10. Capítulo 10: Conceptos de Seguridad en Redes

La seguridad en el ámbito del ecosistema digital es un concepto ligado a la implementación de estrategias y políticas que tienen como objetivo mitigar acciones que estén destinadas a vulnerar la integridad, confidencialidad y autenticidad de los datos. En su forma más simple, se ocupa de garantizar que un "usuario" de la red no pueda leer ni modificar datos que no sean dirigidos estrictamente a él, y que en caso de que un usuario "no autorizado" pueda cometer el ilícito, se pueda detectar el hecho y conservar las pruebas. Se puede decir que la seguridad también tiene que ver con proteger el acceso a servicios remotos no autorizados.

En este capítulo se comenzará definiendo los aspectos que trata la seguridad en las redes, para luego desarrollar la tecnología esencial subyacente, virtualmente, en todas las aplicaciones de seguridad en redes y computadores: la criptografía. Se utilizan dos técnicas fundamentales: cifrado simétrico y cifrado de clave pública, también conocido como cifrado asimétrico, ambas garantizan la confidencialidad del mensaje y serán descriptas en detalles. También se desarrollará el concepto de función Hash, cuya implementación permite implementar la autenticación de mensajes (verifica integridad) y la firma digital (no repudio). Por último, se describirá el alcance y funcionamiento de uncortafuegos (Firewall), desarrollando en mayor profundidad la implementación de una herramienta de firewall de código abierto muy usada en el mundo: iptables.

10.1. Requisitos de Seguridad y Tipos de Ataques

Los problemas de seguridad de las redes se pueden dividir en términos generales en cuatro áreas interrelacionadas: confidencialidad, autenticación, no repudio y control de integridad. La confidencialidad, consiste en mantener la información fuera del alcance de usuarios no autorizados. Esto es lo que normalmente viene a la mente cuando la gente piensa en la seguridad de las redes. La autenticación se encarga de determinar (antes de hacerlo), con quién se está a punto de enviar información confidencial. El no repudio se encarga de la autenticidad de la firma digital, esto implica que un usuario que ha firmado digitalmente un documento, luego no pueda negar el mismo, por ejemplo, que argumente que él nunca generó ese archivo. Por último, el control de integridad tiene que ver con la forma en que podemos estar seguros de que un mensaje recibido realmente fue el que se envió, y no algo que un adversario malicioso modificó en el camino o ideó por su propia cuenta.

En el momento de identificar si la seguridad debe ser implementada en algún sector en particular de una red, la respuesta es que debe ser considerada a lo largo de todas la interred y debe atravesar todas las capas del modelo OSI. Cada capa debe ser protegida para contribuir a la protección total del ecosistema digital.

En la capa física se protegerá contra la intervención de las líneas de transmisión (o mejor aún, las fibras ópticas), se controlará el acceso a sitios sensibles

mediante datos biométricos, se implementarán sistemas de circuitos cerrados para monitorear las instalaciones, etc.

En la capa de enlace de datos, los paquetes de una línea punto a punto se pueden encriptar cuando se envíen desde una máquina y desencriptarse cuando lleguen a otra. Todos los detalles se pueden manejar en la capa de enlace de datos, sin necesidad de que las capas superiores se enteren de ello. Sin embargo, esta solución se viene abajo cuando los paquetes tienen que atravesar varios enrutadores, puesto que se tienen que desencriptar en cada enrutador, dentro del cual son vulnerables a posibles ataques. Además, no se contempla que algunas sesiones estén protegidas (por ejemplo, aquellas que involucren compras en línea mediante tarjeta de crédito) y otras no. Sin embargo, la encriptación de enlace (*"link Encryption"*), como se llama a este método, se puede agregar fácilmente a cualquier red y con frecuencia es útil.

En la capa de red se pueden instalar firewalls para impedir el ingreso de paquetes potencialmente peligrosos. La seguridad de IP también funciona en esta capa.

En la capa de transporte se pueden encriptar conexiones enteras de un extremo a otro; es decir, de proceso a proceso. Para lograr una máxima seguridad, se requiere que todo el camino se encuentre protegido (extremo a extremo).

Por último, los asuntos como la autentificación de usuario y el no repudio sólo se pueden manejar en la capa de aplicación.

Una forma útil de clasificar los ataques a la seguridad (RFC 2828) es en términos de ataques pasivos y ataques activos. Un ataque pasivo intenta averiguar o hacer uso de información del sistema, pero sin afectar a los recursos del mismo. Un ataque activo intenta alterar los recursos del sistema o influir en su funcionamiento.

Ataques Pasivos

Los ataques pasivos consisten en escuchas o monitorizaciones de las transmisiones. La meta del oponente es la de obtener la información que está siendo transmitida. La divulgación del contenido de un mensaje y el análisis de tráfico constituyen dos tipos de ataques pasivos.

La divulgación del contenido de un mensaje se entiende fácilmente. Una conversación telefónica, un mensaje de correo electrónico o un fichero transferido pueden contener información sensible o confidencial. Por ello, desearíamos impedir que un oponente averigüe el contenido de estas transmisiones.

Un segundo tipo de ataque pasivo, el análisis de tráfico, es más sutil. Suponga que disponemos de un medio para enmascarar el contenido de los mensajes u otro tipo de tráfico de información, de forma que, aunque los oponentes capturasen el mensaje, no podrían extraer la información del mismo. La técnica más común para enmascarar el contenido es el cifrado. Pero incluso si tenemos protección por cifrado, el oponente podría todavía observar el patrón de estos mensajes. El oponente podría determinar la localización y la identidad de los computadores que se están comunicando y observar la frecuencia y la longitud de los mensajes intercambiados. Esta información podría serle útil para averiguar la naturaleza de la comunicación que se está realizando.

Los ataques pasivos son muy difíciles de detectar, ya que no suponen la alteración de los datos. Normalmente, el tráfico de mensajes es enviado y recibido de forma aparentemente normal y ni el emisor ni el receptor son conscientes de que un tercero haya leído los mensajes u observado el patrón de tráfico. Sin embargo, es factible impedir el éxito de estos ataques, usualmente mediante cifrado. De esta manera, el énfasis en la defensa contra estos ataques se centra en la prevención en lugar de en la detección.

Ataques Activos

Los ataques activos suponen alguna modificación del flujo de datos o la creación de flujos falsos. Los podemos clasificar en 4 categorías: enmascaramiento, retransmisión, modificación de mensajes y denegación de servicio.

- ✓ Un enmascaramiento tiene lugar cuando una entidad pretende ser otra entidad diferente. Un ataque por enmascaramiento normalmente incluye una de las otras formas de ataques activos. Por ejemplo, se pueden capturar secuencias de autenticación y retransmitirlas después de que tenga lugar una secuencia válida, permitiendo así obtener privilegios adicionales a otra entidad autorizada con escasos privilegios mediante la suplantación de la entidad que los posee.
- ✓ La retransmisión supone la captura pasiva de unidades de datos y su retransmisión posterior para producir un efecto no autorizado.
- ✓ La modificación de mensajes significa sencillamente que algún fragmento de un mensaje legítimo se modifica o que el mensaje se retrasa o se reordena para producir un efecto no autorizado. Por ejemplo, un mensaje con un significado «Permitir a Juan García leer el fichero confidencial de cuentas» se modifica para tener el significado «Permitir a Alfredo Castaño leer el fichero confidencial de cuentas»
- ✓ La denegación de servicio impide o inhibe el normal uso o gestión de servicios de comunicación. Este ataque puede tener un objetivo específico. Por ejemplo, una entidad puede suprimir todos los mensajes dirigidos a un destino concreto (por ejemplo, al servicio de vigilancia de seguridad). Otro tipo de denegación de servicio es la interrupción de un servidor o de toda una red, bien deshabilitando el servidor o sobrecargándolo con mensajes con objeto de degradar su rendimiento.

Los ataques activos presentan características opuestas a las de los ataques pasivos. Mientras que un ataque pasivo es difícil de detectar, existen medidas para impedir que tengan éxito. Por otro lado, es bastante difícil impedir ataques activos de forma absoluta, ya que para hacerlo se requeriría protección física permanente de todos los recursos y de todas las rutas de comunicación. En su lugar, el objetivo consiste en primer lugar en mitigarlos, y en caso de que se produzcan, detectarlos y recuperarse de cualquier interrupción o retardo causados por ellos. Ya que la detección tiene un efecto disuasorio, también puede contribuir a la prevención.

10.2. Criptografía

En esta sección se desarrollan los conceptos básicos de la criptografía como una rama de la criptología. Se analizan los dos esquemas fundamentales de cifrado: de clave pública y de clave privada. Además, se expone la utilidad del criptoanálisis como una herramienta de verificación de validez de los algoritmos de cifrado.

10.2.1. Criptografía como una rama de la criptología

El término criptografía proviene del griego kryptós, "escondido", y graphos, "escritura"; y es el arte de ocultar los mensajes compuestos de signos convencionales, de manera que solo develen su significado a la luz de una clave secreta. Es la ciencia y el arte de reescribir el texto escrito, de manera que sea indescifrable, para quien no posee la clave.

La criptografía parece estar estrechamente relacionada a las comunicaciones electrónicas modernas. Sin embargo, la criptografía es una cuestión bastante antigua, como lo demuestran los jeroglíficos secretos usados en el año 2000 AC en el antiguo Egipto. Desde aquella época la criptografía ha estado presente de alguna forma en casi todas las culturas de lengua escrita.

En la Figura 10.2.1.1 se observan los campos en los que se divide la disciplina científica de la criptología, donde podemos observar que la criptografía es una rama de esta.

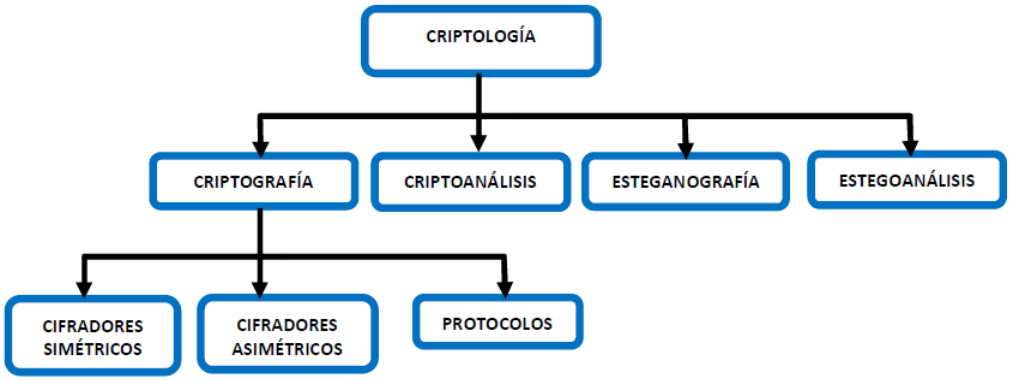

Figura 10.2.1.1: Áreas de desarrollo de la criptología

Criptografía: Se ocupa del estudio de los algoritmos, protocolos y sistemas que se utilizan para proteger la información y dotar de seguridad a las comunicaciones y a las entidades que se comunican.

Criptoanálisis: Se ocupa de intentar capturar el significado de los mensajes construidos mediante la criptografía sin tener autorización para ello. Podríamos decir que el criptoanálisis tiene un objetivo opuesto al de la criptografía. Su propósito es buscar el punto débil de las técnicas criptográficas para explotarlo y así reducir o eliminar la seguridad que

teóricamente aportaba esa técnica criptográfica. A cualquier intento de criptoanálisis se lo llama ataque. Un ataque tiene éxito, y se dice que el sistema se ha quebrado, cuando el atacante consigue romper la seguridad que la técnica criptográfica aportaba al sistema.

Criptosistemas

Un criptosistema, o sistema criptográfico, se puede definir como los fundamentos y procedimientos de operación que participan en el cifrado y descifrado de un mensaje. Todo sistema criptográfico es una tupla: {P, C, K, E y D} con las siguientes propiedades:

- ✓ P es el conjunto de todos los mensajes a transmitir. Es el espacio de textos planos.
- ✓ C es el conjunto de todos los mensajes cifrados. Es el espacio de textos cifrados.
- ✓ K es el conjunto de claves a utilizar. Es el espacio de claves.
- ✓ E es la familia de todas las funciones de cifrado. $E = \{E_k : k \in K\}$ y $E_k : P \rightarrow C$.
- ✓ D es la familia de todas las funciones de descifrado. $D = \{D_k : k \in K\}$ y $D_k : C \rightarrow P$.
- ✓ Para cada $e \in K$, hay un $d \in K$ tal que $D_d(E_e(p)) = p$ para todo $p \in P$. (e puede ser igual a d).

Todo criptosistema debe cumplir, al menos, tres requisitos básicos:

1. Todas las transformaciones de cifrado y descifrado, E_k y D_k, han de ser fácilmente calculables.

2. Los algoritmos de las transformaciones E_k y D_k tienen que ser fácil de implementar

3. La seguridad del sistema debe depender solo del secreto de las claves k y no de los algoritmos de las transformaciones E y D (Principio de Kerchoff).

Además, un buen criptosistema tiene que tener las siguientes cualidades: seguridad, autenticidad y no repudio:

- ✓ Seguridad: es la incapacidad para un criptoanalista de determinar el texto original, a partir del texto cifrado que haya podido interceptar.
- ✓ Autenticidad e integridad: es la incapacidad para un criptoanalista de improvisar, sustituir o modificar un texto cifrado C por un texto C', sin que el receptor lo detecte.
- ✓ No repudio: es la propiedad de que el emisor, después de haber enviado un mensaje, no pueda afirmar que este no es suyo. Esto se realiza por otros medios, como la firma digital, que se adaptan al criptosistema utilizado.

Algoritmos Simétricos (o de clave privada)

Es un método criptográfico en el cual se usa una misma clave para cifrar y descifrar los mensajes. Las dos partes que se comunican tienen un método compartido de cifrado/descifrado y han de ponerse de acuerdo de antemano sobre la clave a utilizar. Una

vez que ambas partes tienen acceso a esta clave, el remitente la usa para cifrar un mensaje, luego se lo envía al destinatario, y éste lo descifra con la misma clave. Toda la criptografía desde tiempos antiguos y hasta 1976 se basó exclusivamente en métodos simétricos. Estos aún están en vigencia, especialmente para cifrado de datos y verificación de integridad de mensajes.

Algoritmos Asimétricos (o de clave pública)

La criptografía asimétrica, es por definición, aquella que utiliza dos claves diferentes para cada usuario, una para cifrar que se la llama clave pública y otra para descifrar que es la clave privada. El nacimiento de la criptografía asimétrica se dio al estar buscando un modo más práctico de intercambiar las claves simétricas. En 1976, Diffie y Hellman propusieron una manera para hacer esto, sin embargo la criptografía asimétrica no tomó forma hasta que el popular método RSA de Rivest, Shamir y Adleman (publicado en 1978) se hiciera conocido. Su funcionamiento está basado en la imposibilidad computacional de factorizar números enteros grandes. Los algoritmos asimétricos se pueden emplear para aplicaciones como firma digital y establecimiento de clave, y también para el cifrado clásico.

Esquemas híbridos

En criptografía, los criptosistemas de clave pública son convenientes en cuanto a que no requieren que el emisor y el receptor compartan un secreto común a fin de comunicarse de forma segura (entre otras propiedades útiles). Sin embargo, a menudo dependen de complicados cálculos matemáticos y por ello son, en general, mucho más ineficientes que los criptosistemas simétricos. En muchas aplicaciones que usan criptografía de clave pública, el alto costo de la encriptación de mensajes largos puede ser prohibitivo. Un criptosistema híbrido es aquel que combina la conveniencia de un sistema de cifrado de clave pública con la eficiencia de un sistema de cifrado de clave simétrica.

Un sistema criptográfico híbrido se construye combinando los dos sistemas criptográficos:

- ✓ Un sistema de encapsulación de clave, que es un sistema de cifrado de clave pública.
- ✓ Un sistema de encapsulación de datos, que es un sistema de cifrado de clave simétrica.

