USERS

INCLUYE VERSIÓN DIGITAL GRATIS

CRIPTOGRAFÍA

DESDE LOS SISTEMAS CLÁSICOS HASTA EL FUTURO DE LA PRIVACIDAD

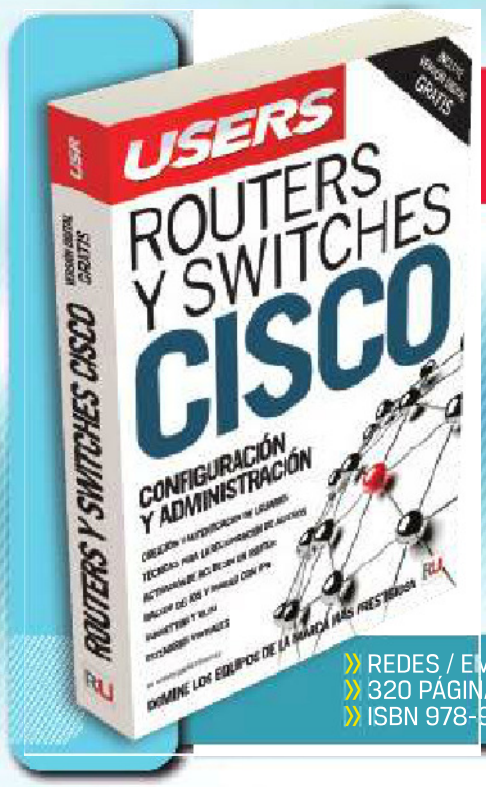
- FUNDAMENTOS MATEMÁTICOS
- MÉTODOS Y SISTEMAS DE CIFRADO
- TÉCNICAS DE ATAQUE
- FIRMAS Y CERTIFICADOS DIGITALES
- CRIPTOANÁLISIS
- PROTOCOLOS CRIPTOGRÁFICOS

por FEDERICO PACHECO

CONOZCA CÓMO SE PROTEGEN HOY LOS DATOS MÁS SENSIBLES

RU

CONÉCTESE CON LOS MEJORES LIBROS DE COMPUTACIÓN

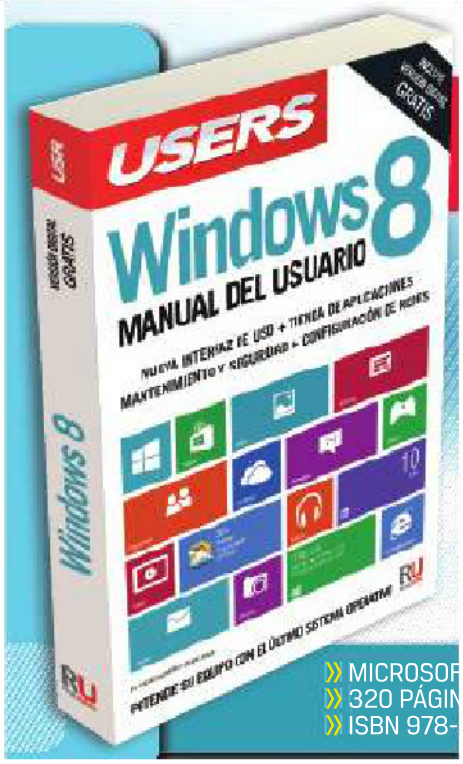


DOMINE LOS EQUIPOS DE LA MARCA MÁS PRESTIGIOSA

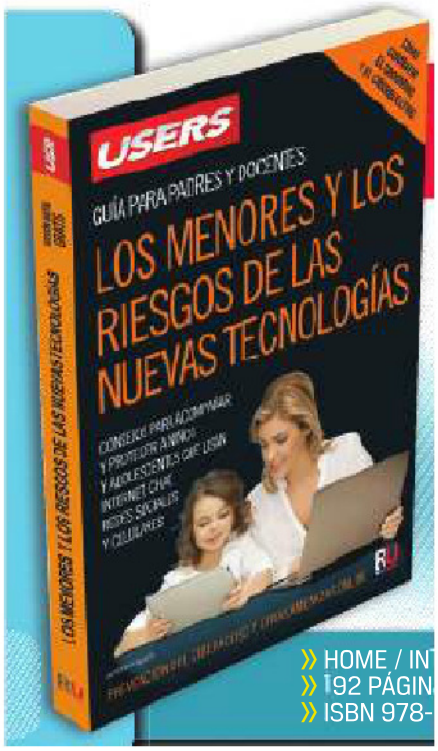
» REDES / EMPRESAS
» 320 PÁGINAS
» ISBN 978-987-1949-34-2

MEJORE EL RENDIMIENTO SIN REDISEÑAR LA ARQUITECTURA

» DESARROLLO / INTERNET
» 320 PÁGINAS
» ISBN 978-987-1949-20-5



» MICROSOF
» 320 PÁGIN
» ISBN 978-



» HOME / IN
» 192 PÁGIN
» ISBN 978-

LLEGAMOS A TODO EL MUNDO VÍA OCA* Y DHL**

MÁS INFORMACIÓN / CONTÁCTENOS

usershop.redusers.com +54 (011) 4110-8700 usershop@redusers.com

*SÓLO VÁLIDO EN LA REPÚBLICA ARGENTINA // **VÁLIDO EN TODO EL MUNDO EXCEPTO ARGENTINA

CRIPTOGRAFÍA

DESDE LOS SISTEMAS CLÁSICOS HASTA EL FUTURO DE LA PRIVACIDAD

por Federico Pacheco

RedUSERS

TÍTULO: Criptografía
AUTOR: Federico Pacheco
COLECCIÓN: Manuales USERS
FORMATO: 24 x 17 cm
PÁGINAS: 208

ISBN 978-987-1949-35-9

Pacheco, Federico
Criptografía. - 1a ed. - Ciudad Autónoma de Buenos Aires: Fox Andina; Buenos Aires: Dalaga, 2014.
208 p.; 24 x 17 cm - (Seriada; 10)

ISBN 978-987-1949-35-9

1. Informática. I. Título
CDD 005.3

VISITE NUESTRA WEB

EN NUESTRO SITIO PODRÁ ACCEDER A UNA PREVIEW DIGITAL DE CADA LIBRO Y TAMBIÉN OBTENER, DE MANERA GRATUITA, UN CAPÍTULO EN VERSIÓN PDF, EL SUMARIO COMPLETO E IMÁGENES AMPLIADAS DE TAPA Y CONTRATAPA.

RedUSERS
COMUNIDAD DE TECNOLOGÍA

redusers.com

Nuestros libros incluyen guías visuales, explicaciones paso a paso, recuadros complementarios, ejercicios y todos los elementos necesarios para asegurar un aprendizaje exitoso.

LLEGAMOS A TODO EL MUNDO VÍA OCA* Y DHL**

* SÓLO VÁLIDO EN LA REPÚBLICA ARGENTINA // ** VÁLIDO EN TODO EL MUNDO EXCEPTO ARGENTINA

usershop.redusers.com usershop@redusers.com + 54 (011) 4110-8700

Federico Pacheco

Especialista en Seguridad de la Información, certificado ACSC (*Advanced Computer Security Certificate*) por la Universidad de Stanford. Se ha orientado profesionalmente a la consultoría, investigación y educación, y ha prestado servicios a empresas y gobiernos.

Cuenta con más de diez años de experiencia docente en distintos niveles de enseñanza y forma parte de las cátedras Seguridad Informática y Criptografía, de la Universidad Tecnológica Nacional (UTN). Participa periódicamente como expositor en conferencias locales e internacionales relacionadas con Seguridad y Educación. Actualmente es vicepresidente de ISSA Argentina y colaborador en varias ONGs. Ha publicado artículos en importantes revistas y sitios web especializados, y ha generado material educativo para diferentes instituciones y para la comunidad en general.

Agradecimientos

A mi editora, Paula Budris, por su paciencia y dedicación.

A Jorge Ramió Aguirre, por compartir su enorme experiencia.

A Jorge Eterovic, por su confianza.

Dedicatoria

A mi madre, que nos dejó físicamente durante la redacción de este libro.

A mis alumnos, que me permiten cumplir con alegría el rol de educador.

Prólogo I

A partir de las revelaciones del ex técnico de la CIA Edward Snowden sobre el espionaje estadounidense a nivel mundial –bajo la excusa de su "lucha contra el terrorismo"–, se evidencia la preocupante vulnerabilidad de la información de personas, organizaciones y países.

Todos conocemos aquel viejo adagio que expresa que si no podemos evitar que nos espíen, al menos debemos hacer algo para que la información obtenida no les sea de utilidad a aquellos que la han tomado. Y es aquí donde la criptografía juega un papel fundamental.

Esta realidad nos impone la necesidad de hacer algo para desmitificar la criptografía. Afortunadamente, llega a mis manos esta excelente obra de Federico Pacheco donde de manera clara, sencilla y con un hilo conductor lógico nos va llevando desde los conceptos básicos en los que se apoyan los desarrollos criptográficos hasta los distintos métodos actualmente en uso.

En particular, el capítulo dedicado a la Infraestructura de Clave Pública (PKI, por sus siglas en inglés) hace un aporte fundamental a la comprensión de su importancia, estándares y aplicaciones. Hoy, a casi diez años de la promulgación de la Ley de Firma Digital en Argentina, tenemos escasas implementaciones funcionando, una materia pendiente tanto desde el ámbito gubernamental como de las organizaciones privadas.

Considero, sin temor a equivocarme, que este libro puede servir tanto a un lector sin demasiados conocimientos teóricos, que quiera entender en qué consiste la criptografía, como también a los docentes que enseñamos en distintos estamentos de la educación, como material de base para desarrollar nuestras materias.

Si este esfuerzo de Federico contribuye a que la criptografía no sea solo una herramienta para expertos y a que su uso se generalice sin falsos prejuicios, habrá logrado su principal objetivo: el de desmitificar una ciencia que durante siglos se mantuvo en el oscurantismo.

Profesor, MBA Jorge Eterovic
Director de la carrera de Ingeniería en Informática,
Universidad de Morón, Argentina

Prólogo II

Para alguien dedicado durante veinte años a la enseñanza universitaria de la criptografía ha sido un verdadero placer haber leído un libro como el publicado por mi buen amigo y colega Federico Pacheco, un incansable entusiasta de la seguridad que cuenta, además, ya con varias publicaciones.

Y este placer se acrecienta, si cabe, al comprobar que Federico ha tomado entre sus referencias algunas de mis publicaciones, ampliando y actualizando la información que allí se entregaba y, a la vez, resumiendo inteligentemente y en pocas líneas los conceptos más importantes de esta apasionante temática.

El libro hace un barrido temporal que contempla desde la criptografía clásica hasta los algoritmos y protocolos modernos que usamos en nuestra vida cotidiana (muchas veces sin darnos cuenta de ello), en un formato amigable y de grata lectura, minimizando –a mi entender de forma muy acertada– los aspectos matemáticos.

Se agradecen las referencias y los apartados “Curiosidades e ideas” y “Datos útiles” que, a modo de pequeñas píldoras, nos dan una visión esquemática de los temas que se van tratando en el libro.

En resumidas cuentas, el libro que tiene en sus manos es un importante aporte de Federico Pacheco a la cultura y a la difusión de la seguridad, en la parcela que le corresponde a la criptografía, ciencia a la que tanto interés viene prestando la humanidad desde tiempos remotos. Y es que la criptografía sigue siendo en la actualidad la única herramienta para asegurar al menos dos de los tres principios básicos de la seguridad de la información: la confidencialidad y la integridad, entendida esta última como integridad del mensaje y, en su caso, autenticidad del emisor.

Estimado lector, en la misma línea de las buenas sensaciones encontradas por quien firma estas letras, mientras leía el libro aprovechando una visita realizada a la hermosa ciudad de Buenos Aires, le deseo una muy feliz lectura.

Dr. Jorge Ramió Aguirre
Profesor de la Universidad Politécnica de Madrid, España
Director de Criptored, Intypedia y Crypt4you

RedUSERS
COMUNIDAD DE TECNOLOGÍA

La red de productos sobre tecnología más importante del mundo de habla hispana

Libros
Desarrollos temáticos en profundidad

Coleccionables
Cursos intensivos con gran despliegue visual

Revistas
Las últimas tecnologías explicadas por expertos

RedUSERS
redusers.com
Noticias actualizadas minuto a minuto, reviews, entrevistas y trucos

Newsletters
Regístrese en redusers.com para recibir un resumen con las últimas noticias

RedUSERS PREMIUM
premium.redusers.com
Nuestros productos en versión digital, con contenido adicional y a precios increíbles

Usershop
usershop.redusers.com
Revistas, libros y fascículos a un clic de distancia y con entregas a todo el mundo

El libro de un vistazo

Esta obra abarca los distintos aspectos de la criptografía en su relación con la seguridad de la información. El tratamiento de los temas incluye desde los más técnicos hasta los menos técnicos, de forma tal que pueda ser aprovechada tanto por aquellas personas que ya cuentan con conocimientos previos como por quienes estén dando sus primeros pasos en el tema.

*01 CONCEPTOS FUNDAMENTALES

Veremos los fundamentos de la criptografía, los criptosistemas y sus clasificaciones, partiendo de las definiciones más elementales y avanzando a través de su evolución en función de la escritura. También veremos algunas técnicas básicas conceptuales que permiten realizar operaciones de cifrado.

*02 CRIPTOGRAFÍA CLÁSICA

En este capítulo veremos el componente histórico de la criptografía, abarcando el período que se considera clásico y contemplando su evolución a través de los distintos métodos y dispositivos. Estudiaremos algunos de los sistemas antiguos más destacados y también el cambio radical que supuso la tecnología electromecánica aplicada al cifrado.

*03 CRIPTOGRAFÍA MODERNA

Comenzaremos a tratar los temas referentes a la era moderna de la criptografía, donde los sistemas clásicos solo quedan en lo anecdótico, y analizaremos los temas principales en los que se basa la criptografía actual. Por tratarse de tópicos relacionados con la matemática, una base de conocimiento de aritmética y álgebra puede ser de gran ayuda.

*04 CIFRADO SIMÉTRICO

En este capítulo veremos el tipo de cifrado llamado simétrico y una clasificación que permite distinguir los algoritmos en función de la cantidad de elementos que procesan. También analizaremos problemas propios de la naturaleza de este tipo de cifrado y algunos de los algoritmos más conocidos y utilizados. Finalmente, veremos los fundamentos del criptoanálisis relacionados con estos últimos.

*05 CIFRADO ASIMÉTRICO

Aquí veremos los algoritmos de tipo asimétrico y analizaremos las características que los distinguen de los simétricos. También nos introduciremos en los problemas matemáticos en los que se basan y veremos algunos de los algoritmos asimétricos más importantes. Finalmente, presentaremos las bases de la criptografía de curva elíptica.

*06 FUNCIONES HASH

Conoceremos las funciones especiales hash, usadas en criptografía para obtener integridad y autenticidad. Veremos los algoritmos más usados, las técnicas para atacarlos y el uso de estas funciones como códigos de autenticación.

*07 PUBLIC KEY INFRASTRUCTURE (PKI)

Veremos los elementos que forman la infraestructura de clave pública, que sirve de base para ofrecer garantías de confidencialidad, integridad y autenticidad. Analizaremos los certificados digitales y la aplicación de esquemas criptográficos para obtener una firma digital.

*08 PROTOCOLOS Y SISTEMAS DE AUTENTICACIÓN

Veremos el uso de protocolos criptográficos y sistemas de autenticación para garantizar los principios de la seguridad en las comunicaciones de red. Conoceremos también las bases del control de accesos, la aplicación de sistemas para gestionar la autenticación y el manejo de sesiones, y algunos de los protocolos más utilizados.

*ApA CRIPTOGRAFÍA CUÁNTICA ON WEB

En este apéndice iremos un paso más allá de la criptografía convencional por medio de una propuesta que combina técnicas de la física moderna con el cifrado de datos, orientado a la comunicación segura.

*ApB ESTEGANOGRAFÍA ON WEB

Veremos otro campo de aplicación de las técnicas criptográficas, que se orienta más al ocultamiento de información ante observadores ocasionales que a su secreto.

INFORMACIÓN COMPLEMENTARIA

A lo largo de este manual, podrá encontrar una serie de recuadros que le brindarán información complementaria: curiosidades, trucos, ideas y consejos sobre los temas tratados. Para que pueda distinguirlos en forma más sencilla, cada recuadro está identificado con diferentes iconos:


CURIOSIDADES E IDEAS


ATENCIÓN


DATOS ÚTILES Y NOVEDADES

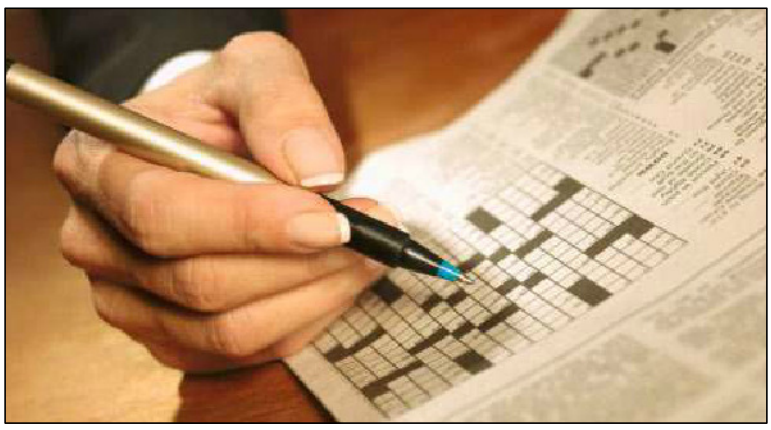

SITIOS WEB

Contenido

*01 Conceptos fundamentales

*02 Criptografía clásica

*03 Criptografía moderna



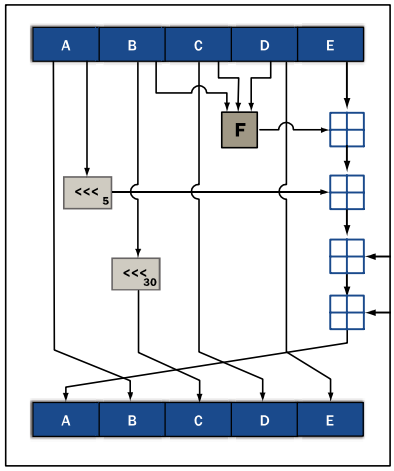
*04 Cifrado simétrico

*05 Cifrado asimétrico

*06 Funciones hash

*07 Public Key Infrastructure (PKI)

*08 Protocolos y sistemas de autenticación



*ApA ON WEB Criptografía cuántica

*ApB ON WEB Esteganografía

Introducción

Hay temas que suelen recibir el asombro de la gente solo por el hecho de ser mencionados. Al hablar, por ejemplo, de biología molecular, mecánica cuántica, biotecnología, epigenética y otras tantas disciplinas, pareciera que quien las estudia cuenta con una capacidad especial, o que debe entender cosas que pocos logran comprender.

La criptografía suele formar parte de ese conjunto de temas, quizás porque el cine y la literatura nos acostumbraron a encontrar cierta magia en los enigmas y misterios donde la información, muchas veces secreta, es la clave para entender algo.

La idea de esta obra es echar luz de una manera simple sobre las técnicas vinculadas a la criptografía, desde sus inicios y usos históricos hasta su forma y aplicaciones actuales. No se propone una profundización demasiado matemática ni la explicación de la complejidad de fenómenos muy abstractos, sino un sobrevuelo por los temas más importantes que permiten comprender las aplicaciones prácticas actuales de la criptografía, sin dejar de lado los detalles técnicos que la explican y señalando la puerta de acceso para quienes quieran adentrarse en los detalles de cada uno de los tópicos.

Por otra parte, resulta importante aclarar que prácticamente todos los temas relacionados con la criptografía que se pueden encontrar en la mayoría de los cursos de seguridad de la información –tanto en su aspecto técnico como de gestión, y tanto en sus facetas ofensivas como defensivas– se desarrollan en este libro, por lo que es de esperar que pueda resultar una bibliografía de consulta útil para el estudio de esos temas en los correspondientes programas de formación académica y profesional.

*01

Conceptos fundamentales

En este primer capítulo nos introduciremos en el mundo de la criptografía a partir de sus definiciones más elementales, que son las que luego utilizaremos para aplicar conceptualmente en los temas de mayor complejidad. También veremos un enfoque histórico de la temática y su evolución a lo largo de los siglos.

Una necesidad básica

Desde los inicios de las sociedades, la comunicación entre pares, tanto oral como escrita, fue un proceso indispensable. El hombre, organizado en grupos y comunidades para poder mejorar sus probabilidades de supervivencia, comprendió desde siempre que la **información** podía derivar en **conocimiento** y éste, en **poder**.

Figura 1. Desde los métodos antiguos hasta los sistemas modernos, el hombre siempre buscó la privacidad de sus comunicaciones.

Una característica intrínseca del ser humano ha sido la existencia de conflictos entre personas, a veces con origen en cuestiones territoriales o de recursos naturales, o por motivos religiosos o ideológicos. Esto derivó en el uso de la fuerza y la existencia de ataques, violencia y guerras, donde el más poderoso resultaba ser el vencedor. En muchos casos, la victoria dependía de un motivo muy evidente: con cuánta información contaba cada uno.

Si durante las guerras se debía enviar información confidencial a las tropas o a otras ciudades, para que pudiera llegar a veces se debían

cruzar territorios de los que no se tenía la certeza de que fueran neutrales o estuvieran controlados por el enemigo. Así, surgió la necesidad de que la información pudiera ser enviada de forma tal que en caso de que alguien no deseado la interceptara, no pudiera entenderla.

> LOS AVANCES EN LA CIENCIA SUELEN ESTAR MOTIVADOS POR NECESIDADES Y OBJETIVOS MILITARES

Las primeras técnicas que podemos considerar criptográficas o de ocultación fueron creadas en ese contexto para permitir el envío de información de manera segura entre partes. Además, en cada época la tecnología determinaba en gran medida el potencial de ataque, por lo que un mayor avance tecnológico implicaba mayor capacidad bélica y por ende, superioridad de poder. A raíz de esto y en términos generales, podemos decir que la mayoría de los avances de la ciencia y la técnica que se dieron en las distintas etapas de la evolución social estuvieron motivados por objetivos militares, para luego trasladarse al ámbito civil (quien conocía el forjado del hierro podía construir mejores espadas, quien conocía la pólvora podía crear explosivos, etcétera).

El desarrollo y los avances en criptografía no escapan a dicha aparente necesidad, pues por una u otra razón siempre han existido motivos para ocultar información de los ojos de terceros.

Figura 2. Las guerras fueron el principal factor que promovió los avances criptográficos en la historia debido a la necesidad de comunicarse secretamente.

Criptodefiniciones

Pese al acuerdo general en cuanto a lo que hoy representa conceptualmente en las ciencias de la computación, se ha tardado mucho en llegar a una definición completa e integral de la criptografía.

En un primer intento de acercarnos al concepto, definiremos a la **criptografía** como un **conjunto de técnicas** basadas en la matemática y aplicadas por medio de la informática que utilizan distintos métodos con el objetivo de **ocultar datos** ante observadores no autorizados, mediante el uso de un algoritmo y al menos una clave.

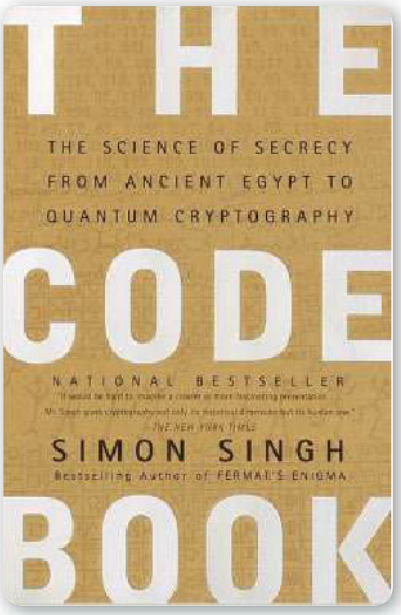


Figura 3. The Code Book (1999), de **Simon Singh**, es una famosa obra sobre la historia de la criptografía que abarca desde el antiguo Egipto hasta los años 90.

La criptografía actual permite, principalmente, proteger la información contra accesos no autorizados, lo que garantiza su **confidencialidad** (secreto) a la vez que provee mecanismos para asegurar la **autenticidad**, la **integridad** y el **no repudio** (una propiedad que evita que pueda negarse la responsabilidad sobre una acción tomada). Su aplicación principal se da tanto en las redes

CRIPTOGRAFÍA EN EL DICCIONARIO

El diccionario de la Real Academia Española propone una definición simple de la palabra. Según la **RAE**, proviene de la conjunción de los vocablos griegos **kryptos** (oculto) y **graphein** (escritura); y la define como el arte de escribir con clave secreta o de un modo enigmático. Esta definición no incluye aspectos modernos y solo la tomamos a los fines de cultura general.

informáticas como en los datos almacenados en medios fijos y extraíbles. Al aparecer la criptografía, es natural que surja como contrapartida la necesidad de analizar la información protegida para determinar si será posible recuperarla aunque no se conozca el sistema utilizado para ocultarla, o bien para obtener la clave. Así nace el **criptoanálisis**, que definiremos como el estudio de los sistemas criptográficos con el objetivo de **descubrir las debilidades** que en ellos pudieran encontrarse, a fin de romper su seguridad sin que sea necesario conocer la clave o secreto utilizado. A quienes trabajan en estas tareas se los llama **criptoanalistas**.

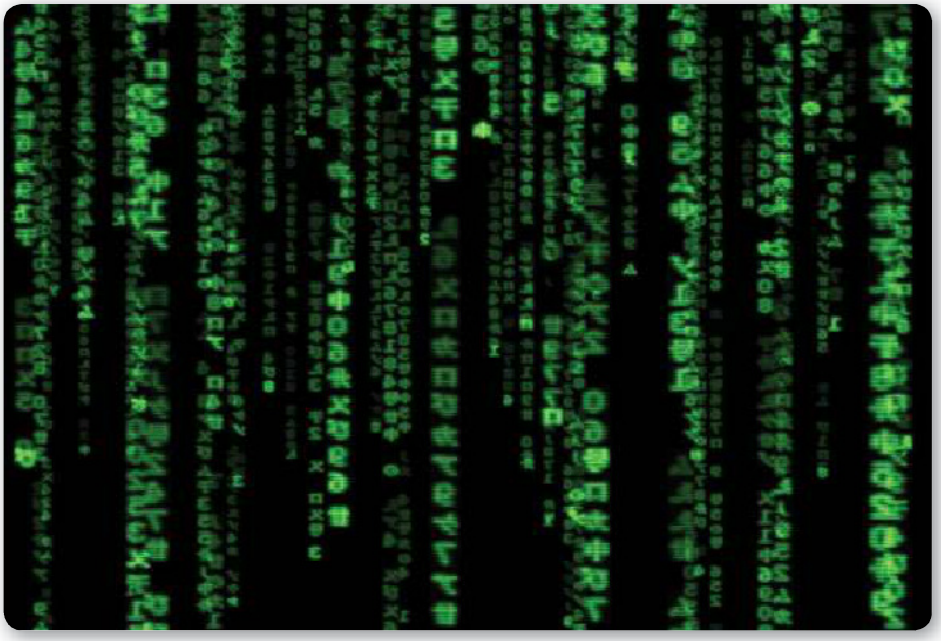


Figura 4. Llamamos **código** a un método estático para transformar un mensaje y hacerlo ininteligible. Opera con un libro de códigos y trabaja a nivel semántico (palabras o frases), a diferencia del **cifrado**, que es dinámico y usa letras o bits.

Las técnicas del criptoanálisis han variado mucho a través de los años. En un principio, se basaban en estudiar y deducir el método de ocultamiento y transformación de la información, en tanto que en la actualidad se basan en el uso de conocimientos y herramientas matemáticas, donde el método de cifrado suele ser conocido mientras que la clave es lo único desconocido del sistema. En ambos casos, la regla general dice que a mayor cantidad de texto cifrado disponible, hay mayor capacidad para abordar el análisis (en sistemas clásicos).

Figura 5. La primera explicación conocida del criptoanálisis se debe a la obra **Manuscrito para Descifrar Mensajes Criptográficos** de **Al-Kindi**, un sabio árabe del siglo IX.

El conjunto de criptografía y criptoanálisis conforma la disciplina científica que denominamos **criptología**. Se suele incluir además dentro de la criptología a la **esteganografía**, que implica el ocultamiento de información de manera que pase desapercibida en su medio habitual, y también su complemento, el **estegoanálisis**, que apunta a estudiar dichas técnicas.

Figura 6. Actualmente, la criptografía es considerada un asunto de seguridad nacional. La **NSA** (**National Security Agency**) es quien la regula en los Estados Unidos.

Clasificaciones

No existe una única manera de clasificar a la criptografía y sus técnicas relacionadas, debido a los múltiples aspectos a describir. No obstante, es posible realizar divisiones según la época histórica en la que nacieron, el tipo de algoritmo y la forma en la que procesan la información.

Según su época

Si tomamos en cuenta la línea de tiempo histórica, podemos realizar una distinción de dos etapas en la criptografía, entre las cuales se produjo un cambio significativo en la manera de concebirla, motivado por los avances tecnológicos. Esto se observa a mediados del siglo XX, considerando que antes de este punto existía una **criptografía clásica** y a partir de ahí comenzó la era de la **criptografía moderna**. Esta división no ha sido solamente cuestión de tiempo sino que fue promovida por los hechos, pues en criptografía moderna se dejaron de cifrar caracteres de un alfabeto para comenzar a cifrar datos codificados en sistema binario. A su vez, el cifrado clásico ha tenido distintas etapas, que pasan por los métodos antiguos, los medievales y los más sofisticados en el siglo XX.

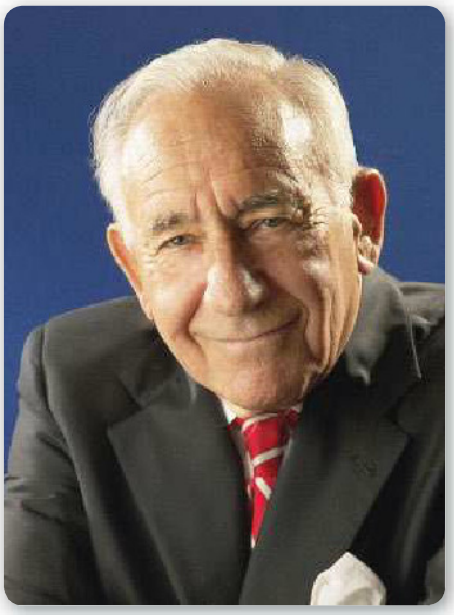


Figura 7. David Kahn es el más grande historiador especializado en criptografía. Cuenta con más de una decena de obras en su haber, entre las que se encuentra **The Codebreakers**.

CRIPTOANÁLISIS EN EL DICCIONARIO

Así como con **criptografía**, el diccionario de la Real Academia Española propone una definición también simple de **criptoanálisis**, como proveniente de la conjunción de dos vocablos griegos: **kryptos** (oculto) y **analýein** (desatar); y la define como el arte de descifrar criptogramas. Tal como la definición de criptografía, solo sirve como dato de cultura general.

Según el algoritmo

Si tenemos en cuenta el tipo de algoritmo utilizado para las operaciones de cifrado (solo para sistemas modernos), podemos clasificarlos entre **algoritmos simétricos** (o de **clave secreta**) y **algoritmos asimétricos** (o de **clave pública**). Los simétricos tienen la característica de que utilizan la misma clave para realizar operaciones de cifrado y descifrado, requiriendo una sola clave por cada par de entidades a comunicar. Los asimétricos utilizan, en cambio, una clave para cifrar que es distinta y complementaria a la que utilizan para descifrar. Esto implica que un mensaje cifrado con una clave solo puede ser descifrado con la otra, y viceversa.

EN EL CIFRADO MODERNO SE USAN ALGORITMOS SIMÉTRICOS Y ASIMÉTRICOS

Podríamos considerar adicionalmente un tipo especial de algoritmo, que no utiliza claves de cifrado sino que solo tiene como objetivo transformar la información de su entrada en un conjunto de datos de tamaño fijo: se trata de los denominados **algoritmos unidireccionales**.

Según el procesamiento

La forma de procesar la información es una categoría que, tal como la anterior, solo aplica a los sistemas modernos (y estrictamente hablando, solo a algoritmos simétricos), pudiendo así dividir los algoritmos en aquellos que le dan tratamiento a la información tomando **bloques de datos** y aquellos que lo hacen tomando **bit por bit**, llamados **algoritmos de flujo**. Analizaremos más en detalle ambos en el **Capítulo 5**, dedicado al cifrado simétrico.

ALGORITMO DE CIFRADO

En el contexto de la criptografía, se habla de **algoritmos de cifrado** cuando se hace referencia al mecanismo usado para transformar un mensaje desde un texto plano (legible y comprensible por cualquiera) a un mensaje cifrado, cuyo contenido es incomprensible si se lo obtiene como tal, sin procesarlo.

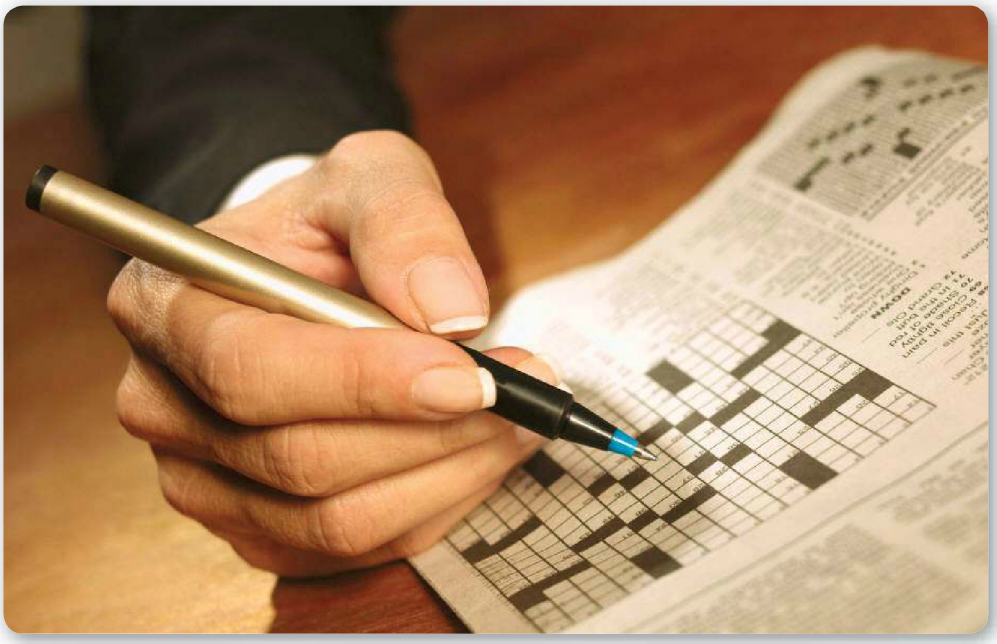


Figura 8. Muchas revistas de pasatiempos incluyen juegos relacionados con la criptografía, donde es necesario deducir mensajes a partir de definiciones o de **criptogramas**.

Criptosistemas

- Un **criptosistema** es el conjunto completo de elementos de un sistema criptográfico, de tal forma que pueda ser utilizado para cumplir con sus funciones. Entre sus componentes se encuentran:

- **Emisor**: quien realiza el proceso de cifrado.
- **Receptor**: quien realiza el proceso de descifrado.
- **Medio**: canal utilizado para intercambiar la información.
- **Algoritmo**: conjunto de transformaciones aplicadas al mensaje para obtener el criptograma, y viceversa.
- **Mensaje**: información que se desea ocultar. También llamado **texto plano** o **texto claro**. El conjunto de todos los posibles mensajes se denomina **espacio de mensajes**.
- **Clave** (o **llave**, o **criptovariable**): pieza de información que, aplicada al algoritmo, permite transformar el mensaje en criptograma y viceversa. Al conjunto de todas las posibles claves se lo llama **espacio de claves**.

- **Criptograma**: mensaje transformado. También llamado **mensaje cifrado**. Al conjunto de todos los posibles criptogramas se lo llama **espacio de criptogramas**.
- **Protocolo**: conjunto de reglas que permiten intercambiar información entre entidades.

Proceso de gestión de claves: método para administrar el conjunto completo de claves de los miembros del sistema.

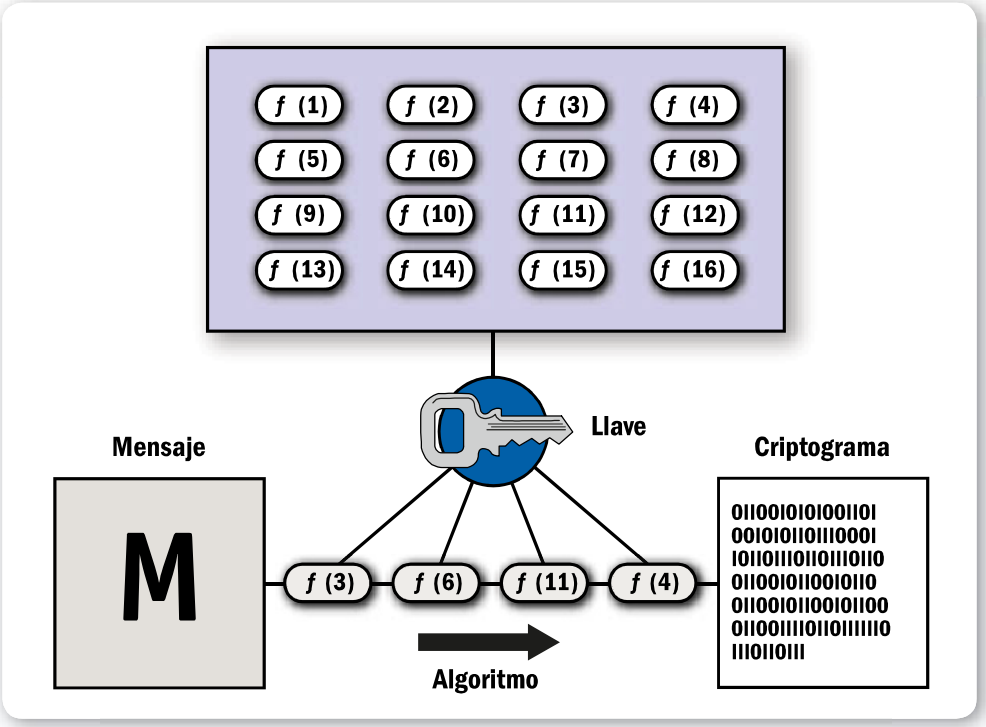


Figura 9. En un **criptosistema**, el mensaje es transformado por medio de un algoritmo y la aplicación de una o más claves o llaves, y se convierte en un criptograma.

Escritura y criptografía

A través de los siglos, la historia de la criptografía ha estado vinculada a la de la escritura. La transmisión de información por escrito se remonta al siglo 34 a.C. con los antiguos sumerios, quienes desarrollaron la escritura cuneiforme (forma de cuña), que ha sido reconocida como una de las más antiguas formas de expresión escrita.

Especialistas sostienen que en los últimos años antes de nuestra era, algunos escribas mesopotámicos escribían sus nombres en códigos numéricos, aunque no en sentido criptográfico (para ocultarlos) sino de manera lúdica. El conocimiento de esta escritura estuvo perdido hasta 1835, cuando el oficial británico Henry Rawlinson encontró unas piedras de acantilado talladas en Behistún (Persia) que poseían textos escritos en las tres lenguas del imperio, a partir de lo cual se logró deducir su alfabeto silábico y descifrarlo. Para 1851, Rawlinson y Hincks (un asiriólogo irlandés) habían logrado interpretar doscientos signos babilonios. Todo un trabajo de criptoanálisis con matices artísticos.

Figura 10. Las tablas de arcilla con inscripciones cuneiformes permanecieron sin poder ser leídas hasta fines del siglo XIX.

¿SECRETO O PRIVADO?

Llamamos **secreto** a una información compartida entre un grupo de entidades o personas, no conocida por nadie ajeno al grupo. El concepto se aplica en criptosistemas simétricos. No debe confundirse **secreto** con **privado**, ya que el último se refiere solo a la esfera personal, en tanto que el primero es compartido por definición.

Casi en simultáneo con el nacimiento de la escritura cuneiforme, los egipcios desarrollaron los jeroglíficos, un sistema de escritura caracterizado por el uso de signos. Quizás, la primera variación de la escritura haya sido en una ciudad al borde del río Nilo llamada Menet Khufu, donde un maestro escriba del siglo 20 a.C. talló unos jeroglíficos que contaban la historia de su señor. El mensaje no era secreto (se leía públicamente sobre las rocas) ni estaba escrito de forma codificada o por símbolos cambiados, sino que le proveía a su texto un aire de dignidad y autoridad. En su tratado **The Codebreakers**, David Kahn relaciona este comportamiento con la diferencia entre escribir "Año 1800" y escribir "En el año mil ochocientos de la era de Nuestro Señor", asignándole al escriba una demostración de elocuente conocimiento duradero para la posteridad, incorporando por primera vez y sin proponérselo, un elemento fundamental de la criptografía: la transformación deliberada de la escritura.

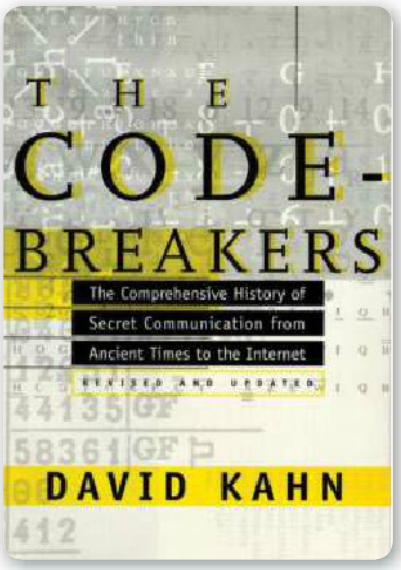


Figura 11. **The Codebreakers** (**1967**) es considerada la obra más completa sobre la historia de la criptografía, al punto que el gobierno de Estados Unidos se resistió originalmente a su publicación.

Al sucumbir la civilización egipcia, el correr de los siglos eliminó el conocimiento sobre el significado de los jeroglíficos, hasta que en el año 1799, el descubrimiento de la Piedra de Rosetta cambió la situación. Esta piedra tenía grabado un decreto en tres sistemas de

LOS FENICIOS Y LA ESCRITURA

Los **fenicios** utilizaron su sistema propio de escritura desde el siglo XII a.C., el cual fue propagado por medio de los mercaderes del mediterráneo y adoptado o adaptado por otras culturas. Se lo considera un alfabeto **abyad**, pues sólo representa **sonidos consonánticos**. Su importancia radica en todos los sistemas de escritura derivados de él.

escritura: jeroglífica, demótica y griega, y fue recién en 1822 que el filólogo francés Jean-François Champollion logró descifrar el texto que estaba escrito, ejerciendo un esfuerzo que podemos considerar un gran trabajo criptoanalítico.

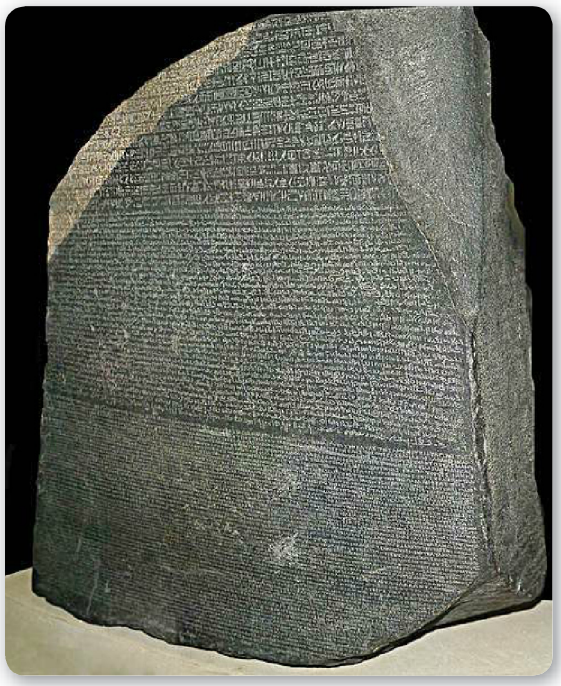


Figura 12. La **Piedra de Rosetta** fue el punto de inflexión para lograr interpretar la escritura jeroglífica egipcia.

Continuando en el mundo antiguo llegamos a China, la única gran civilización de la época en utilizar escritura ideográfica, que curiosamente no parece haber desarrollado tanto como otras la necesidad de cifrado. Solo en el siglo XI apareció explícitamente un escrito donde se recomendaba el uso de un código para el ejército, en la forma de una lista de cuarenta ítems de texto (solicitudes o avisos

AMOR Y SECRETOS

El conocido texto hindú **Kama Sutra**, escrito en el siglo IV a.C. por Vatsiaiana y compuesto por 36 capítulos que versan sobre las artes amatorias, indica que la **escritura secreta** (**mlecchita-vikalpa**) es una de las 64 **habilidades** que deben conocer los amantes. De esta manera, podrán comunicarse y coordinar encuentros sin temor a ser descubiertos.

militares) asignados a los primeros cuarenta ideogramas de un poema, que se debían requerir comunicando el ideograma correspondiente en un lugar especificado de un envío.

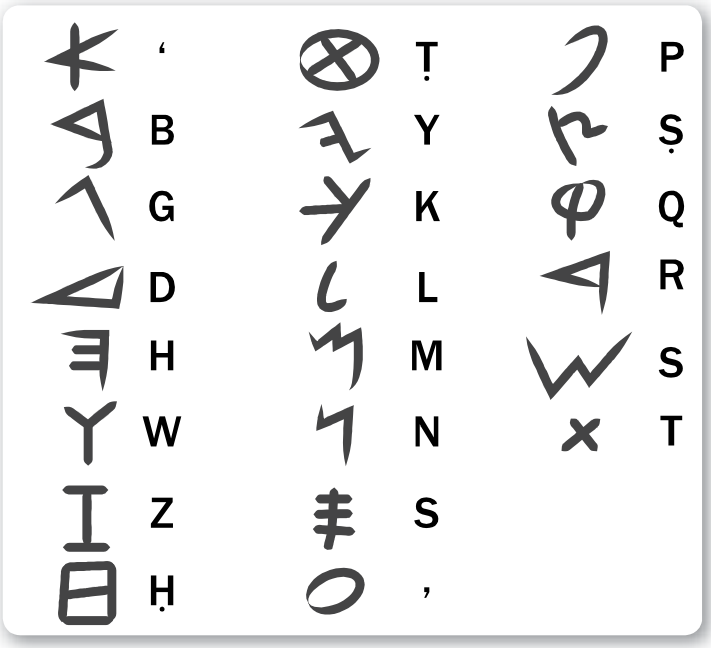


Figura 13. Del alfabeto fenicio derivan otros como el arameo (del cual derivan el árabe y el hebreo) y el griego (base del latín y el cirílico).

En Japón, por su parte, la criptografía no fue utilizada hasta el siglo XIV, y las técnicas más avanzadas recién se conocieron a partir de la apertura del país a occidente a mediados del siglo XIX.

En otro gigante asiático, India, sí se desarrollaron sistemas de comunicación secreta. Uno de ellos, **Artha-sastra**: un tratado sobre gobierno, política económica y estrategia militar, que recomienda a los institutos de espionaje darles las órdenes a sus espías por medio de la escritura secreta. Extrañamente, y según se sabe, fuera de las regiones de Oriente Medio y Europa la criptografía no tuvo un gran desarrollo.

Alfabetos de cifrado

Como hemos mencionado, los sistemas clásicos operan sobre las letras del alfabeto, realizando transformaciones que permiten que cada letra, número o signo termine siendo representado por otro del

mismo alfabeto o de otro, que constituirá el llamado **alfabeto de cifrado**. En su mayoría, estos sistemas utilizan el mismo alfabeto para el mensaje y el texto cifrado. Para realizar transformaciones alfabéticas que puedan representarse con operaciones aritméticas, se puede sencillamente asociar un número a cada letra (A=0, B=1, C=2, etcétera). En términos generales, podríamos tener alfabetos como:

> EN LOS SISTEMAS CLÁSICOS, UNA LETRA ES REPRESENTADA POR OTRA

- **Letras mayúsculas**: 27 caracteres.
- **Letras mayúsculas con números 0-9**: 37 caracteres.
- **Letras mayúsculas y minúsculas**: 54 caracteres.
- **Letras mayúsculas, minúsculas y números**: 64 caracteres.
- **Todos los caracteres imprimibles ASCII**: 94 caracteres.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
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

Figura 14. Establecer una correspondencia numérica en las letras del alfabeto permite realizar operaciones entre ellas para transformar el mensaje en un criptograma.

Excepto en el conjunto de caracteres **ASCII**, en los demás casos no se considera el **espacio** como carácter, y al utilizar ASCII se deben tener en cuenta los caracteres no imprimibles (salto de línea, retorno de carro, etcétera). De hecho, se llama **alfabeto mixto** a aquel que además de contener los elementos del lenguaje incluye símbolos u otros monogramas.

EL ALFABETO

En una lengua (o idioma) se denomina **alfabeto** (o abecedario) al conjunto ordenado de sus letras. También implica los símbolos usados para representarlo por escrito. El término procede de las dos primeras letras griegas: alfa y beta, a la vez derivadas de las fenicias: **alp** (buey) y **bet** (casa). **Abecedario**, por su parte, deriva del nombre de las primeras letras del latín: a, b, c y d.

Estadísticas del lenguaje

Algo que se puede destacar en los distintos idiomas es la **redundancia de letras** que existe en ellos. Es decir, cómo se distribuyen las letras en un párrafo o, más simplemente, cuántas letras hay de cada una posible en un determinado texto. Esto es de gran ayuda en sistemas de cifrado clásicos, donde lo que se cifran son letras individualmente, de tal forma que si encontramos una que se repite en el texto cifrado, podría tratarse de la misma letra en texto claro. Los criptoanalistas se han basado en esto para plantear ataques basados en las estadísticas del lenguaje, según el análisis de las **frecuencias relativas de aparición** de los caracteres en el criptograma. En efecto, incluso si se criptoanalizan sistemas modernos, la metodología indica que el nivel de suposiciones sobre el cifrado debería realizarse desde lo más básico hacia lo más complejo, y no suponer directamente que las técnicas utilizadas son complejas en sí.

Bits b_4 b_3 b_2 b_1	Row	0 (0 0 0)	1 (0 0 1)	2 (0 1 0)	3 (0 1 1)	4 (1 0 0)	5 (1 0 1)	6 (1 1 0)	7 (1 1 1)
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FC	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

Figura 15. Estructura de la tabla **ASCII** de 1968 con dos columnas para caracteres de control, una para caracteres especiales, una para números y cuatro para letras.

Podemos dividir el lenguaje, entonces, según la frecuencia de aparición de sus elementos, quedando al menos tres grupos: alta, media y baja frecuencia. Pueden aparecer diferencias según el texto analizado, pero esto difícilmente se transforma en regla general.

Un texto de un determinado tema puede hacer que una letra aparezca más que otra en relación a la estadística media. Un estudio que se puede tomar como parámetro es el realizado sobre **El Quijote** de Cervantes, de más de un millón y medio de letras, donde el espacio constituye casi un 18% de los caracteres, la letra A el 11% y la E, el 10%.

En la lengua española, algunas estadísticas generales de un texto representativo indican que el 45% son vocales, dentro de las cuales la E y la A son las de mayor frecuencia. Por su parte, las consonantes con más frecuencia son S, R, N, D, L y C (suman 37% entre ellas), y las menos frecuentes son Z, J, Ñ, X, W, K (suman 1,5% entre ellas).

Si el análisis se realiza sobre las palabras del diccionario, la letra con mayor frecuencia resulta ser la A, aunque en el lenguaje escrito la más frecuente sea la E debido a la cantidad de palabras cortas como **que**, **le**, **se** y similares. Además, pueden incluirse los signos de puntuación, el espacio, números, símbolos y otros, lo cual hace variar la estadística.

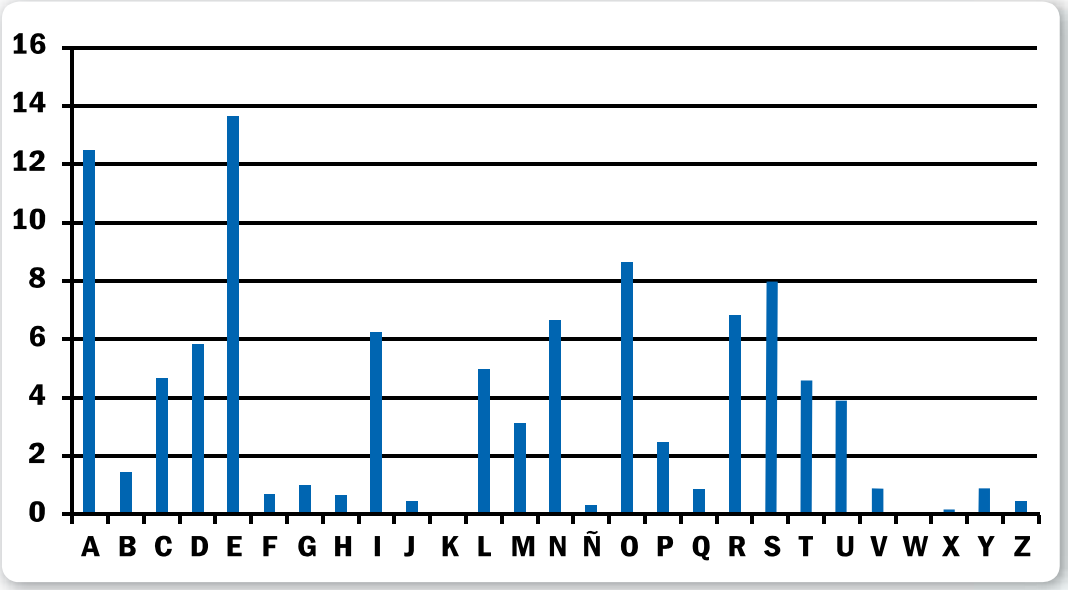


Figura 16. Frecuencia de aparición de las letras del alfabeto español en porcentajes por cada letra.

También se pueden analizar los **digramas** (pares de letras) comunes, así como los **trigramas** o, en general, **poligramas** y palabras de mayor uso. Los digramas de mayor frecuencia relativa en español son DE, ES y EN, en tanto que existen algunos de **frecuencia nula** como QA, KK, ÑL, WZ y otros, que no se utilizan para conformar palabras válidas ni son fin e inicio de dos palabras contiguas.

Sustitución

El método de sustitución se refiere al **reemplazo de caracteres** o **unidades de texto** (pares, tríos, palabras, etcétera) por otros que pueden ser del mismo alfabeto o no. Para descifrarlo, el receptor debe realizar el **reemplazo inverso**; es decir, si se cambió A por B, luego se cambia B por A.

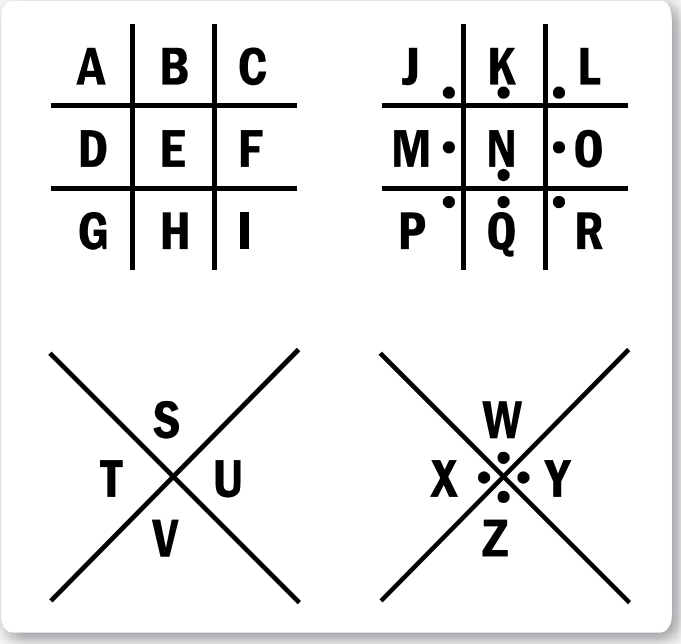


Figura 17. El cifrado **francmasón**, **rosacruz** o **pigpen** es un cifrado por sustitución simple del 1700, basado en el cambio de letras por símbolos según los diagramas.

Monoalfabeto

Un cifrado por sustitución es **monoalfabético** cuando utiliza una **sustitución fija** para todo el mensaje. Por ejemplo, si a la letra A del texto plano se la empareja con la letra H del texto cifrado, se sustituirá siempre de la misma forma (la equivalencia es **unívoca**).

A la vez, en caso de que se esté actuando sobre letras individuales, se lo llama **sustitución simple** o **monográmica**, y en caso de grupos de letras se lo conoce como **polígrámico**, donde se trata el mensaje en bloques de dos o más caracteres sobre los que se aplica la transformación, cambiando **n-gramas** (digramas, trigramas, tetragramas, etcétera) del texto plano por n-gramas del texto cifrado.

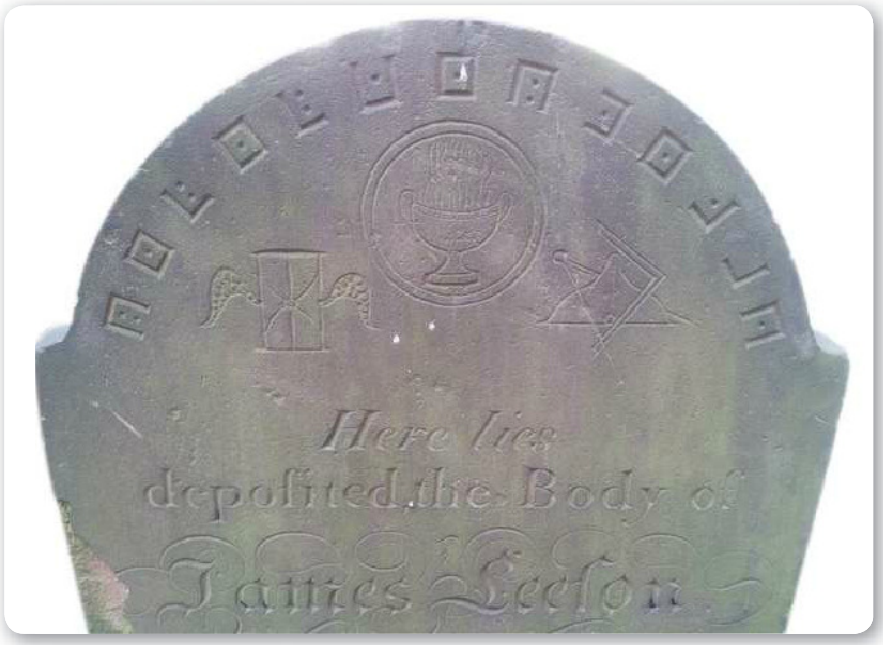


Figura 18. En 1794 se grabó en la lápida de **James Leeson** en el cementerio de Trinity (Nueva York) un mensaje descifrado 100 años más tarde, que decía: **Remember death** (Recuerda la muerte) y usaba el cifrado **francmasón**.

Polialfabeto

En el cifrado **polialfabético**, en cambio, se aplican **diferentes sustituciones** a lo largo de la codificación del mensaje. El secreto en estos casos se basa en conocer el método utilizado para saber cuándo aplicar cada alfabeto. Es decir que las coincidencias entre caracteres van cambiando a medida que se cumplen determinados pasos.

La sustitución polialfabética puede ser a su vez **periódica** (con período igual al tamaño de la clave) o **no periódica** (cuando la clave tiene la misma longitud que el mensaje). Tanto en cifrado

CIFRADO AFÍN

El cifrado de **transformación afín** (o **monoalfabético genérico**) es un cifrado por sustitución donde la cantidad de símbolos del alfabeto del texto plano es la misma que la del alfabeto del texto cifrado. Para encontrar el símbolo del cifrado se usa una función matemática afín en aritmética modular. Es requisito que cada símbolo esté asociado a un orden numérico.

monoalfabético como en polialfabético, la transformación se realiza sobre cada carácter del texto plano uno a uno de manera independiente (por monograma).

De esta manera, podemos tener combinaciones de las distintas características anteriormente explicadas, como por ejemplo las siguientes: **sustitución monográmica monoalfabeto**, **sustitución monográmica polialfabeto** y **sustitución poligrámica**.

Además, existe un caso particular que es el denominado **cifrado por homófonos**, donde un cáracter es reemplazado por otro carácter, de entre varios posibles.

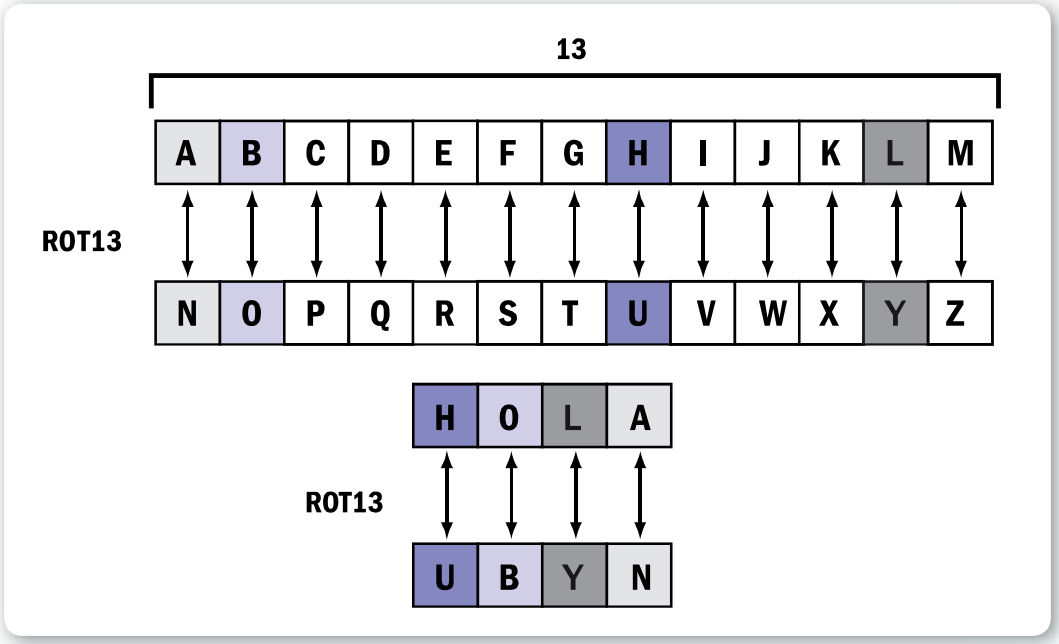


Figura 19. ROT13 (rotar trece posiciones) es un cifrado básico por sustitución sin seguridad de principios de los años 80, donde cada letra corresponde a la que está trece posiciones delante en el abecedario.

