tres posiciones.

Fig. 18 Desplazamiento de bloques



MixColumns

En la operación MixColumns, los cuatro bytes de cada columna de la notación matricial del Estado se combinan utilizando una transformación lineal invertible, como establece [NIS01]. Cada columna se trata como un polinomio y luego se multiplica el modulo $\mathrm{X}^4 + 1$ con un polinomio fijo $a(x) = \{03\}\mathrm{X}^3 + \{01\}\mathrm{X}^2 + \{01\}\mathrm{X} + \{02\}$

Es más sencillo verlo como una multiplicación matricial, donde el Estado siempre se multiplica a la derecha de la misma matriz:

$$\begin{pmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{pmatrix}$$

$$\text{Es decir } \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} B15 & B11 & B7 & B3 \\ B14 & B10 & B6 & B2 \\ B13 & B9 & B5 & B1 \\ B12 & B8 & B4 & B0 \end{pmatrix}$$

Multiplicando de esta manera, siempre teniendo en cuenta los aspectos matemáticos que hemos comentado la suma y multiplicación de bytes, obtendremos el resultado de la operación MixColumns. Dentro del AES, este es, sin duda, uno de los procesos más costosos en cuanto a cantidad de operaciones realizadas. (Ocon, 2013)

AddRoundKey

La operación AddRoundKey consiste en la combinación de la subclave de ronda correspondiente con el Estado. Esta combinación se realiza a través de la operación XOR. En la que representa el proceso de cifrado (3.2) se observa que en la ronda inicial (ronda 0) se realiza esta operación. En

el caso que estamos estudiando, longitud de clave de 128 bits, la subclave de ronda 0 es la propia clave de cifrado. Aunque, es importante señalar que, en el caso de tener claves de 192 y 256 bits, su correspondiente subclave de ronda 0 no es, lógicamente, la clave original, sino que será un fragmento de 128 bits de ´esta, en concreto sus 128 bits más significativos.

2.6 IDEA "INTERNATIONAL DATA ENCRYTION ALGORIRHM"

IDEA pertenece a una clase de criptosistemas de clave secreta que se caracteriza por la simetría de los procesos de cifrado y descifrado, y la posibilidad de implicar la clave de descifrado de la clave de cifrado y viceversa. IDEA toma 64 bits texto sin cifrar y produce salidas de texto cifrado de 64 bits usando una clave de 128 bits. La filosofía de diseño detrás de IDEA es combinar operaciones de diferentes grupos algebraicos incluyendo XOR, módulo adicional 216, y módulo de multiplicación Fermat prime 216+ 1. Todas estas operaciones funcionan en Sub-bloques de 16 bits

Se trata de una serie de complicadas operaciones no lineales. La clave debería ser lo más corta posible pues una clave larga implica un diseño más trabajoso. Considerar la performance en cada etapa del diseño es una buena práctica. El análisis no es una etapa superable, siempre debe hacer procesos de evaluación

3 CRIPTOGRAFIA ASIMETRICA

3.1 ESQUEMA DE CIFRADO PÚBLICO

Esta categoría incluye un conjunto de algoritmos criptográficos que utilizan dos claves distintas para cifrar y para descifrar el mensaje.

Ambas claves tienen una relación matemática entre sí, pero la seguridad de esta técnica se basa en que el conocimiento de una de las claves no permite descubrir cuál es la otra clave. En realidad sería necesario conocer todos los números primos grandes para ser capaz de deducir una clave a partir de otra, pero está demostrado que en la práctica se tardarían demasiados años sólo en el proceso de obtención de los número primos grandes.

Cada usuario cuenta con una pareja de claves, una la mantiene en secreto y se denomina clave privada y otra la distribuye libremente y se denomina clave pública. Para enviar un mensaje confidencial sólo hace falta conocer la clave pública del destinatario y cifrar en mensaje utilizando dicha clave. En este caso los algoritmos asimétricos garantizan que el mensaje original sólo puede volver a recuperarse utilizando la clave privada del destinatario (ver el esquema de la Figura 17).

Dado que la clave privada se mantiene en secreto, sólo el destinatario podrá descifrar el mensaje (Hellman:, Nov. 1976, pp: 644-654.)

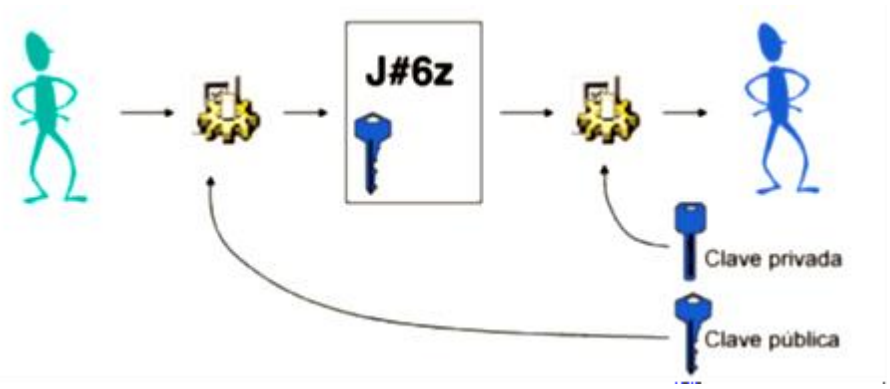


Fig. 19 Esquema de los algoritmos de clave pública

3.2 FUNDAMENTOS GENERICOS DE CIFRADOS ASIMETRICOS

Estos algoritmos pueden trabajar indistintamente con cualquiera de las claves, de manera que un mensaje cifrado con la clave pública sólo puede descifrarse con la clave privada, pero cualquier mensaje cifrado con la clave privada sólo puede ser descifrado con la clave pública. Esta característica permite utilizar este método para otras aplicaciones además de las que sólo requieren confidencialidad, como es el caso de la firma electrónica (como se verá en el artículo de criptografía del próximo número de Anales). Algunas de las características más destacadas de este tipo de algoritmos son las siguientes:

• Se utilizan una pareja de claves denominadas clave pública y clave privada, pero a partir de la clave pública no es posible descubrir la clave privada

• A partir del mensaje cifrado no se puede obtener el mensaje original, aunque se conozcan todos los detalles del algoritmo criptográfico utilizado y aunque se conozca la clave pública utilizada para cifrarlo.

• Emisor y receptor no requieren establecer ningún acuerdo sobre la clave a utilizar. El emisor se limita a obtener una copia de la clave pública del receptor, lo cual se puede realizar, en principio, por cualquier medio de comunicación, aunque sea inseguro. (Standards, Nov. 2001.)

A diferencia de los algoritmos de clave secreta, que existen desde los tiempos de los romanos, los métodos asimétricos son muy recientes. En 1976, Whitfield Diffie y Martin Hellman crearon un método con la ayuda de Ralph Merkle para iniciar una comunicación segura sin haber acordado previamente una clave secreta. El método se conoce como Diffie -Hellman Key Exchange. Poco más tarde se publicó el primer algoritmo asimétrico completo, denominado RSA, que sigue siendo el más utilizado en la actualidad. (Hellman:, Nov. 1976, pp: 644-654.)

3.2.1 FUNCIÓN DE UNA SOLA DIRECCIÓN

FUNCION DE UNA VIA: y=f(x) es “computacionalmente fácil” ($\in$ P) pero x=f-1(y) es “computacionalmente difícil” ($\in$NP)

Este paradigma de “fácil de calcular, difícil de invertir” es tan común en criptografía que las funciones que tienen esta propiedad son denominadas funciones unidireccionales o de una sola

dirección. La criptografía simétrica o de clave secreta y la criptografía asimétrica o de clave pública son funciones unidireccionales.

Una función unidireccional es una función matemática en la cual se conoce un procedimiento de cálculo eficiente y rápido para computar esa función, mientras que no se conoce un procedimiento eficiente para realizar ese mismo cálculo pero a la inversa.

Existen muchos ejemplos de funciones unidireccionales. Uno famoso es el problema de la factorización de números enteros: multiplicar dos primos grandes de centenas o miles de bits es viable y fácil para los ordenadores actuales como explicamos en nuestro ejemplo del algoritmo RSA de clave pública; sin embargo, no se conoce un algoritmo eficiente para invertir esta multiplicación, es decir, recuperar los dos primos partiendo exclusivamente de su producto.

El tema de las funciones unidireccionales es una de las preguntas abiertas más importantes en computación teórica y una ciencia muy práctica en el campo de la criptografía. De momento, no se tienen pruebas matemáticas de la existencia de funciones de una dirección, pero los criptógrafos“entienden” que tales funciones existen y usan algunos candidatos razonables en la construcción de algoritmos criptográficos (PREUKSCHAT, 2014)

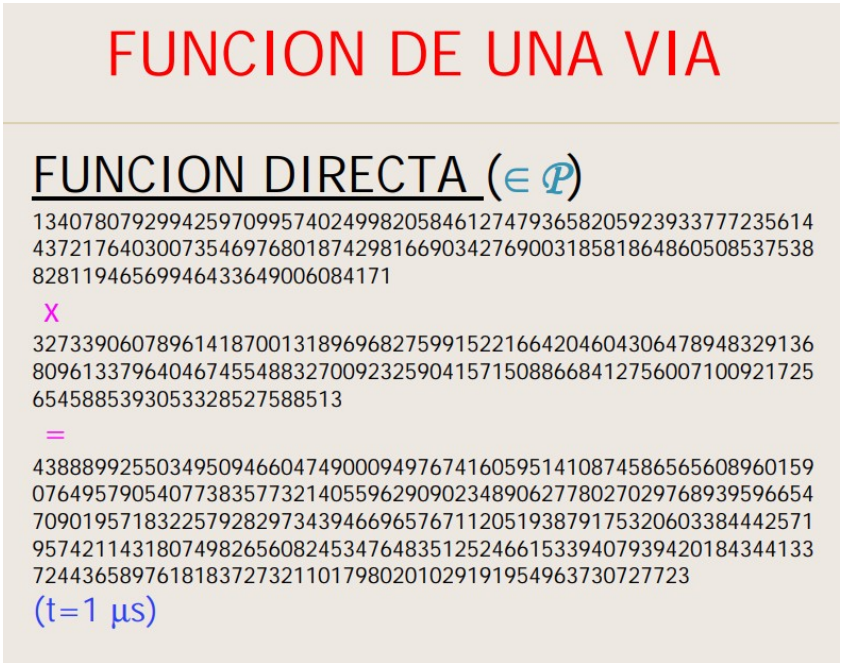


Fig. 20 Ejemplo Función Directa

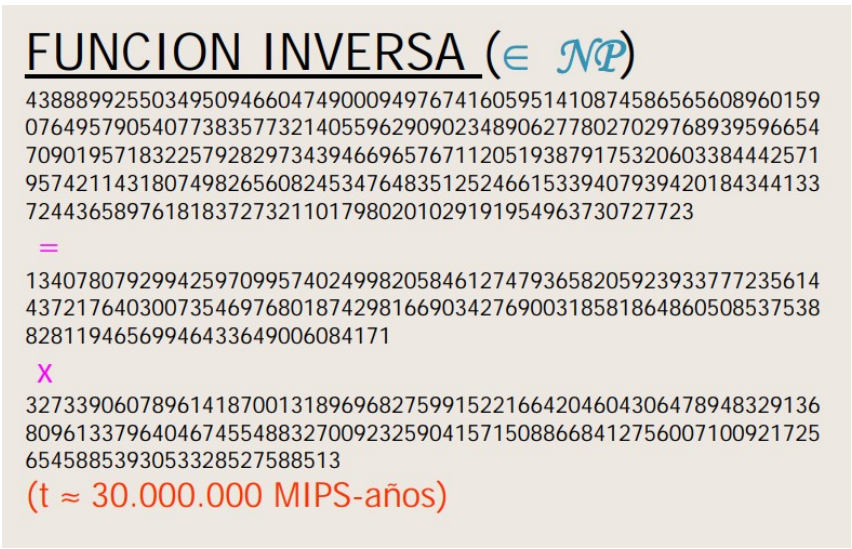


Fig. 21 Ejemplo Función inversa

3.2.2 FUNCIÓN TRAPDOOR

FUNCION TRAMPA DE UNA VIA: y=f(x) es “computacionalmente fácil” (∈ P) y x=f-1(y) es “computacionalmente difícil” (∈ NP), pero contando con una “clave” extra de información (trapdoor information) x=f-1(y) se vuelve computable (∈P)

Definición: f: D → R es una función unidireccional de puerta-trampa si hay una puerta trampa s tal que: - Sin conocimiento de s, la función f es una función unidireccional - Dado s, invertir f es fácil

Ejemplo: fg, p (x) = gx mod p no es una función trapdoor one way. • Ejemplo: RSA es una trampa OWF

RSA como una colección de Trap-door

RSA es un método que depende del parámetro dado por la clave, permite que yo sea un conjunto de índices y D un conjunto finito. Una colección de función de una sola puerta trampa es un conjunto de función F

fi: Di→ Ri

Tal que para todo lo que en fi es una trampa de una sola dirección función

Entonces es necesario un algoritmo que, dado un parámetro de seguridad, seleccione una función aleatoria fi en F junto con una información de trampa. (G´erard Maze)

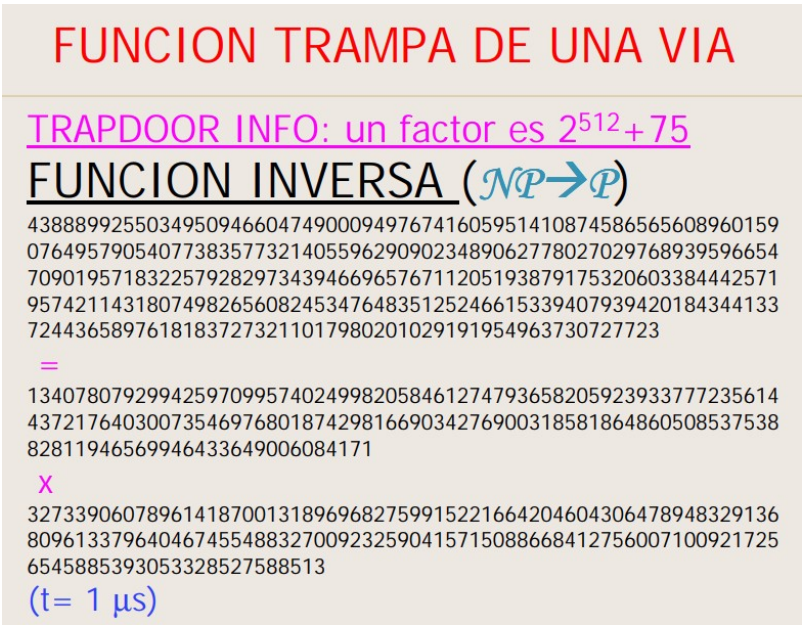


Fig. 22 Ejemplo Función inversa TRAPDOOR

3.3 RSA

3.3.1 FUNDAMENTOS MATEMÁTICOS

Para lograr un entendimiento de los algoritmos de cifrado, tanto de clave pública (asimétricos) como de clave privada (simétricos) se hace necesaria una somera introducción a algunos conceptos básicos de ciertas áreas de la matemática, como son la teoría de números y el álgebra lineal. El objetivo de este capítulo es presentar estos temas para facilitar la comprensión y lectura de los capítulos subsiguientes. No es intención dar una demostración formal de cada teorema, sino exponer los conceptos básicos de los mismos.

Nociones elementales de teoría de números: Usaremos la notación b/a (b divide a a) para significar que a = kb para algún entero k. Diremos que a es un múltiplo de b. Todo entero divide a 0. Si b no divide a a escribiremos b//a. Si b/a y b>0 diremos que b es un divisor de a. Todo entero a es divisible por 1 y a (se los llama divisores triviales). A los divisores no triviales de un entero (si existen) se los denomina factores. Números primos y compuestos Un entero a > 1 tal que sus únicos divisores son 1 y a se denomina primo. Ejercicio: demostrar que hay infinitos números primos. Los enteros que no son primos se denominan compuestos. El número 1 no es ni primo ni compuesto (lo mismo sucede con el 0 y los enteros negativos). El teorema de la división, restos y la equivalencia modular Dado un entero n, el conjunto Z de todos los enteros puede particionarse en dos subconjuntos: los que son múltiplos de n y los que no lo son. Gran parte de la teoría de números se basa en el refinamiento de esta partición obtenida clasificando a los enteros que no son múltiplos de n de acuerdo con el resto que se obtiene al dividirlos por n. Teorema 1 (de la división): Para cualquier entero a y cualquier entero positivo n existen enteros únicos q y r tales que 0≤ r

Dado un número real x se define a la función parte entera [x] como al mayor entero que no supera a x. El valor q = [a/n] es el cociente de la división. El valor r = a mod n es el resto (o residuo) de la división. Por lo tanto, n/a si y solo si a mod n = 0. Por lo tanto, a = [a/n]n + (a mod n) (1) o a mod n = a – *a/n+n (2) Si a mod n = b mod n escribimos a ≡ b (mod n). Se dice en este caso que a es equivalente a b módulo n. Equivalentemente a ≡ b (mod n) si y solo si n / (b-a)

Escribiremos a ≈ b si a no es equivalente a b módulo n. Como a ≡ b (mod n) es una noción de equivalencia, el conjunto Z se particiona en n clases de equivalencia de acuerdo con sus restos al dividirlos por n. La clase de equivalencia módulo n que contiene a un entero a es: [a]n = {a + kn : k∈Z} Escribir entonces a∈*b+n equivale a a ≡ b (mod n). El conjunto de todas esas clases de equivalencia es

$$Zn = ,*a+n: 0 \leq a \leq n-1\} \qquad (1)$$

$$\text{En general se usa la definición: } Zn = \{0,1,..., n-1\} \qquad (2)$$

que debe leerse como equivalente a (1) entendiendo que 0 representa [0]n, 1 representa [1]n, etc.

Naturalmente hay que tener en cuenta a las clases de equivalencia subyacentes. Por ejemplo, una referencia a –1 en Zn debe entenderse como una referencia a [n-1]n dado que –1 ≡ n – 1 mod n. (Leon, 2005)

Divisores comunes y máximo común divisor (MCD)

Si d es un divisor de a y también un divisor de b, entonces diremos que d es un divisor común de a y b. Por ejemplo, los divisores de 30 son: 1,2,3,5,6,10,15,y 30, así que los divisores comunes de 24 y 30 son 1,2,3, y 6. Una propiedad importante de los divisores comunes es que:

$$d / a \text{ y } d / b \text{ implica } d / (a+b) \text{ y } d / (a-b) \qquad (3)$$

Mas generalmente, tenemos que

$$d / a \text{ y } d / b \text{ implica } d / (ax + by) \qquad (4)$$

para cualquier par de enteros x,y. También, si

$$a / b \text{ entonces } /a/ \leq /b/ \text{ o } b = 0 \Rightarrow a / b \text{ y } b / a \Rightarrow a = \pm b \quad (5)$$

El máximo común divisor de dos enteros a y b que no son simultáneamente nulos es el mayor de sus divisores comunes, y se denota por MCD(a, b). Por ejemplo, MCD (24,30)=6, MCD (3,5)=1, MCD (0,9)=9. Si $ab \neq 0$ entonces $1 \leq MCD(a, b) \leq \min(/a/, /b/)$. Definimos MCD (0,0)=0 para tornar estándar las propiedades de la función MCD.

Propiedades de la función MCD:

$$MCD\ (a, b) = MCD\ (b, a) \quad (6)$$

$$MCD\ (a, b) = MCD\ (-a, b) \quad (7)$$

$$MCD\ (a, b) = MCD\ (/a/, /b/) \quad (8)$$

$$MCD\ (a, 0) = /a/ \quad (9)$$

$$MCD\ (a, a) = /a/ \text{ para cualquier } k \in Z \quad (10)$$

Primos relativos

Dos enteros a, b se dice que son primos relativos si MCD (a,b) = 1.

El siguiente teorema establece que si dos enteros son primos relativos con otro entero p, entonces su producto es primo relativo con p.

Teorema 3: Para enteros cualesquiera a,b y p, si MCD (a,p) = 1 y MCD (b,p) = 1 , entonces MCD (ab,p) = 1.

Demostración:

Del Teorema 2 surge que existen enteros w,x,y,z tales que

$aw + px = 1$

$by + pz = 1$ Multiplicando estas ecuaciones resulta

$ab(wy) + p(xby + zaw + pxz) = 1$

Factorización única

Un hecho elemental pero importante acerca de la divisibilidad por primos es el siguiente

Teorema 4: Para cualquier primo p y todos los enteros a,b , si p / ab entonces p / a o p / b.

Demostración: Supongamos que p / ab pero $p // a$ y $p // b$. Por lo tanto, MCD (a,p) = 1 y MCD (b,p) = 1 puesto que los únicos divisores de p son 1 y p, y por hipótesis p no divide ni a a ni a b. El Teorema 3 implica que MCD (ab,p) = 1, lo que contradice la suposición de que p / ab puesto que p / ab implica MCD (ab,p) = p. Esta contradicción completa la demostración.

Una consecuencia del Teorema 4 es que un entero tiene una única factorización en números primos.

El máximo común divisor

En esta sección vamos a presentar el algoritmo de Euclides para calcular el MCD de dos enteros en forma eficiente y a analizar su complejidad computacional. Nos vamos a restringir a los enteros positivos En principio podríamos calcular MCD (a,b) a partir de las factorizaciones en primos de ambos números, o sea:

$$a = p_1^{e1} \ldots p_k^{ek}$$
$$b = p_1^{f1} \ldots p_k^{fk} \qquad (11)$$

(12)

Donde algunos exponentes pueden ser nulos para que podamos usar el mismo conjunto de primos para a y b. Entonces,

$$MCD(a,b) = p_1^{min(e1,f1)} \ldots p_k^{min(ek,fk)} \qquad (13)$$

Como veremos luego, los mejores algoritmos para factorizar NO son polinomiales, o sea que el enfoque anterior para calcular MCD (a,b) no puede conducir a un método eficiente.

El algoritmo de Euclides se basa en el siguiente

Teorema 6 (de la recursión): para cualquier entero no negativo a y cualquier entero positivo b

MCD (a,b) = MCD (b, a mod b)

Demostración:

Demostraremos que MCD (a, b) y MCD (b, a mod b) se dividen mutuamente, de modo tal que por (7) deben ser iguales dado que ambos son no negativos.

Veamos primero que MCD (a,b) / MCD (b, a mod b).

Sea d =MCD(a,b), entonces d / a y d / b. Por (2), a mod b = a – q b donde q = [a/b].

Entonces a mod b es una combinación lineal de a y b y (6) implica que d / a mod b. Entonces, como d / b y d / a mod b, el Corolario 2.3 implica que d / MCD (b, a mod b) o, equivalentemente, que

MCD (a, b) / MCD (b, a mod b) *(14)*

Demostrar que MCD (b, a mod b) / MCD (a,b) es casi lo mismo.

Sea d=MCD (b,a mod b), entonces d / b y d / a mod b.

Como a = q b + a mod b donde q =[a/b] tenemos que a es una combinación lineal de b y a mod b. Por (6), concluímos que d / a, y como d / b, resulta que d / MCD (a,b) por el Corolario 2.3 o, equivalentemente, que:

MCD (b, a mod b) / MCD (a,b) *(15)*

Usando (5) para combinar las ecuaciones (14) y (15) resulta el teorema.

El algoritmo de Euclides

El siguiente algoritmo se describe en el libro "Elementos" (aproximadamente 300 años AC) aunque quizás sea anterior.

Sean a y b dos enteros arbitrarios no negativos

Procedimiento Euclides (a,b)

1. Si b = 0

2. Entonces devolver a

3. de lo contrario devolver Euclides (b, a mod b) (Scolnik, 2004)

La complejidad del algoritmo de Euclides

Analizaremos el peor caso en función de los tamaños de a y b. Supondremos que sin pérdida de generalidad que a > b ≥ 0. Esto se justifica porque si b > a ≥ 0 entonces Euclides (a,b) inmediatamente hace la llamada recursiva Euclides (b,a). O sea que si el primer argumento es menor que el segundo, Euclides "gasta" una llamada recursiva para intercambiar a los mismos.

Similarmente si a = b > 0, el procedimiento termina luego de una llamada recursiva puesto que a mod b = 0

El tiempo total de ejecución de Euclides es proporcional al número de llamadas recursivas que hace. Para analizarlo necesitamos los números de Fibonacci definidos por la siguiente recurrencia:

F0 = 0

F1 = 1

Fi = Fi-1 + Fi-2 para i ≥ 2

Teorema 7: Si a > b ≥ 0 y el procedimiento Euclides (a,b) realiza k ≥1 llamadas recursivas, entonces a ≥ Fk+2 y b ≥ Fk+

Demostración: Por inducción sobre k. Sea k =1. Entonces b ≥ 1 = F2 y como a > b debe ser a ≥ 2 = F3. Como b > a mod b en cada llamada recursiva el primer argumento es estrictamente mayor que el segundo, o sea que la suposición a > b es válida para cada llamada recursiva.

Supongamos que el teorema es cierto para k – 1 llamadas recursivas, y veremos que también lo es para k llamadas. Como k > 0 tenemos que b > 0 y Euclides (a,b) llama a Euclides (b, a mod b) recursivamente y este último hace a su vez k – 1 llamadas recursivas. La hipótesis inductiva implica que b ≥ Fk+1 a mod b ≥ Fk.

Tenemos que

b + (a mod b) = b + (a – *a/b+b) ≤ a

Como a > b > 0 implica que *a/b+ ≥ 1.

Por lo tanto

a ≥ b + (a mod b) ≥ Fk+1 + Fk = Fk+2

Teorema 8 (Lamé): para cualquier entero $k \geq 1$, si $a > b \geq 0$ y $b < Fk+1$, entonces la invocación al procedimiento Euclides (a,b) realiza menos de k llamadas recursivas.

Podemos mostrar ahora que la cota superior del teorema 8 es la mejor posible. Dos números de Fibonacci consecutivos constituyen el peor caso para el algoritmo de Euclides. Dado que Euclides (F3,F2) hace exactamente una llamada recursiva y como $k \geq 2$, tenemos que $Fk+1 \bmod Fk = Fk-1$

$$MCD\ (Fk+1, Fk) = MCD\ (Fk, Fk+1 \bmod Fk) = MCD\ (Fk, Fk-1)$$

Por lo tanto el algoritmo Euclides (Fk+1,Fk) hace $k -1$ llamadas recursivas, llegando a la cota superior del Teorema 8.

El algoritmo de Euclides extendido

El objetivo es extender el algoritmo que vimos para calcular coeficientes enteros x,y que satisfagan

$$d = MCD\ (a,b) = ax + by \qquad (16)$$

Notar que x,y pueden ser nulos o negativos. La utilidad de dicho algoritmo será para calcular inversas multiplicativas modulares. El procedimiento extendido tiene como entrada a un par arbitrario de enteros y devuelve una terna (d,x,y) que satisface (16).

Procedimiento Euclides extendido (a,b)

1. Si b = 0

2. Retornar con (a,1,0)

3. de lo contrario, (d',x',y') ← Euclides extendido (b,a mod b)

4. (d,x,y) ← (d',y',x' –*a/b+ y')

5. retornar (d,x,y) Veamos el cálculo de MCD (99,78)

a	b	[a/b]	d	x	y
99	78	1	3	-11	14
78	21	3	3	3	-11
21	15	1	3	-2	3
15	6	2	3	1	-2
6	3	2	3	0	1
3	0	-	3	1	0

Esto equivale al test " b = 0" en Euclides. Si b = 0, el Euclides extendido devuelve d = a en el paso 2. junto con

x = 1,

y = 0, o sea que

a = ax + by.

Si b ≠ 0, el algoritmo calcula (d',x',y') tal que

d' = MCD (b,a mod b) y

d' = bx' + (a mod b) y' *(17)*

Como en Euclides, tenemos en este caso que d = MCD (a, b) = d' = MCD(b, a mod b).

Para obtener x, y tales que d = ax + by comenzamos por reescribir (17) usando d = d'

d = bx' + (a – *a/b+ b) y' = ay' + b(x' – *a/b+ y')

De este modo, eligiendo x = y' , y = x' – *a/b+ y' se satisface la ecuación d = ax + by, lo que demuestra la correctitud del algoritmo.

Dado que el número de llamadas recursivas en los dos algoritmos es igual, los tiempos de procesamiento solo difieren en un factor constante. (Scolnik, 2004)

Grupos finitos

Un grupo (S, $\oplus$) es un conjunto S con una operación binaria $\oplus$ para la cual se verifican las siguientes propiedades:

1. Cerrado: Para todo a, b ε S $\Rightarrow$ a$\oplus$b ε S

2. Identidad: Existe un elemento neutro e ε S tal que a $\oplus$ e = e $\oplus$ a = a $\forall$ a $\in$ S

3. Asociatividad: $\forall$ a, b, c $\in$ S, (a $\otimes$ b) $\otimes$ c = a $\otimes$ (b $\otimes$ c)

4. Inversa: $\forall$ a $\in$ S $\exists$ un único elemento b $\in$S tal que a $\otimes$ b = b $\otimes$ a = e

Si un grupo satisface la propiedad conmutativa a $\otimes$ b = b $\otimes$ a $\forall$ a, b $\in$ S, se dice que es abeliano y si /S/ < ∞ (donde /S/denota la cardinalidad de S) se dice que el grupo es finito.

Podemos formar dos grupos abelianos finitos utilizando la suma y la multiplicación módulo n, donde n es un entero positivo, basados en las clases de equivalencia de los enteros módulo n. Para definir un grupo en Zn necesitamos operaciones binarias convenientes que se obtienen redefiniendo las operaciones ordinarias de suma y multiplicación. Es simple definir las operaciones de suma y multiplicación en Zn porque la clase de equivalencia de dos enteros determina unívocamente la clase de equivalencia de su suma y su producto. Si a ≡ a' (mod n) y b ≡ b' (mod n) entonces:

a + b ≡ a' + b' (mod n)

ab ≡ a'b' (mod n)

De este modo definimos la suma y la multiplicación módulo n , denotadas por +n y . n mediante

[a]n +n [b]n = [a + b]n

[a]n .n [b]n = [ab]n y análogamente la substracción

[a]n –n [b]n = [a –b]n

Estos hechos justifican la práctica de usar el menor elemento positivo de cada clase de equivalencia como su representante cuando se realizan cálculos en Zn. La suma, la resta y la multiplicación se llevan a cabo como siempre con sus representantes, pero l os resultados se expresan con el representante de su clase (o sea x → x mod n).

Usando esta definición de la suma módulo n definimos el grupo aditivo módulo n como

(Zn ,+n) recordando que /Zn/= n.

Teorema 12: (Zn,+n) es un grupo abeliano finito.

Usando la definición de multiplicación módulo n definimos el grupo multiplicativo módulo n como (Z*n ,.n) donde Z*n denota a los elementos de Zn que son primos relativos con n, o sea

Z*n = { [a]n ∈ Zn : MCD(a,n) = 1}

Para ver que este conjunto está bien definido notemos que para 0 ≤ a < n resulta que

a ≡ (a + kn) (mod n) para todos los enteros k.

Es fácil ver entonces que MCD(a,n) = 1 ⇒ MCD(a +kn,n) = 1

Como [a]n = {a + kn ; k ∈Z} resulta que Z*n está bien definido.

Teorema 13: (Z*n ,.n) es un grupo abeliano finito.

Vimos en el teorema 6 que para a,b,p enteros tales que

MCD (a,p) = MCD (b,p) = 1 entonces MCD (ab,p) =1.

Esto implica que (Z*n ,.n) es cerrado.

De ahora en adelante vamos a denotar a las clases de equivalencia por sus elementos representativos y a las operaciones simplemente por + y . También las equivalencias módulo n serán interpretadas como ecuaciones en Zn. Por ejemplo. son equivalentes las expresiones:

ax ≡ b (mod n) y *a+n .n *x+n = *b+n

También vamos a omitir la explicitación de las operaciones cuando las mismas sean evidentes a partir del contexto. Así

(Zn ,+n) será Zn y (Z*n ,.n) será Z*n.