Cabe observar que para mensajes muy largos la mayor parte del trabajo de cifrado/descifrado de datos se realiza por parte del sistema de clave simétrica (generalmente más eficiente en términos computacionales), mientras que el sistema de clave pública (generalmente más ineficiente en términos computacionales), se utiliza solo para cifrar/descifrar un valor de clave corto. La razón para usar ambas familias de algoritmos es que cada uno tiene sus fortalezas y debilidades específicas.

Protocolos Criptográficos

A grandes rasgos, los protocolos criptográficos tienen que ver con la implementación de algoritmos criptográficos. Los algoritmos simétricos y asimétricos pueden verse como los bloques constructivos con los cuales se implementan ciertas funcionalidades como, por ejemplo, la comunicación segura a través de Internet, esto implica asegurar la confidencialidad de los datos. Son ejemplos de protocolos criptográficos el esquema TLS (Transport Layer Security), que se utiliza en todos los navegadores web y él cual aplica al algoritmo asimétrico RSA en su implementación; por otro lado, el protocolo SSH (Secure Shell) se utiliza para realizar sesiones de consola seguras e implementa en su estructura al algoritmo asimétrico de Diffie-Hellman.

10.2.2. Cifrado simétrico

El cifrado simétrico, también denominado cifrado convencional o de clave única, era el único tipo de cifrado en uso antes de la introducción del cifrado de clave pública a finales de la década de los setenta. Innumerables individuos y grupos, desde Julio César, pasando por la fuerza alemana Uboat, hasta los actuales usuarios diplomáticos, militares y comerciales, han empleado el cifrado simétrico para la comunicación secreta.

Un esquema de cifrado simétrico tiene cinco ingredientes (Figura 10.2.2.1):

- ✓ Texto nativo (plaintext): es el mensaje original o datos que se proporcionan como entrada del algoritmo.
- ✓ Algoritmo de cifrado: el algoritmo de cifrado lleva a cabo varias sustituciones y transformaciones sobre el texto nativo.
- ✓ Clave secreta: la clave secreta es también una entrada del algoritmo de cifrado. Las sustituciones y transformaciones concretas realizadas por el algoritmo dependen de la clave.
- ✓ Texto cifrado (ciphertext): es el mensaje alterado que se produce como salida. Depende del texto nativo y de la clave secreta. Para un mensaje dado, dos claves diferentes producen dos textos cifrados diferentes.
- ✓ Algoritmo de descifrado: es esencialmente el algoritmo de cifrado ejecutado a la inversa. Toma como entradas el texto cifrado y la clave secreta y produce como salida el texto nativo original.

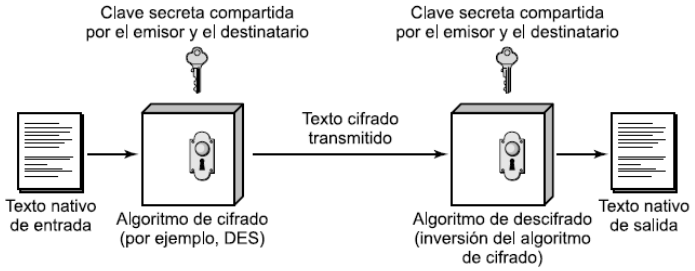

Figura 10.2.2.1: Esquema de un cifrado simétrico

Existen dos requisitos para la utilización segura del cifrado simétrico:

1. Se necesita un algoritmo de cifrado robusto. Como mínimo, es de desear que el algoritmo cumpla que, aunque un oponente conozca el algoritmo y tenga acceso a uno o más textos cifrados, sea incapaz de descifrar el texto o averiguar la clave. Este requisito se suele enunciar de una forma más estricta: el oponente debería ser incapaz de descifrar el texto o descubrir la clave incluso si él o ella poseyera varios textos cifrados junto a sus correspondientes textos nativos.
2. El emisor y el receptor tienen que haber obtenido las copias de la clave secreta de una forma segura y deben mantenerla en secreto. Si alguien puede descubrir la clave y conoce el algoritmo, toda comunicación que utilice esta clave puede ser leída.

Existen dos enfoques generales para atacar el esquema de cifrado simétrico. El primer ataque se conoce como criptoanálisis. Los ataques de criptoanálisis se basan en la naturaleza del algoritmo junto a algún posible conocimiento de las características generales del texto nativo o incluso de algunos pares de texto nativo y cifrado. Este tipo de ataque explota las características del algoritmo para intentar deducir un texto nativo concreto o deducir la clave que se esté utilizando. Si el ataque tiene éxito en la deducción de la clave, el efecto es catastrófico: todos los mensajes cifrados con esa clave, pasados y futuros, están comprometidos.

El segundo método, conocido como ataque por fuerza bruta, consiste en probar cada posible clave sobre un fragmento de texto cifrado hasta que se obtenga una traducción inteligible de texto nativo. La Tabla 10.2.2.2 muestra la cantidad de tiempo que necesita este ataque frente a varias longitudes de clave. La tabla muestra los resultados para cada tamaño de clave, suponiendo que se tarda 1 μs en llevar a cabo un único descifrado, un orden de magnitud razonable para los computadores actuales. Con el uso de una masiva organización paralela de microprocesadores sería posible alcanzar tasas de procesamiento de varios órdenes de magnitud superiores. La última columna de la tabla considera los resultados de un sistema que pudiera procesar 1 millón de claves por microsegundo. Como se puede ver, a este nivel de rendimiento ya no se puede considerar segura en términos computacionales una clave de 56 bits.

Tabla 10.2.2.2: Tiempo promedio para descifrar claves de distinto tamaño

Tamaño de la clave (bits)	Número de claves alternativas	Tiempo necesario a 1 cifrado/μs	Tiempo necesario a 10^6 cifrados/μs
32	$2^{32} = 4{,}3 \times 10^9$	2^{31} μs = 35,8 minutos	2,15 milisegundos
56	$2^{56} = 7{,}2 \times 10^{16}$	2^{55} μs = 1.142 años	10,01 horas
128	$2^{128} = 3{,}4 \times 10^{38}$	2^{127} μs = $5{,}4 \times 10^{24}$ años	$5{,}4 \times 10^{18}$ años
168	$2^{168} = 3{,}7 \times 10^{50}$	2^{167} μs = $5{,}9 \times 10^{36}$ años	$5{,}9 \times 10^{30}$ años

Algoritmo DES (Data Encryption Standard)

DES ha constituido el algoritmo de cifrado dominante desde su introducción en 1977. Sin embargo, dado que DES emplea sólo una clave de 56 bits, fue sólo una cuestión de tiempo que la velocidad de procesamiento computacional dejara a DES

obsoleto. En 1998, la Fundación de la Frontera Electrónica (EFF, Electronic Frontier Foundation) anunció que había roto un reto de DES empleando una máquina de propósito específico para forzar DES, construida por menos de 250.000 dólares. El ataque duró menos de tres días. La EFF ha publicado una descripción detallada de la máquina, permitiendo que otros construyan su propio saboteador. Por supuesto, los precios del hardware continuarán cayendo al mismo tiempo que la velocidad de procesamiento irá aumentando, haciendo de DES un algoritmo inútil.

La vida de DES fue prolongada gracias al uso de triple DES (3DES), que supone la repetición del algoritmo básico DES tres veces, utilizando dos o tres claves únicas, para una longitud de clave de 112 o 168 bits.

El principal inconveniente de 3DES consiste en que el algoritmo por software es relativamente lento. Un inconveniente secundario es que tanto DES como 3DES utilizan un tamaño de bloque de 64 bits. Por razones de eficiencia y de seguridad, es deseable emplear bloques de mayor longitud.

Estándar de cifrado avanzado (AES, Advanced Encryption Standard)

A causa de las mencionadas deficiencias, 3DES no constituye un candidato razonable para su empleo a largo plazo. Como sustituto, el Instituto Nacional de Estándares y Tecnología (NIST, National Institute of Standards and Technology) publicó en 1997 una convocatoria de propuestas para un nuevo estándar de cifrado avanzado (AES, Advanced Encryption Standard), que debería poseer una robustez de seguridad igual o superior a la de 3DES y mejorar significativamente su eficiencia. Además de esos requisitos generales, NIST especificó que AES tenía que ser un *"cifrador"* de bloque simétrico con una longitud de bloque de 128 bits y admitir longitudes de claves de 128, 192 y 256 bits. Los criterios de evaluación incluían seguridad, eficiencia computacional, requerimientos de memoria, disponibilidad de software y hardware y flexibilidad. En 2001 se publicó AES como un estándar federal de procesamiento de información (FIPS 197, Federal Information Processing Standard). En la descripción de esta sección se supone una longitud de clave de 128 bits.

La Figura 10.2.2.3 muestra la estructura global de AES. La entrada de los algoritmos de cifrado y descifrado es un bloque de 128 bits. En FIPS 197, este bloque se representa mediante una matriz cuadrada de bytes. Este bloque se copia en el vector estado, que se modifica en cada etapa del cifrado o descifrado. Tras la etapa final, se copia el contenido de estado en una matriz de salida. De forma similar, la clave de 128 bits se representa como una matriz cuadrada de bytes. Esta clave se expande en un vector de palabras de planificación de clave. Cada palabra consta de cuatro bytes y la planificación total de la clave ocupa 44 palabras para una clave de 128 bits. Dentro de la matriz, los bytes se ordenan por columnas. Así, por ejemplo, los primeros cuatro bytes de un texto nativo de 128 bits de entrada al cifrador ocupan la primera columna de la matriz dentro, los siguientes cuatro bytes ocupan la segunda columna y así sucesivamente. De forma similar, los primeros cuatro bytes de la clave expandida que forman una palabra ocupan la primera columna de la matriz w.

Los comentarios siguientes pueden aclarar el funcionamiento de AES:

1. La clave que se proporciona como entrada se expande en un vector de cuarenta y cuatro palabras de 32 bits, w[i]. Cuatro palabras distintas (128 bits) sirven como clave de ronda para cada vuelta.

2. Se emplean cuatro etapas diferentes: una de permutación y tres de sustitución. Consisten en:

   ✓ Sustituir bytes: en esta etapa se utiliza una tabla, llamada caja-S1 (S-box) para efectuar una sustitución del bloque byte a byte.

   ✓ Desplazar filas: esta etapa efectúa una permutación simple fila a fila.

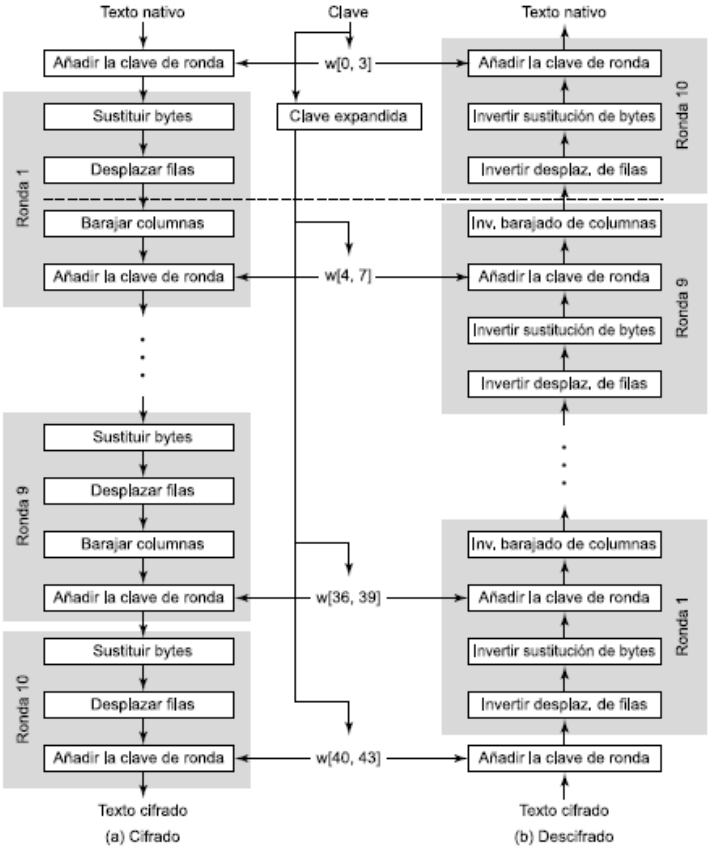

Figura 10.2.2.3: Cifrado y descifrado AES

   ✓ Barajar columnas: se lleva a cabo una sustitución que modifica cada byte de una columna como función de todos los bytes de la misma.

   ✓ Añadir clave de ronda: se efectúa en esta etapa una operación XOR binaria del bloque actual con una porción de la clave expandida.

3. La estructura es bastante sencilla. Para el cifrado y descifrado, el cifrador comienza con una etapa de adición de la clave de ronda, seguida de nueve rondas que incluyen cada una cuatro etapas, seguidas por una décima ronda de tres etapas.

4. Sólo la etapa de adición de la clave de ronda hace uso de la clave. Por esta razón, el cifrado comienza y termina con una etapa de adición de la clave de ronda. Cualquier otra etapa, aplicada al principio o al final, es reversible sin necesidad de conocer la clave y por eso no añadiría seguridad.

5. La etapa de adición de la clave de ronda no sería excepcional en sí misma. Las otras tres etapas juntas desordenan los bits, pero no proporcionarían seguridad por ellos mismos, ya que no utilizan la clave. Podemos ver el cifrado como una sucesión de operaciones alternas de cifrado XOR de un bloque (adición de la clave de ronda), seguido por el barajado del bloque (las otras tres etapas), seguido de cifrado XOR y así sucesivamente. Este esquema es eficiente y muy seguro.

6. Cada etapa es fácilmente reversible. Para las etapas de sustitución de bytes, desplazamiento de filas y adición de la clave de ronda se utiliza una función inversa en el algoritmo de descifrado. Para la etapa de adición de la clave de ronda, la inversión se obtiene mediante la operación XOR de la misma clave de ronda sobre el bloque, debido a que A + A + B = B.

7. Como la mayoría de los cifradores de bloque, el algoritmo de descifrado hace uso de la clave expandida en orden inverso. Sin embargo, el algoritmo de descifrado no es idéntico al de cifrado. Esto es consecuencia de la particular estructura de AES.

8. La ronda final del cifrado y el descifrado consta sólo de tres etapas. De nuevo, es consecuencia de la particular estructura de AES, siendo necesario para que el cifrado sea reversible.

Distribución de Claves

Para que funcione el cifrado simétrico, las dos partes que realizarán un intercambio seguro de datos deben tener la misma clave y ésta debe protegerse para que no sea accesible por otros. Es más, es normalmente deseable realizar cambios frecuentes de la clave para limitar la cantidad de datos comprometidos si un atacante averiguara la clave. Por tanto, la fortaleza de cualquier sistema de cifrado reside en la técnica de distribución de claves empleada, un término que se refiere a los medios para distribuir una clave a las dos partes que deseen intercambiar datos, impidiendo que otros la vean. La distribución de claves se puede lograr de varias formas. Para dos partes A y B:

1. A podría seleccionar una clave y entregársela físicamente a B.

2. Una tercera parte podría seleccionar la clave y entregársela físicamente a B y A.

3. Si A y B han utilizado previa y recientemente una clave, una de las partes podría transmitir a la otra la nueva clave cifrada con la antigua clave.

4. Si A y B tienen cada uno una conexión cifrada con una tercera parte C, C podría entregar a B y A una clave a través de los enlaces cifrados.

Las opciones 1 y 2 exigen una entrega manual de la clave. Éste es un requisito razonable para el cifrado de enlace, ya que cada dispositivo de cifrado de enlace va sólo a intercambiar datos con su pareja del otro extremo de enlace. Sin embargo, para el

cifrado extremo a extremo, la entrega manual es complicada. En un sistema distribuido, cualquier terminal o computador dado puede necesitar efectuar intercambios con muchos otros terminales o computadores a lo largo del tiempo. Así, cada dispositivo necesita varias claves proporcionadas dinámicamente. El problema es especialmente difícil en sistemas distribuidos de área extensa.

