9 SEGURIDAD

Al principio de su existencia, las redes de ordenadores fueron usadas generalmente para el envío de correo electrónico y para compartir recursos, generalmente impresoras, en empresas de mediano/gran tamaño. En estas condiciones la seguridad carecía prácticamente de importancia y no fue objeto de atención. Sin embargo, en la actualidad millones de ciudadanos usan redes para transacciones bancarias, compras, etc., la seguridad aparece como un problema potencial de grandes proporciones. Los problemas de seguridad de las redes pueden dividirse de forma general en cuatro áreas interrelacionadas:

- El secreto, encargado de mantener la información fuera de las manos de usuarios no autorizados.
- La validación de identificación, encargada de determinar la identidad de la persona/computadora con la que se esta hablando.
- El no repudio, encargado de asegurar la “firma” de los mensajes, de igual forma que se firma en papel una petición de compra/venta entre empresas.
- El control de integridad, encargado de asegurar que el mensaje recibido fue el enviado por la otra parte y no un mensaje manipulado por un tercero.

Aunque muchos de estos problemas tratan de resolverse en capas de la red que se encuentran por debajo de la capa de aplicación, por ejemplo en la capa de red pueden instalarse muros de seguridad para mantener adentro (o afuera) los paquetes, en la capa de transporte pueden cifrarse conexiones enteras terminal a terminal, ninguna de ellas resuelve completamente los problemas de seguridad antes enumerados.

La resolución de estos problemas de seguridad se realiza como una parte previa o de apoyo de la capa de aplicación. A continuación se expondrán distintos trabajos que tratan de resolver cada uno de los cuatro problemas de seguridad planteados con anterioridad, esto es, el secreto, la validación de identificación, el no repudio y el control de integridad.

Antes de comenzar este capítulo, pasemos a realizar una serie de definiciones:

1. La Organización Internacional de Estándares (ISO), como parte de su norma 7498 en la que se establece el modelo de referencia para la interconexión de sistemas abiertos, define la ***seguridad informática como una serie de mecanismos que minimizan la vulnerabilidad de bienes y recursos***, donde un bien se define como algo de valor y la vulnerabilidad se define como la debilidad que se puede explotar para violar un sistema o la información que contiene. Para ello, se han desarrollado protocolos y mecanismos adecuados, para preservar la seguridad.
2. El **criptoanálisis**, es la ciencia que se encarga de descifrar los mensajes (los intrusos utilizan estas técnicas), mientras que la **criptografía** busca métodos más seguros de cifrado, y se puede clasificar en:
    - Criptografía **clásica**: cifrados rudimentarios basados en sustitución y trasposición
    - Criptografía **moderna**: cifrados basados en algoritmos parametrizados en base a claves
3. "**seguridad de una red**" implica la seguridad de cada uno de las computadoras de la red
4. "**hacker**": cualquier barrera es susceptible de ser superada y tiene como finalidad la de salir de un sistema informático (tras un ataque) sin ser detectado. Es un programador
5. "**cracker**": no es un programado y utiliza sus ataques para sacar beneficio económico
6. **"Amenaza o ataque":** intento de sabotear una operación o la propia preparación para sabotearla (poner en compromiso), que a su vez, estas amenazas se pueden realizar por:
    - **Compromiso**: la entidad atacante obtiene el control de algún elemento interno de la red, por *ejemplo utilizando cuentas con contraseña trivial o errores del sistema*
    - **Modificación**: la entidad atacante modifica el contenido de algún mensaje o texto
    - **Suplantación**: la entidad atacante se hace pasar por otra persona**Reenvío**: la entidad atacante obtiene un mensaje o texto en tránsito y más tarde lo reenvía para duplicar su efecto**Denegación de servicio**: la entidad atacante impide que un elemento cumpla su función

También es importante resaltar, que los temas que vinculan a seguridad son muy peliagudos y están íntimamente relacionados con **temas legales**, los cuales no debemos de dejar de lado. Muestra de ello, es que en muchos gobiernos el uso de información cifrada está prohibido. Los Gobiernos tratan de implantar reglas (o estándares de cifrado) que ellos mismos puedan descifrar fácilmente. Por ejemplo en Francia y EEUU no están permitidas transacciones cifradas, que el gobierno no sea capaz de descifrar, pues pueden utilizarse para comercio de armas, delincuencia, ...