La inversa multiplicativa de un elemento a se notará (a -1 mod n). La división en Z*n está definida por la ecuación a/b ≡ ab -1 (mod n)

La cardinalidad de Z*n, o sea /Z*n/se denota por ϕ (n), conocida como la función de Euler, que satisface la ecuación

$$\phi(n) = n \prod_{p<n} (1 - 1/p) \qquad (18)$$

donde *p* recorre todos los primos que dividen a n (incluyendo a n si n es primo). Intuitivamente comenzamos con la lista de los restos {0,1,...,n-1} y entonces para cada primo p que divide a n, borramos los múltiplos de p de dicha lista

Consecuencia importante: si *p* es primo, Z*p = {1,2,...,p-1- y ϕ (p)= p-1

Si n es compuesto, entonces

$\phi(n) < n - 1$ (Scolnik, 2004)

Subgrupos

Si $(S, \oplus)$ es un grupo, $S' \subseteq S$, y $(S', \oplus)$ también es un grupo, entonces se dice que $(S', \oplus)$ es un subgrupo de $(S, \oplus)$. Por ejemplo, los enteros pares forman un subgrupo de los enteros con la operación de suma. El siguiente teorema da una herramienta útil para reconocer subgrupos:

Teorema 14 (un subconjunto cerrado de un grupo finito es un subgrupo):

Si $(S, \oplus)$ es un grupo finito y S' es cualquier subconjunto de S tal que si $a,b \in S' \Rightarrow a \oplus b \in S'$, entonces $(S', \oplus)$ es un subgrupo de $(S, \oplus)$.

Teorema 15 (Lagrange): Si $(S', \oplus)$ es un subgrupo de $(S, \oplus)$, entonces $/S'/$ es un divisor de $/S/$. Un subgrupo S' de un grupo S se dice propio si $S' \neq S$.

El siguiente Corolario es útil en el test de Rabin-Miller para generar números pseudoprimos;

Corolario 16: Si S' es un subgrupo propio de S

Resolución de ecuaciones modulares:

Vamos a considerar ahora la resolución de ecuaciones de la forma

$$ax \equiv b \pmod{n} \text{ donde } n > 0 \qquad (19)$$

Teorema 16: Para cualquier par de enteros positivos a,n, si d = MCD (a,n) entonces

$$(a) = (d) = \{0, d, 2d, \ldots, ((n/d) - 1)d\} \text{ y } /(a)/ = n/d \qquad (20)$$

Demostración: primero veremos que d ∈ (a). El procedimiento Euclides extendido (a,n) calcula enteros x', y' tales que d = ax' + ny'. Por lo tanto, ax' ≡ d (mod n), de modo tal que d ∈ (a). Entonces, todo múltiplo de d pertenece a (a) porque un múltiplo de un múltiplo de a es un múltiplo de a. O sea que (a) contiene a todo elemento en (d) lo que implica que (d) ⊆ (a). Veremos ahora que (a) ⊆ (d). Si m ∈ (a), entonces m =ax mod n para algún entero x, o sea que m = ax + ny para algún entero y. Sin embargo, d/a y d/n, así que d/m lo que implica que m ∈ (d). Combinando estos resultados resulta que:

(a) = (d). Para ver que /(a)/= n/d

basta observar que hay exactamente n/d múltiplos de d entre 0 y n - 1 inclusive.

Corolario 17: La ecuación (19) es resoluble si y solo si MCD (a,n)/b

Corolario 18: La ecuación (19) tiene d soluciones distintas módulo n donde d = MCD (a,n) o no tiene soluciones.

Demostración: Si x es solución, entonces b ∈ (a). La sucesión ai (mod n) , i = 0 ,1, ... es periódica con período /(a)/= n/d . Si b ∈ (a) , entonces b aparece exactamente d veces en la sucesión ai mod n , i = 0 ,1, ... n - 1 porque los bloques de valores de (a) se repiten d veces cuando i varía entre 0 y n – 1.

Teorema 17: sea d = MCD (a,n) y supongamos que d = ax' + ny' para un par de enteros x',y' (por ejemplo dados por el procedimiento Euclides extendido). Si d/b, tiene como una solución al valor x0 = x'(b/d) mod n. Demostración: como ax' ≡ d (mod n) tenemos que:

ax0 ≡ ax'(b/d) (mod n) ≡ d(b/d) (mod n) ≡ b (mod n)

y por lo tanto x0 es una solución de:

ax ≡ b (mod n)

Teorema 18: suponiendo que la ecuación ax ≡ b (mod n) es resoluble esto es,

d = MCD(a,n) / b) y que x0 es cualquier solución de la misma.

Entonces esta ecuación tiene exactamente d soluciones distintas módulo n dadas por

$x_i = x_0 + i(n/d)$ para $i = 1, 2, \ldots, d - 1$.

Demostración: dado que $n/d > 0$ y $0 \leq i(n/d) < n$ para $i = 0, \ldots, d - 1$, los valores $x_0, x_1, \ldots, x_{d-1}$ son todos distintos módulo n. Por la periodicidad de la sucesión $a_i \pmod n$ si x_0 es una solución de $ax \equiv b \pmod n$ entonces todo x_i es una solución, hay exactamente d soluciones de modo tal que ellas son $x_0, x_1, \ldots, x_{d-1}$

Hasta aquí hemos desarrollado los conceptos matemáticos necesarios para resolver la ecuación $ax \equiv b \pmod n$. El siguiente algoritmo imprime todas las soluciones de la misma, donde a, b son enteros arbitrarios y n es un entero positivo arbitrario.

Ecuaciones modulares (a,b,n)

1. $(d,x',y') \leftarrow$ Euclides extendido (a,n)
2. si d/b
3. entonces $x_0 \leftarrow x'(b/d) \bmod n$
4. para $i \leftarrow 0$, hasta $d - 1$
5. imprimir $(x_0 + i(n/d)) \bmod n$
6. de lo contrario imprimir " no hay soluciones"

Los siguientes corolarios son casos de particular interés para la Criptografía:

Corolario 19 : para cualquier **n > 1 , si MCD(a,n) = 1**, entonces la ecuación **$ax \equiv b \pmod n$** tiene una única solución módulo n.

El caso **b =1** es muy importante, pues en ese caso estamos calculando la inversa de a módulo n.

Corolario 20: para cualquier $n > 1$, si MCD (a,n) = 1, la ecuación $ax \equiv 1 \pmod n$ tiene una única solución módulo n. De lo contrario, o sea si MCD (a,n) $\neq$ 1, no existe ninguna solución.

El Corolario 26 nos permite usar la notación *$(a^{-1} \bmod n)$* cuando a y n son primos relativos. Si MCD (a,n) = 1, entonces una solución de la ecuación *$ax \equiv 1 \pmod n$* es el entero x dado por el

procedimiento Euclides extendido puesto que la ecuación MCD*(a,n) = 1 = ax + ny* implica que *ax ≡ 1 (mod n).* De este modo *(a -1 mod n)* puede calcularse eficientemente mediante el procedimiento Euclides extendido. (Scolnik, 2004)

Generación de claves

Este algoritmo se basa en escoger 2 números primos grandes elegidos de forma aleatoria y mantenidos en secreto. La principal ventaja de este algoritmo desde el punto de vista de seguridad radica en la dificultad a la hora de factorizar números grandes. RSA es seguro hasta la fecha.

La idea del algoritmo es la siguiente:

Tenemos un mensaje M. Empleando un protocolo reversible conocido como patrón de relleno convertimos el mensaje Men un número m menor que otro número dado n.

Se genera el mensaje cifrado c:

$$c \equiv m^e \pmod{n}$$

Se obtiene m descifrando el mensaje cifrado c:

$$m \equiv c^d \pmod{n}$$

1. Tomamos 2 números primos p y q. Estos tienen que ser aleatorios e impredecibles. Importante impredecibles, porque un proceso puede ser perfectamente aleatorio, pero si se conoce, se puede predecir los valores, y por tanto, resultaría en una baja seguridad.

2. Calculamos n=p*q.

3. Calculamos ϕ(n). La función ϕ de Euler se define como el número de enteros positivos menores o iguales a n y coprimos con n (dos números son coprimos si no tienen ningún divisor común distinto 1 o -1). La función tiene las siguientes propiedades:

ϕ(1)=1.

ϕ(p)= p-1 si p es primo.

ϕ(p^k)= (p-1)*p^(k-1) si p es primo y k un número natural.

ϕ(m*n)=ϕ(m)ϕ(n) si m y n son primos.

De esta forma nos queda que:

ϕ(n)= (p-1)*(q-1)

4.-Escogemos un número e menor que ϕ(n) y que sea coprimo con ϕ(n).

Este número será dado a conocer como exponente de la clave pública.

5.-Obtenemos un número d mediante aritmética modular tal que d = e^(-1) mod(ϕ(n)) o lo que es lo mismo (d*e)-1 tiene que ser divisible por ϕ(n).

De esta forma tenemos la clave pública formada por (n,e) y la privada formada por (n,d).

p = 61 1º nº primo privado

q = 53 2º nº primo privado

n = p*q = 3233 producto p×q

e = 17 exponente público

d = 2753 exponente privado

La clave pública (e, n). La clave privada es (d, n). La función de cifrado e

$$\text{encrypt}(m) = m^e \pmod{n} = m^{17} \pmod{3233}$$

Donde m es el texto sin cifrar. La función de descifrado es:

$$\text{decrypt}(c) = c^d \pmod{n} = c^{2753} \pmod{3233}$$

Donde c es el texto cifrado. Para cifrar el valor del texto sin cifrar 123, nosotros calculamos:

$$\mathbf{encrypt}(123) = 123^{17} \pmod{3233} = 855$$

Para descifrar el valor del texto cifrado, nosotros calculamos:

$$\mathbf{decrypt}(855) = 855^{2753} \pmod{3233} = 123$$

Como hemos mencionado anteriormente, RSA requiere de un esquema de rellenado dado que sino M puede conducirnos a textos cifrados inseguros. Hay múltiples algoritmos de rellenado, entre ellos podemos destacar OAEP (Optimal Asymetric Encription Padding) o SAEP (Simplified Asymetric Encryption Padding). RSA (Rivest, Shamir y Adleman) es un algoritmo de cifrado asimétrico desarrollado en el año 1977 por los anteriormente citados.

Este algoritmo se basa en escoger 2 números primos grandes elegidos de forma aleatoria y mantenidos en secreto. La principal ventaja de este algoritmo desde el punto de vista de seguridad radica en la dificultad a la hora de factorizar números grandes. RSA es seguro hasta la fecha. La idea del algoritmo es la siguiente:

Tenemos un mensaje M. Empleando un protocolo reversible conocido como patrón de relleno convertimos el mensaje Men un número m menor que otro número dado n.

Se genera el mensaje cifrado c:

$$c \equiv m^e \pmod{n}$$

Se obtiene m descifrando el mensaje cifrado c: (Scolnik, 2004)

$$m \equiv c^d \pmod{n}$$

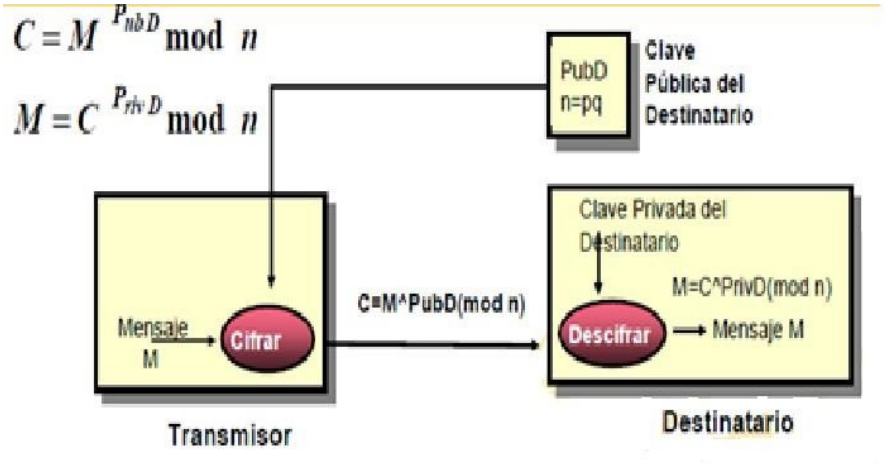


Fig. 23 Algoritmo RSA

3.3.2 FUNCIÓN DE CIFRADO RSA

El cifrado y descifrado se realiza en el anillo entero Zn RSA cifra el texto plano P que es un elemento en Zn = ,0, 1,…, n-1}. Por lo tanto el valor binario de P tiene que ser menor que n. Lo mismo sucede con el descifrado.

Cifrado: Dada la clave pública (n,e) = kpub y el texto plano P, la función de cifrado es:

$$\mathbf{C = ek_{pub}(P) \equiv P^e \bmod n}$$

donde

P y C ϵ Zn

Descifrado: Dada la clave privada d = kpr y el texto cifrado C, la función de descifrado

es:

$$P = dk_{pr}(C) \equiv C^d \bmod n$$

donde

P y C ϵ Zn

P, C y n son números grandes ≥ 1024 bits.

e: exponente de cifrado o exponente público.

d: exponente de descifrado o exponente privado

Si Bob le quiere enviar un mensaje cifrado a Alice, necesita la clave pública (n,e) de esta última y Alice, a su vez, descifra con su clave privada d.

Algunos requerimientos del criptosistema RSA:

Dados e y n debe ser inviable obtener d.

No se pueden cifrar más de n bits. En caso que sea necesario debe hacerse por bloques.

Se necesita un método de exponenciación rápida de manera de poder obtener fácilmente tanto Pe mod n, como Cd mod n (algoritmo de elevar al cuadrado y multiplicar).

Para un n dado debe haber muchos pares de claves públicas/privadas, sino sería viable un ataque de fuerza bruta. (Franchi, 2012)

3.3.3 ALGORITMO DE POTENCIACIÓN MODULAR ÓPTIMA

En el criptosistema RSA, una vez definidos los exponentes e y d Y publicada la clave pública (e, n), tanto los emisores como los receptores utilizan la misma operación para encriptar, desencriptar, firmar y verificar: la exponenciación modular.

Es importante notar que esta operación no es utilizada exclusivamente por el criptosistema RSA, sino que es parte integral de varios otros criptosistemas y esquemas de firma digital, tales como el esquema de intercambio de claves de Diffie-Hellman [4], el esquema de firma de El-Gamal [5] y el Digital Signature Standard (DSS) propuesto por el National Institute for Standard s and Technology (NIST) [6].

Sin embargo, el proceso de exponenciación que se lleva a cabo en un criptosistema basado en el problema de logaritmo discreto como los mencionados es levemente diferente al tratado en este trabajo: en ellos, la base M y el módulo n están dados previamente y es el exponente e el que varía en cada bloque a encriptar o desencriptar, 10 que permitiría utilizar la precomputación de ciertas potencias de la base para acelerar el proceso subsiguiente [7]. En el caso de interés en el presente trabajo, en que buscamos optimizar el proceso de exponenciaci6n dentro del algoritmo RSA, tanto el exponente e como el módulo n son conocidos de manera anticipada, pero no la base M (esto es, el texto plano a ser encriptado o el texto encriptado a ser desencriptado), por lo que este tipo de optimización no resulta aplicable. La optimización de la exponenciación modular que investigaremos busca tomar ventaja de este conocimiento anticipado del exponente e, y existen variados métodos propuestos en la literatura. A continuación presentaremos los más utilizados por sus características de simplicidad o velocidad, buscando luego modificarlos para acelerar aún más el cálculo.

Exponenciaciones modulares directas Si bien el concepto detrás de una exponenciación modular es extremadamente simple (una serie de multiplicaciones modulares hasta arribar al resultado deseado), existen variados métodos para realizar esta operación. La multiplicación modular, operación fundamental en la exponenciación modular, es en general computacionalmente costosa, especialmente en el caso de operando de precisión múltiple como los considerados en el criptosistema RSA. Por lo tanto, el objetivo general es realizar la menor cantidad de multiplicaciones modulares posibles, de manera que la exponenciación modular se realice a su vez de manera rápida. El método más intuitivo y a la vez ingenuo de computar e = Me mod(n) es claramente el comenzar con e := M mod(n) y luego continuar aplicando repetidamente la multiplicación modular c:= C·M mod(n) hasta obtener el valor deseado. Este método requiere e -

1 multiplicaciones modulares, una cantidad de operaciones que puede resultar prohibitiva en el caso en que el valor de e sea lo suficientemente elevado (y en un criptosistema RSA con longitud de clave importante ese será el caso). Por ejemplo, en un sistema con longitud de clave k = 512 bits, el exponente e puede ser un número de hasta 155 dígitos decimales, lo que da una idea de 10 impracticable de este método Sin embargo, no todas las potencias intermedias de M hasta llegar a Me necesitan ser calculadas, y diversos métodos computacionalmente más económicos (y por 10 tanto más veloces) han sido propuestos, comenzando por el denominado método binario y avanzando hacia técnicas más complejas, las cuales se describirán a continuación.

Algoritmos de ventana deslizante Este tipo de algoritmo adaptativo utiliza un acercamiento diferente para lograr ahorros en la cantidad de multiplicaciones a ser realizadas en una exponenciación modular. En un método m-ario puro como los arriba mencionados, la probabilidad de que al particionar la representación binaria del exponente e en grupos de d bits, se logre un bloque conformado por todos bits en O es de z", asumiendo que los bits O y 1 son equiprobables. Como en el paso 4.2 del algoritmo m-ario se saltea una multiplicación cuando el valor del bloque es O, resulta evidente que a medida de que la longitud de bloque d es más grande, la probabilidad de evitar esta multiplicación es menor. Pero por otro lado, la cantidad total de multiplicaciones a realizar se incrementa a medida que el valor de d disminuye. Los algoritmos s de ventana deslizante buscan lograr un punto medio entre estos dos objetivos, buscando utilizar valores de d relativamente grandes, a la vez que el número promedio de bloques en Obusca incrementarse también. Como regla general, los algoritmo s de ventana deslizante buscan descomponer la expansión binaria del exponente e en bloques (o ventanas, de ahí el nombre dado a este algoritmo) a los que denominaremos F;, de tamaño L(F;) (de manera análoga a la notación utilizada en los métodos m-arios). Para mayor claridad, denominaremos ventana cero a aquellas cuyos bits son todos nulos (F; = O), y ventana no-cero a las que contengan por lo menos un bit distinto a O (F; -::j:. O). El número de ventanas puede cambiar dependiendo del exponente siendo tratado, y es probable que exponentes con la misma cantidad de bits generen distinto número de ventanas. Más aún, no se requiere que las ventanas sean todas del mismo tamaño. Para mayor claridad, llamemos d a la longitud de la mayor ventana. Una característica favorable es que como las ventanas se generan particionando el exponente de derecha a izquierda (es decir, desde el bit menos significativo de e hacia el más significativo), y no hay razón para comenzar las ventanas no-cero con un bit distinto al, entonces siempre el bit menos significativo en cada ventana será igual a l. Esto se traduce en que los valores de todas las ventanas no-cero serán siempre impares, lo que implica a su vez que la cantidad de

multiplicaciones en la precomputación será aproximadamente la mitad que en un algoritmo m-ario común, ya que será necesario computar M W sólo para los w impares.

En general, podemos describir el algoritmo de la siguiente manera:

Entrada: M, e, n, d
Salida: $C = M^e \bmod(n)$

1. Computar y almacenar $M^w \bmod(n)$ para $w = 3,5,7,\ldots,2^d - 1$
2. Particionar e en ventanas cero y no-cero F_i de largo $L(F_i) \le d$ para $i = 0,1,2,\ldots,p-1$
3. $C \leftarrow M^{F_{k-1}} \bmod(n)$
4. **Desde** $i = p-2$ **hasta** 0
   - 4.1 $C \leftarrow C^{2^{L(F_i)}} \bmod(n)$
   - 4.2 **Si** $F_i \neq 0$ **entonces** $C \leftarrow C \cdot M^{F_i} \bmod(n)$
5. **Devolver** C

Fig. 24 Algoritmos de ventana deslizante

3.3.4 GENERACIÓN DE NÚMEROS PRIMOS GRANDES

Dos números, a y b, se dicen que son primos relativos o coprimos si el mcd (a,b) =1. Es útil notar que todos los elementos de los conjuntos Zn* y Zp* son primos relativos con respecto al módulo del conjunto

Función de Euler

La función de Euler es de vital importancia para el cálculo de potencias grandes y juega un papel preponderante en la criptografía. Básicamente esta función devuelve la cantidad de números enteros que son menores y coprimos a un número n. Podemos decir que calcula la cantidad de elementos que componen al conjunto Zn*.

Algunas propiedades de la función de Euler:

1. Φ(1) = 0.

2. Φ(p) = p-1, con p primo.

3. Φ(m . n) = Φ(m) . Φ(n) si m y n son coprimos.

4. Φ(pe) = (pe - pe-1) si p es primo

Con las propiedades anteriores podemos hallar el valor de Φ(n). Considerando a:

$$n = p_1^{e1} \times p_2^{e2} \times \ldots\ldots p_i^{ei}$$

entonces podemos encontrar:

$$\Phi(n) = (p_1^{e1} - p_1^{e1-1}) \times (p_2^{e2} - p_2^{e2-1}) \times \ldots\ldots \times (p_i^{ei} - p_1^{ei-1})$$

La complejidad de hallar el valor de Φ(n) para un n grande es directamente dependiente de la complejidad de factorización de n.

n	1	2	3	4	5	6	7	8	9	10	11	12
Φ(n)	1	1	2	2	4	2	6	4	6	4	10	4

Teorema de Euler y pequeño Teorema de Fermat

Estos dos teoremas son también de fundamental importancia en la criptografía.

Según el teorema de Euler si mcd (a,n) = 1, entonces:

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

Otra versión del teorema de Euler (usada en el criptosistema RSA) establece que si

n = p . q, a < n y k es un entero, entonces:

$$a^{k \cdot \phi(n)+1} \equiv a \bmod n$$

Un caso particular del teorema de Euler sucede si consideramos que n es un número primo. El pequeño teorema de Fermat establece que si p es un número primo y a un entero tal que mcd (p,a) = 1, entonces:

$$a^{p-1} \equiv 1 \bmod p$$

Otra versión establece que si p es un número primo y a es un entero, entonces:

$$a^{p} \equiv a \bmod p$$

Dado que los métodos para encontrar un número primo (números de Fermat, números de Mersenne) no han dado resultado, la manera de generar números primos de gran tamaño es elegir un número al azar y verificar posteriormente si es primo. Esta no es una tarea sencilla y para ello existen dos tipos de algoritmos: determinísticos y probabilísticos. (Franchi, 2012)

3.3.5 ALGORITMO DE MILLER-RABIN

Vamos a considerar primero un test de primalidad que “casi funciona” y que de hecho sirve para muchas aplicaciones prácticas. Luego, un refinamiento de este método permite eliminar el problema que el original tiene, tema que veremos más adelante. Diremos que n es un pseudoprimo de base a si n es compuesto…

$$a^{n-1} \equiv 1 \pmod{n}$$

si n es primo entonces n satisface para todo elemento a ∈ Z*n

O sea, que si podemos encontrar un * n a∈ Z tal que n entonces n es compuesto. Sorprendentemente, lo recíproco “casi” es cierto, con lo cual tenemos un criterio “casi” perfecto de primalidad.

Primero testeamos si n satisface con a = 2.

Si no, es compuesto. En caso afirmativo, podría ser primo (de hecho sabemos que o es primo o es un pseudoprimo de base 2)

El siguiente procedimiento pretende testear de este modo la primalidad de n.

Pseudoprimo(n)

1. Si Exponenciación modular $(2,n-1,n)$ no es congruente con $1 \pmod n$

2. entonces devolver COMPUESTO (con seguridad)

3. de lo contrario devolver PRIMO (posiblemente)

Este procedimiento puede tener errores, pero solo de un tipo. Esto es, si dice que n es compuesto, esto es definitivo. Si decimos que n es primo, podemos equivocarnos pues podría ser un pseudoprimo de base 2. La cuestión es: esto sucede con frecuencia

Puede demostrarse que la probabilidad de que el error ocurra con n un número de k bits tiende a cero cuando $k \to \infty$. Estimaciones más precisas (C.Pomerance, On the distribution of pseudoprimes, Mathematics of Computation, 37(156), pp. 587-593, 1981) muestran que un entero de 50 dígitos elegido al azar que es dado por primo por el procedimiento anterior, tiene una probabilidad menor que una en un millón de ser un pseudoprimo de base 2, y si fuera de 100 dígitos la probabilidad se reduce a 10^{-13} . Lamentablemente no podemos eliminar los posibles errores testeando con otra base, por ejemplo a =3 porque existen números n compuestos tales que la ecuación (40) se cumple para todo * n $a \in Z$ .Estos enteros n con dicha propiedad son extremadamente raros, y se denominan números de Carmichael (los primeros tres son 561, 1105 y 1729). De hecho son tan poco frecuentes que solo hay 255 de ellos menores que 100.000.000

Veremos ahora como mejorar el test de primalidad de modo tal que no pueda ser “engañado” por los números de Carmichael

El test de primalidad de Rabin-Miller:

Este test evita los problemas del procedimiento Pseudoprimo (n) mediante dos modificaciones:

1. Prueba diversos valores de a elegidos al azar en vez de un solo valor

2. Mientras calcula cada exponenciación modular, testea si se encuentra una raíz cuadrada no trivial de 1 módulo n. En caso afirmativo, el resultado para n es "COMPUESTO"

A continuación incluímos el pseudocódigo para el test de Rabin-Miller. El dato de entrada n > 2 es el impar que queremos testear para ver si es primo, y s es el número de valores de base * n a∈ Z que se van a ensayar. Suponemos que tenemos un procedimiento Azar (1, n – 1) que calcula enteros al azar a que satisfacen 1≤ a ≤ n –1. El código también usa un procedimiento llamado Testigo(a,n) que da el valor "VERDAD" si y solo si a es un "testigo" de que n es compuesto, o sea que es posible usar a para probar que n es compuesto. Este procedimiento es similar, pero más efectivo, que el procedimiento Pseudoprimo, que estaba basado en el test con a =2. Primero presentamos y justificamos Testigo, y luego veremos como se usa en el test de Rabin-Miller

Algoritmo

```
Miller-Rabin(n,s)
        for(i = 1; i ≤ s; i + +){
                a = rand(1, n − 1);
                if(Testigo(a,n)) devolver(Falso); /* compuesto */
        }
        devolver(Cierto); /* muy prob. primo */
Testigo(a, n)
        < b_1, b_2, ..., b_k > representación binária de n − 1
        for(d = 1,i = k; i ≥ 0; i − −){
                x = d;
                d = (dd) mod n;
                if (d == 1 and x ≠ 1 and x ≠ −1)
                        devolver(Cierto); /* por el corolario */
                if (b_i == 1) d = (da) mod n;
        }
        if(d ≠ 1) devolver(Cierto); /* por el teorema */
        devolver(Falso);
```

Fig. 25 Test de Rabin-Miller

Este pseudocódigo para Testigo se basa en el procedimiento Exponenciación modular. La línea 1 determina la representación binaria de n –1 que se usará para calcular a n-1 . Las líneas 3-9 calculan d como a n-1 mod n, tal como en Exponenciación modular. Pero cada vez que se calcula un cuadrado en la línea 5 las líneas 6-7 controlan si se ha encontrado una raíz cuadrada no trivial de 1. Si es así, devuelve el valor "VERDADERO".

Las líneas 10-11 devuelven "VERDADERO" si el valor computado para a n-1 mod n es distinto de 1, tal como el procedimiento Pseudoprimo devuelve "COMPUESTO" en ese caso. Veamos ahora que si Testigo (a,n) devuelve "VERDADERO" entonces podemos probar que n es compuesto usando a. Si Testigo (a,n) devuelve "VERDADERO" en la línea 11 entonces descubrió que

$$d = a^{n-1} \bmod n \neq 1.$$

Si n es primo, el teorema 31 (Fermat) nos dice que

$$a^{n-1} \equiv 1$$

(mod p) ∀ a ∈ Z*n Por lo tanto n no puede ser primo. Si Testigo (a,n) devuelve “VERDADERO” en la línea 7, entonces encontró que x es una raíz cuadrada no trivial de 1 módulo n, o sea que x no es congruente con ± 1 módulo n aunque x 2 ≡ 1 (mod n).

Se establece que solo si n es compuesto existe una raíz cuadrada no trivial de 1 módulo n, con lo que queda demostrada la correctitud del procedimiento Testigo (a,n).

Veamos ahora el Procedimiento Rabin-Miller (n,s)

1. Para j ← 1 hasta s

2. do a ← Azar (1,n – 1)

3. Si Testigo (a,n)

4. Entonces devolver “COMPUESTO” (con seguridad)

5. devolver “PRIMO” (casi seguro)

Este procedimiento es una búsqueda probabilística de una prueba que n es compuesto. El loop principal (que comienza en la línea 1) elige s valores al azar de a en el intervalo [1,n – 1] en la línea 2. Si uno de esos valores de a es un “testigo” de que n es compuesto, entonces el procedimiento devuelve el valor “COMPUESTO” en la línea 4, y este resultado es siempre correcto debido a la demostración de correctitud del procedimiento Testigo (a,n). Si no se encuentra ningún testigo de que n es compuesto, entonces el procedimiento de Rabin-Miller asume que el número n es primo. Luego veremos que esto es esencialmente correcto si s es lo suficientemente grande, pero siempre existe una probabilidad (que se puede hacer tan pequeña como se quiera) de que existan testigos que el procedimiento no encontró. Veamos un ejemplo con el número de Carmichael 561. Supongamos que elegimos a = 7 como base. La tabla – que ya vimos – era:

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140	280	560
d	7	49	157	526	160	241	298	166	67	1

Testigo encuentra una raíz no trivial de 1 puesto que a 280 ≡ 67 (mod n) y a 560 ≡ 1 (mod n). Entonces a =7 es un testigo y el procedimiento Testigo (7, n) devuelve "VERDADERO" y Rabin-Miller "COMPUESTO". (Scolnick, 2004)

3.3.6 PRIMOS FUERTES

Por último hablaremos de los llamados primos duros (fuertes) y su utilidad en el sistema RSA

Un primo p es duro si p-1 tiene un factor primo r grande, 0<v<1 que tenga un factor primo grande y que r-1 tenga un factor primo grande, para propósitos criptográficos podemos considerar que un primo grande es de alrededor de 100 bits.

El solicitar que los primos para el RSA sean duros, pretende evitar que el producto n=pq pueda ser factorizado por métodos que piden factores pequeños en p±1 y r-1 como el método de Pollar y Williams, sin embargo, primos duros no evitan el Método con Curvas Elípticas el cual se considera que éste puede ser limitado su éxito con la longitud de los primos grandes y su aleatoriedad

Respecto a los primos duros existen dos opiniones que no difieren considerablemente para propósitos criptográficos

La primera que sustenta RSA Data Security es que no son necesarios los primos duros, ya que éstos no evitan el MCE, y el considerar a primos grandes y aleatorios evita en la mayoría de métodos sobre factorización más potentes y por lo tanto en gran probabilidad a los métodos de Pollar y Williams.