La opción 3 constituye una posibilidad válida tanto para el cifrado de enlace como para el cifrado extremo a extremo, pero si un atacante tiene éxito consiguiendo una clave, entonces todas las claves posteriores son reveladas. Incluso si se cambian frecuentemente las claves de cifrado de enlace, dichos cambios deben realizarse manualmente. Por tanto, se prefiere la opción 4 para el cifrado extremo a extremo.

La Figura 10.2.2.4 muestra una implementación de la opción 4 para el cifrado extremo a extremo. En la figura se ha ignorado el cifrado de enlace, que se puede incorporar o no según se requiera. En este esquema se identifican dos clases de claves:

1. Clave de sesión: cuando dos sistemas finales (computadores, terminales, etc.) desean comunicarse, establecen una conexión lógica (por ejemplo: circuitos virtuales). Durante la duración de la conexión lógica, todos los datos de usuario se cifran con una clave de sesión de un solo uso. Al terminar la sesión o la conexión, la clave de sesión se destruye.
2. Clave permanente: es una clave que se emplea entre entidades para la distribución de claves de sesión.

La configuración consta de los siguientes elementos:

1. Centro de distribución de claves (KDC, Key Distribution Center): el centro de distribución de claves determina a qué sistemas se les permite comunicarse entre ellos. Cuando a dos sistemas se les concede el permiso para establecer una conexión, el centro de distribución de claves proporciona una clave de sesión para esa conexión.
2. Módulo de servicio de seguridad (SSM, Security Service Module): este módulo, que puede componerse de funciones de una capa de protocolo, lleva a cabo el cifrado extremo a extremo y obtiene las claves de sesión en nombre de los usuarios.

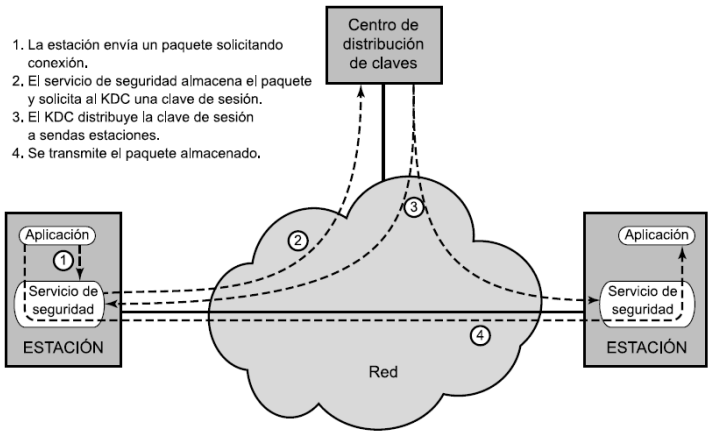

Figura 10.2.2.4: Distribución automática de claves en protocolos orientados a conexión

Los pasos necesarios para establecer una conexión se muestran en la figura. Cuando una estación desea establecer una conexión con otra estación, transmite un paquete de solicitud de conexión (paso 1). El SSM guarda ese paquete y solicita al KDC permiso para establecer la conexión (paso 2). La comunicación entre el SSM y el KDC está cifrada utilizando una clave maestra compartida sólo por este SSM y el KDC. Si el KDC aprueba la solicitud de conexión, genera una clave de sesión y se la entrega a los dos SSM correspondientes, utilizando una clave permanente única para cada SSM (paso 3). El SSM solicitante puede ahora liberar el paquete de solicitud de conexión y se establece así una conexión entre los dos sistemas finales (paso 4). Todos los datos de usuario intercambiados entre los dos sistemas finales son cifrados por sus respectivos SSM empleando la clave de sesión de un sólo uso.

La estrategia de la distribución automática de claves proporciona las características de flexibilidad y dinamismo necesarias para permitir a varios usuarios de terminales acceder a distintas estaciones y a las estaciones les permite intercambiar datos con cada una de las otras.

10.2.3. Cifrado asimétrico

En este capítulo se analizan los fundamentos de los algoritmos de clave asimétrica más populares y de aplicación práctica más difundida. Comenzamos con el reconocido algoritmo RSA, útil tanto para la firma digital como para el intercambio de claves, y base del protocolo SSL. Seguimos con el algoritmo de intercambio de claves de Diffie- Hellman utilizado, entre otros, en el protocolo SSH. Otros algoritmos asimétricos existentes como DSA (aplicación práctica de "ElGamal"[1]) y criptografía de curva elíptica están fuera del alcance de las presentes notas.

[1] El procedimiento de cifrado/descifrado ElGamal se refiere a un esquema de cifrado basado en el problema matemático del logaritmo discreto. El algoritmo de ElGamal puede ser utilizado tanto para generar firmas digitales como para cifrar o descifrar.

Algoritmos asimétricos

Los algoritmos de clave pública son muy diferentes a los algoritmos simétricos como AES o DES. La mayoría de los algoritmos de clave pública están basados en funciones teóricas. Esto es diferente a los cifradores simétricos, donde el objetivo no es tener una descripción matemática compacta entre la entrada y la salida. Aunque dentro de los cifradores simétricos se usan estructuras matemáticas para implementar bloques pequeños (por ejemplo, la S-Box en AES), esto no significa que el cifrador en sí mismo constituya una descripción matemática compacta.

Mientras que la criptografía de clave simétrica se basa en sustituciones y permutaciones de símbolos, la criptografía de clave pública se basa en aplicar funciones matemáticas a números. En la criptografía de clave simétrica, el texto plano y el texto cifrado se toman como una combinación de símbolos. El cifrado y descifrado permutan o sustituyen un símbolo por otro.

En la criptografía de clave asimétrica, el texto plano y el texto cifrado son números; el cifrado y el descifrado son funciones matemáticas que se aplican a números para crear otros números. Se describen algunas cuestiones asociadas con los algoritmos simétricos.

- ✓ **Problema de distribución de clave:** La clave entre Alice y Bob se debe establecer utilizando un canal seguro. El canal de comunicación del mensaje, per se, no es seguro; por lo que no es factible el envío de la clave directamente a través del mismo (lo que hubiera sido la manera más conveniente de hacerlo).
- ✓ **Número de claves:** Una vez resuelto el problema de la distribución de las claves, se puede observar el problema de la gran cantidad de claves. Cada par de usuarios necesita un par de claves separadas. En una red de n usuarios tenemos:

$$\frac{n\,x\,(n-1)}{2}$$

- ✓ pares de claves, y cada usuario tiene que almacenar n-1 claves de forma segura. Para una red corporativa de 1000 personas, se necesitan medio millón de pares de claves que deben ser generadas y transportadas sobre canales seguros.

- ✓ **No es posible el no repudio:** Alice y Bob poseen las mismas facultades debido a que tienen la misma clave. Por lo tanto, la criptografía simétrica no nos defiende de que Alice o Bob se hagan trampa entre sí. Un ejemplo típico es el caso donde Alice envía una orden de compra a Bob por una cantidad x de cañas de pescar, luego Alice se arrepiente y puede tranquilamente decir que ella no fue quien emitió esa orden de compra, ya que Bob también pudo haberla generado, dado que él conoce la clave. De esta manera no hay prueba legal con la que se pueda dirimir el conflicto. Por lo tanto, precisamos un mecanismo para identificar por separado a cada uno de los actores comerciales de este caso. La prevención de esta situación se denomina no repudio y se puede conseguir a través de la implementación de la firma digital usando algoritmos de clave asimétrica.

Con el fin de superar estos inconvenientes, Diffie, Hellman y Merkle tuvieron una propuesta revolucionaria basada en la siguiente idea: no es necesario que la clave que tiene la persona que cifra el mensaje sea secreta. La parte crucial es que solo el

receptor puede descifrar el mensaje con una clave secreta. Para implementar este sistema, una de las partes, Bob, publica una clave pública de cifrado que es conocida por todo el mundo. A su vez retiene una clave secreta que se complementa con la que hizo pública, y la utilizará para el descifrado. Por lo tanto, la clave k de Bob consiste de dos partes, una pública, k_{pub}, y una privada, k_{pr}. En la Figura 10.2.3.1 se muestra un protocolo básico de cifrado de clave pública.

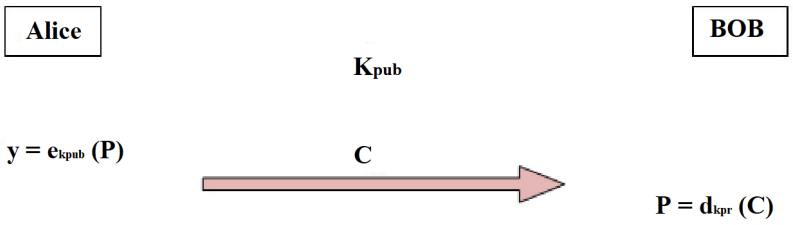

Figura 10.2.3.1: Proceso de comunicación en algoritmo asimétrico

En la figura 10.2.3.2 se muestra cómo usar un algoritmo asimétrico para transportar una clave simétrica de AES. Lo que se hace es cifrar la clave simétrica de AES con algún algoritmo de clave pública. Una vez que Bob descifró la clave simétrica, ambas partes pueden usarla para cifrar y descifrar aplicando cualquier algoritmo simétrico (en este caso AES). La principal ventaja con el esquema de la Figura 10.2.3.1 es que los datos se cifran con un algoritmo de cifrado simétrico, lo cual es mucho más rápido.

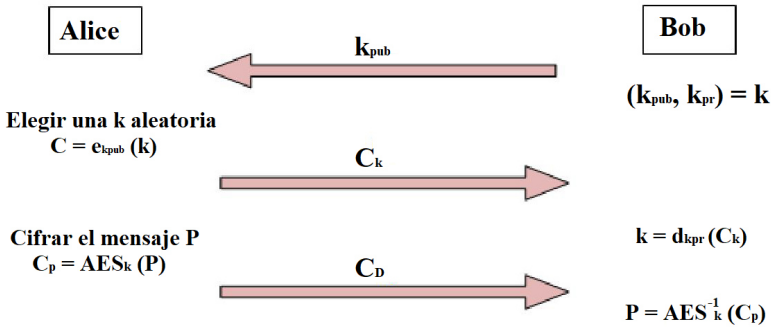

Figura 10.3.2.2: Algoritmo asimétrico para intercambiar claves

Servicios de Seguridad inherentes a la clave pública

Los principales servicios de seguridad que la criptografía de clave pública puede brindar son:

- ✓ Establecimiento de claves. Hay protocolos para establecer claves secretas sobre un canal inseguro. Por ejemplo, el protocolo "Diffie–Hellman key Exchange" (DHKE) o protocolos de transporte de clave basados en RSA.

- ✓ No repudio. Con los algoritmos de firma digital se puede proveer no repudio e integridad de mensaje (RSA, DSA)
- ✓ Identificación. Se pueden identificar usuarios usando protocolos de desafío y respuesta con firma digital.
- ✓ Cifrado. Se pueden cifrar mensajes con algoritmos como RSA o Elgamal

Por un lado, los algoritmos de clave simétrica pueden manejar el cifrado/descifrado, pero se les complica con el manejo de la administración de las claves. Por otro lado, parecería que los algoritmos de clave pública pueden proveer todas las funcionalidades necesarias de seguridad. El problema en este último tipo de algoritmo, es que el cifrado/descifrado es computacionalmente demasiado intenso (100 a 1000 veces más lentos que los algoritmos simétricos) por lo cual no se los utiliza para esta función. Asimismo, los algoritmos de clave privada no son buenos para el manejo de claves y para brindar el servicio de no repudio. ¿Solución?, usar lo mejor de cada mundo, la mayoría de los protocolos reales son protocolos híbridos que incorporan tanto algoritmos simétricos como asimétricos. Un ejemplo de esto es el protocolo SSL/TLS que se usa para conexiones Web seguras.

Una cuestión que queda pendiente con los algoritmos asimétricos es la autenticidad de las claves públicas que son distribuidas libremente. ¿Cómo sabemos que realmente pertenecen a la persona que dice ser y no estamos siendo engañados por un tercero?

Básicamente el problema se resuelve con certificados, que asocian una clave pública a una identidad y con una Autoridad de Certificación (CA) en la que uno confía, y que es la emisora de dichos certificados.

Principales Algoritmos de clave pública

Hay tres grandes familias de algoritmos de clave pública que son de relevancia práctica. Se pueden clasificar en base a su problema computacional subyacente.

- ✓ Esquemas de Factorización de Enteros. Se basan en la dificultad de factorizar números enteros de gran tamaño. El ejemplo clásico es RSA.
- ✓ Esquemas de Logaritmo Discreto. Se basan en el problema del logaritmo discreto en campos finitos. Ejemplos: Intercambio de claves de Diffie-Hellman (DHKE), cifrado Elgamal y el Algoritmo de Firma Digital (DSA = Digital Signature Algorithm).
- ✓ Esquemas de Curva Elíptica (EC - Elliptic Curve). Es una generalización del algoritmo de logaritmos discretos. Ejemplos: Intercambio de claves con implementación de Diffie–Hellman en curvas elípticas (ECDH) e implementación de DSA en curvas elípticas (ECDSA).

Tamaño de las claves y nivel de seguridad

Los algoritmos de clave pública requieren operandos y claves de gran tamaño. Cuanto más grande son éstos más seguros se vuelven los algoritmos. Un parámetro

de comparación entre los distintos algoritmos es el nivel de seguridad. Se dice que un algoritmo tiene un nivel de seguridad de n bits si el ataque requiere 2^n pasos. Para los algoritmos de clave simétrica esta relación es natural, dado que con n bits tenemos un espacio de claves de 2^n. Con los algoritmos de clave pública la relación no es directa y generalmente se precisan muchos más bits para lograr el mismo nivel de seguridad que en un algoritmo de clave privada. En la tabla 10.3.2.3 se muestran la cantidad de bits necesarios para cada algoritmo a fin de alcanzar el nivel de seguridad deseado. Dicha tabla se debe leer de la siguiente forma: por ejemplo, para alcanzar un nivel de seguridad de 128 bits, si se usa una clave simétrica (AES o 3DES), su tamaño debe ser de 128 bits, pero si se usa clave asimétrica (RSA) el tamaño de la clave para obtener el mismo nivel de seguridad debe ser de 3072 bits.

Los ataques de fuerza bruta se contrarrestan, como siempre, eligiendo el nivel de seguridad adecuado (cantidad de bits de la clave). Otra forma de ataque es averiguar alguna manera de calcular la clave privada de la clave pública. Todos los algoritmos son susceptibles a este tipo de ataque. En la historia del criptoanálisis se ha visto numerosas veces que lo que parecía irresoluble desde una perspectiva, se transformaba en una cuestión sencilla cambiando la forma de abordar el problema.

Tabla 10.3.2.3: Tamaño de la clave y nivel de seguridad

Algoritmo	Criptosistema	Nivel de Seguridad (bits)			
		80	128	192	256
Factorización de enteros	RSA	1024 bit	3072 bit	7680 bit	15360 bit
Logaritmo discreto	DH, DSA, Elgamal	1024 bit	3072 bit	7680 bit	15360 bit
Curvas Elípticas	ECDH, ECDSA	160 bit	256 bit	384 bit	512 bit
Clave simétrica	AES, 3DES	80 bit	128 bit	192 bit	256 bit

RSA – Rivest, Shamir y Adleman

Se puede decir que RSA es el algoritmo de clave asimétrica que más se utiliza. Mayormente se lo emplea para:

- ✓ Cifrado de pequeñas cantidades de datos, por ejemplo, claves.
- ✓ Firmas digitales.