Este capítulo lo vamos a estructurar en cuatro partes:

1.- Secretos: criptografía
2.- Protocolos de seguridad: autenticación y validación
3.- Aplicaciones y seguridad: integridad y no repudio, firmas digitales y certificados
4.- Redes y seguridad

9.1 SECRETOS

Resolución del problema de seguridad del secreto.

La resolución del problema del secreto en la red (y del secreto de los mensajes en cualquier sistema de comunicación), ha estado siempre unido al cifrado (codificación) de los mensajes. Hasta la llegada de las computadoras, la principal restricción del cifrado consistía en la capacidad del empleado encargado de la codificación para realizar las transformaciones necesarias. Otra restricción adicional consistía en la dificultad de cambiar rápidamente el método de cifrado, pues esto implicaba entrenar a una gran cantidad de empleados. Sin embargo, el peligro de que un empleado fuera capturado por el enemigo, etc., ha hecho indispensable la capacidad de cambiar el método de cifrado al instante. De estos requisitos se deriva el modelo de la figura siguiente:

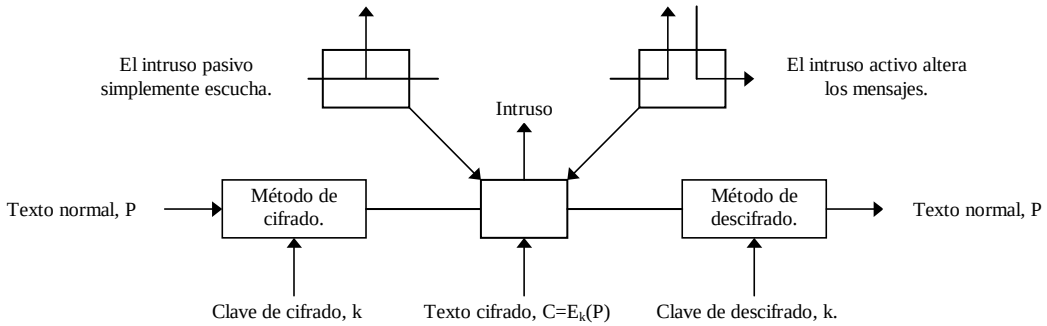


Figura 1 : Modelo de cifrado de datos

Los mensajes a cifrar, conocidos como texto normal, se transforman mediante una función parametrizada por una clave. La salida del cifrado, conocida como texto cifrado, es transmitida después. Si un intruso escucha y copia el texto cifrado, a diferencia del destinatario original, no conoce la clave de cifrado y no puede descifrar fácilmente el texto cifrado.

A partir de aquí usaremos $C=E_k(P)$ para indicar que el cifrado del texto normal P usando la clave K da el texto cifrado C. Del mismo modo $P=D_k(C)$ representa el descifrado de C para obtener el texto normal nuevamente. Por tanto, $D_k(E_k(P))=P$. Esta notación sugiere que E y D son sólo funciones matemáticas de dos parámetros, de los cuales hemos escrito uno (la clave) como subíndice, en lugar de como argumento, para distinguirlo del mensaje.

Todo esto que estamos hablando queda vinculado a la criptografía: KRYPTOS oculto + GRAPHE escrito en grigo.