EL MÉTODO KASISKI

Es usado para atacar sistemas de sustitución polialfabética, buscando repeticiones de cadenas en el criptograma. Si son de tres o más caracteres, es probable que fueran cifrados con la misma parte de la clave. Las distancias entre cadenas serán múltiplo del largo de la clave, que tiende a ser su máximo común divisor (**MCD**). Luego se divide en **subcriptogramas** cifrados por la misma letra y se aplica un ataque estadístico.

Transposición

El método de **transposición**, por su parte, se basa en el **intercambio de los caracteres**, que son utilizados en un orden diferente. Es decir, no se cambian letras como en el caso de sustitución, sino que solo se cambia su orden de acuerdo a un patrón definido (muchas veces un diseño geométrico o lógico). El resultado es la difuminación de la información del mensaje.

Esto no evita que se detecte la distribución característica del lenguaje, pero sirve para separar digramas y trigramas característicos, lo que obliga al criptoanálisis a usar técnicas de **anagramación** para lograr recuperar esa información. A continuación, veremos los casos de transposición por **grupos**, **series**, **columnas**, **filas**, **adición**, **lógica de Boole**, **conversión de base** y **matricial**.

Transposición por grupos

Una de las técnicas de transposición implica el reordenamiento de los caracteres a partir de una **permutación matemática**: $\mathbf{Pi_{x(y)}}$ donde **x** es la acción tomada sobre los caracteres, e **y** es la **posición** de los caracteres **ordenada por x**. La transposición por grupos es **periódica** con **período p**, luego del cual se comienza a repetir.

Transposición por series

La técnica de transposición por series se refiere al **ordenamiento** del mensaje como una **cadena de submensajes**, de modo que el original se transmita como dicha sucesión en la que cada cadena siga

EL PROFESOR RAMIÓ

Ningún hispanoparlante entusiasta de la criptografía puede haber evitado encontrarse con el material producido por el Dr. Jorge Ramió Aguirre, quien es Profesor Titular de la Universidad Politécnica de Madrid y lleva dos décadas enseñando criptografía en Iberoamérica. Este reconocido profesional, además, ha creado los proyectos **Criptored**, **Intypedia** y **Crypt4you**, y los congresos **CIBSI**, **DISI** y **TIBETS**, y **TASSI**.

una serie determinada. Por ejemplo, una cadena puede corresponder a los múltiplos de 5, otra a los números impares, otra a los números primos, etcétera.

Siendo que no cuenta con un período, es más fuerte que la técnica anterior y su seguridad radica en el secreto de las series que se aplican. Una particularidad se halla en que, para realizar las operaciones, requiere que se recorra el mensaje completo hasta el final, lo que lo hace poco eficiente.

Transposición por columnas

El caso de la transposición por columnas puede ser **simple**, **doble** o **con clave**. En el simple, el mensaje se ordena según una cuadrícula y se rellena si quedan espacios en blanco, para luego transmitir las columnas, lo que corresponderá al texto cifrado. La **cantidad de columnas** (**NC**) es la **clave** del sistema. Para descifrarlo, se reordena el mensaje según lo establecido por el tamaño de la cuadrícula para que quede nuevamente en texto claro.

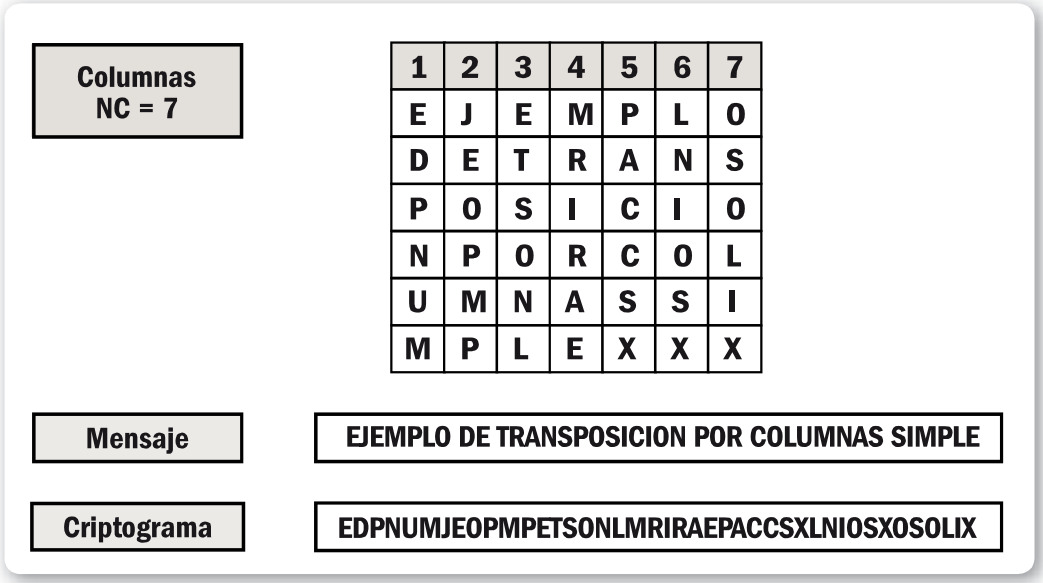


Figura 20. Ejemplo de transposición por columnas con **NC=7**, donde el mensaje es **EJEMPLO DE TRANSPOSICION POR COLUMNA SIMPLE**.

El caso de **columna simple con clave** supone la inclusión de una clave a fin de modificar la posición relativa de las columnas. La clave se ordenará alfabéticamente (numéricamente según su orden alfabético) para

proveer un cambio en las columnas, antes de ser enviado el criptograma. El caso de **transposición doble** busca destruir la adyacencia de las series de letras de corto tamaño de una transposición, por lo que se aplica una nueva permutación al primer resultado.

Transposición por filas

Análogamente al sistema de columnas, podemos operar sobre las filas **escribiendo el mensaje verticalmente** con una cierta **cantidad de filas** (**NF**) que será la **clave**, y el criptograma se obtiene leyendo la grilla horizontalmente. La recuperación del mensaje es trivial, según mencionamos anteriormente para el sistema de columnas. Incluso, podemos hacer más complejo el método utilizando una disposición en forma de **Z** (**zig-zag**).

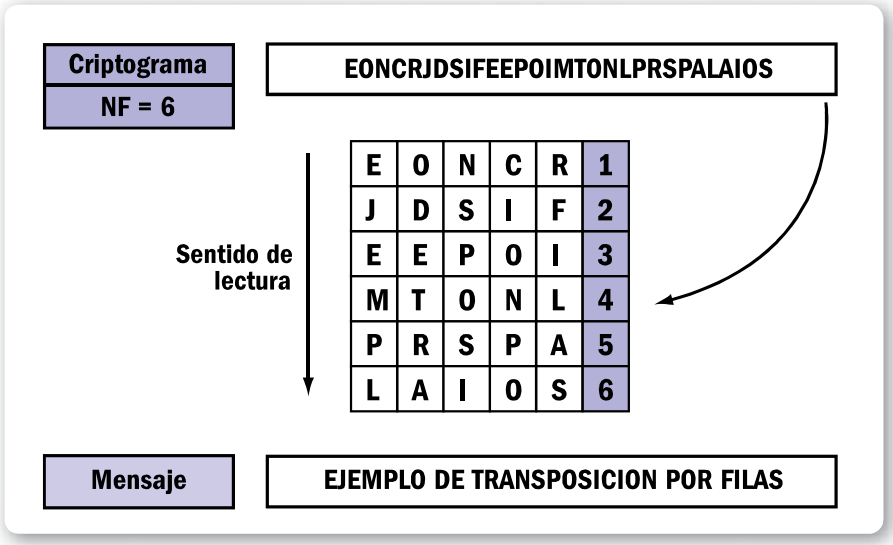


Figura 21. Ejemplo de descifrado de un mensaje cifrado por transposición por filas con **NF=6**.

DISPONIBILIDAD

Definimos **disponibilidad** como la propiedad de la información por la cual es posible garantizar que pueda ser accedida en el momento en que se la requiere. Esta característica, junto a la integridad y la confidencialidad, es uno de los tres pilares básicos de la seguridad de la información, aunque no está garantizada por la criptografía.

Otras transformaciones

La idea de realizar transformaciones para ocultar el mensaje puede ampliarse hacia otras operaciones matemáticas, más allá de la sustitución y la transposición.

De hecho, existen varias transformaciones y algoritmos de cifrado clásicos además de los anteriormente expuestos. Mencionaremos algunos de ellos.

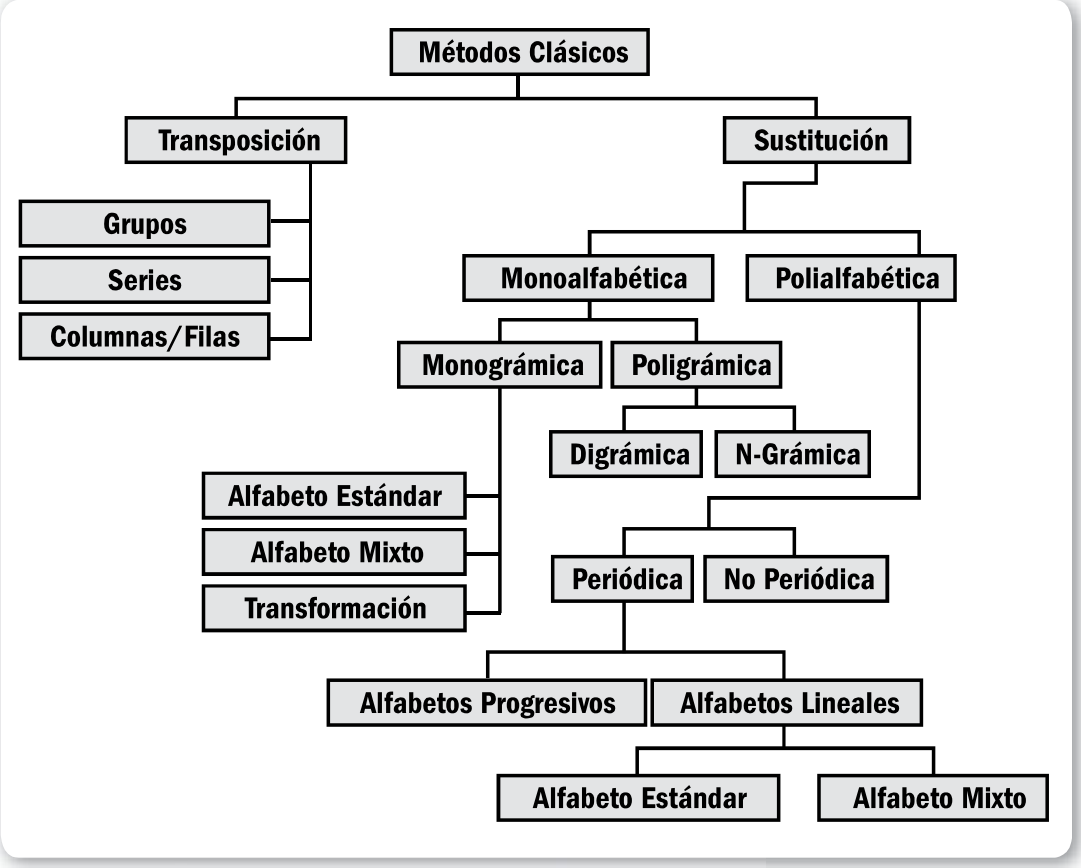


Figura 22. Clasificación de los sistemas de cifrado clásicos.

INTEGRIDAD

Definimos **integridad** como la propiedad de la información por medio de la cual es posible garantizar que se pueda verificar que ésta no ha sido alterada de manera no autorizada respecto a un determinado momento. Esta propiedad, junto a la confidencialidad y la disponibilidad, es uno de los tres pilares básicos de la seguridad de la información.

Transformación por adición

Dado que la **suma** y la **resta** son operaciones con **inversa**, pueden ser utilizadas como funciones de cifrado, de forma similar a la aplicada en los métodos de sustitución. La existencia de una correspondencia entre los caracteres posibilita dos operaciones: **C=EK(M)=M+K** y **C=EK(M)=M-K**.

Para operar, se toma un bloque de **n caracteres** y el carácter cifrado será el valor numérico resultante de la suma o resta de ese número con el que le corresponda a los n caracteres de la clave que coinciden con el bloque.

Extendiendo esta transformación, también podría pensarse en aplicar la operación de **multiplicación**, lo que es posible siempre y cuando se garantice la inversa (divisor distinto de cero y máximo común denominador igual a la unidad).

Para esto, deberíamos reasignar las representaciones alfabéticas y, en lugar de partir de **A=0** (imposición de la aritmética modular, según veremos), deberíamos hacerlo desde **A=1**.

Mensaje	T	R	A	N	S	F	O	R	M	A	C	I	O	N	P	O	R	A	D	I	C	I	O	N
Clave	C	L	A	V	E	C	L	A	V	E	C	L	A	V	E	C	L	A	V	E	C	L	A	V

Valores M	20	18	0	13	19	5	15	18	12	0	2	8	15	13	16	15	18	0	3	8	2	8	15	13
Valores K	2	11	0	22	4	2	11	0	22	4	2	11	0	22	4	2	11	0	22	4	2	11	0	22
C=M+K	22	29	0	35	23	7	26	18	34	4	4	19	15	35	20	17	29	0	25	12	4	19	15	35

C en bloques	222.903.523.726.183.444.191.535.201.729.025.124.195.35

Figura 23. Ejemplo de transformación por adición con bloques de tres caracteres, donde el mensaje es **TRANSFORMACION POR ADICION** y la clave es **CLAVE**.

CONFIDENCIALIDAD

La **confidencialidad** se define como la propiedad de la información que permite garantizar que ésta solo pueda ser conocida por quien esté autorizado. Así, se asegura que ningún usuario no autorizado podrá tener acceso, lo que en criptografía es uno de los objetivos principales. Esta propiedad, junto a la integridad y la disponibilidad, es uno de los tres pilares básicos de la seguridad de la información.

Transformación por conversión de base

Siguiendo en la línea de representación numérica, puede realizarse un **cambio de base** en el mensaje de forma tal que el criptograma corresponda a ese cambio y donde el **secreto** lo conformaría la **base** utilizada. La operación es válida por contar con inversa.

UN MENSAJE REPRESENTADO EN BINARIO PUEDE OPERAR CON UNA CLAVE PASADA A BINARIO

La longitud del mensaje puede resultar mayor o menor que la del criptograma, según el sistema de numeración en relación a la base original. Por ejemplo, puede representarse un mensaje en base decimal, cifrado en bloques de un tamaño fijado, y convertirse cada bloque a base hexadecimal para que el receptor realice la operación inversa al recibirlo.

Transformación por lógica de Boole

Mediante el uso del **álgebra de Boole** es posible cifrar mensajes a través de las operaciones de **negación** (**NOT**) y **OR exclusivo** (**XOR**), que son las que tienen inversas. Si representamos el mensaje en binario, podemos hacerlo operar junto a una clave pasada a binario de igual longitud para obtener el criptograma. Las operaciones serían:

- **Negación**: C=E(M)=**NOT** M y M=**NOT** E(M)
- **XOR**: C=EK(M)=M **XOR** K y M=EK(M) **XOR** K

CRYPT4YOU

Crypt4you (**www.crypt4you.com**) es un aula virtual que ofrece una serie de cursos en modalidad online y gratuitos (**MOOC**, por **Masive Open Online Courses**) sobre diversos temas de criptografía y seguridad de la información. Los cursos incluyen material de texto y videos y están disponibles en idioma español.

Transformación matricial

Esta transformación funciona aplicando operaciones de **producto y suma de matrices**, y teniendo un mensaje **M** transformado mediante un código en una sucesión de unos y ceros que se disponen en una matriz de **I filas** por **I columnas**, al igual que la **clave K**. Las operaciones de cifrado serían entonces: **(C)=(M)+(K)** y **(C)=(M)x(K)**.

Un requisito matemático es que las matrices deben ser de la misma dimensión para que exista inversa en la suma y, para el caso del producto, la matriz clave debe ser **cuadrada**, **no singular** y de **única inversa**. Este método generaliza el cifrado de matriciales propuesto por Lester Hill, que veremos en el **Capítulo 2**.

RESUMEN

El hombre siempre necesitó comunicarse de forma secreta, y la historia de la criptografía encuentra su reflejo en la humana y su evolución técnica. Los criptosistemas son el conjunto de elementos para operar con métodos criptográficos y su clasificación puede ser hecha según distintos criterios, entre los que se encuentra su época, su tipo de algoritmo y su forma de procesar la información. Algunos métodos de cifrado básicos incluyen la sustitución y la transposición de letras de un mensaje con el fin de ocultarlo dentro de un criptograma.

Actividades

TEST DE AUTOEVALUACIÓN

1. ¿Por qué aparece la necesidad de intercambiar mensajes ocultos?
2. ¿Cómo podría definir la criptología?
3. ¿Cuáles son los elementos de un criptosistema?
4. ¿A qué período corresponden la criptografía clásica y la moderna?
5. ¿Cómo se clasifica la criptografía según su tipo de algoritmo?
6. ¿Cuáles son los elementos principales de un criptosistema?
7. ¿Cómo se relaciona el trabajo de un experto en lenguas antiguas y el de un criptoanalista?
8. ¿Qué implica una sustitución polialfabética?

EJERCICIOS PRÁCTICOS

1. Investigue el estudio de la filología y su posible relación con la criptografía.
2. Averigüe las estadísticas de los lenguajes inglés, francés y alemán.
3. Investigue acerca de los problemas que tuvo el libro **The Codebreakers** antes de su primera edición.
4. Investigue sobre la existencia de sistemas de cifrado utilizados fuera de Medio Oriente y Europa en la antigüedad.
5. Identifique otros tipos de transformaciones que pueden realizarse en el cifrado, más allá de las expuestas.

PROFESOR EN LÍNEA

Si tiene alguna consulta técnica relacionada con el contenido, puede contactarse con nuestros expertos: profesor@redusers.com

*02

Criptografía clásica

En este capítulo estudiaremos la parte de la criptografía que consideramos clásica. Abarca desde sus primeros tiempos hasta mediados del siglo XX, y finaliza con la Segunda Guerra Mundial, hecho que promovió grandes avances en materia de cifrado.

Original	א	ב	ג	ד	ה	ו	ז	ח	ט	י	כ	ל	מ	נ	o	ע	פ	צ	ק	ר	ש	ת
Clave	ת	ש	ר	ק	צ	פ	ע	o	נ	מ	ל	כ	י	ט	ת	ז	ו	ה	ד	ג	כ	א

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

Figura 1. Representación del sistema **Atbash** para el alfabeto **hebreo** y su equivalente para el **español**.

La escítala

La **escítala** fue un método de cifrado del siglo V a.C., proveniente de Laconia, una ciudad de la antigua Grecia. Consistía en una cinta de cuero enrollada en un bastón de mando, sobre la que se escribía longitudinalmente un mensaje. Luego, la cinta se desenrollaba y se transportaba por medio de un mensajero. Si un enemigo obtenía la cinta en el camino, no lograba dar con el mensaje al no contar con el bastón del diámetro adecuado para enrollarla, el cual podemos decir que oficiaba de clave.

A fin de lograr un mejor enmascaramiento del mensaje, se agregaban caracteres en todo el contorno de la cinta. La escítala aplica el método de **transposición**, ya que se permutan caracteres y no se modifica el mensaje en su camino hacia el receptor.

Figura 2. La **escítala** se utilizó en la antigua Grecia para el cifrado de mensajes y se basaba en el uso del bastón de mando.

Cifrado de Polybios

Hacia mediados del siglo II a.C., el historiador griego Polybios diseñó un sistema basado en una tabla donde se hacía corresponder a cada letra del alfabeto con un par de letras según su ubicación de fila y columna, por lo que el criptograma era ese conjunto de pares de letras (por ejemplo, la letra **A** quedaría representada por **AA**, la **M** como **CB**, etcétera). Podemos considerar desventajoso el hecho de que la longitud del texto cifrado sea el doble de la del texto sin cifrar.

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

Figura 3. Tabla de cifrar de **Polybios** adaptada al español. Codificaba cada letra como producto de dos letras o números.

Cifrado del César

En el siglo I a.C. apareció el **cifrado del César**, nombrado así en honor al emperador Julio César. Este cifrado aplica al texto un

CIFRADO

En criptografía, llamamos **cifrado** a la **transformación** de un conjunto de datos realizada carácter por carácter (o bit por bit) sin tener en cuenta el significado o la estructura lingüística del mensaje en cuestión, y generalmente mediante el uso de un algoritmo. Tanto éstos como los códigos son métodos usados para **alterar la representación** de un mensaje.

desplazamiento fijo de tres caracteres, a modo de transformación. El alfabeto de cifrado es entonces el mismo que el del texto original, solo que desplazado hacia la derecha.

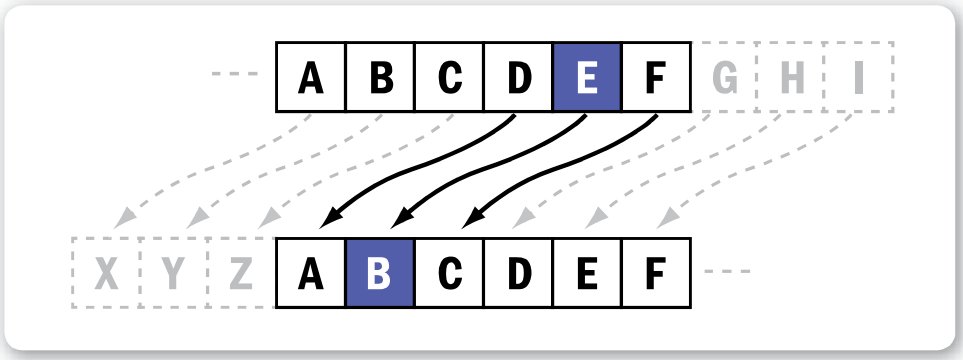


Figura 4. El sistema de cifrado del **César** asignaba un desplazamiento de tres letras en el abecedario para poder construir los mensajes cifrados.

Su principal debilidad es que, conocido el desplazamiento, deja de ser problemático el descifrado. Para resolver esto se puede tomar como clave el desplazamiento y que, en vez de ser 3, sea un valor acordado previamente. Adicionalmente, es posible aumentar aún más la seguridad del sistema mediante el uso de una clave alfabética real (palabra o frase) que será colocada a continuación del desplazamiento, para determinar así las posiciones finales de las letras. Luego de escrita la clave, se completa el resto de las letras en orden para obtener el alfabeto de cifrado. Una última variante podría ser el uso de caracteres o símbolos que no pertenezcan al alfabeto del mensaje en el alfabeto de cifrado.

> EL CIFRADO DEL CÉSAR APLICA UN DESPLAZAMIENTO FIJO DE TRES CARACTERES

CRIPTORED

Criptored, o **Red Temática Iberoamericana de Criptografía y Seguridad de la Información**, es un proyecto nacido en el año 1999 con el objetivo de distribuir contenidos sobre criptografía y ser un punto de encuentro para profesionales, académicos e investigadores de Iberoamérica. La dirección de este sitio web es **www.criptored.upm.es**.

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

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

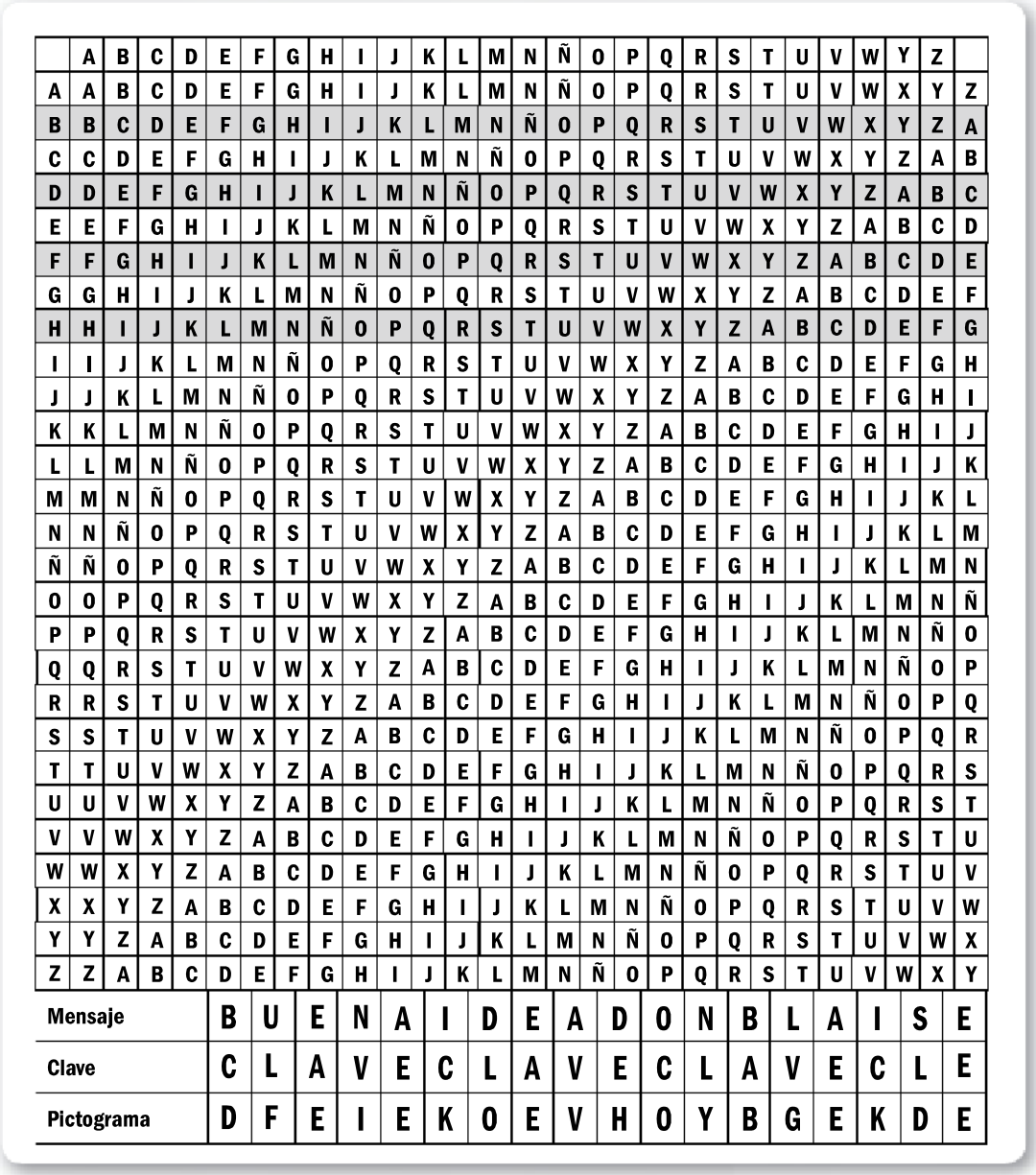
Figura 5. Ejemplos de cifrado del César estándar y con clave **ESTA ES MI CLAVE** y desplazamiento de cuatro. Se ve que los caracteres repetidos de la clave no se vuelven a escribir.

Cifrado de Vigenère

El sistema de Vigenère, inventado por el criptólogo francés Blaise de Vigenère (1523-1596), se basa en el mismo principio que el del César pero con un desplazamiento de caracteres indicado por un número relacionado con un carácter de la clave escrito de manera cíclica debajo del mensaje.

Para obtener el criptograma se utiliza la tabla asociada, o bien el método analítico. Si utilizamos la tabla buscaremos cada letra del mensaje en la primera fila y su correspondiente letra de la clave en la primera columna, y en su intersección se encontrará la letra del criptograma. Para realizar el descifrado, identificamos cada letra de la clave en la primera fila y buscamos en la columna correspondiente la letra del criptograma. Al desplazarnos horizontalmente desde ésta hacia la primera columna, obtenemos la letra que corresponde al texto claro. Curiosamente, algunos afirman que el verdadero creador del método no fue Vigenère sino el criptólogo italiano Giovan Battista Belaso, en el año 1553.

Existieron algunas variantes a este método, como por ejemplo el llamado **autoclave**, que utiliza una clave primaria para comenzar a cifrar y a continuación de ésta se añade el mensaje como resto de la clave. Otra variante fue el método de **Beaufort**, introducido originalmente por Giovanni Sestri en 1710 y popularizado a partir del almirante inglés Sir Francis Beaufort cuando, luego de su muerte en 1857, su hermano publicó sus textos. A diferencia de Vigenère, en este método se restan (analíticamente) la letra original y la clave. En la práctica, esto corresponde a encontrar la letra de la primera columna en lugar de la letra dentro de la tabla.



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

Mensaje	B	U	E	N	A	I	D	E	A	D	O	N	B	L	A	I	S	E	
Clave	C	L	A	V	E	C	L	A	V	E	C	L	A	V	E	C	L	E	
Pictograma	D	F	E	I	E	K	O	E	V	H	O	Y	B	G	E	K	D	E	

Figura 6. La tabla de Vigenère y un ejemplo de mensaje: **BUENA IDEA DON BLAISE**, cifrado con la clave **CLAVE**.

El disco de Alberti

Leon Battista Alberti, un sacerdote católico del siglo XVI, diseñó un disco para cifrar textos que no requería una única correspondencia entre el alfabeto del mensaje y del criptograma. Según su método, dependiendo de una clave, cada letra podía ser cifrada con un carácter diferente. Constaba de un disco exterior con los veinte caracteres del latín (los mismos del español pero sin las

letras H, J, Ñ, K, U, W e Y) más los números 1, 2, 3 y 4 para el uso de códigos especiales, y un disco interior con los caracteres latinos más el signo & y las letras H, K e Y. Con este esquema, pueden definirse hasta veinticuatro sustituciones diferentes (posiciones del disco) mediante la rotación del disco interior.

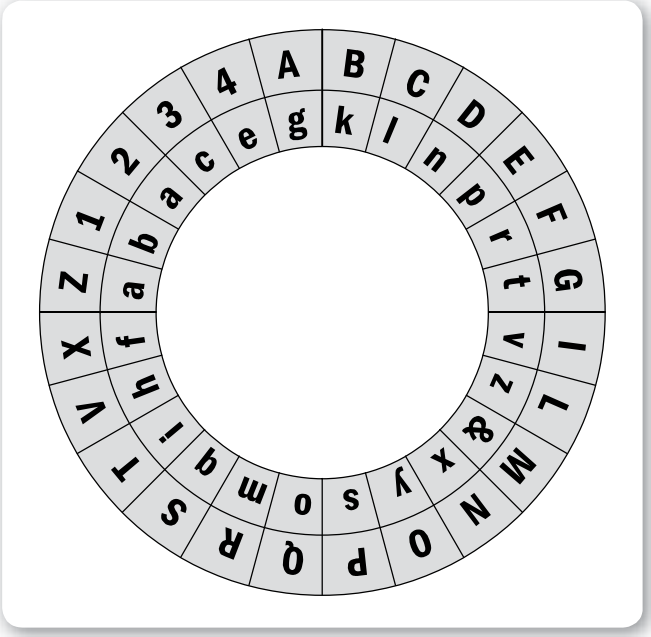


Figura 7. El disco de **Alberti** permitía cambiar el alfabeto de sustitución durante el proceso de cifrado, girando el disco interior cada N caracteres.

El disco de Wheatstone

En 1817, el coronel norteamericano Decius Wadsworth creó un disco de cifrado que no se haría demasiado conocido en ese entonces, pero que años más tarde sería popularizado por el inventor inglés Charles Wheatstone. El sistema, diseñado tiempo antes por Thomas Jefferson,

UN CLÁSICO LIBRO ELECTRÓNICO

Probablemente, el libro electrónico en español sobre criptografía más descargado de la historia sea **Seguridad Informática y Criptografía** del Dr. Jorge Ramió Aguirre, ofrecido desde hace más de una década de manera gratuita para fines educativos o personales. Esta obra puede encontrarse online en el sitio de **Criptored** y se va actualizando a lo largo del tiempo.

se basa en el mismo principio que el disco de Alberti, pero considerando un disco exterior con el alfabeto inglés (26 caracteres) más el espacio en blanco, y un disco interior con los mismos 26 caracteres del inglés sin el espacio, distribuidos de modo aleatorio. Adicionalmente, incluye un par de agujas con engranajes que hacen que al girar el disco externo 27 posiciones, el interno gire 26.

Figura 8. El disco de **Wheatstone** permite cambiar el alfabeto de cifrado por cada vuelta de las agujas, que solo se mueven en sentido horario.

Para cifrar, se debía girar la aguja externa en sentido horario hasta hacer coincidir cada letra del texto con la del disco externo y tomar el carácter correspondiente del círculo interior.

El cambio de relación entre alfabetos se dará en distintas circunstancias, como por ejemplo cuando las letras escritas no están alfabéticamente ordenadas, cuando deben repetirse o cuando debe buscarse el carácter en blanco después de cada palabra. Esto ocasiona que cada letra del propio mensaje influya en la manera de cifrar las siguientes. En la actualidad este concepto se sigue utilizando, en el denominado cifrado moderno.

> EL DISCO DE WHEATSTONE SE BASA EN EL MISMO PRINCIPIO QUE EL DISCO DE ALBERTI

El cilindro de Bazeries

Otro sistema –similar a otro creado de manera independiente por Thomas Jefferson un siglo antes– fue el del criptólogo francés Étienne Bazeries. Este consta de veinte discos con las 26 letras del alfabeto inglés en sus circunferencias, lo que permite establecer una clave sobre la generatriz del cilindro conformado y producir, así, 26 alfabetos distintos.

Para cifrar un mensaje, se divide primero en bloques de veinte letras (una por cilindro) y se colocan longitudinalmente en la línea del visor. El mensaje cifrado puede ser cualquiera de las otras 25 líneas, pudiendo también cambiar la distancia a la generatriz en cada bloque, consiguiéndose así una complejidad de 25! (factorial de 25) alfabetos. Por su parte, el descifrado requiere colocar los caracteres del criptograma en el visor y buscar la generatriz acordada para obtener el mensaje.

Figura 9. El disco de **Jefferson** o de **Bazeries** aplicaba un sistema de ruedas con letras mediante el que se transformaba el mensaje.

EL TALENTOSO THOMAS JEFFERSON

Thomas Jefferson fue el principal autor de la **Declaración de Independencia de los Estados Unidos** de 1776 y el tercer presidente de este país. Pero además, era también un buen inventor, lo que lo llevó a desarrollar en 1795 el **cifrado de rueda**, llamado luego **disco de Jefferson**, en el que se basó el sistema de **Bazeries**.

Cifrador de Playfair

Otra creación del ya mencionado Charles Wheatstone en 1854 para realizar **comunicaciones secretas por telégrafo** es el cifrado de **Playfair**, nombrado por el científico escocés Lord Lyon Playfair, quien impulsó su utilización. En este sistema se toman de a dos letras del mensaje y se transforman en otras dos diferentes, a partir de un conjunto de reglas y una matriz de 5x5 (suficiente para albergar todas las letras del alfabeto inglés y casi todas las del español). Las reglas para transformar dos caracteres del mensaje (**m1** y **m2**) en sus correspondientes del criptograma (**c1** y **c2**) son las siguientes:

> EN UNA MATRIZ DE 5X5 ENTRAN LAS LETRAS DEL ALFABETO INGLÉS Y CASI TODAS LAS DEL ESPAÑOL

- Si **m1** y **m2** están en la misma fila, **c1** y **c2** son las letras de su derecha (si están en un extremo, se toma circularmente).
- Si **m1** y **m2** están en la misma columna, **c1** y **c2** son las letras de abajo (también de manera circular).
- Si **m1** y **m2** están en distintas filas y columnas, **c1** y **c2** corresponden a las letras de la diagonal opuesta (formando un cuadrado entre las cuatro).
- Si **m1** y **m2** son iguales, se inserta un carácter sin significado entre ellos para evitar la repetición y luego se aplican las demás reglas.
- Si la cantidad de letras es impar, se agrega una letra sin significado al final del texto.

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

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

Figura 10. El cifrado de **Playfair** fue usado en la Primera Guerra Mundial por el Reino Unido. Aquí, su tabla completa y el agregado de una clave **CLAVE** para aumentar su complejidad.

Cifrador de Hill

El matemático Lester Hill propuso, en 1929, el uso de matrices para un sistema de cifrado que utiliza **sustitución poligrámica**, el santo grial de su época. Pese a su alta seguridad, en la actualidad ha sucumbido al criptoanálisis.

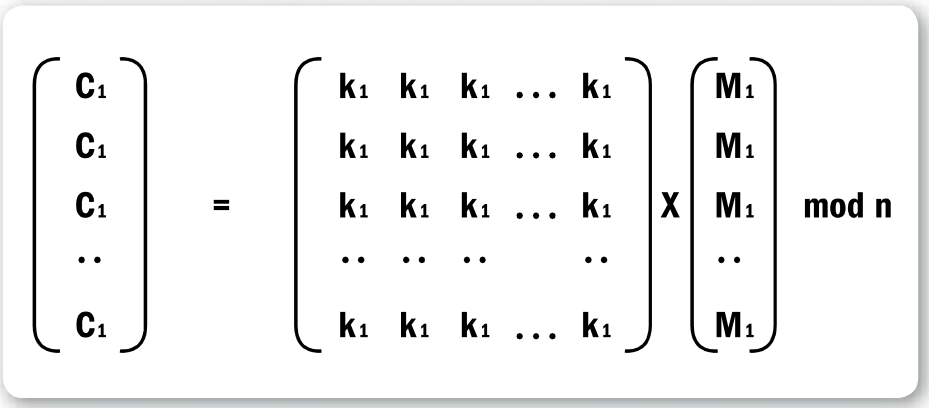


Figura 11. El cifrado de **Hill** usaba transformaciones lineales matriciales en módulo 26. Pese a su alta seguridad, tuvo poco éxito por la dificultad para implementarlo en una máquina, ya que aún no existían las computadoras.

El sistema de Hill implicaba operar matrices, para lo cual debe asegurarse que la **matriz K** (clave) tenga **inversa** ($\mathbf{K^{-1}}$) en el cuerpo **n**. Entonces calculamos $\mathbf{K\text{-}1 = T_{Adj(K)}/|K| \bmod n}$, donde:

- **Adj(K)** es la **matriz adjunta**
- **T** es la **matriz traspuesta**
- **|K|** es el **determinante** (no puede ser cero ni tener factores en común con $_n$)

En caso de que el texto del mensaje no sea múltiplo del bloque N, se lo rellena con caracteres predefinidos.

Si utilizamos este método sobre bloques de ocho caracteres, el espacio de claves es comparable con el de los primeros algoritmos de cifrado modernos. De hecho, si utilizamos un módulo de valor primo, la cantidad de claves se acerca al **máximo** $\mathbf{p^x}$, donde $\mathbf{x = d^2}$ (**d** corresponde al tamaño de **N-grama**). Dada su linealidad, es posible hacer ataques donde se conozca el texto plano, aplicando algún método del álgebra lineal (como el de **Gauss Jordan**) para encontrar la **matriz K**.

Cifrador de Vernam

Gilbert Vernam, un ingeniero de **Bell Labs**, propuso en 1917 un cifrador por **sustitución binaria** que utilizaba el código de los **teletipos** (**Baudot**, de **5 bits**). El sistema aplica la operación **XOR** y una **secuencia aleatoria S** obtenida en base a una **clave** (**K**) que es previamente compartida entre las partes. Por ser la función **XOR** reversible (aplicándose nuevamente), el algoritmo para descifrar es el mismo que el de cifrado. Al poco tiempo, Joseph Mauborgne propuso que la clave tuviera información aleatoria, con lo que el sistema derivó en lo que se denomina **one-time pad** (libreta de un solo uso). En los años 40, Claude Shannon demostró que el cifrado concebido así es irrompible (publicado por **Bell Labs Technical Journal** en 1948) y, de hecho, es el único método conocido que se considera irrompible.

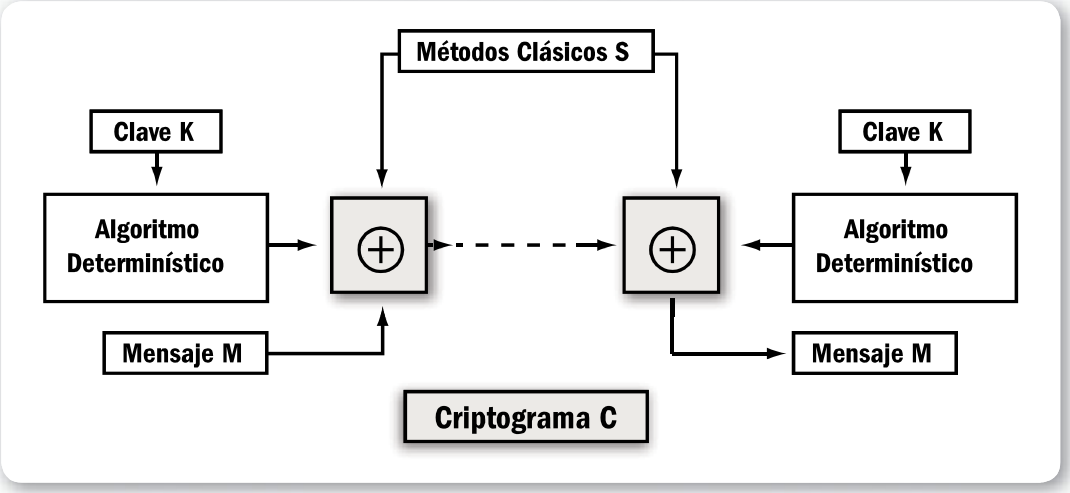


Figura 12. El cifrado de **Vernam** requiere, idealmente, secuencias totalmente aleatorias e intercambio seguro para las claves.

La guerra de las máquinas

Corría la Primera Guerra Mundial, era el año 1917 –el mismo de la Revolución rusa–, cuando el criptoanalista británico William Montgomery interceptó un telegrama en código que el ministro alemán Arthur Zimmermann envió a su embajador en México. El mensaje fue descifrado mediante criptoanálisis y cambió el curso de la historia, ya que anunciaba la guerra con Estados Unidos y solicitaba una alianza

con México. Las consecuencias que acarreó el desciframiento son un claro ejemplo del impacto de la criptografía en la historia del mundo.

Sin embargo, no fue sino hasta la época posterior a la Primera Guerra cuando la tecnología electromecánica comenzó a cumplir un rol más importante en el cifrado, a partir de una motivación (lógicamente) militar. Así, aparecieron las **máquinas de cifrar basadas en rotores**, que combinaban la idea del cifrado automático (en contraste al método manual) con la ya conocida máquina de escribir.

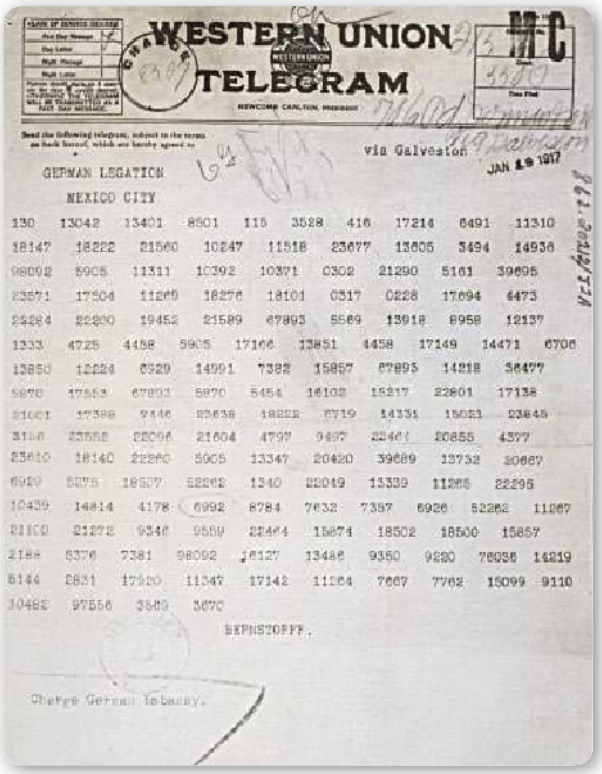

WESTERN UNION
TELEGRAM
via Galveston
JAN 19 1917
GERMAN LEGATION
MEXICO CITY
130 13042 13401 8501 115 3528 416 17214 6491 11310
18147 18222 21560 10247 11518 23677 13605 3494 14936
98092 5905 11311 10392 10371 0302 21290 5161 39695
23571 17504 11269 18276 18101 0517 0228 17694 6473
22284 22290 19452 21589 67893 5569 13918 8958 12137
1333 4725 4458 5905 17166 13851 4458 17149 14471 6706
13850 12224 6929 14991 7382 15857 67893 14218 36477
5870 17553 67893 5870 5454 16102 15217 22801 17138
21001 17388 9466 22538 18222 6719 14331 15081 23845
3184 23552 22098 21604 4797 9497 22464 20855 4577
23610 18140 22290 5905 13347 20420 39689 13732 20667
6929 5875 18557 52292 1340 22019 13339 11265 22295
10439 14814 4178 6992 8784 7632 7357 6926 52262 11067
21100 21272 9346 9559 22464 15874 18502 18500 15857
2184 5376 7381 98092 16127 13486 9350 9220 78036 14219
5144 2831 17920 11347 17142 11264 7667 7762 15099 9110
10482 97556 3569 5870
BERNSTORFF.
Charge German Legation.

Figura 13. El **telegrama Zimmermann** original, interceptado y descifrado por Gran Bretaña, cuyo contenido precipitó el ingreso de Estados Unidos en la primera guerra mundial.

LA MÁQUINA PURPLE

Purple se denominó a la máquina de cifrar japonesa utilizada durante la segunda guerra, creada con colaboración del matemático Teiji Takagi. Su cifrado fue roto por el criptógrafo de origen ruso William Friedman y su equipo en 1940, lo que hizo que pudiera interpretarse el mensaje de Japón a su embajada en Estados Unidos sobre el ataque a **Pearl Harbor** en 1941.

Las máquinas de rotores

El primer equipo electromecánico de cifrado de la historia fue la llamada **máquina de Hebern**, creada por Edward Hebern en 1917, que aplicaba rotores de manera similar a Bazeries pero cableados entre sí de distintas formas. Hebern logró que cada vez que se presionaba una tecla se moviera un rotor, modificando así el alfabeto de cifrado. Su éxito comercial fue escaso, pero sentó las bases para el resto de los sistemas.

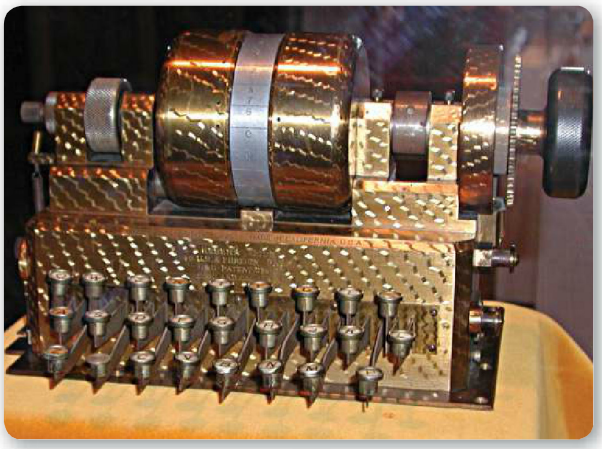


Figura 14. La **máquina de Hebern**, primera máquina de cifrar de su especie, fue creada en 1917 y patentada en 1918.

En 1920 nació **Kryha**, una máquina totalmente mecánica con distintas versiones, creada por el ucraniano Alexander Von Kryha, que fue utilizada por la diplomacia alemana de la época y por Marconi en Inglaterra. Kryha consistía en dos anillos concéntricos que contenían un alfabeto, donde el interior se corría por pasos mediante una palanca, lo que producía el cambio de alfabeto cada vez.
En 1923, el ingeniero sueco Arvid Damm desarrolló **A-21**, que utilizaba un tambor giratorio con 26 tiras de alfabetos que podían ser acomodadas en cualquier orden (similar a la tabla de Vigenère) y un alfabeto de referencia en el visor. Al pasar letra por letra, el tambor movía una tira de a pasos para lograr el cambio de alfabeto de cifrado cada vez.

Figura 15. William Friedman, uno de los más grandes criptógrafos de la historia, lideró el equipo que rompió el cifrado de la máquina japonesa **Purple**.

En 1925, el ingeniero sueco Boris Hagelin creó la **máquina Hagelin** luego de comprar la fábrica de Arvid Damm. Posteriormente, desarrolló varios modelos más de máquinas de cifrado. Su éxito se debió a haber logrado una alta periodicidad mediante el uso de rotores con número de dientes primos entre sí y también al diseño de los llamados **pinwheels**, dispositivos utilizados para producir secuencias pseudoaleatorias cortas de combinaciones, determinadas por el estado inicial de los rotores. El primer modelo se conoció como **B-21**, nombrada luego de la original A-21 de Damm.

La máquina Enigma

Enigma es, probablemente, la máquina de cifrar más famosa de la historia debido a su protagonismo en la Segunda Guerra Mundial. Fue creada por el ingeniero alemán Arthur Scherbius en 1923 y consistía, de forma similar a sus predecesoras, en un conjunto de rotores de 26 contactos eléctricos (uno por cada letra) montados sobre un mismo eje, que se desplazaban a modo de **odómetro** (una vuelta completa de cada rotor daba lugar a un movimiento del siguiente) al presionar cada tecla del teclado con el que contaba. En un sistema con cinco rotores, el período es de 26 elevado a la quinta potencia, o sea, de 11.881.376 pasos. El sistema contaba además con un panel superior con todas las letras del alfabeto, donde cada una se encendía al presionar la letra correspondiente, tanto en el proceso de cifrado como en el proceso de descifrado.

Figura 16. Modelo de una máquina **Enigma** original de cuatro rotores.

Las máquinas Enigma más avanzadas agregaron mayor complejidad en el cifrado, utilizando un código inicial de tres rotores más, intercambiables entre cinco posibles, y un tablero de interconexión que permitía intercambiar letras de a pares (esto último, sólo para los modelos militares).

En 1929, una Enigma no militar fue interceptada en su camino de Berlín (Alemania) a Varsovia (Polonia) por no haber estado rotulada como equipaje diplomático, y fue este ejemplar el que permitió realizar los primeros estudios para su criptoanálisis posterior. Fue así como el matemático polaco Marian Rejewski consiguió, en 1932, deducir parte del cableado de un rotor por la manera en la que cambiaban las letras. En 1939, Polonia compartió esos trabajos con Francia y Gran Bretaña, ante el riesgo de una invasión inminente por parte de Alemania.

Los aliados interceptaron el tráfico alemán durante los años siguientes con el fin de obtener información para romper el cifrado de Enigma, lo que hacia 1940 finalmente comenzaron a lograr. Hoy se cree que este hecho adelantó varios años el final de la guerra. Curiosamente, la información acerca de que el cifrado de Enigma había sido roto en la época de la Segunda Guerra no se reveló internacionalmente sino hasta fines de los años 60.

Figura 17. Bletchley Park (Buckinghamshire, Inglaterra) fue la instalación militar donde se descifraban los mensajes de Enigma durante la segunda guerra.

RESUMEN

En este capítulo estudiamos los sistemas criptográficos más importantes de la historia antigua y clásica, partiendo de los primeros métodos como el del César, el Atbash y el de Polybios. Luego vimos algunos medievales, que utilizaban discos o tablas, como el de Vigenère, Alberti, Playfair, Wheatstone y Bazeries. También vimos sistemas basados en cuestiones matemáticas y lógicas, como el de Vernam y el de Hill. Finalmente, analizamos las máquinas de cifrar, que hicieron su aparición en el siglo XX.

Actividades

TEST DE AUTOEVALUACIÓN

1. ¿Qué caracteriza a la criptografía clásica?
2. ¿Cuáles fueron los principales sistemas desarrollados para ocultar información en la antigüedad?
3. ¿Qué variantes básicas tiene el sistema de cifrado del César?
4. ¿Qué características tiene el cifrado de Vigenère?
5. ¿Por qué no tuvo más éxito el sistema de cifrado de Hill?
6. ¿En la creación de cuáles sistemas de cifrado participó Charles Wheatstone?

EJERCICIOS PRÁCTICOS

1. Identifique los hitos de la historia que acompañaron a los puntos importantes de la historia de la criptografía.
2. Investigue otros sistemas de cifrado clásicos que hayan sido utilizados más allá de los mencionados.
3. Arme modelos en cartón de los sistemas de cifrado de disco y pruebe su funcionamiento. Si tiene conocimientos de programación, puede probar escribir un código que lo simule.
4. Averigüe el método criptoanalítico que fue utilizado para romper el cifrado de la máquina Enigma.
5. Investigue el funcionamiento interno de las máquinas de cifrar de rotores y sus distintos niveles de complejidad.

PROFESOR EN LÍNEA

Si tiene alguna consulta técnica relacionada con el contenido, puede contactarse con nuestros expertos: profesor@redusers.com

*03

Criptografía moderna

La edad moderna de la criptografía comenzó a mediados del siglo XX, de la mano de una gran cantidad de avances. Aquí veremos en detalle los pilares fundamentales que funcionaron como base de conocimiento para desarrollarla y progresar hacia las técnicas utilizadas actualmente.

El fin de lo clásico

Los avances de la técnica fueron mejorando cada vez más los sistemas clásicos, pasando de métodos rudimentarios a otros más sofisticados, de forma tal que los objetivos se consiguieran con mayor garantía. Si bien desde el antiguo Egipto hasta la fecha existieron avances especialmente interesantes en épocas como el Medioevo y el siglo XIX, recién en la Segunda Guerra Mundial ocurrió un quiebre significativo. Este gran cambio se originó en la convergencia de varios adelantos de la matemática y de la tecnología, especialmente de la electrónica y las computadoras.

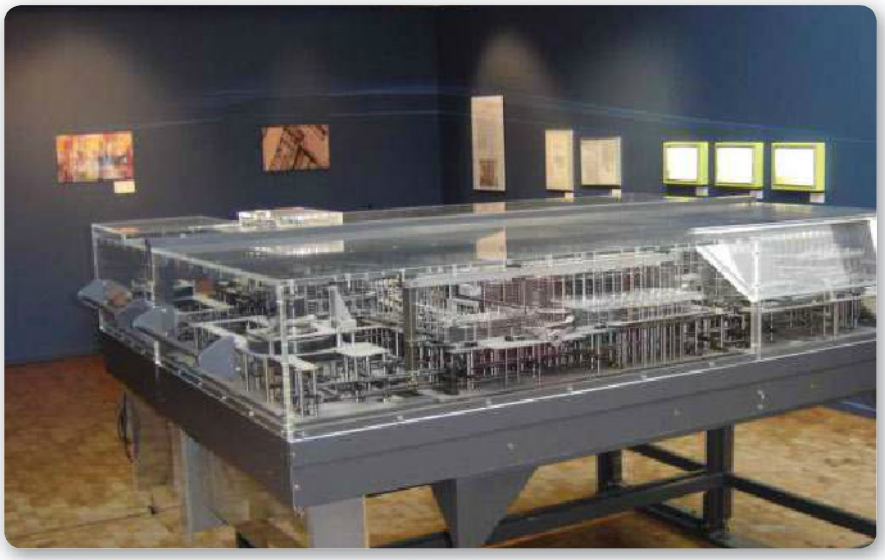


Figura 1. Z1 (1938) es considerada la primera computadora mecánica programable de la historia. Fue destruida junto a sus planos en la Segunda Guerra, en el bombardeo a Berlín de 1943.

La criptografía moderna comenzó con Claude Shannon, el primero en plantear una teoría matemática de la información. En 1949, Shannon publicó el artículo **Communication Theory of Secrecy Systems** en el **Bell System Technical Journal** y, al tiempo, el libro **Mathematical Theory of Communication**, junto a Warren Weaver. Estos trabajos acerca de la teoría de la información y la comunicación sentaron las bases teóricas para la criptografía y el criptoanálisis modernos.

Investigaciones relacionadas habían comenzado en la década 1910 de la mano de Andrei Markovi, luego de Ralp Hartley (precursor del sistema binario) en 1927, y también de Alan Turing, que en 1936 había

trabajado en el esquema de una máquina para tratar la información. Desde Shannon, la criptografía se mantuvo en el ámbito de gobiernos y algunas empresas, y salió a la luz recién hacia los años 70.

Figura 2. Modelo de **Harvard Mark I** (1944), la primera computadora electromecánica basada en la máquina analítica de Charles Babbage.

Fue justamente a principios de la década del 70 cuando los sistemas informáticos tuvieron una enorme evolución y comenzaron a ser utilizados cada vez más, aunque aún no a nivel hogareño. En ese contexto, comenzaron a aplicarse problemas matemáticos a la criptografía, de modo tal que pudiera intercambiarse información sobre la base de claves matemáticamente relacionadas. Así nació el **cifrado asimétrico**, que además dio pie al estudio y profundización de los problemas matemáticos implicados en dichos métodos. También en esa década aparecieron los primeros **algoritmos criptográficos basados en la matemática de los bits**, que podían ser procesados por computadoras y cuyo elemento secreto era una clave también representada en términos de unos y ceros. Ese fue el nacimiento del **cifrado simétrico**.

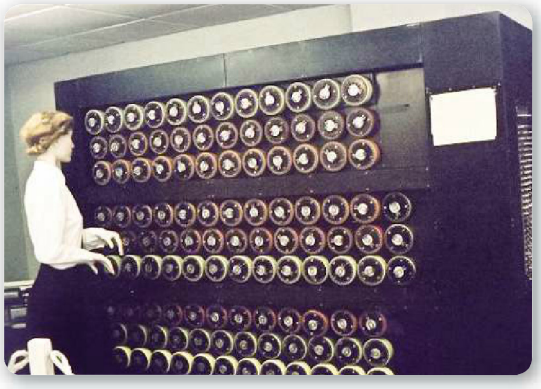


Figura 3. En 1939, Alan Turing diseñó **Bombe**, una máquina electromecánica que simulaba el comportamiento de las alemanas Enigma para intentar romper sus cifrados.

Con los avances matemáticos de la tecnología y de las ciencias de la computación, se dio origen a la criptografía moderna, que se basaría en tres pilares fundamentales: la **teoría de la información**, la **teoría de números** y la **teoría de la complejidad algorítmica**.

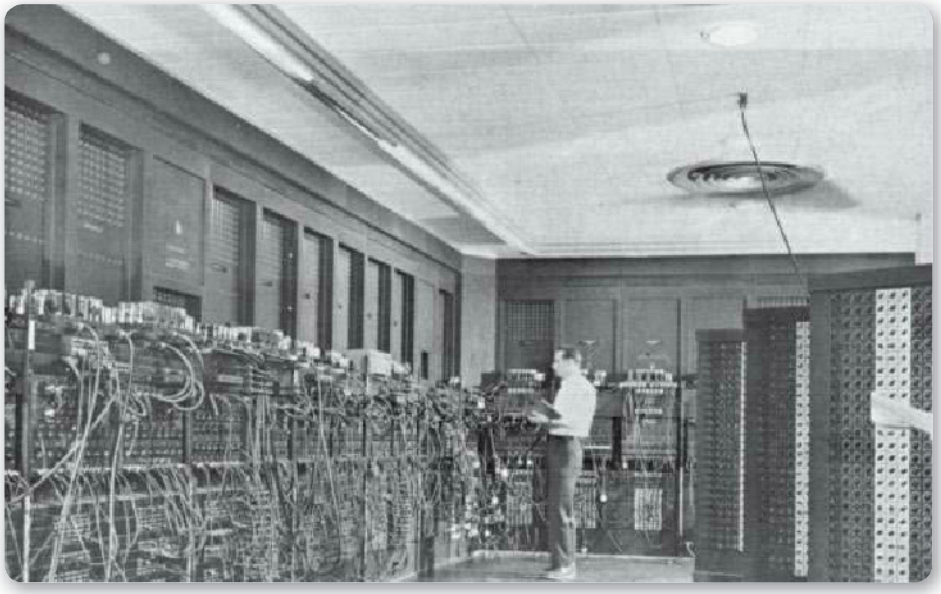


Figura 4. ENIAC (Electronic Numerical Integrator And Calculator) fue la primera computadora electrónica de propósitos generales, en 1946. Pesaba 27 toneladas.

Teoría de la información

En la época de Shannon existía un gran esfuerzo por utilizar más eficientemente los canales de comunicación, en ocasión de los avances militares. Los sistemas de telecomunicaciones estaban emergiendo

LA INFORMACIÓN EN CRIPTOGRAFÍA

El concepto de información puede tener distintas interpretaciones según el contexto. Shannon utiliza un enfoque de ingeniería y matemática. En criptografía, la definimos como el conjunto de datos inteligibles creado a partir de un lenguaje, que debe ser tratado durante su transmisión y almacenamiento para protegerla de personas no autorizadas.

fuertemente con la masificación del teléfono, los teletipos y los sistemas de radio, lo que forzó el desarrollo de bases teóricas sostenibles. El fin de los estudios era garantizar el transporte masivo de datos sin disminución de la **calidad**, con posibilidad también de **compresión** y **control de errores**.

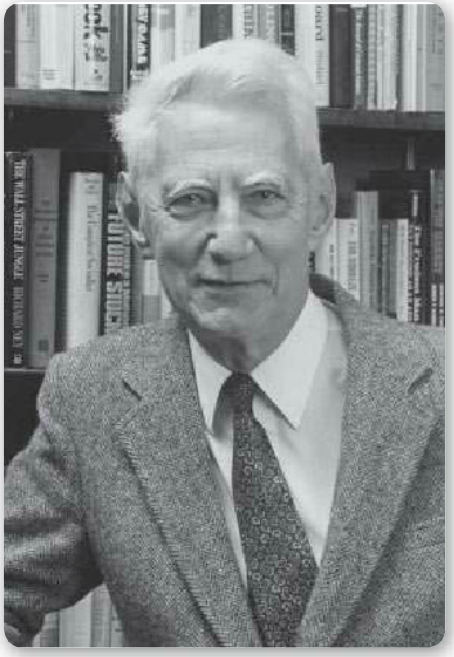


Figura 5. Claude Shannon (1916-2001) fue un ingeniero electrónico y matemático reconocido como el padre de la teoría de la información. También ha aportado al campo de las ciencias de la computación.