Seguramente el algoritmo RSA no está pensado para reemplazar a los algoritmos simétricos dada su lentitud de cómputo y el aumento en el tamaño del mensaje cifrado. Su utilidad radica en lograr un intercambio de claves seguro para que luego algoritmos como AES puedan llevar a cabo el pesado trabajo de cifrado de manera más eficiente. La función de una sola vía en la que se basa RSA es el problema de factorización de números enteros. Multiplicar dos números primos grandes es computacionalmente fácil, no así la operación inversa de factorización. Para comprender su funcionamiento es necesario tener en claro el teorema de Euler y la función phi de Euler dado que juegan un papel muy importante.

Procedimiento para cifrar y descifrar

El cifrado y descifrado se realiza en el anillo entero Z_n. Para una explicación detallada puede consultarse (Stallings, Cryptography and Network Security Principles and Practice, Seventh Edition, 2017). RSA cifra el texto plano P que es un elemento en $Z_n = \{0, 1, \ldots, n-1\}$. Por lo tanto, el valor binario de P tiene que ser menor que n. Lo mismo sucede con el descifrado.

Cifrado: Dada la clave pública $(n,e) = k_{pub}$ y el texto plano P, la función de cifrado es:

$$C = ek_{pub}(P) \equiv P^e \bmod n \text{ donde P y C } \epsilon\ Z_n$$

Descifrado: Dada la clave privada $d = k_{pr}$ y el texto cifrado C, la función de descifrado es:

$$P = dk_{pr}(C) \equiv C^d \bmod n \text{ donde P y C } \epsilon\ Z_n$$

- ✓ P, C y n son números grandes ≥ 1024 bits.
- ✓ e: exponente de cifrado o exponente público.
- ✓ d: exponente de descifrado o exponente privado.

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

Algunos requerimientos del criptosistema RSA:

- ✓ Dados e y n debe ser inviable obtener d.
- ✓ No se pueden cifrar más de n bits. En caso que sea necesario debe hacerse por bloques.
- ✓ Se necesita un método de exponenciación rápida de manera de poder obtener fácilmente tanto Pe mod n, como Cd mod n (algoritmo de elevar al cuadrado y multiplicar).
- ✓ Para un n dado debe haber muchos pares de claves públicas/privadas, sino sería viable un ataque de fuerza bruta.

Generación de claves RSA

1. Elegir dos primos grandes tal que $p \neq q$.
2. $n = p.q$
3. $\Phi(n) = (p-1).(q-1)$
4. Elegir el exponente público e tal que $1 < e < \Phi(n)$ y mcd[2] $(e, \Phi(n)) = 1$ (esto implica que son números coprimos, el máximo común divisor de ambos es el 1), esto implica que el número e es primo con respecto a $\Phi(n)$.

[2] Máximo común divisor

5. Calcular el exponente privado $d = e^{-1} \bmod \Phi(n)$. (d es el inverso de e mod $\Phi(n)$)
6. El par de números (e, n) son la clave pública
7. El par de números (d, n) son la clave privada.
8. Cifrado: La función de cifrado es $C = M^e \bmod n$
9. Descifrado: La función de descifrado es $M = C^d \bmod n$
    - La condición que el mcd (e, $\Phi(n)$) = 1 asegura que existe el inverso multiplicativo de e módulo n y por lo tanto hay una clave privada d.
    - Los números primos del paso 1 se calculan de acuerdo a lo visto en esta misma sección
    - El cálculo de d y e se puede hacer al mismo tiempo usando el algoritmo extendido de Euclides.
    - El tamaño recomendado de cada primo p y q es de 512 bits (aprox. 152 dígitos decimales); por lo que el tamaño del módulo n es de 1024 bits (309 dígitos).
    - Los valores p, q y $\Phi(n)$ podrían ser descartados luego de los cálculos.

Un ejemplo utilizando número pequeños:

1. Se eligen dos números primos, por ejemplo, p=3 y q=11.
2. n = 3 * 11 = 33
3. $\phi(n) = (3-1) * (11-1) = 20$
4. Se calcula e (impar), tiene que ser coprimo con 20. El valor encontrado para e=3. Esto es porque entre 20 y 3 el MCD es el número 1, por lo tanto 20 y 3 son coprimos.
5. Se calcula d como el "inverso multiplicativo" módulo $\phi(n)$ de e (se escribe d = inv (e, $\phi(n)$). Para este caso, la variable Y va a ir tomando valores 1,2,3,…n, hasta que se obtenga un valor entero para la expresión: $d = ((Y * \phi(n)) + 1) / e$

    Probamos con Y = 1, $(1 * 20 + 1) / 3 = 22 / 3 = 7{,}33$

    Probamos con Y = 7, $(7 * 20 + 1) / 3 = 47$, es un número entero

    De esta forma d = 7
6. e=3 y n=33 son la clave pública
7. d=7 y n=33 son la clave privada
8. Cifrado: Mensaje = 5, $C = M^e \bmod n = 5^3 \bmod 33 = 26$
9. Descifrado: $M = C^d \bmod n = 26^7 \bmod 33 = 8031810176 \bmod 33 = 5$

DH – Diffie y Hellman

Una de las formas de intercambiar una clave simétrica es a través de un centro de distribución de claves (KDC = Key Distribution Center). Sin embargo, existe una alternativa para que Alice y Bob puedan intercambiar esa clave sin recurrir a un KDC.

El algoritmo de Diffie-Hellman (en honor a sus creadores, Whitfield Diffie y Martin Hellman) permite acordar una clave secreta entre dos máquinas, a través de un canal inseguro y enviando únicamente dos mensajes. La clave secreta resultante no puede ser descubierta por un atacante, aunque éste obtenga los dos mensajes enviados por el protocolo. La principal aplicación de este protocolo es acordar una clave simétrica con la que posteriormente cifrar las comunicaciones entre dos máquinas.

El protocolo de Diffie-Hellman fue publicado en 1976. Actualmente se conoce que es vulnerable a ataques de hombre en medio (Man in Middle[3]): un atacante podría situarse entre ambas máquinas y acordar una clave simétrica con cada una de las partes, haciéndose pasar por el host A de cara al host B y viceversa. Una vez establecidas las 2 claves simétricas, el atacante haría de puente entre los 2 hosts, descifrando toda la comunicación y volviéndola a cifrar para enviársela al otro host.

El funcionamiento del protocolo se explica en la siguiente Figura 10.3.2.4:

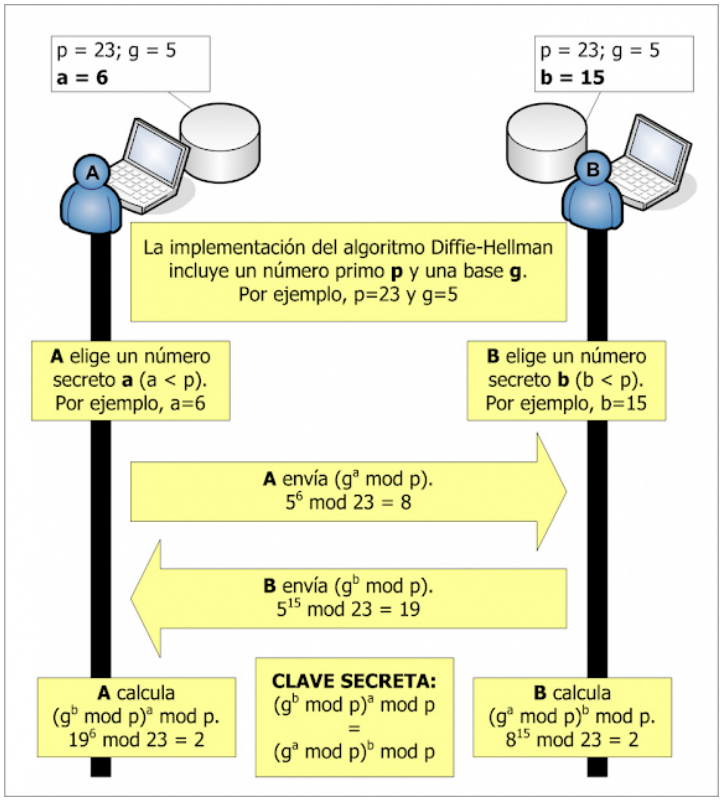

Figura 10.3.2.4: Algoritmo Diffie Hellman

Los valores de "p" y "g" son públicos y cualquier atacante puede conocerlos, pero esto no supone una vulnerabilidad. Aunque un atacante conociese dichos valores y capturara los dos mensajes enviados entre las máquinas A y B, no sería capaz de averiguar la clave secreta.

Las implementaciones actuales del protocolo Diffie-Hellman utilizan números primos muy grandes, lo que impide a un atacante calcular los valores de "a" y

[3] En criptografía, un ataque de intermediario1 (en inglés, man-in-the-middle attack, MitM o Janus) es un ataque en el que se adquiere la capacidad de leer, insertar y modificar a voluntad.

"b". El valor "g" no necesita ser grande, y en la práctica su valor es 2 ó 5. En el RFC 3526 aparecen publicados los números primos que deben utilizarse. A modo de ejemplo, se facilita aquí el número primo de 2048 bytes propuesto. El valor "g" utilizado es 2:

$$p = 2^{2048} – 2^{1984} – 1 + 2^{64} \text{ x } ((2^{1918} \text{ pi}) + 124476)$$

10.2.4. Función Hash

Una función hash H acepta un bloque de datos de longitud variable M como entrada y produce un valor hash de tamaño fijo h = H(M). Una "buena" función hash tiene la propiedad de que el resultado de la aplicación de la función a un gran conjunto de entradas producirá salidas que están distribuidos uniformemente y aparentemente aleatorios. En términos generales, el objetivo principal de una función hash es la integridad de los datos. Un cambio en cualquier bit o bits de M tiene como resultado, con alta probabilidad, un cambio en el valor hash.

El tipo de función hash necesario para las aplicaciones de seguridad se denomina función hash criptográfica. Una función hash criptográfica es un algoritmo para el que es computacionalmente inviable (porque ningún ataque es significativamente más eficiente que el de fuerza bruta) encontrar:

a. Un objeto de datos que se corresponda con un resultado hash preestablecido (la propiedad unidireccional)

b. Dos objetos de datos que correspondan al mismo resultado hash (la propiedad de no colisión).

Debido a estas características, las funciones hash se utilizan a menudo para determinar si los datos han cambiado o no.

La figura 10.2.4.1 muestra el funcionamiento general de una función hash criptográfica. Normalmente, la entrada se rellena con un múltiplo entero de una longitud fija (por ejemplo, 1024 bits), y el relleno incluye el valor de aplicar una función hash al mensaje original. Este valor representa una medida de seguridad que tiene como objetivo aumentar la dificultad de un atacante para producir un mensaje alternativo (fraudulento), con el mismo valor hash.

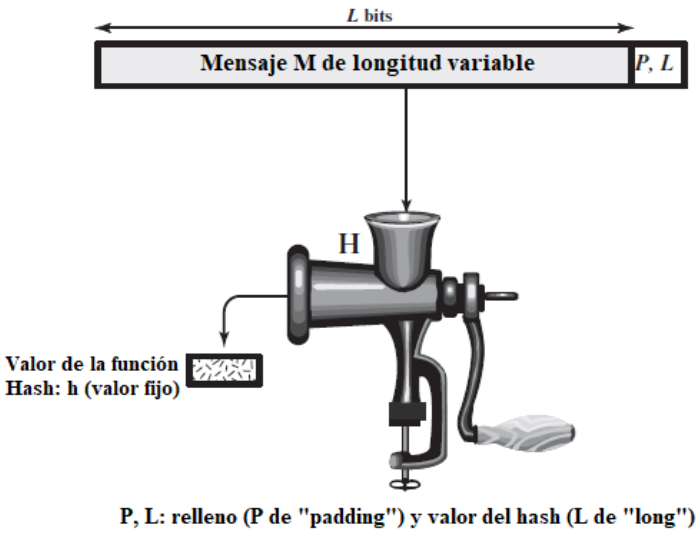

Figura 10.2.4.1: Aplicación de la función "hash" a un mensaje

Características de una función hash segura (función de dispersión)

El objetivo de una función de dispersión es producir una *"huella dactilar"* de un fichero, mensaje, u otro bloque de datos. Para que sea útil para la autenticación, una función de dispersión H debe cumplir las propiedades siguientes:

1. H puede ser aplicada a un bloque de datos de cualquier tamaño.
2. H produce una salida de longitud fija.
3. Para cualquier x dado, es relativamente fácil calcular H(x), haciendo factibles las implementaciones por hardware y software.
4. Para cualquier código h, por limitaciones computacionales, es inviable encontrar un x tal que H(x) = h.
5. Para cualquier bloque x, por limitaciones computacionales, es inviable encontrar un $y \neq x$ para el que H(y) = H(x).
6. Por limitaciones computacionales, es impracticable encontrar una pareja (x, y) tal que H(x) = H(y).

Las tres primeras propiedades son requisitos para la aplicación práctica de una función de dispersión en la autenticación de mensajes.

La cuarta propiedad es la propiedad de *"un solo sentido"*: es fácil generar un código dado un mensaje, pero virtualmente imposible generar un mensaje a partir de un código. Esta propiedad es importante si la técnica de autenticación supone el uso de un valor secreto. El valor secreto no se envía. Sin embargo, si la función de dispersión no es de un solo sentido, un atacante puede descubrir fácilmente el valor secreto. Si el atacante puede observar o interceptar una transmisión, obtendrá el mensaje M y el código de dispersión $MD_M = H(S_{AB} \| M)$. El atacante entonces invierte la función de dispersión para

obtener $S_{AB} \| M = H^{-1}(MD_M)$. Dado que el atacante tiene ahora M y $[S_{AB} \| M]$, es una cuestión trivial obtener S_{AB}.

La quinta propiedad garantiza que no se pueda encontrar un mensaje alternativo que produzca el mismo valor de resumen que un mensaje dado. Esto impide la falsificación cuando se utiliza un código de dispersión cifrado Si no se cumpliese esta propiedad, un atacante sería capaz de realizar la secuencia siguiente: en primer lugar, observar o interceptar un mensaje más su código de dispersión cifrado; en segundo lugar, generar un código de dispersión no cifrado a partir del mensaje; por último, generar un mensaje alternativo con el mismo código de dispersión.

A una función de dispersión que satisfaga las cinco primeras propiedades de la lista anterior se la conoce como función de dispersión débil. Si satisface también la sexta propiedad, entonces se la conoce como función de dispersión robusta. La sexta propiedad protege contra una clase de ataque sofisticado conocida como ataque del cumpleaños[4].

Además de proporcionar autenticación, un resumen de mensaje proporciona también integridad de los datos, ya que realiza la misma función que una secuencia de comprobación de trama: si algún bit es modificado accidentalmente en el tránsito, el resumen del mensaje será erróneo.

Usos

Quizá el algoritmo criptográfico más versátil sea la función hash criptográfica. Se utiliza en una gran variedad de aplicaciones de seguridad y protocolos de Internet. Para entender mejor algunos de los requisitos e implicaciones de seguridad de las funciones hash criptográficas, es útil examinar la gama de aplicaciones en las que se emplea.

Autenticación de mensajes

La autenticación de mensajes es un mecanismo o servicio utilizado para verificar la integridad de un mensaje. La autenticación de mensajes garantiza que los datos recibidos son exactamente los mismos que se enviaron (es decir, que no hay modificaciones, inserciones, eliminaciones o repeticiones). En muchos casos, es necesario que el mecanismo de autenticación garantice que la supuesta identidad del remitente es válida. Cuando se utiliza una función hash para proporcionar autenticación de mensajes, el valor de la función hash suele denominarse *"message digest"*.

La esencia del uso de una función hash para la integridad de los mensajes es la siguiente:

El emisor calcula un valor hash en función de los bits del mensaje y transmite tanto el valor hash como el mensaje. El receptor realiza el mismo cálculo hash en los bits del mensaje y compara este valor con el valor hash entrante. Si hay un desajuste,

[4] Ataque de fuerza bruta que busca colisiones probando todas las combinaciones posibles de 2 textos.

el receptor sabe que el mensaje (o posiblemente el valor hash) ha sido alterado (Figura 10.2.4.2a).