Una regla fundamental de la criptografía es que se debe suponer que el criptoanalista conoce el método general de cifrado usado, esto es, el criptoanalista conoce E, pues la cantidad de esfuerzo necesario para inventar, probar e instalar un método nuevo cada vez que el viejo es conocido siempre hace impracticable mantenerlo en secreto. Aquí es donde entra la clave. La clave consiste en una cadena relativamente corta que selecciona uno de los muchos cifrados potenciales. En contraste con el método general, que tal vez se cambie cada cierto número de años, la clave puede cambiarse con la frecuencia que se requiera, por lo cual nuestro modelo es un método general estable y conocido públicamente pero parametrizado por una clave secreta y fácilmente cambiable. Un ejemplo de esto es una cerradura de combinación. Todo el mundo conoce como funciona, pero la clave es secreta. Una longitud de clave de tres dígitos significa que existen 1000 posibilidades, una longitud de clave de seis dígitos implica un millón de posibilidades.

Desde el punto de vista del criptoanalista, el problema del criptoanálisis presenta tres variaciones principales:

1.- Cuando el criptoanalista tiene una cierta cantidad de texto cifrado pero no tiene texto común se enfrenta al problema de sólo texto cifrado

2.- Cuando el criptoanalista tiene texto cifrado y el texto normal correspondiente se enfrenta al problema del texto normal conocido.

3.- cuando el criptoanalista tiene la capacidad de cifrar textos normales que puede escoger se enfrenta al problema de texto normal seleccionado.

Principios criptográficos fundamentales.

Aunque la criptografía es muy variada como veremos a continuación, existen dos principios fundamentales que sostienen la criptografía y que es importante entender.

El primer principio es que todos los mensajes cifrados deben contener redundancia, es decir, información no necesaria para entender el mensaje. Un ejemplo puede dejar claro la necesidad de esto. Considere una compañía de compras por red que tiene 60000 productos. Pensando en la eficiencia, los programadores han decidido que los mensajes de pedidos deben consistir en el nombre del cliente de 16 bytes seguido de un campo de datos de 4 bytes (2 para la cantidad y 2 para el número del producto). Los últimos 4 bytes deben cifrarse usando una clave muy grande conocida solo por el cliente y por la compañía.

Inicialmente esto puede parecer seguro, y lo es, pues los intrusos pasivos no pueden descifrar los mensajes. Sin embargo, también tiene un fallo que lo vuelve inútil. Supóngase que alguien consigue una lista de compradores de productos (los 16 bytes del número de cliente), entonces es fácil trabajar en un programa para generar pedidos ficticios usando nombres reales de los clientes. Dado que no tiene una lista de claves, pone números aleatorios en los últimos 4 bytes y envía cientos de pedidos. Al llegar estos mensajes, la computadora usa el nombre del cliente para localizar la clave y descifrar el mensaje. Para mala suerte, casi todos los mensajes de 4 bytes descifrados son válidos, pues excepto que el pedido sea de cantidad 0 al descifrarlo o corresponda a un producto cuyo código no existe (cuya probabilidad solo es de un 8.5%), el pedido será atendido.

Este problema puede resolverse agregando redundancia a todos los mensajes. Por ejemplo, si se extienden los mensajes de pedido a 12 bytes, de los cuales los 8 primeros deben ser ceros, entonces este ataque ya no funciona, pues la probabilidad de generar un mensaje valido es prácticamente cero. Sin embargo la adición de redundancia simplifica a los criptoanalistas el descifrado de los mensajes, pues ahora un criptoanalista puede saber que ha descifrado correctamente un mensaje al comprobar que los 8 primeros bytes son ceros. Por ello, una cadena aleatoria de palabras sería mejor para incluir en la redundancia.

Otro ejemplo de este primer principio, puede ser el concepto de un CRC (Cyclic Redundant Check), es decir una información redundante puesta al final de un mensaje, evitando al máximo que otros puedan generar información que pueda ser interpretada e introduzca vulnerabilidad al proceso de comunicaciones.