La segunda opinión es que en efecto, los argumentos de la primera opinión son acertados sin embargo el añadir condiciones a la generación de los primos para que éstos sean duros no ofrece ninguna dificultad ni tiempo extra considerable así que es sugerida la utilización de primos duros. (Jesus, 2002)

3.3.7 TIPOS DE ATAQUES Y VULNERABILIDADES SOBRE EL RSA

Para romper un cifrado RSA, podemos probar varias vías. Aparte de factorizar n, que ya sabemos que es un problema computacionalmente intratable en un tiempo razonable, podríamos intentar calcular ϕ(n) directamente (Si recordamos, la función de Euler ϕ(n)= (p-1) (q-1), y que en general, salvo azar improbable, se tendrá que mcd (M, p) = mcd (M, q) = mcd (M, n) = 1. Y por tanto según el teorema de Euler-Fermat, Mϕ(n) ≡ 1 (mod n) ⇒ (M (p-1) (q-1)) k ≡ 1 (mod n) *ii+), o probar por

un ataque de fuerza bruta tratando de encontrar la clave privada d, probando sistemáticamente con cada uno de los números posibles del espacio de claves. Ambos ataques son, para n grandes, incluso aún más costosos computacionalmente que la propia factorización de n. Necesitaremos actualmente unos 22.020.985.858.787.784.059 (2e64) años para romper una clave cifrada con 128 bits. Sin embargo existen otros tipos de ataques al algoritmo RSA.

Para analizar la seguridad del sistema, se supone que el criptoanalista tiene una cantidad ilimitada de pares (m, c) de mensajes originales y sus correspondientes criptogramas. Las posibles maneras que tiene de atacar el sistema son las siguientes.

Factorizar n.

Calcular Φ(n).

Ataque por iteración.

Ataque de Blakley y Borosh.

Ataque por tiempo

Ataque adaptable de texto encriptado seleccionado

Ataque al secreto de n por cifrado cíclico

Ataque a la clave por paradoja cumpleaños

Factorizar n:

De esta forma obtiene el número Φ(n)= (p-1)*(q-1), y con él la clave privada d, puesto que he es pública y se cumple: e*d-1(mod Φ(n)). Al ser n el producto de solo dos números primos, un algoritmo de factorización requiere como máximo ‘n pasos, pues uno de los dos factores es necesariamente un primo menor que ‘n. Sin embargo, si n fuera el producto de N>2 primos, un algoritmo de factorización necesitaría como máximo n^ (1/N) pasos, que es una cota menor que ‘n, por lo que se concluye que es adecuada la obtención de n como producto de solo dos números

primos. Con respecto al estudio del problema de la factorización, hay que mencionar al precursor de la moderna factorización, el algoritmo de fracciones continuas de Morrison-Brillhart, ya que es uno de los más rápidos. Sin embargo, los dos algoritmos de factorización que resultan más prácticos para grandes enteros corresponden al de factorización con curvas elípticas de Hendrik Lenstra y al de factorización con filtro cuadrático de Carl Pomerance. Ambos algoritmos convierten el problema de la factorización de un entero n en el problema de encontrar soluciones no triviales ('x <> y (mod n)' y 'x <> -y (mod n)') de la ecuación: x3 -y3 (mod n). Si se supone que ni (x+y) ni (x-y) son múltiplos de n enteros, se deduce que el m.c.d. (x+y, n) o bien el m.c.d. (x-y, n) es con seguridad un factor no trivial de n, por lo que se resuelve el problema de la factorización.

Calcular Φ(n):

Como se ve a continuación, esta manera es equivalente a la anterior. Si se tiene Φ(n), dado que p+q=n-Φ(n)+1 y a partir de la suma se puede calcular (p-q) 2 por coincidir con (p-q) 2-4*n, luego se consigue la factorización mediante las formulas q= [(p+q)-(p-q)]/2 y p= [(p+q)+ (p-q)]/2.

Ataque por iteración:

Si un enemigo conoce (n,e,C), entonces puede generar la secuencia: C1-C^e(mod n), ..., Ci-[C(i-1)]^e(mod n), con lo que si existe algún Cj tal que C=Cj se deduce que el mensaje buscado es M=C(j-1) pues [C(j-1)]^e=Cj=C. Ahora bien, en cuanto la igualdad Cj=C se cumple solo para un valor de j demasiado grande, este ataque se vuelve impracticable. Con respecto a esto, Rivest demostró que si los enteros p-1 y q-1 contienen factores primos grandes, la probabilidad de éxito mediante este procedimiento es casi nula para grandes valores de n.

Ataque de Blakley y Borosh:

El sistema RSA, además, tiene una característica muy peculiar, advertida por Blakley y Borosh, y es que no siempre esconde el mensaje. A continuación vemos un ejemplo que lo muestra. Si e=17, n=35 y los mensajes a cifrar son M1=6 y M2=7, entonces se obtiene que 6^17-6(mod 35) y 7^17-7(mod 35). Una situación más peligrosa para el sistema aparece, por ejemplo, con los valores p=97 q=109 y e=865, ya que el criptosistema resultante no esconde ningún mensaje, pues M^865-M(mod 97*109) En general, lo ocurrido en el ultimo ejemplo ocurre siempre que e-1 es múltiplo de p-1 y q-1, pues en ese caso M^e-M(mod p*q). Además, se tiene que para cualquier elección de n=p*q siempre existen al menos 9 mensajes M que no se cifran en realidad, ya que verifican la ecuación M^e-M(mod n). De esos 9 mensajes hay tres fijos, que son M perteneciente a {0,1,-1}.

Para hacer que el sistema RSA sea resistente contra ataques basados en este hecho, es conveniente elegir como claves privadas números primos de la forma p=2*p'+1, donde p' es un primo impar.

Ataque por tiempo

Kocher describió un ingenioso y nuevo ataque contra RSA en 1995: si el atacante Eve conoce el hardware de Alice en suficiente detalle y es capaz de medir los tiempos de decriptación para diversos textos encriptados conocidos, entonces ella podrá deducir la clave de decriptación d rápidamente. Este ataque también puede ser aplicado al esquema de firma RSA. Una manera de evitar este ataque es asegurarse que las operaciones de decriptación tomen una cantidad constante de tiempo para cada texto crifrado. Otra manera es usar la propiedad multiplicativa del RSA. En lugar de calcular cd mod N, Alice primero escoge un valor aleatorio r secreto y calcula (re c) d mod N. El resultado de este cálculo es rm mod N y así el efecto de r puede ser removido multiplicando es resultado por su inversa. Un nuevo valor de r es elegido por cada texto cifrado.

Ataque adaptable de texto encriptado seleccionado

Un ataque adaptable de texto encriptado seleccionado es una forma interactiva de ataque según el cual un atacante envía un cierto número de textos cifrados a ser decriptados, y luego utiliza los resultados de estas decriptaciones para seleccionar subsecuentes textos encriptados. La meta de este ataque es revelar gradualmente información sobre el mensaje encriptado, o sobre la clave de decriptación. Para sistemas de clave pública, este método es generalmente aplicable sólo cuando se tiene la posibilidad de manejar el texto encriptado. – esto es, un texto encriptado puede ser modificado de maneras específicas para que tengan efectos predecibles en la decriptación de ese mensaje. Ataques Prácticos Los ataques adaptables de texto encriptado seleccionado fueron por mucho tiempo considerados como cuestiones meramente teóricas hasta 1998, cuando Daniel Bleichenbacher de Bell Laboratorios demostró un ataque práctico contra sistemas que utilizaban encriptación RSA y la función de codificación enunciada en el PKCS # 1, incluyendo además una versión del protocolo SSL (Secure Socket Layer) utilizado por miles de servidores web por ese entonces. Los ataques de Bleichenbacher tomaron ventaja de defectos del PKCS # 1 para gradualmente revelar el contenido de un mensaje encriptado en RSA y potencialmente revelar las claves de la sesión16. Hacer esto requiere enviar muchos millones de textos encriptados de prueba al dispositivo de decriptación. En términos prácticos, esto significa que la clave de una sesión SSL puede ser expuesta en una razonable cantidad de tiempo, tal vez un día o menos. (Leon, 2005)

Ataque al secreto de n por cifrado cíclico

Se puede encontrar el número en claro N sin necesidad de conocer d, la clave privada del receptor. Como C = N ˚mod n, realizaremos cifrados sucesivos de los criptogramas Resultantes con la misma clave pública hasta obtener nuevamente el cifrado C original.

$$Ci = C^{e}i\text{-}1 \bmod n \quad (i = 1, 2, \ldots) \text{con} \quad C_0 = C$$

Si en el cifrado i-ésimo se encuentra el criptograma C inicial, entonces es obvio que el cifrado anterior (i-1) será el número buscado. Esto se debe a que RSA es un grupo multiplicativo. Para evitarlo hay que usar primos seguros de forma que los subgrupos de trabajo sean lo suficientemente altos.

Un ejemplo significativo: Sea p = 13, q = 19, n = 247, ϕ(n) = 216, e = 29 (d = 149, no conocido) El número a cifrar será M = 123 ⇒ C = 123e29mod 247 = 119

i	Ci
i = 0	C0= 119
i = 1	$C1 = 119^{29} \bmod 247 = 6$
i = 2	$C2 = 6^{29} \bmod 247 = 93$
i = 3	$C3 = 93^{29} \bmod 247 = 175$
i = 4	$C4 = 175^{29} \bmod 247 = 54$
i = 5	$C5 = 54^{29} \bmod 247 = 123$
i = 6	$C6 = 123^{29} \bmod 247 = 119$

El a nente: como hemos obtenido otra vez el criptograma C = 119 n C = 123 se correspondía con el texto en claro. Si usamos prin será mayor y el sistema menos vulnerable. (Acosta, 2011)

Fig. 26 Identificación de SSH mediante clave

Previsión de ataques Para prevenir ataques con este método, es necesario utilizar un esquema de encriptación o codificación que limite el manejo del texto cifrado. Se han propuesto numerosos esquemas de codificación; el estándar más común para encriptación RSA es el OAEP (Optimal Asymmetic Encryption Padding) que es un esquema de relleno probablemente seguro. RSA

Laboratories ha publicado también nuevas versiones de PKCS # 1 que no son vulnerables a estos ataques. (Conde)

3.4 ALGORITMO ELGAMAL

3.4.1 CARACTERISTICAS DEL ALGORITMO ELGAMAL

Fue diseñado en un principio para producir firmas digitales, pero posteriormente se extendió también para codificar mensajes. Se basa en el problema de los logaritmos discretos, que está íntimamente relacionado con el de la factorización, y en el de:

Diffie-Hellman.

Para generar un par de claves, se escoge un número primo n y dos números aleatorios p y x menores que n. Se calcula entonces

$$y = p^x (mód) n$$

La clave pública es (p; y; n), mientras que la clave privada es x.

Escogiendo n primo, garantizamos que sea cual sea el valor de p, el conjunto

$\{p, p^2, p^3 \ldots\}$ es una permutación del conjunto $\{1, 2, \ldots, n-1\}$. Nótese que esto no es necesario para que el algoritmo funcione, por lo que podemos emplear realmente un n no primo, siempre que el conjunto generado por las potencias de p sea lo suficientemente grande.

Firmas Digitales de ElGamal

Para firmar un mensaje m basta con escoger un número k aleatorio, que sea primo relativo con n - 1, y calcular

$$a = p^k (mód\ n)$$

$$b = (m - xa)k^{-1}\ (mód\ (n-1))$$

La firma la constituye el par (a; b). En cuanto al valor k, debe mantenerse en secreto y ser diferente cada vez. La firma se verifica comprobando que

$$y^a a^b = p^m (\text{mód } n)$$

Cifrado de ElGamal

Para cifrar el mensaje m se escoge primero un número aleatorio k primo relativo con

(n-1), que también será mantenido en secreto. Calculamos entonces las siguientes expresiones

$$a = p^k (\text{mód } n)$$

$$b = y^k m \ (\text{mód } n)$$

El par (a; b) es el texto cifrado, de doble longitud que el texto original. Para decodificar se calcula:

$$m = b * a^{-x} \ (\text{mód } n)$$

3.5 ALGORITMO DIFFIE-HELLMAN "AES"

El protocolo de cifrado Diffie-Hellman (recibe el nombre de sus creadores) es un sistema de intercambio de claves entre partes, que no han contactado previamente, a través de un canal inseguro y sin autenticación.

Este protocolo se utiliza principalmente para intercambiar claves simétricas de forma segura para posteriormente pasar a utilizar un cifrado simétrico, menos costoso que el asimétrico.

Se parte de la idea de que dos interlocutores pueden generar de forma conjunta una clave sin que esta sea comprometida.

Se escoge un número primo p y un generador g que será coprimo de p. Estos 2 números son públicos.

Escogemos un número a menor que p y calculamos A = g^a mod p. Enviamos A, p y g al otro interlocutor.

El otro interlocutor escoge un número b menor que p y calcula B = g^b mod p. Nos envía B.

Ahora, ambos podemos calcular K= g^(a-b) mod p.

Para nosotros B^a mod p = K y para nuestro interlocutor A^b mod p =K. Usamos K como clave

Al ser p y g públicos cualquier atacante puede conocerlos. Esto no supone una vulnerabilidad. Aunque el atacante conociera estos dos números y capturase A y B, le resultaría computacionalmente imposible obtener a y/o b y consecuentemente K.

Tomamos p=23 y g=5. Elegimos a=6 y b=15. A=8 y B=19.

(g^a mod p) = 8 → (5^a mod 23) = 8

(g^b mod p) = 19 → (5^b mod 23) = 19

Partiendo de estas ecuaciones obtener a y b es un problema conocido como logaritmo discreto.

a = log discg (g^a mod p) = log disc5 (8)

b = log discg (g^b mod p) = log disc5 (19)

En este caso si podríamos obtenerlos, pues sabiendo que p=23 y que a y b son menores que p solo tendríamos que probar 22 números diferentes. En la realidad se utilizan números primos del orden de 10^200 haciendo computacionalmente imposible la resolución.

Este protocolo es vulnerable a ataque man-in-the-middle. Estos ataques consisten en que un tercero se coloca en medio del canal y hace creer a ambos que es el otro. De esta forma se podría acordar una clave con cada parte y servir de “enlace” entre los dos participantes. Para que este ataque sea funcional se necesita saber que método de cifrado simétrico se va a emplear. Ocultar el algoritmo de cifrado no cumple con el principio de Kerkckhoffs de que la efectividad de un sistema no debe depender de que su diseño permanezca en secreto por lo que conocer el sistema que se va a emplear se hace trivial.

Para evitar esto se puede emplear un protocolo de autenticación de las partes mediante por ejemplo TLS. (CORRALES, CILLERUELO, & CUEVAS, 2013)

3.6 ALGORITMO RW

RW es un algoritmo de cifrado asimétrico basado en el problema de la factorización entera (al igual que RSA). No obstante, este algoritmo no ha sido prácticamente estudiado ni usado por la comunidad criptográfica moderna. (Master, 2004) Como se realiza el algoritmo de RW:

Escoger dos grandes primos **p** y **q** (secretos) y definir n (público) mediante su producto, **n = p * q** (público).

Escoger un entero **B < n** que será parte de la clave pública **(B, n)**

Para cifrar un texto, es necesario previamente codificar el texto en un sistema de base b dividiéndola previamente en bloques de tamaño j-1 de forma que cumpla **b^ j-1 < n < b^ j**

Cifrar el bloque M i según la expresión **C i ≡ M i (M i + B) (mod n)**

Para descifrar el criptograma ***C i*** , hay que construir el mensaje ***M i = a * u + b * v*** a partir de las soluciones de las ecuaciones ***u^ 2 + B * u ≡ C i (mod p) y v^ 2 + B * v ≡ C i (mod q),*** y los enteros a y b tales que ***a ≡ 1 (mod p) , a ≡ 0 (mod q) , b ≡ 0 (mod p) y b ≡ 1 (mod q).*** (Gil, 2002). Al estar basado este algoritmo en el mismo problema que RSA, posee igualmente sus mismas debilidades, y es también extremadamente vulnerable al criptoanálisis cuántico.

Existe una sutil diferencia, pues en el sistema RSA, resolver el problema general de la factorización implica romperlo, pero no de forma inversa. En el sistema de Rabin, ambos preceptosse cumplen. Este defecto de base fue corregido en 1980 por H.C. Williams, dando lugar al algoritmo RW.

3.7 ALGORITMO LEHMANN

Es uno de los test más sencillos para saber si un número p es o no primo:

1. Escoger un número aleatorio a < p.

2. Calcular b = a (p−1)/2 (mód p).

3. Si b 6= 1 (mód p) y b 6= −1 (mód p), p no es primo.

4. Si b ≡ 1 (mód p) o b ≡ −1 (mód p), la probabilidad de que p sea primo es igual o superior al 50 %.

Repitiendo el algoritmo n veces, la probabilidad de que p supere el test y sea compuesto es decir, no primo será de 1 contra 2n. (Lucerna López, 2010)

3.8 ALGORITMO DSA "DIGITAL SIGNATURE ALGORIRHM"

El algoritmo DSA (Digital Signature Algorithm) es una parte el estándar de firma digital DSS (Digital Signature Standard). Este algoritmo, propuesto por el NIST, data de 1991, es una variante del método asimétrico de ElGamal.

Creación del par clave pública-clave privada

El algoritmo de generación de claves es el siguiente:

Seleccionar un número primo *q* tal que $2^{159} < q < 2^{160}$

Escoger *t* tal que *0 ≤ t ≤ 8*, y seleccionar un número primo *p* tal que $2^{511+64t} < p < 2^{512+64t}$, y que además *q* sea divisor de *(p – 1)*.

Seleccionar un elemento $gcÆ *_p$ y calcular $\alpha = g^{\frac{(p-1)}{g}} m\acute{o}d\, p$.

Si $\alpha = 1$volver al paso 3

Seleccione un numero entero aleatorio α,tal que $1 \leq a \leq q - 1$

Calcular $y =$a$^{\alpha}\ m\acute{o}d\, p$.

La clave pública es (p, q, α, y). La clave privada es α.

Generación y verificación de la firma

Siendo h la salida de una función MDC sobre el mensaje m, la generación de una firma se hace mediante el siguiente algoritmo:

Seleccionar un número aleatorio k tal que 0 < k < q.

Calcular $r = (\text{a}^{k}\, m\acute{o}d\ p)\ m\acute{o}d\ q$.

Calcular $k^{-1} m\acute{o}d\ q$

Calcular $s = k^{-1}(\ h + ar)\ m\acute{o}d\ q$.

La firma del mensaje m es par (r, s).

El destinatario efectuará las siguientes operaciones, suponiendo que conoce la clave pública (p, q, α, y), para verificar la autenticidad de la firma:

Verificar que $0 < r < q < y$ $0 < s < q$. En caso contrario, rechazar la firma.

Calcular el valor de *h* a partir de *m.*

Calcular $w = s^{-1} mód\ q$

Calcular $u_1 = w * h\ mód\ q$ y $u_2 = w * r\ mód\ q$

Calcular $v = (a^{u1}\ y^{u2} mód\ p)\ mód\ q$.

Aceptar la firma si y solo si *v=r.*

El funcionamiento de DSA se divide en 3 etapas. Generación de claves, firma y verificación. Las dos primeras las realiza el emisor y la última el receptor. Este algoritmo (como su nombre lo indica) sirve para firmar y no para cifrar información. Una desventaja de este algoritmo es que requiere mucho más tiempo de cómputo que RSA.

3.9 CRIPTOGRAFIA DE CURVA ELIPTICA

Debido a la relación existente entre ambos, muchos algoritmos que se apoyan en el problema de la factorización pueden ser replanteados para descansar sobre los logaritmos discretos. De hecho, existen versiones de curva elíptica de muchos de los algoritmos asimétricos más populares.

3.9.1 CIFRADO ELGAMAL SOBRE CURVAS ELÍPTICAS

Sea un grupo de curva elíptica, definido en $GF(n)$ ó $GF(2^n)$. Sea *p* un punto de la curva. Sea el conjunto (*p*), de cardinal *n*. Escogemos entonces un valor entero *x* comprendido entre 1 y *n − 1,* y calculamos

$$y = xp$$

La clave pública vendrá dada por *(p, y, n),* y la clave privada será *x.*

El cifrado se hará escogiendo un número aleatorio k primo relativo con n. Seguidamente calculamos las expresiones

$$a = kp$$

$$b = m + ky$$

s

Siendo *m* el mensaje original representado como un punto de la curva. El criptograma será el par *(a, b).* Para descifrar, será suficiente con calcular

$$m = -(xa) + b$$

Para curvas elípticas existe un problema análogo al de los logaritmos discretos en grupos finitos de enteros. Esto nos va a permitir trasladar cualquier algoritmo criptográfico definido sobre enteros, y que se apoye en este problema, al ámbito de las curvas elípticas. La ventaja que se obtiene es que, con claves más pequeñas, se alcanza un nivel de seguridad equiparable.

4 CIFRADO PERFECTO Y DISTANCIA DE UNICIDAD.

4.1 CRIPTOGRAFÍA Y SEGURIDAD INFORMÁTICA.

La palabra criptología proviene de las palabras griegas Kryto y logos y significa estudio de lo oculto. Una rama de la criptología es la criptografía, que se ocupa del cifrado de mensajes. Esta se basa en que el emisor emite un mensaje en claro, que es tratado mediante un cifrador con la ayuda de una clave, para crear un texto cifrado. “El texto cifrado, el cual viaja por medio del canal de comunicación establecido, llegará al descifrador que convierte el texto cifrado, mediante otra clave de seguridad, la cual desencripta los datos o el mensaje encriptado para así obtener el texto en claro original conocido como texto plano”. Las dos claves implicadas en el proceso de cifrado/descifrado pueden ser o no iguales dependiendo del sistema de cifrado utilizado.

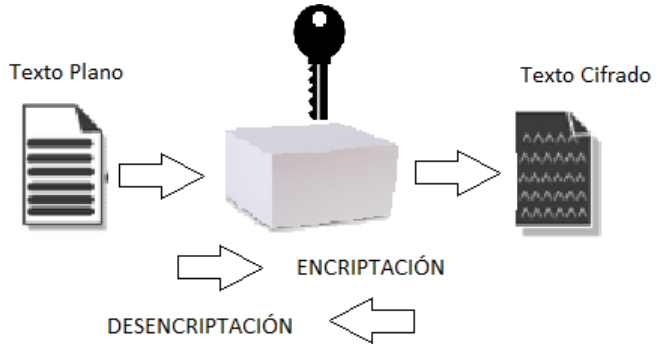


Fig. 27 Encriptación-desencriptación

La criptografía es la disciplina que estudia el modo de transformar un mensaje (texto original) en un texto en cifra (criptograma) mediante una operación de cifrado que hace imposible a un tercero tomar conocimiento del contenido del mensaje. “La criptografía, como cualquier rama de conocimiento posee su propio lenguaje o nomenclatura, para lo cual nosotros debemos conocerla para su mejor entendimiento y evidentemente no se utilizan todos los métodos conocidos más aún si los más conocidos y representativos.” Encriptar un texto significa aplicarle un algoritmo que, en relación a una cierta variable (clave de encriptación), lo transforma en otro texto incomprensible e indescifrable por parte de quien no posee la clave. La función es reversible, por lo cual la aplicación del mismo algoritmo y de la misma clave al texto cifrado devuelve el texto original. La encriptación ha sido inventada y utilizada originalmente para fines de seguridad en las trasmisiones de mensajes militares (Gersson Ribera, 2008).

Debemos tener en cuenta que durante la primera y segundo guerra mundial transcurrió sin computadoras, esto nos lleva a reflexionar que durante las guerras mencionadas se pueden haber propuesto algoritmos muy ingeniosos para codificar mensajes, siendo de muy importancia encontrar el método seguro de transmisión de menajes, pero para hacerlos funcionar a mano resultaba prácticamente imposible, ya fuera por el tiempo necesario o por los problemas que acarreaba un error cometido en un paso intermedio

Unos de estos algoritmos tienen su origen durante el Imperio Romano, en la época del Julio César. César utilizó un esquema criptográfico simple pero efectivo para comunicarse con sus generales. El esquema de César consistía en desplazar cada letra del alfabeto un número determinado de posiciones. Por ejemplo, la letra ”A”” podría ser codificada como ”M””, la ”B” como ”N””, la ”C”” como ”O””, así sucesivamente.

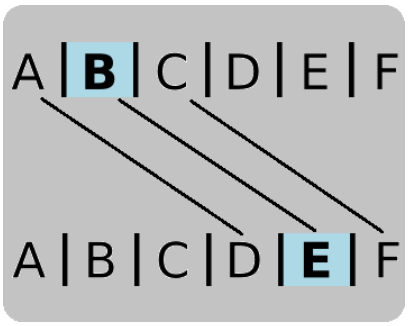


Fig. 28 Funcionamiento Código Cesar

El método de cifrado introducido por Julio César introduce el concepto de clave criptográfica. El desplazamiento de las letras es la clave que se utiliza por César para cifrar el mensaje, necesitándose la misma clave para descifrarlo. Seguidamente se comenzó a hablar de la criptografía hasta los trabajos de Shannon (1950), quién la relacionó con diversas disciplinas como la estadística, la teoría de los números, teorías de la información y de la complejidad. A partir de allí se la denominó ciencia.

“El cifrado que fue introducido por Julio César es el ejemplo más básico de transposición de texto, siendo este el pionero de la encriptación, tenía como fin la transposición de texto según el alfabeto, dicho de mejor manera buscaba que el mensaje llegase a su destino óptimamente, pero de no ser asi, el mensaje debía poseer la capacidad de ser no legible ante personas ajenas.

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

Cifrado 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 ”

4.2 SECRETO PERFECTO.

Para un criptosistema se dice que posee el secreto perfecto si: pp(x|y) = pp(x) " x"P, " y"C. Esto es, la probabilidad de el texto original x dada la probabilidad del texto encriptado y es idéntico a la probabilidad del texto original x.

4.3 SEGURIDAD COMPUTACIONAL

La computadora es un instrumento que estructura información en grandes cantidades, la cual puede ser confidencial y corre con el riesgo de ser divulgada a personas que hagan mal uso de ella, El uso inadecuado del computador comienza desde la utilización de tiempo de máquina para usos ajenos de la organización, la copia de programas para fines comerciales sin derechos de autor, hasta un acceso por vía telefónica a bases de datos a fin de modificar la información con propósitos fraudulentos.

El método más eficiente ante estos peligros con los cuales protegerá los sistemas de computación es el software de control de acceso, en mejores términos los paquetes de control de acceso protegen contra el acceso no autorizado de entes maliciosos, debido a que piden del usuario su contraseña para permitirle el acceso al sistema o la información

Se debe evaluar el nivel de riesgo que puede tener la información para poder hacer un adecuado estudio costo/beneficio entre el costo por pérdida de información y el costo de un sistema de seguridad, para lo cual debemos considerar.

Clasificar la instalación en términos de riesgo (alto, mediano, pequeño)

Identificar aquellas aplicaciones que tengan un alto riesgo

Cuantificar el impacto en el caso de suspensión del servicio en aquellas aplicaciones con un alto riesgo.

Formular las medidas de seguridad necesarias dependiendo del nivel de seguridad que se requiera.

La justificación del costo de implantar las medidas de seguridad para poder clasificar el riesgo e identificar las aplicaciones de alto riesgo, debemos preguntar lo siguiente:

Definido esto, debemos elaborar una lista de los sistemas con las medidas preventivas que se deben tomar, así como las correctivas en caso de desastre señalándole uno a uno su prioridad. (Luis Álvarez, 2005)

s

4.4 SEGURIDAD INCONDICIONAL.

Esta medida concierne a la seguridad del cryptosistema, se da cuando una tercera persona no puede entrar en el sistema computacional. Un criptosistema es definido para ser un seguro incondicional si no puede ser roto, aun con un recurso computacional infinito.

Se dice de un cifrado que es seguro con independencia del tiempo o los recursos ilimi tados que puedan invertirse para tratar de vulnerarlo. Tal cifrado alcanza el secreto perfecto. (Ribagorda,1997). *"Cable recalcar que ningún sistema es inviolable, y que la seguridad no es producto, sino un proceso que se fortalece mediante la gestión que se realiza a lo largo de lapso de tiempo".*

Definición.

El sistema es seguro, inclusive si el atacante posee recursos ilimitados para poder realizar una violación del sistema. En este caso, la seguridad se mide utilizando la teoría de la información y el análisis se realiza empleando la teoría de las probabilidades; el secreto perfecto es fruto de la observación de que el texto cifrado o criptograma no proporciona nada de información a un hipotético adversario. El cifrado Vernam, llamado también OTP (one time pad), este es un claro ejemplo de criptosistema con secreto perfecto el cual posee seguridad incondicional

La seguridad incondicional claramente no puede ser estudiado en el punto de vista de la complejidad computacional, pues aun para nosotros el tiempo de computación es infinita. La forma de trabajo para estudiar la seguridad incondicional, es utilizando la teoría de probabilidades, pues los parámetros tomados no son muy complejos. (Javier Areitio, 2008)

Con posibilidad de ruptura

Esta seguridad a su vez puede desglosarse en cinco tipos.

Seguridad de complejidad teórica.

El sistema se pude romper, pero se necesita más potencia de computación de la que un adversario real podría tener. Se supone que el adversario tiene potencia de computación polinómica; los ataques polinómicos, aunque son factibles, en la práctica pueden ser computacionalmente no factibles. En este caso, la seguridad se mide y analizara utilizando la teoría de la complejidad (Javier Areitio, 2008)

Seguridad Probable.

La prueba de la seguridad se fundamenta en la dificultad en resolver problemas supuestamente difíciles bien conocidos, como, por ejemplo; la factorización o el cálculo de los logaritmos discretos. (Javier Areitio, 2008)

Seguridad computacional o seguridad práctica.

Este modelo consiste en el esfuerzo computacional requerido para romper un criptosistema. Se puede definir un criptosistema como seguro computacional, si el mejor algoritmo para romperlo requiere N operaciones como máximo, donde N es un número grande. El problema es que no es conocido un criptosistema práctico que puede ser probado para ser seguro bajo esta definición.

En la práctica, la gente llama a un criptosistema "seguro computacional" si el mejor método para romper el criptosistema, requiere un irrazonable tiempo grande de uso de un computador (este concepto es diferente a probar la seguridad). Otra aproximación es proveer evidencia de la seguridad computacional mediante la reducción de la seguridad del criptosistema a algún problema conocido o bien estudiado.

Por ejemplo, puede ser capaz de probar una declaración del tipo "dado un criptosistema, este es seguro si dado un número entero n no es factorizable", criptosistemas de este tipo son frecuentemente llamados "probablemente seguros", pero esto puede ser entendido como una aproximación, solo provee una prueba de seguridad relativa para otros problemas, no una prueba absoluta de seguridad. (Javier Areitio, 2008)

Seguridad ad hoc o heurística.

Variedad de argumentos convincentes referentes a que cada ataque con éxito necesita más recursos de los disponibles por un atacante. Los ataques imprevistos siguen siendo una amenaza. (Javier Areitio, 2008)

Seguridad de sistemas inseguros

La seguridad de los sistemas de cifrado actuales se puede observar dentro de un proceso dinámico entro el caos del criptoanálisis y la armonía de la Criptografía. No hay que olvidar que cuando el criptoanálisis no puede descifrar un criptosistema, siempre se puede acudir a la ingeniería social (espías, vulnerabilidades en el factor humano etc.) (Javier Areitio, 2008)

s

4.5 SEGURIDAD PERFECTA EN UN CRIPTOSISTEMA.

Definición

La seguridad del sistema debe depender únicamente de la privacidad de las claves, longitud y no del secreto de los algoritmos de cifrado y descifrado. No hay nada sobre el cifrado simétrico ni de clave pública que haga a uno superior al otro en lo que respecta a la resistencia al criptoanálisis.

Eje: cifrado por desplazamiento.

Método de desplazamiento. Se denominan de esta manera todos aquellos métodos que impliquen la sustitución de un carácter del alfabeto por otro desplazado x posiciones. Evidentemente se trata de una generalización del método de Julio Cesar en el cual la clave puede ser cualquiera en un rango entre 1 y n – 1, siendo n el número de elementos del alfabeto. El número de desplazamientos es la clave.

$C_k = m + k(\mathrm{mod}\, n)$ En el caso de $k = 3$ obtenemos el método de Cesar.