El valor hash debe transmitirse de forma segura. Es decir, el valor hash debe estar protegido de manera que, si un adversario altera o sustituye el mensaje, no pueda también alterar el valor hash para engañar al receptor. Este tipo de ataque se muestra en la Figura 10.2.4.2b. En este ejemplo, Alice transmite un bloque de datos y adjunta un valor hash. Darth intercepta el mensaje, altera o sustituye el bloque de datos bloque de datos, y calcula y adjunta un nuevo valor hash. Bob recibe los datos alterados con el nuevo valor hash y no detecta el cambio. Para evitar este ataque, el valor hash generado por Alice debe estar protegido.

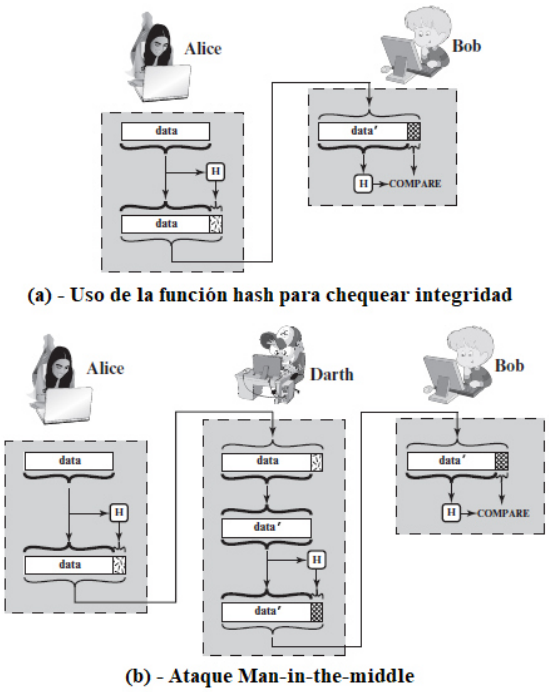

Figura 10.2.4.2: Ataque a la función hash

La figura 10.2.4.3 ilustra una variedad de formas en las que se puede utilizar un código hash para proporcionar autenticación de mensajes, como se indica a continuación.

a. El mensaje más el código hash concatenado se encriptan utilizando una encriptación simétrica. Como sólo A y B comparten la clave secreta, el mensaje debe provenir de A y no haber sido alterado. El código hash proporciona la estructura o redundancia necesaria para lograr la autenticación. Como el cifrado se aplica a todo el mensaje más el código hash, también se proporciona confidencialidad.

b. Sólo se encripta el código hash, utilizando una encriptación simétrica. Esto reduce la carga de procesamiento para aquellas aplicaciones que no requieren confidencialidad.

c. Es posible utilizar una función hash, pero sin encriptación, para la autentificación de los mensajes. La técnica supone que las dos partes que se comunican comparten un valor secreto común "S". A calcula el valor hash sobre la concatenación de M y S y añade el valor hash resultante a M. Como B posee S, puede volver a calcular el valor hash para verificarlo. Como el valor secreto no se envía, un adversario no puede modificar un mensaje interceptado y no puede generar un mensaje falso.

d. La confidencialidad puede añadirse al método (c) encriptando todo el mensaje más el código hash.

Cuando no se requiere confidencialidad, el método (b) tiene una ventaja sobre los métodos (a) y (d), que encriptan todo el mensaje, en el sentido de que se requiere menos computación. No obstante, ha aumentado el interés por las técnicas que evitan el cifrado (Figura 10.2.4.3c). En (Tsudik, 1992) se señalan varias razones para este interés.

- ✓ El software de encriptación es relativamente lento. Aunque la cantidad de datos a cifrar por mensaje sea pequeña, puede haber un flujo constante de mensajes entrando y saliendo de un sistema.
- ✓ Los costes del hardware de encriptación no son insignificantes. Existen implementaciones de chip de bajo coste de DES de bajo coste, pero el coste aumenta si todos los nodos de una red deben tener esta capacidad.
- ✓ El hardware de encriptación está optimizado para datos de gran tamaño. Para pequeños bloques de datos, una alta proporción del tiempo se gasta en la sobrecarga de inicialización/invocación.
- ✓ Los algoritmos de encriptación pueden estar cubiertos por patentes, y hay un coste asociado de la licencia de uso.

Lo más habitual es que la autenticación de los mensajes se realice mediante un código de autenticación de mensajes (MAC), también conocido como función hash con clave. Normalmente, los MAC se utilizan entre dos partes que comparten una clave secreta para autenticar la información intercambiada entre esas partes. Una función MAC toma como entrada una clave secreta y un bloque de datos y produce un valor hash, denominado MAC, que se asocia con el mensaje protegido. Si es necesario comprobar la integridad del mensaje la función MAC puede aplicarse al mensaje y el resultado puede compararse con el valor MAC asociado. Un atacante que altere el mensaje no podrá alterar el valor MAC asociado sin conocer la clave secreta. Obsérvese que la parte verificadora también sabe quién es la parte remitente porque nadie más conoce la clave secreta. Es necesario destacar que la combinación de hashing y encriptación da como resultado una función global que es, de hecho, una MAC (Figura 10.2.4.3b). Es decir, E (K, H(M)) es una función de un mensaje de longitud variable M y una clave secreta K, y produce una salida de tamaño fijo que es segura contra un oponente que no conoce la clave secreta. En la práctica, se diseñan algoritmos MAC específicos que suelen ser más eficientes que un algoritmo de cifrado.

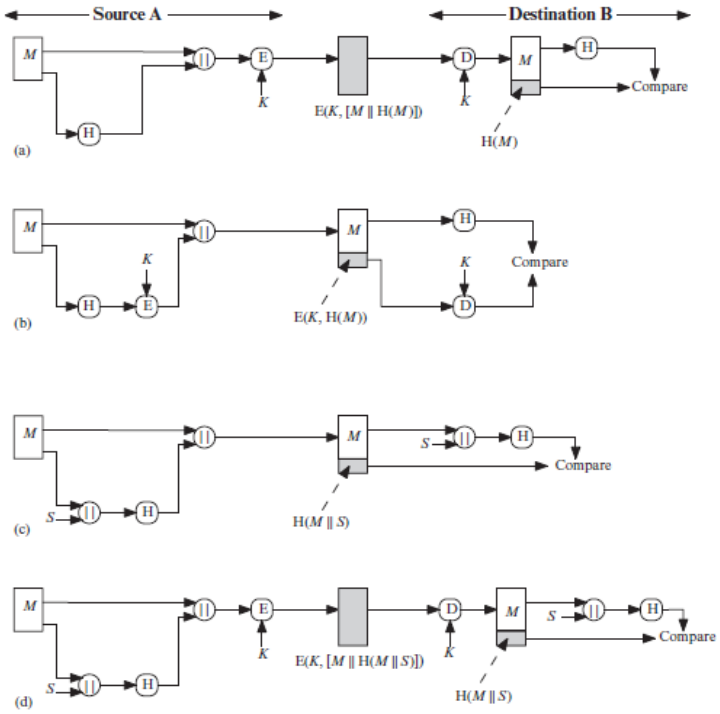

Figura 10.2.4.3: Uso de hash para la autenticación de mensajes

Firma Digital

Otra aplicación importante, que es similar a la de autenticación de mensajes es la firma digital. El funcionamiento de la firma digital es similar al de la MAC. En el caso de la firma digital, el valor hash de un mensaje se encripta con la clave privada del usuario. Cualquiera que conozca la clave pública del usuario puede verificar la integridad del mensaje asociado a la firma digital. En caso, un atacante que desee alterar el mensaje necesitaría conocer la clave privada del usuario.

La figura 10.2.4.3 ilustra, de forma simplificada, cómo se utiliza un código hash para proporcionar una firma digital.

a. El código hash está encriptado, utilizando una encriptación de clave pública con la clave privada del remitente. Como en la figura 10.2.4.3b, esto proporciona autenticación. También proporciona una firma digital, porque sólo el remitente podría haber producido el código hash cifrado. código hash. De hecho, esta es la esencia de la técnica de la firma digital.

b. Si se desea la confidencialidad además de la firma digital, entonces el mensaje más el código hash encriptado con clave privada pueden ser encriptados usando una clave secreta simétrica. Esta es una técnica común.

Otros usos de las funciones Hash

Las funciones Hash se utilizan habitualmente para crear un archivo de contraseñas unidireccional. La implementación consiste en almacenar el hash de una contraseña, en lugar de la contraseña misma. De esta forma, el valor real de la clave no puede ser recuperada por un hacker que pueda acceder al archivo de contraseñas. En términos prácticos, cuando un usuario introduce una contraseña, el hash de esa contraseña se compara con el valor hash almacenado para su verificación. La mayoría de los sistemas operativos utilizan este método de protección de contraseñas.

Las funciones hash pueden utilizarse para la detección de intrusos y de virus. Almacenar H(F) para cada archivo de un sistema y asegurar los valores hash (por ejemplo, en un CD-R). Se puede determinar posteriormente si un archivo ha sido modificado volviendo a calcular H(F). Un intruso necesitaría cambiar F (el archivo) sin cambiar H(F), algo virtualmente imposible.

SHA – Secure Hash Algorithm

El algoritmo de dispersión segura (SHA, Secure Hash Algorithm) fue desarrollado por el NIST y publicado como estándar federal para el procesamiento de la información (FIPS 180) en 1993. En 1995 se publicó una versión revisada denominada FIPS 180-1, conocida generalmente como SHA-1.

El algoritmo toma como entrada un mensaje con una longitud máxima de 264 bits y produce un resumen de mensaje de 160 bits. La entrada se procesa en bloques de 512 bits. La Figura 10.2.4.4 representa el procesamiento general de un mensaje para producir un resumen. El procesamiento consta de los siguientes pasos:

1. Paso 1: Añadir bits de relleno. Se completa el mensaje de forma que su longitud sea congruente con 448 módulo 512 (longitud = 448 mod 512). Es decir, la longitud del mensaje completado es 64 bits menor que un múltiplo de 512 bits. Siempre se incorpora el relleno, incluso si el mensaje tiene ya la longitud deseada. Así, el número de bits de relleno se encuentra en el rango de 1 a 512. El relleno se compone de un único bit con valor 1, seguido por el apropiado número de bits con valor cero.

2. Paso 2: Añadir longitud. Se añade al mensaje un bloque de 64 bits. Este bloque se trata como un entero de 64 bits sin signo (el byte más significativo primero) y contiene la longitud del mensaje original (antes de incorporar el relleno). La inclusión de un valor de longitud hace más difícil un tipo de ataque conocido como el ataque por relleno

   Como resultado de los dos primeros pasos se produce un mensaje cuya longitud es un múltiplo entero de 512 bits. En la Figura 21.8, el mensaje expandido se representa como una secuencia de bloques de 512 bits Y0, Y1, ..., YL.1, de manera que la longitud total del mensaje extendido es de L x 512 bits. De forma equivalente, el resultado es un múltiplo de 16 palabras de 32 bits.

3. Paso 3: Inicializar la memoria temporal de MD. Se utiliza una memoria temporal de 160 bits para almacenar los resultados intermedios y finales de la función de dispersión.

4. Paso 4: Procesar el mensaje en bloques de 512 bits (palabras de 16 bits). El corazón del algoritmo es un módulo que consta de 4 rondas de procesamiento de 20 pasos cada uno. Las cuatro rondas tienen una estructura similar, pero cada una utiliza una función lógica primitiva diferente. Cada ronda toma como entrada el bloque de 512 bits que se esté procesando (Y_q) y el valor de la memoria temporal de 160 bits, actualizando el contenido de la memoria temporal.
5. Paso 5: Producir la salida. Tras procesar los L bloques de 512 bits, la salida de la etapa L-ésima es el resumen de 160 bits del mensaje.

El algoritmo SHA-1 tiene la propiedad de que cada bit del código de dispersión es una función de cada bit de la entrada. El algoritmo produce resultados bien barajados. Es decir, es improbable que dos mensajes seleccionados de forma aleatoria tengan el mismo código de dispersión, incluso aunque manifiesten regularidades similares. A menos que exista alguna debilidad oculta en SHA-1, lo cual no ha sido publicado hasta ahora, la dificultad de que aparezcan dos mensajes con el mismo resumen de mensaje es del orden de 2^{80} operaciones, mientras que la dificultad de encontrar un mensaje con un resumen dado es del orden de 2^{160} operaciones.

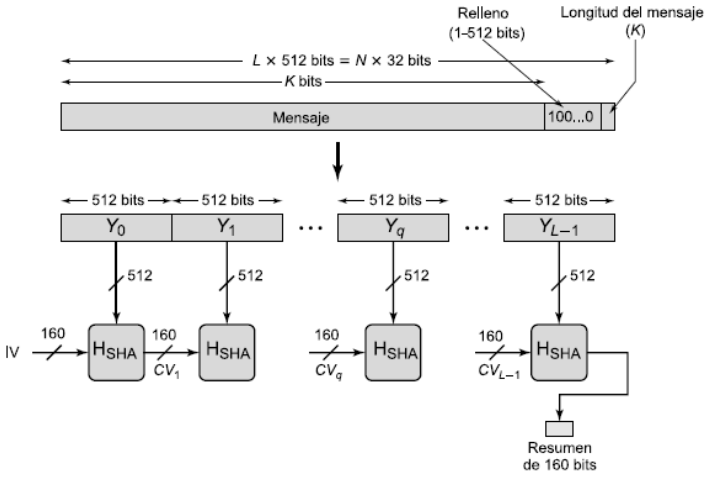

Figura 10.2.4.4: Pasos del algoritmo SHA-1

Evolución de los algoritmos SHA

En 1993 salió a la luz el primer protocolo SHA, también llamado coloquialmente SHA-0. Dos años más tarde, se publicó una variante mejorada más resistente, el SHA-1. Algunos años más tarde se lanzó SHA-2, que tiene cuatro variantes según el número de bits, como son SHA-224, SHA-256, SHA-384 y SHA-512.

Como casi todos los avances en materia de criptografía, los gobiernos del mundo han tenido un papel fundamental debido a las guerras. El algoritmo SHA o Secure Hash Algorithm (Algoritmo de Hash Seguro), ha sido uno de esos avances, y como se mencionó en el párrafo anterior en la medida que la capacidad de procesamiento ha crecido en forma dramática, los algoritmos hash debieron ajustar su complejidad para estar asegurar los niveles de seguridad demandados por el mercado.

SHA-2

En sus inicios el algoritmo SHA (Secure Hash Algorithm o Algoritmo de Hash Seguro) fue creado por la NSA y el NIST con el objetivo de generar hashes o códigos únicos en base a una norma. En 1993 nació el primer protocolo SHA, también llamado SHA-0, pero apenas se utilizó y no tuvo demasiada repercusión. Un par de años más tarde, vio la luz una variante mejorada más resistente y segura, el SHA-1, que se ha utilizado durante muchos años para firmar los certificados digitales SSL/TLS de millones de webs. Unos años más tarde se creó SHA-2, que tiene cuatro variantes según el número de bits de salida, son SHA2-224, SHA2-256, SHA2-384 y SHA2-512. Actualmente, por seguridad ya no se utiliza SHA1, sino que es muy recomendable utilizar SHA2 o SHA3 (dentro de la familia SHA).

Funcionamiento del SHA2

Los algoritmos de hash solamente funcionan en una dirección, podemos generar el hash de cualquier contenido, o la huella digital, pero con el hash o la huella digital no hay forma de generar el contenido inicial. La única forma de hacerlo, es mediante diccionario o fuerza bruta, por lo que nos podría llevar miles de años (actualmente) conseguir la información inicial.