El segundo principio criptográfico es que deben tomarse algunas medidas para evitar que los intrusos activos reproduzcan mensajes viejos. Si no se toman tales medidas, alguien puede conectarse a la línea telefónica de la empresa de pedidos por red y simplemente continuar repitiendo mensajes válidos enviados previamente. Una de tales medidas es la inclusión en cada mensaje de una marca de tiempo válida durante, digamos, 5 minutos. El receptor puede entonces guardar los mensajes unos 5 minutos, para compararlos con los mensajes nuevos que lleguen y filtrar los duplicados. Los mensajes con mayor antigüedad que 5 minutos pueden descartarse, dado que todas las repeticiones enviadas más de 5 minutos después también se rechazarán como demasiado viejas.

A partir de ahora realizaremos un estudio de las técnicas de cifrado, empezando por las técnicas clásicas para pasar a estudiar las técnicas criptográficas modernas.

Criptografía clásica.

Pasemos a analizar los métodos de la criptografía clásica. La criptografía clásica se basa en algoritmos sencillos y claves muy largas para la seguridad. Las técnicas criptográficas clásicas son básicamente dos, el cifrado por sustitución y el cifrado por transposición. Antes de comenzar su exposición, es necesario

exponer el criterio de notación que tomaremos. Tomaremos como texto normal aquel que se encuentra representado por letras minúsculas y como texto cifrado el que se encuentre representado por letras mayúsculas.

Cifrado por sustitución.

El cifrado por sustitución se basa en la sustitución de cada letra o grupo de letras por otra letra o grupo de letras para disfrazarla. Uno de los cifrados por sustitución más antiguos conocidos es el cifrado de Cesar, atribuido al emperador romano Julio Cesar. En este método la letra *a* se convierte en *D*, la *b* en *E*, la *c* en *F*, ... , y *z* se vuelve *C*. Así, el mensaje *ataque* se convierte en *DWDTXH*. Una generalización del cifrado de Cesar permite que el alfabeto de texto cifrado se desplaza k letras en lugar de siempre 3, con lo cual k se convierte en la clave de cifrado.

La siguiente mejora es hacer que cada uno de los símbolos del texto normal, por ejemplo las 26 letras del alfabeto inglés, tengan una correspondencia biunívoca con alguna otra letra. Por ejemplo:

Texto normal: a b c d e f g h i j k l m n o p q r s t u v w x y z
Texto cifrado: Q W E R T Y U I O P A S D F G H J K L Z X C V B N M

Este sistema general se llama sustitución monoalfabética, siendo la clave la cadena de 26 letras correspondiente al alfabeto completo. A primera vista, esto podría parecer un sistema seguro, porque aunque el criptoanalista conoce el sistema general (sustitución letra por letra), no sabe cuál de las $26!=4x10^{26}$ claves posibles se está usando. Sin embargo, si se cuenta con una cantidad pequeña de texto cifrado, el cifrado puede descifrarse fácilmente. El ataque básico aprovecha las propiedades estadísticas de los lenguajes naturales. En inglés, la letra *e* es la más común, seguida de *t*, *o*, *a*, *n*, *i*, etc. Las combinaciones de letras más comunes o digramas son *th*, *in*, *er*, *re* y *an*. Las combinaciones de tres letras más comunes o trigramas son *the*, *ing*, *and* e *ion*.

Un criptoanalista que intenta descifrar una codificación monoalfabética comenzaría por contar la frecuencia relativa de todas las letras del texto cifrado. Entonces podría asignar tentativamente la más común a la letra *e* y la siguiente más común a la letra *t*. Vería entonces los trigramas para encontrar uno común de la forma *tXe*, lo que sugerirá que *X* es *h*. De la misma manera si el patrón *thYt* ocurre con frecuencia, *Y* probablemente representa a la letra *a*. Con esta información puede buscar trigramas frecuentes de la forma *aZW*, que con probabilidad es *and*, y acabar de forma similar descifrando el texto cifrado.