Así aparece la **teoría de la información**, con el objetivo de medir la cantidad de información que contiene un mensaje a partir de la cantidad media de bits necesaria para codificar todos los mensajes posibles utilizando lo que se denomina un **codificador óptimo** (aquel que usa la menor cantidad posible de bits para codificar un mensaje).

Para estudiarla, debemos partir de la base de que un mensaje puede entregar distinta cantidad de información dependiendo del contexto. Por ejemplo, podemos analizarlo en función de su extensión, de su utilidad, de su probabilidad, del entorno, etcétera. Consideramos, además, que frente a varios mensajes posibles (suponiendo a priori todos como **equiprobables**), el que contenga mayor cantidad de información será el que tenga una menor probabilidad de aparición.

TEXTOS FUNDAMENTALES

Las publicaciones de Shannon pueden descargarse de internet. **Communication Theory of Secrecy Systems** (Teoría de la comunicación de sistemas secretos) se encuentra disponible en el sitio **http://netlab.cs.ucla.edu/wiki/files/shannon1949.pdf** y **A Mathematical theory of communication** (Una teoría matemática de las comunicaciones) en **http://essrl.wustl.edu/~jao/itrg/shannon.pdf**.

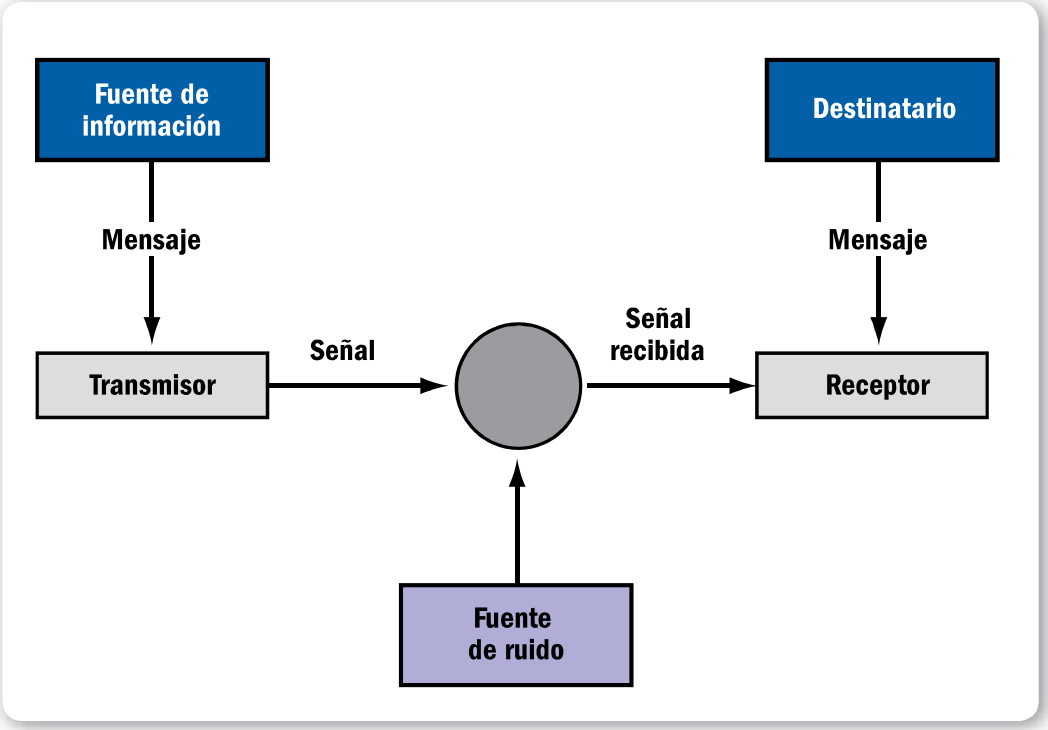


Figura 6. El **esquema de comunicación de Shannon** fue un modelo lineal sencillo en el que se basó para su desarrollo.

Entropía

La teoría de la información define el concepto de **entropía** (que no es la misma entropía de los fenómenos físicos) como la **medida de la incertidumbre de una fuente de información**. También puede explicarse como la **cantidad de información promedio** de todos los símbolos utilizados en un sistema.

> LA ENTROPÍA ES LA MEDIDA DE LA INCERTIDUMBRE DE UNA FUENTE DE INFORMACIÓN

Podría relacionarse con la definición de la física en cuanto a ser una **medida del desorden**. Matemáticamente, la entropía de un mensaje **X** o **H(X)** equivale al valor medio ponderado de la cantidad de información de sus estados, o también puede expresarse como una medida de la incertidumbre media sobre una variable aleatoria y la cantidad de información en bits. Para esto, se habrá definido previamente la cantidad de información como el logaritmo en base dos de la probabilidad de ocurrencia de un determinado estado.

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

Figura 7. El concepto fundamental de la teoría de la información es el de **entropía**. La **base 2** del logaritmo implica que se va a representar la información usando el **código binario** (**bits**).

Según Shannon, la medida de información ha de ser proporcional, o sea que un pequeño cambio en la probabilidad de aparición de un elemento debe alterar poco la entropía. También propone algunas propiedades importantes sobre la entropía:

- El máximo valor se da cuando todos los elementos son de aparición equiprobable.
- Tiene cota superior, dada justamente por el logaritmo.
- No puede ser negativa, lo que es evidente viendo la fórmula, ya que al ser **pi** una probabilidad, su valor estará entre 0 y 1.
- Solo se anula cuando no hay incertidumbre (probabilidad igual a 1).

Muy importante en criptografía, y especialmente en criptoanálisis, es el concepto de **entropía condicional**, que implica que existe una variable aleatoria Y mutuamente dependiente de X, de manera tal que el conocimiento de una brinda información sobre la otra, con lo que permite disminuir su incertidumbre y, por lo tanto, su entropía. Su interés radica en que actúa como parámetro para evaluar el nivel de seguridad de un sistema.

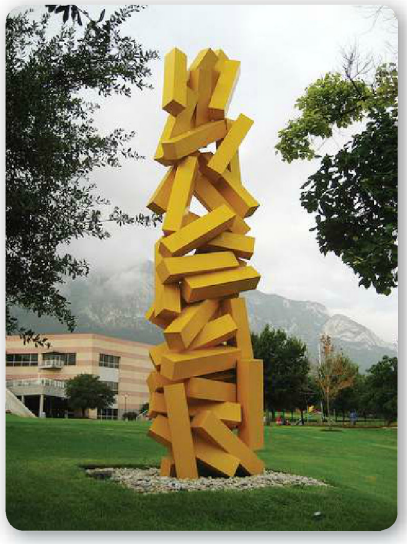


Figura 8. En los jardines de la Universidad de Monterrey (México) se encuentra una escultura dedicada a la **entropía**.

Más conceptos de la teoría

La **tasa** (**ratio**) de la entropía de una sucesión de n variables aleatorias describe su ritmo de crecimiento con el aumento de n. Es decir, la tasa corresponde a la cantidad de bits de información de cada caracter para los mensajes de longitud n caracteres. Análogamente, podemos interpretar la tasa de un idioma como la cantidad de información de cada letra. Para el caso del español, se estima que vale entre 1.2 y 1.5 bits, y para el caso del idioma inglés se encuentra alrededor de 1.3 bits. Esto se da porque las letras de un texto no tienen la misma probabilidad de aparición ni frecuencia, y porque los lenguajes tienen ciertas estructuras que les otorgan **predictibilidad**. En la misma línea, se define la **redundancia** del lenguaje como la diferencia entre la tasa absoluta y la real.

Otra definición de Shannon fue la de interpretar el **secreto de un criptosistema** como la incertidumbre de un mensaje cuando el criptograma es conocido. Extendiendo el concepto, definió el **secreto perfecto** como aquel que se da en un criptosistema cuando el conocimiento del criptograma no ofrece ninguna información sobre el mensaje (la probabilidad de acierto del próximo elemento es igual a la del estado anterior).

También definió como **distancia de unicidad** al bloque de texto cifrado mínimo que se necesita para poder intentar con buenas probabilidades un ataque en busca de la clave. A mayor largo de criptograma, más fácil el criptoanálisis. Si tenemos un cifrador realmente aleatorio, tendremos distintas etapas en el criptoanálisis: en la primera se requerirá mucho esfuerzo para averiguar algo, en la segunda se tendrá una cantidad suficiente de información como para averiguar algo, y en la tercera se alcanzará la certeza mediante la obtención de soluciones únicas.

VARIABLE ALEATORIA

En **probabilidad y estadística**, se dice que una variable aleatoria es una variable que toma valores obtenidos a partir de una prueba aleatoria. Matemáticamente, es una función que asigna eventos a números reales. La probabilidad de aparición de los distintos valores es descrita por la denominada **distribución de probabilidad**.

Confusión y difusión

Shannon también planteó el uso de dos técnicas en particular para tomar en cuenta en los procesos de cifrado: **confusión** y **difusión**. Llamamos **difusión** a las transformaciones realizadas al mensaje con el objetivo de dispersar las propiedades estadísticas del lenguaje en el texto cifrado. La manera de obtenerla es a partir del uso de la ya estudiada técnica de **transposición**.

Por otra parte, llamamos **confusión** a las transformaciones realizadas al mensaje con el objetivo de mezclar sus elementos, logrando con esto un aumento de la complejidad de la dependencia entre el texto cifrado y la clave. La manera de obtenerla es a partir del uso de la ya estudiada **sustitución**.

Estas dos técnicas son equivalentes a las estudiadas en criptografía clásica, solo que antes se orientaban al cifrado de caracteres y aquí al cifrado de bits.

Teoría de números

La rama de las matemáticas que estudia los **conjuntos discretos** (**finitos** o **infinitos numerables**) se denomina **matemática discreta**. La **teoría de números** es una de sus áreas de estudio y se encarga de estudiar las propiedades de los números enteros. Muchos de sus problemas clásicos pueden ser entendidos por no matemáticos, y de hecho el término por el cual se la conoció históricamente es **aritmética**.

Generalizando, se enfoca en los llamados **dominios enteros**, que son **anillos conmutativos** con **elemento unitario** y **cancelación** (para nuestro caso, no es importante su compresión).

EL GRAN EULER

Leonhard Euler fue un matemático y físico suizo del siglo XVIII, el más importante de su época y uno de los más grandes de todos los tiempos. Realizó descubrimientos en **cálculo** y **teoría de grafos** e introdujo mucha de la terminología y **notación moderna**. Además, trabajó en campos como la óptica, la mecánica y la astronomía.

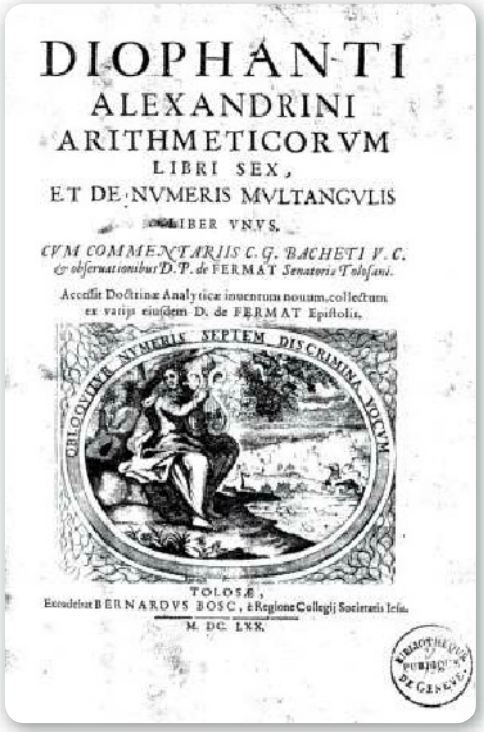

DIOPHANTI
ALEXANDRINI
ARITHMETICORVM
LIBRI SEX,
ET DE NVMERIS MVLTANGVLIS
LIBER VNVS.
CVM COMMENTARIIS C. G. BACHETI V. C.
& observationibus D. P. de FERMAT Senatoris Tolosani.
Accessit Doctrinæ Analyticæ inventum novum, collectum
ex variis ejusdem D. de FERMAT Epistolis.
TOLOSÆ,
M. DC. LXX.

Figura 9. En el año 250 de nuestra era, el matemático griego Diofanto de Alejandría escribió un tratado de trece libros que llamó **Arithmetica**. Aquí, una edición del año 1621.

Congruencia

En teoría de números, una operación fundamental en la que se basan las operaciones de cifrado es la **congruencia**, según la cual al dividir dos números enteros **a** y **b** por un número natural **n**, se obtiene el mismo resto, llamado **módulo**.

Es decir, si tomamos **a** y **b**, decimos que **a es congruente con b en el módulo** (o cuerpo) **n** (**Zn**) si existe algún número entero **k** (al que llamamos **resto**, o residuo) que divide de manera exacta la diferencia entre ellos (**a-b**).

Por ejemplo, si queremos averiguar si **24** es congruente con **6 módulo 4**, hacemos: **24–6=18= k*4**, de lo que resulta **k=6**. Entonces decimos: **24 mod 4=6**. La congruencia se expresa con el siguiente símbolo: ≡. Por lógica, la congruencia está estrechamente vinculada a las cuestiones de divisibilidad de los números.

Tan importante es el tema de la congruencia en matemática discreta que uno de los matemáticos más grandes de todos los tiempos, Carl Friedrich Gauss, en su libro **Disquisitiones**

Arithmeticae (1801), introdujo el concepto de **aritmética modular** y lo definió como un sistema aritmético para clases de equivalencia de números enteros, denominadas **clases de congruencia**.

Divisibilidad

Una de las cuestiones que más interesan en criptografía es el cálculo del **máximo común divisor** (**MCD**) en relación al cálculo de la congruencia. Para abordar esto se pueden utilizar distintos algoritmos, de distinto nivel de dificultad y computabilidad.

Uno de ellos es el llamado **algoritmo de Euclides**, descrito por el matemático griego Euclides en su tratado **Elementos**. También interesa una extensión que se le realiza a fin de expresar al MCD como combinación lineal. El algoritmo puede explicarse de la siguiente forma: considerando que **x** divide a **a** y **b**, entonces **a=x*a'** y **b=x*b'**. Por consiguiente: **a-k*b=x*a'-k*x*b'=x(a'-k*b')**. Esto permite concluir que **x** divide a **(a-k*b)**.

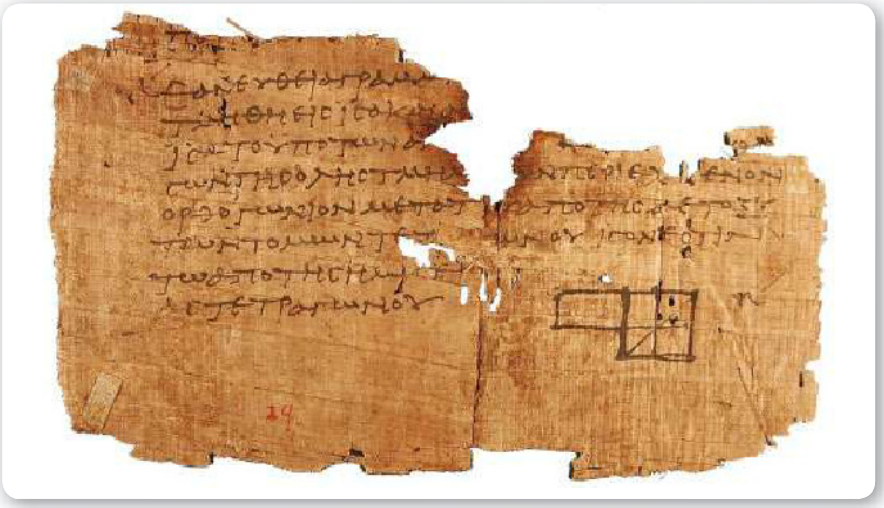


Figura 10. Fragmento del papiro de la obra **Elementos** de Euclides, el padre de la geometría.

Este algoritmo no es eficiente para su implementación directa en software porque tiene como requisito el almacenar en memoria todos los valores de los restos hasta llegar al final, lo que en caso de ser números muy grandes podría representar un tamaño excesivo. Su importancia en criptografía es que nos permite hallar números **primos entre sí** o **coprimos** (cuando su único divisor común es 1).

Figura 11. Ejemplo gráfico del algoritmo de Euclides, originalmente planteado para averiguar divisores de segmentos. En este caso, **EF** es la mayor medida común.

La cuestión de la divisibilidad también es abordada por el llamado **pequeño teorema de Fermat** (que no es el famoso **último teorema de Fermat**) que dice: si **p** es un número primo, entonces para cada número natural **a**, a^p es congruente con **a mod p**. Esto significa que al elevar **a** a la potencia **p**, y a eso restarle **a**, el resultado es divisible por **p**. Esto será fundamental en algoritmos criptográficos basados en el **problema de la factorización de números grandes**, como **RSA**.

Este teorema fue generalizado por medio del **teorema de Euler**, que dice que si **a** y **n** son enteros **coprimos**, entonces $a^{\varphi(n)} \equiv 1 \pmod{n}$. Donde **φ(n)** es la **función fi** (o **phi**) **de Euler**, que se define como la cantidad de enteros positivos menores o iguales a **n** y coprimos con **n**. El tamaño del **grupo multiplicativo** de enteros **módulo n** se puede obtener como resultado de esta función.

PRIMALIDAD

Considerando que un número es primo cuando es divisible solo por sí mismo y por 1, el problema de la primalidad implica la determinación de si un número **n** cumple con esta definición. Los algoritmos que **no logran verificar** que **n** es divisible se denominan **chequeos de primalidad**, en tanto que las **pruebas de primalidad** buscan verificar de manera determinística que **n** sea primo.

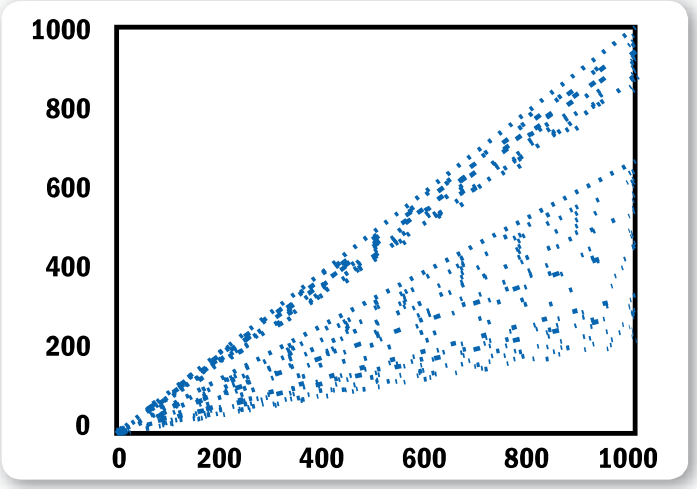


Figura 12. Primeros 1000 valores de la función **fi de Euler**, representados gráficamente.

Inversibilidad

Para recuperar un cifrado (considerado como conjunto de operaciones matemáticas), la función utilizada debe tener **función inversa** (o **recíproca**). Esto requiere definir el **inverso multiplicativo** de un **elemento a**, que es aquel elemento que multiplicado por **a**, da como resultado la unidad. Lo escribimos como **1/a** o $\mathbf{a^{-1}}$. Llevándolo a la aritmética modular, si una función es el valor **a** dentro de un **cuerpo n**, el **inverso multiplicativo modular** $\mathbf{a^{-1}}$ **mod n** será el que tendremos que encontrar para poder realizar la operación de descifrado. Así, si **a*x≡1 mod n**, decimos que **x es el inverso multiplicativo de a en Zn**. Es importante aclarar que el inverso de un elemento en **Zn** no existe siempre, y que si **n** es un **número primo p**, todos los elementos de **Zp** tienen inverso, excepto el cero.

> PARA RECUPERAR MATEMÁTICAMENTE UN CIFRADO LA FUNCIÓN UTILIZADA DEBE TENER INVERSA

Para obtener el inverso multiplicativo se suele aplicar el **algoritmo de Euclides extendido**, que al asumir **|n|<p** se ejecuta en un **tiempo polinomial** (concepto que veremos en las próximas páginas). También podemos aplicar el **método de exponenciación modular directa** (que utiliza la función de Euler) pero esto deriva en tiempos mayores, lo que no es conveniente excepto en ciertas ocasiones particulares.

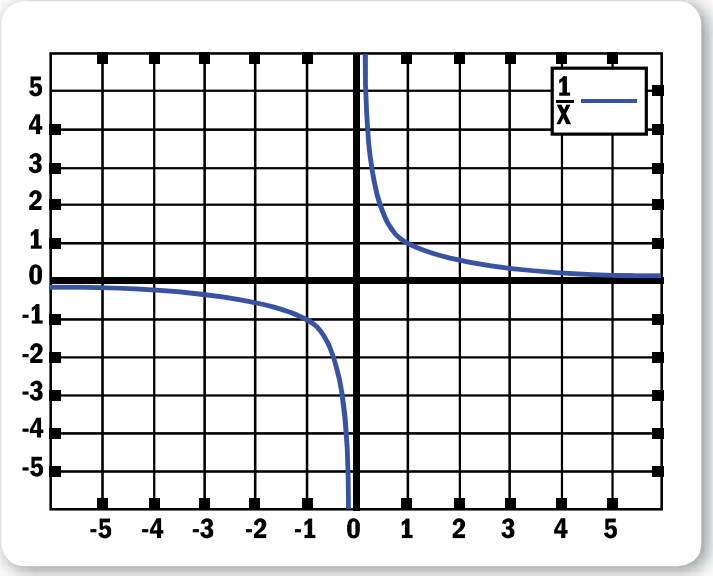


Figura 13. En la función **y=1/x** (hipérbola equilátera), para cada valor de **x** (excepto el 0) su inverso multiplicativo lo representa el **eje y**.

Complejidad algorítmica

Antes de hablar de la complejidad algorítmica, es necesario definir la **teoría de la complejidad computacional**, que es una de las ramas de la teoría de la computación que estudia la clasificación de los problemas computacionales en función de su dificultad. Por ejemplo, clasifica un problema como **inherentemente difícil** si su solución requiere de una cantidad significativa de recursos, independientemente del algoritmo.

Uno de los objetivos roles de esta teoría es definir los límites prácticos de lo que un sistema puede resolver. Además, estudia

TEORÍA DE LA COMPUTACIÓN

Es la rama de las matemáticas que estudia las **capacidades fundamentales de las computadoras**, buscando modelos matemáticos que describan la realización de operaciones y clasificando los problemas computacionales. Sus principales ramas son la **teoría de autómatas**, la **teoría de la computabilidad** y la **teoría de la complejidad computacional**.

todos los posibles algoritmos que se podrían utilizar para resolver un problema en particular.

Así aparece el estudio de la **complejidad algorítmica**, que se enfoca en la cantidad de recursos que requiere un algoritmo para resolver un problema, pudiendo determinar así su eficiencia. Los criterios de evaluación no proporcionan medidas absolutas sino relativas. Esta teoría introduce modelos matemáticos para estudiar los problemas y cuantificar los recursos requeridos para su resolución, y los agrupa además por clases de complejidad, que son los grupos de **problemas de decisión** (aquellos con respuesta sí o no) cuya complejidad está relacionada.

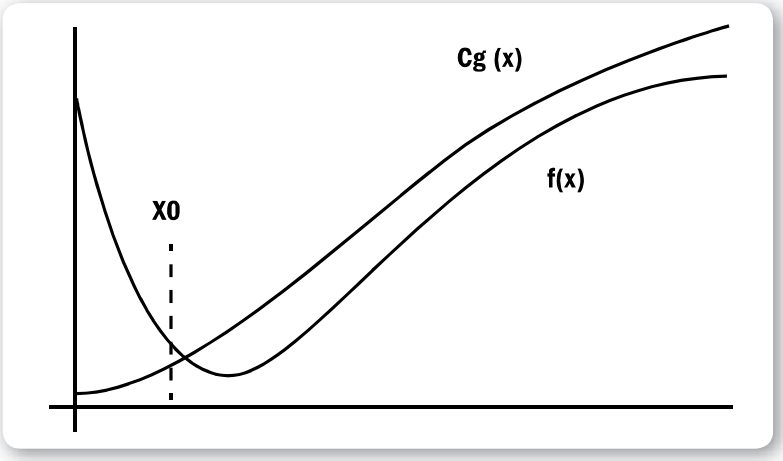


Figura 14. En análisis de algoritmos, se define **cota superior asintótica** a una función que actúa como límite superior de otra cuando su argumento tiende a infinito.

Los recursos que se estudian en general en complejidad computacional son el **tiempo** y el **espacio**. El tiempo estará representado por la cantidad de pasos que tomará la ejecución de

TEORÍA DE LA COMPUTABILIDAD

Es la rama de la teoría de la computación que estudia los **límites de la posibilidad** de solucionar problemas **utilizando algoritmos**. Es de utilidad para conocer de antemano los problemas que no se pueden resolver algorítmicamente. Según su grado de imposibilidad, clasifica los problemas en **computables**, **semicomputables** e **incomputables**.

un algoritmo, y el espacio por la cantidad de memoria. Lo que buscamos es analizar cómo crece la cantidad de pasos respecto al tamaño de una entrada determinada, para saber cuánto tiempo se requiere. En criptografía se estudian en particular dos clases de complejidad, relacionadas con el tiempo:

- **Tiempo Polinomial** (**P**): comportamiento similar al lineal.
- **Tiempo polinomial no determinista** (**NP**): comportamiento exponencial.

Figura 15. Esquema de la **máquina de Turing**, que permitió demostrar conceptualmente que existían problemas que no podían ser resueltos por una máquina.

Clase P

La **clase de complejidad P** consiste en los problemas de decisión que se pueden resolver en una máquina secuencial determinística (**máquina de Turing**) en un **tiempo polinomial**, proporcional al tamaño de la entrada. Es decir, puede que los haya más complejos y menos complejos,

TEORÍA DE AUTÓMATAS

Es la rama de la teoría de la computación que provee **modelos matemáticos** para formalizar el concepto de computadora o algoritmo de forma simplificada y general para que puedan ser analizadas sus capacidades. Los principales modelos son: **autómatas finitos**, **autómatas con pila** y **máquinas de Turing**.

pero los problemas de clase P suelen ser tratables en su mayoría. El cálculo del máximo común divisor se encuentra en esta categoría.

Además, existe una clase llamada **P-completo**, que incluye a los problemas P que pueden reducirse a NP en un tiempo **polilogarítmico** utilizando para ello una **máquina paralela**. De todas maneras, lo que interesa saber no es si se puede resolver un problema con velocidad en una máquina paralela sino si esta lo podría resolver mucho más rápido que una máquina secuencial.

En cuanto a los algoritmos, existe una distinción análoga a la complejidad entre los de **tiempo polinomial** (suficientemente eficiente) y los de **tiempo exponencial** (muy ineficiente). Los de tiempo polinomial tienen **función de complejidad temporal** para una **función polinómica** (la variable es el tamaño de la entrada) y se suelen descubrir analizando en profundidad la estructura del problema. Un algoritmo que no permite acotar su función de complejidad temporal se considera de tiempo exponencial. En ciencias de la computación, se dice que un problema no está bien resuelto hasta tanto no se conozca un algoritmo de tiempo polinomial que pueda resolverlo. Si no puede resolverlo, se considerará **intratable**.

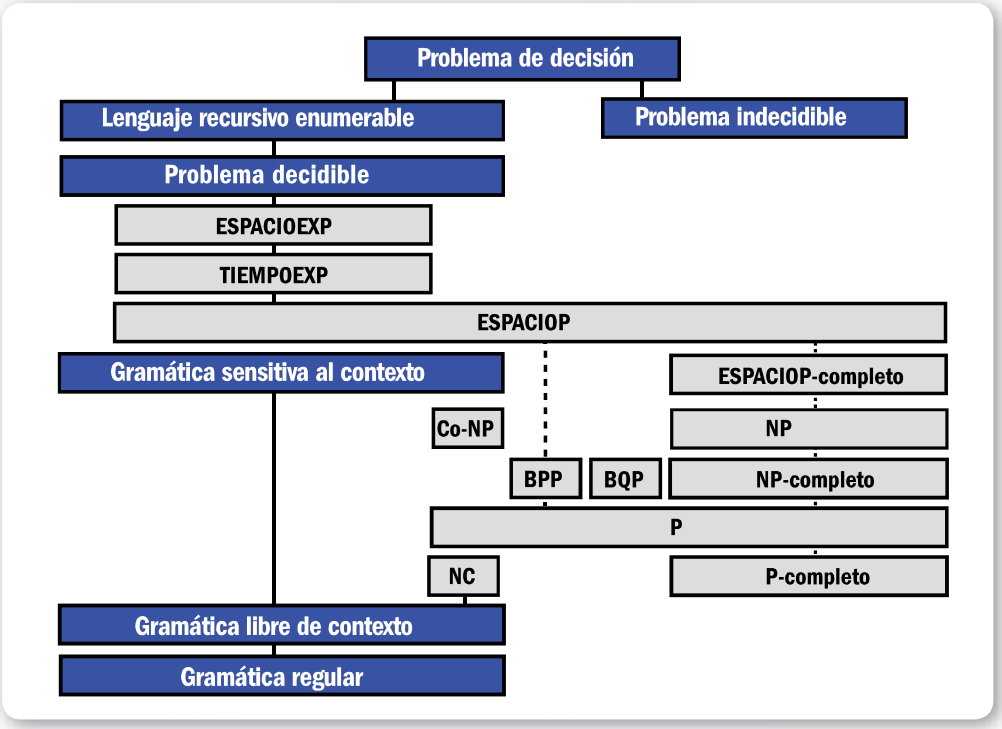


Figura 16. Clasificación completa de los problemas según su clase de complejidad. Las líneas punteadas indican subconjuntos de los que no se sabe si son estrictos.

Clase NP

NP proviene de *Nondeterministic Polynomial Time* (**tiempo polinomial no determinista**) y se refiere a la clase de complejidad que incluye a los problemas que se pueden resolver en tiempo polinomial por medio de una **máquina de Turing no determinista**, que es aquella que cuenta con al menos un par **estado-símbolo** con más de una combinación posible de actuaciones (si existe a lo sumo una, será determinista). El no determinismo puede permitir la reducción de la complejidad de una solución (por ejemplo de exponencial a polinomial).

Esta clase es de interés por el hecho de contener una gran cantidad de problemas de búsqueda y optimización, para los que se espera saber si existe una solución (o una mejor que la conocida). Por ejemplo, el **problema del viajero** (búsqueda de una ruta óptima que pase por todos los nodos de un grafo) está incluido en esta clase.

Al parecer, para ciertos problemas NP (los llamados **NP-completos**) no se puede encontrar un algoritmo que resulte mejor que una **búsqueda exhaustiva**.

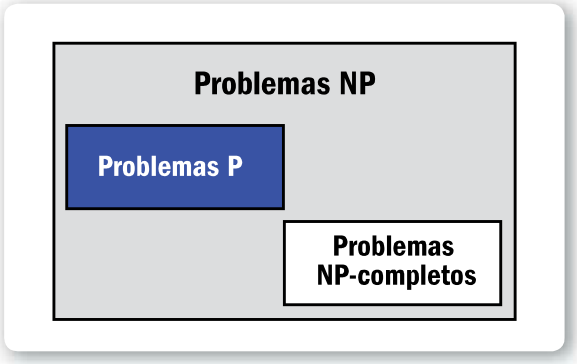


Figura 17. **NP** es una generalización de **P**, por lo que podemos suponer trivialmente que **P** es un **subconjunto** de **NP**. Muchos especialistas creen que lo anterior es estricto, pero esto aún no ha sido demostrado.

LA COMPLEJIDAD DE KOLMOGÓROV

Las ciencias de la computación definen la **complejidad de Kolmogórov** (también conocida como **complejidad descriptiva** o **entropía algorítmica**) como una medida de los recursos computacionales necesarios para describir una cantidad de información determinada. Requiere un **lenguaje descriptivo** para las secuencias (cualquier lenguaje de programación).

En criptografía, es de interés encontrar funciones llamadas **de un solo sentido**, que sean fáciles de calcular en sentido directo pero muy difíciles en sentido inverso, excepto que se conozca un dato denominado **secreto** (o **trampa**). Estas dan lugar a problemas NP como el **problema de la mochila**, el **de la factorización de números grandes** y el **del logaritmo discreto**.

En definitiva, quizás la pregunta más importante en la teoría de la computación involucra a las dos clases mencionadas: **¿Es P igual a NP?** Por el momento, esta cuestión aún no tiene respuesta.

RESUMEN

En este capítulo hemos planteado los pilares de la criptografía moderna, comenzando por la teoría de la información. También vimos la teoría de números, referida a la rama de las matemáticas que incluye problemas que solo involucran números enteros. Finalmente, estudiamos la teoría de la complejidad algorítmica, que analiza los distintos tipos de problemas que puede resolver una computadora con un algoritmo, clasificándolos según lo tratables que puedan ser.

Actividades

TEST DE AUTOEVALUACIÓN

1. ¿Cuáles fueron los factores que dieron origen a la criptografía moderna?
2. ¿Quién fue y qué hizo Claude Shannon?
3. Explique el concepto de entropía según la teoría de la información.
4. Explique la relación entre los elementos del esquema básico de Shannon para la comunicación.
5. ¿Cuál es la importancia de la teoría de números para la criptografía?
6. ¿Qué estudia la teoría de la complejidad algorítmica?
7. ¿Con qué parámetros se mide la eficiencia de un algoritmo?

EJERCICIOS PRÁCTICOS

1. Estudiar otras clases de complejidad computacional.
2. Identificar las computadoras construidas en el mundo entre los años 30 y 50.
3. Investigar las teorías de Alan Turing y su rol en las ciencias de la computación.
4. Estudiar los métodos de detección y corrección de errores según la teoría de la información y la computación.
5. Estudiar los problemas típicos de la teoría de números.

PROFESOR EN LÍNEA

Si tiene alguna consulta técnica relacionada con el contenido, puede contactarse con nuestros expertos: profesor@redusers.com

*04

Cifrado simétrico

Los criptosistemas modernos están basados en dos métodos de cifrado: simétrico y asimétrico, y cada uno posee tanto ventajas como desventajas. En este capítulo veremos el cifrado simétrico, donde una misma clave compartida es utilizada para realizar el proceso de cifrado y también el de descifrado.

Bloque y flujo

En los sistemas clásicos, se tomaban caracteres de a uno o de a varios para operar, pero si representamos información en forma de unos y ceros, la unidad mínima de trabajo es el bit, aunque también podríamos tomar conjuntos de bits para introducir en el algoritmo. Si tomamos de a uno le llamaremos **cifrado de flujo**, y si tomamos de a varios le llamaremos **cifrado de bloque**.

Cifrado de flujo

Aquí se utiliza la idea de Vernam que cumple con el **secreto perfecto** de Shannon: claves equiprobables y de un solo uso, y espacio mayor o igual al del mensaje. En la práctica esto es imposible ya que la clave se comparte por un canal inseguro (si fuera seguro enviaríamos el mensaje) y no puede ser de ser de longitud infinita (desbordaría la capacidad del canal). Para resolverlo, se genera una **secuencia pseudoaleatoria** (aleatoria no es posible) a partir de un **algoritmo determinístico** basado en una **semilla K de n bits** que produzca una secuencia **S**, llamada **secuencia cifrante**, con períodos de 2^n bits, con lo que solo se necesita enviar la semilla de la manera más segura posible. En la práctica, se utilizan semillas del orden de las **centenas de bits**.

XOR ⊕		
A	**B**	**Z**
0	0	0
0	1	1
1	0	1
1	1	0

Figura 1. La operación **XOR** es muy utilizada en criptografía por su funcionamiento, según el cual al tomar cualquier par de variables (**A** y **B**, **A** y **Z** o **B** y **Z**), el resultado de operarlos da siempre el valor de la variable restante.

En el diseño se busca que las secuencias de bits iguales entre dos bits distintos, llamadas **rachas**, se distribuyan estadísticamente para comportarse como aleatorias (tendremos menos rachas largas que cortas). Por otro lado, necesitamos conocer el comportamiento de una secuencia en relación al desplazamiento de **x bits** sobre esta, llamado **función de autocorrelación fuera de fase** o **AC(x)**. A este respecto, aparecen los **postulados de Golomb**:

- **G1:** debe existir la misma cantidad de unos que de ceros, o como máximo un bit distinto de diferencia. Es decir, la probabilidad de recibir 1 ó 0 es la misma.
- **G2:** las rachas deben seguir una progresión geométrica (la mitad de las rachas de longitud 1, un cuarto de longitud 2, un octavo de longitud 3, etcétera). Es decir, con independencia de los bits recibidos anteriormente, la probabilidad de recibir 1 ó 0 sigue siendo la misma.
- **G3:** la autocorrelación debe ser constante para todo valor de x bits de desplazamiento. Es decir, sin importar el fragmento de secuencia elegido, no contiene más cantidad de información el fragmento anterior.

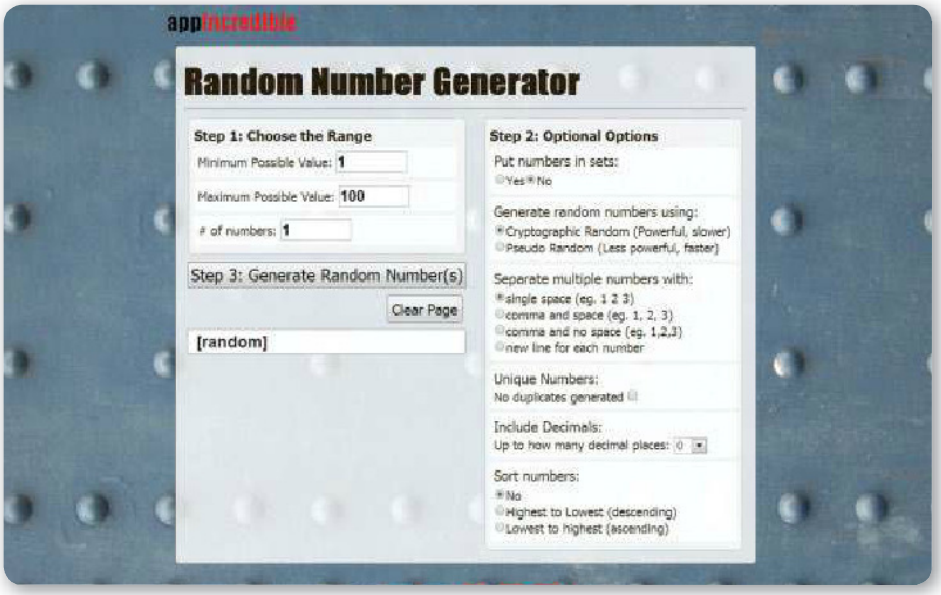


Figura 2. Es posible obtener aleatoriedad mediante servidores en internet llamados **Online Random Number Servers**, la mayoría provistos por universidades y con distintos métodos de generación.

El desafío es crear un **generador de secuencias** con las características antedichas para implementar en software o hardware, con el mejor rendimiento. Las propuestas aparecen principalmente en

forma de algoritmos llamados **PRNG** (*Pseudorandom Number Generators*) o **DRBG** (*Deterministic Random Bit Generator*). Algunos de los más conocidos son: **LCG** (*Linear Congruential Generator*), **LFSR** (*Linear Feedback Shift Register*), **BBS** (*Blum Blum Shub*) y **MWC** (*Multiply-with-carry*).

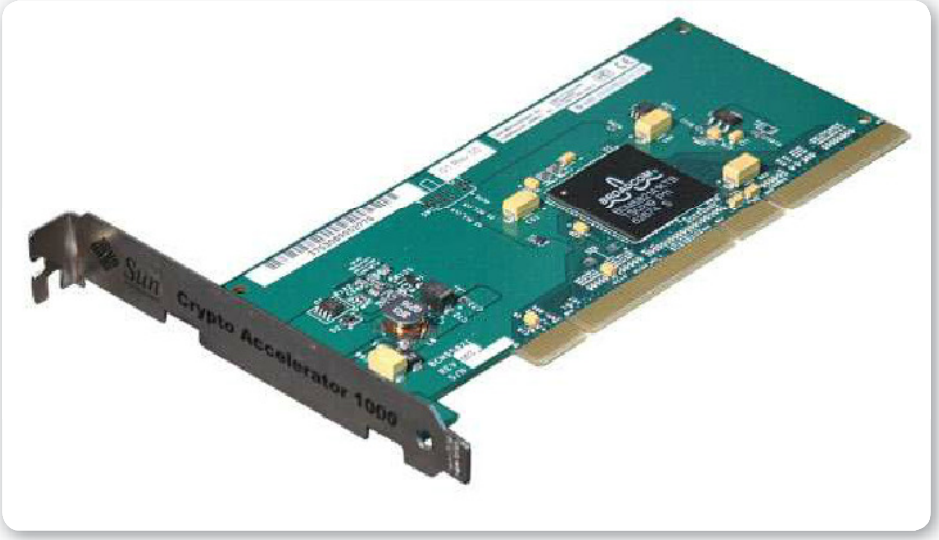


Figura 3. Existe hardware especialmente diseñado para proporcionar aleatoriedad basada en fenómenos físicos: los **TRNG** (**True Random Number Generators**).

El algoritmo de flujo más popular es **RC4** (*Rivest Cipher 4*), diseñado por Ron Rivest en 1987. RC4 es marca registrada y nació secreto, pero en 1994 fue descrito su funcionamiento en la lista de correo **cypherpunks**, con lo que comenzó a ser utilizado masivamente dada su simplicidad, pero bajo el nombre de **ARC4**. Consiste en dos algoritmos: *Key Scheduling Algorithm* (**KSA**) para inicializar la permutación en el vector **S** (**256 bits**) y *Pseudo-Random Generation*

NSA NOS ESPÍA

Los periódicos **The Guardian** y **The New York Times** reportaron que la **National Security Agency** (**NSA**) insertó un **backdoor** en un **PRNG** del estándar **SP 800-90** del **NIST** (el **Dual_EC_DRBG**), que permitiría a la NSA la posibilidad de leer material cifrado que haya utilizado tal algoritmo. Esta información la proveyó el controversial Edward Snowden.

Algorithm (**PRGA**), que modifica los estados y emite un byte de secuencia cifrante por cada iteración.

Otro conocido algoritmo de flujo es **A5/1** (1994), famoso por ser usado en **GSM**. Si bien nació secreto, terminó por ser de dominio público dadas sus debilidades halladas. Cuenta con una versión más débil, diseñada para exportación, llamada **A5/2**.

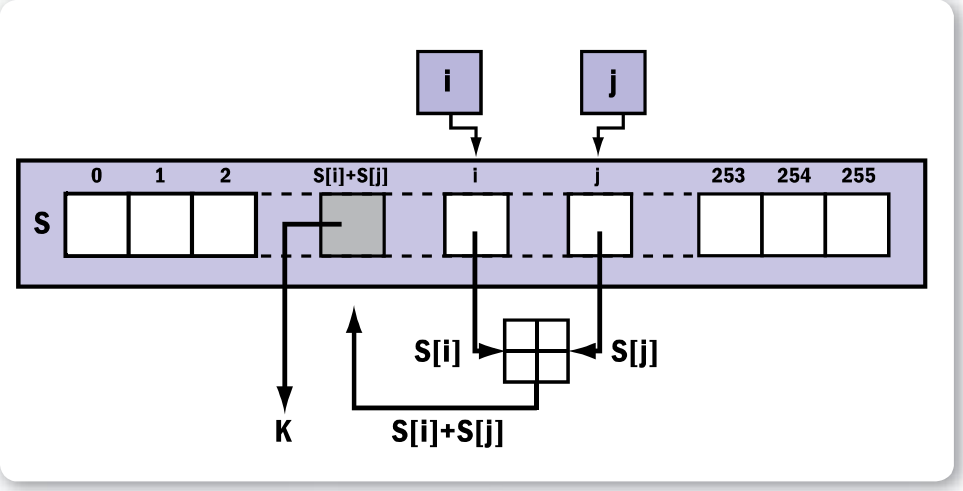


Figura 4. Esquema base de **RC4**. Los **S(i)** y **S(j)** se suman en módulo 256 y el resultado se toma como índice. Luego, **S(S(i) + S(j))** se toma como byte de la secuencia cifrante **K**.

También en 1994 vio la luz **SEAL** (*Software-Optimized Encryption Algorithm*), un algoritmo optimizado para equipos de 32 bits que es, de hecho, una familia de funciones pseudoaleatorias de secuencias cifrantes que puede generar porciones arbitrarias de estas sin comenzar necesariamente desde el principio. Sus creadores fueron Phillip Rogaway y Don Coppersmith, de IBM. En líneas generales, una ventaja del cifrado de flujo es la alta velocidad, ya que no hay que

PRUEBAS DE ALEATORIEDAD

Para la aleatoriedad de secuencias cifrantes y claves existen estándares de **ANSI** (**American National Standards Institute**) y del **NIST** (**National Institute of Standards and Technology**), que proponen quince pruebas estadísticas a modo de instrumento para validar o rechazar hipótesis de modelización probabilística. Pueden encontrarse en **http://csrc.nist.gov/rng**.

esperar un bloque para cifrarlo pues se procesa de a bits, y además es resistente a errores ya que el cifrado es independiente en cada elemento. Como desventaja, podemos citar la baja difusión de los elementos del criptograma y el hecho de que se puedan alterar elementos individualmente.

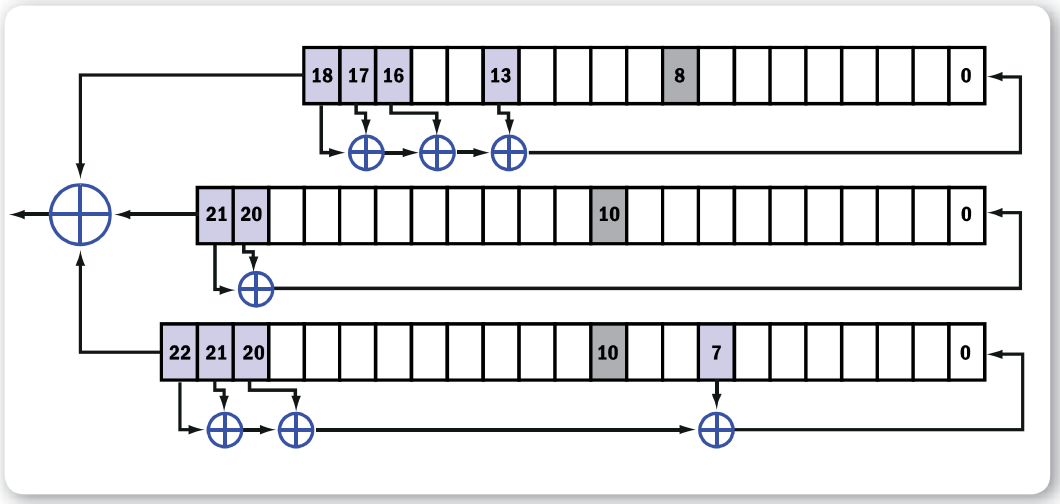


Figura 5. A5/1 usa tres registros **LFSR** de conteo irregular. Un registro es desplazado si su bit de reloj (anaranjado) coincide con uno o ambos de los otros dos registros.

Cifrado de bloque

Aquí el mensaje es agrupado en **fragmentos fijos de bits** para luego ser operados en el algoritmo y con la clave. Aquí, un error en solo un bit afecta a todo el bloque, y se requiere especial tratamiento para detectarlo y corregirlo.

El tamaño ideal del bloque no es una decisión trivial: si es muy pequeño se haría más fácil un ataque estadístico y si es muy

RELLENO DE BLOQUES

Normalmente, el mensaje no es múltiplo del tamaño del bloque, por lo que se realiza un relleno para que así sea. Este relleno puede ser, por ejemplo, de unos al principio o al final, o repitiendo el último elemento. El esquema de rellenado más utilizado es el que establece el estándar **PKCS#5** (**Public-Key Cryptography Standards 5**).

grande bajaría el rendimiento (tardaría más en procesar cada bloque). Esta solución de compromiso lleva a que en la práctica sean de entre 64 y 128 bits.

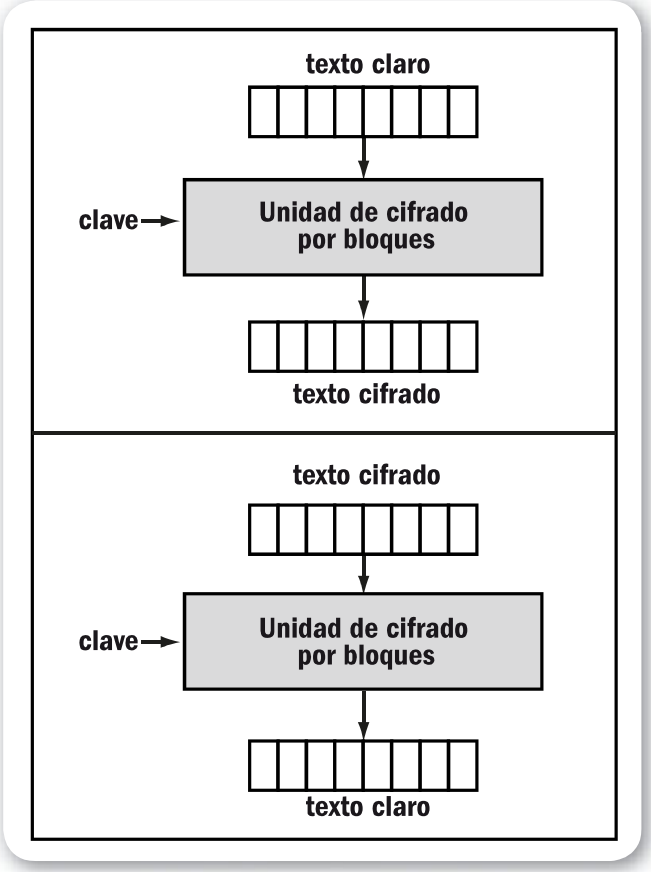


Figura 6. Esquema general de cifrado de bloque. Procesando bloques hay alta difusión de elementos y alta dificultad para introducir bloques extraños sin ser detectados por chequeos de integridad.

Para procesar los bloques de un algoritmo se utilizan distintos modos de operación, cuyos principales esquemas son:

- **ECB** (*Electronic Code Book*): cada bloque se cifra por separado con la misma clave, con la desventaja de que un bloque determinado corresponde siempre al mismo criptograma, pero no propaga errores más allá de un bloque. Este es el modo más elemental de operación.

EN EL CIFRADO DE BLOQUE, UN ERROR EN UN SOLO BIT AFECTA A TODO EL BLOQUE

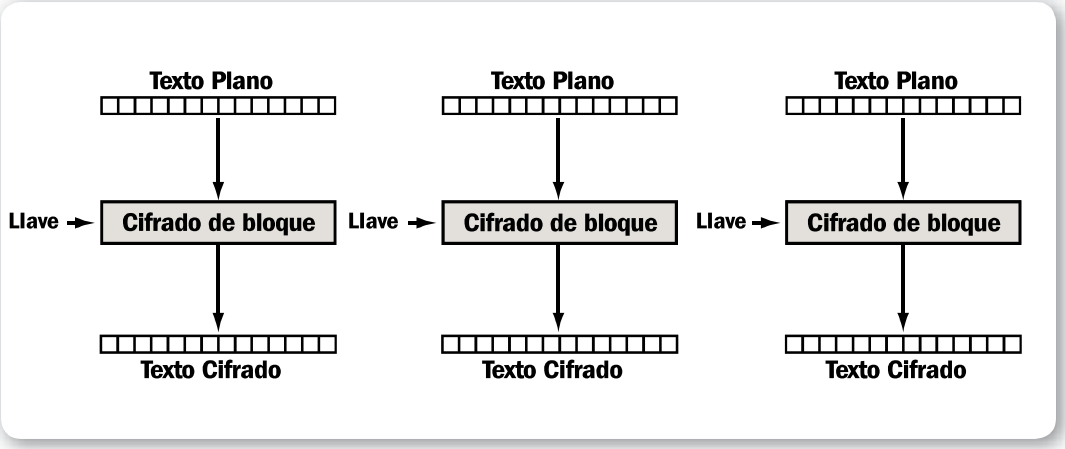


Figura 7. En esta imagen podemos observar el modo de operación más básico, denominado **ECB**.

- **CBC** (*Cipher Block Chaining*): a cada bloque se le aplica un **XOR** con el bloque anterior antes de cifrarlo, y al primero un vector de inicialización (**IV**).
  Así, un bloque repetido resulta en dos criptogramas distintos, pero hace que todo error se propague.

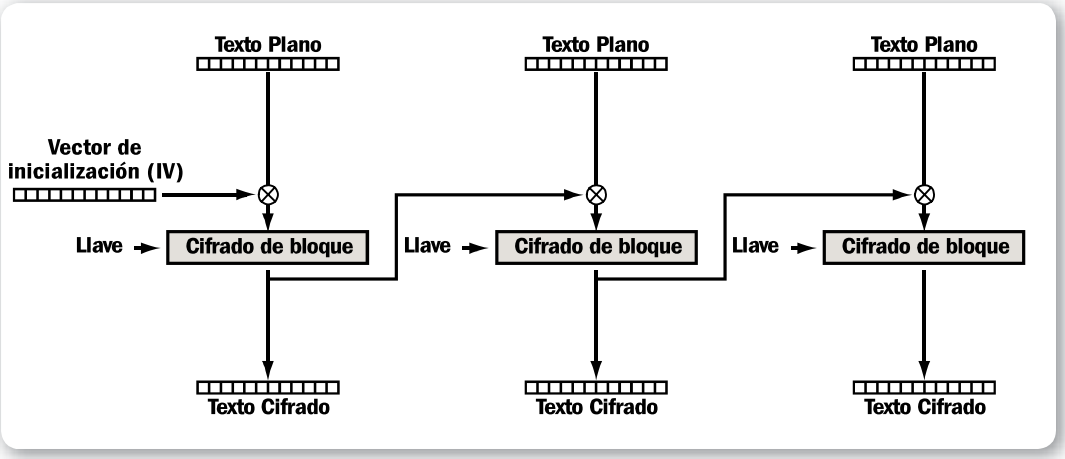


Figura 8. En el modo de operación **CBC** se evita el problema de repetición de bloques de ECB.

- **CFB** (*Cipher FeedBack*): similar a CBC pero el texto plano se aplica después del bloque de cifrado. Funciona como cifrado de flujo auto sincronizado usando un registro de desplazamiento en **IV**, y los errores se propagan en todo el período de la secuencia cifrante. Este método busca complejizar un poco más la combinación de bloques.

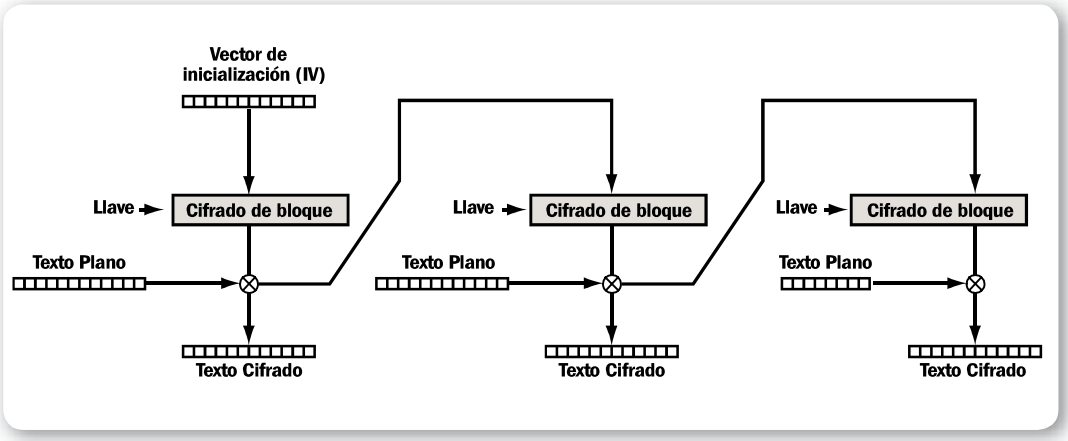


Figura 9. En el modo de operación **CFB** el texto plano se hace operar luego del bloque de cifrado.

- **OFB** (*Output FeedBack*): similar a CFB pero tomando la salida para la próxima operación antes de utilizar el texto plano, permitiendo precalcular la secuencia cifrante. En este caso, un error en un bit afecta solo al bit de su misma ubicación (permite corrección de errores).

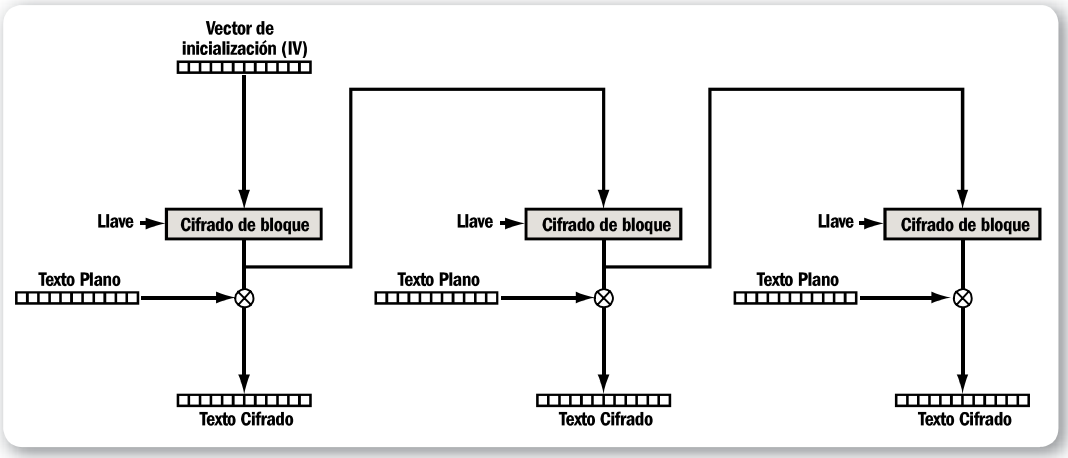


Figura 10. El modo de operación **OFB** permite precalcular la secuencia cifrante.

- **CTR** (*Counter Mode*): utiliza un contador para generar el siguiente bloque del flujo de claves partiendo de un valor arbitrario (llamado **nonce**).
  Actualmente es el modo más recomendado y utilizado, aunque los métodos anteriores se continúan estudiando por cuestiones didácticas.

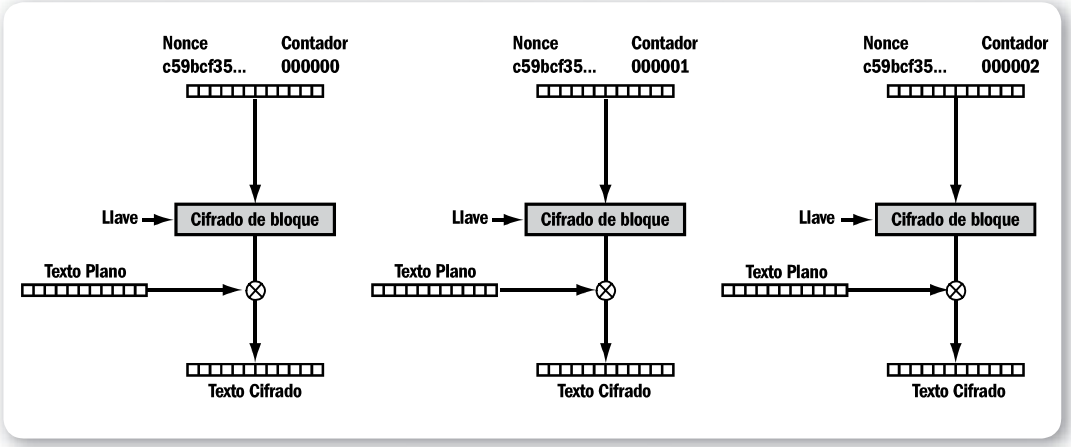


Figura 11. El modo de operación **CTR** es el más utilizado hoy en día.

Para aumentar la seguridad del cifrado de bloques se aplican técnicas que modifican su diseño original. Las principales son el **cifrado múltiple**, donde se aplica el mismo cifrado más de una vez, y el **key whitening** (blanqueamiento de clave), que consiste en combinar datos con partes de la clave antes o después de las vueltas de cifrado.

Modo	Paralelizable	
	Cifrado	Descifrado
ECB	Sí	Sí
CBC	No	Sí
CFB	No	Sí
OFB	No	No
CTR	Sí	Sí

Figura 12. Algunos modos permiten operar en paralelo, lo que resulta siempre una ventaja para el rendimiento.

RED DE FEISTEL

Es un método de cifrado en bloque que debe su nombre al criptógrafo de IBM Horst Feistel. Las operaciones de cifrado y descifrado son idénticas, requiriendo solo invertir el orden de las subclaves. Trabaja con un número dado de vueltas, realizando las mismas operaciones en cada una. Divide la entrada en dos partes y las procesa separadas, de a una mitad por vez.

La **propagación y corrección de errores** es un tema discutido ya que puede proveer información sobre la integridad a un atacante, pero si eso es lo que se busca, deben aplicarse códigos de corrección de errores al criptograma antes de transmitirlo.

También es posible obtener **cifrado autenticado** a través de otros modos de operación, aunque cabe aclarar que los más conocidos y utilizados están patentados.

Algoritmos simétricos

Como hemos dicho, los algoritmos simétricos se caracterizan por utilizar la **misma clave** para el **cifrado** y el **descifrado**, aunque el proceso que realizan puede variar entre ambas operaciones. A continuación, veremos algunos de los más importantes.

DES y 3DES

En 1973, la **NBS** (*National Bureau of Standards*) –que era la vieja denominación del **NIST** (*National Institute of Standards and Technology*)– llamó a concurso público para buscar un algoritmo criptográfico estándar a nivel nacional.

Un año más tarde, la **NSA** (*National Security Agency*) declaró desierto el concurso y publicó especificaciones para otro, que fue ganado por el algoritmo **Lucifer**.

En el año 1976 se adoptó **DES** (*Data Encryption Standard*) como estándar y se autorizó su uso en comunicaciones gubernamentales no clasificadas.

DES RECERTIFICADO

En 1987 y 1993 el **NIST** recertificó a DES, pero en 1997 no volvió a certificarlo y llamó a un concurso público para buscar un nuevo estándar mundial de cifrado, que denominaría **AES**. Entre los años 1997 y 1999 el DES se enfrentó a tres desafíos conocidos como **DES Challenge**, impulsados por la empresa RSA.

Lucifer, desarrollado por Horst Feistel para IBM, fue el primero en utilizar la **red Feistel**. Para el estándar, la NSA limitó su clave de 128 bits a 64 bits, de los cuales solo 56 son efectivos ya que los ocho restantes se utilizan como paridad. El espacio de claves resultó de 2^{56}, es decir, más de 72.000 billones de claves. Si bien la NSA argumentó que el recorte facilitaría el diseño de chips para la época, lo cierto es que se trató de una medida política que les permitiría a ellos criptoanalizar datos en un tiempo razonable.