Entre las muchas y diferentes formas de crear hashes, el algoritmo SHA2-256 es uno de los más usados gracias a su equilibrio entre seguridad y velocidad, es un algoritmo muy eficiente y tiene una alta resistencia a colisiones, algo muy importante para mantener la seguridad de este algoritmo de hash. Para que un algoritmo hash sea seguro, no se deben conocer colisiones. Por ejemplo, el método de verificar los Bitcoins está basado en SHA2-256.

Características de los diferentes tipos de SHA2

- ✓ Tamaño de salida: es el tamaño de caracteres que formarán el hash.
- ✓ Tamaño del estado interno: es la suma hash interna, después de cada compresión de un bloque de datos.
- ✓ Tamaño del bloque: es el tamaño del bloque que maneja el algoritmo.
- ✓ Tamaño máximo del mensaje: es el tamaño máximo del mensaje sobre el que aplicamos el algoritmo.
- ✓ Longitud de la palabra: es la longitud en bits de la operación que aplica en cada ronda el algoritmo.
- ✓ Interacciones o rondas: es el número de operaciones que realiza el algoritmo para obtener el hash final.
- ✓ Operaciones soportadas: son las operaciones que lleva a cabo el algoritmo para obtener el hash final.

SHA-256

Tiene un tamaño de salida de 256 bits, un tamaño de estado interno de 256 bits, un tamaño de bloque de 512 bits, el tamaño máximo del mensaje que puede manejar es de $2^{64} – 1$, la longitud de la palabra es de 32 bits, y el número de rondas que se aplican son 64, así como las operaciones que aplica al hash son +, and, or, xor, shr y rot. La longitud del hash siempre es igual, no importa lo grande que sea el contenido que uses para generar el hash: ya sea de sola una letra o una imagen ISO de 4GB de tamaño, el resultado siempre será una sucesión de 40 letras y números (Figura 10.2.4.5).

SHA2-384

Este algoritmo es diferente en cuanto a características, pero su funcionamiento es el mismo. Tiene un tamaño de salida de 384 bits, un tamaño de estado interno de 512 bits, un tamaño de bloque de 1024 bits, el tamaño máximo del mensaje que puede manejar es de 2128 – 1, la longitud de la palabra es de 64 bits, y el número de rondas que se aplican son 80, así como las operaciones que aplica al hash son +, and, or, xor, shr y rot. Este algoritmo es una versión más segura que el SHA2-256, puesto que se aplican más rondas de operaciones y también puede aplicarse sobre una información más extensa. Este algoritmo de hash se suele utilizar para comprobar la integridad de los mensajes y la autenticidad en las redes privadas virtuales. Un aspecto negativo, es que es algo más lento que SHA2-256, pero en determinadas circunstancias puede ser una muy buena opción usar este.

SHA2-512

Como en todos los SHA2, el funcionamiento es el mismo, cambian una sola característica. Tiene un tamaño de salida de 512 bits. El resto de características son iguales que el SHA2-384. 512 bits de tamaño de estado interno, 1024 bits de tamaño de bloque, 2128 – 1 para el tamaño máximo del mensaje, 64 bits de longitud de palabra, y son 80 el número de rondas que se le aplican. Este algoritmo también aplica las mismas operaciones en cada ronda +, and, or, xor, shr y rot.

SHA2-224

No es un algoritmo muy conocido porque su hermano mayor (SHA2-256) se usa mucho más, ya que la diferencia computacional entre ambos es irrisoria y SHA2-256 está mucho más estandarizado. Sin embargo, se considera necesario mencionarlo porque, por lo menos hasta el momento, no se han encontrado colisiones para este algoritmo, lo que lo convierte en una opción muy segura y altamente recomendable como alternativa a 256.

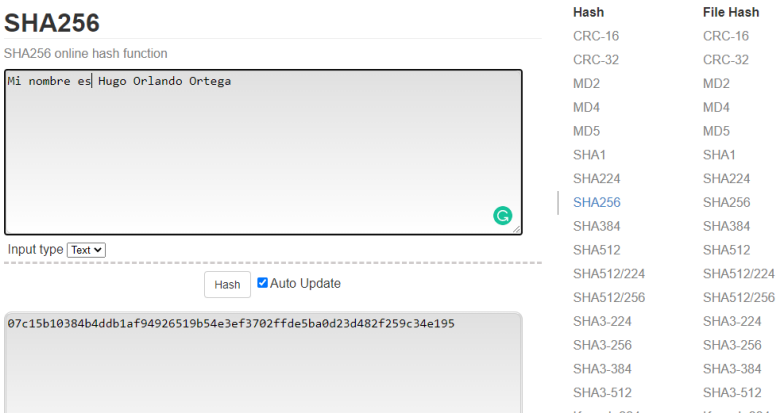

Figura 10.2.4.5: Función Hash en línea (SHA-256)

SHA-3

SHA3 es el algoritmo de hash que pertenece a la familia SHA más nuevo, fue publicado por el NISH en 2015, pero aún no se está utilizando ampliamente. Aunque forma parte de la misma familia, su estructura interna es bastante diferente. Este nuevo algoritmo de hash se basa en la «construcción de esponjas». La construcción de esta esponja se basa en una función aleatoria o permutación aleatoria de datos, permite ingresar cualquier cantidad de datos y generar cualquier cantidad de datos, además, la función es pseudoaleatoria con respecto a todas las entradas anteriores. Esto permite a SHA-3 tener una gran flexibilidad, el objetivo está en sustituir a SHA2 en los típicos protocolos TLS o de VPN que utilicen este algoritmo de hash para comprobar la integridad de los datos y la autenticidad de los mismos.

SHA-3 nació como una alternativa a los SHA2, pero no porque usar SHA-2 sea inseguro, sino porque querían tener un plan B en caso de un ataque exitoso contra SHA2, de esta forma, tanto SHA-2 como SHA-3 convivirán durante bastantes años, de hecho, SHA-3 no se utiliza masivamente como sí ocurre con SHA-2.

10.2.5. Implementación de protocolos criptográficos

Un protocolo criptográfico o también llamado protocolo de cifrado de seguridad, es un protocolo abstracto o concreto que realiza funciones relacionadas con la seguridad, aplicando métodos criptográficos.

En este tipo de protocolo describe la forma en que un algoritmo debe usarse. Los protocolos criptográficos se usan ampliamente para transporte de datos seguros a nivel de aplicación. Un protocolo criptográfico comúnmente incorpora por lo menos uno de los siguientes aspectos:

- ✓ Establecimiento de claves
- ✓ Autenticación de entidades

- ✓ Cifrado simétrico y autenticación de mensajes
- ✓ Transporte de datos en forma segura a nivel de aplicación
- ✓ Métodos de no repudio

Entre los protocolos criptográficos más importantes podemos encontrar a TransportLayer Security (TLS). Es usado en conexiones web (HTTP) seguras. Posee un mecanismo de autenticación de entidades basado en el sistema X.509, una fase de configuración de claves, en la cual se decide una clave de cifrado simétrico mediante el uso de criptografía de clave pública, y una función de transporte de datos de nivel de aplicación. Estos tres aspectos tienen interconexiones importantes. El TLS estándar no provee apoyo para no repudio.

Como ya se mencionó, otro protocolo criptográfico de gran uso es Secure Socket Shell (SSH). Este permite mantener conexiones cifradas y asegura la identidad de las partes.

Protocolo SSH (Secure Socket Shell)

Se verá la aplicación de la criptografía de clave pública dentro de la estructura de un protocolo de uso ampliamente difundido como es SSH.

SSH es un protocolo que especifica cómo conducir comunicaciones seguras en una red de datos y proporciona las siguientes características de seguridad (Figura 10.2.5.1):

- ✓ Cifrado: SSH cifra todas las comunicaciones con variedad de algoritmos para elegir.
- ✓ Autenticación de dos factores: SSH puede requerir un nombre de usuario/contraseña o clave pública para la autenticación. Además, estas dos opciones pueden utilizarse juntas para conformar una autenticación de dos factores.
- ✓ Integridad: SSH puede crear una huella digital de los datos transferidos desde una entidad a otra, lo que garantiza que los datos no han sido modificados o manipulados de ningún modo

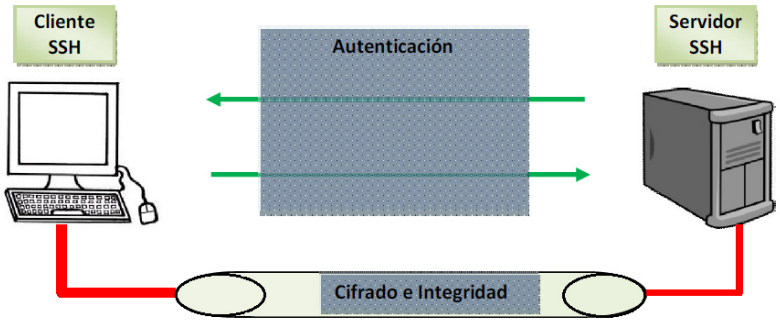

Figura 10.2.5.1: Servicio SSH

Es decir, SSH permite conexiones de red entre equipos, con fuertes garantías de que las partes en ambos extremos de la conexión son genuinas. También asegura que los datos que pasan por estas conexiones llegan sin modificar y no son leídos por terceros no deseados.

Como pude verse en el esquema de la Figura 10.2.5.2, SSH se organiza en un conjunto de tres protocolos por encima de la capa TCP. Por medio de un análisis del proceso de asociación entre el cliente y el servidor veremos cómo los algoritmos de clave pública juegan su papel dentro de este protocolo. Por un lado, se utiliza el algoritmo DH para el intercambio seguro de clave de sesión, y por otro lado nos valemos de un algoritmo asimétrico como RSA o DSA para la autenticación del servidor.

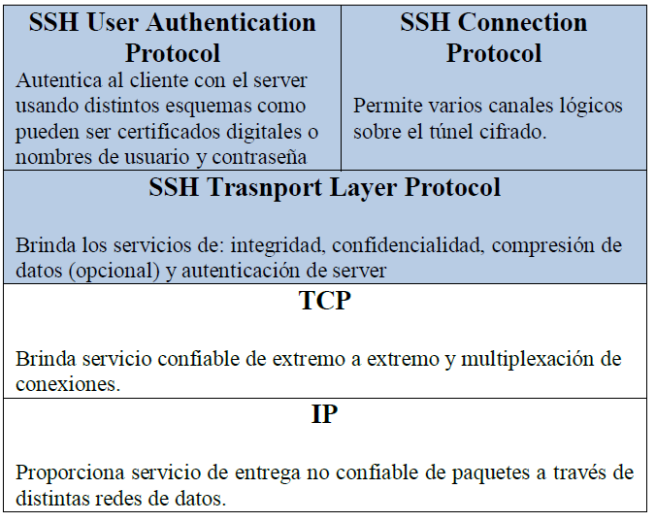

Figura 10.2.5.2: Esquema de servicios del protocolo SSH

En la siguiente Figura 10.2.5.3 se muestra una captura de Wireshark, dónde se puede observar la secuencia de conexión entre un servidor y un cliente OpenSSH con SSH v2.0. Posteriormente, en la Tabla 10.2.5.4, se resumen los pasos, indicando los números de paquetes implicados en cada transacción.

```
 1 TCP    55672 > ssh [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=26417 TSecr
 2 TCP    ssh > 55672 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=1
 3 TCP    55672 > ssh [ACK] Seq=1 Ack=1 Win=14656 Len=0 TSval=26417 TSecr=120407
 4 SSHv2  Server Protocol: SSH-2.0-OpenSSH_5.1p1 Debian-3ubuntu1\r
 5 TCP    55672 > ssh [ACK] Seq=1 Ack=40 Win=14656 Len=0 TSval=26420 TSecr=120410
 6 SSHv2  Client Protocol: SSH-2.0-OpenSSH_5.3p1 Debian-3ubuntu6\r
 7 TCP    ssh > 55672 [ACK] Seq=40 Ack=40 Win=5824 Len=0 TSval=120411 TSecr=26420
 8 SSHv2  Client: Key Exchange Init
 9 TCP    ssh > 55672 [ACK] Seq=40 Ack=832 Win=7424 Len=0 TSval=120411 TSecr=26420
10 SSHv2  Server: Key Exchange Init
11 SSHv2  Client: Diffie-Hellman GEX Request
12 SSHv2  Server: Diffie-Hellman Key Exchange Reply
13 SSHv2  Client: Diffie-Hellman GEX Init
14 SSHv2  Server: Diffie-Hellman GEX Reply
15 TCP    55672 > ssh [ACK] Seq=1000 Ack=1696 Win=19328 Len=0 TSval=26437 TSecr=120419
16 SSHv2  Client: New Keys
17 TCP    ssh > 55672 [ACK] Seq=1696 Ack=1016 Win=8960 Len=0 TSval=121545 TSecr=27354
18 SSHv2  Encrypted request packet len=48
```

Figura 10.2.5.3: Captura de secuencia Hanshake SSH con herramienta Wireshark

Tabla 10.2.5.4: Pasos de una conexión SSH

Primera Parte	Conexión TCP	Paquetes 1 - 3
	Anuncio de versiones	Paquetes 4 y 5
	Negociación de algoritmos	Paquetes 6 y 7
Segunda Parte	Intercambio de claves	Paquetes 8 -11
Tercera Parte	Comienzo del cifrado	Paquete 12
Cuarta Parte	Pedido de autenticación del cliente	---------

10.3. Firewall

La capacidad de conectar una computadora en cualquier lugar, con cualquier otra computadora de cualquier lugar, es una ventaja a medias. Para los usuarios domésticos navegar en Internet representa mucha diversión. Para los gerentes de seguridad empresarial es una pesadilla. Muchas empresas tienen grandes cantidades de información confidencial en línea secretos comerciales, planes de desarrollo de productos, estrategias de marketing, análisis financieros, etc. Si esta información cayera en manos de un competidor, las consecuencias podrían ser devastadoras.

Además del peligro de la fuga de información, también existe el peligro de la infiltración de información. En particular, los virus, gusanos y otras pestes digitales pueden abrir brechas de seguridad, destruir datos valiosos y hacer que los administradores pierdan mucho tiempo tratando de arreglar el daño que hayan hecho. Por lo general, se importan debido a los empleados descuidados que desean ejecutar algún nuevo juego ingenioso.

En consecuencia, se necesitan mecanismos para mantener los bits “buenos” dentro y los bits “malos” fuera. Un método es utilizar IPsec. Este método protege a los datos en tránsito entre los sitios seguros. Sin embargo, IPsec no hace nada por proteger a la LAN de la compañía contra las pestes digitales y los intrusos. Para saber cómo alcanzar ese objetivo, necesitamos dar un vistazo a los firewalls.

Los firewalls (servidores de seguridad) son simplemente una adaptación moderna de la vieja estrategia medieval de seguridad: excavar un foso defensivo profundo alrededor de su castillo. Este diseño obligaba a que todos los que entraran o salieran del castillo pasaran a través de un único puente levadizo, en donde los encargados de la E/S los pudieran inspeccionar. En las redes es posible el mismo truco: una compañía puede tener muchas redes LAN conectadas de forma arbitraria, pero todo el tráfico que entra y sale de la compañía debe pasar a través de un puente levadizo electrónico (firewall), como se muestra en la Figura 10.3.1. No existe ninguna otra ruta.

El firewall actúa como un filtro de paquetes. Inspecciona todos y cada uno de los paquetes entrantes y salientes. Los paquetes que cumplen cierto criterio descrito en reglas formuladas por el administrador de la red se reenvían en forma normal. Los que fallan la prueba simplemente se descartan.

Por lo general, los criterios de filtrado se proporcionan como reglas o tablas que listan los orígenes y destinos aceptables, los orígenes y destinos bloqueados y las reglas predeterminadas acerca de lo que se debe hacer con los paquetes que entran y salen a otras

máquinas. En el caso común de una configuración TCP/IP, un origen o destino podría consistir en una dirección IP y un puerto. Los puertos indican el servicio deseado. Por ejemplo, el puerto TCP 25 es para el correo y el puerto TCP 80 es para HTTP. Algunos puertos simplemente pueden estar bloqueados. Por ejemplo, una empresa podría bloquear los paquetes entrantes para todas las direcciones IP combinadas con el puerto TCP 21 (puerto del servicio FTP, File Transfer Protocol).