Por ejemplo, intentemos descifrar el siguiente fragmento de un poema de Carrol Lewis:

```
kfd ktbd fzm eubd kfd pzyiom mztx ku kzyg ur bzha kfthcm
ur mfudm zhx mftnm zhx mdzythc pzq ur ezsszcdm zhx gthcm
zhx pfa kfd mdz tm sutythc fuk zhx pfdkfdi ntcm fzld pthcm
sok pztk z stk kfd uamkdim eitdx sdruid pd fzld uoi efzk
rui mubd ur om zid uok ur sidzkf zhx zyy ur om zid rzk
hu foiia mztx kfd ezindhkdi kfda kfzhgdx ftb boef rui kfzk
```

El porcentaje de ocurrencias de letras, digramas, trigramas y palabras en el idioma ingles es:

Letras		Digramas		Trigramas		Palabras	
E	13.05	TH	3.16	THE	4.72	THE	6.42
T	9.02	IN	1.54	ING	1.42	OF	4.02
O	8.21	ER	1.33	AND	1.13	AND	3.15
A	7.81	RE	1.30	ION	1.00	TO	2.36
N	7.28	AN	1.08	ENT	0.98	A	2.09
I	6.77	HE	1.08	FOR	0.76	IN	1.77
R	6.64	AR	1.02	TIO	0.75	THAT	1.25
S	6.46	EN	1.02	ERE	0.69	IS	1.03

H	5.85	TI	1.02	HER	0.68	I	0.94
D	4.11	TE	0.98	ATE	0.66	IT	0.93
L	3.60	AT	0.88	VER	0.63	FOR	0.77
C	2.93	ON	0.84	TER	0.62	AS	0.76
F	2.88	HA	0.84	THA	0.62	WITH	0.76
U	2.77	OU	0.72	ATI	0.59	WAS	0.72
M	2.62	IT	0.71	HAT	0.55	HIS	0.71
P	2.15	ES	0.69	ERS	0.54	HE	0.71
Y	1.51	ST	0.68	HIS	0.52	BE	0.63
W	1.49	OR	0.68	RES	0.50	NOT	0.61
G	1.39	NT	0.67	ILL	0.47	BY	0.57
B	1.28	HI	0.66	ARE	0.46	BUT	0.56
V	1.00	EA	0.64	CON	0.45	HAVE	0.55
K	0.42	VE	0.64	NCE	0.45	YOU	0.55
X	0.30	CO	0.59	ALL	0.44	WHICH	0.53
J	0.23	DE	0.55	EVE	0.44	ARE	0.50
Q	0.14	RA	0.55	ITH	0.44	ON	0.47
Z	0.09	RO	0.55	TED	0.44	OR	0.45

En nuestro texto, el porcentaje de aparición de letras es:

Z	11.03	T	5.88	S	2.57	L	0.74
D	10.29	H	5.51	B	2.21	Q	0.37
K	8.82	X	3.68	E	2.21	V	0.00
F	8.46	R	3.68	Y	2.21	J	0.00
M	7.72	O	2.94	A	1.84	W	0.00
U	6.62	P	2.57	G	1.10		
I	5.88	C	2.57	N	1.10		

Por lo tanto, comparando tenemos que lo más probable es que E se corresponda con la z o con la d y la T con la otra letra o la k o la f.

Observemos la aparición de palabras. La palabra kfd aparece 5 veces y la palabra zhx aparece 6 veces, entonces, lo más probable es que una corresponda a la palabra THE y la otra corresponda a la palabra AND.

Veamos el análisis letra a letra:

Supongamos que kfd corresponde con AND y zhx corresponde con THE, entonces la frecuencia de aparición de las letras en el texto sería:

Letra	**Porcentaje**	**Sustitución**	**Porcentaje**	**Diferencia**
k	8.82	A	7.81	1.01
f	8.46	N	7.28	1.18
d	10.29	D	4.11	6.18
z	11.03	T	9.02	2.01
h	5.51	H	5.85	0.34
x	3.68	E	13.05	9.37

Si suponemos ahora que kfd corresponde con THE y zhx corresponde con AND, entonces la frecuencia de aparición de las letras en el texto sería:

Letra	**Porcentaje**	**Sustitución**	**Porcentaje**	**Diferencia**
k	8.82	T	9.02	0.20
f	8.46	H	5.85	2.61
d	10.29	E	13.05	2.76
z	11.03	A	7.81	3.22
h	5.51	N	7.28	1.77
x	3.68	D	4.11	0.43