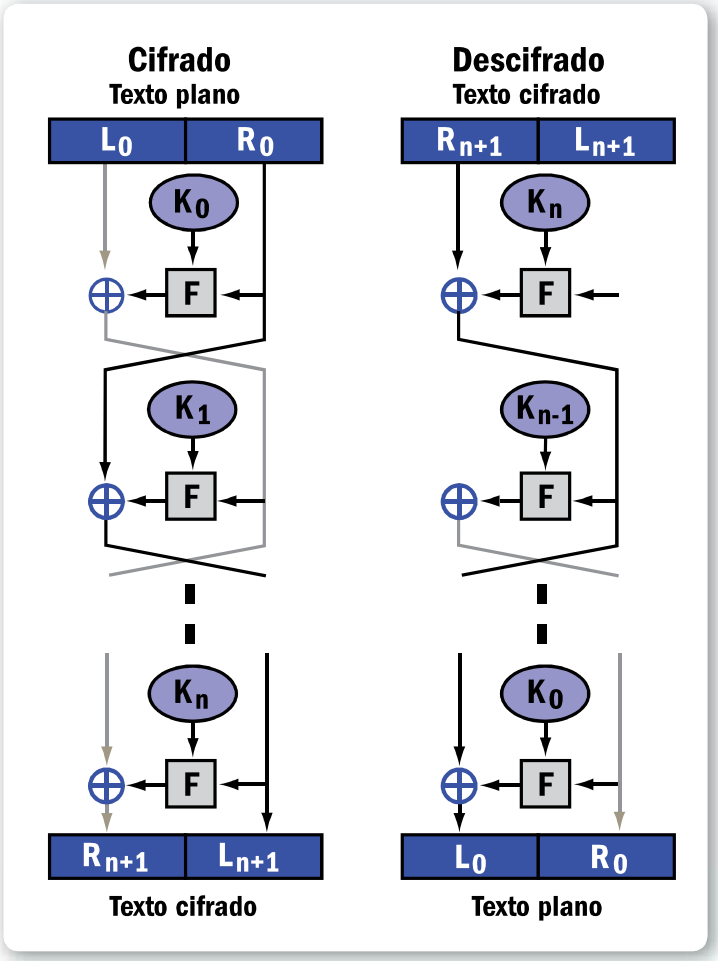


Figura 13. En una **red de Feistel**, el esquema de cifrado y descifrado es el mismo. En cada vuelta se procesa una mitad y la otra es entrada de la siguiente.

Los bloques centrales aplican **técnicas de permutación y sustitución**, y para poder operar los bloques en las funciones **XOR** se realizan permutaciones con expansión y compresión a fin de igualar la cantidad de bits de mensaje y clave.

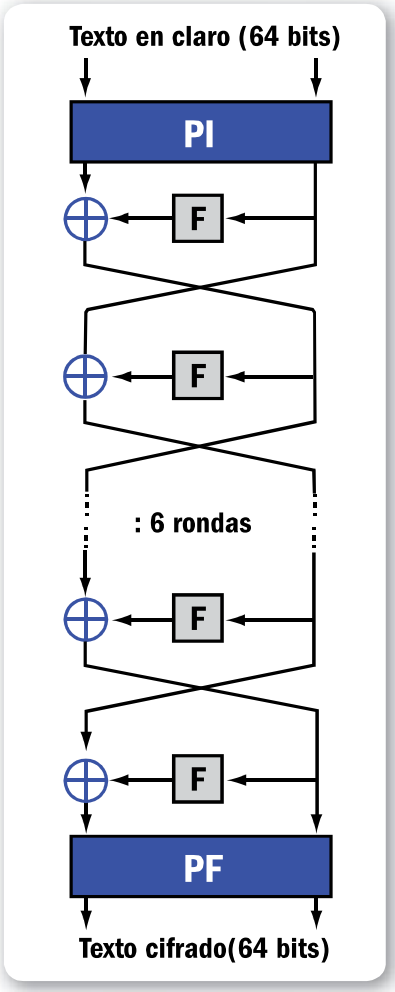


Figura 14. Proceso general de **cifrado en DES**: permutación inicial, 16 vueltas y permutación final. Trabaja alternadamente con cada sub-bloque de 32 bits a partir de entradas de 64 bits.

Una mitad realiza una **permutación con expansión** de 32 a 48 bits y el resultado se dividen en bloques de 6 bits para ingresar a las ocho **Cajas-S** (**cajas de sustitución**) que son **matrices predefinidas** que transforman los 6 bits a 4 bits, reduciendo los 48 totales a 32.

DES CHALLENGE

DES Challenge fue un desafío que proponía quebrar la seguridad de DES. Existieron tres instancias, la primera (DES Challenge I) fue en 1997 y se logró romper la clave en 96 días con 80.000 PCs distribuidas en internet evaluando 7.000 millones de claves por segundo. Para esto, se debió recorrer el 25% del espacio de claves.

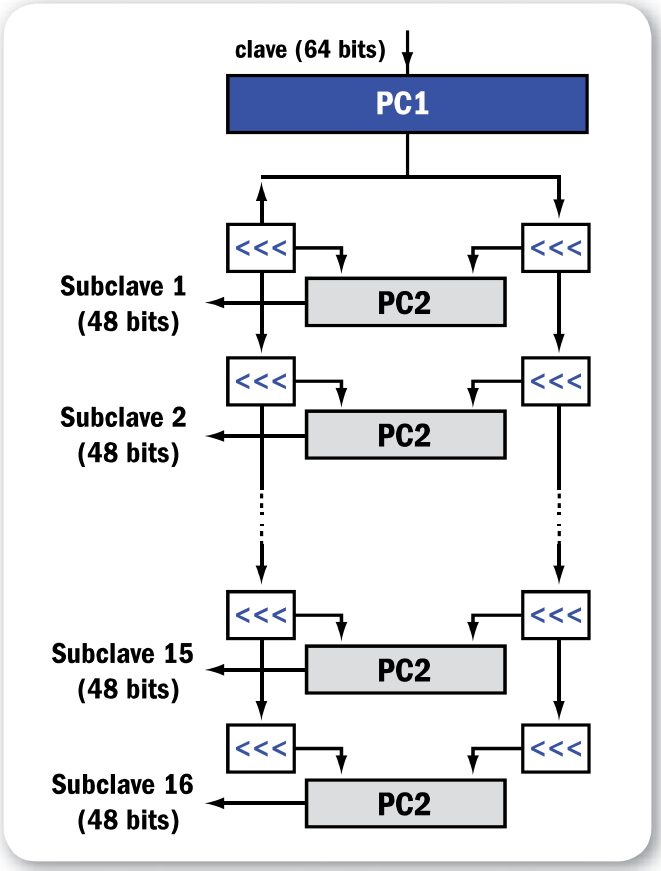


Figura 15. En la generación de subclaves se hace una permutación con compresión de 64 bits (clave original) a 48 bits, para poder entrar a una **XOR** con la otra mitad expandida.

Luego, el bloque de 32 bits llega a una permutación y, finalmente, a una nueva **XOR** junto con la mitad no procesada en la vuelta. Para la próxima iteración, el proceso se invierte y en total se realiza 16 veces hasta obtener el criptograma. Al ser una red Feistel, el proceso de descifrado es el mismo, pero invirtiendo el orden en el que se ingresan las subclaves.

DES CHALLENGE II

El segundo desafío (1998) se dividió en dos. En el primero (enero) se rompió la clave en 39 días con un ataque distribuido por **distributed.net**, a 34.000 millones de claves/seg. En el segundo (julio), la **EFF** (**Electronic Frontier Foundation**) utilizó **DES Cracker** (un hardware especializado) y lo quebró en 56 horas a 90.000 millones de claves/seg.

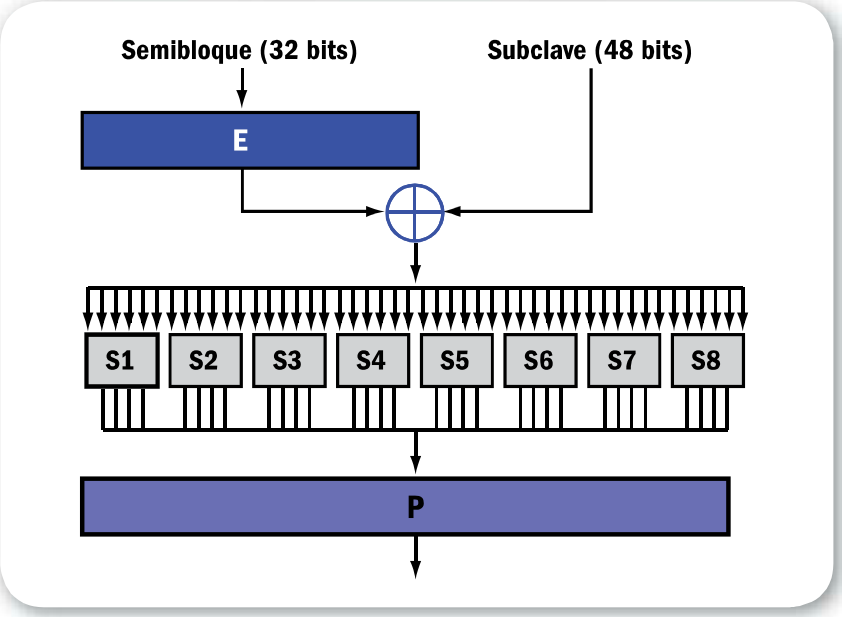


Figura 16. Esquema de la función **f**, donde se opera sobre medio bloque (32 bits) a través de cuatro etapas: expansión (**E**), mezcla (**XOR**), sustitución (**Si**) y permutación (**P**).

Una variante del DES fue el **3DES**, desarrollado por IBM en 1998 y pensado para aprovechar la gran propagación que había tenido DES, para que tanto el hardware como el software existente que lo implementaban no debieran sufrir grandes modificaciones para lograr un mayor nivel de seguridad.

3DES opera aplicando tres veces el DES. Si bien podría suponerse que aplicándolo dos veces se duplicaría el tamaño de la clave, esto no es así, sino que solo aumenta en 1 bit su longitud efectiva. Además, sería susceptible a un ataque de encuentro a medio camino (**meet-in-the-middle**).

Dependiendo de cómo se realice el proceso, surgen las implementaciones comunes:

- **DES-EEE3**: se cifra tres veces con una clave diferente cada vez (el más seguro).
- **DES-EDE3**: primero se aplica la operación de cifrado, luego se aplica la de descifrado y finalmente la de cifrado otra vez, todas con distintas claves.
- **DES-EEE2/DES-EDE2**: similares a las anteriores, pero usando la misma clave en el primero y último paso.

Longitud de clave	Tiempo para romper la clave
40 bits	2 segundos
48 bits	9 minutos
56 bits	40 horas
64 bits	14 meses
72 bits	305 años
80 bits	78.250 (216) años
96 bits	5.127.160.311 (232) años
112 bits	336.013.578.167.538 (248) años
128 bits	22.020.985.858.787.784.059 (264) años

Figura 17. Basados en la potencia de cómputo que en 1999 logró romper DES, podemos analizar los valores de claves y el tiempo requerido para romperlas por fuerza bruta. Se ve que su crecimiento es exponencial.

Normalmente, se utilizan los métodos de dos claves para conseguir una clave efectiva de 112 bits, aunque la real es de 192 bits (3x64 bits). 3DES no llega a considerarse lo que se llama un **cifrado múltiple** ya que todas las subclases no son independientes.

3DES NO ES UN CIFRADO MÚLTIPLE YA QUE LAS SUBCLASES NO SON TODAS INDEPENDIENTES

Esto es porque DES no es lo que en matemática se define como un **grupo**, que sí sería si al cifrar un mensaje con una clave **k1** y su resultado cifrarlo con una clave distinta **k2**, existiera una única clave **k3** con la cual se pudiera obtener el mismo resultado.

Es decir, si en DES se cifra dos veces el mismo bloque con dos claves distintas, el tamaño efectivo de la clave también crece.

DES CHALLENGE III

En el último desafío (1999) se unieron **DES Cracker** y **distributed.net** con 100.000 PC conectadas a internet y lograron romper la clave en 22 horas, evaluando 245.000 millones de claves por segundo tras recorrer el 22% del espacio de claves. Luego de esto, no volvió a realizarse otro desafío.

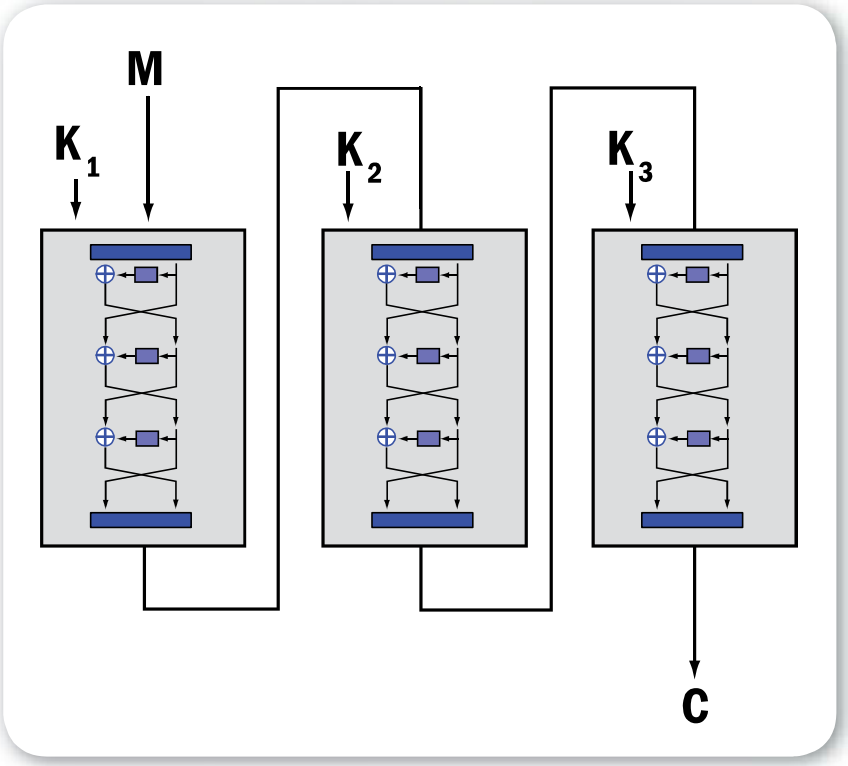


Figura 18. Esquema de **3DES** que aún se utiliza en ciertas aplicaciones de uso comercial, aunque está siendo reemplazado por haber sucumbido ante la potencia de cálculo actual.

IDEA

IDEA (*International Data Encryption Algorithm*) fue diseñado por Xuejia Lai y James Massey en 1991. No utiliza red de Feistel en búsqueda de una mayor eficiencia ya que por cada vuelta se modifican todos los bits del bloque y no solamente los correspondientes a una mitad. Opera con bloques de 64 bits pero utiliza clave de 128 bits efectivos, lo que garantiza un espacio de claves de $\mathbf{2^{128}}$: más de 3,4 cuatrillones de claves.

Las operaciones que utiliza son: **XOR**, **suma en módulo 65.536** y **multiplicaciones módulo 65.537** (un número primo que asegura el inverso multiplicativo).

Estas operaciones modulares restringen la cantidad de elementos a utilizar y cuando se alcanza ese valor máximo, se vuelve a empezar. Esto fue fundamental para la implementación en **hardware de 16 bits**, ya que es la máxima cantidad de elementos que pueden manejar dispositivos con esta arquitectura.

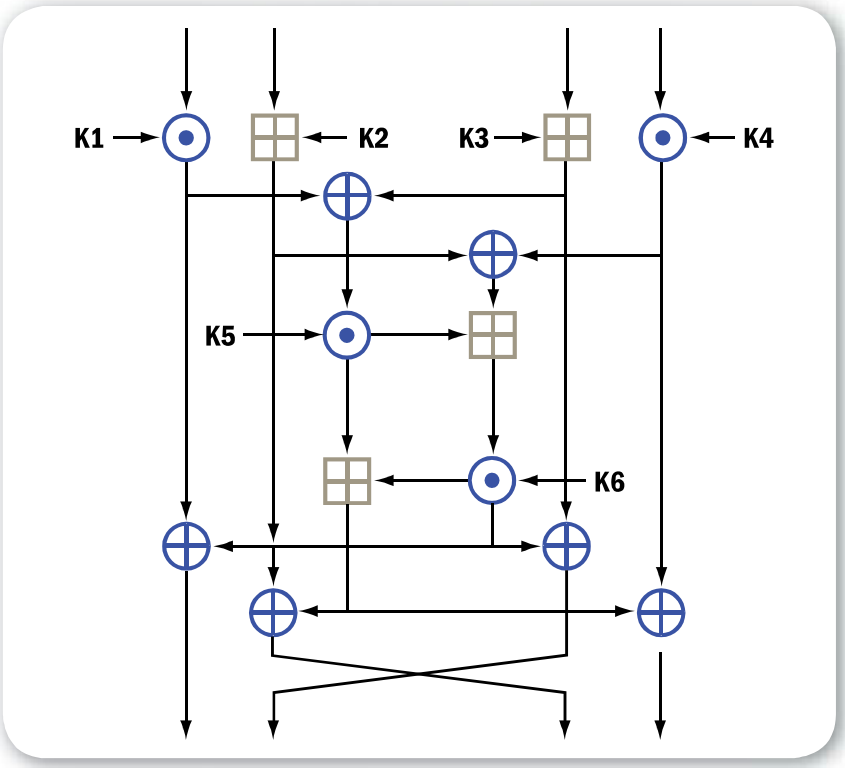


Figura 19. Esquema de una vuelta en **IDEA**, que realiza ocho vueltas en total, más media vuelta al final del proceso.

Para operar, al bloque de entrada se lo divide en cuatro de 16 bits y se generan **52 sub-claves de 16 bits**, de las cuales se utilizan seis por vuelta. Los cuatro subbloques se procesan con las claves en ocho vueltas y luego se aplica la transformación final con cuatro subclaves más que invierte la operación inicial; se obtiene, así, el criptograma de salida.

Computacionalmente se lo considera seguro, ya que el ataque por fuerza bruta resulta muy dificultoso dada la gran cantidad de claves. Es fuerte ante el **criptoanálisis diferencial** e inmune

LA IDEA DE IDEA

IDEA surgió como mejora del algoritmo **IPES** (**Improved Proposed Encryption Standard**), que derivó de **PES** (**Proposed Encryption Standard**). Es libre para uso no comercial, y fue licenciado y registrado como marca por la compañía **MediaCrypt**, aunque sus patentes vencieron en 2011. Al ser europeo, no sufrió las limitaciones de exportación de los algoritmos americanos, lo que motivó su adopción.

bajo ciertos supuestos. Si bien no se han reportado debilidades ante el **criptoanálisis lineal**, se han encontrado algunas **claves débiles** (Joan Daemen, 1992) que son evitables al momento de la implementación. En efecto, sus mejores criptoanálisis públicos descubiertos son:

- **Ataque de colisión** que requiere 2^{24} textos planos seleccionados para romper cinco rondas con una complejidad de 2^{126} (Demirci, 2003).
- **Ataque lineal-diferencial de alto orden** que requiere 2^{64} textos planos seleccionados para romper seis vueltas con una complejidad de 2^{126}.8 operaciones (Biham, 2007).

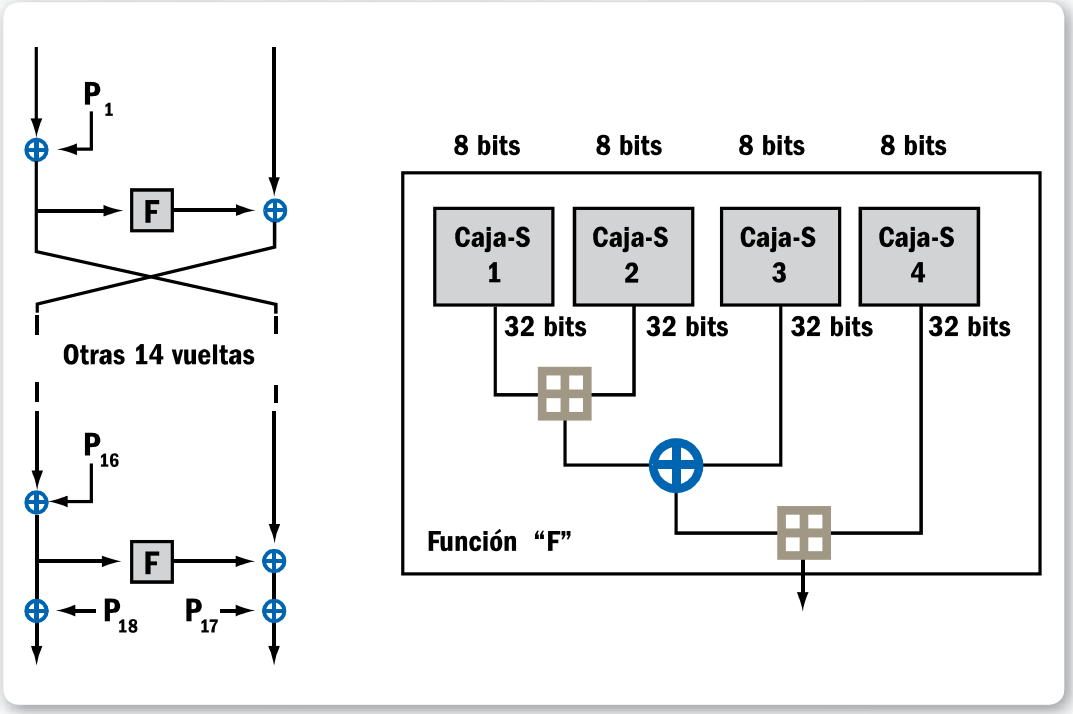


Figura 20. Esquema del algoritmo **Blowfish** (Bruce Schneier, 1993). Trabaja con bloques de 64 bits y claves desde 32 a 448 bits. Realiza 16 vueltas tipo **Feistel** y utiliza **Cajas-S**. **Pi** son los vectores de subclaves.

AES

En 1997, el NIST llamó nuevamente a concurso en un proceso que demoró tres años y del que surgió, en 2001, un nuevo algoritmo para ser utilizado como estándar: **AES**.

Figura 21. Sin contar la expansión de la clave, podemos determinar **tres etapas** en la operación de **AES**, cada una con sus distintas funciones internas.

En rigor, AES no es Rijndael sino una variación (similar a DES y Lucifer) ya que, por ejemplo, usa tamaño de bloque fijo de 128 bits y tamaños de claves de 128, 192 o 256 bits; y Rijndael soporta claves múltiplo de 32 bits (con mínimo de 128 y máximo de 256). No usa una red Feistel sino una **red de sustitución-permutación**, aunque aplica **cajas-S** en una de las etapas de operación, denominada **capa no lineal**.

LA FINAL DEL CONCURSO

En la ronda final de la competencia en busca de un nuevo estándar quedaron cinco algoritmos, en el siguiente orden: **Rijndael**, **Serpent**, **Twofish**, **RC6** y **MARS**. El ganador fue el desarrollado por los ingenieros y criptógrafos belgas Joan Daemen y Vincent Rijmen, de donde proviene el nombre del algoritmo **Rijndael**, como palabra compuesta entre sus nombres.

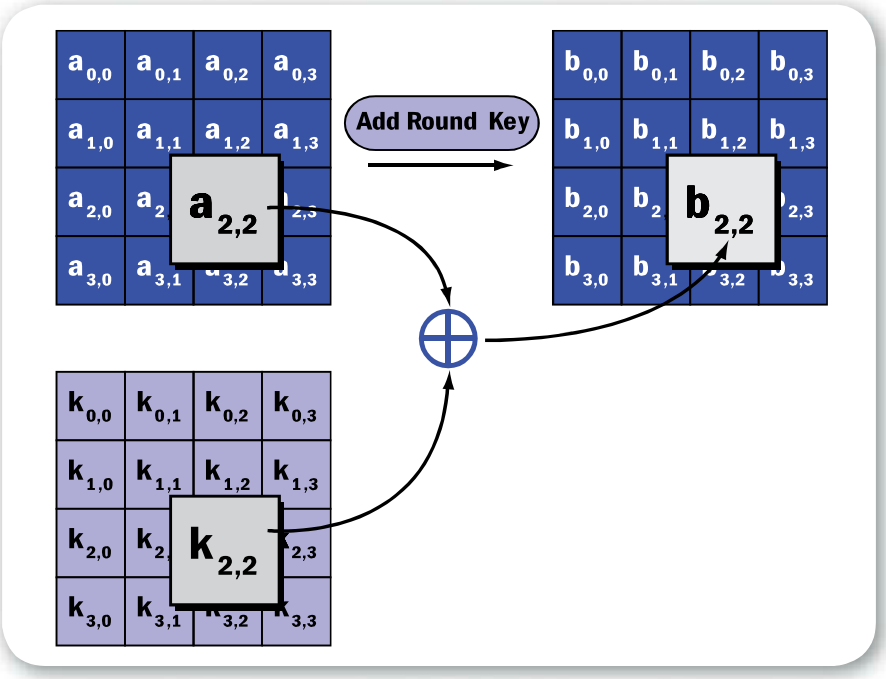


Figura 22. En la función **AddRoundKey** se realiza un **XOR** entre cada byte del **state** y uno de la **subclave**.

AES es relativamente rápido y fácil de implementar, y su proceso de cifrado es diferente al de descifrado. Su unidad básica de tratamiento es el byte y opera en una matriz de **4×4 bytes**, llamada **state**.
Sus funciones internas son las siguientes:

- **AddRoundKey**: combinación de un byte con una clave derivada.
- **SubBytes**: sustitución no lineal.
- **ShiftRows**: transposición por rotación cíclica.
- **MixColumns**: transformación lineal sobre las columnas.
- **SubBytes**: sustitución de bits.
- **ShiftRows**: rotación cíclica de filas.

EL TÍO BRUCE

Una figura destacable en el mundo de la criptografía y la seguridad informática es, sin dudas, Bruce Schneier, un criptógrafo creador de varios algoritmos importantes como Blowfish y Twofish, y escritor de varios libros de referencia, entre los que se encuentra **Criptografía Aplicada**. Schneier es autor de varias frases clásicas de la seguridad y en la actualidad se dedica a difundir sobre temas de privacidad.

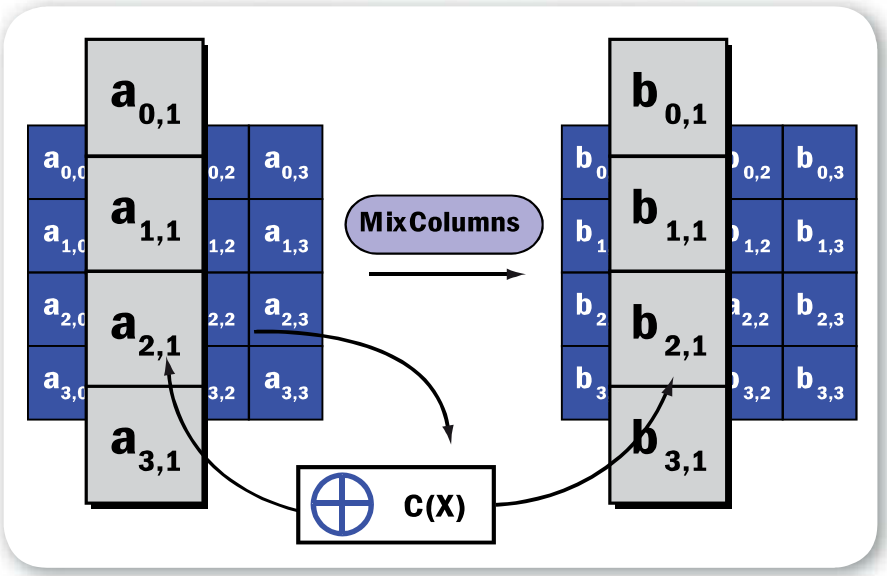


Figura 23. La función **MixColumns** es, básicamente, un producto matricial, basada en multiplicar cada columna del **state** con un polinomio fijo **c(x)**.

Problemáticas inherentes

El cifrado simétrico, pese a su alto rendimiento con claves relativamente pequeñas, no es precisamente ideal para todo tipo de entorno y aplicación, pues cuenta con algunas limitaciones que se derivan de la naturaleza de los algoritmos y no son problemas de alguno en especial.

PESE A SU RENDIMIENTO CON CLAVES PEQUEÑAS, EL CIFRADO SIMÉTRICO TIENE LIMITACIONES

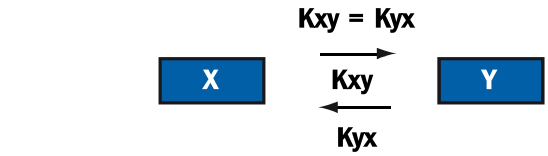
- **La gestión de claves es compleja**: la cantidad de claves crece en proporción a $\mathbf{n^2}$ (para **n** grande), con lo que se torna difícil de manejar cuando se requieren muchas.
- **La distribución de claves no puede asegurarse**: no provee mecanismos para enviar claves de manera segura a través de un medio inseguro.
- **Sin firma digital**: no permite firmar digitalmente en el sentido estricto del concepto, aunque sí permite la autenticación parcial.

Algoritmo	Tamaño de Clave (Bits)	Tamaño de Bloque (Bits)	Nro. de Etapas
DES	56	64	16
3DES	112 0 168	64	48
AES	128,192 o 256	128	10,12 o 14
IDEA	128	64	8
Blowfish	Variable hasta 448	64	16
RC5	Variable hasta 2048	64	Variable hasta 256

Figura 24. Comparativa de algunos de los algoritmos simétricos más comunes.

A pesar de esto, si se lo utiliza de la forma correcta se pueden aprovechar sus ventajas, pero para salvar definitivamente los problemas mencionados debemos aplicar una estrategia diferente, que se basa en el uso de los **algoritmos asimétricos**, que veremos en el **Capítulo 5**. No obstante, dichos algoritmos también tienen sus correspondientes limitaciones.

Nro. de claves N	Nro. de claves N
1	1
3	3
6	6
10	10
15	15
45	45
4950	4950
499500	499500
49995000	49995000
N	N



	A	B	C	D	E	F
A		Kab	Kac	Kad	Kae	Kaf
B	Kba		Kbc	Kbd	Kbe	Kbf
C	Kca	Kcb		Kcd	Kce	Kcf
D	Kda	Kdb	Kdc		Kde	Kdf
E	Kea	Keb	Kec	Ked		Kef
F	Kfa	Kfb	Kfc	Kfd	Kfe	

Figura 25. La cantidad de claves necesarias para un criptosistema basado en algoritmos simétricos requiere **n*(n-1)/2 claves** y tiende a **$n^2/2$**.

Criptoanálisis en cifrado simétrico

Para estudiar el criptoanálisis en general y el de los algoritmos simétricos en particular, debemos comenzar por comprender los distintos tipos de ataque según la información previa con la que se cuenta. Así, surgen cuatro casos básicos según esta clasificación general de ataques:

- **Sólo texto cifrado**: solo se tiene acceso a una cierta cantidad de textos cifrados.
- **Texto plano conocido**: solo se conoce una cierta cantidad de criptogramas de los que se conocen los textos planos.
- **Texto plano elegido**: se pueden conocer los criptogramas de una cierta cantidad de textos planos a elección. En esta categoría pueden distinguirse dos formas de ataque: **por lotes** (**batch**), donde todos los textos planos pueden obtenerse antes del cifrado, y el **adaptativo**, donde la elección de cada texto plano se basa en el conocimiento del cifrado previo. Existe un caso particular donde se puede conocer un par de criptogramas cifrados con claves diferentes desconocidas, pero que tienen una relación conocida entre ambas.
- **Texto cifrado elegido**: se pueden conocer los textos planos de una cierta cantidad de criptogramas a elección. También podemos considerar las formas de lote y adaptativo, como en el caso anterior, y análogamente, el **ataque de claves relacionadas**.

En otro enfoque de análisis, podemos decir que la alinealidad de los algoritmos simétricos hace que el único ataque naturalmente factible sea el de **fuerza bruta**, que implica **probar todas las combinaciones**

ATAQUES TEÓRICOS Y PRÁCTICOS

Podemos distinguir los ataques prácticos, asociados fundamentalmente a las **implementaciones**, de los teóricos, que usualmente se refieren a una debilidad bajo ciertas **consideraciones matemáticas** (reducción de la complejidad) y están orientados a reducir el tamaño efectivo de la clave, lo que muchas veces en la práctica no representa un alto riesgo.

posibles. La idea del criptoanálisis es reducir este problema desde dos perspectivas: el **aumento de la capacidad de procesamiento** (probar más cantidad de claves en menos tiempo) y la **reducción de la complejidad** a partir de debilidades propias del algoritmo.

En 1998, el criptógrafo Lars Knudsen realizó una clasificación particular para cifrado de bloques y propuso las siguientes categorías en función de la **calidad y la cantidad de información** hallada:

- **Ruptura total**: en este caso, el atacante obtiene la clave secreta directamente.
- **Deducción global**: aquí, el atacante halla un algoritmo equivalente para el cifrado y descifrado de mensajes, pero sin obtener la clave.
- **Deducción local**: en este caso, el atacante puede obtener mensajes en texto plano o cifrados adicionales a los que ya conocía.
- **Deducción de información**: basado en la teoría de Shannon, el atacante obtiene información que antes desconocía.
- **Distinción del algoritmo**: en este caso, el atacante puede discernir entre la información cifrada aleatoriamente.

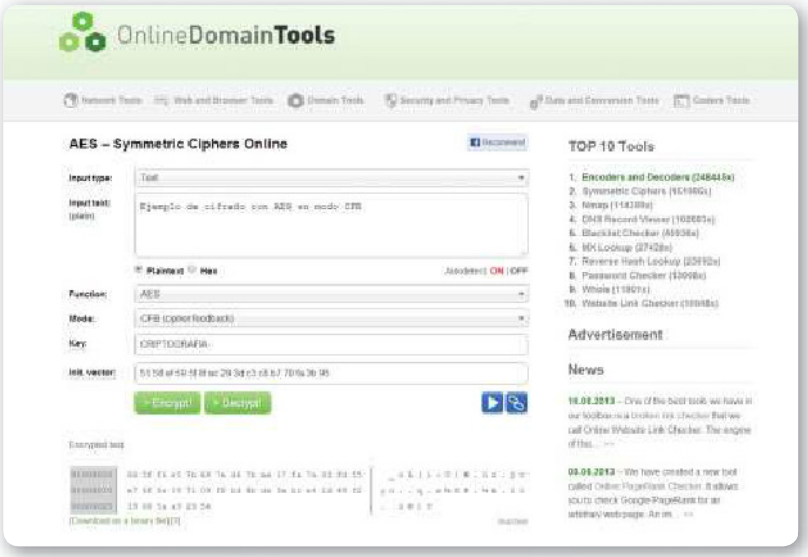


Figura 26. Existen sitios web que permiten probar los principales algoritmos en distintos modos de operación, como **http://aes.online-domain-tools.com**.

No obstante, existen algunos ataques específicos para algoritmos de bloque (muchos de ellos también aplicables al cifrado de flujo), entre los que se encuentran:

- **Criptoanálisis diferencial**: es una forma general de análisis que estudia cómo las diferencias en una entrada pueden afectar un resultado a la salida. Para esto, se aprovecha el conocimiento de las redes de transformaciones cuando presentan un comportamiento no aleatorio. El desarrollo se le atribuye a Eli Biham y Adi Shamir en los 80, aunque Don Coppersmith de IBM reveló, en 1994, que la técnica había sido descubierta en 1974, durante el diseño de Lucifer, y bautizada **ataque tickle**. Esta técnica fue generalizada por Lars Knudsen en 1994 al proponer el **criptoanálisis diferencial de alto orden**, donde en vez de analizar diferencias entre textos, se consideran las diferencias entre las diferencias, obteniendo así un modelo más abstracto. Algunas variantes son el **criptoanálisis diferencial imposible** y el **improbable**. Al mismo tiempo, un caso particular es el **ataque boomerang**, propuesto por David Wagner en 1999, que permite que las diferencias sean analizadas solo sobre una parte del cifrado; y el **ataque sándwich**, que agrega una capa para reducir los casos de estudio.

Figura 27. Lars Knudsen es un criptógrafo danés que contribuyó largamente al tema mediante el desarrollo de técnicas de criptoanálisis.

- **Criptoanálisis integral**: técnica especialmente aplicable al cifrado de bloques basados en redes de sustitución y permutación creada por Lars Knudsen, también llamada **ataque Square**. Stefan Lucks lo generalizó en lo que se dio en llamar **ataque de saturación**. Esta técnica utiliza juegos de textos planos escogidos, de los cuales una parte puede mantenerse constante y otra variar, a diferencia de lo que ocurre con el criptoanálisis diferencial, donde la operación XOR es fija.
- **Criptoanálisis lineal**: forma general de análisis basada en encontrar aproximaciones afines a las acciones de cifrado. Es uno de los dos ataques más extendidos en cifrado de bloques, junto al diferencial, y se lo atribuye a Mitsuru Matsui. La técnica lineal cuenta con dos etapas: la primera es construir las ecuaciones lineales relacionadas al texto plano, criptograma y bits de la

llave que tienen un alto sesgo (o sea, cuyas probabilidades de aparición son cercanas a 0 ó a 1); la segunda etapa es utilizar las ecuaciones en conjunción con pares texto plano-criptogramas para derivar los bits de la clave. Con el correr de los años, se propusieron variaciones que incluían múltiples aproximaciones lineales o inclusión de expresiones no lineales, lo que llevó al desarrollo del **criptoanálisis de partición**. Esta generalización fue propuesta por Carlo Harpes en 1995 y se basa en la división de posibles textos planos y criptogramas en particiones eficientemente computables, donde la distribución del criptograma es significantemente no uniforme cuando el texto plano es seleccionado uniformemente de un bloque dado de la partición. Un tipo específico de criptoanálisis de partición es el llamado **criptoanálisis de módulo n**, que utiliza las clases de congruencia de módulo entero para las particiones y fue sugerido en 1999 por John Kelsey, Bruce Schneier y David Wagner.

Figura 28. Eli Biham es un criptógrafo israelí que creó el criptoanálisis diferencial junto a Adi Shamir.

- **Meet-in-the-middle**: creado por Whitfield Diffie y Martin Hellman en 1977, esta técnica se basa en modelar el diseño de un sistema como secuencia de procesos con el fin de que encontrar la descripción de esos procesos, dados los valores de la entrada (**E**) y la salida (**S**) de

¿ROMPER TODO?

Podría suponerse que no hay manera de evitar que un algoritmo sea roto tarde o temprano, pero no debemos preocuparnos por el momento ni por la imposibilidad de que el criptoanálisis pueda tener resultados concretos, sino por el hecho de que la criptografía, utilizada de la manera correcta, puede proveer prácticamente en todos los casos los resultados esperados.

forma que exista un flujo intermedio (**I**) definible entre ellos. Es un típico caso de situación de **compromiso espacio-tiempo**, ya que el tiempo se reduce a costa de la necesidad de almacenamiento (los resultados intermedios).

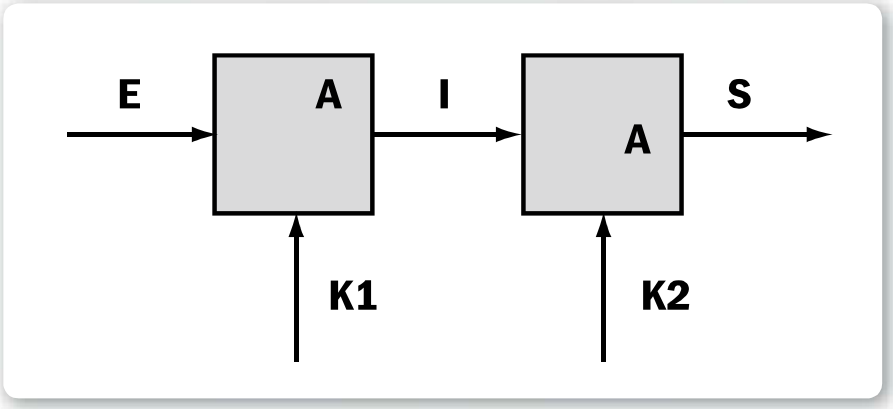


Figura 29. Esquema de la técnica criptoanalítica **Meet-in-the-middle**, que no debe confundirse con el **Man-in-the-middle** utilizado en redes de computadoras.

- **Criptoanálisis estadístico**: se basa en las estadísticas obtenidas de la visualización de la información cifrada en relación a la descifrada. Donald Davies propuso esta técnica en 1987 y fue mejorada en 1994 por Eli Biham y Alex Biryukov. En 1998, Thomas Pornin propuso un método para mejorar la resistencia de los cifrados a esta técnica.
- **Ataque de deslizamiento**: técnica diseñada para lidiar con la idea de que incluso un cifrado débil puede ser robusto si se incrementa la cantidad de vueltas (lo que sí podría evitar un análisis diferencial), haciendo que ese número pase a ser irrelevante. En lugar de observar los aspectos de aleatoriedad de los datos del cifrado, se analiza el ciclo de la clave, con lo que el único requerimiento es que el cifrado pueda dividirse en vueltas que atraviesen una función idéntica. Fue descrito por David Wagner y bautizado por Bruce Schneier en 1999, aunque la idea original proviene de las publicaciones de Edna Grossman y Bryant Tuckerman de IBM en 1977.
- **Ataque XSL** (*eXtended Sparse Linearisation*): está basado en la modelización del algoritmo en términos de sistemas de ecuaciones cuadráticas que proveen información al ser

resueltos. Es notable por requerir cantidades realistas de texto plano conocido, a diferencia de otros tipos de ataques, como el diferencial o el lineal.

En función de estos ataques, podríamos pensar en querer mejorar los algoritmos de cifrado. La forma trivial en algoritmos simétricos es aumentar el tamaño de la clave, aunque no es una solución académicamente aceptada ya que lo que se busca es mejorarlos desde el punto de vista conceptual y matemático.

RESUMEN

En este capítulo estudiamos los algoritmos simétricos, que utilizan la misma clave para el cifrado y el descifrado. Pueden ser de flujo, cuando procesan de a un bit; y de bloques, cuando procesan conjuntos de bits. Vimos los más importantes de cada tipo: RC4, A5 y SEAL entre los de flujo; y DES, 3DES, AES, IDEA y Blowfish, entre los de bloque. Profundizamos en DES por su importancia histórica y su estándar sucesor: AES. Analizamos algunas problemáticas propias del cifrado simétrico y, finalmente, vimos algunas técnicas de criptoanálisis.

Actividades

TEST DE AUTOEVALUACIÓN

1. ¿Qué diferencia al cifrado de bloque del de flujo?
2. ¿Qué persiguen los postulados de Golomb?
3. ¿Qué es una red de Feistel?
4. ¿Por qué dejó de utilizarse el algoritmo DES como estándar?
5. ¿Qué son IDEA y Blowfish?
6. ¿Por qué no es recomendable el uso del modo ECB en el tratamiento de bloques?
7. ¿Cuáles son los tres principales problemas inherentes al cifrado simétrico?

EJERCICIOS PRÁCTICOS

1. Busque la lista de algoritmos de bloque más utilizados en la actualidad, con sus características principales.
2. Busque la lista de algoritmos de flujo más utilizados.
3. Analice el rendimiento de distintos algoritmos aplicados al mismo archivo mediante un software de cifrado.
4. Busque herramientas de software para realizar criptoanálisis.
5. Utilice alguna herramienta online para cifrar y descifrar un texto.

PROFESOR EN LÍNEA

Si tiene alguna consulta técnica relacionada con el contenido, puede contactarse con nuestros expertos: profesor@redusers.com

*05

Cifrado asimétrico

El cifrado asimétrico se caracteriza por utilizar una clave diferente para realizar el proceso de cifrado y descifrado. Además, el mecanismo de operación es bastante diferente al de los algoritmos simétricos, ya que no se basa en cambiar bits sino en operaciones matemáticas.

Algoritmos asimétricos

En los algoritmos asimétricos o de clave pública, se usan dos claves por cada parte (emisor o receptor), llamadas **pública** y **privada**. Estas claves están mutuamente relacionadas por una **función con trampa** que es fácil de resolver en un sentido y difícil en el sentido contrario.

EN LOS ALGORITMOS ASIMÉTRICOS SE USAN DOS CLAVES POR CADA PARTE: LA PÚBLICA Y LA PRIVADA

Si ciframos con una, debemos descifrar con la otra. Su seguridad está asociada a la resolución de un problema matemático de difícil solución en un tiempo razonable.

Dada su complejidad, son mucho más lentos que los simétricos, por lo que se usan para cifrar datos pequeños (como la clave secreta de un algoritmo simétrico).

Típicamente, se utilizan para el intercambio de claves y la firma digital, donde se busca autenticar y garantizar el **no repudio** (no será posible negar que se realizó una determinada acción). Si se autentican el mensaje y el emisor, hablamos de **firma digital completa**.

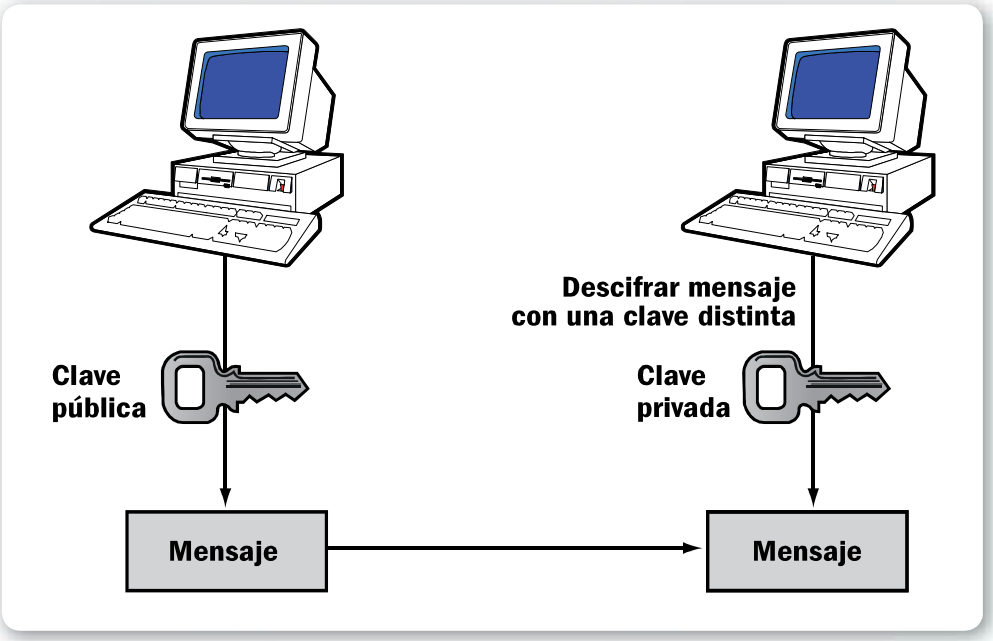


Figura 1. En un algoritmo asimétrico, el cifrado y el descifrado utilizan **claves distintas** (pública y privada).

Problemas y algoritmos

Los problemas principales de la matemática relacionados con la criptografía son:

- La **factorización de enteros**, vinculada a cuestiones de divisibilidad y primalidad, que lleva al estudio de los números primos.
- Los **logaritmos discretos**, que son a la teoría de grupos lo que los logaritmos comunes al análisis matemático, y pertenecen al álgebra abstracta (que estudia las estructuras algebraicas).

Pese a ser distintos problemas, el logaritmo discreto y la factorización de enteros comparten ciertas propiedades:

- No se puede acotar su resolución a un tiempo polinomial al tratar con números grandes (cientos de dígitos).
- Para ambos, existe un algoritmo eficiente en computadoras cuánticas.
- Un algoritmo para un problema puede ser adaptado para el otro.
- Su dificultad de cálculo ha sido base de sistemas criptográficos.

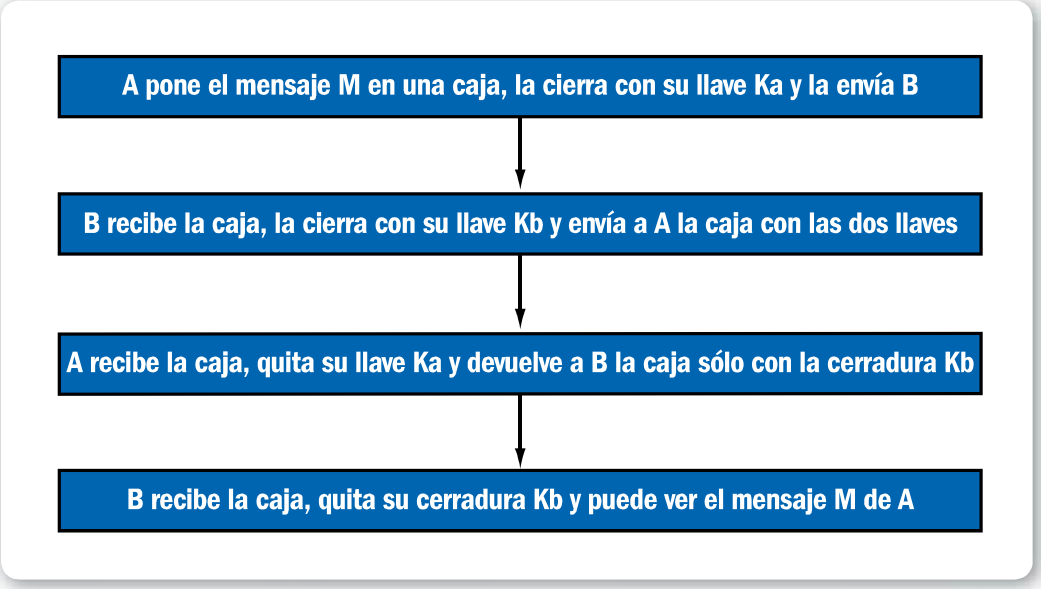


Figura 2. El protocolo del mensaje en una caja permite demostrar la efectividad del uso de una llave personal. El mensaje se mantiene **confidencial** pero **sin autenticidad** del emisor.

Factorización de enteros

La factorización de enteros busca descomponer un número no primo en divisores no triviales, que al multiplicarse resultan el número original. Si el número es muy grande (de **b bits**) producto de dos primos de aproximadamente igual tamaño, no hay algoritmo que pueda factorizarlo en tiempo polinomial, o $\mathbf{O(b^k)}$ (**k** es una constante cualquiera). Hay algoritmos más rápidos que $\mathbf{O(a^b)}$ para **a>1**, los mejores son **súperpolinomiales** pero **subexponenciales**, y de hecho el mejor tiempo asintótico de ejecución es el del algoritmo **CGCN** (*General Number Field Sieve*) o **criba general del cuerpo de números**. Esto es así para las computadoras actuales, pues para una computadora cuántica, Peter Shor describió en 1994 un algoritmo que puede resolverlo en tiempo polinomial (tarda un tiempo $\mathbf{O((\log n)^3)}$ y en un espacio $\mathbf{O(\log n)}$), pero no está al alcance de la práctica.

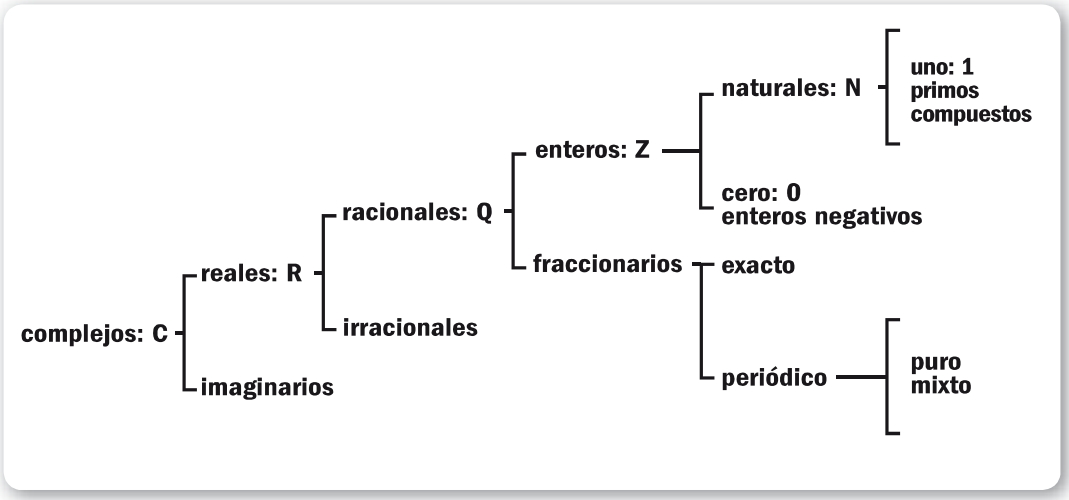


Figura 3. Clasificación global de los números. El **Teorema Fundamental de la Aritmérica** afirma que todo número entero positivo puede descomponerse de una única manera en factores primos.

TEORÍA DE GRUPOS

Es una rama del **álgebra abstracta** que estudia las estructuras algebraicas llamadas **grupos**, su clasificación, sus propiedades y sus aplicaciones. Los grupos son pilares de otras estructuras más avanzadas como los **anillos**, **cuerpos** y **espacios vectoriales**. Se define la **cardinalidad** como el orden de un grupo, en base a lo que pueden clasificarse en grupos de orden **finito** o **infinito**.

La mayor parte de los algoritmos generales de factorización (cuyo tiempo de ejecución depende solo del tamaño del entero a factorizar) se basa en el método llamado de **congruencia de cuadrados**; entre ellos se encuentran:

- **Criba racional**
- **Criba cuadrática**
- **Algoritmo de Dixon**
- **Factorización con fracciones continuas**
- **Factorización de formas cuadradas de Shanks**
- **Criba general del cuerpo de número**

En el caso de los algoritmos de propósitos específicos, el tiempo de ejecución depende de las propiedades de los factores desconocidos, y algunos de ellos son:

- **Método de factorización de Fermat**
- **División por tentativa**
- **Algoritmo rho de Pollard**
- **Algoritmo p-1 de Pollard**
- **Algoritmo p+1 de Williams**
- **Factorización de curva elíptica de Lenstra**
- **Método de factorización de Euler**
- **Criba especial del cuerpo de números**

El estudio de factorización nos obliga a hablar de la **primalidad** (propiedad de un número de ser primo). El procedimiento más antiguo que se conoce para hallar números primos es el de la **criba de Eratóstenes** (siglo II a.C.). Este dice que para obtener todos los primos

ALGORITMOS Y ESTÁNDARES

Toda la información relativa a los algoritmos criptográficos que constituyen estándares internacionales, avalados por las distintas organizaciones, puede obtenerse directamente en sus sitios web. Por ello es recomendable visitar los siguientes enlaces: **IETF** (**www.ietf.org**), **ISO** (**www.iso.org**), **IEC** (**www.iec.ch**) y **NIST** (**www.nist.gov**).

menores a **n**, listamos los números de **1** a **n** y tachamos primero todos los pares, luego los múltiplos de **3** (el siguiente número), luego los de **5** (el primo siguiente), y así sucesivamente. El matemático árabe Al-Banna (siglo XIII) propuso una mejora para su eficiencia, afirmando que es suficiente con iterar hasta los divisores primos de **n** menores que su raíz cuadrada. La criba no determina la **primalidad** sino que arroja una lista de números primos, pero puede determinarse para un **n** si este permanece en la lista luego de aplicar el método para **n-1**.

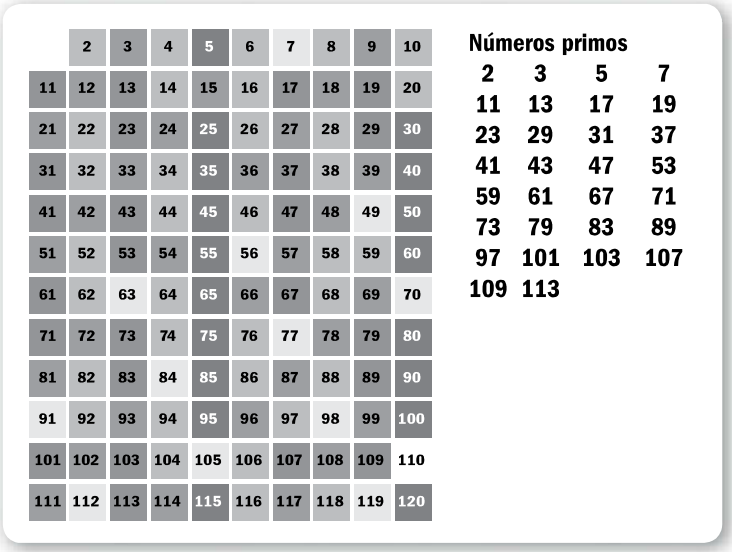


Figura 4. Ejemplo de **criba de Eratóstenes** para números primos menores que 121, donde los distintos tonos de gris representan los números que se descartan por cada pasada.

Más adelante, el matemático italiano del siglo XI Leonardo de Pisa (más conocido como Fibonacci) propuso un método para saber si un **n** es primo, que consistía en verificar que ningún otro número primo inferior a la raíz cuadrada de **n** divide a **n**. Es muy ineficiente pero es determinista (siempre tiene solución).

En 1878, Edouard Lucas propuso una prueba para determinar si un **número de Mersenne** es primo. Un número es de Mersenne si es una unidad menor que una potencia de **2** (**$Mn=2^n-1$**) y un **número primo de Mersenne** es un **Mn** que es primo (**Mp**). La prueba fue optimizada por Derrick Lehmer en la década de 1930 y, a partir de entonces, se la conoce como **prueba de Lucas-Lehmer** y se la considera un **test verdadero de primalidad** (es **determinístico**). La prueba dice que siendo **Mp** un número de Mersenne, se define la sucesión **{Si}** para **i≥0** de la siguiente

forma: **Si=4** para **i=0** y **S²i-1-2** para todo otro valor de **i**. Luego, **Mp** es primo cuando $\mathbf{s_{p-2} \equiv 0 \bmod Mp}$, y compuesto en cualquier otro caso. Al valor $\mathbf{s_{p-2} \bmod Mp}$ se lo denomina **residuo Lucas–Lehmer de p**. Según el algoritmo implementado, se puede obtener una complejidad de $\mathbf{O(n^2 \log n \log \log n)}$, donde **n** es la longitud del número.

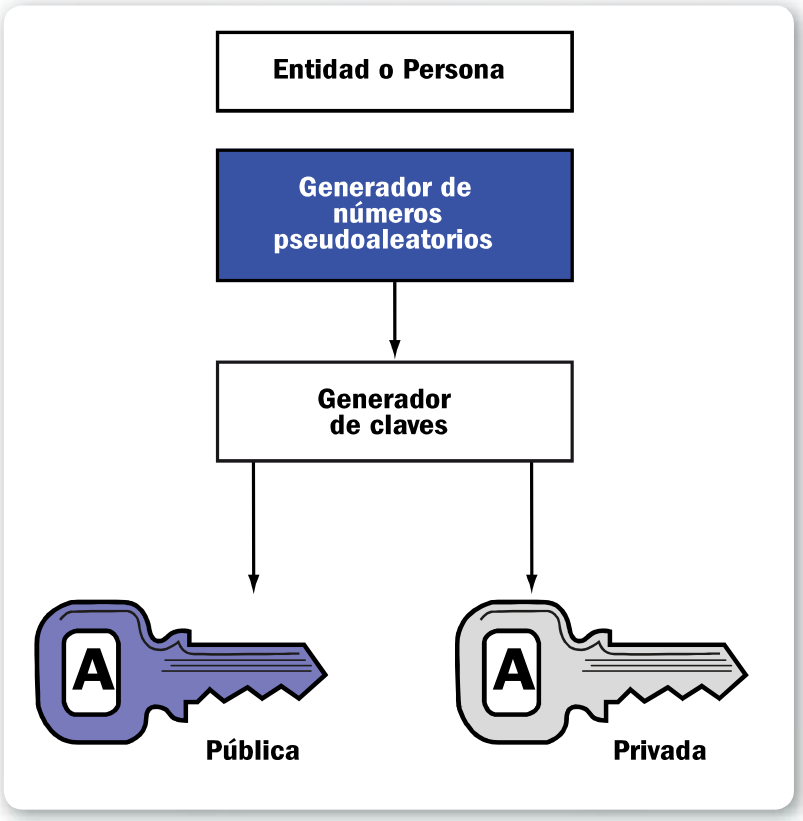


Figura 5. En un sistema asimétrico, el primer paso siempre consiste en la generación del par de claves válidas.

Existen otras pruebas **determinísticas**, que mencionaremos en la próxima página sin entrar en detalles.

CUERPO MATEMÁTICO

Un **cuerpo** (o **campo**) es una estructura algebraica donde se pueden realizar las operaciones de **adición** y **multiplicación**, y cumplen las **propiedades** asociativa, conmutativa y distributiva de la multiplicación respecto de la adición. También debe existir **inverso** aditivo y multiplicativo, y **elemento neutro** para adición y multiplicación (para permitir operaciones de sustracción y división).

- **Teorema de Pépin** (**1877**)
- **Teorema de Proth** (**1878**)
- **Test de Pocklington** (**1914**)
- **División por tentativa**
- **APR** (*Adleman, Pomerance, Rumely*)

También existen **pruebas probabilísticas**, que se basan en la idea de hacer no estricta la corrección de la prueba en cuestión, a fin de obtener un comportamiento polinomial.

Estos se suelen fundamentar en el **pequeño teorema de Fermat**, que ofrece una condición necesaria para que un número sea primo. A continuación mencionaremos las dos pruebas más básicas usadas para probar primalidad:

- **Test de Fermat**: consiste en elegir varios enteros aleatorios **a** entre **2** y **n-2** y calculando $\mathbf{r \equiv a^{n-1} \bmod n}$. Esto devuelve un **número compuesto** (natural, no primo y distinto de 1) si algún valor de **r** es distinto de **1**, y primo en cualquier otro caso. Luego de unas pocas repeticiones, es muy baja la probabilidad de un falso positivo (que un compuesto pase como primo). La contra es que los números llamados **de Carmichael** (pseudoprimos en cualquier base) pasan la prueba para todo valor de **a**.
- **Test de Miller-Rabin**: propuesto por G. Miller y modificado por M. Rabin para que sea un algoritmo probabilístico incondicional (contrario al determinista). Suponiendo un número impar **n>1** del que queremos saber si es primo, y siendo **m** un valor impar tal que **n−1=2k m** y **a** un entero aleatorio entre **2** y **n-2**, al cumplirse: $\mathbf{a^m \equiv m +/- 1 \bmod n}$ o bien $\mathbf{a^{m.2\wedge r} \equiv -1 \bmod n}$ al menos para un entero **r** entre **1** y **k-1**, se dice que **n** es un **primo probable** (lo contrario lo descarta). En caso afirmativo, se elige otro valor de **a** y se itera para reducir el margen de error (las operaciones son rápidas por la exponenciación binaria). En una iteración, un número es probable primo con probabilidad menor al 25%, pero si asumimos la **hipótesis de Riemann generalizada** (no comprobada), es posible demostrar que si se verificaron todos los valores **a** hasta $\mathbf{2(\ln n)^2}$ y **n** sigue pasando la prueba, entonces **es primo**. Su complejidad es de **orden logarítmico** de cuarto grado: $\mathbf{O((\ln n)^4)}$.