Este método es muy fácil de romper con la simple aplicación de técnicas estadísticas, una tabla de las posibles permutaciones o un estudio exhaustivo de las claves hasta encontrar la correcta, ya que solo hay n posibles. (Jose Soler, 2000)

4.5.1 Teorema de cifrado perfecto.

Se llama cifrado (o transformación criptográfica) a una transformación del texto original (texto plano, inicial) que lo convierte en el llamado texto cifrado o criptograma. Correlativamente, se llama descifrado a la transformación mediante la llave de seguridad la cual permite recuperar el texto original a partir del cifrado.

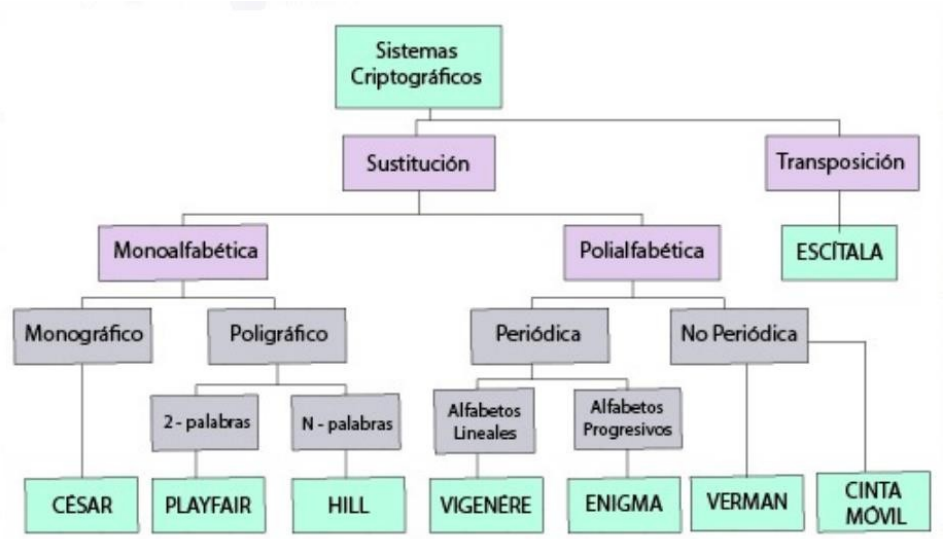


Fig. 29 Sistemas de cifra clásicos.

4.6 CIFRADO DE VERMAN.

En 1917 Gilbert S. Vernam, ingeniero del MIT que trabajaba en los laboratorios de la empresa AT&T, diseña un dispositivo criptográfico para comunicaciones telegráficas basado en los 32 códigos de Baudot que usaban los teletipos desarrollados por su compañía. Estos códigos representan los caracteres del lenguaje con cinco elementos que pueden ser el espacio o la marca (el cero y el uno) diseñado para transmisiones telegráficas. Es decir, un sistema binario con 5 dígitos: 2^5= 32 caracteres.

Este cifrador, que tuvo una gran aplicación durante la Primera Guerra Mundial, basa su seguridad en el secreto de una clave aleatoria que se supone tan larga o más que el mensaje y que luego de usarse debe destruirse. Este sistema por sustitución polialfabética es el caso límite del cifrado Vigenère. (Artech House, 1989)

Funcionamiento

De esta forma, el cifrador de Vernam genera un flujo de bits de texto cifrado de la forma:

$$C = E_k(M) = C_1C_2C_3 \ldots C_n$$

$$Donde\ C_1 = (m_i + k_i)\ mod\ 2 \quad para\ i = 1,2,\ldots,n;\ \ o\ sea, c_i = m_i\ XOR\ k_i$$

Mensaje:	Atacar el jueves trece a las seis horas	
Plano:	ATACARJUEVESTRECEASEISH	23 caracteres
Clave:	TCEGIKMOQSUWYBDFVABCDGT	23 caracteres

En código Baudot

Plano		Clave	Plano		Clave		Cifra		Cifra
A	XOR	T	00011	XOR	10000	=	10011	=	W
T	XOR	C	10000	XOR	01110	=	11110	=	V
A	XOR	E	00011	XOR	00001	=	00010	=	2
C	XOR	G	01110	XOR	11010	=	10100	=	H
A	XOR	I	00011	XOR	00110	=	00101	=	S
R	XOR	K	01010	XOR	01111	=	00101	=	S
J	XOR	M	01011	XOR	11100	=	10111	=	Q
U	XOR	O	00111	XOR	11000	=	11111	=	3
E	XOR	Q	00001	XOR	10111	=	10110	=	P
V	XOR	S	11110	XOR	00101	=	11011	=	4
E	XOR	U	00001	XOR	00111	=	00110	=	I
S	XOR	W	00101	XOR	10011	=	10110	=	P
T	XOR	Y	10000	XOR	10101	=	00101	=	S
R	XOR	B	01010	XOR	11001	=	10011	=	W
E	XOR	D	00001	XOR	01001	=	01000	=	1
C	XOR	F	01110	XOR	01101	=	00011	=	A
E	XOR	V	00001	XOR	11110	=	11111	=	3
A	XOR	A	00011	XOR	00011	=	00000	=	6
S	XOR	B	00101	XOR	11001	=	11100	=	M
E	XOR	C	00001	XOR	01110	=	01111	=	K
I	XOR	D	00110	XOR	01001	=	01111	=	K
S	XOR	G	00101	XOR	11010	=	11111	=	3
H	XOR	T	10100	XOR	10000	=	00100	=	5

Cifrado	WV2HS SQ3P4 IPSW1 A36MK K35

Fig. 30 Funcionamiento Cifrador Vernam

Para la operación de descifrado utilizaremos el mismo algoritmos por la propiedad invlutica de la operación OR exclusico, esto es:

C_i XOR k_i = (m_i XOR k_i)

Como k_i XOR k_i = 0 para k_i = 0 y k_i=1 ; se obtiene c_i XOR k_i = m_i

La ventaja de esta operación es que el proceso de cifrado y descifrado es el mismo, variando únicamente la cinta a utilizar. “Mediante el analisis del cifrado de Vernam se concluye en la base de un sistema perfecto de cifrado el cual se dispone de una cinta que posee una clave aleatoria tan larga como el mensaje a cifrar y que no se puede repetir jamás, lo cual hace imposible de violar sin importar la capacidad del computador”

Este método es conocido con el nombre de one time pad. El One-Time Pad (OTP) es el único cifrado que ha sido probado como incondicionalmente seguro, e inquebrantable en la práctica. Ha sido probado también que cualquier cifrado inquebrantable e incondicionalmente seguro debe ser en principio un One-Time Pad

¿Porque es inviolable este cifrado?

De acuerdo con Alfred Menezes en su libro, Handbook of Applied Cryptography (Manual de criptografía aplicada), se puede decir que un sistema es totalmente secreto, o incondicionalmente seguro, cuando el texto cifrado que se observa no proporciona información complementaria acerca de la cadena de texto en claro original. Si suponemos que L es el número de bits en la cadena de texto en claro, i va de 1 a L en las siguientes definiciones:

pi = el bit n° i en la cadena de texto en claro

ci = el bit n° i en la cadena de texto cifrado

ki = el bit n° i en la cadena de clave

P (pi) = la probabilidad de que pi se ha enviado

P (pi | ci) = la probabilidad de que pi se ha enviado dado que ci se ha observado

Se puede decir que un sistema es perfectamente secreto cuando P (pi) = P(pi | ci). En los sistemas de cifrado de flujo tradicionales, el método más común de mezclar bits de datos de texto en claro con bits de clave es a través de la operación XOR en los bits correspondientes. El hecho de que la clave sea completamente aleatoria nos lleva a varias conclusiones. La probabilidad de observar un bit de la clave es la misma que la de cualquier otro bit y el hecho de conocer valores previos de la clave no nos aporta nada sobre valores futuros. (Alfred Menezes, 1997)

Usa una clave constituida por una sucesión de símbolos (bits o caracteres) llamada serie cifrante, operando o-exclusivo XOR cada símbolo de ésta con el correspondiente del texto en claro. Debido a la definición de la función XOR, el descifrado se realiza, igualmente, operando con dicha función

cada bit de la misma serie cifrante con el correspondiente del texto cifrado. Si la serie cifrante no se repite, es aleatoria, y de longitud igual o mayor al texto a cifrar, éste cifrado alcanza el secreto perfecto. Además, es el único que verifica tal condición. Cada carácter mi se representa con 5 bits en código Baudot que se suma OR exclusivo con la correspondiente clave ki de una secuencia binaria aleatoria.

Algunos inconvenientes

Requiere libretas de un solo uso perfectamente aleatorias.

La generación e intercambio de las libretas de un solo uso tiene que ser segura, y la libreta tiene que ser al menos tan larga como el mensaje.

Hace falta un tratamiento cuidadoso para asegurarse de que siempre permanecerán en secreto para cualquier adversario, y es necesario deshacerse de ellas correctamente para evitar cualquier reutilización parcial o completa.

El problema práctico es que la llave no tiene un tamaño pequeño y constante, sino que tiene el mismo tamaño del mensaje, y una parte de la llave no se debería usar nunca dos veces (o el cifrado podría romperse). De esta manera, hemos trasladado el problema de intercambiar datos secretos por el problema de intercambiar llaves aleatorias secretas de la misma longitud. Sin embargo, este cifrado, supuestamente, ha estado en uso generalizado desde su invención, y mucho más desde la prueba de seguridad por Claude Shannon en 1949. Aunque la verdad es que la seguridad de este cifrado fue conjeturada anteriormente, fue Shannon quien encontró como probarla formalmente. (Claude Shannon, 1949)

4.7 ENTROPÍA DE TEXTOS PLANOS Y CIFRADOS.

4.7.1 FORMAS DE REPRESENTAR UN TEXTO

Para representar un texto en una computadora, los caracteres deben ser traducidos a una forma binaria. Una manera de expresar estos caracteres es utilizando el código ASCII. En este código, cada carácter es representado por siete bits. Por ejemplo, el carácter “a” es representado como la secuencia 1100001.

Otra manera de expresar un texto es por medio del código Morse. El mismo consiste en representar los caracteres a través de puntos y rayas. El código Morse es más eficiente que el código ASCII ya que asigna códigos cortos a los caracteres de mayor frecuencia y códigos largos a los de menor frecuencia. El resultado es que en promedio menor cantidad de bits son utilizados

Para codificar un determinado texto. El criterio de asignación de códigos se basa en el análisis de la frecuencia de aparición de los caracteres del alfabeto inglés. Esta diferencia nos lleva a analizar el concepto de eficiencia en la codificación que está íntimamente ligada con la Entropía.

Definición.

La Entropía se define como la cantidad mínima promedio de bits necesaria para representar un mensaje.

La Entropía es un factor fundamental al evaluar la eficiencia de un código, ya que en el mejor de los casos, un método de codificación utilizará en promedio una cantidad de bit igual a H(X), mientras que en la generalidad de las veces, se tendrá un valor superior a dicha cantidad. La codificación de Huffman es un método óptimo para la compresión de textos, es decir, el promedio de la longitud de bits del mensaje codificado se aproxima en gran medida al valor de la Entropía lo que hace que sea el método de codificación más eficiente. (Gabriel Silvester, Luciano Gregorio, Jorge Skigin, 1998)

4.7.2 TEOREMA DE ENTROPÍA

$$H(X) = -\sum_{i=1}^{n} p(X = x_i) \log_2 p(X = x_i)$$

$$H(X \mid y) = -\sum_{x} p(x \mid y) \log_2 p(x \mid y)$$

$$H(X \mid Y) = -\sum_{y} \sum_{x} p(y) p(x \mid y) \log_2 p(x \mid y)$$

Teorema. $H(X,Y) \leq H(X) + H(Y)$
$H(X,Y) = H(X) + H(Y)$ sii X e Y son independientes

Teorema. $H(X,Y) = H(Y) + H(X \mid Y) = H(X) + H(Y \mid X)$

Corolario. $H(X \mid Y) \leq H(X)$
$H(X \mid Y) = H(X)$ sii X e Y son independientes

Teorema de entropía (José Sempere, 2006)

En el teorema de entropía se presentan dos teoremas y un corolario los mismos que están en función de dos variables, las cuales son independientes.

La compresión de Huffman

La codificación de Huffman fue inventada por D. A. Huffman en el año 1952. Es un método de compresión estadístico, basado en la observación previa de lo que se va a comprimir. La compresión de Huffman se basa en la sustitución de cadenas de caracteres por códigos que ocupan menos espacio. Como la mayoría de los métodos de compresión, busca las cadenas de datos que se repitan más veces en una secuencia: cuanto mayor sean estas cadenas y más se repitan mayor será el grado de compresión, pero por lo general esto no ocurre así. Estadísticamente está comprobado que las cadenas grandes casi no se repiten dentro de una secuencia de datos. Es más probable encontrar cadenas pequeñas que se repitan. (Gabriel Silvester, Luciano Gregorio, Jorge Skigin, 1998)

El algoritmo de Huffman

Lo primero que se hace es estudiar o fabricar una tabla de frecuencias de los caracteres del texto plano, ordenando los mismos de menor a mayor frecuencia.

Después se hace un árbol binario. Un árbol binario es una estructura de datos en la cual el nodo raíz apunta a dos nodos, denominados hijos y cada uno de ellos apunta a su vez a otros dos nodos. Esta estructura se repite hasta llegar a nodos especiales llamados hojas, los cuales tienen la característica de no poseer hijos. Las ramas de la izquierda se representan con un 0, mientras que las de la derecha se representan con un 1. El árbol se construye de abajo hacia arriba siguiendo todos estos pasos. Todos los caracteres utilizados en el texto y sus respectivas frecuencias quedan finalmente almacenados en las hojas del árbol.

Se toman los dos nodos de menor frecuencia y se colocan en una estructura de árbol mínima, es decir, un nodo raíz y sus hijos. Los hijos contienen los caracteres de menor frecuencia dentro del texto plano. La raíz contendrá como información, la suma de las frecuencias de sus hijos.

A continuación, se escogen otros dos nodos de menor frecuencia excluyendo los dos utilizados anteriormente e incluyendo el nodo raíz generado en el paso anterior dando lugar a un nuevo árbol. El nodo raíz nuevamente contendrá la suma de frecuencias de sus dos nodos hijos.

Este procedimiento se repite hasta llegar a la formación de un único árbol el cual contiene tantas hojas como cantidad de caracteres distintos posea el texto analizado.

En ésta instancia se está en condiciones de obtener la forma óptima de codificación del texto plano. Se puede obtener el código asociado a cada carácter recorriendo el árbol generado anteriormente desde la hoja que corresponde al carácter buscado hasta la raíz.

El árbol obtenido depende del texto plano con lo que textos distintos producirá árboles distintos.

La descompresión posterior de los datos se realiza siguiendo la estructura de árbol. Se ingresa por la raíz y de acuerdo a la codificación del carácter queda unívocamente determinado el camino que lleva hasta la hoja que contiene el carácter del texto plano.

El árbol diseñado para la compresión de archivos se almacena en el propio archivo, para saber cómo interpretar los datos allí almacenados a la hora de descomprimir. Los datos comprimidos se almacenan en formato binario y se graban en bytes. Esto hace que los datos comprimidos no finalicen exactamente en los límites de un byte. Para solventar este problema se puede llenar con ceros ese espacio restante hasta completar el byte. Aunque el inconveniente principal es que se pueden obtener caracteres extra al descomprimir, debido a la incorrecta interpretación de esos ceros.

Una solución más elegante consiste en incluir un código especial para indicar el final de la serie de datos. Esto plantea una reorganización del árbol de codificación y del propio algoritmo de compresión, que debe detectar ese código “artificial” indicativo del final de datos.

Otra solución es incluir la longitud del texto plano dentro del encabezado del archivo comprimido. A medida que se descomprime se lleva la cuenta de los bytes generados hasta llegar al valor indicado.

4.8 REDUNDANCIA DEL LEGUAJE.

En ingeniería de computadores, son aquellos en los que se repiten aquellos datos o hardware de carácter crítico que se quiere asegurar ante los posibles fallos que puedan surgir por su uso continuado.

Se presenta como una solución a los problemas de protección y confiabilidad. Este tipo de sistemas se encarga de realizar el mismo proceso en más de una estación, ya que si por algún motivo alguna dejara de funcionar o colapsara, inmediatamente otro tendría que ocupar su lugar y realizar las tareas del anterior.

Las técnicas de redundancia han sido usadas por la industria militar y aeroespacial por muchos años para alcanzar una alta confiabilidad. Una base de datos replicada es un ejemplo de sistema distribuido redundante. (Dominic Welsh 1988)

Definición.

En teoría de la información, la redundancia es una propiedad de los mensajes, consistente en tener partes predictibles a partir del resto del mensaje y que, por tanto, en sí mismo no aportan nueva información o "repiten" parte de la información.

Descriptivamente, la redundancia constituye un factor comunicativo estratégico que consiste en intensificar, subrayar y repetir la información contenida en el mensaje a fin de que el factor de la comunicación ruido no provoque una pérdida fundamental de información.

La redundancia D del lenguaje será la diferencia entre la ratio absoluta y la ratio real:

D = R –r

3.25<D<3.55

Esto nos muestra el número de bits extras (bits redundantes) necesarios para codificar un mensaje suponiendo un alfabeto de 27caracteres (codificación con 5 bits puesto que 2^5 = 32 y 2^4= 16) será aproximadamente igual a 3.5

D/R seráun factor proporcional, luego:

% Red. Lenguaje (D/R) < 74.73

Distancia de unicidad y claves espurias.

Para un cifrador la distancia de unicidad, también llamada punto de unicidad, es el valor mínimo de caracteres del texto cifrado que se necesitan para reducir a una el número de claves posibles y, por tanto, romper el cifrado. Es decir, después de intentar todas las posibles claves (mediante fuerza bruta) sólo hay una que puede hacer que el descifrado tenga sentido. La distancia de unicidad es posible a causa de la redundancia de los idiomas humanos. El concepto de distancia de unicidad fue introducido por C. E. Shannon.

La distancia de unicidad permite medir el secreto de un cifrador a partir de la cantidad de incertidumbre (entropía) de la clave condicionada por el conocimiento del texto cifrado:

(Hc(K)). Si Hc(K)=0

Entonces no hay incertidumbre y el cifrador es teóricamente rompible teniendo los suficientes recursos. La distancia de unicidad es la longitud mínima del texto cifrado que se necesita para determinar de forma única la clave.

Un cifrador se dice que es incondicionalmente seguro si (Hc(K)) nunca se aproxima a 0 incluso para longitudes largas de texto cifrado. De la propia definición de secreto perfecto se puede concluir que un cifrador que tiene secreto perfecto no tiene distancia de unicidad y por tanto es incondicionalmente seguro. Shannon usaba el término secreto ideal para describir sistemas que no logran el secreto perfecto pero sin embargo no se pueden romper porque no dan suficiente información para determinar la clave.

“ El valor de la distancia o punto de unicidad para un cifrador que se desarrolla en un idioma redundante es indispensable, pues mediante el concepto del mismo se puede reducir a tan solo una la clave del texto cifrado, permitiendo así que la clave sea descifrada, quitándole la característica de ser inquebrantable.

Para que un cifrador sea totalmente seguro nunca debe aproximarse a 0, también contiene secreto perfecto dentro de su peculiaridad, desencadenando a consecuencia que no tenga distancia de unicidad, con estas características se convierte en irrompible; incl uso usando el secreto ideal de Shannon se fracasaría por escases de información para establecer la clave.”

Claves espurias.

Se refiere a una relación matemática en la cual dos acontecimientos no tienen conexión lógica, aunque se puede implicar que la tienen debido a un tercer factor no considerado aún (llamado "factor de confusión" o "variable escondida"). La relación espuria da la impresión de la existencia de un vínculo apreciable entre dos grupos que es inválido cuando se examina objetivamente

Ejemplo

Un ejemplo estadístico de una relación espuria puede ser ilustrado examinando las ventas de helados de una ciudad. Estas son más altas cuando la tasa de sofocamientos es mayor. Sostener que la venta de helados causa los sofocamientos sería implicar una relación espuria entre las dos. En realidad, una ola de calor puede haber causado ambas. La ola de calor es un ejemplo de variable escondida. (José Sempere, 2006)

Teoremas de claves espurias (falsas) y distancia de unicidad.

Claves falsas y distancia de unicidad (I)

Equivocación de clave $$H(K|C) = -\sum_{c \in C} \sum_{k \in K} p(k,c) \log_2 p(k|c)$$

Entropía de un lenguaje L $$H_L = \lim_{n \to \infty} \frac{H(P^n)}{n}$$

Redundancia de un lenguaje L $$R_L = 1 - \frac{H_L}{\log_2 |P|}$$

Claves consistentes con un texto cifrado y

$$K(y) = \{k \in K \mid \exists x \in P^n, p_{P^n}(x) > 0, e_k(x) = y\}$$

Claves falsas y distancia de unicidad (II)

Número de claves falsas en promedio $\overline{s_n} = \sum_{y \in C^n} p(y)\,|K(y)| - 1$

Teorema **Si** $|C| = |P|$ **y** $p_K(k) = \frac{1}{|K|}$ **entonces** $\overline{s_n} \geq \frac{|K|}{|P|^{nR_L}} - 1$

Distancia de unicidad n_0

$$n_0 \approx \frac{\log_2 |K|}{R_L \log_2 |P|}$$

(José Sempere, 2006).

5 METODO CLASICO DE CIFRADO

5.1 DEFINICIÓN

O también llamado como CIFRADO DE CESAR Y SU CRIPTOANÁLISIS. El conocido método de Cesar. Este método fue el empleado por Julio Cesar en sus campañas durante el siglo 1 a.e. para transmitir información en secreto. Era, como decimos, una sustitución que consistía en cifrar un mensaje empleando un alfabeto equivalente al original, pero desplazado en 3 letras. Es decir, al cifrar por ejemplo las palabras GALOS IRREDUCTIBLES con este algoritmo, Cesar obtendría:

JDÑRVLUUHGXFWLEÑHV

El primer criptoanálisis aplicable sería el método de complementación al componente original, es decir, probar las 27 permutaciones posibles obtenidas como resultado de desplazar una posición cada vez las letras de una palabra del criptograma[5]. Por ejemplo, para JDÑRV, analizando los 27 resultados es más que probable que encontremos una única palabra que tenga significado en español, con lo cual solo tendríamos que utilizar el mismo desplazamiento obtenido para descifrar el resto del texto.

El segundo criptoanálisis[6] aplicable a este tipo de cifrado es el análisis de frecuencias. De una forma un tanto más elegante que el anterior, mediante este método, estudiamos la frecuencia relativa de aparición de las diferentes letras del texto cifrado, para compararlas con una descripción estadística que hayamos obtenido del lenguaje, en el que sospechemos o sepamos que se encuentra el mensaje original También se puede realizar un estudio sobre las 95 palabras más usadas, o sobre los diagramas y trigramas que constituyen el inicio y terminación más frecuente de las palabras de un lenguaje.

[5] *Criptogramas: Se llama criptograma al mensaje que fue escrito utilizando algún tipo de clave*

[6] *Criptoanálisis*: *El criptoanálisis es la ciencia opuesta a la criptografía, ya que, si ésta trata principalmente de crear y analizar criptosistemas seguros, la primera intenta romper esos sistemas, demostrando su vulnerabilidad*

Por ejemplo, para un texto en español lo suficientemente representativo, se ha obtenido la siguiente distribución de frecuencias:

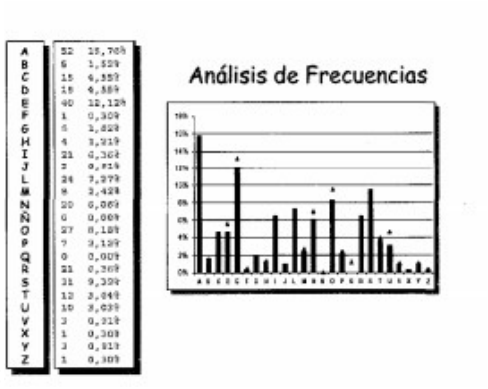


Fig. 31 Análisis de frecuencia de criptograma(Garc, Criptograf, & Surgi, n.d.)

Este tipo de criptoanálisis es terriblemente efectivo para cifrado monoalfabético. Al emplearlo, se letras por otras siguiendo siempre la misma congruencia lineal, hace que las propiedades estadísticas d (1l criptograma y del texto en claro sean exactamente las mismas. Simplemente hay que llevar a cabo un análisis estadístico de los símbolos del criptograma e intentar solaparlo o encajarlo con la distribución de los símbolos de nuestro idioma De esta forma, hallaremos el desplazamiento que fue aplicado al cifrar el texto original y podremos descifrar, inmediatamente, el resto del mensaje. (Garc et al., n.d.)

Este método fue empleado por Julio Cesar en sus campañas durante el siglo 1 a.e. para transmitir información en secreto. Era, como podremos decir, una sustitución que consistía en cifrar un mensaje empleando un alfabeto equivalente al original, pero desplazado en 3 letras en este tipo de cifrado podemos apreciar un análisis de frecuencias. De una forma un tanto más elegante que el anterior, atreves de este método, estudiamos la frecuencia relativa de aparición de las diferentes letras del texto cifrado, para compararlas con una descripción

Por Sustitución

En criptografía, el cifrado por sustitución es un método de cifrado por el que unidades de texto plano son sustituidas con texto cifrado siguiendo un sistema regular; las "unidades" pueden ser una sola letra (el caso más común), pares de letras, tríos de letras, mezclas de lo anterior, entre otros. El receptor descifra el texto realizando la sustitución inversa. Los cifrados por sustitución son comparables a los cifrados por transposición. En un cifrado por transposición[7], las unidades del texto plano son cambiadas usando una ordenación diferente y normalmente bastante

[7] *Cifrado por Transposición: El método de cifrado por transposición consiste en reordenar datos para cifrarlos a fin de hacerlos ininteligibles*

compleja, pero las unidades en sí mismas no son modificadas. Por el contrario, en un cifrado por sustitución, las unidades del texto plano mantienen el mismo orden, lo que se cambia son las propias unidades del texto plano. Existen diversos tipos de cifrados por sustitución. Si el cifrado opera sobre letras simples, se denomina cifrado por sustitución simple; un cifrado que opera sobre grupos de letras se denomina, poligráfico.

Tradicionalmente los textos cifrados se escribían en bloques de igual longitud, omitiendo los signos de puntuación y los espacios; esto tenía dos efectos: permitía una transmisión más eficiente libre de errores y evitaba distinguir las palabras por los contornos. Estos bloques se denominaban "grupos", y a veces un "conteo de grupost" (es decir el número de grupos) proporcionaba una forma de chequeo adicional. Por ejemplo, los grupos de cinco letras eran tradicionales, de la época del telégrafo: SIAAZ QLKBA VAZOA RFPBL UAOAR Si la longitud del mensaje ocurría que no era divisible entre cinco, se podía entonces rellenar con ceros hasta el final. O con caracteres que no dieran sentido obvio al texto, de esta forma el receptor no podía fácilmente descartarlos. El alfabeto empleado en el cifrado es a veces diferente al del alfabeto originario; por ejemplo, en el cifrado francmasón, el texto cifrado consiste en un conjunto de símbolos derivados de una red como, por ejemplo: Tales características hacen más seguro el cifrado ya que el descifrador tiene que buscar más posibilidades para encajar el texto en un alfabeto A-Z

Tipos de Sustitucion

Sustitucion Monoalfabetica

Siendo la clave secreta la cadena de 26 letras correspondiente al alfabeto completo. Para la clave anterior, el texto plano “stri” se transformaría en el texto cifrado “LZKO”.

Este sistema puede parecer seguro, porque, aunque el criptoanalista conozca el sistema general (sustitución letra por letra), no sabe cuál de las 26! $\cong 4X10^{26}$claves posibles se está usando. Intentar una por una no es una alternativa recomendable. A 1 nanosegundo por solución, una computadora tardaría 10^{10}años en probar todas las claves.

No obstante, si se cuenta con una cantidad pequeña de texto cifrado, puede descifrarse fácilmente aprovechando la estadística lingüística(Unidos, Guerra, & Cada,n.d.)

Sustitucion por alfabetos independientes

En este caso se generan dos alfabetos totalmente independientes haciéndose la sustitución de los caracteres cuyas posiciones coincidan. Aunque el número de posibles permutaciones es muy elevado, el criptoanálisis de este método, al igual que el de todos los de sustitución

monoalfabética, es muy sencillo basándonos en las propiedades estadísticas del lenguaje. Uno de los problemas que presenta este método es la dificultad de memorizar los alfabetos. Existe sin embargo una forma sencilla de generar un alfabeto de sustitución sin la necesidad de memorizar las equivalencias entre los mismos. Para ello basta con utilizar una palabra como clave. El alfabeto se genera poniendo la palabra clave al principio eliminando las letras duplicadas y poniendo a continuación el resto del alfabeto normal en un orden prefijado. Por ejemplo, si utilizamos la palabra seguridad como clave, el alfabeto generado sería:

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

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

Una variante más segura de este método es el de distribuir los caracteres del alfabeto en forma de tabla y coger los datos en columnas bien en orden posicional o en orden alfabético.(Soler Fuensanta, 2012)

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

Soler Fuensanta, (2012)

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

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

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

Sustitucion Homofóbica

El principal problema de los métodos de substitución monoalfabéticos es el mantener las propiedades estadísticas del alfabeto original. Para eliminar este problema existen dos soluciones, la utilización de alfabetos homófonos[8] y la utilización de sistemas de sustitución polialfabeticos. La utilización de alfabetos homófonos tiene el inconveniente de ampliar el tamaño del mensaje,

[8] *Alfabetos Homófonos: Tiene el inconveniente de ampliar el tamaño del mensaje, característica no deseable, pero tiene la ventaja de eliminar las frecuencias características del lenguaje original,*

característica no deseable, pero tiene la ventaja de eliminar las frecuencias características del lenguaje original, y en algunos casos como la utilización de determinadas tablas de homófonos, la posibilidad de tener una entropía máxima del mensaje. Básicamente una sustitución homofónica es una correspondencia de uno a muchos en lugar de una correspondencia uno a uno típica de los métodos de sustitución monoalfabéticos. Se asigna a cada letra del alfabeto original un conjunto de elementos de un alfabeto ampliado o de elementos agrupados de un alfabeto. Los subconjuntos que se asignan a cada una de las letras del alfabeto original deben ser evidentemente disjuntos. Por ejemplo una asignación podría ser la siguiente.(Soler Fuensanta, 2012)

Texto en claro	Homófonos
A	01,05,23,12,17
B	18,03,56
C	22,25,02
D	09,27
E	06,26,87,54,36

Soler Fuensanta, (2012)

en el primer caso, la primera aparición de la letra en claro A daría como resultado el 01 como cifra, la segunda aparición el 05 y así sucesivamente hasta la finalización del conjunto de homófonos para esa letra, punto a partir del cual se empezaría el proceso otra vez con el 01. En este caso se habla de cifrado homofónico secuencial. En el caso de escoger aleatoriamente cualquier homófono se habla de cifrado homofónico aleatorio. Un tipo de sustitución homofónica muy interesante es la formada por las tablas de homófonos. En este caso el alfabeto se representa por una tabla de homófonos que serán los que formarán el mensaje cifrado, siendo la clave la tabla igual que en el caso anterior y la dirección de cifrado (horizontal o vertical). Con estas tablas se puede conseguir una entropía máxima del contenido del mensaje. Para ello se coge la frase a cifrar y una frase cuyo significado sea el contrario, se escriben una encima de la otra y se utiliza una para las filas y otra para las columnas. El cifrado será la intersección de ambas letras. La entropía es máxima en este caso pues si el descifrado se realiza verticalmente dará un resultado, y si se hace horizontalmente dará el contrario. Si no se conoce la dirección de cifrado, es imposible estar seguro de cuál es el mensaje real. Sea por ejemplo la matriz siguiente:

	T	O	D	N	A

T	012	015	023	056	152
O	142	136	825	651	874
D	354	625	177	853	444
N	123	145	156	167	198
A	199	299	399	499	599

Soler Fuensanta, (2012)

Mensaje: TODO

Mensaje falso: NADA

Filas T O D O

Columnas N A D A

Cifrado 056 874 177 874

En el caso hipotético de que el criptoanalista obtuviese la tabla, no podría estar seguro del mensaje, ya que si se escogen las columnas el mensaje da por resultado NADA y si se escogen las filas el resultado es TODO. Solo podría obtenerse la solución correcta en el caso de que se supiese con certeza cual es la dirección de cifrado.(Soler Fuensanta, 2012)

5.2 SUSTITUCION POLIALFABÉTICA

En todos los cifrados anteriores se utilizaba un solo alfabeto para cifrar, lo que permitía el análisis del cifrado por métodos estadísticos al conservar el texto cifrado las características básicas del alfabeto original. Un método para evitar este problema es utilizar varios alfabetos de forma que se disimulen las características del lenguaje fuente. Esta es la base de los sistemas denominados polialfabéticos.(Soler Fuensanta, 2012)

5.3 SUSTITUCION BIALFABÉTICA

En el siglo II a.C. el historiador griego Polibio describe un sistema de señales a distancia basado en el empleo de antorchas. Para evitar posibles accidentes puede ser conveniente que en el aula estas antorchas sean sustituidad mediante un gráfico con diez círculos que, a modo de semáforo pueden cambiar de color (o quizás podría ser entretenido sustituirlas por diez linternas) Se requiere de un alfabeto con 25 letras así que podemos entender Q=K y que V=W con las veinticinco letras construimos un damero de la siguiente forma: (Casteñeda, C., Pino, G.1997)

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

Casteñeda, Pino, G.1997

Entre los diferentes tipos de sustitución podremos apreciar métodos

5.4 POR AFÍN

Los procesos de cifrado en los cuales el alfabeto de cifrado consiste en desplazar cierta cantidad de caracteres, son enmarcados bajo el nombre de cifrado afín. Los cifrados afines son basados en aritmética modular, sin embargo, en un alfabeto de n caracteres tendremos solo n posibles cifrados afines, esto hace que un ataque por fuerza bruta sea factible. En general, asignando valores entre 0 y n−1 para un alfabeto de n caracteres, con una clave de cifrado dada, el cifrado afín será de la siguiente forma:

$C \equiv M + \text{clave} \pmod{n}$

Donde M es el carácter en el mensaje, C es el correspondiente carácter cifrado Para poder obtener el mensaje original basta correr el alfabeto hacia la izquierda el número que indique la clave. A continuación, veremos algunos ejemplos del cifrado afín

Ejemplo 6. Cifrar el mensaje: “Una noche, una noche toda llena de perfumes, de murmullos y de músicas de alas”.