Se observa que en la segunda sustitución, la diferencia de porcentajes es mucho menor que en la primera, por lo cual suponemos como correcta la segunda sustitución, quedando el texto como:

THE TtbE HAm eubE THE pAyiom mAtD Tu TAyg ur bANa THtNcm

ur mHuEm AND mHtnm AND mEAytNc pAq ur eAssAcEm AND gtNcm
AND pHa THE mEA tm sutytNc HuT AND pHETHEi ntcm HAlE ptNcm
soT pAtT A stT THE uamTEim eitED sEruiE pE HAlE uoi eHAT
rui mubE ur om AiE uoT ur siEATH AND Ayy ur om AiE rAT
Nu Hoiia mAtD THE eAinENTEi THEa THANgED Htb boeH rui THAT

Si observamos ahora los trigramas, el trigrama ING debe aparecer con frecuencia, por lo cual, observando el texto encontramos 5 ocurrencias de tNc, por lo cual podemos suponer que la t corresponde a la I y la c a la G. Además, en la tabla siguiente puede observarse que la diferencia en porcentaje de aparición es pequeña.

Letra	Porcentaje	Sustitución	Porcentaje	Diferencia
T	5.88	I	6.77	0.89
C	2.57	G	1.39	1.18

Realizando la sustitución:

THE TIbE HAm eubE THE pAyiom mAID Tu TAyg ur bANa THINGm
ur mHuEm AND mHInm AND mEAyING pAq ur eAssAGEm AND gINGm
AND pHa THE mEA Im suIyING HuT AND pHETHEi nIGm HAlE pINGm
soT pAIT A sIT THE uamTEim eiIED sEruiE pE HAlE uoi eHAT
rui mubE ur om AiE uoT ur siEATH AND Ayy ur om AiE rAT
Nu Hoiia mAID THE eAinENTEi THEa THANgED HIb boeH rui THAT

Continuando, la palabra ur aparece 6 veces, si suponemos corresponde a OF, y con la tabla de apariciones de letras, tenemos:

Letra	Porcentaje	Sustitución	Porcentaje	Diferencia
u	6.62	O	8.21	1.59
r	3.58	F	2.88	0.70

Por lo cual, realizando la sustitución:

THE TIbE HAm eObE THE pAyiom mAID TO TAyg OF bANa THINGm
OF mHOEm AND mHInm AND mEAyING pAq OF eAssAGEm AND gINGm
AND pHa THE mEA Im sOIyING HOT AND pHETHEi nIGm HAlE pINGm
soT pAIT A sIT THE OamTEim eiIED sEFOiE pE HAlE Ooi eHAT
FOi mObE OF om AiE OoT OF siEATH AND Ayy OF om AiE FAT
NO Hoiia mAID THE eAinENTEi THEa THANgED HIb boeH FOi THAT

Si realizamos ahora la comparación entre las letras que aparecen con más frecuencia en el texto y que aún no han sido sustituidas (m el 7.72 e i el 5.88) con las de mayor frecuencia no encontradas todavía (R el 6.64 y S el 6.46), observamos que si sustituimos m por S e i por R, obtenemos palabras como HAS, THINGS, SHOES, ARE, con lo cual, realizando la sustitución:

THE TIbE HAS eObE THE pAyRoS SAID TO TAyg OF bANa THINGS
OF SHOES AND SHInS AND SEAyING pAq OF eAssAGES AND gINGS
AND pHa THE SEA IS sOIyING HOT AND pHETHER nIGS HAlE pINGS
soT pAIT A sIT THE OaSTERS eRIED sEFORE pE HAlE OoR eHAT
FOR SObE OF oS ARE OoT OF sREATH AND Ayy OF oS ARE FAT
NO HoRRa SAID THE eARnENTER THEa THANgED HIb boeH FOR THAT