Otros puertos no se bloquean tan fácilmente. La dificultad es que los administradores de red desean seguridad, pero no pueden cortar la comunicación con el mundo exterior. Ese arreglo sería mucho más simple y eficiente para la seguridad, pero las quejas de los usuarios finales no terminarían nunca. Aquí es donde pueden ser útiles los arreglos como la DMZ (Zona Desmilitarizada, del inglés DeMilitarized Zone), que se muestra en la Figura 10.3.1. La DMZ es la parte de la red de la empresa que se encuentra fuera del perímetro de seguridad. Cualquier cosa puede pasar aquí. Al colocar una máquina tal como un servidor web en la DMZ, las computadoras en Internet se pueden comunicar con ella para navegar por el sitio web de la empresa. Ahora el firewall se puede configurar para bloquear el tráfico TCP entrante al puerto 80, de modo que las computadoras en Internet no puedan usar este puerto para atacar a las computadoras en la red interna. Para poder administrar el servidor web, el firewall puede tener una regla que permita conexiones entre las máquinas internas y el servidor web.

Con el tiempo, los firewall's se han vuelto mucho más sofisticados en una carrera armamentista contra los atacantes. En un principio, los firewall's aplicaban una regla que se establecía de manera independiente para cada paquete, pero se dificultaba el proceso de escribir reglas que permitieran una funcionalidad útil y bloquearan a la vez todo el tráfico no deseado. Los firewall's con estado asocian paquetes a las conexiones y usan campos del encabezado TCP/IP para mantener un registro de las conexiones. Esto permite usar reglas que, por ejemplo, permitan a un servidor web externo enviar paquetes a un host interno, pero sólo si ese host interno establece primero una conexión con el servidor web externo. Dicha regla no es posible con diseños sin estado, en los que se deben pasar o descartar todos los paquetes del servidor web externo.

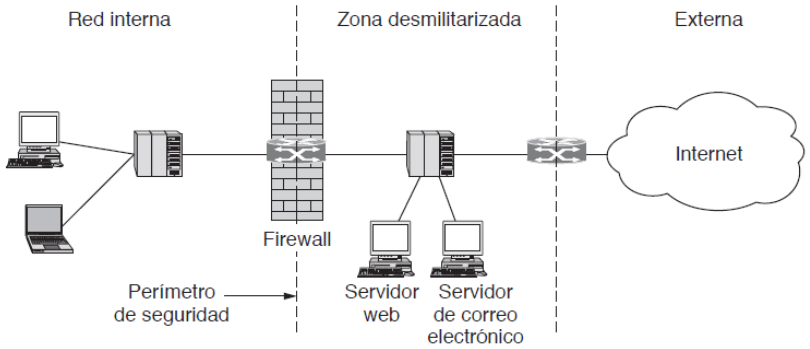

Figura 10.3.1: Firewall que protege una red interna

Otro nivel de sofisticación superior al procesamiento con estado es que el firewall implemente puertas de enlace a nivel de aplicación. Este procesamiento implica que el firewall analice el interior de los paquetes, todavía más allá del encabezado TCP, para ver lo que está haciendo la aplicación. Con esta capacidad es posible diferenciar el tráfico HTTP que se utiliza para navegar por la web del tráfico HTTP utilizado para compartir archivos entre iguales. Los administradores pueden escribir reglas para evitar la compartición de archivos de igual a igual dentro de la empresa, pero permitir la navegación web que es vital para los negocios. En todos estos métodos se puede inspeccionar tanto el tráfico de salida como el de entrada; por ejemplo, para evitar que documentos confidenciales se envíen por correo electrónico fuera de la empresa.

La discusión anterior deja en claro que los firewall's violan la distribución estándar en capas de los protocolos. Son dispositivos de capa de red, pero husmean en las capas de transporte y de aplicación para realizar su operación de filtrado. Esto los hace frágiles. Por ejemplo, los firewall's tienden a depender de las convenciones de numeración de los puertos para determinar el tipo de tráfico que transporta un paquete. A menudo se utilizan los puertos estándar, pero no todas las computadoras ni todas las aplicaciones lo hacen. Algunas aplicaciones de igual a igual seleccionan puertos en forma dinámica para evitar que las detecten (y bloqueen) fácilmente. La encriptación mediante IPSEC u otros esquemas oculta la información de las capas superiores al firewall. Por último, un firewall no puede comunicarse fácilmente con las computadoras que se comunican a través de él para decirles qué políticas se están aplicando y por qué se va a cortar su conexión. Simplemente debe pretender ser un cable roto. Por todas estas razones, los puristas de las redes consideran que los firewall's son una imperfección en la arquitectura de Internet. Sin embargo, Internet puede ser un lugar peligroso si usted es una computadora. Los firewall's ayudan con ese problema, por lo que es probable que se sigan utilizando.

Incluso aunque el firewall esté configurado a la perfección, de todas formas, existirán muchos problemas de seguridad. Por ejemplo, si un firewall está configurado para permitir sólo paquetes entrantes de redes específicas (por ejemplo, las demás plantas de la empresa), un intruso afuera del firewall puede introducir falsas direcciones de origen para evadir esta verificación. Si un miembro interno desea enviar documentos secretos, puede encriptarlos o incluso fotografiarlos y enviar las fotos como archivos JPEG, los cuales pueden evadir cualquier filtro de correo electrónico. Y no hemos analizado todavía el hecho de que, aunque tres cuartas partes de los ataques provienen del exterior del firewall, por lo general los ataques más dañinos son los que provienen del interior; por ejemplo, de empleados descontentos (BUSINESS, 2009).

Hay un problema distinto con los firewall's: proporcionan un solo perímetro de defensa. Si se viola esa defensa, se cierran todas las apuestas. Por esta razón, los firewall's se utilizan comúnmente en una defensa por capas. Por ejemplo, un firewall puede proteger la entrada a la red interna y cada computadora puede ejecutar también su propio firewall. Sin duda, los lectores que piensen que un solo punto de verificación de seguridad es suficiente no han tomado recientemente un vuelo internacional en una aerolínea programada.

Además, hay otra clase de ataques que los firewall's no pueden manejar. La idea básica de un firewall es evitar que entren intrusos y que salga información secreta. Por desgracia, hay personas que no tienen nada mejor que hacer que tratar de inhabilitar ciertos

sitios. Para ello envían grandes cantidades de paquetes legítimos al destino, hasta que el sitio se colapsa debido a la carga. Por ejemplo, para derribar un sitio web, un intruso puede enviar un paquete TCP SYN para establecer una conexión. A continuación, el sitio asignará una ranura en la tabla para la conexión y enviará como respuesta un paquete SYN 1 ACK. Si el intruso no responde, la ranura de la tabla se conservará durante algunos segundos hasta que expire. Si el intruso envía miles de solicitudes de conexión, todas las ranuras de la tabla se llenarán y no será posible establecer ninguna conexión legítima. Los ataques en los que el objetivo del intruso es bloquear el destino en lugar de robar datos se conocen como ataques DoS (Negación de Servicio, del inglés Denial of Service). Por lo general, los paquetes de solicitud tienen direcciones de origen falsas, por lo que no es fácil rastrear al intruso. Los ataques DoS contra grandes sitios web son comunes en Internet.

Una variante aún peor es aquella en la que el intruso ha entrado en cientos de computadoras en cualquier parte del mundo, y después ordena a todas ellas que ataquen al mismo objetivo y al mismo tiempo. Este método no sólo incrementa el poder del intruso, sino que también reduce la probabilidad de detectarlo debido a que los paquetes provienen de una gran cantidad de máquinas que pertenecen a usuarios ingenuos. Un ataque de este tipo se conoce como ataque DDoS (Negación de Servicio Distribuida, del inglés Distributed Denial of Service). Es difícil defenderse de un ataque como éste. Incluso si la máquina atacada puede reconocer rápidamente una solicitud falsa, le toma algún tiempo procesar y descartar la solicitud, y si llegan suficientes solicitudes por segundo, la CPU pasará todo su tiempo ocupándose de ellas.

10.4. Firewall de uso libre: iptables

iptables se usa para crear, mantener y revisar las tablas de filtrado de paquetes en el kernel de Linux, y se estructura de la siguiente manera:

- ✓ Existen diferentes tablas (“tables”) dentro de las cuales puede haber varias cadenas (“chains”).
- ✓ Cada cadena consiste en una lista de reglas con las que se comparan los paquetes que pasan por el firewall. Las reglas especifican qué se hace con los paquetes que se ajustan a ellas (target).

Para cada paquete que recibe el firewall, se examina la primera regla de la cadena correspondiente. Si el paquete no se ajusta a esa regla, se continúa examinando la siguiente hasta que se ajusta con alguna. En ese momento se ejecuta el target:

DROP: el paquete se descarta, no puede pasar

ACCEPT: el paquete continúa su camino normal

Si se llega al final de una cadena predefinida, y el tipo de paquete no se ajusta a ninguna regla, se ejecuta un target por defecto, llamado “chain policy”. El “chain policy” establece, por tanto, la política por defecto de un determinado firewall.

Tablas

Existen varias tablas, que tienen diferentes objetivos:

1. La tabla "filter" es la tabla por defecto, y se utiliza para especificar filtros de paquetes. Contiene 3 "chains" predefinidas:
   - ✓ INPUT: Se consulta para los paquetes que van dirigidos al propio firewall
   - ✓ FORWARD: La atraviesan los paquetes enrutados a través de esta máquina, es decir, aquellos paquetes en los que el origen y el destino son equipos de redes diferentes.
   - ✓ OUTPUT: Para paquetes generados localmente

2. La tabla "nat" se consulta cada vez que se ve un paquete que inicia una nueva conexión, con el objetivo de alterar algún parámetro de esa conexión. Permiten compartir una IP pública entre muchos equipos, por lo que resultan imprescindibles en el protocolo IPv4. Con ellas podemos añadir reglas para modificar las direcciones de IP de los paquetes, y contienen dos reglas: SNAT (IP masquerading) para la dirección de origen y DNAT (Port Forwarding) para las direcciones destino. Tiene 3 "chains" predefinidas:
   - ✓ PREROUTING: se consulta con los paquetes que entran en la máquina firewall, tan pronto como llegan, antes de decidir qué hacer con ellos.
   - ✓ OUTPUT: se utiliza para alterar paquetes generados localmente, antes de enrutarlos.
   - ✓ POSTROUTING: para alterar los paquetes que están a punto de salir de la máquina.
3. La tabla "mangle" es una tabla especial, destinada a alterar determinados parámetros de los paquetes (TOS, TTL …), que se utiliza para realizar configuraciones complejas del firewall. Cuenta con 5 chains predefinidas: INPUT, OUTPUT, PREROUNTING, POSTROUTING Y FORWARD. Las tablas "mangle" se encargan de modificar los paquetes, y para ello tienen las opciones
   - ✓ TOS: Type Of Service es usado para definir el tipo de servicio de un paquete y se debe usar para definir cómo los paquetes deben ser enrutados, no para paquetes que vayan hacia Internet. La mayoría de los routers no hacen caso del valor de este campo o pueden actuar de forma imperfecta si se usan para su salida a Internet.
   - ✓ TTL: cambia el campo de tiempo de vida de un paquete. Sus siglas corresponden a Time To Live y, por ejemplo, se puede usar para cuando no queremos ser descubiertos por ciertos proveedores de servicios de Internet (ISP) que sean demasiado fisgones.
   - ✓ MARK: usado para marcar paquetes con valores específicos, consiguiendo liminar el ancho de banda y generar colas mediante CBQ (Class Based

Queuing). Posteriormente pueden ser reconocidas por programas como iproute2 para realizar los diferentes enrutamientos dependiendo de la marca que tengan o no estos paquetes.

Flujo de paquetes a través de iptables

La Figura 10.4.1 muestra todos los caminos que puede seguir una trama que atraviesa el firewall:

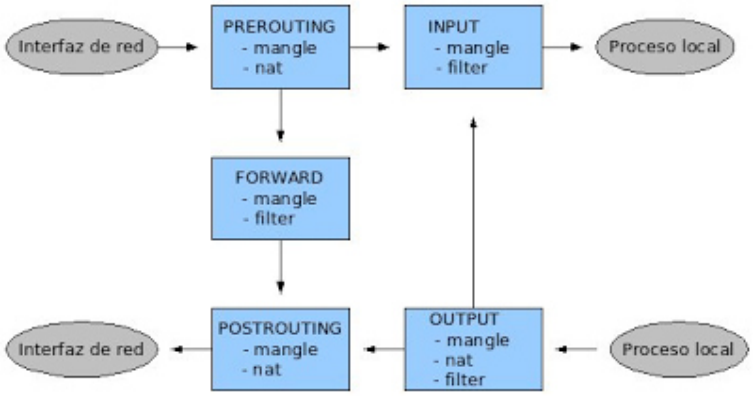

Figura 10.4.1: Flujo de tramas en un firewall

El paquete entra en el firewall por una interfaz de red, por tanto, primero se comprobarían las reglas de la cadena PREROUTING.

A continuación, se comprobarían las reglas de la cadena FORWARD, ya que el paquete no va destinado a un proceso del firewall, si no que va a atravesarlo, saliendo por la interfaz que lo conecta con la red externa.

Por último, si el paquete no ha sido filtrado y sigue adelante, antes de salir del firewall por la otra interfaz de red, se comprueban las reglas de la cadena POSTROUTING.

Sintaxis iptables

iptables [-t table] -[AD] chain rule-specification [options]

iptables [-t table] -I chain [rulenum] rule-specification [options]

iptables [-t table] -R chain rulenum rule-specification [options]

iptables [-t table] -D chain rulenum [options]

iptables [-t table] -[LFZ] [chain] [options]

iptables [-t table] -N chain

iptables [-t table] -X [chain]

iptables [-t table] -P chain target [options]

iptables [-t table] -E old-chain-name new-chain-name

Donde "table" es el nombre de la tabla (filter, nat o mangle) a la cual aplicar un "Argumento":

- ✓ -A –append → agrega una regla a una cadena.
- ✓ -D –delete → borra una regla de una cadena especificada.
- ✓ -R –replace → reemplaza una regla.
- ✓ -I –insert → inserta una regla en un lugar de una cadena.
- ✓ -L –list → muestra las reglas que le pasamos como argumento.
- ✓ -F –flush → borra todas las reglas de una cadena.
- ✓ -Z –zero → pone a cero todos los contadores de una cadena.
- ✓ -N –new-chain → permite al usuario crear su propia cadena.
- ✓ -X –delete-chain → borra la cadena especificada.
- ✓ -P –policy → explica al kernel qué hacer con los paquetes que no coincidan con ninguna regla.
- ✓ -E –rename-chain → cambia el orden de una cadena.

Dependiendo del argumento existen condiciones. Las condiciones más usadas son:

- ✓ -p –protocol → la regla se aplica a un protocolo.
- ✓ -s –src –source → la regla se aplica a una IP de origen.
- ✓ -d –dst –destination → la regla se aplica a una Ip de destino.
- ✓ -i –in-interface → la regla de aplica a una interfaz de origen, como eth0.
- ✓ -o –out-interface → la regla se aplica a una interfaz de destino.

Si las condiciones se aplicaron a los protocolos TCP o UDP, se pueden considerar condiciones específicas de los protocolos TCP/UDP:

- ✓ -sport –source-port → selecciona o excluye puertos de un determinado puerto de origen.
- ✓ -dport –destination-port → selecciona o excluye puertos de un determinado puerto de destino.