Desplazando UNA letra el alfabeto.

Con lo cual la relación entre los caracteres será:

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

Luego, el mensaje y el mensaje codificado serán:

Una noche, una noche toda llena de perfumes, de murmullos y de músicas de alas

Vob opdif,

vob opdif upeb mmfob ef qfsgvnft, ef nvsnvmmpt z ef nvtjdbt ef bmbt

Desplazando DOS letras el alfabeto

Con lo cual la relación entre los caracteres será:

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

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

Luego, el mensaje y el mensaje codificado serán:

Una noche,

una noche toda llena de perfumes, de murmullos y de músicas de alas

Wpc pqejg,

wpc pqejg vqfc nngpc fg rgthwogu, fg owtownnqu a fg owukecu fg cncu

Desplazando QUINCE letras el alfabeto

Con lo cual la relación entre los caracteres será:

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

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

Luego, el mensaje y el mensaje codificado serán:

Una noche,

una noche toda llena de perfumes, de murmullos y de músicas de alas

Jcp cdrwt,

jcp cdrwt idsp aatcp st etgujbth, st bjgbjaadh n st bjhxrph st paph

El cifrado afín más famoso es aquel en el que se desplazan 3 caracteres, conocido como el cifrado Cesar (en honor a Julio Cesar)1 , esta cifra fue empleada para enviar mensajes a sus tropas. Utilizando el cifrado Cesar obtenemos Una noche, una noche toda llena de perfumes, de

murmullos y de musicas de alas Wpc pqejg, wpc pqejg vqfc nngpc fg rgthwogu, fg owtownnqu a fg owukecu fg cncu (Triana,2011)

En el método de Afín consiste en desplazar cierta cantidad de caracteres, que son enmarcados bajo el nombre de cifrado afín. Los cuales son basados en aritmética modular, sin embargo, en un alfabeto de n caracteres tendremos solo n posibles cifrados afines, esto nos permite hacer que un ataque por fuerza bruta sea factible.

5.5 Por Vigenere

El diplomático francés Blaise de Vigenère en su obra de 1586 Traité des chiffres ou secrètes manières décrire propone la utilización de una tabla cuadrada de alfabetos regulares. Este método fue utilizado durante siglos en los ejércitos de diferentes países, entre otros por el ejército confederado en la guerra de secesión en Estados Unidos, considerándose indescifrable hasta la aparición del libro de Kasiski en 1863.(Soler Fuensanta, 2012)

El cifrado de Vigenère está basado en el cifrado del Cesar, por lo cual es un cifrado de sustitución. A diferencia del cifrado del cesar, en el cual cada símbolo del texto plano le es sumada una constante k, en el cifrado de Vigenère se tiene un cifrado del Cesar por cada símbolo de una palabra clave. Con lo cual, si la palabra clave tiene una longitud m, se tienen m corrimientos diferentes sobre el texto encriptado. De esta forma, no siempre un mismo símbolo en el texto claro se convierte en el mismo símbolo en el texto encriptado(Gómez et al., 2012)

Es utilizado durante siglos en los ejércitos de diferentes países, entre otros por el ejército confederado en la guerra de secesión en Estados Unidos, considerándose indescifrable hasta la aparición del libro de Kasiski en 1863 a diferencia de otros métodos como el cifrador cesar en el cual cada símbolo del texto plano le es sumada una constante k, en el cifrado de Vigenère se tiene un cifrado del Cesar por cada símbolo de una palabra clave.

5.6 Por Hill

Lester Hill (1891-1961) era un profesor de matemáticas que entró en la historia de la criptografía por un artículo aparecido en 1929 en The American Mathematical Monthly con el título "Cryptography in an algebraic alphabet". En este artículo Hill proponía por primera vez la utilización de ecuaciones en aritmética modular para cifrado de información. En 1931 en otro artículo proponía la utilización de matrices para el cifrado de información. Es un sistema criptográfico de sustitución polialfabético, es decir, un mismo signo, en este caso una misma letra, puede ser representado en un mismo mensaje con más de un carácter. Este método de sustitución poligráfica parte de la utilización de una matriz cuadrada invertible K que se utiliza como clave. El proceso de cifrado y descifrado es como sigue: (Hill, 1929)

CIFRADO

1) Codificar el mensaje en claro en forma numérica.

2) Dividir el mensaje en trozos de longitud l, siendo l el rango de la matriz K.

3) Realizar con los trozos del mensaje la operación C = M.K, siendo C el vector que contiene el texto cifrado y M el mensaje en claro.

DESCIFRADO

1) Dividir el mensaje cifrado en trozos de longitud l.

2) Calcular k^{-1} .

3) Hacer la operación k^{-1} . C =k^{-1} . K. M = I .M = M.

En realidad, el método de Hill no es excesivamente práctico, ya que las operaciones con matrices son lentas y además obligan en el caso de no utilizar dispositivos electrónicos a guardar la clave K en un medio físico. Sin embargo, es la primera aproximación seria de las matemáticas a la criptografía y quizás la primera vez que se utilizaba un problema matemático, fuera de los clásicos de la permutación de elementos, como medio para cifrar información.(Soler Fuensanta, 2012)

Podemos decir que el método de Hill a comparación de los otros métodos mencionados no es excesivamente práctico, ya que las operaciones con matrices son lentas y además obligan en el caso de no utilizar dispositivos electrónicos a guardar la clave K en un medio físico. Sin embargo, es la primera aproximación seria de las matemáticas a la criptografía y quizás la primera vez que se utilizaba un problema matemático, fuera de los clásicos de la permutación de elementos, como medio para cifrar información.

5.6.1 CIFRADO DE HILL POR GAUSS JORDAN

El método consiste en escribir una matriz 2N-grámica con los elementos del texto en claro y los elementos del criptograma. En esta matriz realizamos operaciones lineales (multiplicar filas por un número y restar filas entre sí) con el objeto de obtener los vectores unitarios. Por ejemplo, podemos romper la matriz clave K teniendo: (Aguirre,2006)

Por ejemplo, podemos romper la matriz clave K teniendo:

M = ENU NLU GAR DEL AMA NCH ADE CUY ONO ...

C = WVX IDQ DDO ITQ JGO GJI YMG FVC UÑT ...

$$\left(\begin{array}{ccc|ccc} E & N & U & W & V & X \\ N & L & U & I & D & Q \\ G & A & R & D & D & O \\ D & E & L & I & T & Q \\ A & M & A & J & G & O \\ N & C & H & G & J & I \\ A & D & E & Y & M & G \\ C & U & Y & F & V & C \\ O & N & O & U & Ñ & T \end{array}\right) = \left(\begin{array}{ccc|ccc} 4 & 13 & 21 & 23 & 22 & 24 \\ 13 & 11 & 21 & 8 & 3 & 17 \\ 6 & 0 & 18 & 3 & 3 & 15 \\ 3 & 4 & 11 & 8 & 20 & 17 \\ 0 & 12 & 0 & 9 & 6 & 15 \\ 13 & 2 & 7 & 6 & 9 & 8 \\ 0 & 3 & 4 & 25 & 12 & 6 \\ 2 & 21 & 25 & 5 & 22 & 2 \\ 15 & 13 & 15 & 21 & 14 & 20 \end{array}\right)$$

5.6.2 OPERACIONES EN LA MATRIZ DE GAUSS GORDAN

Vamos a dejar en la primera columna un número uno en la fila primera y todas las demás filas un cero. Luego multiplicamos el vector (4 13 21 | 23 22 24) por el inv (4, 27) = 7. Así obtenemos 7(4 13 21 | 23 22 24) mod 27 = (1 10 12 | 26 19 6). Si esto no se puede hacer con la primera fila movemos los vectores. Hecho esto vamos restando las filas respecto de esta primera como se indica: (Aguirre, 2006)

$$\left(\begin{array}{ccc|ccc} 4 & 13 & 21 & 23 & 22 & 24 \\ 13 & 11 & 21 & 8 & 3 & 17 \\ 6 & 0 & 18 & 3 & 3 & 15 \\ 3 & 4 & 11 & 8 & 20 & 17 \\ 0 & 12 & 0 & 9 & 6 & 15 \\ 13 & 2 & 7 & 6 & 9 & 8 \\ 0 & 3 & 4 & 25 & 12 & 6 \\ 2 & 21 & 25 & 5 & 22 & 2 \\ 15 & 13 & 15 & 21 & 14 & 20 \end{array}\right)$$

a) 2^a fila = 2^a fila – $13*1^a$ fila mod 27
b) 3^a fila = 3^a fila – $6*1^a$ fila mod 27
c) 4^a fila = 4^a fila – $3*1^a$ fila mod 27
d) 5^a fila ya tiene un 0
e) 6^a fila = 6^a fila – $13*1^a$ fila mod 27
f) 7^a fila ya tiene un 0
g) 8^a fila = 8^a fila – $2*1^a$ fila mod 27
h) 9^a fila = 9^a fila – $15*1^a$ fila mod 27

5.6.3 OPERACIONES DE LA MATRIZ DE GAUSS JORDAN

Repetimos este procedimiento ahora para algún vector en cuya segunda columna tenga un número con inverso en 27 y lo mismo para la tercera columna, moviendo si es preciso los vectores. Como la mitad izquierda de la matriz 2N era el texto el claro, la parte derecha de la matriz con vectores unitarios corresponderá a la traspuesta de la clave. (Aguirre, 2006)

$$\left(\begin{array}{ccc|ccc} 1&0&0&2&5&7\\ 0&1&0&3&5&8\\ 0&0&1&4&6&9\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0 \end{array}\right) \Rightarrow \quad K = \begin{pmatrix} 2&3&4\\ 5&5&6\\ 7&8&9 \end{pmatrix}$$

Compruebe que la clave es la utilizada en este cifrado.

El Metodo de cifrado de Hill se basa en distintas operaciones cada una describe diferentes maneras de realizar una encriptación mediante matrices

5.7 POR PERMUTACIÓN

Una permutación es la variacion del orden o de la disposicion de los elementos de un conjunto.De manera formal se puede decir que una permutacion de un conjunto X es una funcion biyectiva de dicho conjunto en si mismo.Recordando que una funcion biyectiva es aquella funcion en donde todos los elementos del conjunto de salida tienen una imagen distinta en el conjunto de entrada, y a cada elemento del conjunto de entrada le corresponde un elemnto del conjunto de salida.

5.7.1 PERMUTACION JV

Dada dos numeros enteros positivos n,m tales que $0 \leq n \leq m! - 1$, de acuerdo al algoritmo de las divisiones de Euclides el numero n se puede escribir

$n=C_0(m-1)! + C_1(m-2)! + \ldots\ldots + C_{m-2}(1)! + C_{m-1}(0)!$

Observamos que n pertenece al conjunto:$N_m=\{n \in N \ 0 \leq n \leq m! - 1\}$

Se puede comprobar $0 \leq C_i < (m - i)$, donde $0 \leq i \leq (m - 2)$

$N_m=\{n \in N \ 0 \leq n \leq m! - 1\}$ y

$$\pi_{m\{\pi \, \pi es \ una \ permutacion \ del \ arreglo \ 0,1,\ldots\ldots,m-1\}}$$

El algoritmo que se puede construir a partir de estas premisas es el siguiente:

Paso0: Se define un arreglo de orden creciente

$X[0] = 0$, $X[1] = 1$, $X[2]$ =2, $X[3]$ =3, , , $X[m-1]$ =m-1

Paso 1:DE 2 se tiene que $C_0 < m$, por lo tanto $X[C_0]$es uno de los elemntos del arreglo dado en el paso 0. $X[C_0]$ se elimina del arreglo y un nuevo arreglo se define desde $X[0]$hasta $X[m-2]$

Paso 2:Denueco, usando la expresion 2, se tiene que $C_1 < m - 1$;por lo tanto $X[C_1]$es uno de los elemntos definidos en el paso1.C ontinuando con el orden de las ideas del paso anterior, $X[C_1]$se elimina del arreglo definido en el paso 1 y un nuevo arreglo se construye desde $X[0]$ hasta $X[m-3]$

Paso m-1:Procediendo de forma repetitiva como en los pasos anteriores, se llega al final del proceso, debiendo tener el arreglo $X[C_{m-2}]$ y $[0]$ (Standard, 2015)

Método cifrado por Permutación podemos referirnos que es la variacion del orden o de la disposicion de los elementos de un conjunto.De manera formal se puede decir que una permutacion de un conjunto X es una funcion biyectiva de dicho conjunto en si mismo.

6 CRIPTOANÁLISIS Y ATAQUES DE SEGURIDAD

6.1 PRINCIPIOS DE KERCHHOFF

Los seis principios de KERCKHOFFS aplicados a la criptografía son:

1. El sistema debe ser en la práctica indescifrable, en caso de que no lo sea matemáticamente. 2. El sistema no debe ser secreto y no debe ser un problema que éste caiga en manos del enemigo.
3. La clave del sistema debe ser fácil de memorizar y comunicar a otros, sin necesidad de tener que escribirla; será cambiable y modificable por los interlocutores válidos.
4. El sistema debe poder aplicarse a la correspondencia telegráfica.
5. El sistema debe ser portable y su uso no deberá requerir la intervención de varias personas. 6. El sistema debe ser fácil de usar, no requerirá conocimientos especiales ni tendrá una larga serie de reglas (CRIPTO_RED, s.f.)

6.2 TIPOS DE ATAQUES

El Criptoanálisis (es uno de los dos componentes que conforman la Criptología junto a la Criptografía que se define como la ciencia que trata de escribir mensajes que nadie salvo el receptor deseado lo pueda leer) puede definirse como la ciencia que engloba un conjunto de técnicas tendentes a descifrar comunicaciones cifradas sin conocer las claves correctas, obtener el polinomio de realimentación de un PRNG a partir de la salida conocida, obtener el esquema interno de un determinado mecanismo criptográfico, etc. Existen muchas técnicas, las más importantes para un diseñador/implementador práctico de sistemas son:

6.2.1 ATAQUES SÓLO A TEXTO CIFRADO

El atacante no conoce nada acerca de los contenidos del mensaje y debe trabajar a partir sólo del texto cifrado. En la práctica a menudo es posible adivinar algo de texto sin cifrar ya que muchos tipos de mensaje tienen cabeceras de formato fijo. Incluso las cartas y documentos ordinarios empiezan de una forma muy predecible. También puede ser posible adivinar que algún bloque de texto cifrado contiene una palabra común.

6.2.2 ATAQUE A TEXTO SIN CIFRAR CONOCIDO

El atacante (criptoanalista, adversario o espía) conoce o puede adivinar/inferir el texto sin cifrar para algunas partes del texto cifrado. La tarea consiste en descifrar el resto de los bloques de texto cifrado utilizando esta información. Esto se puede hacer determinando la clave utilizada para cifrar los datos o utilizando algún “atajo”.

6.2.3 Ataque a Texto Sin Cifrar Elegido

El atacante puede tener algo de texto que quiere, cifrado con la clave desconocida. La tarea consiste en determinar la clave utilizada para el cifrado. Algunos métodos de cifrado (por ejemplo, el algoritmo de clave pública o asimétrico RSA) son extremadamente vulnerables a los ataques de texto sin cifrar elegido. Cuando tales algoritmos se utilizan debe tenerse extremo cuidado para diseñar todo el sistema de modo que un atacante nunca pueda conseguir texto sin cifrar elegido cifrado.

6.2.4 ATAQUE "MAN-IN-THE-MIDDLE"

Atacante mediante intromisión. El adversario se coloca en medio de las partes legítimas que se comunican o "meet-in-the-middle". Este ataque es relevante para protocolos de intercambio de claves y comunicación criptográfica. La idea es que cuando dos partes se intercambian claves para comunicaciones seguras (por ejemplo utilizando Diffie-Hellman) un adversario se coloca entre las partes en la línea de comunicaciones. El adversario entonces realiza un intercambio de clave separado con cada parte. Las partes finalizarán utilizando una clave diferente cada una de las cuales es conocida por el adversario. El adversario entonces descifrará las comunicaciones con la clave adecuada y las cifrará con la otra clave para enviarla a la otra parte. Las partes creerán que se están comunicando de forma segura, pero de hecho el adversario está escuchando y entendiendo todo. Una forma de prevenir los ataques de este tipo es que ambos extremos calculen una función criptográfica unidireccional hash (por ejemplo, MD5, SHA/SHA1, etc.) del intercambio de clave (o al menos de las claves de cifrado), la firmen utilizando un algoritmo de firma digital y envíen la firma al otro extremo. El receptor entonces verificará que la firma viene de la otra parte deseada y que el «hash» de la firma coincide con el calculado localmente. Este método se utiliza por ejemplo en Photuris.

6.2.5 ATAQUES DE TIEMPO

Este ataque es muy reciente y se basa en la medida repetida de los tiempos de ejecución exactos de las operacioes de exponenciación modular. Es relevante al menos a los métodos criptográficos RSA, Diffie-Hellman y de Curvas Elipticas. Las implementaciones de algoritmos criptográficos a menudo realizan cálculos en tiempo no constante, debido a optimizaciones del rendimiento. Si dichas operaciones implican parámetros secretos, estas variaciones de tiempo pueden fugar cierta información y si se proporciona suficiente conocimiento de las implementaciones, un cuidadoso análisis estadístico puede incluso conducir a la recuperación total de estos parámetros secretos. La idea inicial fue presentarda por Kocker en 1996 y demandaba que el atacante dispusiese de un conocimiento muy detallado de la implementación del sistema a atacar. Actualmente otros investigadores han mejorado el método de Kocker y no se re quiere un conocimiento tan detallado del sistema.

6.2.6 CRIPTOANÁLISIS DIFERENCIAL, LINEAL Y LINEAL-DIFERENCIAL

El criptoanálisis diferencial es un tipo de ataque que puede aplicarse a cifradores de bloque iterativos (como por ejemplo, los algoritmos simétricos o de clave secreta DES, 3DES, IDEA, etc.). Estas técnicas fueron introducidas por Murphy en un ataque sobre FEAL-4, pero fueron mejoradas y perfeccionadas posteriormente por Biham y Shamir que las utilizaron para atacar el DES. El criptoanálisis difere ncial es básicamente un ataque sobre texto sin cifrar escogido y se basa en un análisis de la evolución de las diferencias entre dos textos sin cifrar relacionados cuando se cifran bajo la misma clave. Mediante un cuidadoso análisis de los datos disponibles, las probabilidades pueden asignarse a cada una de las posibles

claves y eventualmente la clave más probable se identifica como la correcta. El criptoanálisis diferencial se ha utilizado contra cifradores muy grandes con grados de éxito variables. En ataques contra el DES su efectividad es limitada, por lo que fue muy cuidadoso el diseño de las cajas S durante el diseño del DES a mediados del año 1970. Nyberg, Knudsen Lai, Massey y Murphy han realizado estudios sobre protección de cifradores contra criptoanálisis diferencial. El criptoanálisis diferencial ha sido útil en ataques a otros algoritmos criptográficos como por ejemplo las funciones criptográ- ficas unidireccionales «hash». PROTOCOLO, TARJETA INTELIGENTE, ETC. PREGUNTA RESPUESTA SECRETO DIFERENCIA DE TIEMPO El criptoanálisis lineal fue ideado por Matsui y Yamagishi para un ataque sobre FEAL. Fue extendido por Matsui para atacar al DES. Es un ataque de texto sin cifrar conocido y utiliza una aproximación lineal para describir el comportamiento del cifrador de bloque. Dados suficientes pares de texto sin cifrar y los correspondientes textos cifrados, se pueden obtener los bits de información acerca de la clave y cantidades crecientes de datos darán normalmente una elevada probabilidad de éxito. Existen una variedad de mejoras al ataque básico. Langford y Hellman introdujeron un ataque denominado criptoanálisis lineal-diferencial que combina los elementos del criptoaná- lisis diferencial con los del criptoaná- lisis lineal. Así mismo, Kaliski y Robshaw demostraron que un ataque criptoanalítico lineal utilizando múltiples aproximaciones, puede permitir una reducción de la cantidad de datos requerido para un ataque con éxito. Nyberg, Knudsen y O'Conner han estudiado la protección de cifradores contra ataques criptoanalíticos lineales.

6.2.7 ATAQUES DE DICCIONARIO

Utilizados para romper palabras de paso de los sistemas operativos. En el «ataque de diccionario» no se pretende obtener la clave, si no directamente el texto en claro ya que el método de cifrado es público y aunque no se disponga de la clave se puede reproducir. Este es el caso de los sistemas de cifrado de claves de acceso en sistemas operativos. Cuando un usuario quiere darse de alta en un sistema, introduce su código y su clave de acceso, ésta se cifra y posteriormente se compara el resultado con la clave cifrada que se almacena en el fichero de claves. Si son iguales, el sistema considera que el usuario es quien dice ser y le permite el acceso. Los programas rompedores de "palabras de paso" parten del hecho de que se ha obtenido una copia del fichero de claves (por ejemplo, el denominado /etc/passwd de Unix) y comprueban si existe alguna cuenta sin clave, en cuyo caso utilizan, y con el resto se realiza el siguiente ataque. Por una parte se dispone de un diccionario en claro con gran cantidad de palabras y combinaciones muy comunes y válidas como claves. Posteriormente se realiza su cifrado con la utilidad del sistema a atacar o una copia de la misma, se comprueba el resultado del cifrado con el contenido del fichero de claves y en el caso de que se produzca una coincidencia inferimos cuál es la palabra en claro.

6.2.8 ATAQUE DE BÚSQUEDA EXHAUSTIVA

El ataque de búsqueda exhaustiva o fuerza bruta para encontrar la clave de un cifrador, es útil cuando el tamaño de la clave a atacar es reducido. Se realiza generando aleatoriamente todos los valores posibles de las claves de acceso y transformándolas. De esta forma se puede elegir la clave de acceso cuya transformada coincida con la interceptada. Para claves de gran longitud este ataque se agiliza con un hardware ASIC de ruptura de elevadísimo costo. (Areito, 1998)

6.3 ATAQUE SOBRE CIFRADOS DE SUSTITUCIÓN MONOALFABÉTICA

Desde los inicios de la escritura se ha visto la necesidad de transmitir mensajes de manera que el significado permanezca oculto para aquellos que no sean el destinatario, para tal fin han sido desarrollados diversos métodos de cifrar mensajes, entre ellos el cifrado por sustitución monoalfabética,

uno de los métodos clásicos de cifrado. Se establece un algoritmo, basado en herramientas computacionales y algebra lineal numérica, con el cual es posible atacar mensajes cifrados por sustitución monoalfabetica. (Triana Laverde)

6.3.1 EL CIFRADO DE CESAR Y SU CRIPTOANÁLISIS

Por supuesto, no podíamos olvidar el primer de los métodos de sustitución monoalfabética, el conocido método de Cesar. Este método fue el empleado por Julio Cesar en sus campañas durante el siglo 1 AC, para transmitir información en secreto. Era, como decimos, una sustitución que consistía en cifrar un mensaje empleando un alfabeto equivalente al original, pero desplazado en 3 letras. Es decir, al cifrar por ejemplo las palabras GALOS IRREDUCTIBLES con este algoritmo, Cesar obtendría:
JDÑRVLUUHGXFWLEÑHV
El primer criptoanálisis aplicable sería el método de complementación al componente original, es decir, probar las 27 permutaciones posibles obtenidas como resultado de desplazar una posición cada vez las letras de una palabra del criptograma. Por ejemplo, para JDÑRV, analizando los 27 resultados es más que probable que encontremos una única palabra que tenga significado en español, con lo cual solo tendríamos que utilizar el mismo desplazamiento obtenido para descifrar el resto del texto.
El segundo criptoanálisis aplicable a este tipo de cifrado es el análisis de frecuencias. De una forma un tanto más elegante que el anterior, mediante este método, estudiamos la frecuencia relativa de aparición de las diferentes letras del texto cifrado, para compararlas con una descripción estadística que hayamos obtenido del lenguaje, en el que sospechemos o sepamos que se encuentra el mensaje original También se puede realizar un estudio sobre las palabras más usadas, o sobre los diagramas y trigramas que constituyen el inicio y terminación más frecuente de las palabras de un lenguaje. Por ejemplo, para un texto en español lo suficientemente representativo, se ha obtenido la siguiente distribución de frecuencias:

Tabla 2 . Tabla de frecuencias del idioma español

A	11,96%	B	0,92%	C	2,92%
D	6,87%	E	16,78%	F	0,52%
G	0,73%	H	0,89%	I	4,15%
J	0,30%	K	0,01%	L	8,37%
M	2,12%	N	7,01%	Ñ	0.29%
O	8,69%	P	2,77%	Q	1,53%
R	4,94%	S	7,88%	T	3.31%
U	4.80%	V	0.39%	W	0.01%
X	0.06%	Y	1.54%	Z	0.15%

Este tipo de criptoanálisis es terriblemente efectivo para cifrado monoalfabético. Al emplearlo, se pone en evidencia la principal vulnerabilidad de este método pues, el hecho de sustituir unas letras por otras siguiendo siempre la misma congruencia lineal, hace que las propiedades estadísticas del criptograma y del texto en claro sean exactamente las mismas. Simplemente hay que llevar a cabo un análisis estadístico de los símbolos del criptograma e intentar solaparlo o encajarlo con la distribución de los símbolos de nuestro idioma De esta forma, hallaremos el desplazamiento que fue aplicado al cifrar el texto original y podremos descifrar, inmediatamente, el resto del mensaje. Por ejemplo, si interceptamos el si guiente mensaje cifrado:
DQR FLPFXHPWD DPWHV GH FULVWR WRGD ND JDÑLD HVWD RFXSDGA OHPRV XPD SHTXHQD DÑGHD GH LUUHGXFWLEÑHV JDÑRV TXH UHVLVWH DKRUD B VLHOSUH DÑ LPYDVRU
El análisis de frecuencias de los símbolos de este criptograma es:

Tabla 3 Frecuencia de aparición de cada símbolo en el criptograma

1 1 _ 18 1 5 6 16 _ 2 1 8 _ _ 6 2 6 2 8 3 2 7 10 7 6 1 _
A B C D E F G H I J K L M N Ñ O P Q R S T U V W X Y Z

Si se observa atentamente esta tabla, que refleja el número de apariciones de cada letra en el texto cifrado, y se compara con el gráfico de frecuencias, se puede hallar rápidamente un encaje con otro alfabeto equivalente desplazado. Únicamente hay que tratar de hacer concordar las letras más frecuentes en castellano con las más frecuentes en el criptograma. Así, sabiendo que una de las letras más utilizadas en español es la A, se le puede hacer corresponder la D, que es la más frecuente en nuestro texto cifrado. Cuatro posiciones a la derecha de la D, encontramos la H, también con un altísimo número de apariciones. Se puede entonces pensar que dicha H puede ser la sustituta de la letra e, también muy frecuente en nuestro idioma. En efecto, en el alfabeto español la A y la E distan exactamente cuatro lugares, con lo que se intuye que vamos por el buen camino. Se puede corroborar nuestra hipótesis procediendo de la misma manera con las letras menos frecuentes del lenguaje. Finalmente, concluimos que se trata de un cifrado monoalfabético con la siguiente correspondencia entre letras:

Tabla 4 Correspondencia entre las letras del criptograma y las del texto en claro

Criptograma	Frecuencia	Texto claro
A	1	X
B	1	Y
C	-	Z
D	18	A
E	1	B
F	5	C
G	6	D
H	16	E
I	-	F
J	2	G
K	1	H
L	8	I
M	-	J
N	-	K
Ñ	6	L
O	2	M
P	6	N
Q	2	Ñ
R	8	O
S	3	P
T	2	Q
U	7	R
V	10	S
W	7	T
X	6	U
Y	1	V
Z	-	W

Es decir, el desplazamiento de este cifrado monoalfabético es 3.
Para comprobarlo, aplicamos este cambio al mensaje cifrado y obtenemos: AÑO CINCUENTA ANTES DE CRISTO TODA LA GALIA ESTA OCUPADA MENOS UNA PEQUEÑA ALDEA DE IRREDUCTIBLES GALOS QUE RESISTE AHORA Y SIEMPRE AL INVASOR. (Morant)

6.4 Ataque sobre cifrados Polialfabéticos.

La sustitución polialfabética es una generalización de los sistemas de sustitución monoalfabeto. Este tipo de sustitución consiste en cifrar empleando una clave compuesta, es decir, de dos símbolos o más, que se usa cíclicamente.