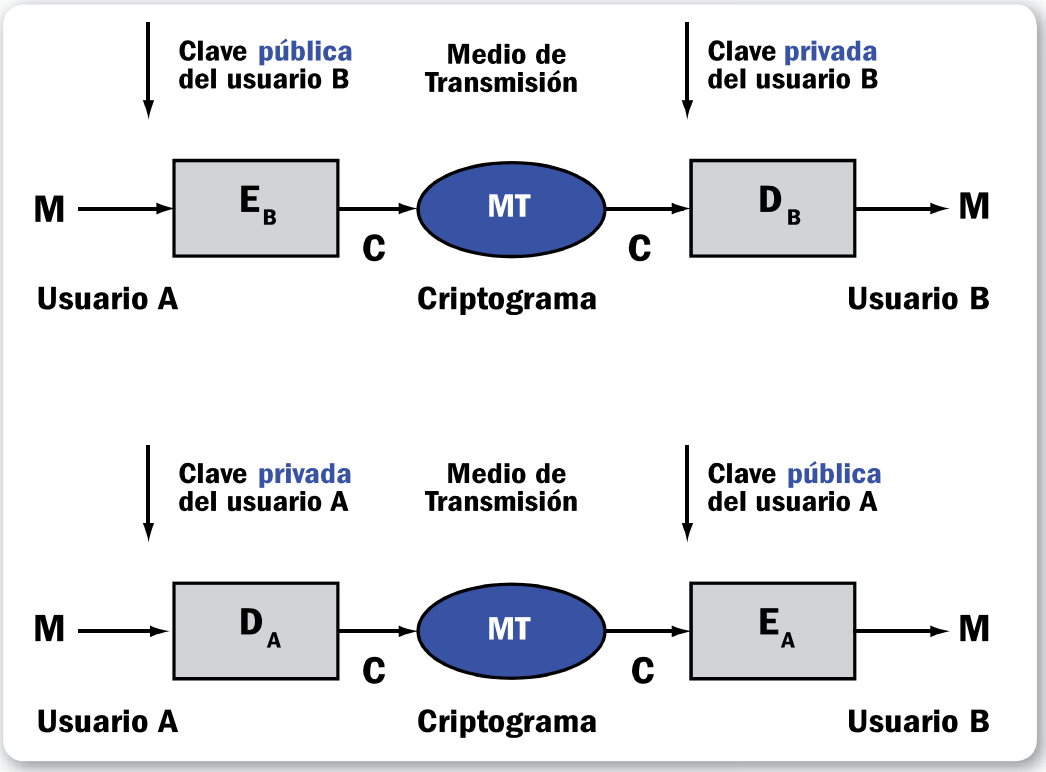


Figura 6. Esquemas de cifrado con llave pública del receptor para obtener **confidencialidad** (arriba) y con llave privada del emisor para obtener **autenticidad** (abajo).

Otras **pruebas de primalidad probabilísticas** que solo mencionaremos sin detallar son:

- **Prueba curva elíptica**
- **Prueba Baillie–PSW**
- **Prueba Frobenius cuadrática**
- **Prueba Solovay–Strassen**

En el año 2002, M. Agrawal, N. Kayal y N. Saxena propusieron el **test AKS** para definir primalidad en un tiempo polinomial, siendo el más avanzado que existe y resultando el primero en ser simultáneamente **general**, **polinomial**, **determinista** e **incondicional**. Se basa en una generalización del **pequeño teorema de Fermat** a los polinomios, afirmando que si **n** y **a** son coprimos, con **n** primo, se cumple la congruencia: $\mathbf{(x+a)^n \equiv (x)^n + (a \bmod n)}$. O sea, si se eleva el polinomio **(x+a)** a la **n** y luego se divide por **n**, el residuo es $\mathbf{x^{n+a}}$, y si se cumple la congruencia, **n** debe ser primo.

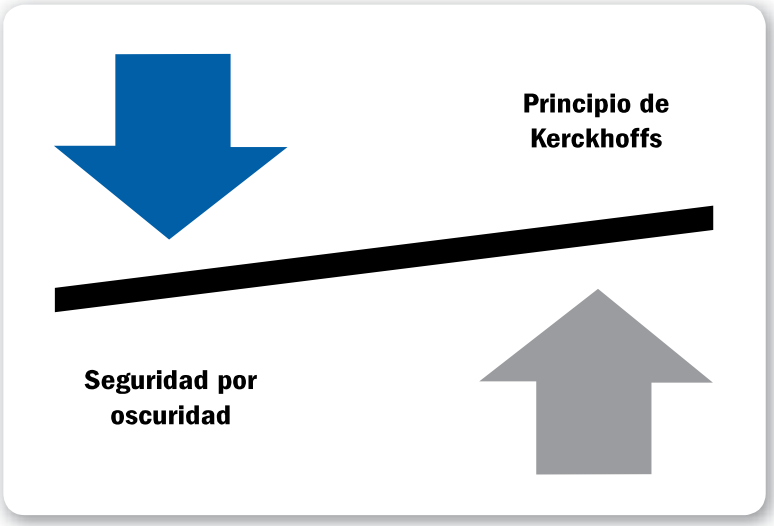


Figura 7. Según el **principio de Kerckhoffs** (1883), la seguridad de un criptosistema no debe depender del secreto de su diseño sino del secreto de las claves, lo cual se opone al concepto de **seguridad por oscuridad**.

Logaritmos discretos

El logaritmo de un número es el exponente al que se debe elevar una base para obtener dicho número, y su operación inversa es la exponenciación. Luego, el logaritmo discreto de **n** en base **b** (siendo **b** y **n** elementos de un **grupo cíclico finito G**) es la solución **x** de la ecuación $\mathbf{b^x=n}$. Matemáticamente es $\mathbf{x=log_b(n)}$. Calcularlo es computacionalmente difícil por el cuerpo **n**, aunque su inversa (exponenciación discreta) es muy fácil, lo que se aprovecha para imposibilitar hallar **x** en un tiempo razonable.

Un algoritmo básico para computarlo consiste en elevar **b** a potencias más y más altas **k** hasta encontrar **n**. Esto se llama **multiplicación por tentativa** y requiere un tiempo lineal en el tamaño del grupo **G**, y exponencial en el número de dígitos en el tamaño del grupo.

PRIMOS FUERTES

Los primos fuertes se definen diferente en teoría de números que en criptografía. Para la primera, corresponde a un primo tal que sea mayor que la media aritmética de sus primos anterior y posterior. En criptografía, para ser fuerte, además de ser grande, debe cumplir con **p=2*q+1** (con **q** primo).

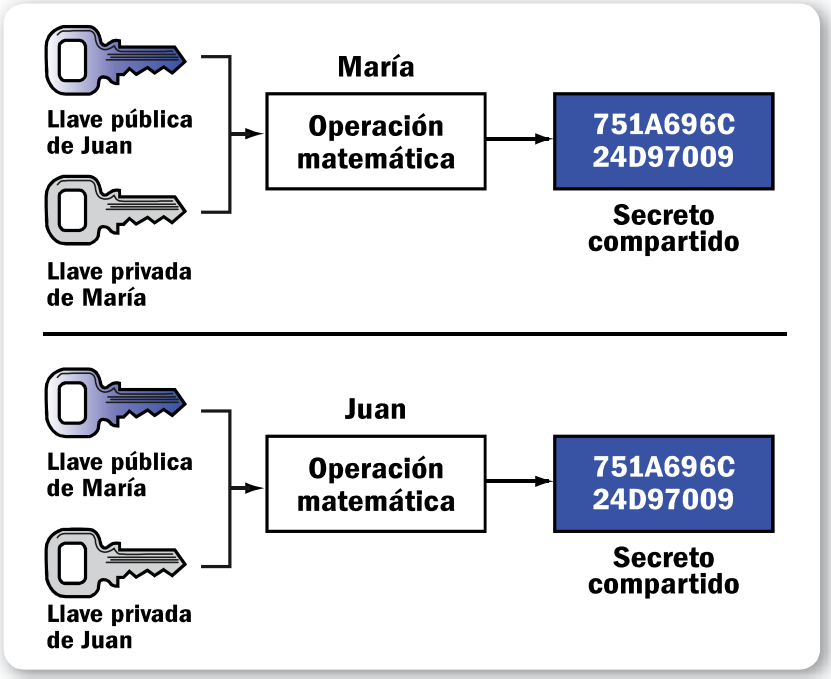


Figura 8. Los sistemas asimétricos se utilizan para **intercambiar claves** de manera segura, a partir de **parámetros públicos**.

Algunos algoritmos más sofisticados corren en tiempo lineal a la raíz cuadrada del tamaño del grupo, por ende exponencial a la mitad de los dígitos del tamaño, pero ninguno corre en tiempo polinomial (en el número de dígitos en el tamaño del grupo). Ejemplos de estos son:

- **Baby-step giant-step**
- **Rho de Pollard para logaritmos**
- **Lambda de Pollard**
- **Pohlig–Hellman**
- **Index calculus**
- **Criba de cuerpo de función**
- **Criba de cuerpo de número**

Un concepto relacionado es el de **raíz primitiva módulo n** (o **generador**). Se dice que **g** es una raíz primitiva **módulo n** (**n** natural) si genera como grupo a **Zn*** (todos los elementos invertibles **módulo n**). Si **p** es primo, entonces existe algún generador **módulo p**, que es el valor que elevado a todos los restos del cuerpo reducido **módulo n** permite generar el cuerpo completo. Así, **g** es un generador si para todo **a** y **b** mayor o igual que **1** y menor

o igual que **(p-1)** entonces g^a **mod p=b**. Dado que el orden de **Zn*** es **φ(n)**, siendo φ la función **phi de Euler**, una raíz primitiva será un elemento que genera este orden.

No es trivial determinar cuántas raíces hay en un primo **p**, ya que existen muchos generadores del cuerpo dentro del conjunto completo de restos. Si conocemos la factorización de **p-1** (**q1**, **q2**... **qn**) siendo **qi** los factores primos de **p-1**, decimos que un **g** será generador en **p** si para todo **qi** se cumple que $g^{(p-1)/qi}$ **mod p** es distinto de **1** (si algún resultado es **1**, **g** no es generador).

Si **g** es una raíz primitiva **módulo p**, un entero **a** no divisible por **p** se puede escribir como $a=g^r$ **mod p** para un único **r {1,2,..., p-1}**. Fijados **a** y **g**, encontrar **r** es lo que se denomina **problema del logaritmo discreto**.

Diffie Hellman

Es un algoritmo desarrollado por Whitfield Diffie y Martin Hellman, publicado en 1976. En rigor de la verdad, Diffie-Hellman (**DH**) es solo un **protocolo de intercambio de claves**, ya que no permite cifrado ni firma. En DH no se envía la clave por el canal, sino una serie de parámetros a través de los cuales, utilizando funciones matemáticas, ambos extremos pueden reconstruir una clave compartida. Su seguridad radica en la complejidad de calcular logaritmos discretos.

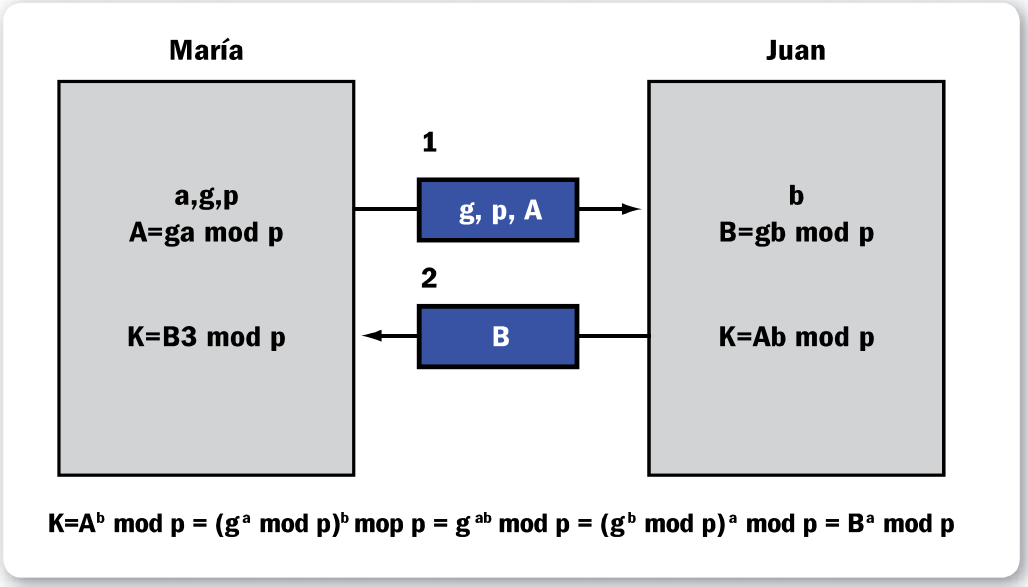


Figura 9. Parámetros del algoritmo **Diffie-Hellman** e intercambio para regenerar la clave en cada extremo.

Supongamos que **María** y **Juan** quieren realizar un intercambio de claves; los pasos serían:

- Ambos establecen un número primo **p** grande (alrededor de 200 dígitos) y un número **g** (generador de dicho primo) que debe ser una raíz primitiva de **p**, ambos parámetros públicos.
- María genera un número aleatorio **a** (menor que **g**) y le envía a Juan el resultado de $g^a \equiv \mathbf{mod\ p}$.
- Juan genera un número aleatorio **b** (menor que **g**) y le envía a María el resultado de $g^b \equiv \mathbf{mod\ p}$.
- Juan calcula $\mathbf{A=(g^{a*b} \equiv mod\ p)}$ y elimina **b**.
- María calcula $\mathbf{B=(g^{a*b} \equiv mod\ p)}$ y elimina **a**.

El resultado de ambas operaciones será el mismo, y se transformará en el **secreto compartido** entre ellos. También puede generalizarse para más de dos usuarios. Dado que **p** y **g** son públicos, si un atacante interceptara un valor $\mathbf{C=g^i \equiv mod\ p}$ debería calcular $\mathbf{i=log_g\ C\ mod\ p}$ que, dados los órdenes de magnitud, es computacionalmente imposible.

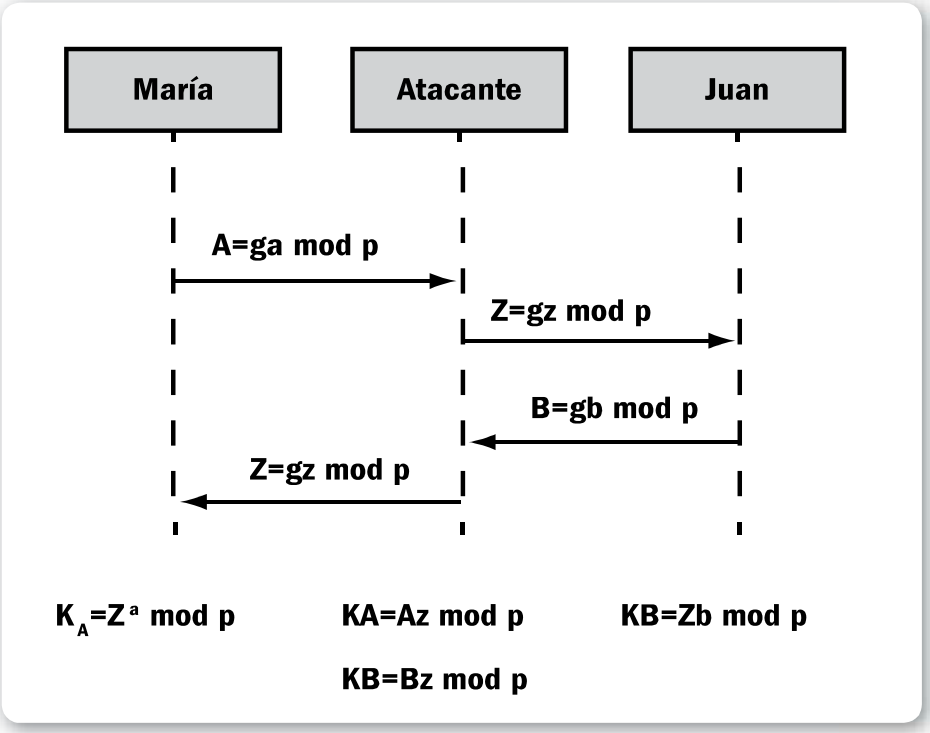


Figura 10. Al no haber autenticación inicial, DH es susceptible a ataques de **MITM** (**man-in-the-middle**), lo que supone la intercepción de un atacante para hacerle creer a María que él es Juan y viceversa.

RSA

Es un algoritmo desarrollado en 1977 por Ron Rivest, Adi Shamir y Leonard Adleman en el **MIT** (las letras **RSA** son las iniciales de sus apellidos). Fue patentado en 1983, aunque la patente expiró en 2000. RSA permite **cifrado**, **intercambio de llaves** y **autenticación**, y su seguridad yace en el problema de la factorización de números enteros, siendo actualmente el tamaño típico de las claves (**cuerpo**) de 1024 ó 2048 bits (se pueden alcanzar los 4096 y 8192 bits).

Número RSA	Cifras decimales	Cifras binarias	Premio ofrecido	Factorizado en	Factorizado por
RSA-100	100	330	-	abril de 1991	Arjen K. Lenstra
RSA-110	110	364	-	abril de 1992	Arjen K. Lenstra y M.S. Manasse
RSA-120	120	397	-	junio de 1993	T. Denny et al.
RSA-129	129	426	$100 USD	abril de 1994	Arjen K. Lenstra et al.
RSA-130	130	430	-	abril de 1996	Arjen K. Lenstra et al.
RSA-140	140	463	-	febrero de 1999	Hernan J.J. te Riele et al.
RSA-150	150	496	-	abril de 2004	Kazumaro Aoki et al.
RSA-155	155	512	-	agosto de 1999	Hernan J.J. te Riele et al.
RSA-160	160	530	-	abril de 2003	Jens Franke et al. Universidad de Bonn
RSA-576	174	576	$10,000 USD	diciembre de 2003	Jens Franke et al. Universidad de Bonn
RSA-640	193	640	$20,000 USD	noviembre de 2005	Jens Franke et al. Universidad de Bonn
RSA-200	200	663	-	mayo de 2005	Jens Franke et al. Universidad de Bonn
RSA-704	212	704	$30,000 USD	abierto	
RSA-768	232	768	$50,000 USD	abierto	
RSA-896	270	896	$75,000 USD	abierto	
RSA-1024	309	1024	$100,000 USD	abierto	
RSA-1536	463	1536	$150,000 USD	abierto	
RSA-2048	617	2048	$200,000 USD	abierto	

Figura 11. La **competencia RSA** promovió la investigación en factorización de enteros hasta 2007. La mayoría de los números grandes no fueron factorizados.

En RSA, los mensajes se representan por números, y se basa en el producto conocido de dos primos grandes (del orden de los 300 dígitos

ALGORITMO MERKLE-HELLMAN

Uno de los primeros algoritmos asimétricos fue el creado en 1978 por Ralph Merkle y Martin Hellman. Era mucho más simple que RSA, pues se basaba en el llamado **problema de la mochila de decisión**, pero no tuvo éxito dado que fue roto en 1982 por Adi Shamir y, además, no ofrecía mecanismo de firma digital.

o más) tomados al azar y secretos. Podemos distinguir tres etapas: la **generación de claves**, el **cifrado** y el **descifrado**. La generación de claves se realiza de la siguiente manera:

1. Se escogen dos números primos grandes **p** y **q** (con tests de primalidad).
2. Se calcula el parámetro **n=(p*q)** (**n** será el módulo de la clave).
3. Se calcula **phi(n)=[(p-1)(q-1)]** (**phi** es la función **φ de Euler**).
4. Se elige un entero positivo **e** menor que **phi(n)** y **coprimo** con este. Se anuncia **e** como exponente de la clave pública (no debería ser muy pequeño).
5. Se determina un **d** (por aritmética modular) que haga cumplir la congruencia **d=[e^{-1}*mod phi(n)]**. El **d** será inverso multiplicativo (o multiplicador modular inverso) de **[e*mod phi(n)]**). Entonces, **[(d*e)-1)]** es dividido exactamente por **[phi(n)=(p-1)(q-1)]**, lo que se calcula con el **algoritmo de Euclides extendido**. Luego, **d** será el exponente de la clave privada. Finalmente, tenemos que:

- La clave pública es **(n,e)** (módulo y exponente de cifrado).
- La clave privada es **(n,d)** (módulo y exponente de descifrado), y debe mantenerse en secreto.

El proceso de cifrado será de la siguiente manera:

- **A** da a conocer a **B** su clave pública (**n**,**e**) y mantiene su clave privada.
- Para que **B** envíe un mensaje **M** a **A**, lo transforma en un número entero **m<n** a través de un protocolo reversible predeterminado y calcula el criptograma **c** con la operación **c≡ m^e mod n** (por exponenciación binaria) para luego transmitirlo.

CIFRADO MALEABLE

En 1992 D. Dolev, C. Dwork y J. Carter establecieron que un cifrado es **maleable** si al conocerse un criptograma **Ci** se puede crear otro criptograma **Cj**, tal que los resultados obtenidos al descifrarlos estén **relacionados** de forma conocida. Esto permite alterar un criptograma sin saber la clave para cambiar el texto plano y evita que se pueda probar la integridad o autenticidad.

El proceso de descifrado será de la siguiente manera:

- **A** recupera **m** a partir de **c** usando su exponente **d** de la clave privada, calculando **m≡c^d (mod n)**. Luego, recupera **M** con el mismo protocolo reversible.

Para romperlo, se debería obtener la clave privada a partir de la pública y el parámetro **n**. Actualmente, para claves reales es computacionalmente imposible invertir la función para obtener **d**. No obstante, para que sea efectivo, deben tomarse ciertas consideraciones en la implementación al elegir los valores de **p** y **q**:

- Que sean del orden de los 500 bits.
- Que no sean muy cercanos para no simplificar la factorización de **n**: suponiendo **p>q**, entonces **(p-q)/2** es un entero muy pequeño y **(p+q)/2** es un entero algo superior a $\mathbf{n^{1/2}}$.

De esta manera, el algoritmo permanecerá seguro en tanto no se conozcan métodos rápidos para factorizar enteros grandes en producto de primos.

ElGamal

Este algoritmo, publicado en 1985 por Taher ElGamal, extiende los conceptos matemáticos de **DH** para soportar un criptosistema completo y obtener firma digital y cifrado. Su seguridad se basa en el **problema del logaritmo discreto**.

Los cálculos de cifrado y descifrado se realizan sobre un **grupo cíclico** (grupos en los que existe un generador **g** tal que todos sus elementos puedan expresarse como potencia de **g**), por lo que

COMPETENCIA RSA

La competencia de factorización RSA es un desafío que propuso la empresa RSA en 1991 a fin de promover la investigación en teoría de números sobre la factorización de enteros grandes. Se publicó una lista de los llamados **números RSA**, que son **semiprimos** (números con dos factores primos), ofreciendo un premio para quien los factorizara con éxito.

la seguridad depende de la dificultad en el cálculo de logaritmos discretos en el grupo. Siendo **G** un **grupo multiplicativo de enteros módulo p**, a fin de generar las claves se realizan los siguientes pasos:

- Se elige un primo **p** tal que **(p–1)** tenga un factor primo grande.
- Se elige aleatoriamente un generador **g** (valor público).
- Se elige un valor **a** mayor que **2** y menor que **(p-1)**, que será la **clave privada**, que debería ser muy grande.
- Se calcula $\mathbf{A=g^a\ (mod\ p)}$, que resulta ser la **clave pública (p, g, a)**.

Para cifrar un mensaje **M** se opera de la siguiente manera:

- Se convierte **M** en un elemento **m** del grupo **G**.
- Se elige un número **b** aleatorio, mayor que **2** y menor que **(p-1)**.
- Se calcula $\mathbf{y_1=g^b\ (mod\ p)}$ y $\mathbf{y_2=A^b\ m\ (mod\ p)}$.
- Se obtiene como criptograma la **tupla** (secuencia ordenada de objetos): $\mathbf{C^b(m,b)=(y_1, y_2)}$.
- Para descifrar el criptograma se hace $\mathbf{y_1^x\ y_2\ (mod\ p)}$ donde **x=p-1-a**. Operando y aplicando el pequeño teorema de Fermat resulta: $\mathbf{y_1^{(p-1-a)}\ y_2=m\ (mod\ p)}$.
- ElGamal es **incondicionalmente maleable**, por lo tanto susceptible a ataques de **texto plano escogido**: si el atacante conoce un mensaje y su criptograma, y el emisor genera un nuevo criptograma con la misma clave privada, se puede obtener el nuevo texto plano. Una extensión no vulnerable a esto es el sistema **Cramer-Shoup**. Además de esto, tiene la desventaja de que duplica la longitud de un mensaje al cifrarlo. Respecto de su rendimiento, si el módulo utilizado es **n**, los procesos de cifrado y descifrado toman tiempos de **O(log n)**.

POHLIG Y HELLMAN CON CLAVE SECRETA

Un mes antes de nacer **RSA**, S. Pohlig y M. Hellman propusieron un algoritmo de clave secreta basado en el problema del logaritmo discreto. Este no pudo competir con la velocidad de los algoritmos simétricos comunes y tampoco contaba con firma digital real, por lo que solo servía para otorgar confidencialidad a mensajes o números.

Curvas elípticas

Las matemáticas definen las curvas elípticas como un tipo de **curva plana** que se representa mediante ecuaciones de tercer grado (**cúbicas**): **y2=x3 + ax + b**. Vale aclarar que estas curvas no son elipses.

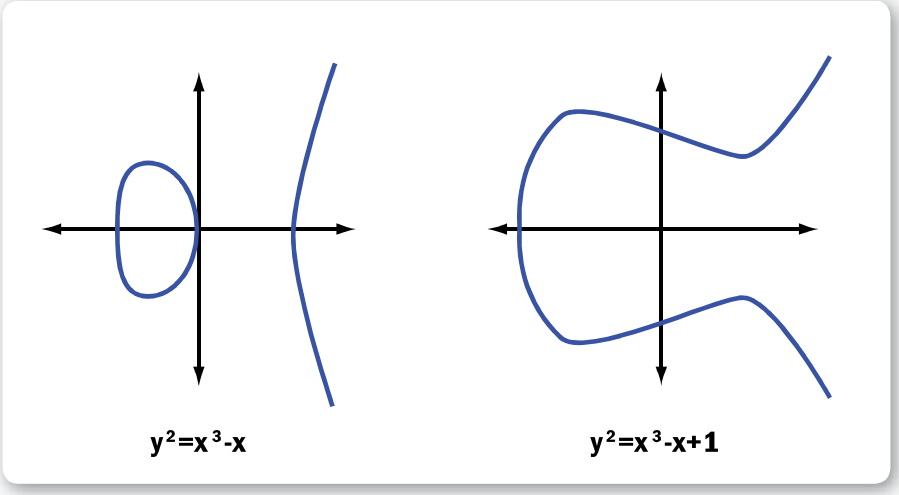


Figura 12. Curvas elípticas definidas por su ecuación característica. Su uso más famoso fue el de haber servido para probar el **último teorema de Fermat**.

No tienen **vértices** ni **autointersecciones** (son regulares) y permiten definir de forma geométrica una operación binaria para su conjunto de puntos. Si se tiene un conjunto de puntos **G** que conforman todas las soluciones posibles de la ecuación de la curva elíptica, más un punto en el infinito y una operación de adición, se consigue un **grupo abeliano**. Si **x** e **y** pertenecen a un **cuerpo finito**, tenemos un **grupo abeliano finito**, sobre el cual el problema del logaritmo discreto resulta más complicado de resolver que cuando tratamos solamente con cuerpos finitos, aunque esto no está del todo demostrado. En ese caso, le llamamos **problema del logaritmo discreto en curvas elípticas**.

La **criptografía de curva elíptica** (o **ECC**, por *Elliptic Curve Cryptography*) resulta entonces una variante de la criptografía asimétrica basada en las curvas elípticas, y fue propuesta de manera independiente por Neal Koblitz y Victor Miller en 1985.

Se estima que puede utilizar claves más cortas que otros métodos (se gana en velocidad) con un nivel equivalente de seguridad. En rigor, no es un algoritmo sino una herramienta matemática que puede implementarse por algoritmos como **DH**, **DSA** y **ElGamal**, haciendo

que sea posible reescribir un algoritmo que usa grupos finitos en base a los grupos de puntos racionales de las curvas elípticas.

Característica	Cifrado	
	Simétrico	Asimétrico
Confidencialidad	Sí	Sí
Autenticación	Parcial	Completa
Firma digital	No	Sí
Longitud de clave	Pequeña	Grande
Vida de clave	Corta (Sesion)	Larga
Cantidad de claves	n (n-1)/2	n*2
Velocidad	Alta	Baja
Aplicaciones	Cifrado de mucha información	Firma e intercambio de claves

Figura 13. Resumen comparativo entre algoritmos asimétricos y simétricos.

Se escoge un punto base **G** (**público**) para usar con la curva **E(q)** y un número entero aleatorio **k** que será la clave privada. Luego se calcula **P=k*G**, que corresponde a la clave pública. Si un usuario **A** tiene la clave privada $\mathbf{k_A}$ y la pública $\mathbf{P_A}$, y un usuario **B** tiene $\mathbf{k_B}$ y $\mathbf{P_B}$, entonces **A** podría calcular $\mathbf{k_A}*\mathbf{P_B}=(\mathbf{k_A}*\mathbf{k_B})*\mathbf{G}$, y **B** podría calcular el mismo valor con $\mathbf{k_B}*\mathbf{P_A}=(\mathbf{k_B}*\mathbf{k_A})*\mathbf{G}$, obteniéndose así la clave secreta compartida.

A nivel de estandarización, el **NIST** y el **ANSI** establecieron requisitos para el tamaño de la clave de 1024 bits (**RSA** y **DSA**) y de 160 bits para curvas elípticas, respecto a un bloque simétrico de 80 bits de clave. El **NIST** publicó una lista recomendada de curvas elípticas de cinco tamaños de claves (80, 112, 128, 192, 256). Si se trabaja sobre un grupo binario, la clave asimétrica será del doble de tamaño que una simétrica, en ECC.

RESUMEN

En este capítulo estudiamos el cifrado asimétrico, que utiliza un par de claves (pública y privada) para operar, requiriendo que lo que se cifra con una se descifre con la complementaria. Estudiamos también los principales problemas matemáticos en los que se basan: factorización de números grandes y logaritmos discretos. Vimos el algoritmo más famoso de intercambio de claves, Diffie-Hellman, el más popular por su seguridad, RSA, y el no menos importante ElGamal. Finalmente, analizamos la criptografía de curva elíptica.

Actividades

TEST DE AUTOEVALUACIÓN

1. ¿Qué diferencia hay entre los algoritmos simétricos y los asimétricos respecto a la cantidad de claves que se requieren?
2. ¿Por qué los algoritmos asimétricos son más lentos que los simétricos?
3. ¿En qué consiste el problema del logaritmo discreto?
4. ¿En qué consiste el problema de la factorización de enteros?
5. ¿Qué características tienen en común ambos problemas?
6. ¿Cuál es el uso principal que se le da al sistema de cifrado de Diffie-Hellman?
7. ¿Cuáles fueron las razones por las que RSA ha sido el algoritmo asimétrico más popular?

EJERCICIOS PRÁCTICOS

1. Analice otros algoritmos asimétricos, menos conocidos que los presentados.
2. Realice ejemplos numéricos para verificar el funcionamiento de los algoritmos estudiados.
3. Averigüe cuáles han sido los últimos avances en factorización de enteros.
4. Averigüe cuáles fueron las contribuciones de los creadores de RSA al campo de la criptografía y el criptoanálisis más allá de la creación del algoritmo.
5. Estudie el funcionamiento de los principales algoritmos de factorización de enteros.

PROFESOR EN LÍNEA

Si tiene alguna consulta técnica relacionada con el contenido, puede contactarse con nuestros expertos: profesor@redusers.com

*06

Funciones hash

Las funciones hash son un tipo especial de funciones matemáticas que merecen un estudio en el ámbito de la criptografía por sus múltiples usos, principalmente asociados a la autenticación y la integridad.

Origen y necesidad

Se denomina **función de hash** a aquella **función computable** por un algoritmo que convierte una entrada en un rango de salida finito, en general de tamaño fijo. Dicha función puede decirse que proyecta un conjunto **M** (**preimagen**) en general grande, sobre un conjunto **D** (**imagen**) en general más pequeño (por eso, también se las llama **resumen** o **digest**). El objetivo principal es funcionar como representación reducida de una entrada.

Figura 1. Las funciones de hash convierten una entrada de tamaño arbitrario en una salida de tamaño fijo.

No todas estas funciones son utilizadas en el ámbito de la criptografía, sino solo aquellas llamadas **funciones hash criptográficas**, que se destacan por contar con ciertas propiedades que las hacen más seguras. No son obvias las propiedades específicas ya que las aplicaciones son diversas, pero seguro habrán de ser **deterministas** (que cada mensaje tenga siempre el mismo resultado), **fáciles de procesar**, **uniformes** y **con efecto avalancha** (cada bit de salida depende de todos los de entrada), para que no se pueda predecir un resultado a partir de otros.

Es posible determinar una clasificación en función de su objetivo entre aquellos métodos que persiguen la integridad, llamados **códigos**

de detección de modificaciones (entre los que están las funciones hash criptográficas dedicadas que veremos luego) y los que apuntan a la autenticación de origen, llamados **códigos de autenticación de mensajes** (que veremos al final del capítulo). Estas denominaciones son técnicamente incorrectas, pero se las acepta así.

Sus usos son múltiples, como por ejemplo en autenticación de usuarios de sistemas operativos, comprobando que el hash correspondiente al password ingresado coincide con el almacenado en el sistema.

Colisiones

En el contexto de las funciones de hash, se le llama **colisión** a la situación en la que dos entradas diferentes a un algoritmo generan el mismo resultado a la salida. Esto es matemáticamente esperable, por definición, en el caso de que la entrada sea más grande que el tamaño de la salida estándar del algoritmo (si esta es fija), ya que siempre se tiende a reducir la información. De hecho, un determinado valor de hash puede provenir de infinitas potenciales entradas.

Dentro de la clasificación de las funciones encontramos los mecanismos de hash llamados **perfectos**, que no producen colisiones porque su función de enumeración asigna una única posición a cada valor. Por otro lado, están los llamados **mecanismos puros**, donde la función puede asignar el mismo valor a dos entradas, llamadas **sinónimos**. Estas últimas deben tener métodos de tratamiento, entre los que se encuentran las **estructuras cerradas**, que no utilizan un nuevo espacio, y las **abiertas**, que sí lo hacen, y a su vez pueden ser

MEZCLAME UN HASH

El término **hash** proviene del significado de los verbos del inglés **cortar** y **mezclar**, ya que las funciones hash "recortan" la entrada y "mezclan" la salida. Se dice que el primero en aplicar el término fue H. Luhn, de **IBM**, en 1953, aunque no comenzó a utilizarse masivamente hasta la década del 60, luego de que R. Morris lo aplicara en una publicación técnica.

estáticas (cuando no crece la estructura) o **dinámicas**, cuando la estructura crece con los elementos.

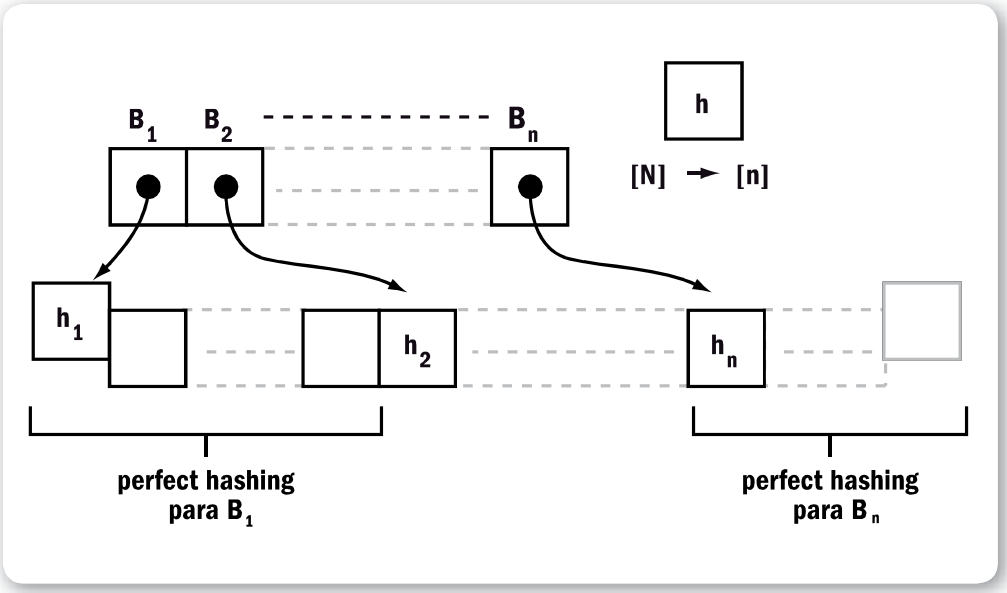


Figura 2. Cuando una función hash no tiene colisiones se la denomina **perfecta** (**perfect hashing**).

Es esperable que se deseen construir funciones lo más resistentes posibles a colisiones, pero en la práctica se considera suficiente con que éstas solo puedan ocurrir en un tiempo que no sea útil para un potencial atacante. En este sentido, nombramos dos clases de resistencia:

- **Resistencia débil**: cuando, conocido **M**, es posible computacionalmente encontrar otro **M'** tal que **h(M)=h(M')**.
- **Resistencia fuerte**: cuando es computacionalmente difícil encontrar un par (**M**, **M'**) de forma tal que **h(M)=h(M')**.

PRINCIPIO DEL PALOMAR

Este principio, enunciado por primera vez por Dirichlet en 1834, dice que si se distribuyen **n** palomas en **m** palomares, siendo **n>m**, habrá uno con más de una paloma. Generalizando, **m huecos** pueden alojar un máximo de **m objetos**, y el agregar uno más fuerza a reutilizar alguno. Esto es análogo al funcionamiento de una función hash.

Ataques a funciones hash

Los ataques a funciones hash tienen como objetivo el descubrimiento de mensajes cuyos resultados de resumen coincidan. Para esto se estudian las propiedades de los algoritmos a fin de reducir las operaciones necesarias para conseguirlo. Estos ataques no incluyen los errores en las implementaciones.

Ataque de colisión

El proceso de encontrar dos valores arbitrarios cuyos hashes colisionan se llama **ataque de colisiones**. Es decir, hallar **m1** y **m2** de tal forma que **h(m1)=h(m2)** siendo **m1** y **m2** dos mensajes cualesquiera y **h1** y **h2** sus resultados correspondientes, sin prefijar ninguno de los elementos.

A través de la llamada **paradoja del cumpleaños**, los ataques de colisión son a los algoritmos de hash lo que los ataques de fuerza bruta son a los algoritmos simétricos. Es decir, es un ataque inherente a su naturaleza y siempre puede realizarse.

> SEGÚN LA PARADOJA DEL CUMPLEAÑOS, UN HASH DE N BITS PUEDE SER ROTO EN $2^{N/2}$ EVALUACIONES

Según la paradoja del cumpleaños, un hash de **n bits** puede ser roto en $2^{n/2}$ evaluaciones. Sin embargo, es posible conseguir ataques más eficientes utilizando técnicas criptoanalíticas que reduzcan el valor del exponente. Cuando se descubre un ataque más rápido que un ataque por paradoja del cumpleaños, se considera que una **función de hash está rota**.

HASHES DE ARCHIVOS

El uso básico de las funciones hash en sistemas es el de validar la integridad de los archivos. Es decir, aplicar la función al archivo y obtener el resultado, que se almacenará en un lugar seguro. Si en un momento posterior se vuelve a aplicar la función y se obtiene el mismo resultado, puede garantizarse que el archivo no ha sido modificado.

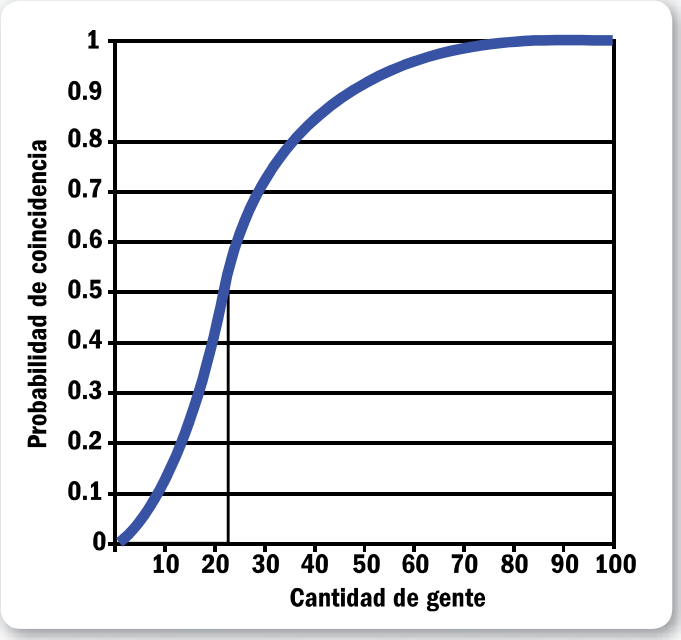


Figura 3. La paradoja del cumpleaños dice que entre 23 personas, la probabilidad de que dos cumplan años el mismo día es del 50,7% y para 57 o más, la probabilidad es de más del 99%.

Si bien sabemos que matemáticamente es posible conseguir las colisiones, el problema práctico es hacerlo de forma tal que los mensajes tengan sentido en el contexto del ataque. Algunos formatos de archivo permiten verificar si una parte de él posee un valor u otro, de manera de poder controlar lo que se muestra. Esto hace que ciertos tipos de documentos sean más o menos vulnerables a ataques de colisión. En estos casos, lo que se busca particularmente es modificar un archivo a fin de que lo visualizado sea diferente, pero su valor de hash coincida con el original. En el peor de los casos, podría haber un archivo firmado digitalmente y otro alterado que tenga una firma válida correspondiente al original.

Además, podemos determinar un tipo de ataque de colisión que podemos llamar de **prefijo escogido**, donde dados dos prefijos **p1** y **p2**, se busca encontrar dos mensajes **m1** y **m2** tales que **h(p1||m1)=h(p2||m2)**, donde || es el **operador de concatenación**.

Esta variante de ataque es específica para hashes basados en la estructura de **Merkle-Damgard**. En la práctica, la idea es elegir dos documentos arbitrarios distintos y adicionarles dos valores concatenados diferentes que resulten en que ambos documentos tengan el mismo hash.

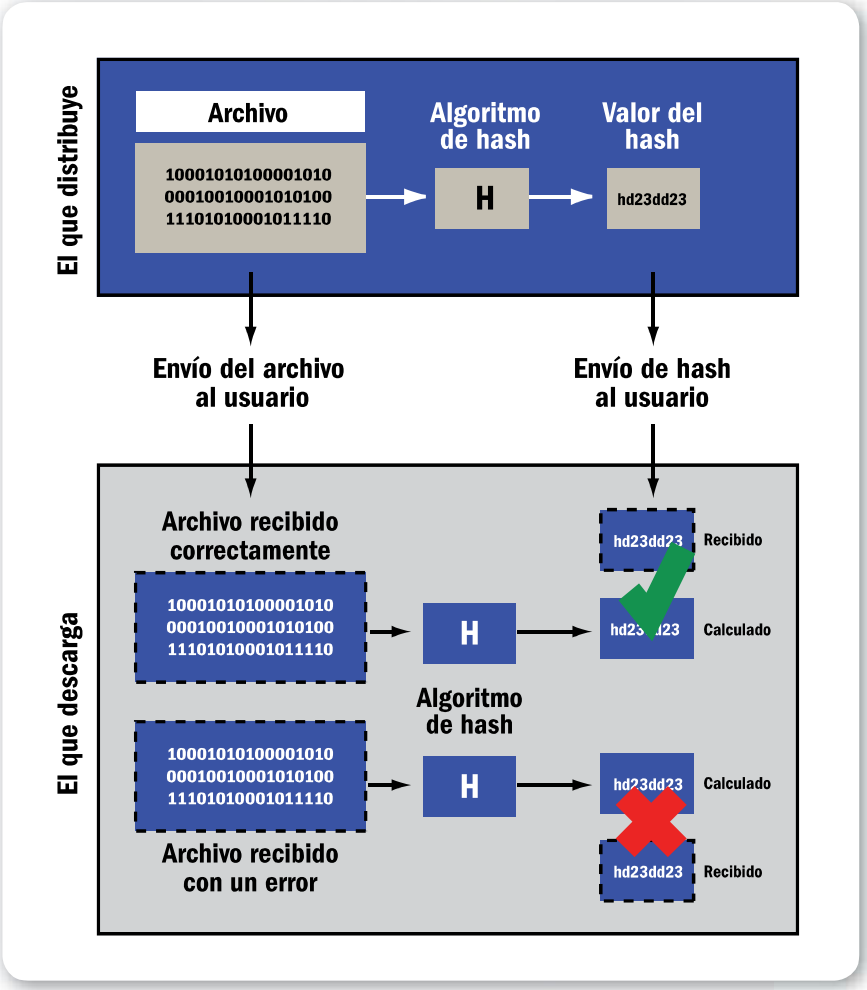


Figura 4. Un uso común de las funciones de hash es el de verificar **descargas** de internet, comprobando que el hash calculado coincida con el calculado originalmente por quien la pone a disposición.

Muchas aplicaciones de las funciones hash en criptografía son resistentes a colisiones, como en los casos en que el atacante no puede controlar el valor de la entrada a la función, lo cual puede llegar a conseguirse por diseño.

PRIMITIVA CRIPTOGRÁFICA

Se llama así a la combinación de **algoritmos**, **protocolos** y **prácticas** que sirven para establecer un criptosistema e incluyen comúnmente esquemas y algoritmos de cifrado y de firma digital, y funciones hash. Se toman como bloques básicos para el diseño, por lo que deben ser muy confiables.

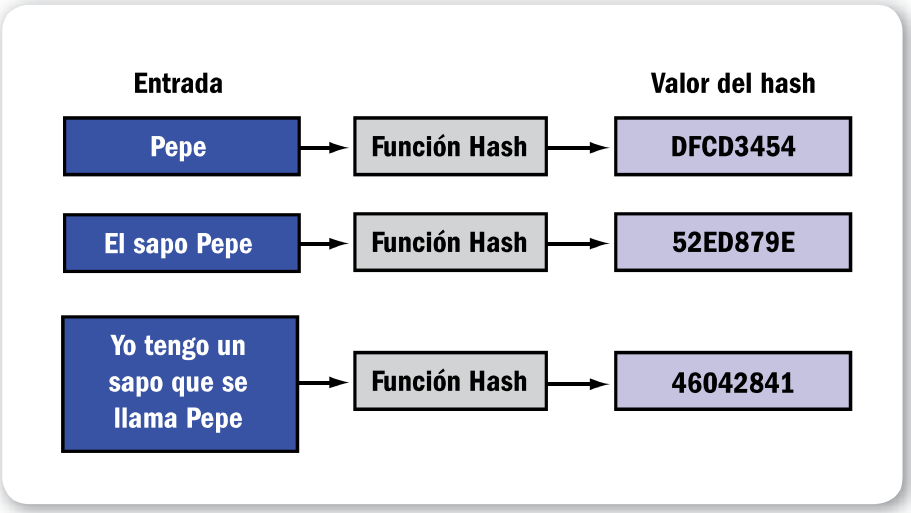


Figura 5. La probabilidad de que una función hash correctamente elegida y utilizada en su contexto tenga una colisión es muy baja.

Ataque de preimagen

Este ataque implica la búsqueda de un valor que coincida con otro en particular, por lo que se lo considera más serio que un ataque de colisiones común. Es decir, dado **m1**, se busca hallar un **m2** tal que **h(m2)=h(m1)**.

Las funciones de hash criptográficas deberían ser resistentes a estos ataques, y podemos considerar dos tipos de resistencia:

EL ATAQUE DE PREIMAGEN BUSCA UN VALOR QUE COINCIDA CON OTRO EN PARTICULAR

- **Resistencia a la primera preimagen**: para cualquier salida, no es computacionalmente factible encontrar una entrada que la produzca.
- **Resistencia a la segunda preimagen**: no es computacionalmente factible encontrar una segunda entrada que tenga la misma salida que una entrada especificada. Esto equivale a la resistencia a las colisiones, aunque ésta no garantiza resistencia a la segunda preimagen.

En una **función segura**, un ataque de preimagen debería realizarse a través de la fuerza bruta, lo que implica una complejidad de 2^n, siendo **n** el tamaño del hash. En ese caso, se consideraría resistente.

Funciones hash no criptográficas

Dado que no todas las funciones de hash buscan brindar seguridad contra ataques deliberados sino que en muchos casos buscan la detección de errores no intencionales, existen algunos tipos que vale la pena mencionar.

El primero es la **suma de verificación** (**checksum**), un pequeño dato utilizado para detectar modificaciones accidentales que puedan haber aparecido en la transmisión o almacenamiento de una información. El valor se envía con el dato para que el receptor lo verifique y compruebe que no haya diferencias entre lo transmitido y lo recibido.

El procedimiento utilizado para calcular el checksum se denomina **función checksum**. Los **dígitos verificadores** y **bits de paridad** son casos especiales de checksum apropiados para pequeños datos. Algunos códigos de corrección de errores están basados en checksums especiales que, en ciertos casos, además permiten recuperar los datos originales.

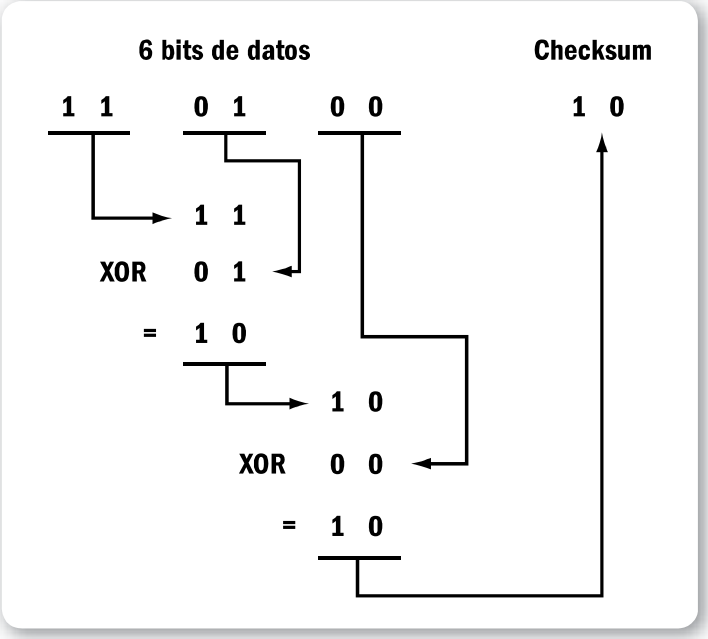


Figura 6. **Checksum** simple basado en **XOR**.

La manera más sencilla de calcular un checksum es llamada **chequeo de redundancia longitudinal** (**LRC**), que consiste en partir el dato en palabras de **n** bits y aplicar una operación **XOR**

entre ellas, siendo el resultado adosado al mensaje como palabra adicional. Un variante a esta última es la **suma modular**, en la que la palabra final se almacena como un número binario no signado, descartando el desborde y adicionando al mensaje el **complemento a dos** del resultado.

Como esto no permite detectar varios errores a la vez, aparecen los algoritmos más utilizados en la práctica: **Fletcher**, **Adler-32**, **Luhn**, **Verhoeff**, **Damn** y **CRC** (**chequeo de redundancia cíclica**) que si bien aumentan el costo computacional, hacen más confiable el sistema.

En particular, **CRC** es un código de detección de errores propuesto por W. Peterson en 1961 y se llama así porque el valor de la verificación (**chequeo**) es una redundancia (expande el mensaje sin agregar información) y el algoritmo se basa en códigos cíclicos. El término suele ser usado para designar tanto a la función como a su resultado, y su cálculo se basa en el residuo de una división de polinomios.

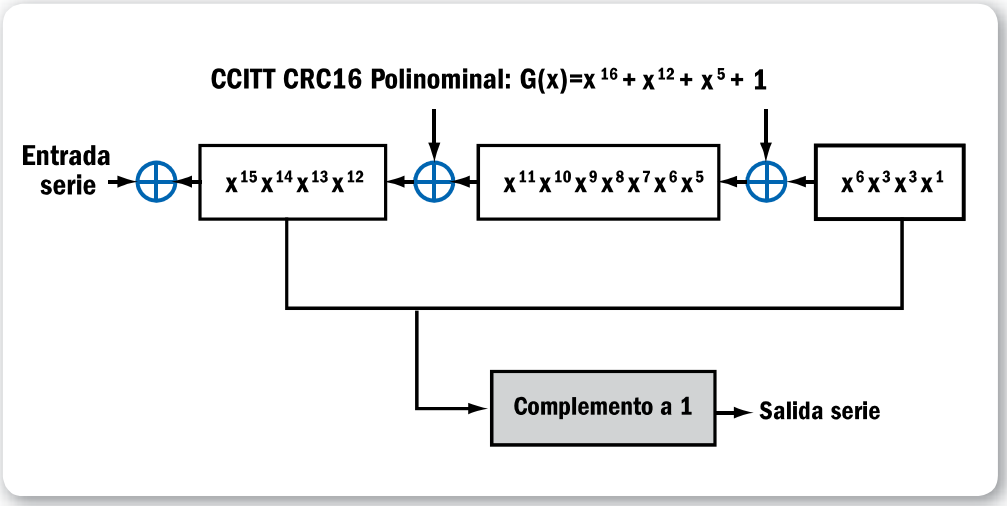


Figura 7. Estructura del estándar **CRC-16 CCITT** que representa al polinomio $X^{16+X12+X5+1}$.

A título informativo, algunos algoritmos más complejos pero no criptográficos que pueden encontrarse son los siguientes (en orden cronológico): **Zobrist** (1969), **Pearson** (1990), **FNV** (*Fowler–Noll–Vo*, 1991), **Java hashCode** (1995), **Jenkins** (1997), **MurMurHash** (2008) y **CityHash** (2011). Sin llegar a ser criptográficos, estos algoritmos tienen distinta complejidad.

Funciones hash criptográficas

Los **hashes criptográficos** cuentan con una serie de fortalezas que los hace útiles para su uso en seguridad. En general, cuando se habla de algoritmos de hash en seguridad, se hace referencia a estos. El nombre **hash** se utiliza tanto para referirse a los algoritmos como al valor de salida (resumen), que en general se expresa en hexadecimales.

Message Digest

La serie **Message Digest** (**MD**) es una familia de funciones hash creada por Ron Rivest, entre los que se encuentran **MD2**, **MD4**, **MD5** y **MD6** (**MD1** no fue publicado y **MD3** fue experimental). **MD5** (1991) es, probablemente, el algoritmo de hash más usado que haya existido y se creó como reemplazo de **MD4**, que a su vez reemplazaba a **MD2**.

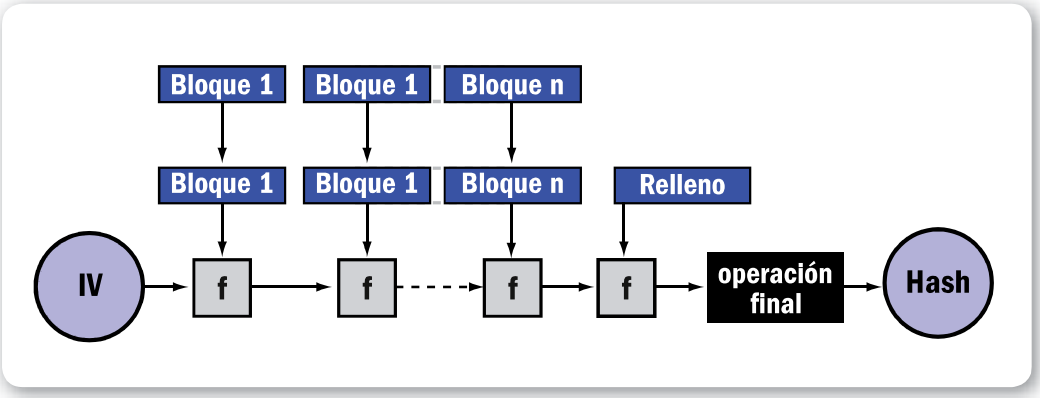


Figura 8. Estructura de **Merkle-Damgard** utilizada en **MD5**, **SHA-1** y **SHA-2**.

MD2

Message Digest 2, o **MD2** (1989), fue el primero de los MD optimizado para sistemas de 8 bits. Opera rellenando el mensaje a un múltiplo de 128 bits y agregando un **checksum**. Además, usa un bloque de 48 bytes y una tabla de 256 bytes formada con decimales de **Pi**. Permuta cada byte del bloque auxiliar 18 veces por cada 16 bytes de entrada y el primer bloque parcial del auxiliar resulta el hash.

MD5 funciona de la siguiente manera:

1. Un mensaje **M** se hace múltiplo de 512 bits para su tratamiento (**congruencia módulo 512**), reservando los últimos 64 bits para el indicador de longitud.
2. Con el primer bloque de 512 bits y los 128 bits de cuatro vectores iniciales fijos **ABCD** de 32 bits cada uno se realizan distintas operaciones lógicas.
3. La salida de las operaciones (de 128 bits) se convierte en el nuevo conjunto de cuatro vectores **A'B'C'D'** que se operan nuevamente con el segundo bloque de 512 bits, repitiendo el proceso hasta el último bloque del mensaje.
4. Los últimos 128 bits corresponden al valor del hash.

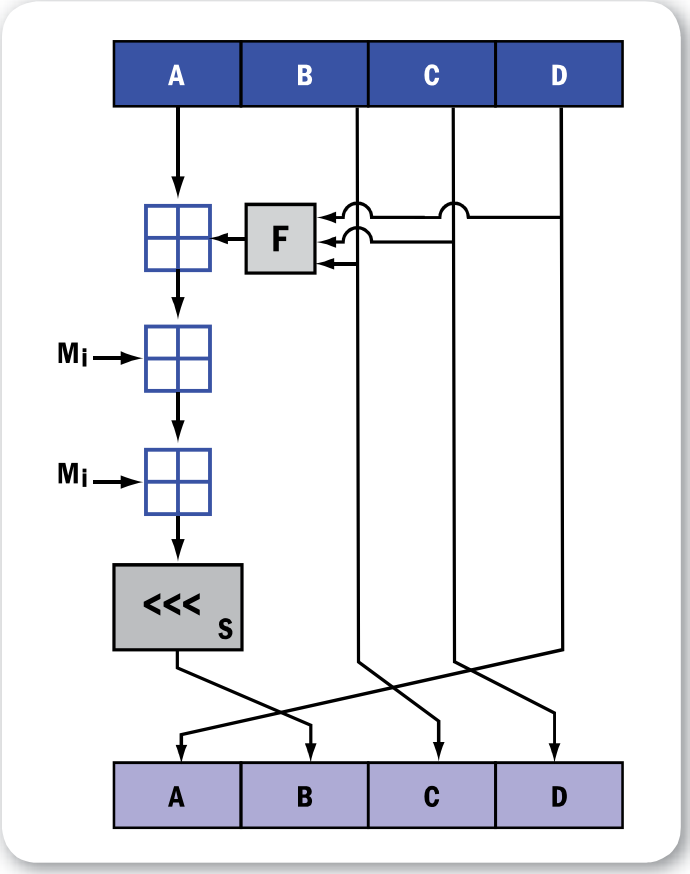


Figura 9. MD5 consiste en 64 operaciones agrupadas en cuatro vueltas de 16 etapas y usa una **función no lineal F** por vuelta. **Mi** es un bloque de 32 bits (mensaje) y **Ki** es una constante de 32 bits distinta por operación. También aplica desplazamientos a la izquierda y sumas **módulo 2^{32}**.

MD5 fue considerado seguro desde su origen, pero en los últimos años ha sufrido diversos ataques que revelaron algunas vulnerabilidades importantes, de modo que comenzó a ser reemplazado con el correr del tiempo. En efecto, en 2004 se propuso **MD5CRK**, un proyecto de ataque distribuido (similar a **distributed.net**) con el fin de demostrar la debilidad de MD5. Estuvo vigente durante cinco meses, tras lo que los investigadores chinos Wang, Feng, Lai y Yu demostraron la existencia de colisiones obtenidas con equipos IBM P960 en clúster, en tan solo una hora, mediante un ataque analítico.

En 2007 se encontró un **ataque de colisión de prefijo escogido**, que requiere 2^{50} evaluaciones. Esto se extendió en la práctica en un ataque a certificados **X.509** en 2008, falsificando la firma de certificados digitales como si provinieran de una determinada autoridad certificante, afectando así a los sitios que utilizan **SSL** (**HTTPS**) para establecer canales seguros.

En 2008, Ron Rivest junto a otros investigadores propusieron **MD6**, de cara al concurso por el nuevo estándar SHA-3 buscado por el NIST. Este utiliza una estructura del tipo **árbol de Merkle** para permitir **paralelización** del cómputo en entradas grandes. A fines del mismo año, Douglas Held descubrió un error de **buffer overflow** en la **implementación original** de referencia de MD6, lo que llevó a una nueva implementación en software el año siguiente.

En cuanto al concurso del NIST, no pasó ni siquiera a la segunda ronda, debido a no poder demostrarse de manera estricta su resistencia a ataques diferenciales. En 2011 se presentó una versión mejorada de MD6, resistente a ataques diferenciales para versiones reducidas en vueltas.