Los "target" (qué hacer con el paquete) dependen de la tabla a la cual son aplicados. Los más usados son:

Para la tabla "filter"

- ✓ ACCEPT: se acepta el tráfico que coincide con la regla
- ✓ DROP : se descarta el tráfico que coincide con la regla
- ✓ REJECT: se rechaza el tráfico que coincide con la regla

Para la tabla "nat"

- ✓ DNAT: al tráfico que coincida con la regla, le será modificada la cabecera IP, en este caso la dirección ip destino.

- ✓ SNAT: al tráfico que coincida con la regla, le será modifica la cabecera IP, en concreto en este caso la dirección ip origen (Se modifica la dirección ip privada interna por la dirección ip pública de la interface externa del router)
- ✓ MASQUERADE: igual que la regla anterior, pero se usa cuando la dirección ip pública de la interface externa del router no es fija)

Es tiempo de aplicar los comandos

Se irá explicando a continuación, los comandos básicos de configuración de iptables. Se verá algún ejemplo sencillo, para luego desarrollar un ejemplo completo de configuración de un firewall con DMZ.

Listado de reglas

iptables [-t tabla] [opciones] -L [chain]

iptables -L -n –v

Donde,

-L : Muestra las reglas.

-v : Muestra información detallada.

-n : Muestra la dirección ip y puerto en formato numérico. No usa DNS para resolver nombres. Esto acelera la lista.

Borrado de contadores

iptables [-t tabla] -Z [chain]

Borra los contadores de una determinada chain, o de todas ellas. Es habitual colocar este comando al principio de todos los script's de configuración, para borrar las reglas que existieran de antemano. Se puede combinar con la opción -L para mostrar la información justo antes de borrarla.

Borrado de todas las reglas

iptables -F

iptables -X

iptables -t nat -F

iptables -t nat -X

iptables -t mangle -F

iptables -t mangle -X

iptables -P INPUT ACCEPT

iptables -P OUTPUT ACCEPT

iptables -P FORWARD ACCEPT

Donde:

-F : Borra todas las reglas.

-X : Borra cadenas

-t table_name : Selecciona una tabla y elimina reglas

-P : Establece la política por defecto (como DROP, REJECT o ACCEPT)

Creación y borrado de reglas

iptables [-t table] -A chain rule-spec

iptables [-t table] -I chain [rulenum] rule-spec

iptables [-t table] -R chain rulenum rule-spec

iptables [-t table] -D chain rule-spec

iptables [-t table] -D chain rulenum

Las opciones son las siguientes:

-A: añade una regla al final de la lista

-I: inserta una regla al comienzo de la lista o en el punto especificado

-R: reemplaza una regla (especificada por su número de regla) por otra

-D: borra una regla determinada

Para especificar una regla podemos usar los siguientes parámetros:

-p [!] protocolo: el protocolo del paquete a comprobar. Puede ser 'tcp', 'udp', 'icmp' o 'all'

-[sd] [!] dirección[/máscara]: dirección ip origen (s) o destino (d) del paquete

-[io] [!] iface: nombre de la interfaz de entrada (i) o de salida (o) del paquete

-j target: especifica el target de dicha regla. Puede ser una acción predefinida (ACCEPT, DROP), una extensión o el nombre de una chain

Establecimiento de una política por defecto

iptables [-t table] -P chain target

Establece el "target" que se ejecutará para los paquetes que no cumplan con ninguna regla de la chain especificada.

Ejemplo de Filtrado de Tráfico

Se verá un ejemplo sencillo para entender cómo se deben escribir las reglas en iptables. La la Figura 10.4.2, que representa una instalación sencilla, con una máquina que actúa como router y firewall conectando dos redes distintas.

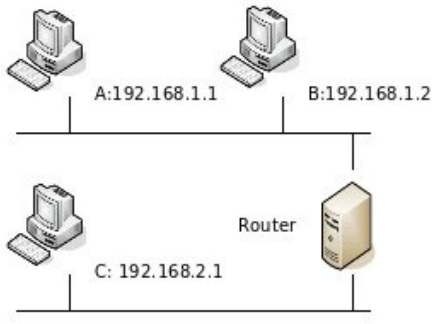

Figura 10.4.2: Redes conectada por un firewall/router

El objetivo es que sólo la máquina B pueda hablar con C, y además sólo pueda usar el protocolo TCP. Los paquetes que no cumplan estas condiciones serán descartados. Lo primero que se debe decidir es ¿en qué tabla y en qué cadena se agregaran las reglas? En este caso, se desea realizar filtrado de paquetes, por lo que la tabla "filter" es la indicada. Respecta a la elección de la cadena correspondiente (INPUT, OUTPUT o FORWARD). Como los paquetes que se quieren filtrar son los que van a atravesar el Firewall de una red a otra, lo correcto es elegir la cadena FORWARD. Los comandos que se deben usar para conseguir estos objetivos son los siguientes:

```
iptables -F FORWARD
iptables -P FORWARD DROP
iptables -t filter -A FORWARD -p tcp -s 192.168.1.2 -d 192.168.2.1 -j ACCEPT
iptables -t filter -A FORWARD -p tcp -s 192.168.2.1 -d 192.168.1.2 -j ACCEPT
```

- ✓ La línea 1 borra las reglas que pudiera haber en la cadena FORWARD de la tabla filter (que es la tabla por defecto).
- ✓ La línea 2 establece la política por defecto a DROP (denegar), por lo tanto, los paquetes que no cumplan con las reglas que especifiquemos serán rechazados.
- ✓ Las líneas 3 y 4 permiten el tráfico entre las máquinas B y C según las reglas especificadas en el enunciado.
- ✓ -t filter, significa que se va a usar la tabla "filter" (es la tabla por defecto)
- ✓ -A FORDWARD, añadiendo la siguiente regla a la cadena FORDWARD
- ✓ -p TCP, selecciona los paquetes cuyo protocolo sea TCP
- ✓ -s 192.168.1.2, cuya dirección origen sea 192.168.1.2
- ✓ -d 192.168.1.1, cuya dirección destino es 192.168.1.1

- ✓ -j ACCEPT, acepta esos paquetes para su reenvío

Extensiones de las reglas

Con las opciones vistas hasta el momento se puede controlar los parámetros más básicos de la cabecera IP del paquete. Puede ser necesario un control más estricto, por ello existen una serie de extensiones que permiten utilizar opciones nuevas:

-m extensión: activa una extensión para poder especificar los parámetros del paquete.

Algunas extensiones de las reglas:

- ✓ tcp: añade las siguientes opciones:

  --sport [!] port[:port]: especifica el puerto o rango de puertos origen

  --dport [!] port[:port]: especifica el puerto o rango de puertos destino

  [!] --syn: la regla concordará sólo con los paquetes cuyo bit SYN=1 y los flags ACK y FIN valgan 0. Los datagramas con estos valores se utilizan para abrir las conexiones TCP

- ✓ udp: añade las siguientes opciones:

  --sport [!] port[:port]: especifica el puerto o rango de puertos origen

  --dport [!] port[:port]: especifica el puerto o rango de puertos destino

  Ejemplo: se supone que se quiere crear una regla para permitir el paso a las peticiones a un servidor web que tiene la IP 172.16.0.254.

  iptables -A FORWARD -p tcp -d 172.16.0.254 --dport 80 -j ACCEPT

- ✓ icmp: añade la opción --icmp-type tipo, que indica qué tipo ICMP debe tener el paquete (echo-request, echo-reply, network-unreachable…)
- ✓ mac: añade la opción --mac-source [!] dir_mac, que especifica la dirección MAC que debe tener el paquete.

  Ejemplo: se quiere permitir que se realice ping al firewall desde la máquina del administrador, que tiene IP dinámica y MAC 00:21:00:16:82:9f

  iptables -A INPUT -p icmp --icmp-type echo-request -m mac --mac-source 00:21:00:16:82:9f -j ACCEPT

  iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT

- ✓ state: Añade la opción --state valor, que indica el estado en el que debe estar la conexión correspondiente a dicho paquete. Los tipos de estado principales son:

  NEW: el paquete corresponde a una conexión nueva

  ESTABLISHED: el paquete está asociado a una conexión ya establecida

  RELATED: el paquete corresponde a una conexión nueva, pero relacionada con una que ya está establecida (como un canal de datos FTP o un error de ICMP)

- ✓ multiport: Nos permite indicar con las opciones --sports y --dports varios puertos.

Extensiones de target

- ✓ MASQUERADE: sólo es válida en la chain POSTROUTING. Indica que la dirección origen del paquete (y de todos los de esa conexión) será cambiada por la IP local de esta máquina. Muy útil para IP dinámica (es lo que se conoce como NAT)

  Ejemplo:

  iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE

- ✓ SNAT: sólo es válida en la chain POSTROUTING. Indica que la dirección y puerto origen del paquete (y de todos los de esa conexión) sea modificada según se especifica con la opción --to-source. El target será SNAT (en lugar de MASQUERADE) cuando tengamos una IP fija.

  Ejemplo:

  iptables -t nat -A POSTROUTING -o eth1 -j SNAT --to-source 189.29.35.15

- ✓ DNAT: sólo es válida en las chains PREROUTING y OUTPUT. Cambia la dirección IP destino (y de todos los futuros de esta misma conexión) por el especificado con la opción --to-destination. Es lo que se conoce como 'abrir el puerto' en el router.

  Por ejemplo, si la máquina 192.168.1.2 de nuestra red local aloja un servidor web que queremos que sea accesible desde la red externa:

  iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT --to-destination 192.168.1.2:80

Configuración integral de un Firewall usando iptables

En este ejemplo se supone que nuestra máquina GNU/Linux está conectada a un ROUTER que nos proporciona el acceso a Internet por la interfaz eth2 (Figura 10.4.3). Este router está configurado como monopuesto (no hace NAT y deja pasar todo el tráfico a nuestra máquina Linux).

Las direcciones de la máquina son las siguientes:

- ✓ eth0: 192.168.1.1/24
- ✓ eth1: 192.168.2.1/24
- ✓ eth2: dirección IP obtenida por DHCP (dinámica)

En la DMZ se cuenta con un servidor WEB y un servidor SSH los cuales deberían ser accesibles desde Internet y la red local. Las direcciones son:

- ✓ Servidor WEB: 192.168.2.2/24
- ✓ Servidor FTP: 192.168.2.3/24

Las máquinas de la red local podrán navegar por Internet accediendo a servidores WEB, servidores WEB Seguros, servidores FTP y servidores DNS. El resto de conexiones serán filtradas.

Todas las máquinas de la red local tienen direcciones del rango 192.168.1.0/24.

Se determina que en el firewall hay instalado un servidor ssh para que el administrador pueda conectarse desde la red local.

Se presenta a continuación un script, con una parte configurable por el usuario, para que pueda ser fácilmente adaptado a otras instalaciones y requisitos:

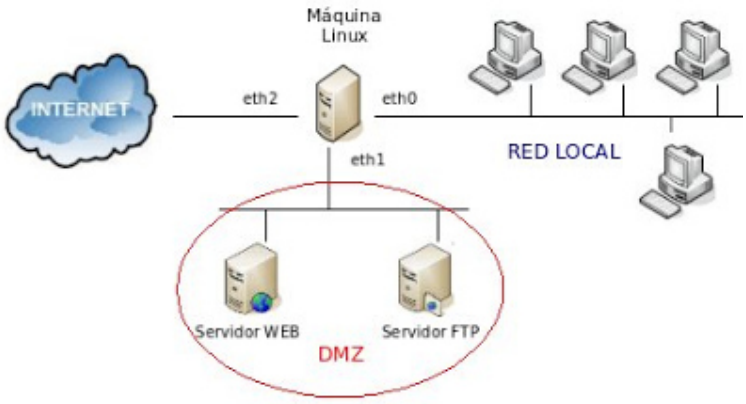

Figura 10.4.3: Firewall que se configura con “iptables”

SECCIÓN CONFIGURABLE POR EL USUARIO

REDLOCAL Dirección IP de la red local interna

BCASTLOCAL Dirección de Broadcast de la red local

IFAZLOCAL Nombre del interfaz de red local

CUALQUIERA Dirección de red 0.0.0.0

IFAZEXT Nombre del interfaz de red externo

WEB Dirección IP del servidor WEB

FTP Dirección IP del servidor FTP

REDDMZ Dirección IP de la red DMZ

IFAZDMZ Nombre del interfaz de red de la DMZ

TCPLOCAL Lista de puertos TCP que permitimos usar en la red local

```
# UDPLOCAL    Lista de puertos UDP que permitimos usar en la red local

REDLOCAL = "192.168.1.0/24"
BCASTLOCAL = "192.168.1.255"
IFAZLOCAL = "eth0"

CUALQUIERA = "0/0"
IFAZEXT = "eth2"

WEB = "192.168.2.2"
FTP = "192.168.2.3"
REDDMZ = "192.168.2.0/24"
IFAZDMZ = "eth1"

TCPLOCAL = "www,ftp,ftp-data,https"

UDPLOCAL = "domain"


            ###IMPLEMENTACIÓN###

# Borramos todas las reglas que pudiera haber anteriormente, y los contadores

iptables -F
iptables -t nat -F
iptables -X         #borra las cadenas creadas por el usuario
iptables -Z
# Cambiamos la política por defecto en todas las cadenas de la tabla filter
# Chain Policy -> DROP

iptables -P FORWARD DROP
```

```
iptables -P INPUT DROP
iptables -P OUTPUT DROP

# Permitimos que se acceda al cortafuegos por ssh

iptables -A INPUT -s $REDLOCAL -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -d $REDLOCAL -p tcp --sport 22 -j ACCEPT

##CONEXIONES TCP

# Permitimos las conexiones establecidas, desde la red externa y desde la DMZ hacia la red local en los puertos permitidos

iptables -A FORWARD -d $REDLOCAL -p tcp  -m state --state ESTABLISHED -m multiport --sports $TCPLOCAL -j ACCEPT

# Permitimos el tráfico desde la red local hacia la red externa y hacia la DMZ en los puertos permitidos

iptables -A FORWARD -s $REDLOCAL -p tcp -m multiport --dports $TCPLOCAL -j ACCEPT

# Permitimos el tráfico desde la red local y desde la red externa al servidor WEB  por el puerto 80

iptables -A FORWARD -d $WEB -p tcp --dport 80 -j ACCEPT

# Permitimos las conexiones establecidas desde el servidor WEB a la red local y a la red externa

iptables -A FORWARD -s $WEB -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT
```

Permitimos el tráfico desde la red local y desde la red externa al servidor FTP por el puerto 20 y 21

iptables -A FORWARD -d $FTP -p tcp --dport 20:21 -j ACCEPT

Permitimos las conexiones establecidas desde el servidor FTP a la red local y a la red externa

iptables -A FORWARD -s $FTP -p tcp --sport 20:21 -m state --state ESTABLISHED -j ACCEPT

Permitimos las conexiones relacionadas desde el servidor FTP a la red local y a la red externa

iptables -A FORWARD -s $WEB -p tcp --sport 20:21 -m state --state RELATED -j ACCEPT

##CONEXIONES UDP

Permitimos las conexiones establecidas desde la red externa hacia la red local en los puertos permitidos

iptables -A FORWARD -d $REDLOCAL -i $IFAZEXT -p udp -m state --state ESTABLISHED -m multiport --sports $UDPLOCAL -j ACCEPT

Permitimos el tráfico desde la red local hacia la red externa en los puertos permitidos

iptables -A FORWARD -s $REDLOCAL -o $IFAZEXT -p udp -m multiport --dports $UDPLOCAL -j ACCEPT

Hacemos NAT para las máquinas de la red local y para las máquinas de la DMZ

iptables -t nat -A POSTROUTING -o eth2 -j MASQUERADE

Si la dirección IP Pública fuera estática haríamos SNAT

Hacemos DNAT para que las peticiones que recibimos de Internet lleguen a los servidores

iptables -t nat -A PREROUTING -i eth2 -p tcp --dport 80 -j DNAT --to-destination $WEB:80

iptables -t nat -A PREROUTING -i eth2 -p tcp –dport 20 -j DNAT --to-destination $FTP:20