Un buen ejemplo de cifrado polialfabético es el cifrado de Vigenére, que se sirve de una tabla para facilitar las operaciones de cifrado y descifrado. Es interesante resaltar el hecho de que cada una de la filas de esta tabla que un cifrado de Cearo La primera tiene un desplazamiento de O, la segunda de 1, y así sucesivamente

Tabla 5 Tabla 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

A diferencia de los cifrados monoalfabéticos, los polialfabéticos no conservan la misma distribución de frecuencias del texto original. Son más próximos a un cifrador ideal (como el de Vemam) ya que su distribución de símbolos se acerca más a una Uniforme. Sin embargo, pese a ser más segura que la monoalfabética, la sustitución polialfabética no es inmune al criptoanálisis. Efectivamente, un oficial prusiano ll amado Kasiski (1805 - 188 l), elaboró un método para hallar el número de alfabetos en una sustitución polialfabética. El método se basa en la idea de que en todos los idiomas aparecen grupos de caracteres con más frecuencia que otros. Es decir, por ejemplo en castellano, existen ciertos di gramas y trigramas que tienen mucha más probabilidad de darse que otros, en un texto, como son: es, de, os, en, la, con, etc . Si un texto se cifra con un número x de alfabetos de forma cíclica y, si un grupo de caracteres aparece un número y de veces en un texto, éste será cifrado aproximadamente y/n veces con el mismo alfabeto. En resumen, en un criptograma los eficientemente extensos encontraremos, ineludiblemente, repeticiones. Por ejemplo, supongamos que ciframos el siguiente alfabeto con la clave G/N (3 alfabetos):

D	E	S	C	O	N	F	I	A	N	Z	A		C	O	N		E	L	...
G	I	N	G	I	N	G	I	N	G	I	N		G	I	N		G	I	...
J	M	F	I	W	Z	L	P	N	S	H	N		I	W	Z		K	T	

Se puede ver cómo, separada por 9 espacios, aparece repetida la cadena IWZ. Así pues, se puede ya deducir que los posibles periodos de nuestra clave serán el, 3 ó 9. Hay que decidirse por una de las

alternativas, así que escogemos el 3 (la correcta, por otro lado) para avanzar con el método. A continuación, lo que se debe hacer e reordenar el texto cifrado en un número de columnas igual al periodo de clave opuesto correcto. Esta es la estrategia ganadora pues, tras hacer esto, el problema podrá ser tratado por columnas independientes. Cada u na de esa columnas será un simple cifrado monoalfabético, de criptoanálisis simple, como ya se ha visto en el apartado anterior. El problema de descifrar un criptograma que presentaba una distribución de frecuencias considerablemente uniforme ha quedado reducido a tres análisis frecuenciales independientes, uno para cada columna. Es interesante resalta r que, algunas veces, puede n a parecer repeticiones que son fruto de la casualidad. Pero juega a nuestro favor el hecho de que estas repeticiones casuales están distribuidas al azar. (Starbase, s.f.)

6.5 TEST DE KASISKI

MÉTODO KASISKI Ideado en 1863 para romper el sistema criptográfico Vigenere. Se basa en que ciertas secuencias de letras aparecen muchas veces en los textos originales, y consecuentemente los textos cifrados con Vigenere tienen las correspondientes secuencias de letras cifradas también muy repetidas. Por tanto, la distancia entre las cadenas más repetidas en el texto cifrado debe ser un múltiplo de la longitud de la clave K, y un buen candidato a longitud será el producto de los factores más repetidos en las distancias encontradas.
Por ejemplo: Texto original: TOBEORNOTTOBE Clave: NOWNOWNOWNOWN
Texto cifrado: GCXRCNACPGCXR
Los pasos de este tipo de criptoanálisis son:

- Se buscan los poligramas más repetidos y las distancias desde sus puntos de inicio para cada uno.
- Se sabe que la longitud de la clave divide a todas esas distancias, por lo que también dividirá a su mcd.
- Se calculan los factores de cada distancia y se descubren los factores más repetidos (candidatos a longitud de clave).
- Se divide el criptograma en filas según cada longitud candidata (empezando por la mayor) y se compara la distribución de frecuencias de cada columna con la del idioma usado

En otras palabras el test de KASISKI Sirve para estimar el largo de la clave, se basa en observar que dos segmentos idénticos del texto claro, serán transformados al mismo texto cifrado siempre que la distancia entre segmentos sea una d, tal que d = 0 mod m. Siendo m la long. de la clave.
El test busca dentro del texto cifrado pares de segmentos idénticos de un largo de por lo menos tres y guarda la distancia entre posiciones de los dos segmentos. El largo de la clave podría ser el m.c.d. de los di (es una suposición)(DSIC-UPV, 2012)

6.6 ÍNDICE DE COINCIDENCIA

Dado un texto cifrado es posible establecer una medida de dispersión de las frecuencias de los símbolos del mensaje respecto a una distribución uniforme:

$$MD = \sum_{i=0}^{26} \left(p_i - \frac{1}{n} \right)^2 = \sum_{i=0}^{26} \left(p_i^2 - \frac{2p_i}{n} + \frac{1}{n^2} \right)^2 = \sum_{i=0}^{26} \left(p_i^2 \right) - 0{,}037$$

En un texto donde los símbolos presentan una distribución propia del castellano:

$$IC = \sum_{i=0}^{26} (p_i^2) = 0{,}072 \quad \Rightarrow \quad 0 \leq MD \leq 0{,}035$$

Intuitivamente, el IC mide la probabilidad de que dos símbolos tomados al azar de un texto cifrado sean iguales.

El IC puede estimarse utilizando la frecuencia de los símbolos en el criptograma:

$$IC \simeq \frac{\sum_{i=0}^{26} f_i(f_i - 1)}{N(N-1)}$$

Donde f_i denota el número de ocurrencias del carácter *i*-esimo en un criptograma de N símbolos

De este modo:

$$IC = MD + 0{,}037 \quad \Rightarrow \quad 0{,}037 \leq IC \leq 0{,}072$$

$p = 1$	$IC = 0{,}072$	$p = 4$	$IC = 0{,}046$
$p = 2$	$IC = 0{,}054$	$p = 10$	$IC = 0{,}040$
$p = 3$	$IC = 0{,}049$	$p >>$	$IC = 0{,}037$

Criptograma (más extenso):

```
FSGWHAYPVJFRNIIYVRHLRMVRGNPBSGWDÑNWGBAGÑHFCUÑN
FCBSMHAAANAUEVNHDCFVJFRNICFAHCUÑNWGÑSOUFMUMCGS
AXHSJKZUEIWZCUFHYLQYJIYHMYKÑBSOFSFXWYCFTOAGÑAP
UQYUJIYWGÑNQCWRHSBJLUDJLHYKVJKRNWAFSIHAJYKGCVÑ
XWFUNAUWGKWPCWQYMRWFCFHTCSWQYLPMADÑAJUUCHWAGAC
KAACHAKHPULVGIYCUÑWACHWGGSGUEDFAÑNWAFFHGXAJYKG
JLZJÑVGARHMCNWGÑKIWMIMSYCLHULSOWFJFSMWPOWAÑWGF
HGYCFHYFHJLQYWANSAWZÑMWGULVXWÑNIRMHRFKRNNYÑSQJ
VRÑHQJWGJWGFWKRJEISVRVAYSICWHPJFJCFPYFHYI ...
```

p = 1 ⇒IC = 0,0471363

p = 2 ⇒IC = {0,0479231, 0,0463764}

p = 3 ⇒IC = {0,0739754, 0,0753505, 0,072086}

p = 4 ⇒IC = {0,0470939, 0,046151, 0,0487458, 0,0464554}

Una vez detectado el número de alfabetos, puede aplicarse un análisis de frecuencias como el ya visto Efectivamente, el número de alfabetos es 3

Mensaje:

LASCONEXIONESPUEDENSERDEMUCHASCLASESHISTORicas
NOHAYNINGUNAMISOPINIONESPOLITicasESTABANYATOMA
NDOFORMAMUCHOANTESDEQUEOYERAHABLARDELINGUISTIC
AYLAQUEESTUDIEENAÑOSPOSTERIORESENLAUNIVERSIDAD
ERAUNAESPECIEDETECNOLOGIADESCRIPTIVACONENMIOPI
NIONPOCASIMPLICACIONESMASAMPLIASENLOSDIVERSOSM
OVIMIENTOSESTRUCTURALISTASFUERONFRECUENTESLOSI
NTENTOSDEENSANCHARESASIDEASPEROELRESULTADODETO
DOESOESCREOMUYDEBILYPOCOCONVINCENTE ...

6.7 ATAQUE DE TEXTO CONOCIDO SOBRE EL MÉTODO DE HILL

Vamos a analizar dos tipos de ataques al cifrado de Hill como son el ataque por fuerza bruta y el ataque con texto en claro y texto cifrado conocido. Con estos dos tipos de ataques lo que se pretende es obtener la clave pero no todos los tipos de ataques están encaminados a obtener la clave. En otros tipos de ataques el criptoanalista lo que pretende es obtener cierta información del texto cifrado, obtener una parte del texto original, etc. Un primer ataque y el más básico es el ataque por fuerza bruta que consiste en lo siguiente. Conociendo solo el texto cifrado probar todas las posibles claves para descifrar el texto obteniendo así l a clave utilizada en el cifrado. Vamos a ver unos ejemplos de este tipo de ataque utilizando varios tamaños de clave y los dos alfabetos que hemos implementado. Sabiendo que la clave es de tamaño 3, es decir, una matriz cuadrada de orden 3 y utilizando un alfabeto de 27 símbolos, tendríamos que probar en el peor de los casos 279 = 7625597484987 posibles claves distintas. Sea n el número de símbolos del alfabeto que estamos utilizando, para un tamaño de clave d hay n d 2 matrices distintas entre las que podemos escoger una clave, aunque hay que descartar aquellas que no son invertibles.

Sean el número de símbolos del alfabeto que estamos utilizando, para un tamaño de clave d hay n d 2 matrices distintas entre las que podemos escoger una clave, aunque hay que descartar aquellas que no son invertibles. Podríamos calcular el coste en tiempo que tardaría una maquina en probar todas las claves posibles, obteniendo de esta manera la clave utilizado en el cifrado de un mensaje. Asumiendo que probar cada clave tiene un coste de una operación básica por segundo y utilizando una capacidad computacional de 20000 MIPS [15] (Millions of Instructions Per Second o millones de instrucciones por segundo).

20000000000 operaciones por segundo

En la siguiente tabla podemos ver el tiempo utilizado en probar todas las claves posibles (todas las matrices posibles) y para los dos alfabetos que hemos utilizado.

Tabla 6 Costes para distintos tamaños de claves en el ataque por fuerza bruta

Tamaño de clave	Tiempo para $n = 27$	Tiempo para $n = 256$
3	6 min 21 seg	7487 años
4	126468 años	$5,39 \cdot 10^{20}$ años
5	$9,64 \cdot 10^{17}$ años	$2,54 \cdot 10^{42}$ años
6	$5,36 \cdot 10^{33}$ años	$7,88 \cdot 10^{68}$ años
7	$2,17 \cdot 10^{52}$ años	$1,59 \cdot 10^{100}$ años

Como resultado de este análisis podemos observar que el cifrado de Hill tiene una complejidad de orden exponencial. El coste es de O(n d 2), donde n está determinado, solo puede ser 27 ´o 256, y d es el parámetro variable, por lo tanto tiene un coste exponencial. Ahora bien, existe otro tipo de ataque que es el ataque con el texto cifrado y el texto original conocido. Para este tipo de ataque además de conocer el texto cifrado el criptoanalista conoce también el texto original siendo mucho más fácil y rápido obtener la clave.

Cuando ciframos realizamos la siguiente operación de matrices:

$$K \cdot M_1 = C_1 \Longrightarrow K \cdot \begin{pmatrix} m_1 \\ m_2 \\ m_3 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}$$

El atacante que conoce el mensaje original y el mensaje cifrado y sabiendo que la clave es de orden d puede generar la matriz P, que estará formada por d bloques de tamaño d con los elementos del mensaje original. Es decir, P será una matriz cuadrada de orden d formada por los elementos del mensaje original.

$$P = \begin{pmatrix} m_1 & m_4 & m_7 \\ m_2 & m_5 & m_8 \\ m_3 & m_6 & m_9 \end{pmatrix}$$

También puede generar la matriz C que estará formada igual que la matriz P pero con los correspondientes bloques cifrados.

$$C = \begin{pmatrix} c_1 & c_4 & c_7 \\ c_2 & c_5 & c_8 \\ c_3 & c_6 & c_9 \end{pmatrix}$$

Por lo tanto para obtener la clave a partir del texto cifrado y del texto original solo hay que realizar la siguiente operación de matrices: K = C · P −1 Para poder llevar a cabo esta operación es necesario que la matriz del texto original P tenga inversa P −1. La matriz P tendrá inversa si el m.c.d entre el determinante de P y el modulo del alfabeto que estamos utilizando (por ejemplo el alfabeto módulo 27) es igual a 1. En el caso de que la matriz P no tenga inversa no se podrá obtener la clave con esa matriz P del texto original. Pero podemos generar otra matriz con el siguiente bloque del mensaje original que sí tenga inversa, siempre que el mensaje original sea lo suficiente extenso. Vamos a ver un ejemplo de este tipo de ataque. Texto original =”el cifrado de Hill” Texto cifrado =”kknwmobwnvejbplzilqj” Sabemos que la clave es de tamaño 4 Texto que vamos a utilizar para obtener la clave:

Texto original

[e , l , t , c] [i , f , r , a] [d , o , t , d] [e , t , h , i]

Números correspondientes a los caracteres del texto original

[4 , 11 , 26 , 2] [8 , 5 , 17 , 0] [3 , 14 , 26 , 3] [4 , 26 , 7 , 8]

Texto cifrado

[k , k , n , w] [m , o , b , w] [n , v , e , j] [b , p , l , z]

Números correspondientes a los caracteres del texto original

[10 , 10 , 13 , 22] [12 , 14 , 1 , 22] [13 , 21 , 4 , 9] [1 , 15 , 11 , 25]

Construimos la matriz del texto original (P) con los números de sus caracteres:

$$P = \begin{pmatrix} 4 & 8 & 3 & 4 \\ 11 & 5 & 14 & 26 \\ 26 & 17 & 26 & 7 \\ 2 & 0 & 3 & 8 \end{pmatrix}$$

Construimos la matriz del texto cifrado (C) con los números de sus caracteres:

$$C = \begin{pmatrix} 10 & 12 & 13 & 1 \\ 10 & 14 & 21 & 15 \\ 13 & 1 & 4 & 11 \\ 22 & 22 & 9 & 25 \end{pmatrix}$$

Determinante de P = 398

m.c.d (398, 27) = 1

Matriz inversa de P:

$$P^{-1} = \begin{pmatrix} 3 & 11 & 16 & 23 \\ 14 & 0 & 3 & 14 \\ 1 & 14 & 24 & 14 \\ 9 & 19 & 14 & 6 \end{pmatrix}$$

Realizamos la operación de matrices K = C · P −1 y obtenemos la clave K:

$$K = C \cdot P^{-1} = \begin{pmatrix} 4 & 14 & 9 & 19 \\ 4 & 14 & 25 & 0 \\ 21 & 3 & 2 & 3 \\ 14 & 6 & 12 & 10 \end{pmatrix}$$

Una vez hemos obtenido la clave realizamos una comprobación. Vamos a cifrar el primer bloque del mensaje original con la clave que hemos obtenido.

* e , l , t , c + −→ * 4 , 11 , 26 , 2 +

$$K \cdot M_1 = \begin{pmatrix} 4 & 14 & 9 & 19 \\ 4 & 14 & 25 & 0 \\ 21 & 3 & 2 & 3 \\ 14 & 6 & 12 & 10 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 11 \\ 26 \\ 2 \end{pmatrix} = \begin{pmatrix} 10 \\ 10 \\ 13 \\ 22 \end{pmatrix}$$

* 10 , 10 , 13 , 22 + −→ * k , k , n , w +

Como podemos observar el resultado de cifrar el primer bloque con la clave obtenida es igual a los primeros cuatro primeros caracteres del texto cifrado. (Castejon)

7 SEGURIDAD INFORMATICA DE PROTOCOLOS: MAC Y HASH.

7.1 SERVICIOS SUMINISTRADOS PARA SEGURIDAD INFORMÁTICA A TRAVÉS DE LAS PRIMITIVAS CRIPTOGRÁFICAS

En el capítulo de cifrado Asimétrico vimos que permite autentificar información, es decir, poder asegurar que un mensaje m proviene de un emisor A y no de cualquier otro. Asimismo vimos que la autentificación debía hacerse empleando una función resumen y no codificando el mensaje completo. En esencia, una función resumen (hash, en inglés), proporciona una secuencia de bits de pequeña longitud, que va asociada al mensaje aunque contiene menos información que éste, y que debe resultar muy difícil de falsificar. Existen funciones resumen que emplean en sus cálculos una clave adicional los denominados MAC[9]y otras que no la usan, denominadas genéricamente MDC[10] (modification detection codes).

(Rey, 2015)

La forma de conseguir alcanzar los tres criterios de seguridad usando un sistema asimétrico sería:

7.1.1 CONFIDENCIALIDAD

Es suficiente con que el emisor cifre los datos con la clave pública del receptor. Como sólo se pueden descifrar mediante la clave privada del receptor, éste será el único que pueda verlos.

[9] MAC (message authentication code) el resultado de cifrar el último bloque del mensaje

[10] MDC (modification detection codes) funciones de compresión, que dan como resultado bloques de longitud fija

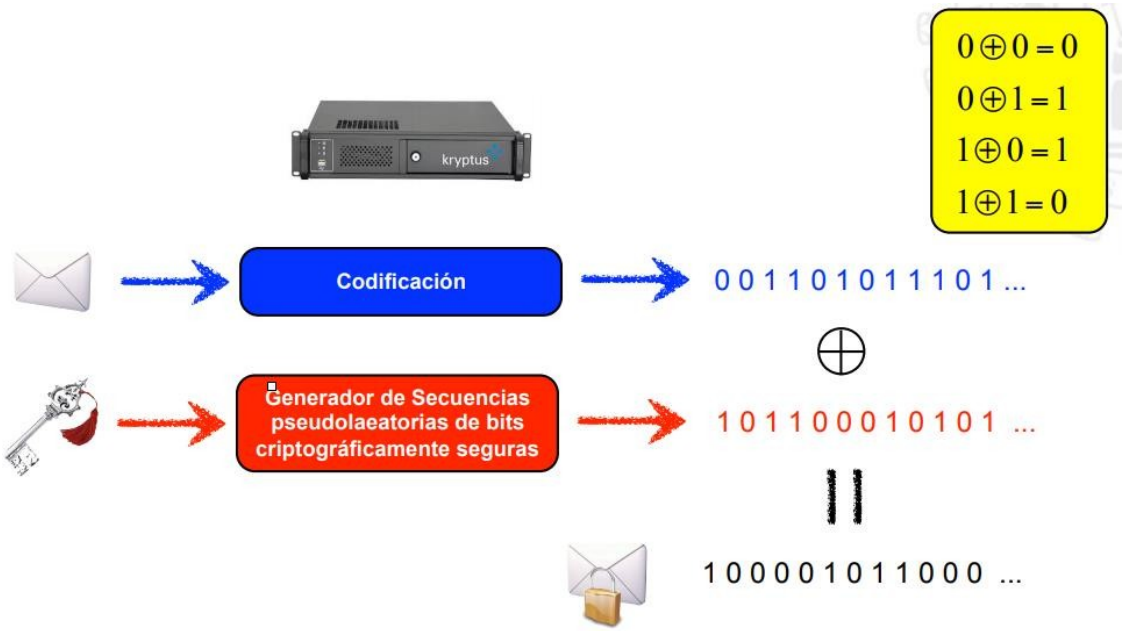


Fig. 32 confidencialidad de claves

7.1.2 VENTAJAS DE LOS CRIPTOSISTEMAS DE CLAVE SECRETA:

- Gran rapidez en el cifrado.
- Cifrado de gran cantidad de datos.
- Tamaño de las claves pequeño (128-256 bits).
- Los algoritmos se basan en sencillas operaciones matemáticas.

7.1.3 DESVENTAJAS DE LOS CRIPTOSISTEMAS DE CLAVE SECRETA:

- Distribución y gestión en red de la clave secreta.
- La clave secreta se debe cambiar con cierta frecuencia.

7.1.4 INTEGRIDAD

Al cifrar un mensaje con la clave privada del emisor, el único que puede alterar el contenido es el propio emisor. Se puede combinar con el método anterior, cifrando primero con la clave privada propia para garantizar la integridad, y después con la clave pública del receptor para la confidencialidad.

La integridad se garantiza mediante el uso de funciones resumen (hash functions) o MAC.

- función MD5 [11]
- funciones SHA[12] (SHA-1, SHA-2, SHA-3)
- función RIPEMD-160
- función Tiger

Las funciones resumen deben verificar las siguientes condiciones:

Si se cambia un único bit del mensaje m, el resumen h=f(m) debería cambiar (por término medio) en la mitad de los bits.

Resistencia a la preimagen: dado un resumen, debe ser computacionalmente muy dificil obtener el mensaje.

Resistencia a la segunda preimagen: dado un mensaje, debe ser computacionalmente difícil encontrar otro mensaje diferente de manera que sus resúmenes coincidan.

Resistencia a colisiones: debe ser computacionalmente difícil encontrar dos mensajes disti ntos con el mismo resumen. En el caso de que no se cumplieran estas condiciones, la función resumen sería vulnerable y su seguridad se vería comprometida por el ataque de la paradoja del cumpleaños.

Paradoja del cumpleaños: ¿cuántas personas debe haber en una sala para que la probabilidad de que dos de ellas celebren su cumpleaños el mismo día sea superior al 50%?

“Confidencialidad Es suficiente con que el emisor cifre los datos con la clave pública del receptor mientras que Integridad Al cifrar un mensaje con la clave privada del emisor, el único que puede alterar el contenido es el propio emisor “

[11] MD5 de uno de los más populares algoritmos de generación de signaturas, debido en gran parte a su inclusión en las primeras versiones de PGP

[12] SHA fue desarrollado por la NSA, para ser incluido en el estándar DSS (Digital Signature Standard)

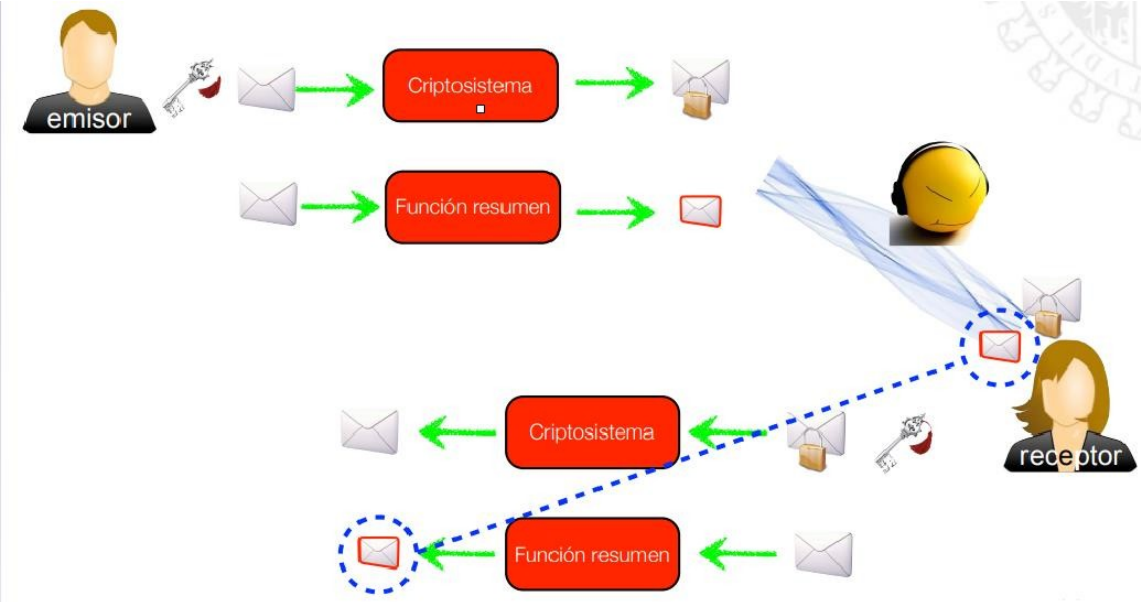


Fig. 33 integridad con funciones resumen

Los Message Authentication Codes (MAC) son algoritmos parecidos a las funciones resumen pero con la diferencia de que hacen uso de una clave para calcular el resumen.

7.1.5 AUTENTICIDAD

Al cifrar con la clave privada del emisor, se garantiza que nadie más puede haber enviado el mensaje.

- El mensaje es auténtico y no se ha suplantado la personalidad del emisor.
- La autenticidad se garantiza mediante el uso de los esquemas de firma digital.
- Los esquemas de firma digital se basan en el uso de los criptosistemas de clave pública.

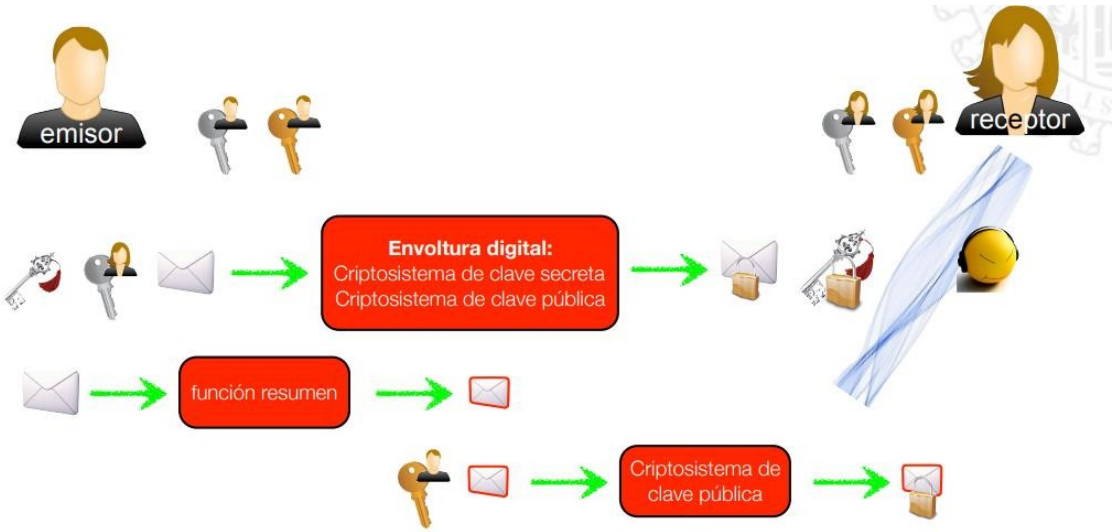


Fig. 34 Esquema de firma digital

En la actualidad para poseer una autenticidad se a empleado diferentes tipos de firmas digitales tales como:

- Firma digital DSA (Digital Signature Algorithm).
- Firma digital RSA. ‣ Firma digital con curvas elípticas.
- Firma digital con curvas hiperelípticas.

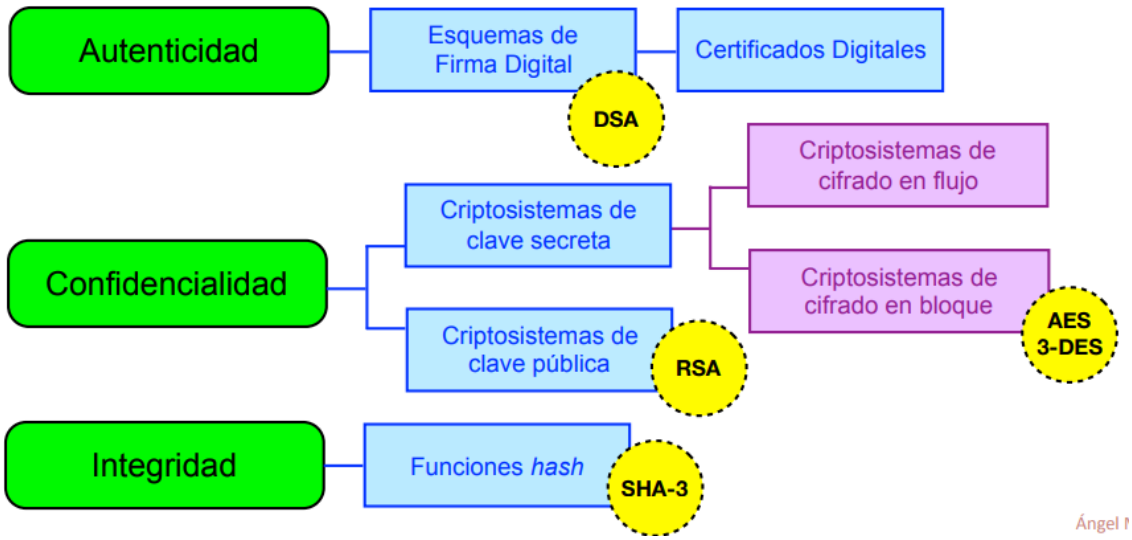


Fig. 35 uso de algoritmos matemáticos

7.1.6 FIRMA DIGITAL

La firma electrónica es una aplicación de la criptografía asimétrica para garantizar la autenticidad e integridad de los datos. Como el cifrado de todo el mensaje puede ser computacionalmente costoso y poco eficiente, lo que se hace en esta técnica es, en primer lugar, generar un hash representativo de los datos que lo componen. Una vez generado el hash del mensaje, se cifra con la clave privada del emisor, constituyendo la firma digital que se adjuntará con el mensaje. El receptor, a su vez, generará el hash del mensaje y descifrará con la clave pública del emisor la firma digital, comprobando que ambos hashes (el recibido y el calculado) son los mismos. (Heineken Team, 2001)

El uso de firmas digitales proporciona los siguientes beneficios:

- Integridad de datos
- Autenticidad del emisor
- Timestamping: Si se incluye la fecha en el hash, es posible saber el momento de la firma. Para que esta información sea fiable, hay que asegurar que la fecha introducida en el momento de firmar es

correcta, por lo que se recurre a autoridades de timestamping que se encargan de dar fechas firmadas.

- Confirmación de recepción: La firma por parte del receptor, especialmente si incluye timestamping, es una forma de certificar la recepción de un documento.

La firma electrónica es una aplicación de la criptografía asimétrica para garantizar la autenticidad e integridad de los datos ya que puede ser computacionalmente costoso y poco eficiente, lo que se hace en esta técnica es, en primer lugar, generar un hash representativo de los datos que lo componen.

7.2 VULNERABILIDADES DE LOS SERVICIOS SUMINISTRADOS

Realmente la seguridad es la facultad de estar a cubierto de algún riesgo o amenaza. Desde este punto de vista la seguridad total es muy difícil de logra, puesto que implicaría describir todos los riesgos y amenazas a que puede verse sometido el sistema. Lo que se manifiesta en los sistemas no es la seguridad, sino más bien la inseguridad o vulnerabilidad. No se puede hablar de un sistema informático totalmente seguro, sino más bien de uno en el que no se conocen tipos de ataques que puedan vulnerarlo, debido a que se han establecido medidas contra ellos. Algunos tipos de vulnerabilidad de un sistema son los siguientes:

Vulnerabilidad física. Se encuentra en el nivel del edificio o entorno físico del sistema. Se relaciona con la posibilidad de entrar o acceder físicamente al sistema para robar, modificar o destruir el mismo.

Vulnerabilidad natural. Se refiere al grado en que el sistema puede verse afectado por desastres naturales o ambientales que pueden dañar el sistema, tales como el fuego, inundaciones, rayos, terremotos, o quizás más comúnmente, fallos eléctricos o picos de potencia. También el polvo, la humedad o la temperatura excesiva son aspectos a tener en cuenta.

Vulnerabilidad del hardware y del software. Desde el punto de vista del hardware, ciertos tipos de dispositivos pueden ser más vulnerables que otros. Así, ciertos sistemas requieren la posesión de algún tipo de herramienta o tarjeta para poder acceder a los mismos. Ciertos fallos o debilidades del software del sistema hacen más fácil acceder al mismo y lo hacen menos fiable. En este apartado se incluyen todos los bugs en los sistemas operativos, u otros tipos de aplicaciones que permiten atacarlos. (Lujan, 2016)

Vulnerabilidad de los medios o dispositivos. Se refiere a la posibilidad de robar o dañar los discos, cintas, listados de impresora, etc.

Vulnerabilidad por emanación. Todos los dispositivos eléctricos y electrónicos emiten radiaciones electromagnéticas. Existen dispositivos y medios de interceptar estas emanaciones y descifrar o

reconstruir la información almacenada o transmitida. Vulnerabilidad de las comunicaciones. La conexión de los ordenadores a redes supone sin duda un enorme incremento de la vulnerabilidad del sistema. Aumenta enormemente la escala del riesgo a que está sometido, al aumentar la cantidad de gente que puede tener acceso al mismo o intentar tenerlo. También se añade el riesgo de intercepción de las comunicaciones:

- Se puede penetrar al sistema a través de la red.
- Interceptar información que es transmitida desde o hacia el sistema.