MD4

Message Digest 4, o **MD4** (1991), es el antecesor de **MD5** e influenció a varios otros. Boer y Bosselaers le encontraron debilidades apenas fue publicado y en 1995 Dobbertin demostró un ataque de colisiones que demoraba segundos. En 2007 se publicó el mejor ataque, que requirió menos de dos operaciones. Se utilizó en **NTLM** en Windows NT, XP, Vista y 7, y en 2011 fue declarado obsoleto (RFC 6150).

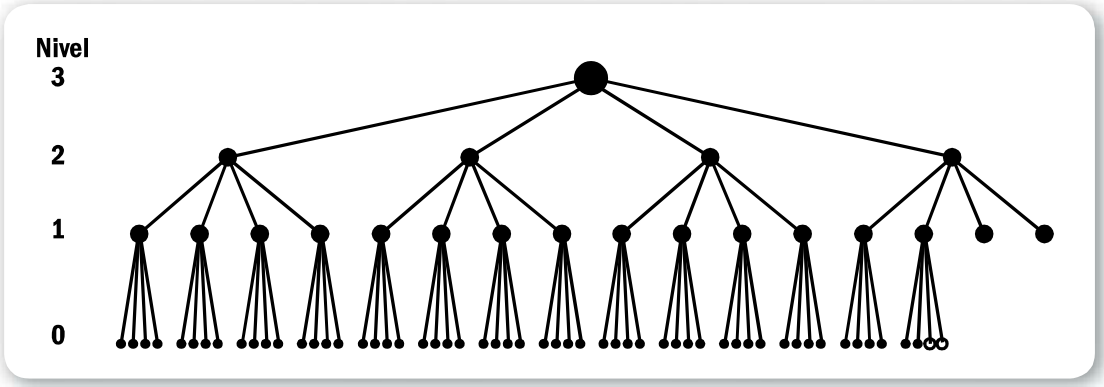


Figura 10. **MD6** utiliza una estructura tipo **árbol de Merkle**, combinando el esquema jerárquico puro con el paralelo.

Secure Hash Algorithm

La serie **Secure Hash Algorithm** (**SHA**) es una **familia** de funciones de hash creadas por la NSA de USA y publicadas por el NIST. Su primer algoritmo fue llamado **SHA** (o posteriormente **SHA-0**) y fue publicado en 1993, y el segundo y más conocido fue **SHA-1**. En 2001 se dio a conocer **SHA-2**, que contiene versiones para distintos tamaños de resumen: **SHA-224**, **SHA-256**, **SHA-384** y **SHA-512**.

Dada su procedencia, la familia fue observada muy de cerca por los criptógrafos, pero no se le encontraron ataques efectivos, aunque los descubrimientos publicados desde 2004 para MD5 se le pudieron aplicar en gran medida debido a que se basa en una estructura similar.

SHA-1 utiliza para su función de compresión una estructura de **Merkle-Damgard** para evitar las colisiones, soporta mensajes de un tamaño máximo de 2^{64} bits, procesando bloques de 512 bits y realizando un total de 80 vueltas. A diferencia de MD5, el vector inicial tiene una palabra adicional de 32 bits (**E**), que es lo que lleva al resultado de 160 bits.

SISTEMAS DE ARCHIVOS ÍNTEGROS

Existen programas para comprobar la integridad de sistemas de archivos completos o carpetas seleccionadas mediante la aplicación de las funciones de hash sobre archivos y carpetas en base a una política predefinida, generando una copia llamada "copia de oro" que se usará como patrón para validar futuras comprobaciones.

Característica	MD5	SHA-1	RIPEMD-160
Longitud del resumen	128 bits	160 bits	160 bits
Tamaño de procesamiento	512 bits	512 bits	512 bits
Número de pasos	64	80	160
Tamaño máximo del mensaje	Sin límite	264 bits	Sin límite

Figura 12. Comparativa entre **SHA-1**, **MD5** y **RIPEMD-160**.

SHA-2, por su parte, se transformó en el nuevo estándar del NIST en 2002, y tuvo algunas modificaciones significativas respecto a su antecesor, mejorando así su seguridad. Cuenta con distinto tamaño de bloques (32 y 64 bits) para sus diferentes **versiones principales** (256 y 512) y tiene **versiones reducidas** (**truncadas**) de 224 y 384 bits, que se computan con diferentes vectores iniciales. El algoritmo está patentado por el gobierno de USA, pero su uso es libre.

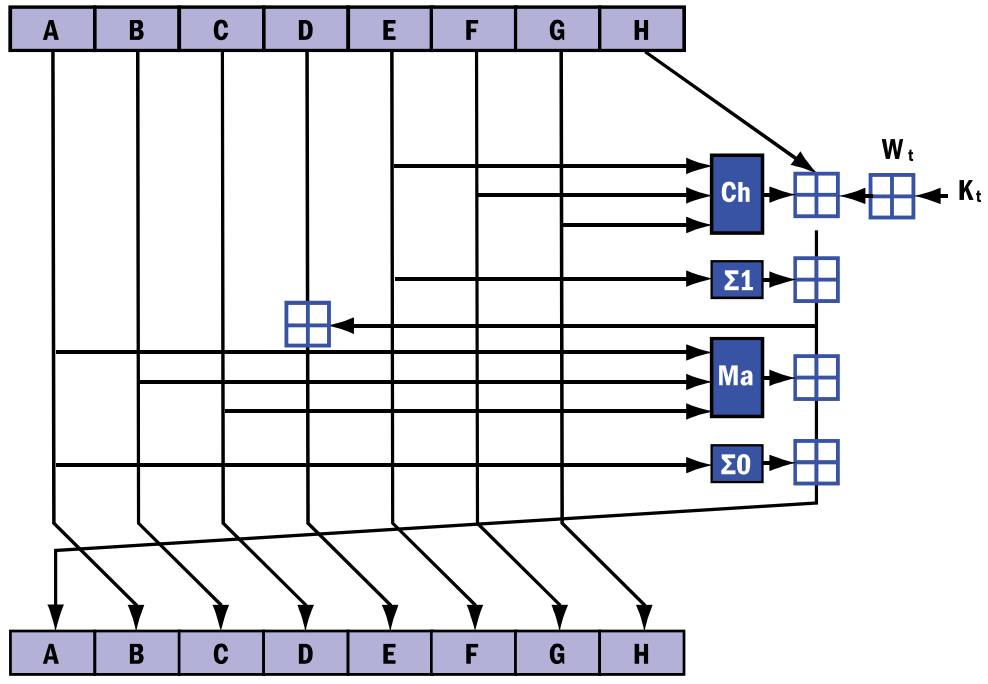


Figura 13. Cada vuelta de **SHA-2** utiliza cuatro operaciones lógicas especiales (**Ch**, **Ma**, **E0** y **E1**) para operar sobre ocho palabras (desde la **A** hasta la **H**) y aplica **sumas módulo 2^{32}**.

En 2012, **SHA-3** se transformó en el más avanzado de la familia, luego de un concurso público tal como ocurrió con DES y AES. El algoritmo ganador fue **Keccak**, diseñado por Guido Bertoni, Joan Daemen, Michael Peeters y Gilles Van Assche, sobre la primitiva llamada **RadioGatún**. Soporta las mismas longitudes de hash que su antecesor y su estructura interna difiere significativamente del resto de la familia. Utiliza una construcción llamada **esponja**, que implica una serie de estados internos finitos que procesan una secuencia de bits de entrada de cualquier longitud para producir una secuencia de salida de la longitud deseada. En Keccak se aplica una operación **XOR** a los mensajes contra los bits de un estado inicial, que luego es permutado.

En la versión para SHA-3 se utilizan vectores de 64 bits de 5x5 (1600 bits), y permite también palabras más pequeñas en potencias de 2 para ser usado en forma de alternativas más simples. Su implementación en hardware fue notablemente más rápida que la de los demás finalistas de la competencia del NIST: **BLAKE**, **Grøstl**, **JH** y **Skein**.

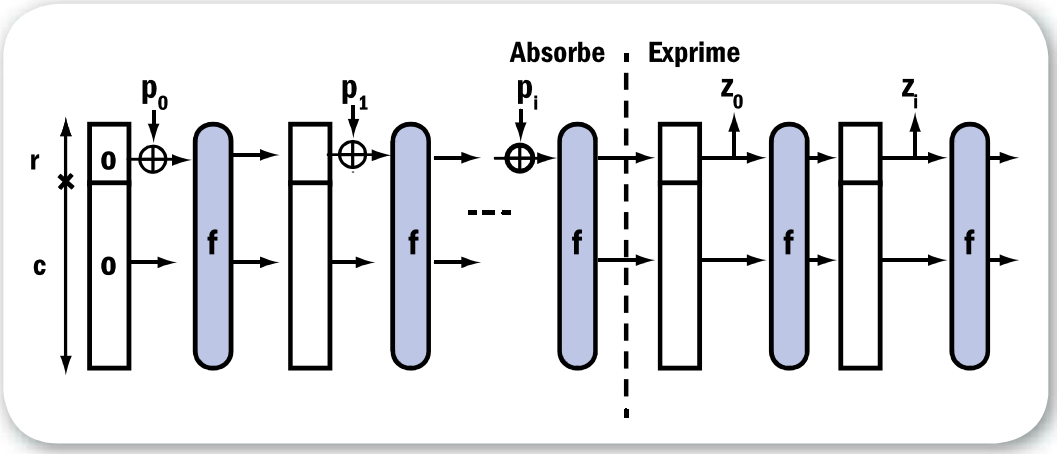


Figura 14. Construcción en esponja de **SHA-3** donde **Pi** son las entradas y **Zi** las salidas procesadas. La capacidad no usada **C** debería ser el doble de la resistencia a colisiones deseada, y **r** es el número de bits procesados por permutación de bloque (depende del tamaño del hash).

Tiger

En 1996, Ross Anderson y Eli Biham publicaron **Tiger**, un algoritmo optimizado para sistemas de 64 bits (no comunes en ese entonces). Devuelve un resumen de 192 bits y cuenta con versiones de 128 y 160

bits (**Tiger/128** y **Tiger/160**), que son en realidad **versiones truncadas**. Se basa en el esquema de **Merkle-Damgard** y aplica **árboles de Merkle**.

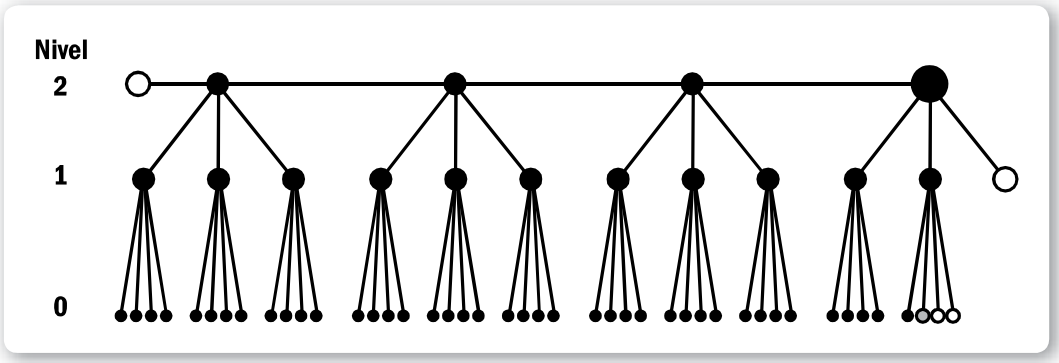


Figura 15. Tiger utiliza árboles de Merkle de una forma que se conoce como **Tiger Tree Hash** (**TTH**), que se aplica en algunos programas de P2P.

Su función de compresión opera con palabras de 64 bits, manteniendo tres palabras de estado y procesando ocho palabras de datos. Realiza 24 rondas en las que combina operaciones **XOR**, **suma/resta**, **rotación** y **cajas-S**, además de un complejo **algoritmo de derivación de claves**.

Pese a ser rápido en software, las grandes **cajas-S** (cuatro cajas de 256 entradas de 64 bits) hacen que su implementación en hardware sea complicada. Pese a sus características, no se lo incluyó en el estándar **OpenPGP**, pues se optó por **RIPEMD-160**.

Existe una variante llamada **Tiger2**, donde el mensaje se rellena mediante el agregado de un byte hexadecimal de **0x80** (como en MD4, MD5 y SHA), a diferencia del valor **0x01** de la versión original y de los valores diferentes usados en SHA-2.

ARCO IRIS AL ATAQUE

Las **tablas arcoiris** (**Rainbow Tables**) son una técnica utilizada para atacar contraseñas almacenadas como hashes en aplicaciones y sistemas operativos. La técnica consiste en precalcular todos los posibles valores resultantes de los hashes que podría arrojar el sistema, lo cual reduce el descubrimiento de contraseñas a una simple búsqueda binaria.

Whirlpool

En el año 2000, Vincent Rijmen y Paulo Barreto diseñaron **Whirlpool** (no está patentado), que fue recomendado por el proyecto **NESSIE** y adoptado como parte del estándar internacional **ISO/IEC 10118-3**. Se basa en la lógica de **AES** (cocreado por Rijmen) y **Square** (un cifrado de bloques), y utiliza una construcción **Miyaguchi-Preneel**. Entrega hashes de 512 bits de entradas que pueden llegar hasta los 2^{256} bits.

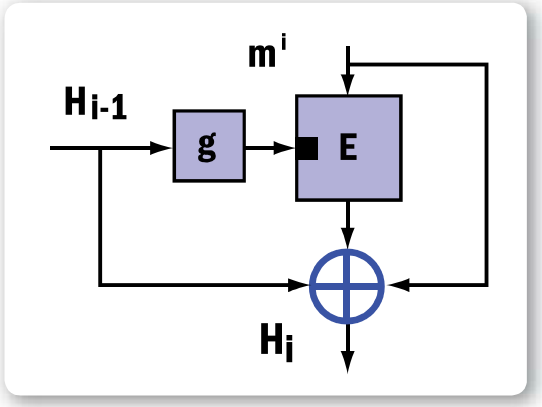


Figura 16. **Whirlpool** se basa en la función de compresión **Miyaguchi-Preneel**, donde **E** es la función de cifrado, **mi** es el mensaje, **Hi** es el valor del hash, **Hi-1** es el hash anterior y **g** es una función para adaptar tamaños de bloques.

Utiliza una matriz de estado de 8x8 y el proceso de cifrado consiste en actualizar el estado de la matriz en diez rondas. Tuvo dos revisiones desde su publicación; en la primera (2001) se cambiaron las propiedades de las **cajas-S**, que simplificaba su implementación en hardware, y en la segunda (2003) se modificó la matriz de difusión. El mejor ataque realizado fue en 2009 mediante una técnica denominada **rebound attack**, realizando 2^{120} operaciones.

NESSIE

NESSIE (**New European Schemes for Signatures, Integrity and Encryption**) fue un proyecto europeo cuyo objetivo era identificar primitivas criptográficas. Funcionó entre los años 2000 y 2003, y es comparable con el proceso de **AES** del **NIST** y el **CRYPTREC** japonés. Muchos de los más importantes criptógrafos actuales participaron del proyecto.

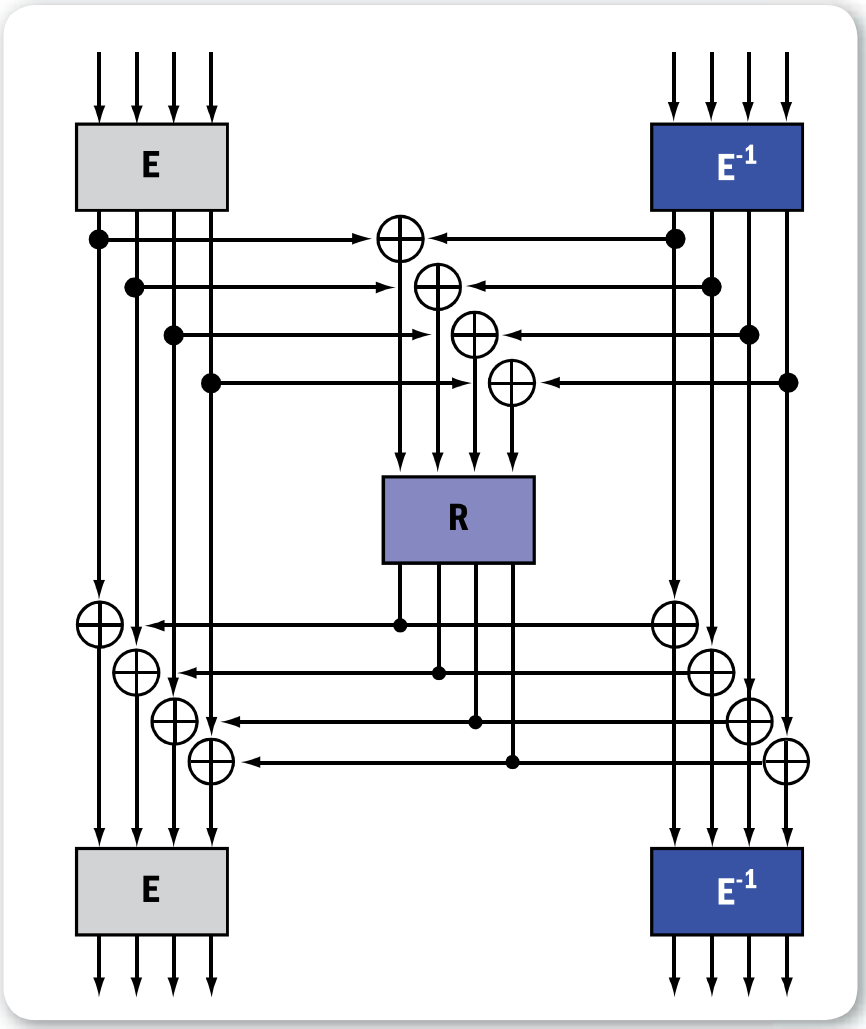


Figura 17. La función de cifrado de Whirlpool (en su última revisión) se basa en mini cajas de 4x4: la **exponencial** (**E**) y su inversa, y la **pseudo aleatoria** (**R**).

RIPEMD

En 1996, Hans Dobbertin, Antoon Bosselaers y Bart Preneel publicaron **RIPEMD** (*RACE Integrity Primitives Evaluation Message Digest*), un algoritmo no patentado basado en MD4 pero similar en funcionamiento y en robustez a SHA-1. Fue estandarizado en la misma norma ISO/IEC que Whirlpool, SHA-1 y SHA-2. Originalmente devolvía un hash de 128 bits, pero luego de unas mejoras, su versión más común fue la revisión **RIPEMD-160**. Existen versiones de mayor tamaño de resumen: 128, 256 y 320 bits (RIPEMD-128, RIPEMD-256 y RIPEMD-320), pero no ofrecen mayor seguridad sino que reducen la probabilidad de colisiones al

devolver resúmenes más largos. Debido a su menor popularidad frente a otros algoritmos, no ha recibido tanta atención en cuanto a los posibles ataques, aunque en 2004 se publicó un ataque de colisión para la versión original que no afecta a sus versiones mejoradas.

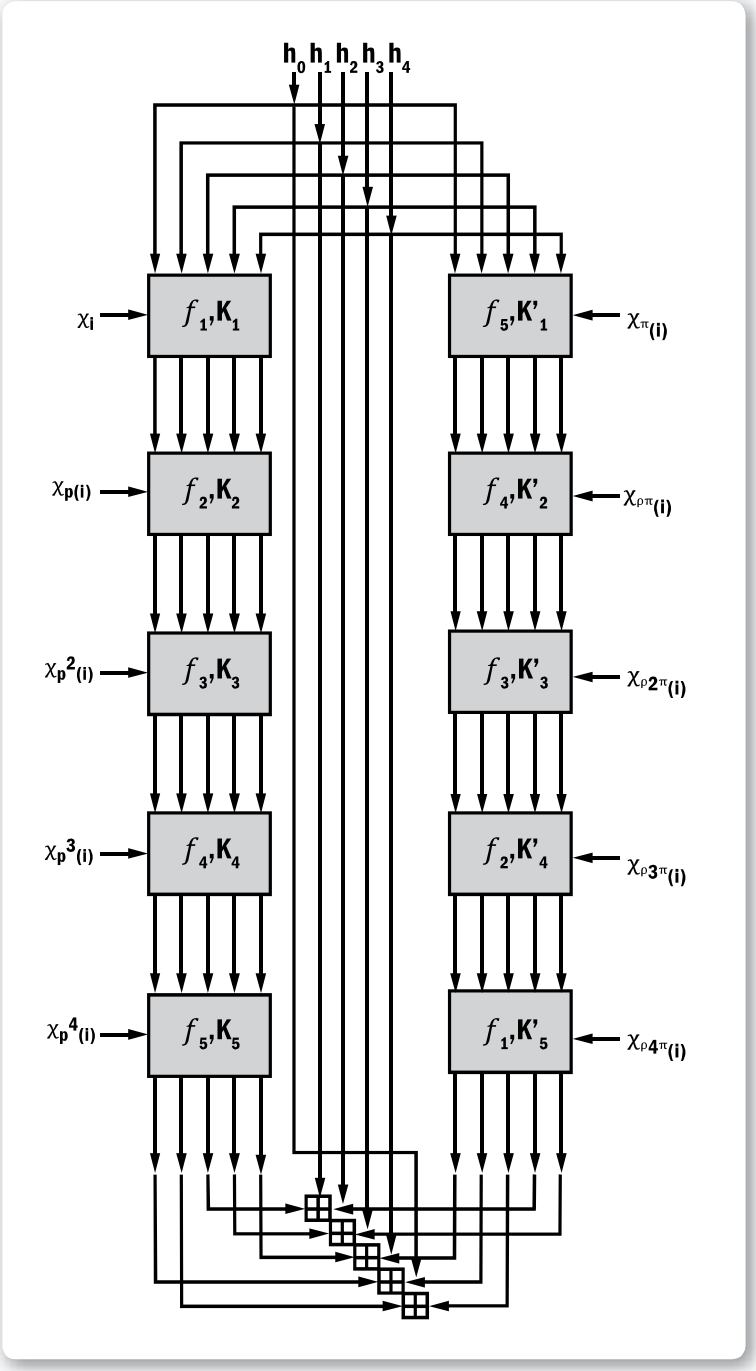


Figura 18. En **RIPEMD-160**, las entradas son bloques de 16 palabras, **Xi** (cada una con su permutación según la vuelta) y una variable encadenada de cinco palabras $\mathbf{h_0h_1h_2h_3h_4}$. La salida es un nuevo valor de la variable.

Otros

Existen otros algoritmos de hash criptográficos menos importantes, que han tenido su lugar hasta que fueron comprobados ataques satisfactorios contra ellos o porque fueron superados por otros más modernos. Algunos ejemplos son:

- **HAVAL**: creado en 1992 por Zheng, Pieprzyk y Seberry, puede producir hashes de longitud variable (128, 160, 192, 224 y 256 bits) y permite especificar la cantidad de vueltas (3 a 5). En 2004, el equipo de Wang descubrió un ataque de colisión efectivo que hizo cuestionable su uso futuro.
- **Snefru**: creado en 1990 por Ralph Merkle, fue nombrado como el **faraón egipcio**, continuando la tradición de los algoritmos **Khufu** y **Khafre** del mismo autor. Produce hashes de 128 y 256 bits, y sucumbió ante el criptoanálisis diferencial de Biham y Shamir utilizado para encontrar colisiones, por lo que fue modificado incrementando las vueltas de 2 a 8, lo que lo hizo más robusto.
- **GOST**: publicado (desclasificado) en 1994, fue creado por la agencia de comunicaciones soviética basado en el cifrado de bloques **GOST** (que era una red Feistel). Produce hashes de 256 bits, operando en 32 vueltas; sufrió un ataque de colisión en 2008 en un tiempo de 2^{105} y de preimagen en 2^{192}.

Solo a título informativo, también existen los siguientes algoritmos (en orden cronológico): **MDC-2** (1990), **N-Hash** (1990), **Panamá** (1998), la familia **FSB** (2003), **VSH** (2005), **HAS-160** (2005), la familia **SWIFFT** (2008), **ECOH** (2008), **Spectral** (2008), **CubeHash** (2008) y **Fugue** (IBM, 2009).

Códigos de autenticación

Cuando además de **integridad** se desea obtener **autenticidad** de origen aparecen los **códigos de autenticación**, que conforman primitivas criptográficas orientadas a verificar ambas a la vez. Veamos algunos de los más utilizados.

CBC-MAC

Este tipo de construcciones tiene como objetivo convertir un algoritmo de **cifrado simétrico de bloques** en una función de **comprobación de autenticidad**. Su funcionamiento es simple: se aplica el algoritmo en **modo CBC**, descartando todo el resultado excepto el bloque final, que será el dato verificador (**tag**). La interdependencia de bloques garantiza que no pueda predecirse el resultado final ante el cambio de algún bloque. Normalmente, el vector de inicialización es cero y, tal como se puede suponer, utiliza una clave secreta que se aplica al esquema de bloques.

> SI EL ALGORITMO DE BLOQUE ES SEGURO, CBC-MAC ES SEGURO PARA MENSAJES DE TAMAÑO FIJO

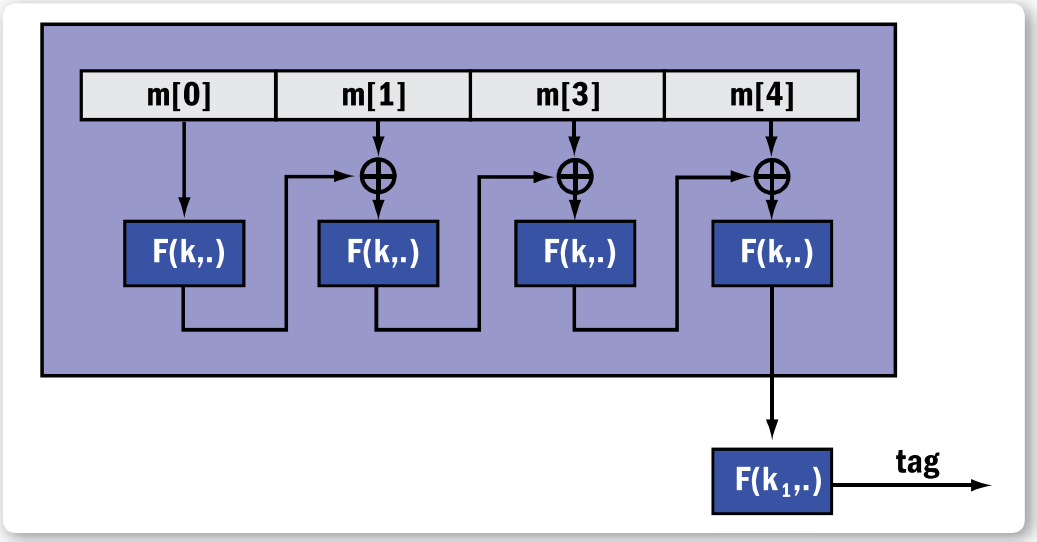


Figura 19. CBC-MAC aplica un cifrado de bloque para autenticación.

Si el algoritmo de bloque utilizado es seguro (es decir, si es una **permutación pseudoaleatoria**), entonces **CBC-MAC** es seguro

PROBLEMAS DEL CBC-MAC

La debilidad de CBC-MAC no puede resolverse por la simple adición de la información de longitud del mensaje al final del proceso, sino que hay tres enfoques: usar distintas claves (**input-length key separation**), incluir la clave en el primer bloque (**length-prepending**) y cifrar el último bloque (el más seguro).

para mensajes de tamaño fijo. Sin embargo, no es seguro por sí solo para mensajes de longitud variable, ya que si un atacante conoce los **tags t1** y **t2** de dos mensajes **m1** y **m2** producidos con la misma clave, puede generar un tercer mensaje **m3** válido cuyo **tag** sea **t2**, aplicando una **XOR** entre el primer bloque de **m2** y **t1** y concatenando **m1** con el resultado. Es decir: **m3=m1||[(m1' XOR t1)||m2'||...mx']**.

HMAC

En 1996, M. Bellare, R. Canetti, y H. Krawczyk propusieron **HMAC** –o **Keyed-Hash MAC**– que se basa en el uso de una función de hash más la aplicación de una clave secreta para generar un mecanismo de autenticación. El esquema más utilizado actualmente es **HMAC-SHA-256**, aunque **HMAC-MD5** y **HMAC-SHA-1** fueron muy populares por su uso en **IPSec** y **TLS**.

La construcción de HMAC según el **RFC 2104** es la siguiente: **HMAC(K,m)=H((K XOR opad) || H((K XOR ipad) || m))**. Donde **H** es la función de hash, **K** es la clave rellenada con ceros si es menor al bloque o aplicándole la función de hash si es mayor, **m** es el mensaje, **opad** es el rellenado de salida e **ipad** es el rellenado de entrada.

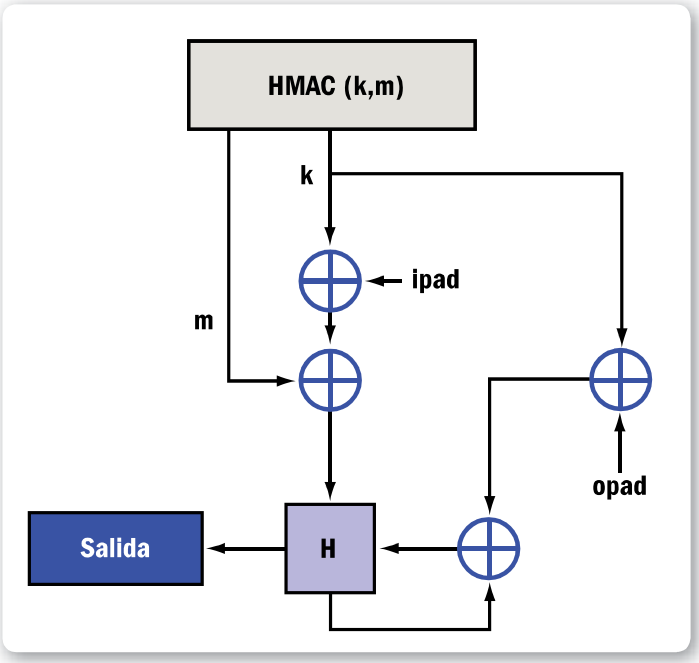


Figura 20. HMAC agrega una clave secreta al procesamiento de un algoritmo de hash.

UMAC

Se llama **UMAC** (*Universal MAC*) o MAC basado en hashing universal al método de autenticación que utiliza una función hash seleccionada de una familia de funciones en base a un proceso secreto que se aplica luego al mensaje. El resultado del procedimiento es cifrado para que no se pueda deducir la función utilizada.

Se basa en la idea de que un potencial atacante debe intercambiar información con el sistema si desea verificar la validez de su código generado, por lo que no es posible para éste realizar un ataque offline.

Otros

Además de éstos, existen otros esquemas propuestos para autenticación de mensajes, como por ejemplo:

- **CMAC** (*Cipher-based MAC*): propuesto por Black y Rogaway, es una variación de CBC-MAC que resuelve sus deficiencias para mensajes de longitud variable, aunque requiere de tres claves.
- **OMAC** (*One-key MAC*): creado por T. Iwata y K. Kurosawa, es una mejora de CMAC que usa menos información de claves. Existen oficialmente dos algoritmos, **OMAC1** (equivalente a CBC) y **OMAC2**, ambos libres de patentes.
- **PMAC** (*Parallelizable MAC*): creado y patentado por P. Rogaway, permite hacer más eficiente el método de autenticación, paralelizando el procesamiento de los bloques pero en detrimento de la seguridad ofrecida. En funcionalidad, es similar a OMAC.

RESUMEN

En este capítulo estudiamos las funciones de hash, usadas principalmente para obtener autenticación e integridad. Vimos los algoritmos más importantes de la actualidad, como MD5 y SHA-1. Además, vimos otros menos utilizados como Tiger, Haval, RIPEMD y Whirlpool. También analizamos la posibilidad de utilizar algoritmos de hash como parte de códigos de autenticación de mensajes mediante códigos MAC. Finalmente, vimos las técnicas generales de ataque contra algoritmos de hash.

Actividades

TEST DE AUTOEVALUACIÓN

1. ¿Qué es una función hash y cuál es su importancia en la criptografía?
2. ¿Por qué se aplican algoritmos de hash a las contraseñas de un sistema operativo?
3. ¿Por qué existen funciones de hash no criptográficas?
4. ¿Qué ventajas y desventajas tiene SHA-1 respecto a MD5?
5. Además de MD5 y SHA-1, ¿qué otros algoritmos de hash se usan?
6. ¿Cómo se consigue verificar la integridad con el uso de un hash?
7. ¿Cómo se ataca una función hash?

EJERCICIOS PRÁCTICOS

1. Analice las especificaciones vigentes del estándar SHA-3 y compárelo con el resto de la familia SHA.
2. Investigue en qué campos no se han utilizado MD5 y SHA-1 en favor de otros algoritmos.
3. Pruebe el rendimiento de distintos algoritmos de hash utilizando la misma computadora y un software para tal fin.
4. Investigue las ventajas del uso de Rainbow Tables para ataques a hashes.
5. Verifique la paradoja del cumpleaños entre distintos grupos de personas.

PROFESOR EN LÍNEA

Si tiene alguna consulta técnica relacionada con el contenido, puede contactarse con nuestros expertos: profesor@redusers.com

*07

Public Key Infrastructure (PKI)

Una infraestructura de clave pública es un conjunto de tecnologías (hardware y software), políticas y procedimientos que se usan para cifrar y firmar digitalmente con garantías de seguridad criptográfica. En este capítulo veremos sus elementos y aplicaciones.

Estructura PKI

Además de su utilización en firmado digital, las **PKI** (*Public Key Infrastructure*) pueden aprovecharse para la autenticación (usuarios y aplicaciones) e identificación de partes en una comunicación, para el cifrado y autenticación de documentos o emails, la garantía de confidencialidad en redes y la obtención del no repudio. Todo esto, sin necesidad de claves secretas (cifrado simétrico).

Los procesos de una **PKI** constan de al menos tres actores: un usuario iniciador de la operación, un sistema que da constancia de la operación y garantiza la validez de los certificados utilizados, y un usuario destinatario del procedimiento. Además, es importante la aplicación de políticas y procedimientos que garanticen el correcto funcionamiento más allá de la fortaleza técnica.

Componentes y autoridades

Una PKI se compone de distintos elementos, entre los que se encuentran los **poseedores** (usuarios) de pares de claves y certificados asociados a la clave pública, y los **repositorios**, que almacenan la información de la PKI. Hay repositorios de al menos dos tipos: **de certificados** y **de listas de revocación**, que son las que contienen las referencias a los certificados inválidos. Además, existen entidades que llamamos **autoridades**, entre las que encontramos:

- **Autoridad Certificante** (**AC**) o **de certificación**: emite y revoca certificados, siendo quien legitima la relación entre un usuario y una clave pública.
- **Autoridad de Registro** (**AR**): verifica el vínculo entre el certificado y la identidad del titular.

TERCERA PARTE DE CONFIANZA

En criptografía, una **tercera parte de confianza**, o **TTP** (**trusted third party**) es una entidad que facilita las interacciones entre dos partes que no confían entre sí pero confían en la entidad. La TTP revisa las transacciones críticas entre partes para evitar fraude; las **autoridades de certificación** son el clásico ejemplo de TTP.

- **Autoridad de Validación** (**AV**): comprueba la validez de los certificados.
- **Autoridad de sellado de tiempo** (**TSA**, *TimeStamp Authority*): firma los documentos para probar su existencia en un momento determinado.

Una adecuada política de certificación debe definir el ámbito de actuación, la relación entre las AC y el procedimiento de emisión y validación de certificados.

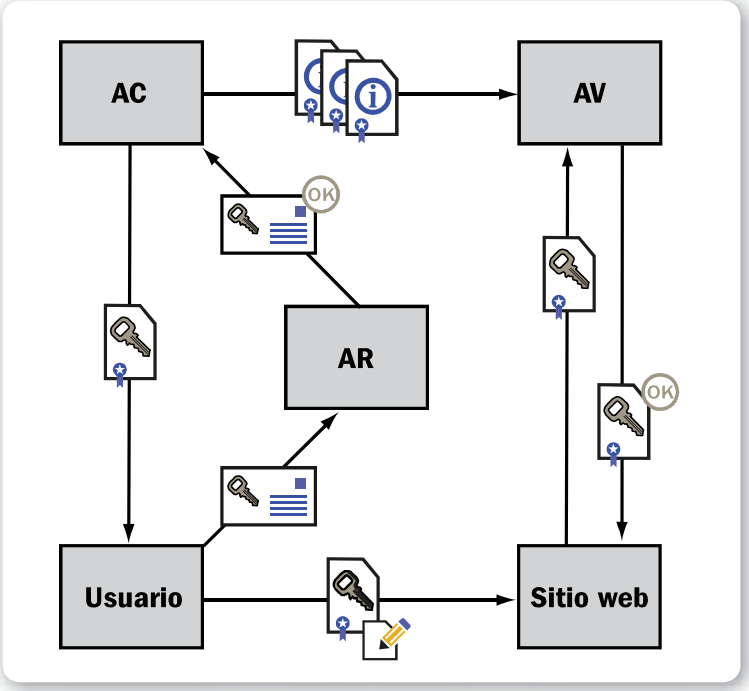


Figura 1. Esquema de las distintas autoridades que se pueden encontrar en una arquitectura **PKI**.

Certificados digitales

Se llama **certificado digital** (o **electrónico**) a un documento con un formato especial, firmado digitalmente por una autoridad certificante, que permite vincular a una entidad con los datos de verificación de una firma (incluidos en el certificado), validando así su identidad.

Hay certificados digitales de distintos tipos, en función de la información que contienen y la entidad que relacionan. Aunque los estándares PKI no los especifican, **VeriSign** define **clases** que determina tal como vemos en la página siguiente.

- **Clase 1**: para individuos (orientado al email).
- **Clase 2**: para organizaciones (requiere prueba de identidad).
- **Clase 3**: para servidores y firma de software (requiere distintas verificaciones).
- **Clase 4**: para transacciones comerciales online entre empresas.
- **Clase 5**: para organizaciones privadas o seguridad gubernamental.

El uso más común de los certificados digitales se encuentra en los servidores web con HTTPS, que permiten que un navegador valide su autenticidad de cara al usuario, algo fundamental en el comercio electrónico y también en las aplicaciones que manejan datos financieros.

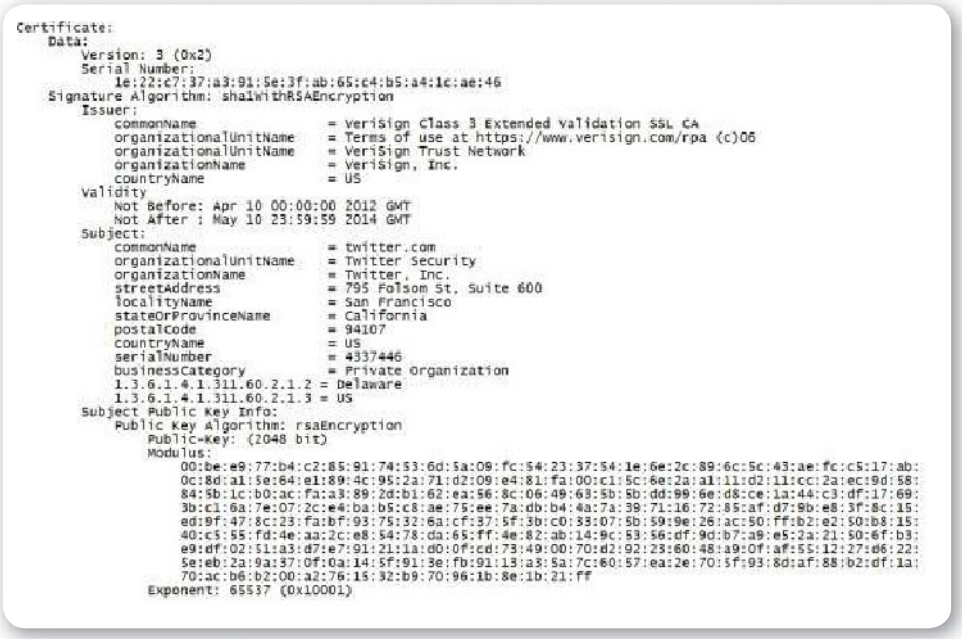


Figura 2. Ejemplo de un certificado decodificado de Twitter otorgado por VeriSign.

X.500

Es una serie de estándares **ITU-T** sobre servicios de directorio para redes. Fue desarrollado en colaboración con ISO para incorporarlo en el modelo OSI. Incluye la definición de varios protocolos, entre los que se encuentran el de acceso a directorios (**DAP**) –cuya versión liviana (**LDAP**) terminó por reemplazarlo– y **X.509**, orientado a infraestructuras de clave pública.

CRL

Para verificar si los certificados son válidos o están revocados, PKI cuenta con las **CRL** (*Certificate Revocation List*), que son listas de números de serie de certificados que fueron revocados por una autoridad (ya no son válidos).

Normalmente, los certificados se emiten con un período de validez determinado para reducir el riesgo de que sea falsificado (la fecha se incluye en el mismo certificado). Además de la caducidad, un certificado puede ser invalidado por otros motivos, como por ejemplo que la clave privada haya sido perdida o robada, que se pierda la condición que permitió expedirlo, que algún dato sea erróneo, por orden judicial o por pedido del usuario.

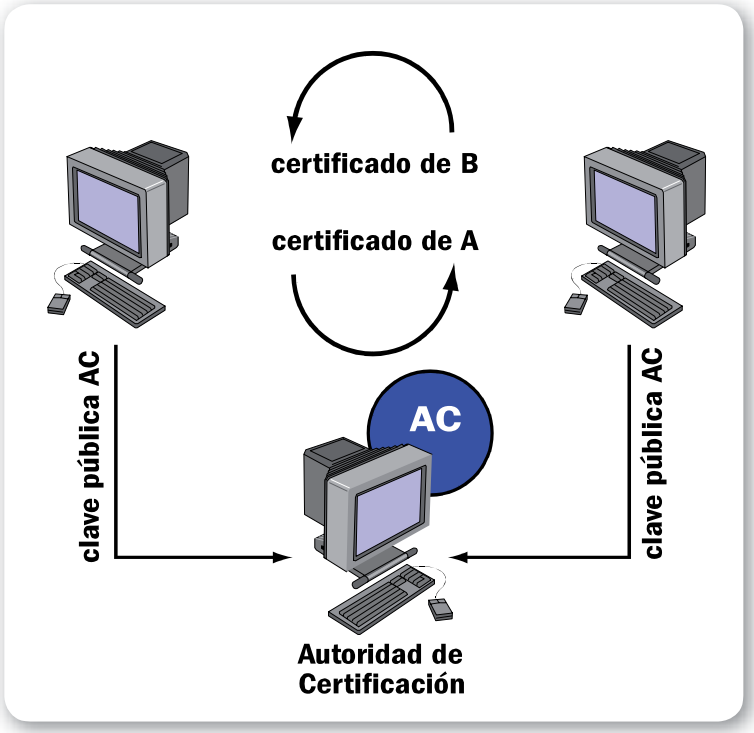


Figura 3. El usuario **A** le envía a **B** su certificado (clave pública firmada por **AC**) y éste comprueba con esa autoridad su autenticidad. Lo mismo sucede en sentido contrario.

Una CRL debe estar firmada digitalmente por la CA que la emite para constatar su validez, y quien necesite validar un certificado descargará la lista actualizada desde el servidor de la autoridad emisora y comprobará si se encuentra allí. No se requiere actualizar la CRL en cada verificación sino solo en determinadas circunstancias o cuando ha

pasado el tiempo aconsejable. Dado este comportamiento asincrónico, un certificado podría estar revocado pero aún no aparecer en la CRL si no se actualizó, y esta ventana de no confianza podría llevar a firmar con un certificado no válido.

El método no es eficiente a largo plazo ya que la lista aumenta de tamaño indefinidamente, por lo que surge una alternativa para comprobar la validez de certificados mediante el **protocolo OCSP**.

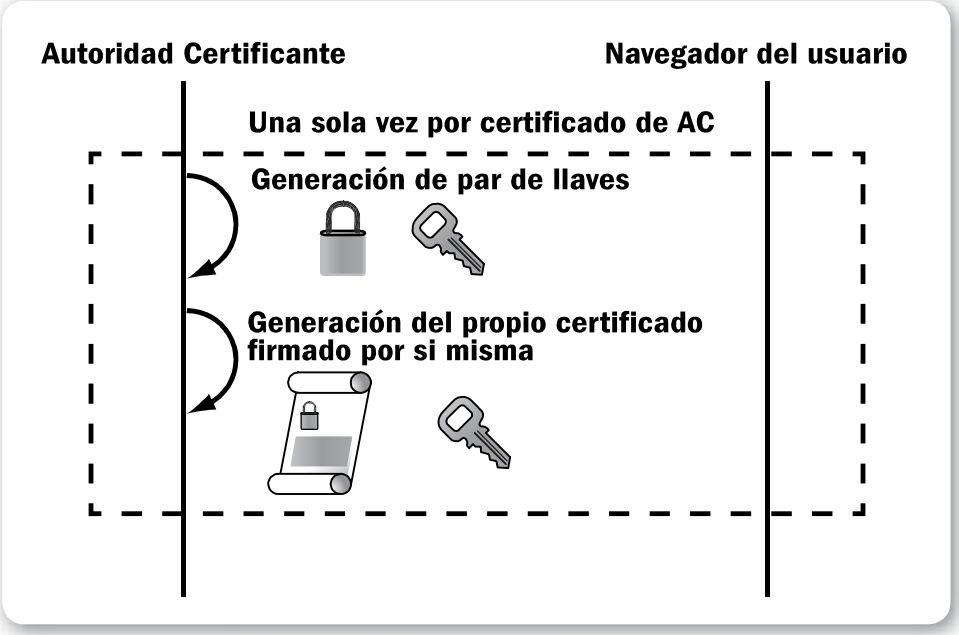


Figura 4. La **AC** debe firmar su propio certificado, que luego será verificado por el usuario.

OCSP

Definido en el **RFC 6960**, el protocolo **OCSP** (*Online Certificate Status Protocol*) permite consultar validez en tiempo real sin tener que descargar una CRL, con la desventaja de que se requiere estar online. OCSP permite determinar el estado de validez de un certificado **X.509** utilizando mensajes de HTTP. A los servidores OCSP se los conoce como **OCSP responders**, y retornan una respuesta positiva si el certificado es válido, revocado o desconocido, y un código de error si no puede procesar el pedido.

Además de ser más eficiente que las CRL, permite comprobar de a un certificado en particular sin necesidad de acceder a toda la información de una lista. También permite implementar cadenas de confianza,

soportar más de un nivel de CA, encadenar pedidos entre **responders** y utilizar extensiones.

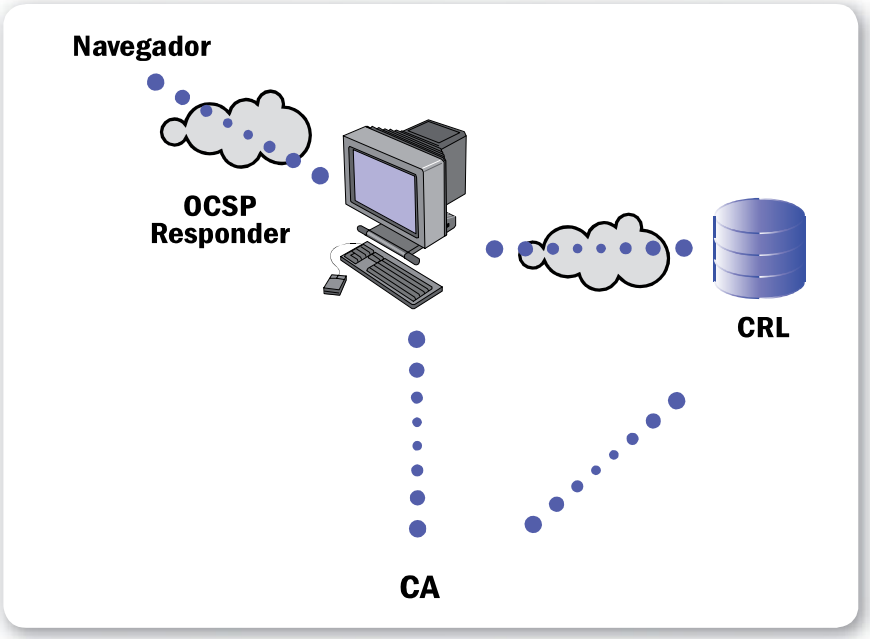


Figura 5. Esquema del protocolo **OCSP** y los participantes.

Una **petición OCSP** se compone de identificadores de los certificados a validar; formados por su número de serie, el hash del *Distinguished Name* (**DN**) del emisor y el hash de su clave pública. Una petición puede incluir consultas de más de un certificado aunque pertenezcan a distintas CA. El firmado de las peticiones no es obligatorio, sino que depende de la autoridad de validación OCSP.

En cuanto a su seguridad, es vulnerable a **ataque de repetición** (**replay attacks**) ya que si se intercepta un mensaje de "certificado válido" puede ser repetido en otro momento. Se propuso resolver esto con el uso de **nonces** como valor testigo (**token**), pero no fue aceptado porque no siempre se utilizan las extensiones para nonces.

Dado que las CA deben responder en tiempo real a todos los clientes que soliciten peticiones de OCSP, en sitios web de alto tráfico puede resultar ineficiente. Además, el chequeo introduce cuestiones de privacidad, dado que requiere que el cliente contacte a un tercero para realizar la validación en lugar de verificarlo directamente de la contraparte. Para esto se adaptó el protocolo y se conformó **OCSP stapling**, que resuelve ambos problemas haciendo que el dueño del certificado sea quien consulte al **OCSP responder** a intervalos regulares

de tiempo, obteniendo respuestas firmadas y con **timestamp**. Cuando un cliente se conecta, la respuesta se incluye en el **TLS handshake** por medio de la extensión **TLS CSR** (*Certificate Status Request*). Así también se protegen los hábitos de navegación del usuario.

Como la respuesta es firmada por la CA y no por el dueño del certificado, se evita el fraude de autoverificación. La extensión **TLS CSR** (**OCSP stapling**) se especifica en el **RFC 6066**. Como limitación, podemos decir que solo soporta una respuesta por vez, lo que no es suficiente para sitios que usan diferentes certificados para una página.

PKI y la seguridad

Algunos renombrados expertos como Bruce Schneier y Peter Gutmann han estudiado los problemas de seguridad relativos a una PKI. Entre los temas principales se destacan:

- **Complejidad de los certificados**: por un lado, los estándares cuentan con excesivas funcionalidades no utilizadas ni relevantes. Por otro lado, la gran mayoría de los usuarios comunes de internet carecen de habilidades para comprender los riesgos técnicos de la criptografía. A fin de simplificar la complejidad, las empresas suelen incluir agregados propios de manera automática y transparente, que luego funcionan como brecha para posibles atacantes.
- **Debilidad estructural**: temas relacionados a la propia arquitectura de una PKI, como el uso de listas negras para determinar la validez de los certificados (en general las listas blancas son lo más prudente en seguridad). Además, no se cuenta con estado de revocación histórico y pueden combinarse datos de usuarios y entidades para obtener nueva información. En ciertas circunstancias, las cadenas

de certificados pueden llevar a una validación costosa en términos de tiempo, y puede haber inconvenientes cuando existe confianza bilateral por ser PKI estrictamente jerárquica.

- **Problemas con las autoridades**: dado que se puede elegir la AC, muchas veces se decide por precio y no por calidad. Además, las licencias suelen denegar muchas garantías y existen dudas acerca de la fortaleza de la clave en función del tiempo de expiración.
- **Problemas de implementación**: las fallas de diseño, bugs de software, temas de interoperabilidad e interpretaciones del estándar son problemas en sí mismos.
- **Debilidades de los algoritmos**: en 2005, A. Lenstra y B. de Weger demostraron (basándose en colisiones de MD5) que podían crear certificados con firmas idénticas pero diferente clave pública. En 2007, junto a M. Stevens, probaron que podían agregar sufijos a dos archivos distintos y producir firmas idénticas. En 2008, Alexander Sotirov y Stevens demostraron que podían crear una AC aceptada por los navegadores. En 2009 continuaron las demostraciones de la mano de Michael Zusman, Sotirov y Dan Kaminsky.

> LAS DEBILIDADES EN LOS ALGORITMOS LOGRARON PONER EN JAQUE A LA ESTRUCTURA PKI

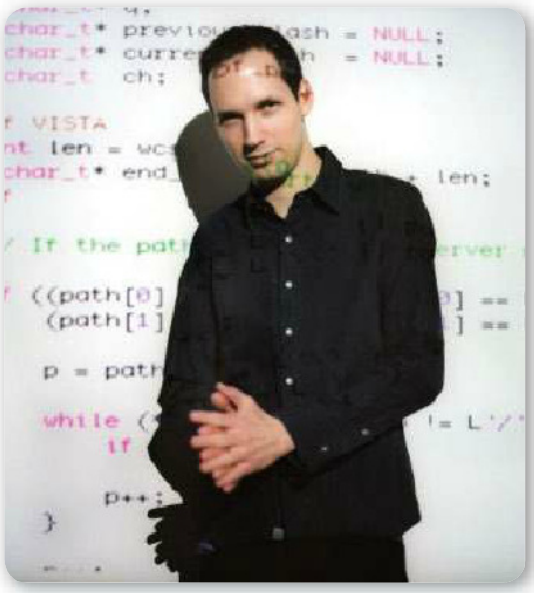


Figura 6. Alex Sotirov es uno de los investigadores de seguridad que pusieron en jaque a los certificados digitales.

Estándar X.509

Relacionado con el estándar **X.500** de la **ITU**, o en español **UIT-T** (Sector de Normalización de las Telecomunicaciones de la **UIT**), **X.509** corresponde a un estándar para las PKI creado en 1988, que especifica formatos de certificados y algoritmos de comprobación de ruta de certificación. La última versión es la 3 (2008).

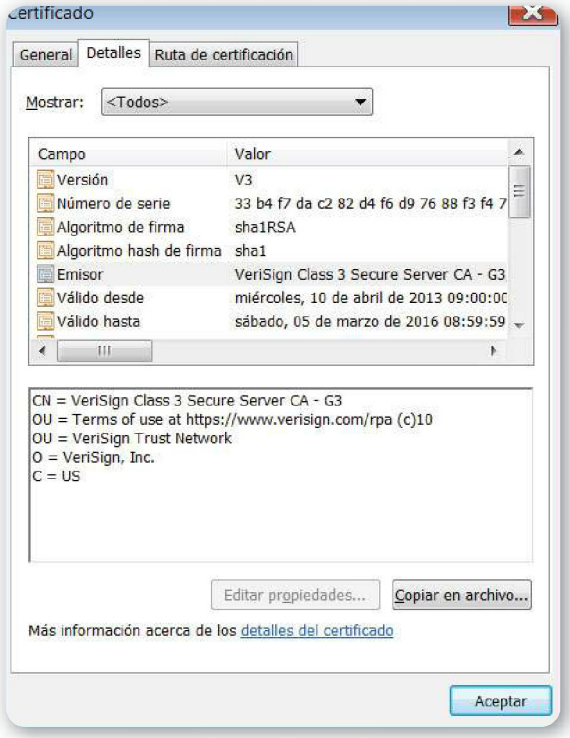


Figura 7. Detalle de un certificado digital **X.509** en **Chrome**.

X.509 considera la existencia de autoridades certificantes conformando un **sistema jerárquico estricto**, lo opuesto a modelos que se basan en la confianza (como **Web of Trust**) y permiten que los nodos puedan validar certificados. Se basa en criptografía de clave pública y define un esquema para brindar servicios de autenticación por medio de certificados a usuarios de directorios X.500. Los certificados se cifran con la clave privada de la AC y cualquier usuario puede verificarlos con la pública. X.509 estandariza el uso de CRL, aunque luego define OCSP para validar certificados.

X.500 nunca fue implementado del todo y, de hecho, el grupo de **IETF** que trabajaba en el (**PKIX**) debió realizar una adaptación del estándar para crear X.509.

Figura 8. Estructura de un certificado **X.509**.

Los certificados X.509 pueden importarse y exportarse entre distintos formatos. Algunos de ellos son:

- Intercambio de información personal **PKCS #12** (**.p12**): permite almacenar certificados y claves privadas (cifradas). Evolucionó de **.pfx** (*Personal inFormation eXchange*).
- Estándar de sintaxis de cifrado de mensajes **PKCS #7** (**.p7b** y **.p7c**): sin datos (solo certificados o **CRLs**).
- **DER** (*Distinguished Encoding Rules*) binario codificado X.509 (**.der** y **.cer**): admite almacenamiento de un certificado único, sin clave privada ni ruta de certificación.

MODELO DE ORÁCULO ALEATORIO

Es un método para generar elementos de seguridad en protocolos criptográficos, modelizando algoritmos a modo de oráculos (**cajas negras**) que responden a pedidos con respuestas aleatorias. Fue propuesto por Bellare y Rogaway y se usa, por ejemplo, en esquemas de relleno (**padding oracle**).

- **X.509** codificado **Base64**: similar al anterior, con diferente codificación.
- **PEM** (*Privacy-enhanced Electronic Mail*): **DER** codificado en **Base64**. Puede incluir certificados o claves entre las líneas **BEGIN** y **END**. Extensiones **.crt** y .**pem**.

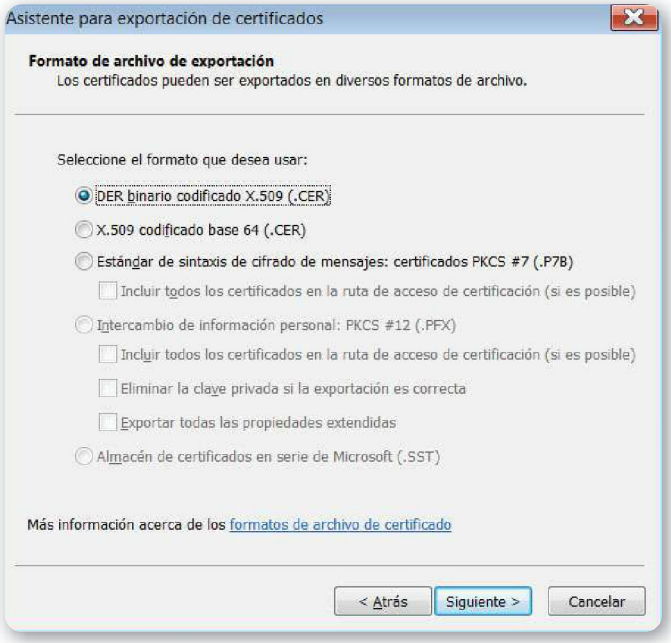


Figura 9. Es posible importar y exportar certificados usando distintos formatos.

PKCS

PKCS (*Public-Key Cryptography Standards*) es un conjunto de especificaciones técnicas vinculadas a la criptografía de clave pública. Fue creado en 1991 por la empresa **RSA Security**, que en 2006 fue adquirida por la compañía **EMC Corporation**.

MUCHAS CONSIDERACIONES

La firma digital obliga a considerar muchos temas relacionados con el cifrado asimétrico. A nivel matemático, serán tanto para la selección de los números primos como para las operaciones que se realizan. Esto lleva al estudio de las claves privadas parejas, los números no cifrables, el cifrado cíclico y los primos seguros.

Figura 10. En el sitio de **RSA Laboratories** se ofrece la información oficial de **PKCS**.

PKCS forma parte de importantes estándares como **ANSI PKIX**, **X9**, **SET**, **S/MIME** y **SSL**, aunque no fueron considerados estándares de la industria por estar bajo el control de sus creadores en un ámbito privado. No obstante, en los últimos años, algunos de los estándares PKCS empezaron a ser incluidos en las organizaciones de estandarización como **IETF**.

PKCS		
#	Versión	Qué estandariza
PKCS#1	2.1	Algoritmo RSA
PKCS#2	-	Obsoleto - Incluído en #1
PKCS#3	1.4	Intercambio de claves Diffie-Hellman
PKCS#4	-	Obsoleto - Incluído en #1
PKCS#5	2.0	Cifrado basado en contraseñas
PKCS#6	1.5	Sintaxis de certificados extendidos (obsoleto por X.509v3)
PKCS#7	1.5	Sintaxis de mensajes criptográficos en PKI
PKCS#8	1.2	Sintaxis de información de clave privada
PKCS#9	2.0	Tipos de atributos seleccionados
PKCS#10	1.7	Solicitud de certificación de una clave pública
PKCS#11	2.20	Interfaz (API) de dispositivos criptográficos
PKCS#12	1.0	Sintaxis de intercambio de información personal
PKCS#13	En desarrollo	Criptografía de curva elíptica
PKCS#14	En desarrollo	Generación de números pseudo-aleatorios
PKCS#15	1.1	Formato de información de dispositivos criptográficos

Figura 11. Tabla de estándares **PKCS** donde podemos ver su versión actual y qué es lo que estandariza cada uno.

La firma digital

Uno de los objetivos principales del establecimiento de una PKI es posibilitar el firmado digital (**FD**) de un mensaje o documento, lo que se realiza por medio de algoritmos y protocolos criptográficos. El mecanismo permite determinar al receptor que el emisor es quien dice ser (autenticación y no repudio de origen) y verificar que el mensaje no ha sufrido modificaciones desde su firmado (integridad). Esto puede aplicarse en autenticidad de software, envío de documentos para procedimientos legales, transacciones electrónicas comerciales, voto electrónico, factura electrónica y mucho más.

Figura 12. La firma hológrafa ha sido el mecanismo histórico más usado para verificar identidad y manifestar la voluntad.

Considerando que una primitiva criptográfica no provee necesariamente seguridad en sí misma, se requiere conformar esquemas, que constan de un **algoritmo de generación** de firmas, uno de **firmado** y otro de **verificación**. Para la verificación se debe contar con el algoritmo en cuestión y comprobar que la clave del firmante sea válida. La validez suele estar limitada a un período determinado de tiempo, comprobable por la fecha de caducidad o por métodos de no revocación. También es importante verificar el **sellado de tiempo** (**timestamp**), que permite fijar el momento en el que se firmó para que luego sea decidida la validez en función de ese sello. Por ejemplo, un documento firmado podría ser considerado válido solo durante 2 horas luego de haberse firmado.

En cuanto a los métodos de los protocolos de FD, se puede distinguir entre **sistemas con árbitro** (tercero de confianza) y **sistemas sin árbitro** (se valida todo entre ambas partes). La FD puede utilizar **información aleatoria**, lo que obliga a tener una fuente de aleatoriedad segura, aunque resulta mejor en cuanto al secreto de la clave; o bien puede usar **información determinista**.

Figura 13. El firmado digital puede realizarse también con dispositivos físicos, llamados normalmente **tokens criptográficos**.