Realizando un análisis palabra a palabra, podemos reconocer algunas (por ejemplo Ayy debe ser ALL, etc.) y procediendo de esta forma, obtenemos el texto descifrado:

THE TIME HAS COME THE WALRUS SAID TO TALK OF MANY THINGS
OF SHOES AND SHIPS AND SEALING WAX OF CABBAGES AND KINGS

AND WHY THE SEA IS BOILING HOT AND WHETHER PIGS HAVE WINGS
BUT WAIT A BIT THE OYSTERS CRIED BEFORE WE HAVE OUR CHAT
FOR SOME OF US ARE OUT OF BREATH AND ALL OF US ARE FAT
NO HURRY SAID THE CARPENTER THEY THANKED HIM MUCH FOR THAT

Otro enfoque posible para descifrar el cifrado por sustitución es adivinar una palabra o frase probable. Por ejemplo, dado el siguiente texto cifrado de una compañía contable (mostrado en bloques de cinco caracteres):

```
CTBMN  BYCTC  BTJDS  QXBNS  GSTJC  BTSWX  CTQTZ  CQVUJ
QJSGS  TJQZZ  MNQJS  VLNSX  VSZJU  JDSTS  JQUUS  JUBXJ
DSKSU  JSNTK  BGAQJ  ZBGYQ  TLCTZ  BNYBN  QJSW
```

Una palabra muy probable en un mensaje de una compañía contable en inglés es *financial*. Usando nuestro conocimiento de que *financial* tiene la letra i repetida, con cuatro letras intermedias entre su aparición, buscamos letras repetidas en el texto cifrado con este espaciado. Encontramos 12 casos, en las posiciones 6, 15, 27, 31, 42, 48, 56, 66, 70, 71, 76 y 82. Sin embargo, sólo dos de éstos, el 31 y el 42, tiene la siguiente letra (correspondiente a *n* en el texto normal) repetida en el lugar adecuado. De estos dos, sólo el 31 tiene también la *a* en la posición correcta, por lo que sabemos que *financial* comienza en la posición 30. A partir de aquí, la deducción de la clave es fácil usando las estadísticas de frecuencia del texto en inglés.

Otros métodos de cifrado por sustitución son:

- Cifrado de **Polybius**, se introduce el alfabeto dentro de una tabla por filas, donde la table tiene un número de columnas determinado, este número de columnas es la clave conociendo el algoritmo. El texto normal se codifica en base a las coordenadas de las letras del texto normal dentro de dicha tabla. La clave de este cifrado está en la disposición del alfabeto en la tabla, es decir el número de columnas de la tabla.

	1	**2**	**3**	**4**	**5**	**6**	**7**
1	A	B	C	D	E	F	g
2	H	I	J	K	L	M	N
3	Ñ	O	P	Q	R	S	t
4	U	V	W	X	Y	Z	+

Si P=HOLA, entonces el cifrado consiste en codificar *(fila, columna)* por tanto el cifrado es (2,1), (3,2), (2,5), (1,1)

- Cifrado de **Trithemius,** es un método de sustitución progresivo, basado en el cifrado de Julio Cesar, donde el valor de k varía incrementalmente de forma conocida en los extremos. Ejemplo: clave k=+2 y texto normal P="Hola" *=>H(+2)=J, o(+3)=r, l(+4)=o, a(+5)=f:: p*or tanto el texto cifrado sería *C="Jrof"*

Cifrado por transposición.