Vulnerabilidad humana. La gente que administra y utiliza el sistema representa la mayor vulnerabilidad del sistema. Toda la seguridad del sistema descansa sobre el administrador del mismo que tiene acceso al máximo nivel y sin restricciones al mismo. Los usuarios del sistema también suponen un gran riesgo al mismo. Ellos son los que pueden acceder al mismo, tanto físicamente como mediante conexión. Existen estudios que demuestran que más del 50% de los problemas de seguridad detectados son debidos a los usuarios de los mismos. Por todo ello hay una clara diferenciación en los niveles de los distintos tipos de vulnerabilidad y en las medidas a adoptar para protegerse de ellos

7.2.1 VALORACIÓN DE AMENAZAS Y DETERMINACIÓN DEL IMPACTO

El paso siguiente para la medición del nivel de riesgo es la determinación del impacto adverso como resultado de la ejecución de una amenaza. Este impacto se puede describir en términos de pérdida o degradación de alguna de las tres características básicas: confidencialidad, integridad y disponibilidad. Para cada activo y amenaza debe estimarse la degradación, es decir el porcentaje en que la amenaza daña al activo en estudio estableciendo un valor entre 0 % (no lo daña) y 100 % (lo daña absolutamente) para cada una de las características de confidencialidad, integri dad y disponibilidad. (Lujan, 2016)

- Pérdida de Integridad: Se refiere al requerimiento de que el activo o la información sea protegido contra la modificación no autorizada. Se pierde integridad si se realizan cambios no autorizados en los sistemas o se pierde parte de los datos almacenados sea por un evento accidental o intencionado.
- Pérdida de Disponibilidad: El hecho de que la información o un sistema no esté disponible para sus usuarios, ya sea por la pérdida de datos o la destrucción de elementos necesarios, puede afectar a la efectividad operacional y consecuentemente al cumplimiento de la misión de una organización.
- Pérdida de Confidencialidad: La confidencialidad hace referencia a la protección de la información contra la divulgación no autorizada. El impacto producido por un evento de estas características, sea en forma no autorizada, intencional o inadvertida, puede variar entre la pérdida de confianza en la institución hasta la posibilidad de acciones legales contra la misma.

7.3 JUSTIFICACION DE LAS FUNCIONES FCS: MAC Y HASH[13]

El control de acceso define a que objetos puede acceder cada sujeto. Por objeto entendemos cualquier entidad que contiene información, y puede ser físico o abstracto. Los sujetos acceden a los objetos, y pueden ser usuarios, procesos, programas u otras entidades.

La Matriz de Control de Acceso o Matriz de acceso es un modelo abstracto formal de seguridad que caracteriza los permisos de cada sujeto con respecto a todos los objetos en el sistema. Las filas de la matriz de acceso representan dominios y las columnas objetos. La entrada acceso (i,j) define el conjunto de operaciones que un sujeto en el dominio D_i puede invocar con el objeto O_J . Hay que tener en cuenta que la Matriz de Control de Acceso es solo un modelo teórico de permisos. Una implementación literal de este array bidimensional tendría excesivos requerimientos de memoria. (Lopez, 2007)

objeto / dominio	F_1	F_2	F_3
D_1	ejecutar	-	escribir
D_2	ejecutar	leer	ejecutar
D_3	ejecutar	-	-

Fig. 36 Matriz de acceso

Control de Acceso Discrecional (Discretionary Access Control DAC): es una política determinada por el propietario de un objeto. El propietario es el que decide quién puede acceder al objeto y que privilegios tiene.

Control de Acceso Obligatorio (Mandatory Access Control MAC): ´esta es una política de acceso determinada por el sistema, no por el propietario.

Control de Acceso Basado en Roles (Role-Based Access Control RBAC): esta política permite que los privilegios sean asignados a roles arbitrarios. Estos roles se pueden asignar después a usuarios reales.

“El control de acceso nos define que los objetos pueden acceder cada sujeto. Por objeto entendemos cualquier entidad que contiene información, y puede ser físico o abstracto. Los sujetos acceden a los objetos, y pueden ser usuarios, procesos, programas u otras entidades. “

[13] *HASH: Está diseñado para ser una función unidireccional(una función que es imposible de invertir)*

7.3.1 CONTROL DE ACCESO DISCRECIONAL (DAC)

Es un tipo de control de acceso definido por el TCSEC (Trusted Computer System Evaluation Criteria, estandar del DoD) que consiste en que el creador de los objetos es el que determina el acceso a dichos objetos. (Heineken Team, 2001)

Se basa en dos conceptos fundamentales:

- Propietario: todo objeto en el sistema tiene un propietario. En la mayoría de los sistemas DAC, el propietario inicial es el que ha creado o causado el objeto.
- Derechos de acceso y permisos: un propietario puede asignar a otros sujetos recursos.

Se implementa mediante:

- Bits de permisos: Se especifica a un usuario como el propietario de un archivo y cada archivo o directorio se afilia a un grupo. Se otorga permiso de solo lectura, escritura o ejecución a un grupo. En un momento dado un usuario pertenece a un grupo y adquiere los derechos de ese grupo.
- Sistema de Contraseñas: el propietario asigna a cada archivo una contraseña.
- Lista de Capacidades: cada objeto tiene un solo propietario, el cual otorga o cancela privilegios a los demás sujetos sobre dicho objeto. Cada usuario tiene una lista de capacidades que contiene los nombres de los objetos a los que tiene acceso y sus permisos correspondientes. Dicha lista es mantenida por el sistema operativo, y los usuarios no pueden acceder a ella directamente.
- Lista de Control de Acceso (ACL): los archivos y directorios tienen conjuntos de permisos configurados para el propietario del archivo, el grupo asociado con el archivo y todos los otros usuarios del sistema. Sin embargo, estos permisos tienen sus limitaciones. Por ejemplo, no se pueden configurar diferentes permisos para usuarios diferentes. Una lista de control de acceso es un conjunto de entradas de usuario, grupo y modo asociado a un archivo que especifica los permisos de acceso para todas combinaciones posibles de identificación de usuario o identificación de grupo.

7.3.2 CONTROL DE ACCESO OBLIGATORIO (MAC)

MAC es un tipo de control de acceso definido por el TCSEC que consiste en restringir el acceso a los objetos en función de la" sensibilidad" (representada por una etiqueta) de la información contenida en dichos objetos. Su característica más importante es que es la política de seguridad del sistema la única que determina el acceso a los objetos, retirándole este privilegio al creador de los objetos. MAC es usado en sistemas multinivel que procesan datos altamente confidenciales, como información clasificada gubernamental o militar. Un sistema multinivel es aquel que maneja múltiples niveles de clasificación entre sujetos y objetos. Conceptos básicos:

- Etiquetas: en un sistema basado en MAC, todos los sujetos y objetos deben tener una etiqueta asignada. Para acceder a un objeto, el sujeto debe tener un valor de etiqueta igual o mayor que dicho objeto.
- Importación y exportación de datos: controlar dicho flujo de información de y para otros sistemas es una función crıtica de los sistemas basados en MAC. Deben asegurar que las etiquetas son mantenidas de forma adecuada

7.3.3 CONTROL DE ACCESO BASADO EN ROLES (RBAC)

Los permisos para realizar ciertas operaciones son asignados a roles específicos. Los usuarios son asignados a roles particulares, a través de los cuales adquieren permisos para realizar funciones particulares.

RBAC se diferencia de las ACLs usadas por los sistemas discrecionales en que asigna permisos a operaciones específicas con significado en la organización, en lugar de a objetos de datos de bajo nivel. Por ejemplo, una lista de control de acceso puede ser usada para permitir o denegar el acceso a unos ficheros, pero no se le podría decir en que formas puede ser modificado dicho fichero. Además, el uso de roles como medio para asignar privilegios a usuarios simplifica en gran medida el manejo y creación de usuarios.

El uso de RBAC para manejar privilegios de usuario está muy extendido. Sistemas como Microsoft Active Directory, SELinux, FreeBSD, Solaris, Oracle DBMS, PostgreSQL 8.1, SAP R/3 implementan alguna forma de RBAC

7.4 FUNCIONES MAC

La criptografía es la técnica utilizada para cifrar mensajes y asegurar su confidencialidad en base a una serie de códigos y algoritmos que se encargan de cambiar la representación del lenguaje de dicho mensaje. El Message Authentication Code, también conocido como MAC una clave secreta que sólo conocen la persona que envía el mensaje y el destinatario para defenderse de los ataques que se puedan producir. El MAC utilizado se puede calcular en base a la función MAC = CK(M), en la que M es el mensaje de longitud arbitraria; si ese mismo código es encontrado en el sistema de la persona que recibe el mensaje entonces se comprueba su autenticidad e integridad.

7.4.1 FUNCIONAMIENTO

A pesar de que es uno de los métodos de seguridad más utilizados y conocidos lo cierto es que existen varias formas de romper la seguridad que genera una técnica criptográfica; de forma teórica a los mecanismos usados o de forma concreta a su implementación ya sea software o hardware. Este tipo de sistemas de seguridad funcionan por probabilidad, y aunque sea increíblemente pequeña, el atacante tiene una probabilidad mayor de 0 de conseguir su objetivo, como ocurre con todos los sistemas incondicionalmente seguros. También existe la posibilidad de explotar los canales ocultos, canales de comunicación no intencionados difícilmente detectables y que viola la política de seguridad con la que contamos por lo que se pueden usar para romper el sistema. Todos estos son las formas de engañar o destruir el sistema de seguridad de forma teórica pero además en la práctica puede ocurrir que el software o hardware estén diseñados de forma incorrecta y se creen más brechas.

Existe una variante de este sistema llamada HMAC que se basa en clave hash y calcula un código de encriptación MAC implicando una función hash criptográfica. Se puede utilizar tanto la función MD5 como la SHA-1 para el cálculo de HMAC, con un algoritmo resultante denominado HMAC-MD5 o HMAC-SHA1 que varía del MAC base. Al ser un algoritmo específico, se puede utilizar esta clave para aplicaciones que requieran claves MAC, y la fortaleza de su seguridad de pende de la longitud de la clave secreta utilizada. La diferencia fundaméntela es que con HMAC lo que se hace es coger un fragmento del texto del mensaje y cifrarlo en vez de cifrar el mensaje entero. Una desventaja de HMAC es su lentitud, pero otras muchas variantes de MAC como VMAC, UMAC y CMAC, que son más rápidas pero mucho menos utilizadas.

One-time Mac muy parecida a One-time encryption es la función utilizada cuando queremos definir un valor MAC para un solo mensaje y que ayuda a que el proceso sea mucho más rápido y ágil, además de que igualmente nos asegura que no se lee ni modifica el mensaje en ningún momento. Basado en esta idea de One-time Mac nos encontramos también con Carter-Wegman Mac, que se extiende en una función pseudoaleatoria que permite utilizar una sola clave para mensajes consecutivos. Basado también en la función MAC podemos encontrarnos con CBC MAC, capaz de encriptar mensajes de cualquier longitud, desde uno a varios bloques. Esto lo consigue ya que cada bloque depende de la encriptación del anterior creando una cadena de bloques. Es importante especificar la longitud del texto dentro del primer bloque para poder seguir con los demás ya que intentarlo sin saber esto puede causar problemas y dificultar la lectura. Para asegurarnos de su confidencialidad es aconsejable ir cambiando su clave secreta de vez en cuando usando este algoritmo.

Debemos elegir qué tipo de código de autenticación preferimos según el mensaje que queramos encriptar teniendo en cuenta su longitud, si queremos usar una clave cada vez o si por el contrario intercambiamos demasiados mensajes con la otras persona y queremos la misma siempre, la velocidad, si usamos hardware o preferimos software etc. teniendo en cuenta las ventajas y desventajas de cada método.

7.4.2 EJEMPLOS DE PROTOCOLOS MAC

Las cuentas de Google ya cuentan en sus sistemas con una medida de seguridad novedosa que deja atrás esos códigos SMS necesarios hasta ahora para verificar la cuenta; de esta forma se podía saber si el dueño del smartphone usado para intentar acceder a la cuenta es el mismo que creó la cuenta en su día. Ahora podemos aumentar la seguridad en nuestras redes sociales y acceder a nuestra cuenta con un sistema llamado TOTP un poco más complejo pero que nos aporta una mayor seguridad y de un solo uso facilitándonos una one time password. El servidor elegirá un número que utilizará como clave que transferirá a nuestro dispositivo y servirá como base para futuros códigos de autentificación. Cuando se nos pide finalmente el código de autenticación el sistema utiliza la clave que ya ha guardado para combinarla con la hora actual que detecta en el móvil para poder generarlo. Este sistema es algo más complejo de lo que puede parecer ya que la clave principal en la que se basa y se queda guardada en nuestro dispositivo se consigue cuando hacemos la configuración inicial de nuestra app de Gmail escaneando un código QR. Este sistema TOTP se basa en el hash anteriormente mencionado y lo utiliza como identificador único e inimitable. La clave generada y la hora se combinan utilizando el algoritmo SHA1 par a conseguir el código hash, pero como la hora está basada en

bloques de 30 segundos así se nos permite disponer del tiempo suficiente para conseguir el código final e introducirlo antes de que caduque.

“La criptografía es la técnica utilizada para cifrar mensajes y asegurar su confidencialidad en base a una serie de códigos y algoritmos que se encargan de cambiar la representación del lenguaje de dicho mensaje, tanto como su funcionamiento de sistemas de seguridad funcionan por probabilidad, y aunque sea increíblemente pequeña, el atacante tiene una probabilidad mayor de 0 de conseguir su objetivo, como ocurre con todos los sistemas incondicionalmente seguros“

7.5 FUNCIONES HASH

Las funciones criptográficas hash cifran una entrada y actúan de forma parecida a las funciones hash (Una función *hash* H es una función computable mediante un algoritmo tal que:

$$H: U \rightarrow M$$
$$x \rightarrow h(x)$$

Tiene como entrada un conjunto de elementos, que suelen ser cadenas, y los convierte en un rango de salida finito, normalmente cadenas de longitud fija. Es decir, la función actúa como una proyección del conjunto U sobre el conjunto M.), ya que comprimen la entrada a una salida de menor longitud y son fáciles de calcular.

Este tipo de funciones se caracterizan por cumplir propiedades que las hacen idóneas para su uso en sistemas que confían en la criptografía para dotarse de seguridad. Estas propiedades las hacen resistentes frente ataques maliciosos que intentan romper esa seguridad.

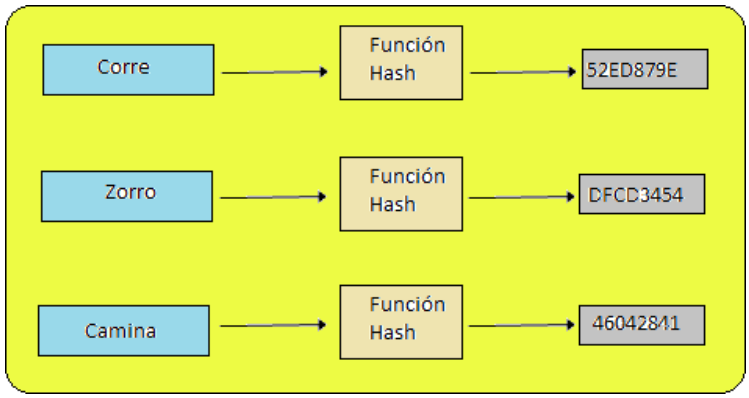


Fig. xx Ejemplo de criptografía mediante Hash

PROPIEDADES

La calidad de una función resumen viene definida con base en la satisfacción de ciertas propiedades deseables en el contexto en el que se va a usar.

7.5.1 BAJO COSTO

Calcular el valor *hash* necesita poco costo (computacional, de memoria, etc.).

7.5.2 COMPRESIÓN

Una función *hash* comprime datos si puede mapear un dominio con datos de longitud muy grande a datos con longitud más pequeña

7.5.3 UNIFORME

Se dice que una función resumen es uniforme cuando para una clave elegida aleatoriamente es igualmente probable tener un valor resumen determinado, independientemente de cualquier otro elemento.

Para una función *hash* H uniforme del tipo H:$\{0,1\}^m \rightarrow$,0,1-^{n}, es decir:

- Las cadenas están construidas sobre un alfabeto de 2 símbolos (Alfabeto binario)
- El dominio es el conjunto de las cadenas de longitud m
- El rango es el conjunto de las cadenas de longitud n

podemos decir que a cada resumen le corresponde 2^{m-n} mensajes y que la probabilidad de que dos mensajes den como resultado la misma salida es 2^{-n}

Para algoritmos de búsqueda, si todas las entradas son igualmente probables, se busca esta propiedad para minimizar el número de colisiones ya que cuantas más colisiones haya, será mayor el tiempo de ejecución de las búsquedas.

7.5.4 DE RANGO VARIABLE

En algunas funciones resumen el rango de valores resumen puede ser diferente a lo largo del tiempo. Ejemplo: funciones *hash* usadas para tablas resumen que necesitan expandirse. En estos casos a la función *hash* se le debe pasar un parámetro que le permita saber en qué rango se mueve la ejecución para hallar el valor resumen.

7.5.5 INYECTIVIDAD Y FUNCIÓN HASH PERFECTA

Se dice que la función resumen es inyectiva cuando cada dato de entrada se mapea a un valor resumen diferente. En este caso se dice que la función resumen es perfecta. Para que se dé, es necesario que la cardinalidad del conjunto dominio sea inferior o igual a la cardinalidad del conjunto imagen. Normalmente, sólo se dan funciones *hash* perfectas cuando las entradas están preestablecidas. Ejemplo: mapear los días del año en números del 1 al 366 según el orden de aparición.

7.5.6 DETERMINISTA

Una función *hash* se dice que es determinista cuando dada una cadena de entrada siempre devuelve el mismo valor *hash*. Es decir, el valor *hash* es el resultado de aplicar un algoritmo que opera solo sobre la cadena de entrada. Ejemplos de funciones *hash* no deterministas son aquellas funciones *hash* que dependen de parámetros externos, tales como generadores de números pseudoaleatorios o la fecha. Tampoco son deterministas aquellas funciones *hash* que dependen de la dirección de memoria en la que está almacenada la cadena de entrada. Esa dirección es accidental y no se considera un cambio de la cadena entrada en sí. De hecho puede cambiar dinámicamente durante la propia ejecución del algoritmo de la función *hash*.

7.5.7 CON NORMALIZACIÓN DE DATOS

En algunas aplicaciones, las cadenas de entrada pueden contener características que son irrelevantes cuando comparamos las cadenas. Por ejemplo, en algunas aplicaciones las mayúsculas pueden ser irrelevantes. Por tanto, para hallar el valor hash es interesante ignorar las distinciones no relevantes entre las cadenas de entrada. De esta forma cadenas distintas con diferencias no relevantes, tienen asociados valores *hash* iguales.

"Funciones Hash Este tipo de funciones se caracterizan por cumplir propiedades que las hacen idóneas para su uso en sistemas que confían en la criptografía para dotarse de seguridad,cuya propiedad la calidad de una función resumen viene definida con base en la satisfacción de ciertas propiedades deseables en el contexto en el que se va a usa"

7.5.8 ESTRUCTURA DE FUNCIONES HASH

7.5.8.1 EJEMPLOS Y APLICACIONES DE PROTOCOLOS HASH

Las funciones son usadas en múltiples campos. Ejemplos:

Herramienta básica para la construcción de utilidades más complejas:

- Construcción de estructuras de datos: Su uso en distintas estructuras de datos hacen más eficientes las búsquedas, y/o permiten asegurar los datos que contienen. Ejemplos: tablas resumen, árboles de Merkle, listas resumen.
- Construcción de esquemas de compromiso. Los esquemas de compromiso permiten que una entidad elija una valor entre un conjunto finito de posibilidades de tal forma que no pueda cambiarla. Esa entidad no tiene que revelar su elección hasta si acaso el momento final (la elección puede permanecer oculta).
- Construcción de algoritmos de cifrado/descifrado. Por ejemplo se usa en la construcción de cifradores de flujo y de cifradores de bloque.
- Construcción de algoritmos generadores de números pseudoaleatorios.
- Construcción de cadenas pseudoaleatorias. Por ejemplo el llamado modelo de oráculo aleatorio se basa en considerar que funciones *hash* con ciertas propiedades se comportan como funciones que escogen cadenas al azar, se usa para el estudio de la seguridad los esquemas criptográficos.
- Construcción de algoritmos de testeo de pertenencia o no a un conjunto.- Se han usado funciones *hash* para la construcción de acumuladores criptográficos y filtros de Bloom. Estas tecnologías permiten establecer mecanismos que permiten pronunciarse, a veces con cierto grado de error, sobre la pertenencia o no a cierto conjunto.
- Construcción de métodos de generación de sellos de tiempo confiables.

7.5.8.2 HERRAMIENTA PARA PROTEGER LA INTEGRIDAD

- En la firma digital
  - Como dato que se firma: en los algoritmos de firma convencionales normalmente en lugar de firmar todo el contenido se suele ser firmar solo el valor *hash* del mismo. Algunas de las motivaciones para hacer esto son:

    Cuando se usa para firmar algoritmos de firma por bloques donde los mensajes son más largos que el bloque, no es seguro firmar mensajes bloque a bloque ya que un enemigo podría borrar bloques del mensaje firmado o insertar bloques de su elección en el mensaje antes de que sea firmado. Al usar una función hash hacemos una transformación que hace a la firma dependiente de todas las partes del mensaje.

Normalmente los valores hash son mucho más cortos que los datos originales de entrada. Se puede mejorar mucho la velocidad de firma firmando el valor hash en lugar de firmar el dato original.

Si los mensajes a firmar pueden tener cierta estructura algebraica y el algoritmo de firma se comporta de forma que el sistema resultante puede ser vulnerable a criptoanálisis con ataques de texto escogido, podemos usar funciones hash para destruir esta estructura algebraica.

7.5.8.3 HERRAMIENTAS VINCULADAS A LA AUTENTICACIÓN Y CONTROL DE ACCESO

- Autenticación de entidades: por ejemplo, es frecuente el uso para este propósito de funciones resumen deterministas con clave secreta que tienen ciertas propiedades (Códigos de autenticación de mensajes). En estos esquemas tanto el servicio de autenticación, o verificador, como la entidad que se quiere autenticar mantienen en secreto la clave de la función resumen. El esquema funciona de la siguiente forma: El que se quiere autenticar genera un mensaje y calcula su valor resumen. Estos dos datos se mandan al verificador. El verificador comprueba que el valor resumen se corresponde con el mensaje enviado y de esta forma verifica que la entidad tiene la clave secreta y por otra parte puede asegurar que el mensaje es íntegro (no ha sido modificado desde que se calculó el valor resumen). Observar que el esquema no tiene la propiedad del no-repudio por parte del que se quiere autenticar ya que el verificador, al disponer de la clave secreta, puede generar también los valores resumen.
- Protección de claves: para comprobar la corrección de una clave no es necesario tener la clave almacenada, lo que puede ser aprovechado para que alguien no autorizado acceda a ella, sino almacenar el valor resumen resultante de aplicar una función resumen determinista. De esta forma para verificar si una clave es correcta basta con aplicar la función resumen y verificar si el resultado coincide con el que tenemos almacenado. Si además queremos que la contraseña sea de un solo uso podemos usar cadenas de resumen como en S/KEY

Herramienta para la identificación y la rápida comparación de datos: Se pueden usar funciones *hash* para proporcionar una identificación de objetos o situaciones. Una buena función *hash* para este propósito debería ser rápida y asegurarse de que dos objetos o situaciones que se considerar iguales den lugar al mismo valor *hash*. Observar que dos objetos o situaciones pueden ser considerados iguales sin ser idénticos. Por ejemplo podemos considerar iguales a dos ficheros que son distintos bit a bit porque realmente son la digitalización de la misma película. Es labor del diseño de la función *hash* capturar la esencia del criterio de igualdad. Por otra parte la evaluación de la función *hash* debería ser poco costosa para facilitar la rápida comparación de elementos candidatos a ser iguales y de esta forma poder implementar algoritmos de búsqueda rápidos.

- Huellas digitales: el uso de funciones *hash* aplicados a cadenas permiten obtener valores *hash* que pueden usarse detectar fácilmente la aparición de esos datos en distintos sitios. Pueden ser usados para distintos usos como búsqueda de virus, autenticación con datos biométricos, detección de copias, etc. La idea puede usarse más allá de textos y ser aplicado a cualquier tipo de contenido multimedia:[2829] Las funciones *hash* específicamente diseñadas para este propósito obtienen valores *hash* que permiten detectar características intrínsecas del contenido multimedia, de forma que se pueda identificar si dos archivos diferentes se corresponden con el mismo contenido multimedia. Como aplicación práctica de este tipo de algoritmo tenemos los programas que se ejecutan en dispositivos móviles y que son capaces de adivinar el título de la canción que está

sonando en la habitación solamente capturando el sonido y comparándolo con estos valores *hash*. Este tipo de algoritmos también se puede utilizar para protección de contenidos multimedia ya que permite validar automáticamente si cierto fichero multimedia está protegido o no por derechos de autor.

- Identificación de contenidos: en algunas aplicaciones se usa el valor *hash* de un contenido multimedia para identificar ese contenido independientemente de su nombre o ubicación. Esto es ampliamente usado en redes Peer-to-peer que intercambian de archivos, tales como Kazaa, Ares Galaxy, Overnet, BitTorrent.
- Identificar un registro en una base de datos y permitir con ello un acceso más rápido a los registros (incluso más rápido que teniendo índices).
- Algorítmos de búsqueda de subcadenas: los algoritmos de búsqueda de subcadenas tratan el problema de buscar subcadenas, a la que llaman patrón, dentro de otra cadena a la que llaman texto. Hay algoritmos de este tipo que usan funciones *hash* en su implementación. Ejemplo: algoritmo Karp-Rabin.

7.5.9 DEBILIDADES Y VULNERABILIDADES DE LAS FUNCIONES HASH

Vulnerabilidades y puntos débiles: La mayor problemática que sufren las funciones hash son las colisiones de hash. Se descubrieron las primeras de este algoritmo en el año 1996, lo cuál motivó el desarrollo de nuevos algoritmos. Esto es algo inevitable para estos algoritmos (Ya que al dejar una cadena de la misma longitud siempre, es factible que una cadena muy larga de lo mismo que una de menor tamaño, por dar una posibilidad), sin embargo, es algo que es poco deseable que ocurra.

2. Ataques típicos

Al igual que todos los algoritmos, es susceptible de sufrir un ataque por fuerza bruta. Para los casos de algoritmos de tipo de hash, surgen nuevos ataques, como:

a. Ataque de diccionario: Usar una base de datos que tenga los pares cadena hash - texto claro, de tal manera que introduciendo la cadena hash obtengamos el texto claro (Estos diccionarios son buenos para claves genéricas que siempre se usan como “1234” o “pass”, pero no sirve para nada en caso de que no exista una hash registrado para una clave (Por ejemplo, si utilizamos claves largas y con dígitos raros, como “^¨*^SF”·$·343rdfd3*¨*Ç¨*”).

b. Ataque de cumpleaños: Por la probabilidad de que dos cadenas similares, pero con varias diferencias que generen una misma cadena hash.

c. Rainbow tables

3. Medidas de defensa incluidas y fortalezas

La longitud del hash generado es de 128 bits

Desventajas

Necesidad de ampliar el espacio de la tabla si el volumen de datos almacenados crece. Se trata de una operación costosa.

Dificultad para recorrer todos los elementos. Se suelen emplear //**listas**// para procesar la totalidad de los elementos.

Desaprovechamiento de la memoria. Si se reserva espacio para todos los posibles elementos, se consume más memoria de la necesaria; se suele resolver reservando espacio únicamente para //punteros// a los elementos.

Vulnerabilidades y puntos débiles: La mayor problemática que sufren las funciones hash son lascolisiones de hash el cual esto es algo inevitable para estos algoritmos

8 CIFRADO DE FLUJO

8.1 SECUENCIA PSEUDOALEATORIAS

Los generadores criptográficamente aleatorios tienen la propiedad de que, a partir de una porción de la secuencia arbitrariamente grande, resulta computacionalmente intratable el problema de predecir el siguiente bit de la secuencia. Evidentemente, en el caso que nos ocupa, esta característica se convertirá en una ventaja, ya que es precisamente lo que necesitamos: que por un lado no pueda calcularse la secuencia completa a partir de una porción de ésta, y que a la vez pueda reconstruirse completamente conociendo una pieza de información como la semilla del generador

8.2 TIPOS DE GENERADORES DE SECUENCIA

Los generadores que se emplean como cifrado de flujo pueden dividirse en dos grandes grupos, dependiendo de que se empleen o no fragmentos anteriores del mensaje cifrado a la hora de calcular los valores de la secuencia. Comentaremos brevemente en esta sección sus características básicas

8.2.1 SÍNCRONOS

Un generador síncrono es aquel en el que la secuencia es calculada de forma independiente tanto del texto en claro como del texto cifrado. En el caso general, ilustrado en la, viene dado por las siguientes ecuaciones:

$$s_i+1 = g\ (s_i, k)$$

$$o_i = h\ (s_i, k)$$

$$c_i = w\ (m_i, o_i)$$

Donde k es la clave, si es el estado interno del generador, s0 es el estado inicial, o_i es la salida en el instante i, m_i y c_i son la i–ésima porción del texto claro y cifrado respectivamente, y w es una función reversible, usualmente or exclusivo. En muchos casos, la función h depende únicamente de s_i, siendo k = s_o

Existe también una debilidad intrínseca a los métodos de cifrado de flujo basados en generadores síncronos que vale la pena destacar: si un atacante conoce parte del texto claro, podrá sustituirlo por otro sin que lo advierta el legítimo destinatario. Supongamos que m_i es una porción del mensaje original conocida por el atacante, y c_i el trozo de mensaje cifrado correspondiente a él. Sabemos que

$$c_i = w\ (m_i\ ,\ o_i)$$

siendo o_i el trozo de secuencia Pseudoaleatorias que fue combinado con el texto en claro. Puesto que w es una función reversible, podemos recuperar los o_i)asociados al fragmento conocido m_i. Calculamos entonces:

$$c_i' = w(m_i', o_i)$$

Siendo un mensaje falso de nuestra elección. Seguidamente sustituimos los originales por los que acabamos de obtener. Cuando el destinatario descifre el mensaje alterado, obtendrá la porción de mensaje, en lugar del original, de forma totalmente inadvertida. Esta circunstancia aconseja emplear estos métodos de cifrado en combinación con técnicas que garanticen la integridad del mensaje(Lucena, 2010)

8.2.2 ASÍNCRONOS

Un generador de secuencia asíncrono o auto-sincronizado es aquel en el que la secuencia generada es función de una semilla, más una cantidad fija de los bits anteriores del mensaje cifrado, Formalmente:

$$o_i = h\ (k, c_{i-t}, c_{i-t+1}, \ldots, c_{i-1})$$

$$c_i = w\ (o_i\ ,\ m_i)$$

Donde k es la clave, m_iy c_i son la i–ésima porción del texto claro y cifrado respectivamente y w es una función reversible. Los valores c_{-t} , c_{-t+1}, . . . , c_{-1} constituyen el estado inicial del generador.

Esta familia de generadores es resistente a la pérdida o inserción de información, ya que acaba por volver a sincronizarse automáticamente, en cuanto llegan t bloques correctos de forma consecutiva. También será sensible a la alteración de un mensaje, ya que si se modifica la unidad de información c_i , el receptor tendrá valores erróneos de entrada en su función h hasta que se alcance el bloquec_{i+t}, momento a partir del cual la transmisión habrá recuperado la

sincronización. En cualquier caso, al igual que con los generadores síncronos, habrá que introducir mecanismos de verificación

Una propiedad interesante de estos generadores es la dispersión de las propiedades estadísticas del texto claro a lo largo de todo el mensaje cifrado, ya que cada dígito del mensaje influye en todo el criptograma. Esto hace que los generadores asíncronos se consideren en general más resistentes frente a ataques basados en la redundancia del texto en claro.