Otro criterio para la firma digital es si se requiere el mensaje original para verificarla o no. Hablamos de **esquemas con recuperación de mensaje** cuando no requieren el original pues se recupera con la firma, y de **esquemas con apéndice** cuando implica la aplicación de una función de **hash**, que representará el mensaje.

> ENTRE LOS MÉTODOS DE LOS PROTOCOLOS DE FD, HAY SISTEMAS CON ÁRBITRO Y SIN ÁRBITRO

El firmado digital se remonta a 1976 con W. Diffie y M. Hellman, quienes describieron el concepto y sus conjeturas asociadas. Al tiempo nació **RSA**, que no solo servía para firmar y requirió de la creación de un esquema, que fue el aplicado en el primer software comercial que ofreció firmado digital: **Lotus Notes 1.0** (**1989**).

FIRMAS SIMÉTRICAS

Pese a la existencia del cifrado asimétrico, hay esquemas basados en cifrado simétrico: **Rabin simétrico**, **Lamport-Diffie**, **Desmedt** y **Matyas-Meyer**. Su desventaja es el tamaño de claves y su baja reutilización. R. Merkle propuso algunas mejoras y D. Bleichenbacher y U. Marer generalizaron los métodos.

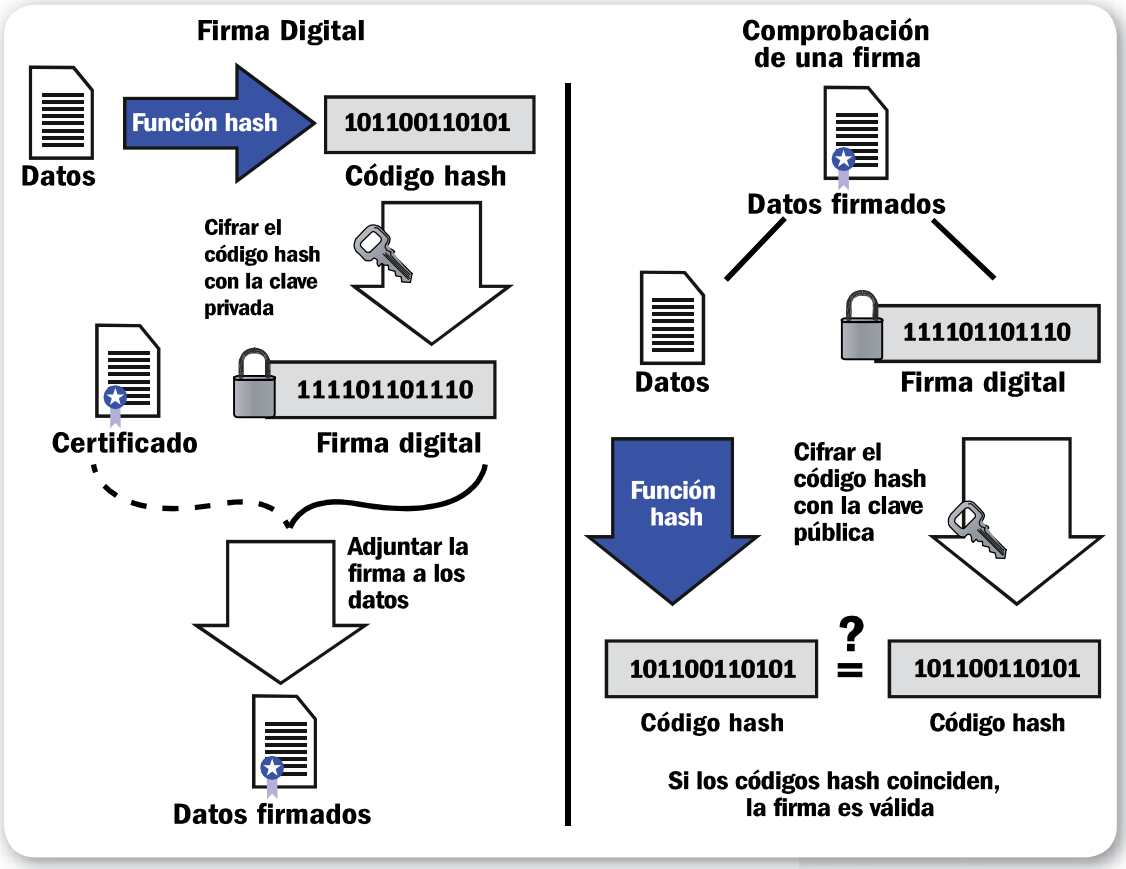


Figura 14. Esquema general de firmado de un mensaje y comprobación del firmado digital.

Pronto aparecieron otros esquemas como el de Lamport, el de Merkle y el de Rabin, pero no fue sino hasta 1988 cuando surgió **GMR**, el primer esquema que definió rigurosamente los requerimientos de firmado. Fue creado por S. Goldwasser, S. Micali y R. Rivest, describía una **jerarquía de modelos de ataque** y se presentaba como el primero en poder probar la prevención contra **falsificación existencial** (**existential forgery**) en ataques de texto plano escogido.

SEGURIDAD HACIA ADELANTE

R. Anderson propuso en 2002 las **firmas seguras hacia adelante** (**forward secure signatures**), que adicionan a los algoritmos de firma un algoritmo de actualización de claves de forma que la clave privada pueda cambiarse mientras la pública permanezca invariante, y que el conocimiento de una clave en un momento no provea información para la averiguación de otras.

Para que sea aceptable en un criptosistema, una firma digital debe ser fácil de generar, infalsificable, no rechazable, solo posible de generar por el dueño, fácil de autenticar y dependiente tanto del mensaje como del autor.

Esquemas de firma

A continuación veremos los tres esquemas de firma más comunes y una breve descripción de algunos otros. No es casual que los tres esquemas pertenezcan a la criptografía asimétrica.

Firma RSA

Un esquema de firma RSA es cualquiera que esté basado en el **problema RSA**. De hecho, para poder funcionar de manera segura, debe combinarse el algoritmo con algún **esquema de relleno** (**padding scheme**). El primer esquema utilizado fue el desarrollado en la versión 1.5 del estándar **PKCS #1**, conocido como **RSA-SSA-PKCS1** (*RSA Signature Scheme Algorithm*). El más actual es de 2012, está basado en el esquema probabilístico de M. Bellare y P. Rogaway y es conocido como **RSA-SSA-PSS** (*Probabilistic Signature Scheme*).

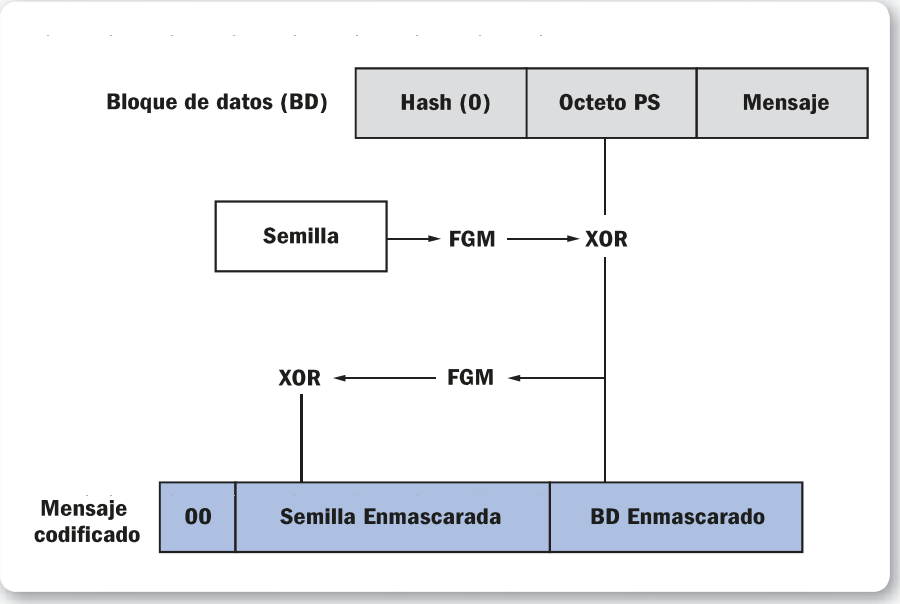


Figura 15. Esquema de relleno **OAEP** (**Optimal Asymmetric Encryption Padding**) usado en **RSA**; donde **PS** es un octeto especialmente construido y **FGM** es una función de generación de máscara.

Si bien hemos visto el esquema de generación de firmas de RSA en el **capítulo 5**, vale aclarar que en este caso, el mensaje cifrado corresponderá al hash del mensaje que se desea firmar, y la firma se considera válida si dicho hash coincide con el obtenido por el destinatario al descifrar el hash y compararlo con el del documento con el cual se produjo.

Firma ElGamal

El **esquema ElGamal** se basa en el uso del algoritmo de cifrado del mismo nombre, aunque en la práctica se utiliza más una variante (DSA).

Para generar las claves se opera de manera análoga a la expuesta en la explicación del funcionamiento del algoritmo ElGamal en dos etapas: la **elección de parámetros compartidos** y el **cálculo de las claves** en sí. Los parámetros incluyen una función de hash **H**, un número primo **p** grande (cuyo cálculo de logaritmos discretos **módulo p** sea dificultoso) y un **generador pseudoaleatorio g** para el **grupo multiplicativo Zp***. Así, obtenemos (**p**,**g**,**y**) donde **y** es la clave pública.

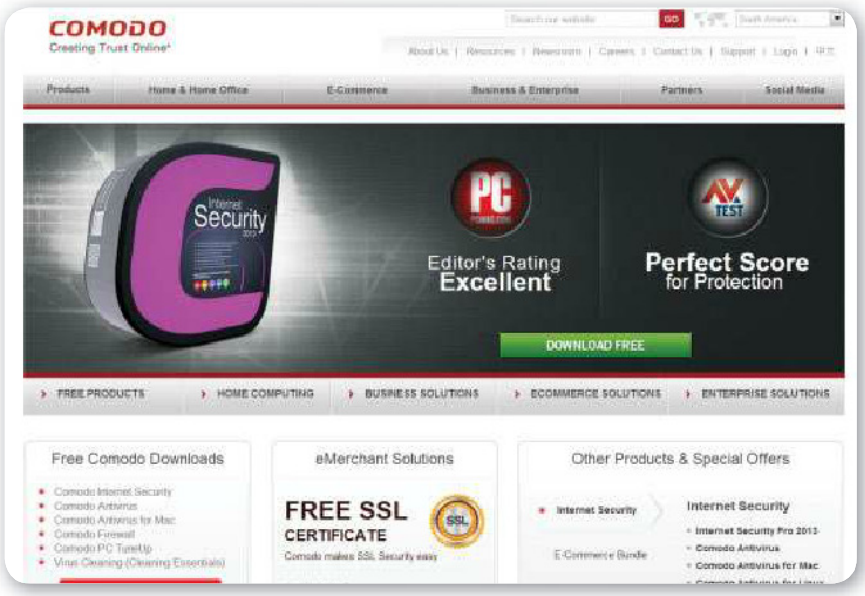


Figura 16. Comodo Group es una de las AC más importantes del mundo. En 2011 sufrió un incidente de seguridad por el que debió revocar nueve certificados.

Para generar una firma se elige un número aleatorio **k** mayor que **0** y menor que **(p-1)** y que no tenga divisores con **(p-1)**; y luego se calcula

r≡gk mod p y **s≡(H(m)- x r) k-1 mod (p-1)** (si da cero se repite).

Así, el par **(r,s)** obtenido corresponde a la firma de **m**. Para verificarlo, se debe comprobar que **0<r<p**, **0<s<(p-1)** y **gH(m)≡yr rs mod p**. De esta manera, quien verifica siempre aceptará firmas válidas, lo que puede demostrarse mediante el **Pequeño Teorema de Fermat**.

Recordemos que ElGamal duplica el tamaño del mensaje, lo que se resuelve con **DSA**. Para mejorar la seguridad, el firmante deberá elegir un valor uniformemente aleatorio diferente para cada operación de firma, a fin de evitar que se pueda deducir información sobre la clave privada.

Firma DSA

DSA (*Digital Signature Algorithm*) es un estándar **FIPS** (*Federal Information Processing Standard*) para firmas digitales del gobierno de Estados Unidos. Fue propuesto por el NIST en 1991 para su estándar **DSS** (*Digital Signature Standard*), cuya última revisión fue en julio de 2013 (**FIPS 186**). No se usa para cifrado sino solo para firmado, y requiere más tiempo de cálculo que RSA.

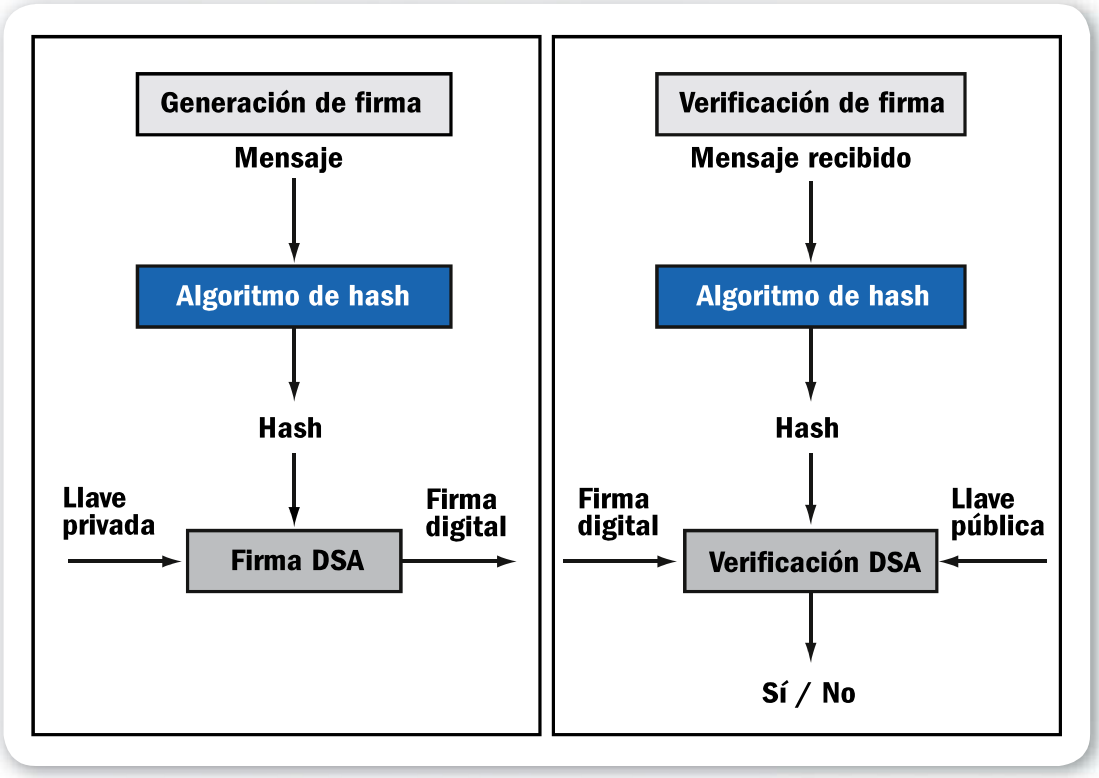


Figura 17. Esquema de firma **DSA**.

Para la generación de claves se realiza lo siguiente:

- Seleccionar un número primo **p** de **L** bits, divisible por 64 que sea mayor a 512 y menor a 1024.
- Seleccionar un número primo **q** de 160 bits, tal que **p−1=qz** (**z** es un número natural).
- Seleccionar un **h** mayor que **1** y menor que **(p−1)**, tal que **g=hz(mod p)>1**.
- Seleccionar aleatoriamente un **x** mayor que **1** y menos que **(q-1)**.
- Calcular **y=gx(mod p)**.

Luego, los valores **p**, **q**, **g** e **y** son los **datos públicos**, y **x** es el valor de la **clave privada**. Para **firmar** se selecciona un **número aleatorio s** mayor que **1** y menor que **q** y se calcula:

- **s1=(gs mod p)mod q** (si **s1** se anula no existe inverso y debe repetirse el proceso).
- **s2=s-1(H(M)+s1*x)mod q**, donde **H(M)** es el hash del mensaje **M** (si **s2** se anula no existe inverso y debe repetirse el proceso).

El par (**s1**, **s2**) resulta ser la firma. Al igual que el anterior, quien verifica siempre aceptará firmas válidas, lo cual requiere verificación:

- Rechazar la firma si no se satisfice: **0<r<q** o **0<s<q**
- Calcular **w=(s2)-1(mod q)**
- Calcular **u1=H(m)*w(mod q)**
- Calcular **u2=s1*w(mod q)**
- Calcular **v=[gu1*yu2mod p] mod q**
- La firma resulta válida si **v=s1**

CIFRADO BASADO EN IDENTIDAD

También llamado **IBE** (**Identity-Based Encryption**), es una primitiva que se comporta de modo similar a los sistemas de clave pública, usando algo propio del usuario (como el email). La primera implementación fue de Adi Shamir en 1984 y en 2001 D. Boneh y M. Franklin presentaron un esquema realmente sólido.

El esquema de DSA puede ser modificado para operar sobre puntos de curvas elípticas en vez de exponenciaciones (**PLD**), dando origen al esquema **ECDSA** (*Elliptic Curve Digital Signature Algorithm*), que puede ofrecer el mismo nivel de seguridad que DSA o RSA pero con números más pequeños. Sus primitivas de base son la suma de puntos y la multiplicación escalar.

La generación de llaves incluirá la elección de una curva elíptica **E**, un punto **P** de orden **n** perteneciente a ella y un número aleatorio **d** (**1<d<n-1**), y de allí se derivará el cálculo de **Q=dP**, donde **d** es la clave privada y **Q** la pública.

Otros esquemas

Más allá de los esquemas de firma anteriores, vale la pena destacar algunas otras estructuras que también pueden garantizar los objetivos:

- **Esquema de Lamport** (L. Lamport, en 1979): también llamado **de firma de un solo uso**, es un esquema basado en cifrado simétrico que permite construir firmas a partir de cualquier función segura de una sola vía (usualmente de hash). Esto podría darle seguridad contra algoritmos cuánticos. Cada llave puede ser utilizada para firmar un solo mensaje, aunque combinado con **árboles de hash** (**hash trees** o **árboles de Merkle**) puede aplicarse a muchos.
- **Esquema de Merkle** (R. Merkle, fines de los 70): también basado en cifrado simétrico, utiliza árboles de Merkle y firmas de un solo uso como el de Lamport. Es una alternativa al firmado tradicional con esquemas como DSA y RSA. Su ventaja es la resistencia contra algoritmos de computadoras cuánticas, dependiendo solo de la existencia de funciones de hash seguras.
- **Esquema de Rabin** (M. Rabin, en 1979): fue uno de los primeros propuestos y el primero relacionado a la dificultad del problema de la factorización de enteros. Es uno de los más estudiados por su simplicidad y su rol prominente en los albores de la criptografía de clave pública. Es **existencialmente infalsificable** (**existentially unforgeable**) en el **modelo del oráculo aleatorio**, asumiendo la intratabilidad del problema matemático antedicho.
- **Esquema BLS** (D. Boneh, B. Lynn y H. Shacham, 2004): utiliza el **emparejamiento bilineal** (un concepto de la teoría de

grupos) para verificación y las firmas son elementos de grupo en una curva elíptica. Esto último provee defensa contra **ataques de cálculos de índice** (algoritmo probabilístico para calcular logaritmos discretos), permitiendo firmas más cortas que en RSA. Posee **infalfisicación existencial bajo ataques de texto plano escogido** (a esto se lo llama **provable security**), asumiendo la existencia de oráculos aleatorios y la intratabilidad del PLD.

- **Esquema de firma no denegable** (D. Chaum y H. van Antwerpen, 1989): cuenta con **verificación interactiva**, lo que permite al firmante limitar quién puede verificar la firma; y utiliza un **protocolo de desautorización**, que permite determinar si una firma es falsificada para evitar denegar firmas válidas.
- **Esquema GGH** (O. Goldreich, S. Goldwasser y S. Halevi, 1995): se basa en la resolución del problema del **vector más cercano** (**CVP**) en un enrejado o **lattice** (un concepto de teoría de grupos). El firmante debe resolver el problema del CVP sobre un punto que representa el mensaje y el verificador comprueba la suficiencia del resultado para el punto en cuestión.
- **Esquema Schnorr**: está basado en el **algoritmo de Schnorr**, cuya seguridad radica en la intratabilidad del PLD. Es considerado el más simple de los esquemas con **seguridad comprobable** en un modelo de oráculo aleatorio. Es eficiente, genera firmas cortas y su patente expiró en el 2008.

Solo a título informativo, no podemos dejar de mencionar **ESING**, **Guillou-Quisquater**, **Ohta-Okamoto**, **Okamoto**, y **Feige-Fiat-Shamir**, aunque tambien existen otros menos conocidos.

FIRMAS CON DISPOSITIVOS FÍSICOS

Las **smart cards** y otros dispositivos pueden usarse para el firmado digital siempre y cuando se defina un esquema, pudiendo utilizar claves simétricas o asimétricas almacenadas en ellos. La problemática radica en la instalación y el almacenamiento seguro de las claves, de forma que no puedan ser recuperadas físicamente (**tamper resistant**).

Criptografía y leyes

En muchos países está permitido utilizar la **firma electrónica** para manifestar la voluntad, cuyo proceso está vinculado a procedimientos informáticos. No obstante, el verdadero valor legal se obtiene mediante el uso de la específicamente llamada **firma digital** (en algunos países **firma electrónica avanzada**, **reconocida** o **certificada**), que permite realizar operaciones como la firma de un contrato. Las leyes equiparan a estas firmas con la **firma manuscrita**, dándole así validez legal en su territorio nacional para identificar al firmante. La firma digital, entonces, asocia la identidad de una persona a un mensaje o documento digital, y debe ser susceptible de verificación por terceras partes, tal que se permita identificar al firmante y detectar alteraciones del documento posteriores a su firma. Así, salvo prueba en contrario, se presume que toda firma digital pertenece al titular del certificado.

Para que se pueda aplicar esto, es necesario contar con autoridades válidas por ley que permitan avalar las identidades, generar las firmas y proveer los certificados manteniendo su ciclo de vida. De esta forma, la FD con **validez legal** queda asociada a procesos y procedimientos estrictos que deben cumplirse para que pueda ser considerada válida, y nada tiene que ver con la **digitalización óptica** de una firma manuscrita. En Argentina, por ejemplo, la **ley 25.506** (2001) regula la aplicación de la firma digital y le da la misma validez que a la **firma hológrafa** para casos que no sean disposiciones por causa de muerte, derecho de familia, actos personalísimos, casos bajo exigencias incompatibles por disposiciones o acuerdo de partes.

RESUMEN

En este capítulo vimos cómo se combinan los componentes estudiados en una infraestructura común llamada PKI, que se relaciona con autoridades y por medio de políticas logra garantizar la autenticidad y la confidencialidad. Estudiamos la necesidad del uso de certificados de clave pública y su ciclo de vida, y los métodos de chequeo y revocación como CRL y OCSP. También vimos el rol del estándar X.509 y PKCS, y analizamos la importancia de la firma digital, sus conceptos asociados y los esquemas principales de implementación.

Actividades

TEST DE AUTOEVALUACIÓN

1. ¿Qué componentes tiene una PKI?
2. ¿Qué garantiza una firma digital?
3. ¿Qué ventaja tiene una CRL sobre OCSP?
4. ¿Qué es OCSP stapling?
5. ¿Para qué se creó el estándar X.509?
6. ¿Qué estandariza PKCS?
7. ¿Por qué el esquema DSA se utiliza más que ElGamal?

EJERCICIOS PRÁCTICOS

1. Compruebe manualmente la validez de los certificados de cinco sitios web conocidos.
2. Averigüe el costo de las distintas clases de certificados de VeriSign.
3. Identifique cinco autoridades certificantes privadas.
4. Realice un ejemplo numérico con el sistema DSA.
5. Investigue las leyes de firma digital de distintos países.

PROFESOR EN LÍNEA

Si tiene alguna consulta técnica relacionada con el contenido, puede contactarse con nuestros expertos: profesor@redusers.com

*08

Protocolos y sistemas de autenticación

En este último capítulo estudiaremos algunos de los protocolos criptográficos más importantes y sus usos principales en autenticación, tomando como contexto su utilidad en los sistemas de control de accesos.

Protocolos criptográficos

En telecomunicaciones, un **protocolo** es un conjunto de reglas para comunicar dos o más partes de un sistema a fin de transmitir información.

Así se define la sintaxis, la semántica, la sincronización y los métodos de corrección de errores. Pueden implementarse por hardware, software o ambos; y aunque pueden ser de propósitos muy variables, algunas de las características que pueden proporcionar típicamente son: detección de conexión física, **handshaking** (saludo inicial), negociación de características, saludo final, formateo de mensajes, tratamiento de errores y seguridad, entre otras.

Los protocolos criptográficos y sistemas de autenticación aplican métodos y técnicas de cifrado para obtener confidencialidad, integridad, autenticación y no repudio en una comunicación de datos. Para esto, deben describir la manera en la que se debe utilizar los algoritmos, y sus formas de operación. Existen de diversos tipos en función de lo que proveen, como establecimiento de claves, autenticación, cifrado, transporte seguro y no repudio.

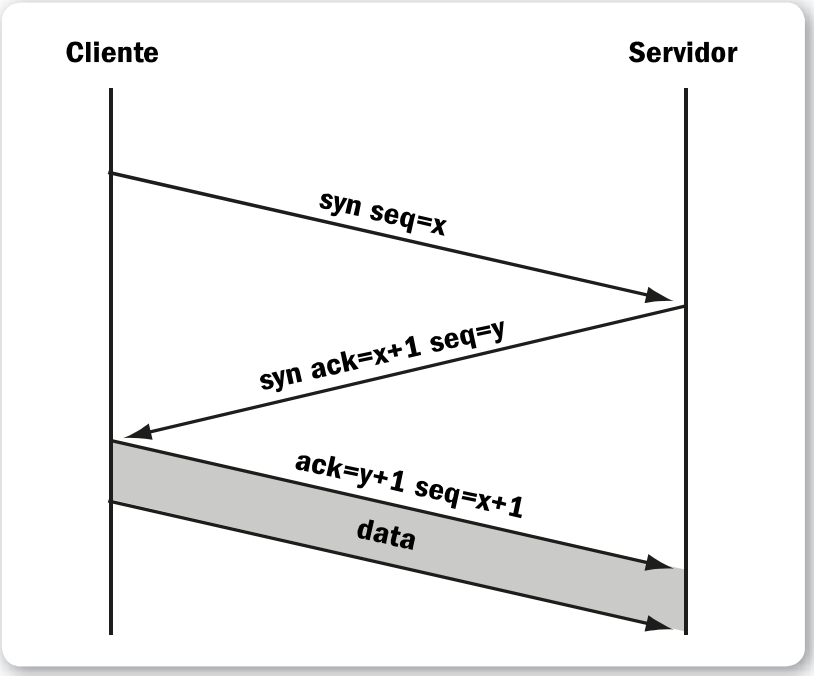


Figura 1. TCP Handshake es un clásico ejemplo de protocolo de establecimiento de comunicaciones de red.

Principios de control de acceso

Cuando hablamos de **control de acceso** nos referimos a la capacidad de permitir acceso a un sistema o recursos solamente a entidades autorizadas (usuarios, programas y procesos), buscando proteger datos y sistemas. En la implementación de un sistema de control de accesos hay distintas fases, que podemos resumir en **establecimiento**, **mantenimiento**, **revocación** y **auditoría**.

En un esquema de acceso, un **objeto** será la **entidad pasiva**, en tanto que un **sujeto** será la **entidad activa**. Así, llamamos **acceso** al **flujo** entre sujeto y objeto.

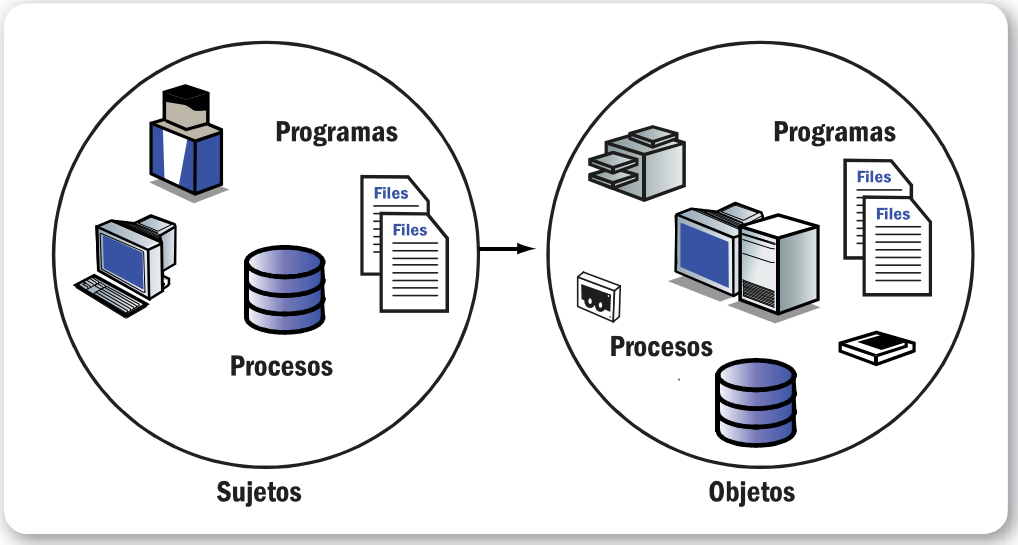


Figura 2. Decimos que un **dominio** es el conjunto de objetos a los que puede acceder un sujeto, y un **grupo** es el conjunto de sujetos y objetos asociados por características.

MÁS ALLÁ DE LO BÁSICO

Algunos protocolos criptográficos van más lejos que la sola autenticación, buscando, por ejemplo, firmas sin conocimiento previo o demostración de atributos sin revelación de identidad. Según cuántas partes intervengan en un protocolo, existirán los de **dos partes** y los de **múltiples partes**, lo que dará origen a distintas técnicas para ampliar las capacidades básicas.

El acceso, que en general se realiza entre un cliente y un servidor, puede ser planteado en tres instancias: **identificación**, **autenticación** y **autorización**.

En la primera, una entidad se da a conocer frente al sistema, presentando alguna especie de elemento (nombre de usuario, número, etcétera). En la segunda, el sistema comprueba que la entidad sea quien dice ser, verificando así su identidad. Finalmente, en la tercera, el sistema (habiendo verificado la identidad) permite acceder a la entidad a los recursos que le corresponden según sus permisos asociados.

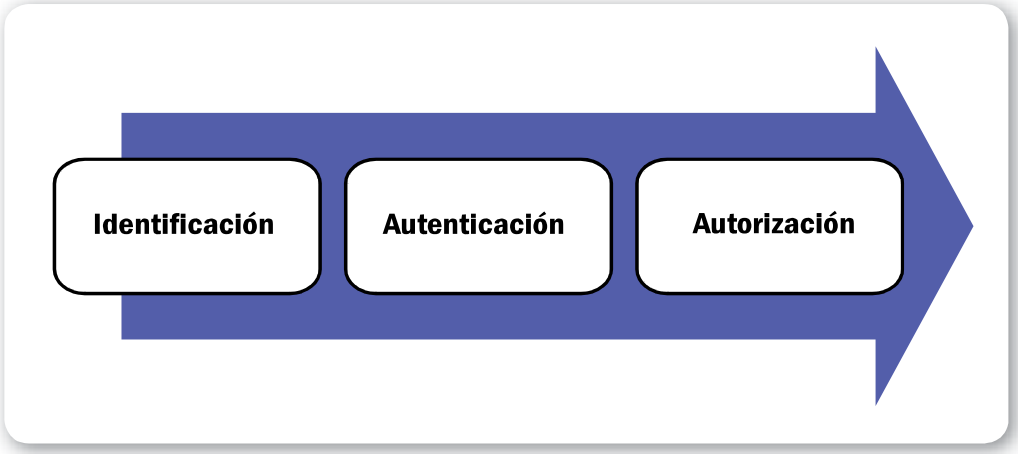


Figura 3. El acceso a un sistema puede interpretarse como un proceso dividido en tres fases consecutivas.

Factores de autenticación

La validación de los factores de autenticación se puede realizar por métodos diferentes, que pueden combinarse para aumentar el nivel de seguridad proporcionado (**múltiples factores**). Los métodos son:

MODELOS DE CONTROL DE ACCESO

Los modelos de CA se distinguen principalmente entre los **discrecionales** (**DAC**), donde el usuario designa quién tiene permiso para acceder a sus recursos; y los **mandatarios** (**MAC**), donde lo decide el sistema en base a etiquetas. También se encuentran los llamados modelos **no discretos** (**Lattice based**, **Role based** o **Task based**) y los formales, como **Biba**, **Take/Grant**, **Clark/Wilson** y **Bell/LaPadula**.

- **Algo que uno sabe**: elementos conocidos (como un password, passphrase, código, respuesta secreta, etcétera).
- **Algo que uno tiene**: elementos físicos (como un token, dongle, smartcard, llave, tarjeta de coordenadas, etcétera).
- **Algo que uno es**: sistemas biométricos que determinan características físicas o del comportamiento (como la huella digital, la retina, la voz, etcétera).

Figura 4. La huella dactilar es uno de los principales factores de autenticación biométrica.

Así, si se usa algo que uno tiene y algo que uno conoce, se tiene una autenticación de **dos factores** (**two factors**). Si se desea un alto nivel de seguridad, no es eficiente combinar dos factores de la misma categoría (un password y un código, por ejemplo) sino que deben ser de distintas (como un password y un token).

Sistemas AAA

Llamamos sistemas **AAA** (*Authentication, Authorization, Accounting*) a aquellos que proveen autenticación, autorización y auditoría, lo que suele obtenerse por medio de uno o más protocolos destinados a tal fin.

Autenticación y autorización equivalen a los conceptos anteriormente vistos, en tanto que auditoría (o **contabilidad**

o **trazabilidad**) está relacionado al seguimiento de las acciones de una entidad durante la utilización de un sistema. Dicha información puede luego usarse para administración, estadística y facturación, entre otros. Puede realizarse en tiempo real (en el momento en que se produce la acción) o por procesamiento posterior (**batch**).

RADIUS

RADIUS (*Remote Authentication Dial-In User Service*) es un protocolo de AAA que trabaja sobre el puerto **1812 UDP**. Fue creado en 1991 por **Livingston Enterprises** para sus servidores **NAS** (*Network Access Server*) y luego publicado por **IETF** como **RFC 2138** y **RFC 2139**. En la actualidad, está definido principalmente en el **RFC 2865** para **autentificación** y **autorización**, y en el **RFC 2866** para **accounting**.

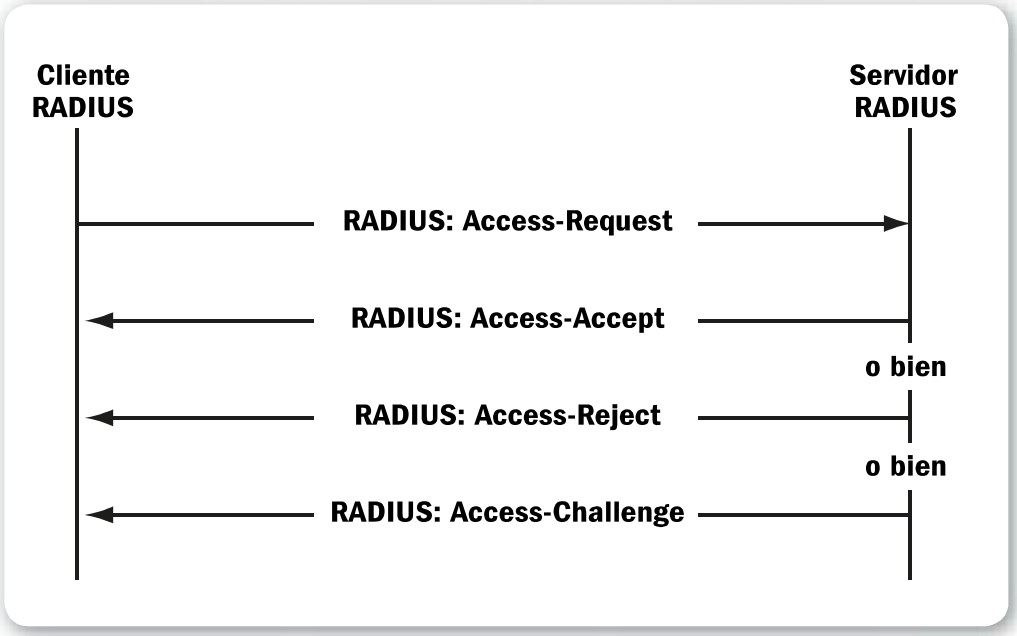


Figura 5. Flujo de autenticación y autorización en **RADIUS** con sus posibles respuestas: aceptación, rechazo o desafío.

Se lo ha encontrado históricamente en conexiones de acceso telefónico a internet (**Dial-Up**) o en la conexión inicial de un módem **DSL** o **cable modem**, e incluso en redes inalámbricas. Los datos que se envían son en general el usuario y la contraseña, que llega a un dispositivo llamado Network Access Server (NAS) sobre **PPP** (*Point-to-Point Protocol*), que redirecciona el pedido a un servidor RADIUS

(con el protocolo característico) que valida la información mediante protocolos como **PAP**, **CHAP** o **EAP**. En caso afirmativo, se autoriza al usuario a acceder a los recursos del sistema y se le asignan los parámetros de red como la dirección **IP**. RADIUS administra las sesiones, notifica inicio y fin de conexiones (ya sea para fines de tarifación o estadísticos), y demás funciones de control de accesos.

Además, existen los **Proxy RADIUS**, que pueden reescribir paquetes al vuelo. Es un protocolo extensible y, de hecho, han proliferado varios dialectos propios de cada fabricante. Cuenta con implementaciones libres y comerciales. **FreeRADIUS** es la versión libre más conocida.

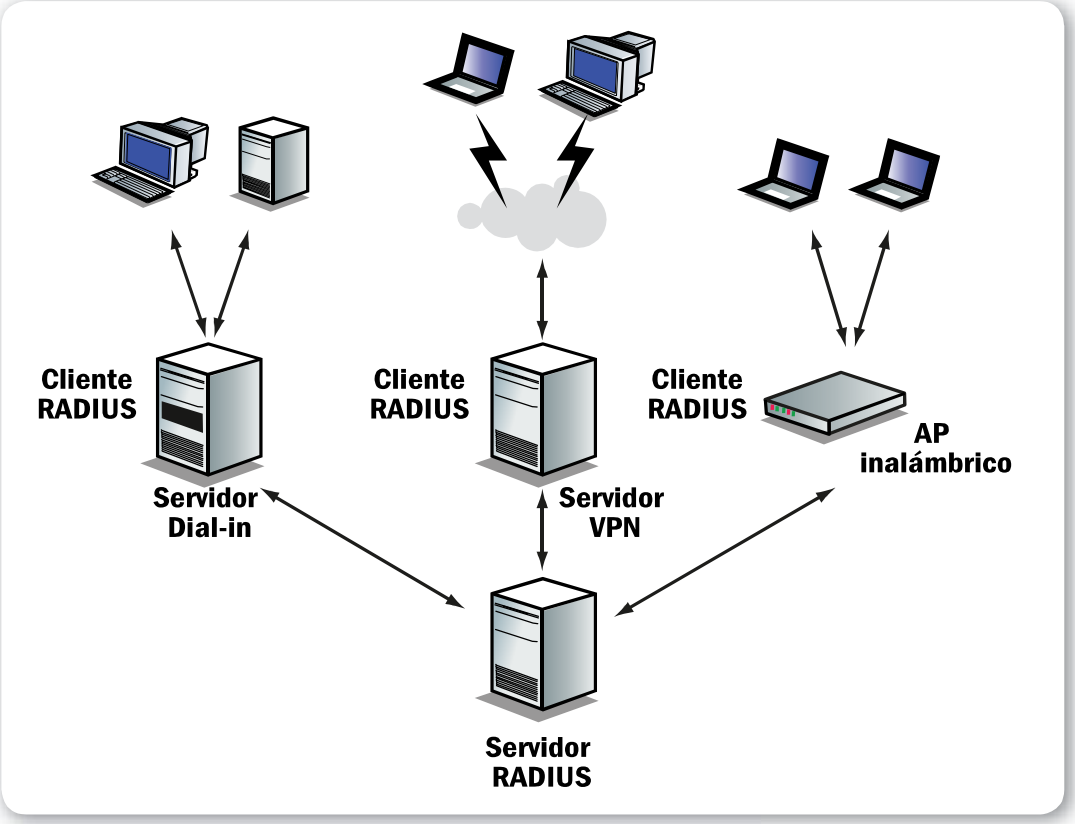


Figura 6. Posibles escenarios de uso de RADIUS.

RADIUS funcionó como base para crear **DIAMETER**, otro protocolo AAA cuyo nombre alude a una broma del lenguaje (**DIAMETER**= **2** x **RADIUS**) e intenta extender los servicios de AAA a las nuevas tecnologías y arquitecturas. Puede trabajar localmente o en modo **alerta**, **sondeo** y **captura** (**roaming de AAA**), lo que le posibilita prestar servicios móviles con cierta flexibilidad.

DIAMETER

DIAMETER es un sistema de AAA no retrocompatible pero que permite ser actualizado desde RADIUS. A diferencia del anterior, usa protocolos de transportes confiables como **TCP** (no **UDP**), permite seguridad a nivel de transporte (**IPSEC** o **TLS**) y cuenta con un mayor espacio de direcciones para sus **pares atributo-valor** (**AVP**, *Attribute Value Pairs*) y sus identificadores.

DIAMETER no es cliente-servidor sino **peer-to-peer**, puede usarse en modo **stateless** (sin registro de estado) o **statefull** (con registro de estado), admite descubrimiento dinámico de pares, permite negociar capacidades, cuenta con notificación de errores y es más compatible con las aplicaciones móviles por sus avances respecto al manejo del roaming. Finalmente, permite definir nuevos comandos y atributos (**extensiones**) con mayor facilidad que su antecesor.

> DIAMETER ES PEER-TO-PEER Y PUEDE USARSE EN MODO STATELESS O STATEFULL

El flujo de mensajes en DIAMETER se inicia con un requerimiento de capacidades o **CER** (*Capabilities-Exchange-Request*), que se responde con una respuesta de capacidades o **CEA** (*Capabilities-Exchange-Answer*). Si no se intercambian mensajes, de tanto en tanto se envía un mensaje de detección de comunicación o **DWR** (*Device-Watchdog-Request*), que se responde con una respuesta válida **DWA** (*Device-Watchdog-Answer*).

Cualquier parte puede finalizar la comunicación con un mensaje **DPR** (*Disconnect-Peer-Request*) para indicar que desea el fin del intercambio, a lo que la otra debería responder con un **DPA** (*Disconnect-Peer-Answer*).

OPEN ID

Es un estándar de identificación descentralizada, que sirve para identificarse online y contra servidores que lo soporten, sin tener que crear un nuevo usuario, sino solo presentando su identificador creado en un servidor **proveedor de identidad** (**IdP**). Su seguridad depende de la confianza entre el **cliente OpenID** y el proveedor de identidad, que además puede realizarse con distintos métodos.

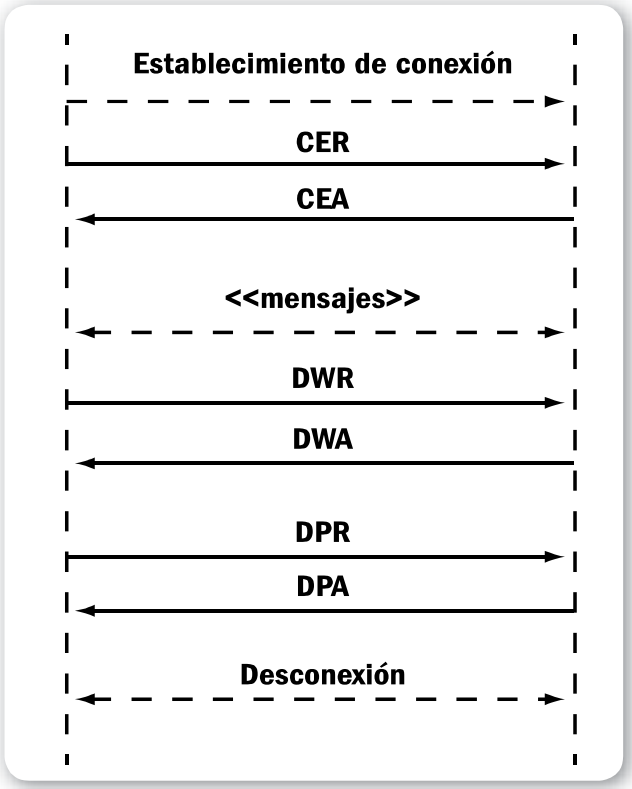


Figura 7. Flujo de mensajes en **DIAMETER**, donde se incluye el envío de mensajes de detección de comunicación (**DWR** y **DWA**).

TACACS

TACACS (*Terminal Access Controller Access Control System*) es un sistema de AAA creado en 1984 para los viejos servidores **UNIX**. TACACS permite a un servidor de acceso remoto (RAS) comunicarse con un servidor de autenticación para determinar si un usuario debe tener o no acceso a la red, administrando las contraseñas de forma

ESCRIBA SU CONTRASEÑA

Una manera eficiente de validar contraseñas de usuarios en su acceso a un sistema operativo o aplicación es utilizando funciones de hash. Cuando el usuario escribe su contraseña, el sistema le aplica la funcion de hash correspondiente y obtiene el resultado, que es comparado con el valor que tiene almacenado el software. Así también se evita almacenar contraseñas en texto plano.

centralizada en una base de datos. Está definido en el **RFC 1492**, utiliza el puerto 49 y su código es de dominio público.

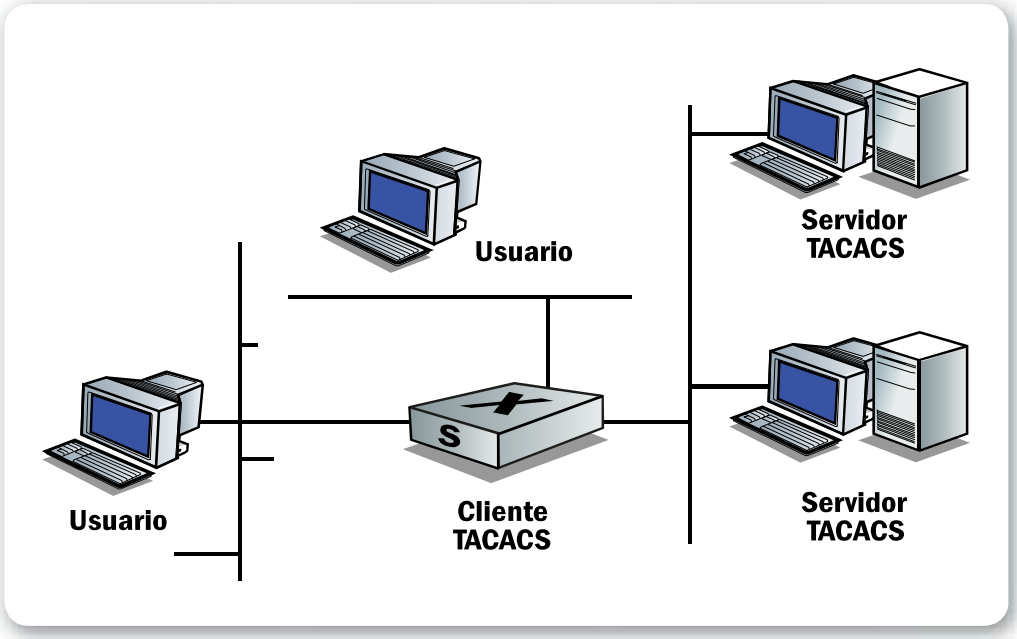


Figura 8. Arquitectura **TACACS**, compuesta por servidores, clientes y usuarios.

Derivó en **XTACACS** (*Extended TACACS*), una extensión propietaria de **Cisco Systems** (1990) sin retrocompatibilidad. Luego, la misma empresa realizó nuevas mejoras que derivaron en **TACACS+** (*TACACS plus*) en 1993, permitiendo el cambio de contraseñas y autenticación de dos factores, no existentes en los anteriores. Sus ventajas lo hicieron reemplazar ampliamente a sus predecesores.

Kerberos y Sesame

Kerberos es un estándar y un protocolo de autenticación para redes, cuya última versión es de junio de 2013. Posibilita la comunicación entre partes sobre un canal inseguro permitiendo que cada una demuestre su identidad a la otra. Utiliza un modelo cliente-servidor donde ambos verifican la identidad del otro, posibilitando la autenticación del inicio de la conexión y los mensajes individuales. El modelo está basado en la **criptografía simétrica**

(aunque existen extensiones para utilizar cifrado simétrico) y exige la existencia de un **tercero de confianza** (**TTP**).

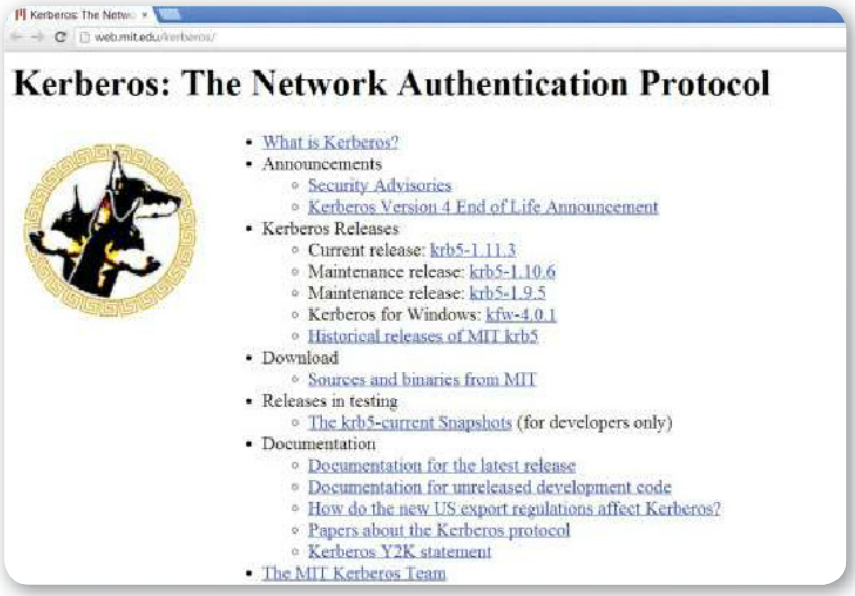


Figura 9. Sitio original de **Kerberos**, del **MIT**.

Por requisito de diseño, Kerberos debía ser **seguro**, **confiable**, **no evitable** por el usuario, **transparente** (que el usuario no note su uso) y **portable**. Por otra parte, asume que las contraseñas no serán débiles y que la seguridad física de los equipos se encuentra garantizada. Cuenta con tres componentes básicos:

- **KDC** (*Key Distribution Center*): el tercero de confianza, que almacena la base de datos de usuarios y contraseñas.
- **AS** (*Authentication Service*): encargado de la primera identificación.
- **TGS** (*Ticket Granting Service*): encargado de proporcionar contraseñas de sesión y tickets para acceder a los servicios.

Se puede dividir en función de tres fases:

1. **Inicio de Sesión**: el usuario solicita y obtiene las credenciales necesarias para acceder al sistema.
2. **Solicitud de Acceso**: el usuario obtiene las credenciales necesarias para solicitar acceso al servicio final.
3. **Fase de Acceso**: el usuario le presenta las credenciales al servicio final y lo accede.

En breve, el sistema funciona haciendo que el cliente se autentique a sí mismo contra el **AS** para demostrar al **TGS** que está autorizado para que se le entregue un ticket, con el que podrá acceder al servicio en cuestión. En todos los mensajes hay claves de sesión de las partes que se intercomunican y claves secretas cifradas para que no puedan ser vistas por quien observe el ticket, sino solo por la parte con la que se comparte el secreto.

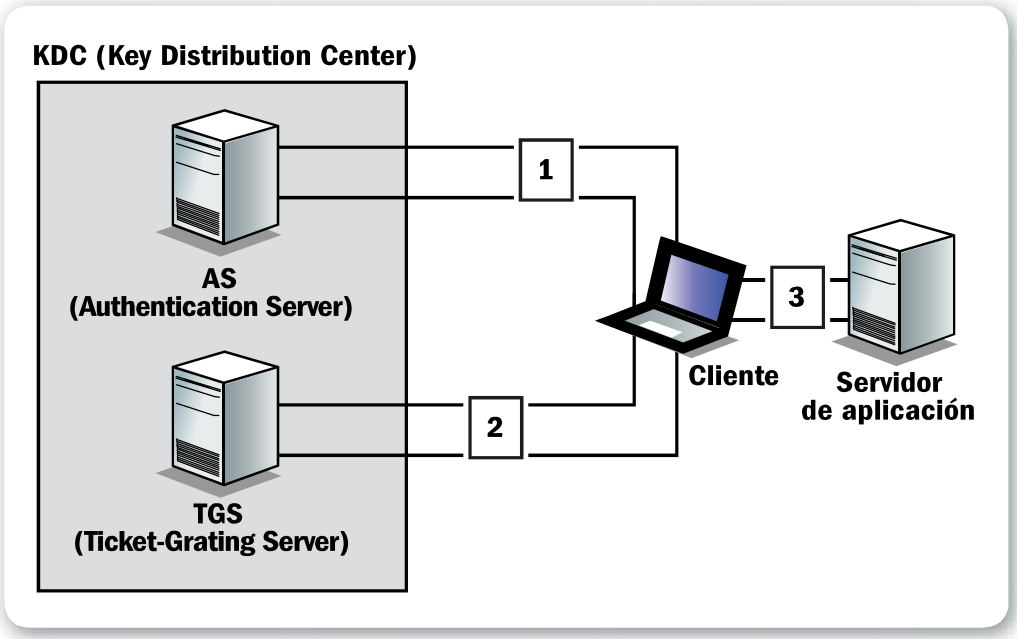


Figura 10. Esquema simplificado de pasos de **Kerberos**, donde el cliente debe contactar a los correspondientes servidores antes de obtener el servicio en cuestión.

Pese a todas sus ventajas, la implementación suele ser compleja por distintos factores, como la necesidad de un **reloj de red sincronizado** (**NTP**) para las **marcas de tiempo**, la alta generación

CANCERBERO Y EL MIT

Kerberos fue desarrollado por el **MIT** en los años 80 en el marco del proyecto **Athena**, con el apoyo de **DEC** e **IBM**. El nombre hace referencia a **Cancerbero**, el perro de tres cabezas que, según la mitología, cuidaba las puertas de los infiernos (**Hades**). Se basó en el protocolo de **Needham-Schroeder** (1978) referido al cifrado simétrico, pese a que había una versión para cifrado asimétrico.

de **tráfico**, la **centralización** (efecto **cuello de botella**) y la necesidad de que las aplicaciones soporten el protocolo en su autenticación (llamado **kerberización**).

SESAME (*Secure European System for Applications in a Multi-vendor Environment*) es un proyecto europeo apoyado por la comisión de las comunidades europeas que ofrece un sistema de **single sign-on** (**único login**) con características de control de acceso distribuido y protección criptográfica.

Es un conjunto de componentes de infraestructura de seguridad que provee una base para crear productos. Es similar a Kerberos pero cuenta con extensiones, entre las que se destaca el soporte para control de accesos basado en roles utilizando el llamado **PAS** (*Privilege Atribute Server*). SESAME refuerza las políticas de control, gestionadas de forma centralizada, y tiene una base de datos central asociada a cada usuario (manejada por el PAS). La información del control de acceso es llevada al servidor de aplicación en forma de **PAC** (*Privilege Attribute Certificates*), que son análogos a los tickets de Kerberos.

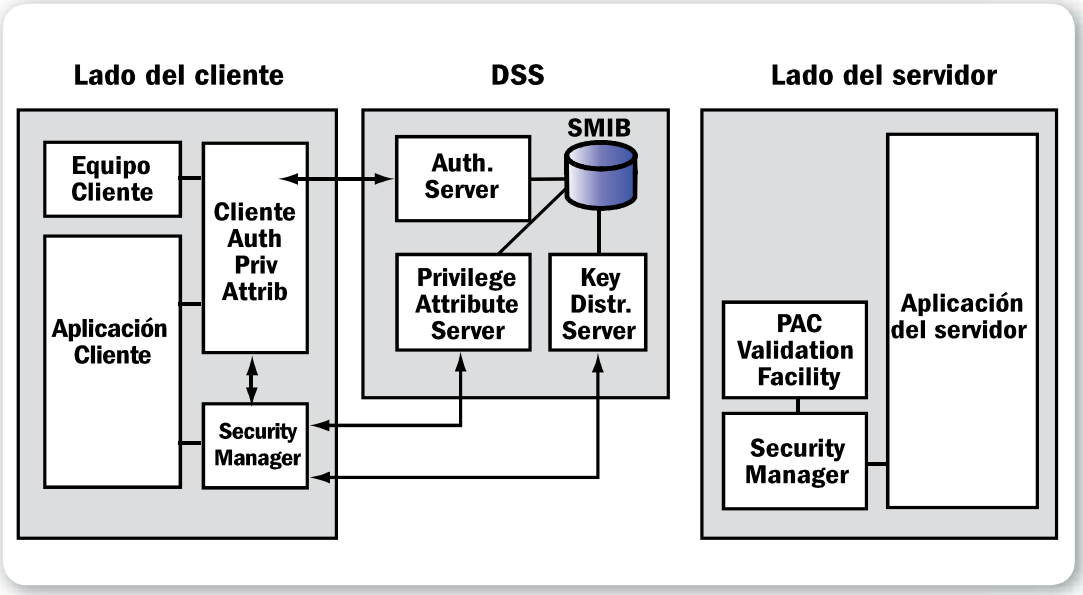


Figura 11. Arquitectura de **SESAME**, donde puede verse la forma de interacción entre los principales elementos.

El PAS forma parte del **DSS** (*Domain Security Server*) junto al **AS** (*Authentication Server*) y al **KDS** (*Key Distribution Server*), y todos almacenan su información en el **SMIB** (*Security Management Information Base*).

Otros componentes del sistema son el **SACM** (*Secure Association Control Manager*), que es la interfaz con la aplicación objetivo y sus servidores de seguridad, y el **PVF** (*PAC Validation Facility*) que es utilizada por la anterior para validar los PAC.

PPP

PPP (*Point-to-point Protocol*) surgió como la evolución natural del ya obsoleto **SLIP** (*Serial Line Internet Protocol*), diseñado por Rick Adams en 1984. Funciona en capa de enlace y su objetivo es proveer autenticación, cifrado y compresión.

Se lo utilizó ampliamente en todo tipo de redes, incluyendo conexiones de cable serial, líneas telefónicas, redes celulares y más. Fue muy popular para realizar conexiones a internet, para lo que se requirieron adaptaciones como **PPPoE** (*PPP over Ethernet*) ya que los paquetes IP no pueden transmitirse tal cual están vía modem. Esto lo llevó a ser el tipo de conexión estándar en las líneas **DSL** (*Digital Subscriber Line*).

PPP opera en distintas etapas: **establecimiento de conexión**, **autenticación**, **configuración de red**, **transmisión** y **finalización**; y garantiza la recepción ordenada, como todo protocolo de capa de enlace.

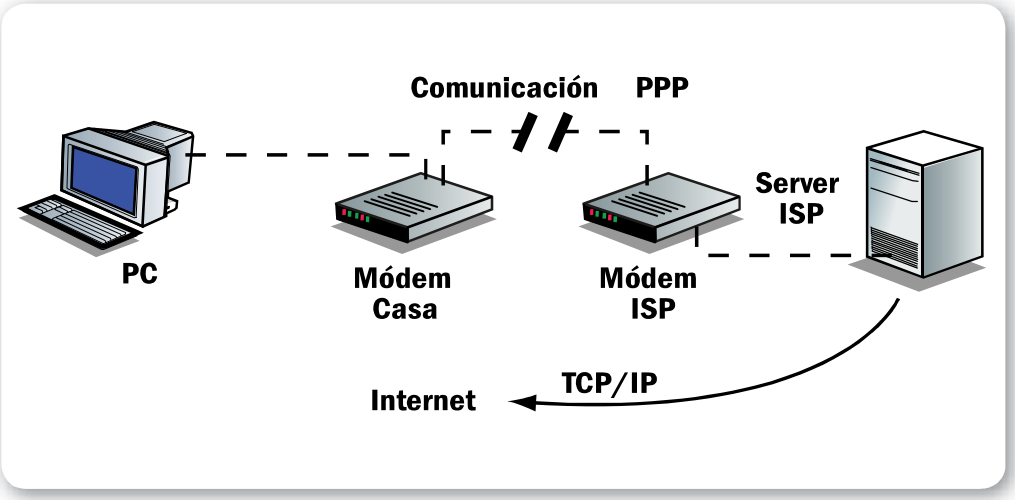


Figura 12. Uso básico de **PPP** para una PC hogareña que se conecta a un ISP.

PAP, CHAP y EAP

Un conjunto de protocolos de autenticación que veremos en un mismo contexto es el compuesto por **PAP** (*Password Authentication Protocol*), **CHAP** (*Challenge Handshake Authentication Protocol*) y **EAP** (*Extensible Authentication Protocol*). Estos se suelen usar como parte del mecanismo de autenticación de otros protocolos, como PPP.

> PAP, CHAP y EAP suelen usarse para la autenticación de otros protocolos

PAP es un protocolo básico que permite autenticar usuarios contra servidores. Es usado en PPP y transmite contraseñas en texto plano, por lo que no se recomienda su uso.

CHAP es un protocolo de desafío mutuo y se utiliza tanto en autenticación cableada como inalámbrica. También es usado en PPP, y realiza verificaciones periódicas de la identidad del usuario basadas en un secreto compartido (contraseña). CHAP contiene tres pasos:

1. El servidor envía un pedido de autenticación al cliente.
2. El cliente responde con un valor de hash.
3. El servidor chequea la respuesta con su propio cálculo del hash, aprobando o no el enlace.

Como ventaja, no puede ser vulnerado a priori con **ataques de repetición**, ya que usa un identificador incremental y un valor variable para verificar, pero tiene la desventaja de requerir que el cliente mantenga de su lado el secreto en texto claro. Además, al basarse en funciones hash, debe tenerse cuidado con sus posibles

EAP, LOCK-STEP Y FRAGMENTACIÓN

EAP es un protocolo **lock-step**, lo que implica que solo admite un paquete en transmisión, siendo esto ineficiente. Además, no soporta fragmentación, por lo que si se lo usa para autenticar con certificados (siendo que éstos son más grandes que el **MTU** válido), la cantidad de **round-trips** cliente-servidor crece para simular fragmentación.

ataques conocidos. Una implementación muy difundida es **MS-CHAP**, de **Microsoft**, que también tiene debilidades, como las publicadas por Bruce Schneier, Jochen Eisinger y Moxie Marlinspike, por lo que la única recomendada de esta serie es **MS-CHAP v2**.