8.3 REGISTROS DE DESPLAZAMIENTO RETROALIMENTADOS

Los registros de desplazamiento retroalimentados (feedback shift registers, o FSR en inglés) son la base de muchos generadores de secuencia síncronos para cifrados de flujo. Dedicaremos esta sección a analizar su estructura básica y algunas de sus propiedades.

8.3.1 REGISTROS DE DESPLAZAMIENTO RETROALIMENTADOS LINEALES

Estos registros, debido a que permiten generar secuencias con períodos muy grandes y con buenas propiedades estadísticas, además de su bien conocida estructura algebraica y su facilidad para ser implementados por hardware, se encuentran presentes en muchos de los generadores de secuencia propuestos en la literatura.

Un registro de desplazamiento retroalimentado lineal L es un conjunto de L estados, {S_0, S_1, . . . , S_{L-1}}, capaces de almacenar un bit cada uno Esta estructura viene controlada por un reloj que coordina los flujos de información entre los estados. Durante cada unidad de tiempo se efectúan las siguientes operaciones:

1. El contenido de S0 es la salida del registro
2. El contenido de Si es desplazado al estado S_I−1, para 1 ≤ i ≤ L − 1.
3. El contenido de s_{L-1}se calcula como la suma módulo 2 de los valores de un subconjunto prefijado de L.

Un generador de estas características devolverá, en función de los valores iniciales de los estados, y del subconjunto concreto de L empleado en el paso 3, una secuencia de salidas de carácter periódico en algunos casos, la secuencia será periódica si ignoramos una cierta cantidad de bits al principio

8.3.2 REGISTROS DE DESPLAZAMIENTO RETROALIMENTADOS NO LINEALES

Un registro de desplazamiento retroalimentado general (o no lineal) L es un conjunto de L estados, $\{S_0, S_1, \ldots, S_{L-1}\}$, capaces de almacenar un bit cada uno. Durante cada unidad de tiempo se efectúan las siguientes operaciones:

1. El contenido de S_0 es la salida del registro.
2. El contenido de S_i es desplazado al estado S_{I-1}, para $1 \leq i \leq L - 1$.
3. El contenido de S_{L-1} se calcula como una función booleana

$$f(S_{J-1}, S_{J-2}, \ldots, S_{J-L}),$$

donde S_{j-i} es el contenido del registro S_{L-i} en el estado anterior

8.3.3 COMBINACIÓN DE REGISTROS DE DESPLAZAMIENTO

En la mayoría de los casos, los registros de desplazamiento retroalimentados no lineales presentan unas mejores condiciones como generadores de secuencia que los generadores de tipo lineal. Sin embargo, la extrema facilidad de implementación por hardware de estos últimos ha llevado a los diseñadores a estudiar diferentes combinaciones de registros lineales, de tal forma que se puedan obtener secuencias mejores.

En general, se emplearían n generadores lineales y una función f no lineal para combinar sus salidas, de tal forma que cada bit de la secuencia se obtendría mediante la expresión

$$f(R_1, R_2, \ldots, R_n)$$

siendo R_i la salida del i–ésimo registro de desplazamiento lineal.(Lucena, 2010)

8.4 CIFRADO POR BLOQUES EN MODO OFB

El cifrado por bloques es una unidad de cifrado de clave simétrica que opera en grupos de bits de longitud fija, llamados bloques, aplicándoles una transformación invariante. Cuando realiza cifrado, una unidad de cifrado por bloques toma un bloque de texto plano o claro como entrada y produce un bloque de igual tamaño de texto cifrado. La transformación exacta es controlada

utilizando una segunda entrada. El descifrado es similar: se ingresan bloques de texto cifrado y se producen bloques de texto plano.

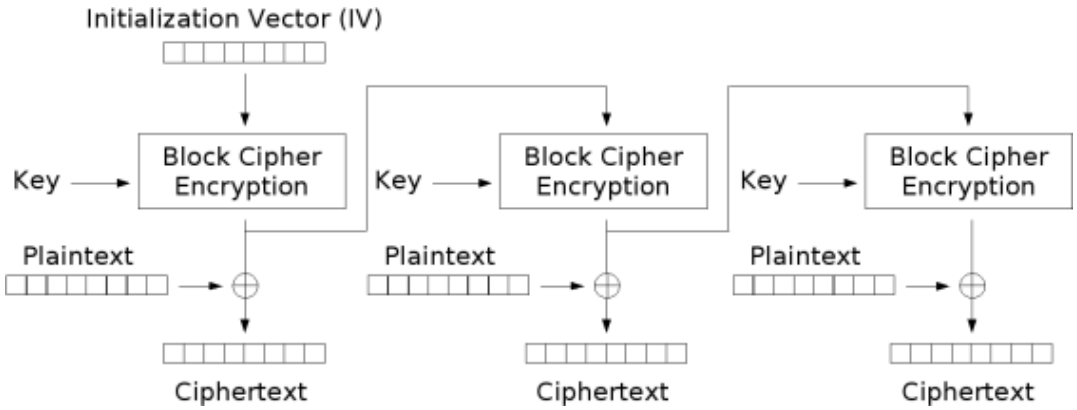


Fig. 25 Modo de encriptación OFB(PAAR,1998)

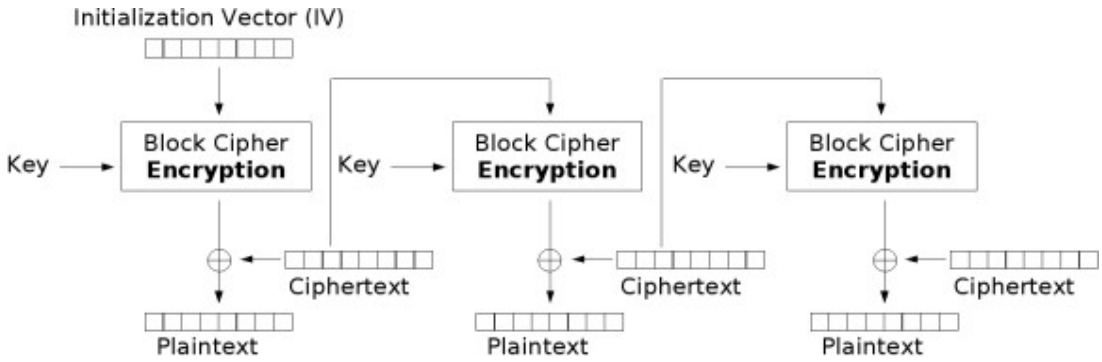


Fig. 26 Modo de des encriptación OFB (PAAR,1998)

Los modos cipher feedback (CFB) y output feedback (OFB) hacen que el cifrado en bloque opere como una unidad de flujo de cifrado: se generan bloques de flujo de claves, que son operados con XOR y el texto en claro para obtener el texto cifrado. Al igual que con otras unidades de flujo de cifrado, en OFB al intercambiar un bit en el texto cifrado produce texto cifrado con un bit intercambiado en el texto plano en la misma ubicación, en CFB un bit erróneo en el texto cifrado genera (1+64/m) bloques de texto claro incorrectos (siendo m la longitud del flujo en el que se divide el bloque). Bueno para las actividades de seguridad y alta disponibilidad. (PAAR,C., PELZL, J. 1998).

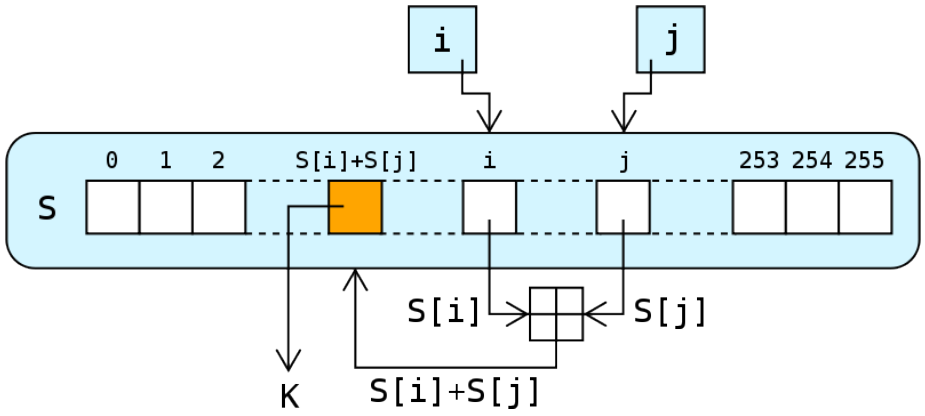
8.5 Algoritmo RC4

RC4 es un esquema de cifrado de flujo (no basado en bloques) simétrico. Fue diseñado por Ron Rivest (la R de RSA) en 1987. Originalmente era secreto, pero se filtró en 1994 a través de una lista de correo.

Es un esquema de cifrado extremadamente simple y puede implementarse en software de forma muy eficiente. Esto lo ha convertido en uno de los esquemas de cifrado más utilizados del mundo.

Sin embargo, RC4 hace tiempo que no es considerado un algoritmo seguro. RC4 es conocido por ser el mismo esquema de cifrado usado por WEP (Wired Equivalent Privacy), sistema criptográfico totalmente roto hoy en día.

Menos conocido es que RC4 es usado aún en aproximadamente la mitad de transmisiones TLS que ocurren en el mundo actualmente, desde para consultar tu correo hasta para establecer transferencias bancarias. Hoy en día el interés por el RC4 parte de querer conocer hasta qué punto está roto y cómo de vulnerables son los sistemas que lo utilizan.



8.6 ESTRUCTURA Y FUNCIÓN MDC

En general, los MDC[14] se basan en la idea de funciones de compresión, que dan como resultado bloques de longitud fija a partir de bloques de longitud fija b, con a < b. Estas funciones se encadenan de forma iterativa, haciendo que la entrada en el paso i sea función del i -ésimo bloque del mensaje (mi) y de la salida del paso i − 1 En general, se suele incluir en alguno de los bloques del mensaje m —al principio o al final—, información sobre la longitud total del mensaje. De esta forma se reducen las probabilidades de que dos mensajes con diferentes longitudes den el mismo valor en su resumen.(Lucena, 2010)

[14] *Se basan en la idea de funciones de compresión, que dan como resultado bloques de longitud fija*

8.7 ALGORITMO MD5

Se trata de uno de los más populares algoritmos de generación de signaturas, debido en gran parte a su inclusión en las primeras versiones de PGP. Resultado de una serie de mejoras sobre el algoritmo MD4[15], diseñado por Ron Rivest, procesa los mensajes de entrada en bloques de 512 bits, y produce una salida de 128 bits.

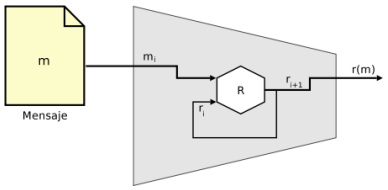


Fig27. Estructura iterativa de una función resumen. R representa la función de compresión, m es el mensaje completo, mi el i-ésimo trozo de m, y ri la salida de la función en el paso i(Lucena, 2010)

Siendo m un mensaje de b bits de longitud, en primer lugar, se alarga m hasta que su longitud sea exactamente 64 bits inferior a un múltiplo de 512. El alargamiento se lleva a cabo añadiendo un 1 seguido de tantos ceros como sea necesario. En segundo lugar, se añaden 64 bits con el valor de b, empezando por el byte menos significativo. De esta forma tenemos el mensaje como un número entero de bloques de 512 bits, y además le hemos añadido información sobre su longitud.

8.7.1 HACKEADO DEL ALGORITMO MD5

En el mes de Agosto de 2004 se publica la noticia acerca de la vulnerabilidad de algoritmo MD5, descubierta por el matemático francés Antoine Joux. Este algoritmo es frecuentemente utilizado para calcular el resumen o "hash" de un determinado conjunto de bytes. Una función de "hash" deber cumplir dos condiciones de seguridad esenciales:

1.- Imposibilidad de obtener el texto original a partir de su resumen.

2.- No debe ser computacionalmente factible encontrar dos textos que generen el mismo valor de resumen o hash. Este hecho se conoce como colisión. MD5 produce un valor de "hash" de 128 bits, que supone una media de 2^64 intentos para encontrar una colisión.

[15] *Algoritmo diseñado por Ron Rivest (autor de la RC2, RC4 y RC5 y uno de los coautores del RSA). Producen un resume de 128 bits del mensaje*

Debido al descubrimiento de un método fácil para generar colisiones de hash, la tendencia actual es el uso del algoritmo SHA-1 como función de resumen. SHA-1 [16](Security Hash Algorithm 1) fue desarrollado por la "National Security Agency" en 1995, produce un valor de resumen de 160 bits lo que proporciona una media de 2^80 intentos para encontrar una colisión por el método de prueba y error, nivel de seguridad bastante alto.

Uno de los más prestigiosos criptógrafos de la actualidad, Bruce Schneier, aborda el hallazgo del método para encontrar colisiones en SHA-1 en 2^69 intentos en su página web. Valorando la magnitud del descubrimiento, Schneier afirma que es un avance interesante para el mundo de la criptografía que puede ayudar en el desarrollo de nuevos algoritmos de seguridad. (ORTEGA, J., LOPEZ, M. 2006)

8.8 ALGORITMO SHA-1

SHA-1(Secure Hash Algorithm 5 o Algoritmo de Hash Seguro 1),toma como entrada mensajes con longitud máxima de 264 bits y produce un numero 160 bits Digital Signature Algorithm(DSA9 es el estándar de United States Federa Govemment para firma digital.Su desarrollo se atribuye a David W.Karavitz, de la National Security Agency.Fue presentado por NIST en agosto de 1991,adoptado como estándar en 1993 y con última versión de 2000.Es un algoritmo exclusivo de firma electrónica basado en clave publica pero no vale para comunicaciones confidenciales (Mayor & Andrés, n.d.)

El algoritmo SHA-1 fue desarrollado por la NSA, para ser incluido en el estándar DSS (Digital Signature Standard). Al contrario que los algoritmos de cifrado propuestos por esta organización, SHA-1 se considera seguro[17] y libre de puertas traseras, ya que el hecho de que el algoritmo sea realmente seguro favorece a los propios intereses de la NSA. Produce firmas de 160 bits, a partir de bloques de 512 bits del mensaje original

SHA-1 se considera seguro y libre de puertas traseras, ya que el hecho de que el algoritmo sea realmente seguro favorece a los propios intereses. Produce firmas de 160 bits, a partir de bloques de 512 bits del mensaje original

8.8.1 ATAQUES CONTRA SH-1

[16] *(Security Hash Algorithm 1) produce un valor de resumen de 160 bits lo que proporciona una media de 2^80 intentos para encontrar una colisión por el método de prueba y error*

[17] *Desafortunadamente, la seguridad de SHA-1 ha quedado puesta en entredicho debido a los avances conseguidos en 2004 y 2005 por un equipo de criptólogos chinos, liderado por Xiaoyun Wang*

La resistencia del algoritmo SHA-1 se ha visto comprometido a lo largo del año 2005. Después de que MD5, entre otros, quedara seriamente comprometido en el 2004 por parte del equipo de investigadores chinos, el tiempo de vida de SHA-1 quedó visto para sentencia45 . El mismo equipo de investigadores chinos, compuesto por Xiaoyun Wang, Yiqun Lisa Yin y Hongbo Yu46 (principalmente de la Shandong University en China), ha demostrado que son capaces de romper el SHA-1 en al menos 269 operaciones, unas 2000 veces más rápido que un ataque de fuerza bruta (que requeriría 280 operaciones). Los últimos ataques contra SHA-1 han logrado debilitarlo hasta 263. Según el NIST: "Este ataque es de particular importancia para las aplicaciones que usan firmas digitales tales como marcas de tiempo y notarías. Sin embargo, muchas aplicaciones que usan firmas digitales incluyen información sobre el contexto que hacen este ataque difícil de llevar a cabo en la práctica"

A pesar de que el NIST contempla funciones de SHA de mayor tamaño (por ejemplo, el SHA-512, de 512 bits de longitud), expertos de la talla de Bruce Schneier abogan por, sin llamar a alarmismos, buscar una nueva función hash estandarizada que permita sustituir a SHA -1. Los nombres que se mencionan al respecto son Tiger, de los creadores de Serpent, y WHIRLPOOL, de los creadores de AES[18]

8.9 SEGURIDAD DE FUNCIONES MDC

Puesto que el conjunto de posibles mensajes de longitud arbitraria es infinito, y el conjunto de posibles valores de una función MDC es finito, inevitablemente habrá valores para la función que se correspondan con más de un mensaje. De hecho, puede demostrarse que al menos un valor de la función MDC se corresponde necesariamente con infinitos mensajes, y es razonable sospechar que, en general, cada uno de los posibles valores de la función va a corresponder con infinitos mensajes. En consecuencia, siempre va a ser posible, dado un valor r(m), encontrar un m0 tal que r(m) = r (m0). La fortaleza de las funciones MDC radica, pues, en la dificultad que plantea encontrar el m0. Llamaremos colisión a un par de mensajes (m, m0) tales que r(m) = r (m0). De lo argumentado en el párrafo anterior, podemos deducir que todos los algoritmos MDC presentan colisiones. Distinguiremos no obstante dos tipos de estrategias para hallarlas, con objeto de delimitar el grado de compromiso que pueden provocar en un algoritmo concreto:

De preimagen: El atacante parte de un mensaje m, y calcula otro mensaje m0 que colisiona con el primero.

[18] *Advanced Encryption Standard (AES) es uno de los algoritmos de cifrado más utilizados y seguros actualmente disponibles*

De colisión propiamente dicha: El atacante se limita a buscar dos valores m y m0 que colisionen, pero desconoce inicialmente tanto sus valores como el que tomará la función resumen.

En el primer caso, el MDC queda comprometido de manera grave, ya que bastará con sustituir un m con el m0 que hayamos calculado para falsificar un mensaje. De todas formas, es bastante difícil que el m0 tenga un aspecto válido. Piénsese por ejemplo en un mensaje m de texto: nuestra técnica de preimagen tendría que ser capaz de generar otro mensaje m0, también de texto, y con un significado concreto, para que la falsificación tuviera interés. Lo más habitual es que el m0 obtenido tenga un aspecto más o menos aleatorio, lo cual le conferiría una utilidad mucho más limitada, como por ejemplo la intoxicación de redes de comunicación con datos erróneos que puedan pasar por auténticos(Lucena, 2010)

Podemos demostrar que al menos un valor de la función MDC se corresponde necesariamente con infinitos mensajes, sospechar que, en general, cada uno de los posibles valores de la funciónva a corresponder con infinitos mensajes

9 Bibliografía

(s.f.).

Acosta, E. S. (27 de Diciembre de 2011). *criptoanalisisy teoria de codigos.* Obtenido de : https://esacosta.files.wordpress.com/2012/09/cripto-ufv-1_0-linea.pdf

Angel, J. d. (2003). *Criptografía para Principiantes* .

Ardita , J., Caratti , M., Do Cabo, R., & Giusto , M. (1998). *ESTEGANOGRAFÍA.* Argentina.

Areito, J. (1998). Analisis y Evaluacion de la Seguridad enRedes. *Revista española de electronica Nº 521*.

Castejon. (s.f.). *Criptografia.* Obtenido de Criptografia, Ataque sobre el metodo de Hill: https://repositori.udl.cat/bitstream/handle/10459.1/45714/Castej%C3%B3n.pdf?sequence=1

Cavalli, J. I., Guanca, D. A., & Juncos, J. R. (2002). *Firmas Digitales-Infraestructura PKI*. Obtenido de http://www.hfernandezdelpech.com.ar/PUBLICAtrabajosFirmaDigitalInfrPKI.htm

Conde, A. (s.f.). *Vulderabilidades Criptograficas.* Obtenido de https://eseida.wikispaces.com/file/view/Vulnerabilidades+Criptogr%C3%A1ficas.pdf

CORRALES, H., CILLERUELO, C., & CUEVAS, A. (2013). *Criptografia y Metodos de Cifrado.*

CRIPTO_RED. (s.f.). *http://www.criptored.upm.es/thoth/material/texto/pildora007.pdf*.

Dávila, L. M. (25 de 9 de 2011). *INVESTIGACION DE CRIPTOGRAFIA.* Obtenido de https://es.slideshare.net/CesarCuamatzi/criptografia-11362555

DSIC-UPV. (2012). *Criptografia Clasica.* Obtenido de Criptoanalisis polialfabetico: http://users.dsic.upv.es/asignaturas/eui/cri/CrClasica.pdf

EcuRed. (6 de Octubre de 2011). *Conocimiento con todos y para Todos*. Obtenido de https://www.ecured.cu/Criptoan%C3%A1lisis

FACET. (s.f.). *Espectro Extendido.* Obtenido de https://catedras.facet.unt.edu.ar/ft/wp-content/uploads/sites/123/2017/03/11-Espectro-Expandido-1.pdf

Franchi, M. R. (2012). *Algoritmos de Encriptacin de Clave Aaimetrica.*

G´erard Maze, B. M. (s.f.). *Trapdoor functions.* Obtenido de http://user.math.uzh.ch/maze/Articles/DissJoli.pdf

Gil, P. C. (2002). *INTRODUCCION A LA CRIPTOGRAFÍA.*

Heineken Team. (2001). *SEGURIDAD Y PROTECCIÓN DE LA INFORMACIÓN.* Obtenido de http://tecno.unsl.edu.ar/redes%202008/seguridadinformatica.pdf

Hellman:, W. D. (Nov. 1976, pp: 644-654.). *"New Directions in Cryptography",IEEE Transactions on Information.*

Hernan Cordova, P. C. (Octubre de 2005). *Revista Tecnologica ESPOL.* Obtenido de https://44f9e77d-a-62cb3a1a-s-sites.googlegroups.com/site/ocwcursodecomunicacionesunefa/captulo-5/Estudio%2CModelamientoySimulaci%C3%B3ndeSistemasdeEspectroEnsanchado.pdf?attachauth=ANoY7cr4KyKzFyBmKKUqntNApI2NmkIkgyGEjupJE-xY6jDsW3XgK-faIW6pAu6vBgr_4UmbP

HUERTA, A. V. (2002). *SEGURIDAD EN UNIX Y REDES.*

Iglesias, P. F. (28 de 04 de 2015). *Esteganografía, el arte de ocultar informacion sensible*. Obtenido de https://www.pabloyglesias.com/mundohacker-esteganografia/

Inteco. (2001). *Esteganografia, El arte de ocultar Información.* Obtenido de http://www.egov.ufsc.br/portal/sites/default/files/esteganografia1.pdf

Jesus, J. d. (2002). *Ssitema RSA, Fundamentos matematicos Primos Fuertes.* Obtenido de http://www.siger.gob.mx/docs/rsa.pdf

Leon, K. (2005). *Fundamentos matematicos criptografia asimetrica.* Obtenido de http://tesis.pucp.edu.pe/repositorio/bitstream/handle/123456789/131/LEON_KATIA_ENCRIPTACION_RSA_TEXTO.pdf?sequence=1

Litwak, N., & Escalante, J. (2004). Seguridad Informática y Criptografía. En *Seguridad Informática y Criptografía.* Argentina.

Lopez, G. F. (2007). *Seguridad en Sistemas Operativos.* Coruña. Obtenido de http://sabia.tic.udc.es/docencia/ssi/old/2006-2007/docs/trabajos/03%20-%20Seguridad%20en%20Sistemas%20Operativos.pdf

Lucerna López, J. M. (2010). *Criptografía y Seguridad en Computadoras.*

Lujan, U. N. (2016). *Riesgo vs. Seguridad de la Información.*

Master, D. (2004). *CRIPTOSISTEMAS INFORMÁTICOS.*

MASTER, D. (2004). *CRIPTOSISTEMAS INFORMÁTICOS.*

Mendoza, N. S.-J. (8 de mayo de 2017). *"SPREAD SPECTRUM" UDABOL Universidad de Aquino Bolivia.*

Miret, J. M. (06 de 06 de 2013). *Alan Turing: El descifrado de la máquina Enigma.* Obtenido de http://blogs.elpais.com/turing/2013/06/alan-turing-el-descifrado-de-la-maquina-enigma.html

Morant, J. (s.f.). Seguridad y proteccion de la Informacion. *ED. Centros de Estudios Ramon Areces, S.A.*

Noelia Litwak, J. E. (2004). *Seguridad Informatica y Criptografia.* Argentina. Obtenido de http://exa.unne.edu.ar/informatica/SO/Criptografia04.pdf

Ocon, M. G. (2013). *Implementación del algoritmo de cifrado.* Madrid. Obtenido de https://e-archivo.uc3m.es/handle/10016/17607

Patricia, X. S. (2009). Obtenido de http://hdl.handle.net/10016/6173

Pousa., L. A. (6 de MARZO de 2013). ALGORITMO DE CIFRADO SIMÉTRICO AES ACELERACION DE TIEMPO DE COMPUTO SOBRE ARQUITECTURA MULRICORE. Obtenido de http://postgrado.info.unlp.edu.ar/Carreras/Especializaciones/Redes_y_Seguridad/Trabajos_Finales/Pousa_Adrian.pdf

PREUKSCHAT, A. (14 de Enero de 2014). *Aplicaciones y funciones unidireccionales*. Obtenido de https://www.oroyfinanzas.com/2014/01/aplicacion-funciones-hash-unidireccionales-bitcoin/

Propiedades de las secuencias pseudoaleatorias. (s.f.). Obtenido de http://bibing.us.es/proyectos/abreproy/11479/fichero/3-Sistemas+CDMA.pdf

Rey, A. M. (2015). *Criptografia matematica para proteger la informacion.* Obtenido de http://diarium.usal.es/delrey/files/2013/12/Criptografia-Matematicas-para-proteger-la-informacion1.pdf

Scolnick, H. D. (2004). *Fundamentos Matematicos Primalidad Rabin-Miller.* Obtenido de http://www-2.dc.uba.ar/materias/crip/docs/rsamath01.pdf

Scolnik, H. D. (Abril de 2004). *Fundamentos Matematicos RSA.* Obtenido de http://www-2.dc.uba.ar/materias/crip/docs/rsamath01.pdf

Sevilla, U. d. (2008). *CRIPTOGRAFÍA.* Sevilla. Obtenido de http://ma1.eii.us.es/Material/Cripto_ii_Simetrica.pdf

SM.1055, R. U.-R. (1997). *EFICACIA EN LA UTILIZACIÓN DEL ESPECTRO* .

software, F. l. (2018). Encriptación con AES . Obtenido de http://www.fls-es.com/images/AES.pdf

Solis, P. (s.f.). *Sistemas de Espectro Disperso.* Obtenido de http://prof.usb.ve/tperez/docencia/3413/contenido/SS.pdf

Spohr, J. R. (3 de Junio de 2007). *Sistema de Gestión de la Seguridad de la Información.* Obtenido de http://www.iso27000.es/download/doc_sgsi_all.pdf

Stallings, W. (1997). *Seguridad en las Comunicaciones.* Obtenido de http://www.isa.uniovi.es/docencia/redes/Apuntes/tema8.pdf

Standards, N. B. (Nov. 2001.). *Data Encryption Standard,.* Obtenido de National Bureau of Standards.

Starbase. (s.f.). *Criptografia*. Obtenido de http ://starbase.cs.trincoll.edu/-crypto/historical/vigenere.htm

Triana Laverde, J. G. (s.f.). *http://www.morfismos.cinvestav.mx/P.* Obtenido de http://www.morfismos.cinvestav.mx/Portals/morfismos/SiteDocs/Articulos/Vol19_2/PARTE-6-Triana-Ruiz.pdf?ver=2016-02-11-123636-227

UDLAP, C. (s.f.). *Secuencias de Longitud Máxima ESPECTRO EXTENDIDO.* Obtenido de http://catarina.udlap.mx/u_dl_a/tales/documentos/lem/tecuanhuehue_r_j/capitulo2.pdf

UIT, A. d. (s.f.). *UTILIZACIÓN DE TÉCNICAS DE ESPECTRO ENSANCHADO.* Obtenido de https://www.itu.int/dms_pubrec/itu-r/rec/sm/R-REC-SM.1055-0-199407-I!!PDF-S.pdf

UNAM. (2004). *Criptografía.* Obtenido de http://www.matem.unam.mx/rajsbaum/cursos/web/presentacion_seguridad_1.pdf

Ventura, J. B. (24 de 10 de 2011). *Criptografia*. Obtenido de https://sites.google.com/site/ybnbackup/sistemas-de-encriptacion-1/criptografia

W, C. l. (1998). *Sistema de Comunicaciones Digitales.*

Zafra, J. N. (s.f.). *SPREAD SPECTRUM.* Obtenido de https://revistas.udistrital.edu.co/ojs/index.php/reving/article/view/3502/5063

Ing. Diego Fernando Veloz Chérrez
Diego Fernando Veloz Chérrez, Máster en Tecnologías de la Información con especialidad de Seguridad y redes de la Universidad de Griffith Australia. Nació en Riobamba – Ecuador y se mudó a Quito para estudiar Ingeniería en Electrónica y Telecomunicaciones en la Escuela Politécnica Nacional. Siempre estuvo involucrado en el ámbito de las comunicaciones inalámbricas y aplicación de tecnologías innovadoras, siendo uno de los pioneros en el uso e implementación de la tecnología NFC en Ecuador. Tiene experiencia en la empresa privada llegando a ocupar cargos directivos. Postuló en becas nacionales obteniendo una beca para estudiar su postgrado en Australia. Actualmente es docente de la Facultad de Informática y Electrónica de la Escuela Superior Politécnica de Chimborazo en donde se desempeña en diferentes ámbitos: Coordinador de la sección estudiantil IEEE. Coordinador de proyecto de Vinculación ESPOCH-GADM Riobamba. Miembro del Grupo de Investigación de Comunicaciones inalámbricas de la carrera de Telecomunicaciones. Director de proyectos de tesis de pregrado y postgrado. Coordinador de convenio ESPOCH-CNT. Tiene amplia experiencia en el campo de las redes, seguridad de redes, tecnologías inalámbricas, ethical hacking, implementación de sistemas de gestión y en el Internet de las Cosas.

Ing. Marcela Estefanía Mora Campana
Profesional en el área de Electrónica Telecomunicaciones y Redes graduada de la Escuela Superior Politécnica de Chimborazo. Realice mis practicas preprofesionales en el Gobierno Autónomo Descentralizado Municipal del Cantón Riobamba desarrollando actividades en el Soporte de redes LAN y Wifi, Mantenimiento de equipos electrónicos y de telecomunicaciones, Cableado estructurado, Soporte técnico informático ,además de participar y apoyar en actividades que requieran mi especialización, habilidades y destrezas, manejo y administración de plataforma .En la culminación de mi carrera elabore un trabajo de investigación denominado "Diseño e implementación de un prototipo de sistema IoT para la monitorización de datos de presión arterial a través de un sensor foto electrónico " Cuento con cursos, talleres y seminarios en el campo de las Redes, Telecomunicaciones.

Ing. Fabricio Javier Santacruz Sulca
El Ing. Fabricio Javier Santacruz Sulca nacido en Riobamba, Ecuador, realizó sus estudios de grado en la prestigiosa Universidad de la Calabria en Italia, donde obtuvo el título de Ingeniero Electrónico y más tarde, un postgrado en la misma universidad, alcanzando el título de Máster en Ingeniería de las Telecomunicaciones. Con su amplia experiencia en el campo de las telecomunicaciones, es actualmente docente de la Escuela Superior Politécnica de Chimborazo en la facultad de Informática y Electrónica, donde comparte sus conocimientos y habilidades con las futuras generaciones de ingenieros y técnicos en el área de las telecomunicaciones. Su experiencia y formación han contribuido significativamente al desarrollo y avance de las comunicaciones en el ámbito académico y profesional, siendo este libro una importante fuente de conocimiento para aquellos interesados en el campo de las telecomunicaciones.

ISBN: 978-9942-33-670-5

compAs
Grupo de capacitación e investigación pedagógica
@grupocompas.ec
compasacademico@icloud.com