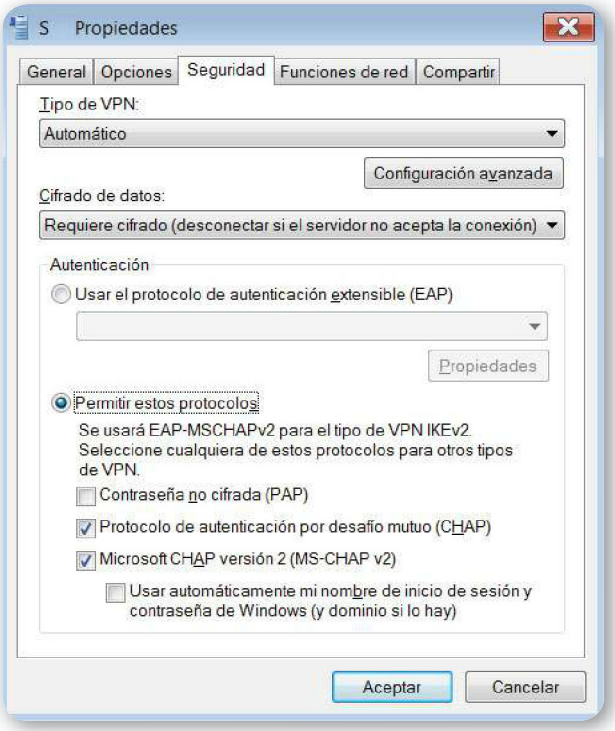


Figura 13. Al configurar una conexión remota es posible determinar el mecanismo de autenticación deseado.

LA ESTRUCTURA DE EAP ES ADAPTABLE A PROTOCOLOS CABLEADOS E INALÁMBRICOS

Por su parte, podría decirse que EAP (*Extensible Authentication Protocol*) es más avanzado que los anteriores, ya que no es un mecanismo específico sino mas bien un esquema de soporte para ser utilizado junto a otros protocolos.

Provee funciones de autenticación (llamados **métodos EAP**) y otras específicas como **EAP-MD5**, **EAP-TLS**, **EAP-IKE**, **EAP-SIM**, entre otras. Su estructura es adaptable a protocolos inalámbricos y cableados, lo que le da mucha versatilidad, aunque fue pensado principalmente para acceso a internet. Al no requerir conectividad IP, solo provee soporte para transporte de protocolos de autenticación. Además, tanto RADIUS como DIAMETER pueden encapsular mensajes EAP.

El proceso de autenticación EAP se realiza en cuatro fases:

1. El servidor envía una solicitud de autenticación al cliente, el cual debe responder según el tipo que solicite (**identidad**, **notificación**, **MD5-Challenge**, **Generic Token-Card**, etcétera).
2. El cliente responde al servidor con el tipo correspondiente en la respuesta.
3. El servidor envía otra solicitud, que debe ser respondida todas las veces requeridas a fin de cumplimentar con la posible fragmentación.
4. Si el servidor no puede autenticar al cliente, transmite un mensaje de falla; si lo consigue envía un paquete de éxito.

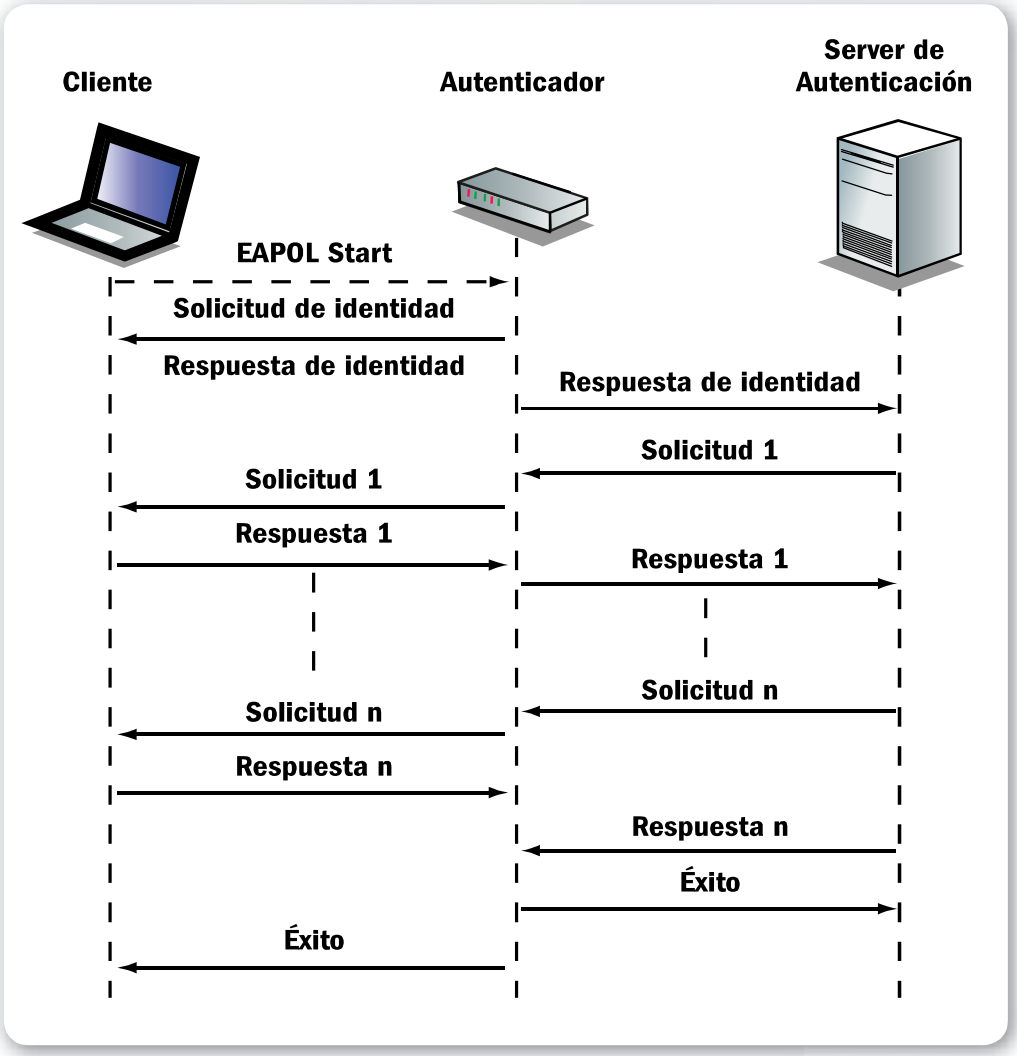


Figura 14. Autenticación **EAP**, comenzando por un mensaje **EAPOL** (**EAP over LAN**).

NTLM

NTLM (*NT LAN Manager*) es un conjunto de protocolos de autenticación y cifrado para redes Microsoft, sucesor de **LANMAN** (*LAN Manager*). La versión 2 (**NTLMv2**) introducida en **Windows NT SP4** mejoró la seguridad respecto a la primera, pero Microsoft dejó de recomendarlo desde 2010 ya que no permite el uso de algoritmos fuertes como **AES** o **SHA-256** sino que se basa en otros más antiguos y obsoletos. Microsoft recomienda reemplazar la autenticación NTLM por Kerberos, que es el protocolo por defecto en **AD** (*Active Directory*). No obstante, el soporte de NTLM se mantiene para poder autenticar clientes heredados, fuera del dominio o sin capacidad de manejar Kerberos. Su funcionamiento básico se puede detallar en siete pasos:

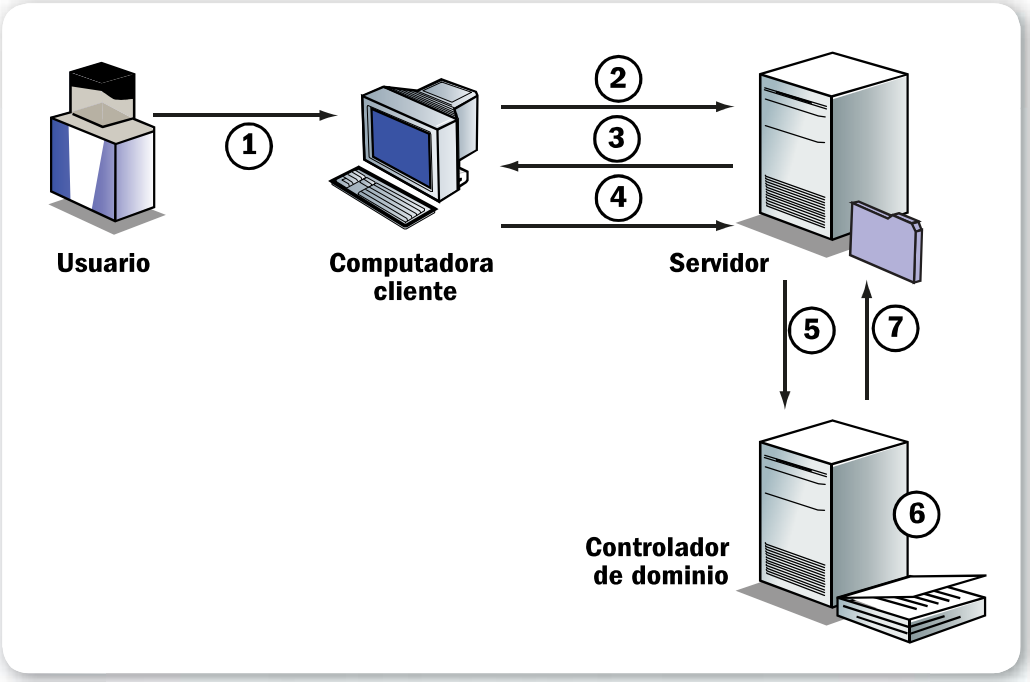


Figura 15. Esquema de pasos de **NTLM**, donde cada número corresponde a los pasos de su funcionamiento.

1. Un usuario se loguea en el sistema.
2. El cliente envía el nombre de usuario en texto plano al servidor.
3. El servidor crea un desafío y se lo envía al cliente.
4. El cliente cifra el desafío con el hash del password del usuario y lo envía al servidor.
5. El servidor envía los tres elementos al controlador de dominio (**DC**).

6. El DC recupera el hash del password del usuario de su base local **SAM** (*Security Account Manager*) y lo usa para cifrar el desafío.
7. El DC compara el resultado de lo recibido con lo obtenido, y si coinciden notifica la autenticación exitosa.

SSL/TLS

SSL (*Secure Sockets Layer*) es un protocolo criptográfico que provee seguridad en la capa de transporte (modelo OSI) y fue desarrollado originalmente por Netscape para ser incluido en su navegador web. Su sucesor fue **TLS** (*Transport Layer Security*), cuya versión 1.0 coincide casi totalmente con la versión 3.0 de SSL (1996); de hecho, en general se habla de SSL a secas para referirse a cualquiera de ellos.

SSL opera de una manera modular, cuenta con un diseño extensible, es retrocompatible y tiene soporte para negociación de capacidades entre las partes.

En general, sólo el servidor es autenticado y no así el cliente, ya que requeriría de una PKI. Para su operación utiliza diversos algoritmos, tanto simétricos como asimétricos y de hash.

Sus fases básicas son:

- Negociación de la versión del protocolo.
- Negociación del algoritmo a utilizar.
- Intercambio de claves públicas y autenticación basada en certificados digitales.
- Cifrado del tráfico basado en algoritmo simétrico.

HTTP + SSL

SSL se utiliza con **HTTP** para formar **HTTPS**, la versión segura de **HTTP** que utiliza el **puerto 443** para crear un canal cifrado, cuyo nivel depende del servidor y del navegador web. Suele ser utilizado en sitios donde se maneja dinero y datos privados, y también se usa para establecer túneles tipo **VPN** (como con **OpenVPN**).

SSL cuenta con tres subprotocolos, que son el **Handshake** (establecimiento de la conexión), el **Change Cipher Spec** (cambio de parámetros) y el **Alert** (indicación de cambio de estado o notificaciones).

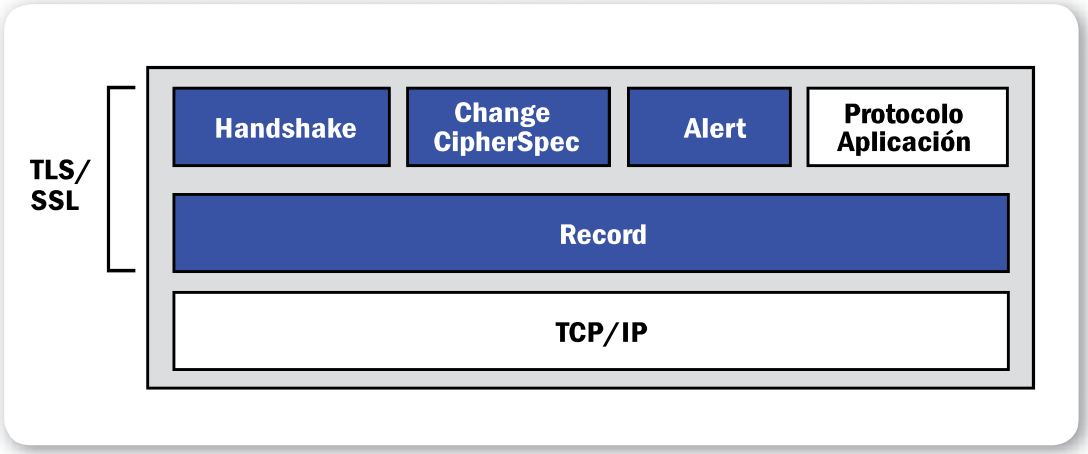


Figura 16. Capas de subprotocolos de **SSL/TLS**.

Cuenta con una **capa de registros** (**records**) de intercambio de información, que puede ser comprimida, cifrada y empaquetada con un código MAC y tiene un campo que especifica el protocolo de nivel superior. Al iniciar la conexión, el nivel de registro encapsula otro protocolo: el Handshake.

El mensaje inicial se denomina **ClientHello** y especifica los parámetros de conexión, y es contestado con un **ServerHello** en el que el servidor elige los parámetros a partir de las opciones dadas previamente.

El mensaje final del Handshake envía un hash de todos los datos intercambiados, y se aplica una función pseudoaleatoria que divide los datos en dos mitades, las procesa con distintos algoritmos (MD5 y SHA) y realiza un **XOR** con ellos.

Muchos clientes y servidores proporcionan SSL nativamente, pero si la aplicación no tiene soporte nativo se pueden usar aplicaciones adicionales bajo el concepto de **wrappers** (como **Stunnel**), aunque **IETF** en 1997 recomendó a los protocolos de aplicación ofrecer actualizar a TLS a partir de conexiones sin cifrar, en lugar de utilizar un puerto distinto para comunicaciones cifradas. De esa manera, TLS ha comenzado a soportarse de manera estándar en prácticamente todas las aplicaciones.

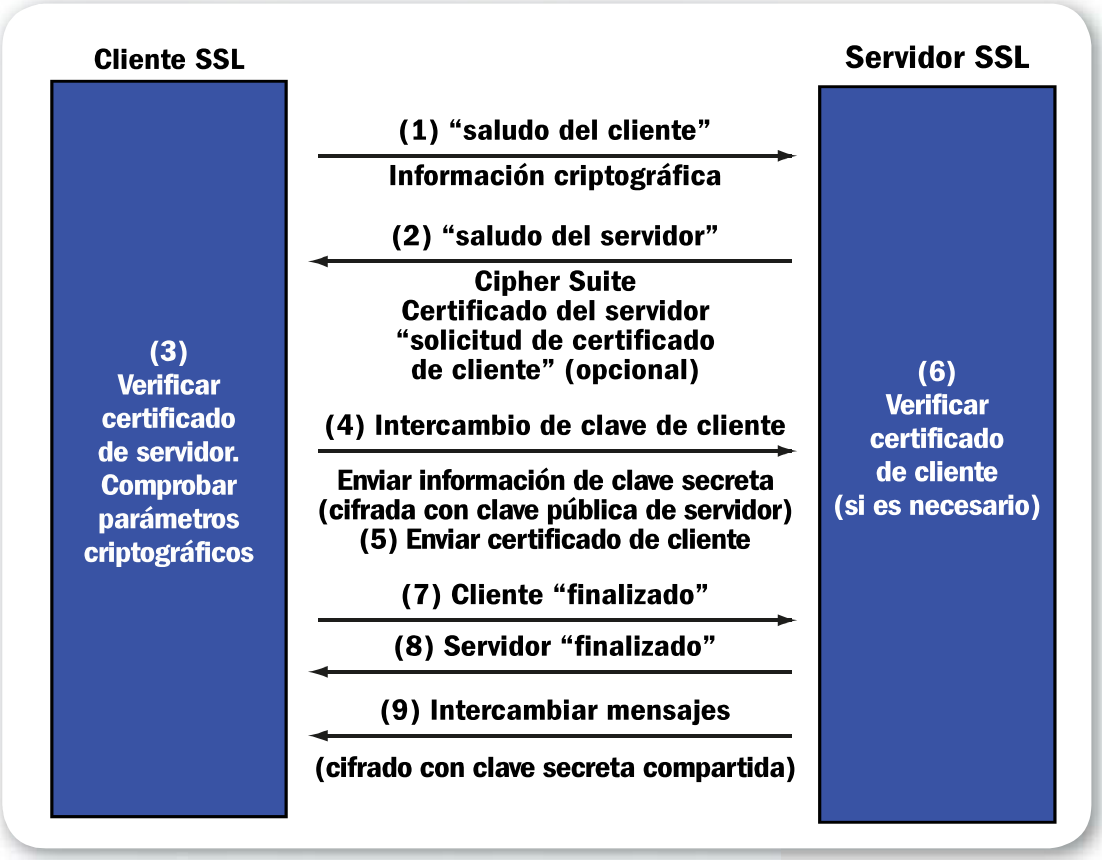


Figura 17. Esquema del **handshake** de **SSL/TLS**.

PGP

PGP (*Pretty Good Privacy*) es un software criptográfico creado en 1991 por Phil Zimmermann, y es utilizado principalmente en el correo electrónico. Permite autenticar mensajes (mediante firma digital), cifrar datos y realizar otras funciones adicionales como eliminación

LA EMPRESA Y LA LEY

Zimermann fue acusado de violar la **ley de exportación de software criptográfico** de Estados Unidos, aunque la acusación fue retirada. En 1996, creó la empresa **PGP Inc.**, que sería adquirida por **Network Associates Inc.** en 1997, aunque en 2002 elimina su línea PGP y al software lo adquiere una nueva empresa, **PGP Corporation**, que fue comprada por **Symantec** en 2010.

segura. Trabaja de manera híbrida, combinando algoritmos simétricos, asimétricos y de hash, y además puede aplicar compresión de datos (a fin de reforzar la seguridad contra el criptoanálisis). También puede usarse para proteger datos locales. A partir de PGP, **IETF** creó el estándar **OpenPGP**.

PGP crea una clave de sesión de un único uso que se aplica a un algoritmo simétrico para cifrar el mensaje, y cifra luego la clave de sesión con la clave pública del receptor (esto adjunto al mensaje). Para descifrar se realiza el proceso inverso. Las claves se almacenan en archivos llamados **llaveros** (uno de públicas y otro de privadas).

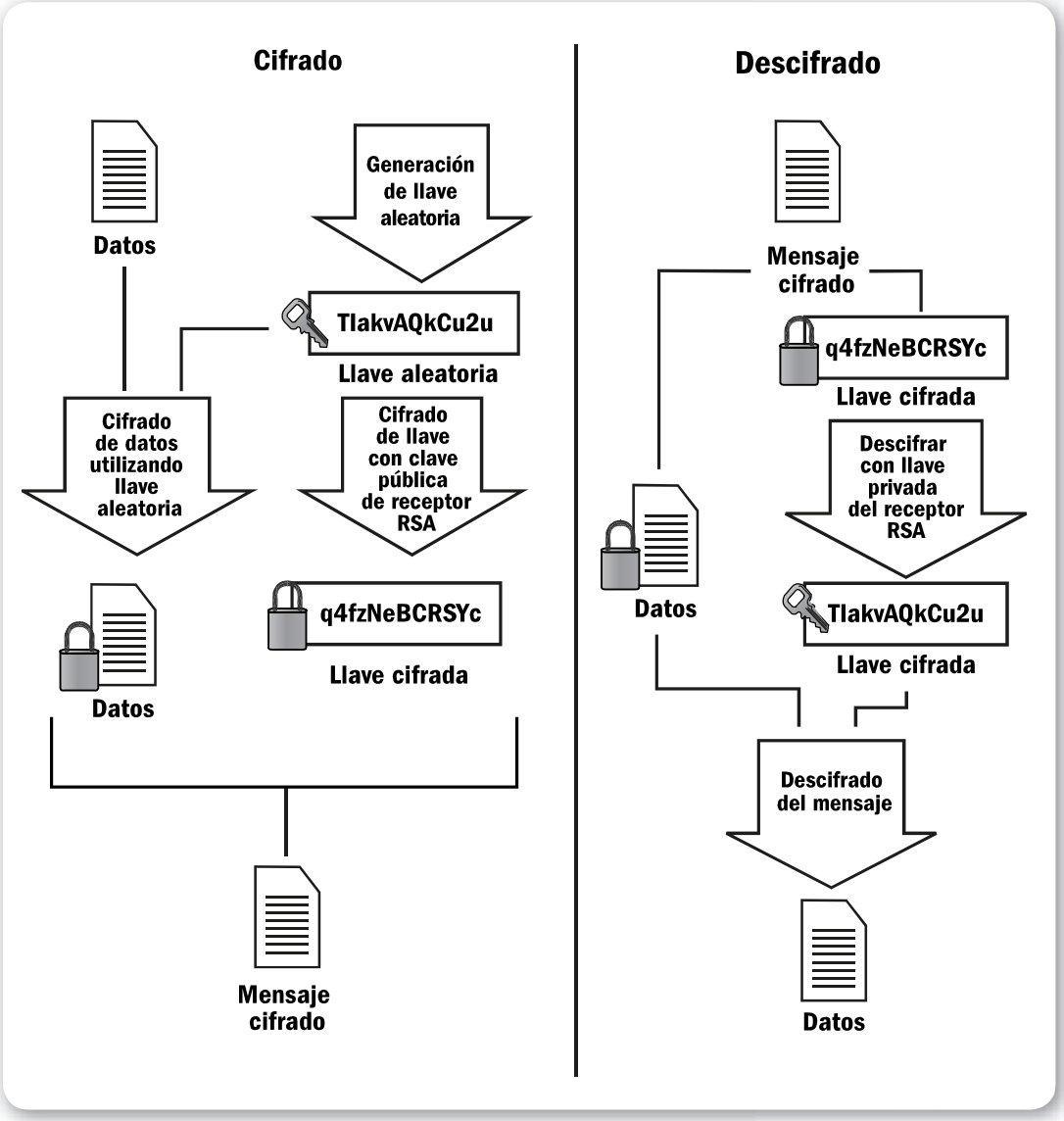


Figura 18. Esquema de cifrado y descifrado con **PGP** utilizando criptografía híbrida.

La verificación de las claves públicas se realiza mediante un sistema de **anillos de confianza**, que implica que cada usuario puede avalar la identidad del dueño de una clave en función del conocimiento que tiene sobre él. De este modo, si una persona de nuestra confianza avala a otra, ésta última pasa a tener para nosotros una mayor garantía de identidad (**confianza transitiva**), pudiendo definirse así varios niveles de confianza (no es lo mismo que 1 o 100 personas avalen una identidad).

IPSec

IPsec (*Internet Protocol security*) es una serie de protocolos que buscan garantizar la seguridad del protocolo IP (capa 3 del modelo OSI) añadiendo cifrado y autenticación, e incluyendo también protocolos para el intercambio de llaves. No requiere modificaciones en las aplicaciones para ser usado, dado que trabaja en el nivel de red.

Fue desarrollado para **IPv6** y adaptado a **IPv4**, es el estándar para **VPNs** y es independiente de los algoritmos de cifrado. Se compone estructuralmente de una arquitectura, un conjunto de protocolos y una serie de mecanismos de autenticación y cifrado. Los protocolos a los que se hace referencia son:

- **AH** (*Autentication Header*): provee autenticación e integridad, pero no confidencialidad.
- **ESP** (*Encapsulating Security Payload*): provee confidencialidad.
- **ISAKMP** (*Internet Security Association & Key Management Protocol*): mecanismo de intercambio de claves para cifrado y autenticación de **AH** y **ESP**. Incluye **IKE** (*Internet Key Exchange*) y utiliza **Diffie-Hellman**.

IPsec puede operar de dos maneras:

- **Modo Transporte**: donde se protege el dato del paquete IP (se cifran solo los datos, no la cabecera). Sirve para comunicación punto a punto entre equipos, proveyendo confidencialidad, y requiere implementar el protocolo en ambos extremos.

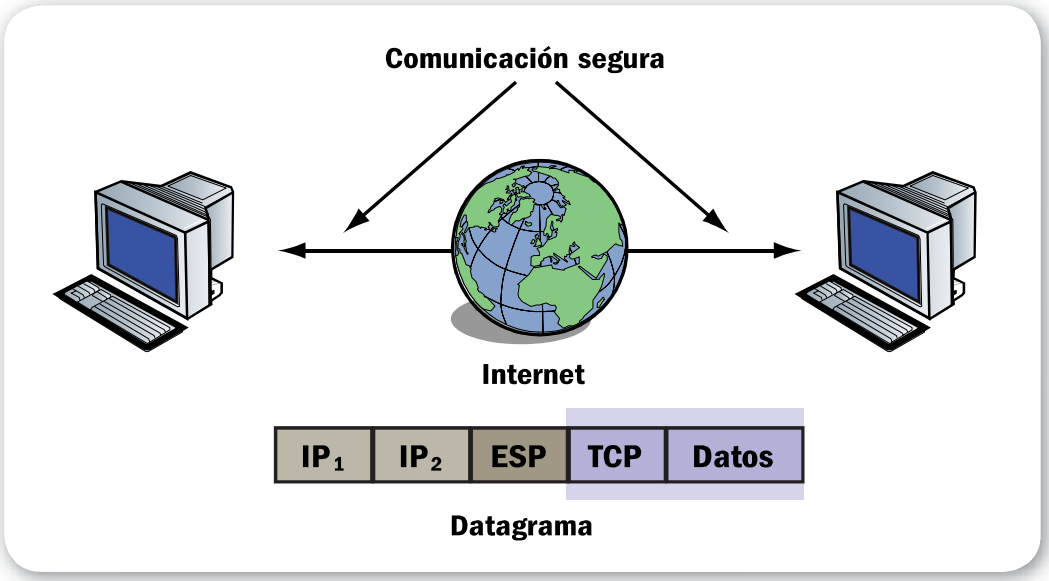


Figura 19. En el **modo transporte** de IPSec se protegen solo los datos.

- **Modo Túnel**: se protegen los paquetes IP completos, sirve para comunicación punto a punto entre **gateways**, proveyendo confidencialidad en el túnel sin necesidad de que los equipos entiendan el protocolo. A los paquetes se les agrega una nueva cabecera (la del túnel) y se cifra todo el paquete incluyendo la cabecera original.

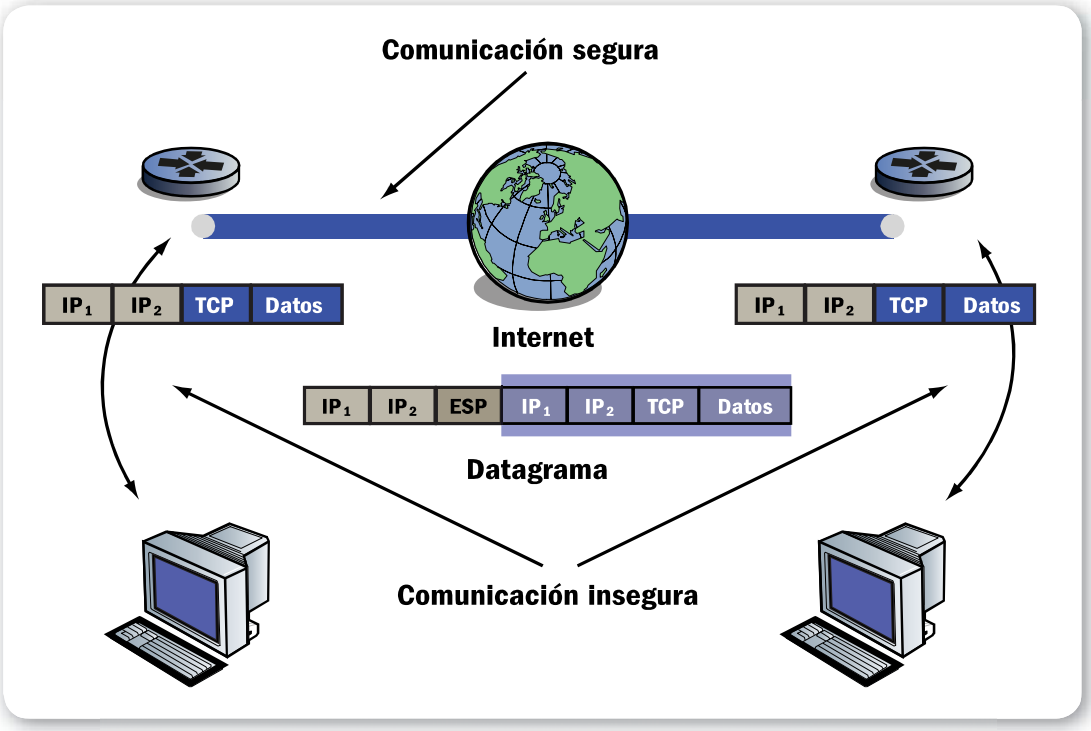


Figura 20. En el **modo túnel** de IPSec se protege todo el paquete.

Para establecer la comunicación cifrada, el modo de comunicación y el acuerdo de claves se utiliza la llamada **SA** (*Security Associations*), ya sea manualmente o a través de un protocolo. Una SA sería una conexión lógica entre dos sistemas y es unidireccional, ya que una comunicación IPSec está formada por dos SA. Cada SA está conformada por tres parámetros:

- **SPI** (*Security Parameter Index*): identificador único de cada SA.
- **IP-DA** (*IP Destination Address*): dirección del receptor (**unicast**, **multicast** o **broadcast**).
- **SP** (*Security Protocol*): el modo de operación (transporte, túnel), el protocolo usado (**ESP**, **AH**). Solo se puede especificar uno de los dos protocolos, por lo que pueden necesitarse hasta cuatro SA para una conexión si se requieren más.

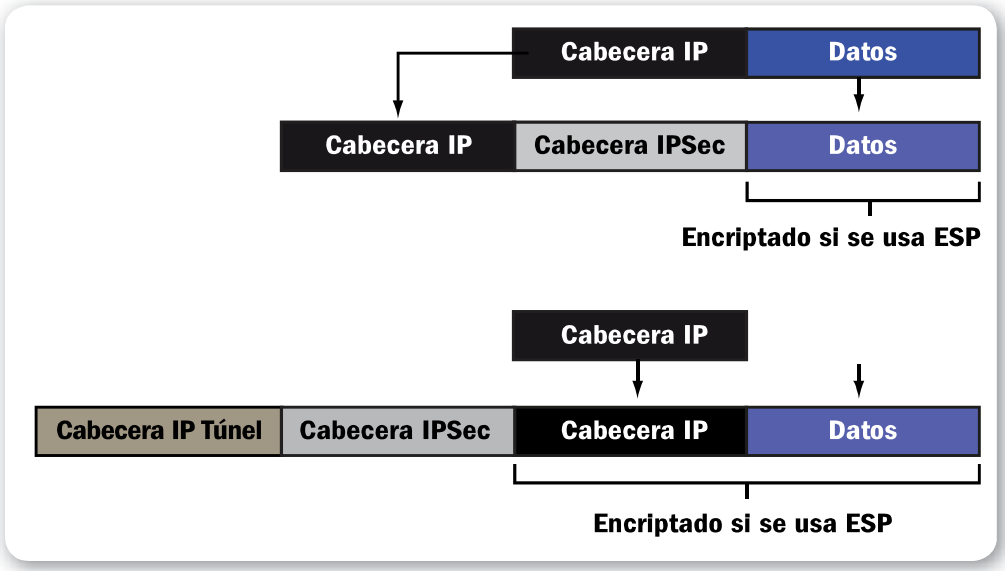


Figura 21. Estructuras del datagrama IP para **modo transporte** (arriba) y **modo túnel** (abajo).

RESUMEN

Vimos protocolos y sistemas de autenticación para realizar comunicaciones seguras basadas en el correcto uso de la criptografía, lo que permite la transferencia segura de información y garantías de autenticación sobre canales inseguros. Conocimos el uso de los protocolos AAA, SSL/TLS, PAP, CHAP, EAP e IPSec. También vimos PGP, que cumplió un importante papel en la práctica de la criptografía.

Actividades

TEST DE AUTOEVALUACIÓN

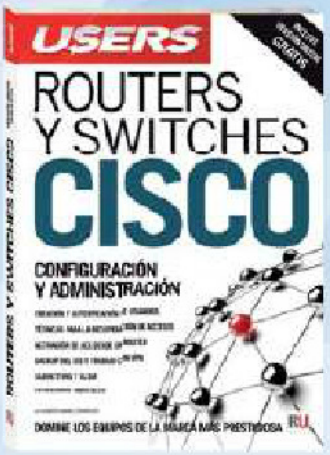
1. ¿Cuáles son los pasos del acceso a un sistema?
2. ¿Cuáles son los tres tipos de factores de autenticación?
3. ¿Qué es un protocolo criptográfico?
4. ¿Para qué sirve un sistema AAA?
5. ¿Para qué se utiliza típicamente RADIUS?
6. ¿Qué diferencia principal hay entre Kerberos y Sesame?
7. ¿Qué son PAP, CHAP y EAP?

EJERCICIOS PRÁCTICOS

1. Investigue, para un sistema operativo cualquiera, el proceso completo de control de acceso y sus variantes.
2. Estudie los distintos elementos de la autenticación biométrica.
3. Realice una conexión de acceso remoto a cualquier servidor, probando las distintas opciones de autenticación permitidas.
4. Instale GPG, genere un par de claves y utilícelo para firmar correos.
5. Investigue las vulnerabilidades conocidas en los últimos años sobre el protocolo SSL/TLS.

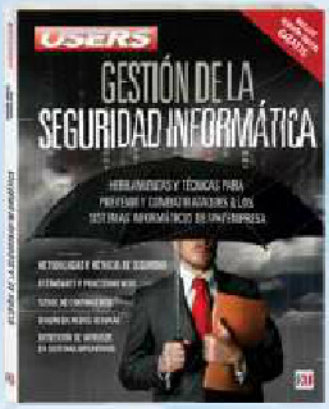
PROFESOR EN LÍNEA

Si tiene alguna consulta técnica relacionada con el contenido, puede contactarse con nuestros expertos: profesor@redusers.com



Capacítese para obtener una certificación Cisco y amplíe sus oportunidades laborales en el rubro de las telecomunicaciones.

→ 320 páginas / ISBN 978-987-1949-34-2



Conozca herramientas y técnicas necesarias para prevenir y combatir ataques a los sistemas informáticos de una empresa.

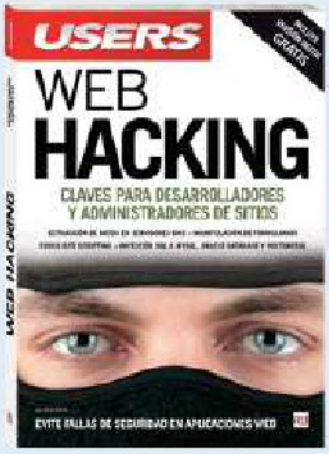
→ 192 páginas / ISBN 978-987-1949-30-4

Este libro revela técnicas y herramientas indispensables a la hora de encarar una estrategia de marketing en medios sociales.

→ 192 páginas / ISBN 978-987-1949-32-8

Con los mismos datos, puede obtener resultados muy diferentes: implemente herramientas interactivas de inteligencia empresarial.

→ 192 páginas / ISBN 978-987-1949-29-8



Indispensable para desarrollares y administradores de sitios, este libro explica las técnicas de ataque utilizadas por los hackers.

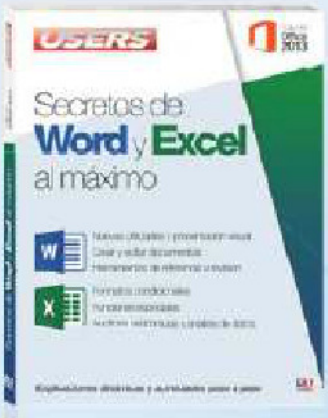
→ 320 páginas / ISBN 978-987-1949-31-1

El libro indicado para quienes buscan aprender a confeccionar y administrar bases de datos en Microsoft Access desde cero.

→ 192 páginas / ISBN 978-987-1949-27-4

Aproveche la versatilidad de PowerPoint para crear presentaciones y especialícese en el manejo de bases de datos con Access.

→ 192 páginas / ISBN 978-987-1949-28-1

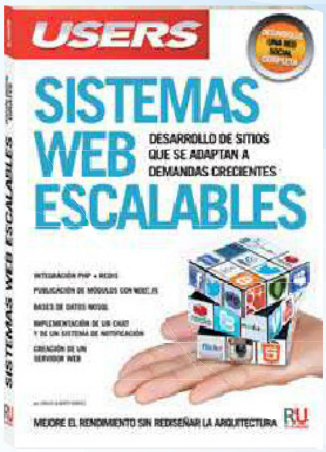


Manténgase actualizado: conozca las nuevas herramientas de Word y trabaje con las funciones avanzadas de Excel

→ 192 páginas / ISBN 978-987-1949-26-7

Aprenda a utilizar Excel 2013 y desarrolle planillas adaptadas a sus necesidades de registro y seguimiento de información.

→ 192 páginas / ISBN 978-987-1949-25-0



Cree su propia red social e implemente un sistema capaz de evolucionar en el tiempo y responder al crecimiento del tráfico.

→ 320 páginas / ISBN 978-987-1949-20-5

Conozca la integración con redes sociales y el trabajo en la nube, en aplicaciones modernas y más fáciles de utilizar.

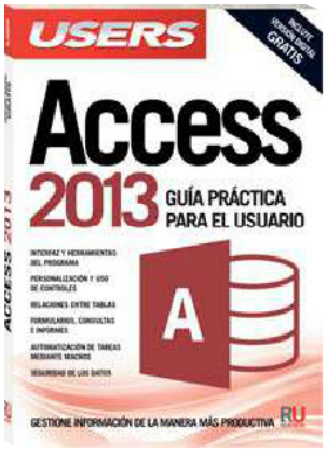
→ 320 páginas / ISBN 978-987-1949-21-2

Conozca claves y herramientas más potentes de esta nueva versión de Excel y logre el máximo de efectividad en sus planillas

→ 320 páginas / ISBN 978-987-1949-18-2

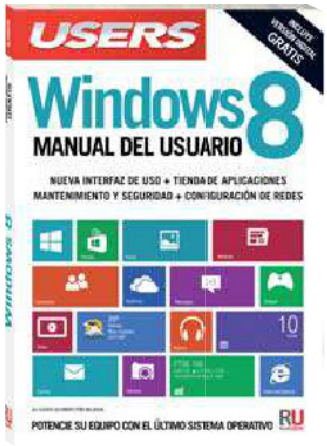
Consejos y secretos indispensables para ser un técnico profesional e implementar la solución más adecuada a cada problema

→ 320 páginas / ISBN 978-987-1949-19-9



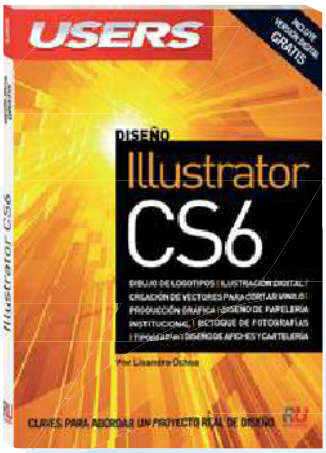
Simplifique tareas cotidianas de la manera más productiva y obtenga información clave para la toma de decisiones.

→ 320 páginas / ISBN 978-987-1949-17-5



Acceda a consejos indispensables y aproveche al máximo el potencial de la última versión del sistema operativo más utilizado.

→ 320 páginas / ISBN 978-987-1949-09-0

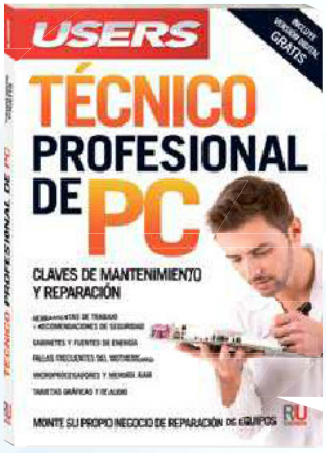


La mejor guía a la hora de generar piezas de comunicación gráfica, ya sean para web, dispositivos electrónicos o impresión.

→ 320 páginas / ISBN 978-987-1949-04-5

Aprenda a simplificar su trabajo, convirtiendo sus datos en información necesaria para solucionar diversos problemas cotidianos.

→ 320 páginas / ISBN 978-987-1949-08-3

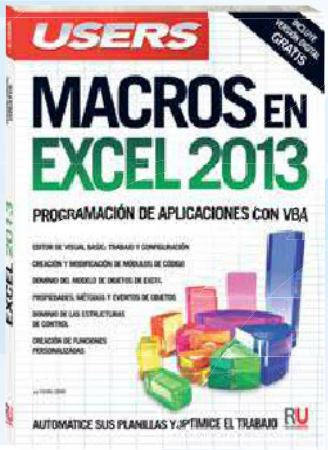


Acceda a consejos útiles y precauciones a tener en cuenta al afrontar cualquier problema que pueda presentar un equipo.

→ 320 páginas / ISBN 978-987-1949-02-1

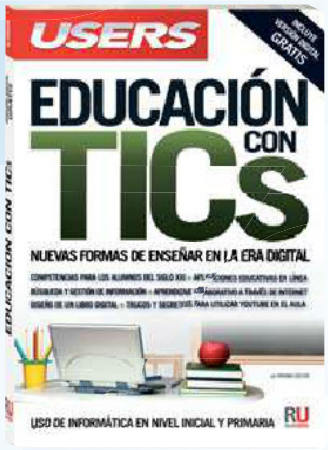
El libro indicado para enfrentar los desafíos del mundo laboral actual de la mano de un gran sistema administrativo-contable.

→ 352 páginas / ISBN 978-987-1949-01-4



Un libro ideal para ampliar la funcionalidad de las planillas de Microsoft Excel, desarrollando macros y aplicaciones VBA.

→ 320 páginas / ISBN 978-987-1857-99-9

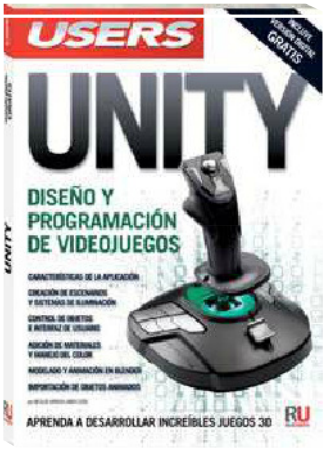


Un libro para maestros que busquen dinamizar su tarea educativa integrando los diferentes recursos que ofrecen las TICs.

→ 320 páginas / ISBN 978-987-1857-95-1

Libro ideal para introducirse en el mundo de la maquetación, aprendiendo técnicas para crear verdaderos diseños profesionales.

→ 352 páginas / ISBN 978-987-1857-74-6



Esta obra reúne todas las herramientas de programación que ofrece Unity para crear nuestros propios videojuegos en 3D.

→ 320 páginas / ISBN 978-987-1857-81-4

Esta obra nos enseña sobre el diseño y prueba de circuitos electrónicos, sin necesidad de construirlos físicamente.

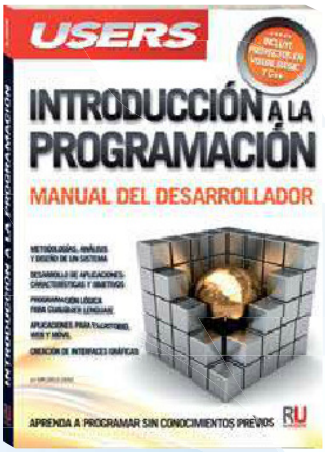
→ 320 páginas / ISBN 978-987-1857-72-2

Obra imperdible para crear infraestructura virtual con las herramientas de Vmware según los requerimientos de cada empresa.

→ 320 páginas / ISBN 978-987-1857-71-5

Esta obra reúne todos los conocimientos teóricos y prácticos para convertirse en un técnico especializado en Windows.

→ 320 páginas / ISBN 978-987-1857-70-8



Libro ideal para iniciarse en el mundo de la programación y conocer las bases necesarias para generar su primer software.

→ 384 páginas / ISBN 978-987-1857-69-2

CURSOS

CON SALIDA LABORAL

Los temas más importantes del universo de la tecnología, desarrollados con la mayor profundidad y con un despliegue visual de alto impacto: explicaciones teóricas, procedimientos paso a paso, videotutoriales, infografías y muchos recursos más.

- 25 Fascículos
- 600 Páginas
- 2 DVDs / 2 Libros

Curso para dominar las principales herramientas del paquete Adobe CS3 y conocer los mejores secretos para diseñar de manera profesional. Ideal para quienes se desempeñan en diseño, publicidad, productos gráficos o sitios web.

- 25 Fascículos
- 600 Páginas
- 4 CDs

Obra ideal para ingresar en el apasionante universo del diseño web y utilizar Internet para una profesión rentable. Elaborada por los máximos referentes en el área, con infografías y explicaciones muy didácticas.

Brinda las habilidades necesarias para planificar, instalar y administrar redes de computadoras de forma profesional. Basada principalmente en tecnologías Cisco, busca cubrir la creciente necesidad de profesionales.

- 25 Fascículos
- 600 Páginas
- 3 CDs / 1 Libros

Criptografía cuántica

En este apéndice iremos un paso más allá de la criptografía convencional por medio de una propuesta que combina técnicas de la física moderna con el cifrado de datos, orientado a la comunicación segura.

Funcionamiento

En estos tiempos, la palabra **cuántica** utilizada para adjetivar conceptos hace dudar a cualquier persona racional que se precie de sí. Abundan términos new age como "curación cuántica", "medicina cuántica", "meditación cuántica" y otros tantos, haciendo alarde pseudocientífico sobre la base de los indiscutibles avances de la física moderna.

> LA CRIPTOGRAFÍA CUÁNTICA IMPLICA LA APLICACIÓN DE LOS PRINCIPIOS DE LA MECÁNICA CUÁNTICA

La **criptografía cuántica** no se encuentra entre las categorías discutibles ni subjetivas, sino que implica de manera directa la aplicación de los principios de la **mecánica cuántica** a la seguridad de las comunicaciones. Su principal propiedad es que puede garantizar la protección contra escuchas en una comunicación y durante la creación de la clave de sesión.

Y es que la criptografía cuántica es una consecuencia directa del **principio de incertidumbre de Heisenberg**, según el cual al medir una variable en un sistema de escala subatómica, se ve afectada la medición. Esto la diferencia de la criptografía asimétrica tradicional, que se basa en la complejidad de funciones matemáticas.

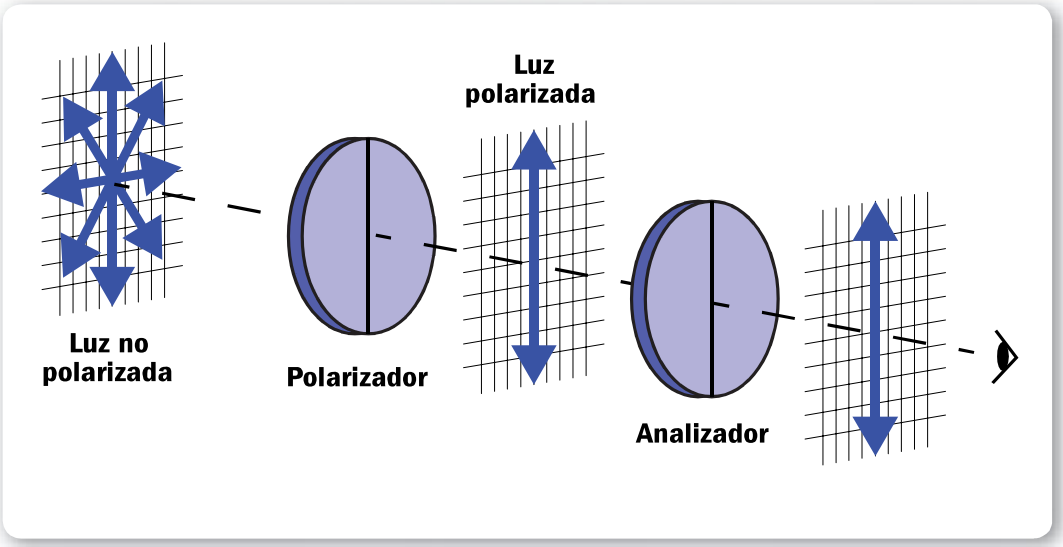


Figura 1. Las partículas usadas en criptografía cuántica son los **fotones**, cuyos estados cuánticos son los **estados de polarización**, ya que la luz puede polarizarse de manera que solo sea detectada por un filtro con su mismo sentido.

Protocolo BB84

Si bien las primeras ideas se gestaron en los años 70, recién en 1984 se publicó el primer protocolo cuántico, **BB84** (C. Bennett y G. Brassard). Se basa en la transmisión de fotones polarizados a través de un canal cuántico (como una fibra óptica) y un canal público (que puede no ser cuántico) para el envío de la información necesaria para la definición de la clave compartida. En ambos canales se asume la existencia de posibles intrusos.

Cada bit será representado por un **fotón**, y se codificarán mediante **estados no ortogonales**, como rectilíneos (**+**) o diagonales (**x**), que conformarán las **bases**. Ambas partes (**A** y **B**) pueden generar y transmitir fotones polarizados. El protocolo opera así:

1. **A** genera una secuencia aleatoria de bases y las utiliza para enviar a **B** (por el canal cuántico) un fotón polarizado al azar por cada base. **A** debe registrar la polarización con la que se emitió cada fotón.
2. **B** no sabe qué bases usó **A** para la generación, por lo que debe medir la polarización de los fotones con una base aleatoria entre las posibles (**+** o **x**) y registrar los resultados. Según la mecánica cuántica, si el fotón fue creado con una base distinta, el resultado es azaroso (de hecho la medición polariza el resultado y se pierde el dato original).
3. Ambos se envían por el canal público las bases utilizadas para generar y medir. Luego, descartan las mediciones en las que no coincidieron sus bases (en promedio, la mitad) y los bits restantes conforman la clave secreta.
4. Si hubo un error (o intruso) en el canal cuántico, la polarización pudo alterarse, por lo que también deben verificar que los bits que no se descartaron coincidan con lo que se envió. Un intruso no podría generar los mismos fotones originales por el **teorema de no-clonación**, según el cual es imposible reproducir la información sin conocer el estado cuántico que la describe. Para detectar una posible intrusión, **A** y **B** se revelan mutuamente porciones de la clave obtenida y, si difieren lo suficiente, anulan la comunicación asumiendo una intrusión.
5. **A** y **B** envían mensajes cifrados con la clave secreta, a través del canal cuántico o del público.

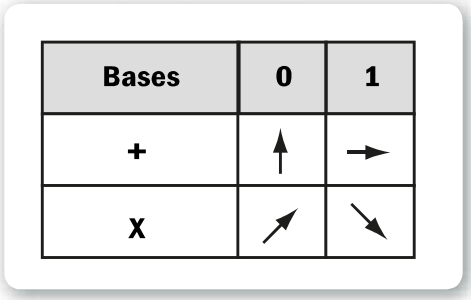


Figura 2. Cada bit se codifica mediante estados no ortogonales rectilíneos (**+**) y diagonales (**x**), que conforman las bases.

Protocolo E91

Otro protocolo cuántico es **E91** (A. Ekert, 1991), que se basa en pares de **fotones entrelazados** y funciona de forma similar a BB84, con la diferencia de que requiere adicionalmente una fuente que genere pares de fotones entrelazados para que uno llegue a **A** y otro a **B** (la fuente puede tenerla cualquiera de ellos o un tercero de confianza). Al medir cada uno la polarización de sus fotones, ambos tendrán resultados opuestos, por principio de la mecánica cuántica. El resto funciona como en BB84 (intercambio de bases, detección de intrusos, etcétera).

La ventaja de E91 se halla en que la clave es creada de modo realmente aleatorio, pues no es posible conocer previamente la polarización que tendrá cada fotón. En definitiva, la criptografía cuántica se perfila hacia el futuro, al punto de que muchos físicos ya trabajan en métodos para poder usarla con dispositivos móviles. La técnica más reciente es **rfiQKD** (*reference frame independent quantum key distribution*), que funciona incluso con altos niveles de ruido en el canal.

ETAPA NO COMERCIAL

La critografía cuántica no está aún en su etapa comercial, dada la complejidad para construir los equipos que puedan realizar las funciones de emisión y recepción de información mediante haces de luz al nivel que se los requiere. En 2002 se creó el primer producto comercializable, aunque por el momento no se ha masificado.

Esteganografía

En este apéndice veremos otro campo de aplicación de las técnicas criptográficas que se orienta más al ocultamiento de la información ante observadores ocasionales que a su secreto.

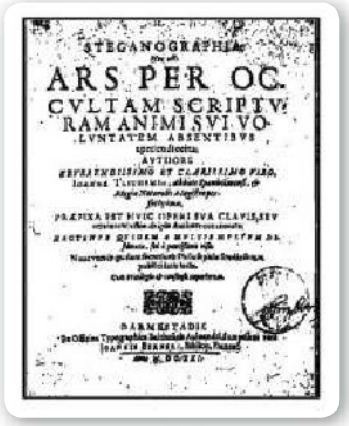
Desarrollo y evolución

Podemos considerar a la **esteganografía** como una rama de la criptología en la que se utilizan técnicas para ocultar mensajes dentro de otros mensajes (portadores) de manera que no se pueda detectar su existencia.

Esto crea un **canal encubierto** (**covert channel**) para la transmisión de información secreta que permita a ciertos usuarios recuperar mensajes solo si saben cómo estos fueron guardados dentro del portador, pasando desapercibidos para otros usuarios que tengan acceso al canal.

Así como sucede con la criptografía, podemos distinguir entre la esteganografía **clásica** y la **moderna**, donde la seguridad de la primera se basa en el secreto del canal y la técnica usada; y en la segunda se usan canales digitales como archivos de texto, de imágenes, de música, de video y demás.

La primera referencia a la esteganografía se encuentra en el libro **Las historias** de Herodoto, donde describe a un personaje que toma un cuadernillo de dos tablillas, raspa la cera que las cubre y graba en la madera un mensaje, que luego vuelve a cubrir con cera. En el mismo libro se describe el envío de mensajes escritos en la cabeza rapada de un mensajero, que luego de crecerle el pelo, viajaba a entregar el mensaje.


STEGANOGRAPHIA
ARS PER OC
CVLTAM SCRIPTV
RAM ANIMI SVI VO
LVNTATEM ABSENTIVS

En el siglo XV, Giovanni della Porta diseñó un método para esconder un mensaje en un huevo cocido, que solo podía ser leído si se pelaba su cáscara, y consistía en escribir en su superficie con una tinta especial.

Figura 1. Steganographia (Johannes Trithemius, siglo XVI) es el libro que dio origen a la palabra. Abarca métodos de ocultamiento de mensaje y algunas cuestiones esotéricas.

Técnicas

En todos los casos se busca insertar bits en zonas de los archivos o paquetes de datos de red, que una vez recuperados representen un mensaje codificado ya sea en el mismo tipo de formato o en otro.

Las técnicas modernas incluyen, por ejemplo, el uso de ocultamiento de datos en una imagen a modo de marcas de agua, los algoritmos de transformación específicos y la inserción de datos en los bits menos significativos de imágenes y otros archivos multimedia que representen escalas (de colores, sonidos, etcétera).

Es importante tener en cuenta que la información insertada debe tener una proporción adecuada para al tamaño del medio portador. Así, insertar un texto de un millón de caracteres en una imagen en formato JPG de 640 x 480 píxeles no será una gran idea, en tanto que puede ocultarse una imagen completa dentro de otra más grande o de un video, según la técnica.

También puede incluirse información directamente dentro de los documentos de texto, agregando espacios en blanco de forma preestablecida, cosa que pasará desapercibida para un lector que no sepa de la existencia de información oculta.

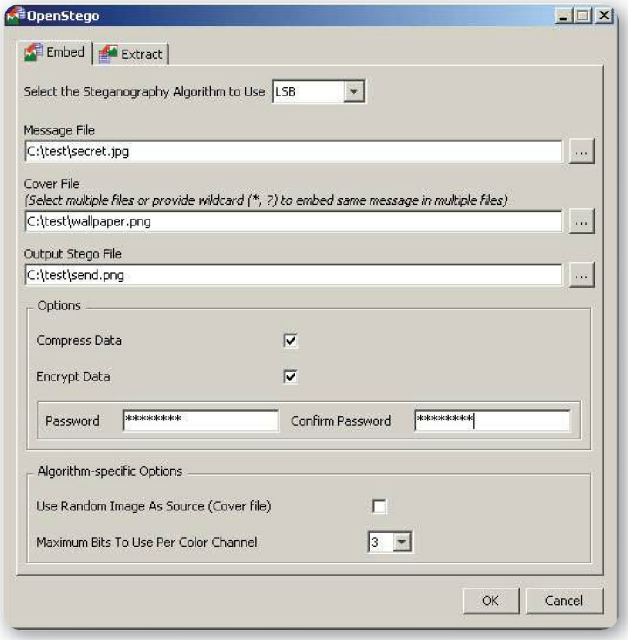


Figura 2. OpenStego es un software con licencia libre que permite insertar mensajes en archivos de imágenes.

En líneas generales, puede incluirse información en muchos tipos de archivos, dependiendo en gran medida de su formato específico. Incluso en archivos binarios ejecutables se dispone de espacios para agregar texto en el propio formato, aunque sería demasiado evidente utilizar las características que ya prevén los formatos. Siempre será

necesario procesar el archivo en cuestión para que se le agregue la información correspondiente al mensaje a transmitir.

Nuevas técnicas sugieren la inserción de pequeños retardos en las comunicaciones de red para codificar información. Una manera muy útil de agregar seguridad al envío de mensajes mediante un portador es cifrar previamente esa información con algoritmos bien probados (como podría ser AES-256), que no aumentarían el tamaño del mensaje e impedirían su interpretación de forma directa incluso en caso de que fuera recuperado por medio de técnicas de análisis.

Nivel de protección	1	2	3	4
Uso de algoritmo	Sí	Sí	Sí	Sí
Uso de clave	No	Sí	Sí	Sí
Influencia de la clave en la distribución de bits del mensaje	No	No	Sí	Sí
Influencia de la clave en la distribución y selección de bits del mensaje	No	No	No	Sí

Figura 3. Pueden conseguirse distintos niveles de seguridad en función de la forma en que se construya el **esquema esteganográfico**.

El contrapunto de la esteganografía es entonces el **esteganálisis**, que busca detectar en principio si existe información adicionada en un determinado medio y, en caso de que exista, determinar su contenido. Obviamente no es posible conseguir una detección totalmente efectiva, sino que los resultados del esteganálisis son probabilísticos. E incluso en caso de que no se detecte nada, no puede garantizarse que no exista información oculta.

LA ESTEGANOGRAFÍA Y EL PRISIONERO

La utilidad de la esteganografía se puede ver en el llamado **problema del prisionero** (que no es lo mismo que el **dilema del prisionero**), donde dos prisioneros quieren comunicarse para planear un escape y solo pueden pasarse mensajes por medio del guardia, por lo que sus mensajes no pueden tener contenido explícito y requieren establecer un canal encubierto.

USERS

CRIPTOGRAFÍA

En esta obra se realiza un análisis de las distintas áreas que conforman la criptografía: marco histórico, bases teóricas, aplicaciones y elementos prácticos. Al mismo tiempo, se emprende un recorrido que abarca tanto la criptografía clásica como los algoritmos y protocolos modernos, utilizados cotidianamente.

Se trata de un material que puede resultar de utilidad tanto para el lector que, sin demasiados conocimientos teóricos, busque comprender en qué consisten las técnicas de protección de datos, como así también para profesionales de seguridad informática que quieren profundizar sus conocimientos.

> **La criptografía permite proteger la información contra accesos no autorizados, garantizando su confidencialidad, autenticidad e integridad.**

EN ESTE LIBRO APRENDERÁ:

- **Criptografía clásica:** sistemas históricos, sustitución y transposición. Dispositivos y máquinas de cifrado.
- **Criptografía moderna:** teoría de la información, teoría de números, complejidad algorítmica, congruencia, factorización y números primos.
- **Cifrado simétrico y asimétrico:** cifrado de bloque y de flujo. Algoritmos 3DES, AES, IDEA, RC4/5, Blowfish, Diffie-Hellman, RSA, El Gamal y curvas elípticas.
- **Funciones hash:** algoritmos MD5, SHA, Tiger, Whirlpool, RIPEMD. Ataques a funciones unidireccionales. Códigos de autenticación.
- **Public Key Infrastructure (PKI):** firmas y certificados digitales. Estandar X.509. PKCS. DSS y DSA.
- **Protocolos y sistemas de autenticación:** sistemas AAA. RADIUS, Diameter y TACACS+. Kerberos y Sesame. SSL/TLS, PGP, IPSec.

SOBRE EL AUTOR

Federico Pacheco es especialista en seguridad de la información orientado a la consultoría, la investigación y la educación. Cuenta con más de 10 años de experiencia docente en distintos niveles de enseñanza y ha generado gran cantidad de material educativo para instituciones y para la comunidad en general.

NIVEL DE USUARIO

Básico / Intermedio

CATEGORÍA

Seguridad

ISBN 978-987-1949-35-9

REDUSERS.com
En nuestro sitio podrá encontrar noticias relacionadas y también participar de la comunidad de tecnología más importante de América Latina.

PROFESOR EN LÍNEA
Ante cualquier consulta técnica relacionada con el libro, puede contactarse con nuestros expertos: profesor@redusers.com.