Los cifrados por sustitución conservan el orden de los símbolos de texto normal, pero los disfrazan. Los cifrados por transposición en contraste, reordenan las letras pero no las disfrazan. Un ejemplo de cifrado por transposición es la transposición columnar. La clave del cifrado es una palabra o frase que no contiene letras repetidas. En este ejemplo, la clave es MEGABUCK. El propósito de la clave es numerar las columnas, estando la columna 1 bajo la letra clave más cercana al inicio del alfabeto, y así sucesivamente. El texto normal se escribe horizontalmente en filas, el texto cifrado se lee por columnas, comenzando por la columna cuya letra clave es la más baja.

```
M E G A B U C K
7 4 5 1 2 8 3 6
p l e a s e t r
a n s f e r o n
```

Texto normal:

```
pleasetransferonemilliondollarstom
```

```
e m i l l i o n
d o l l a r s t
o m y s w i s s
b a n k a c c o
u n t s i x t w
o t w o a b c d
```

```
yswissbankaccountsixtwotwo
```

Texto cifrado:

```
AFLLSKSOSELAWAIATOOSSCTCLNMOMANTES
ILYNTWRNNTSOWDPAEDOBUOERIRICXB
```

Para descifrar un cifrado por transposición, el criptoanalista debe primero ser consciente de que está tratando con un cifrado por transposición. Observando la frecuencia de *E*, *T*, *A*, *O*, *I*, *N*, etc., es fácil ver si se ajustan al patrón usual del texto normal. De ser así, es evidente que se trata de un cifrado por transposición, pues en tal cifrado cada letra se representa a sí misma.

El siguiente paso es adivinar la cantidad de columnas. En muchos casos, puede adivinarse una palabra o frase probable por el contexto del mensaje. Por ejemplo, supóngase que nuestro criptoanalista sospecha que la frase de texto normal *milliondolars* aparece en algún lugar del mensaje. Observe que los diagramas *MO*, *IL*, *LL*, *LA*, *IR* y *OS* ocurren en el texto cifrado como resultado de que esta frase da la vuelta. Si se hubiera usado una clave de longitud siete, habrían ocurrido los diagramas *MD*, *IO*, *LL*, *LL*, *IA*, *OR* y *NS*. De hecho, para cada longitud de clave, se produce un grupo diferente de diagramas de texto cifrado. Buscando las diferentes posibilidades, el criptoanalista con frecuencia puede determinar fácilmente la longitud de la clave.

El paso restante es ordenar las columnas. Cuando la cantidad de columnas, k, es pequeña, puede examinarse cada uno de los pares de columnas k(k-1) para ver si la frecuencia de sus diagramas es igual a la del texto normal. El par con mejor concordancia se supone correctamente ubicado. Ahora cada columna restante se prueba tentativamente como el sucesor de este par. La columna cuyas frecuencias de digramas y trigramas produce la mejor concordancia se toma tentativamente como correcta. La columna antecesora se encuentra de la misma manera. El proceso completo se repite hasta encontrar un orden potencial. Es probable que el texto normal sea reconocible en algún punto (por ejemplo, sí aparece *milloin*, quedará claro dónde está el error).

Por ejemplo, intentemos descifrar el siguiente fragmento de un libro de texto sobre computación, por lo que "computer" es una palabra de probable aparición. El texto normal consiste por entero en letras (sin espacios). El texto cifrado se dividió en bloques de cinco caracteres para hacerlo más legible.

```
aauan cvlre rurnn dltme aeepb ytust iceat npmey iicgo
gorch srsoc nntii imiha oofpa gsivt tpsit lbolr otoex
```

Si la palabra “computer” aparece en el texto y la transposición columnar es de clave menor que 8, entonces, según el caso deberían aparecer juntas las siguientes letras de la palabra “computer” según el tamaño de la clave

Tamaño	Letras	Tamaño	Letras	Tamaño	Letras
2	CM	3	CP	4	CU
5	CT	6	CE	7	CR

Observando el texto vemos que aparece CE, por lo cual el tamaño de la clave es de tamaño 6. Ordenando el texto en seis columnas, obtenemos:

```
ADIGIT
ALCOMP
UTERIS
AMACHI
NETHAT
CANSOL
VEPROB
LEMSFO
RPEOPL
EBYCAR
```

```
RYINGO
UTINST
RUCTIO
NSGIVE
NTOITX
```

Que directamente puede leerse, sin necesidad de realizar ninguna transposición columnar como:

```
A DIGITAL COMPUTER IS A MACHINE THAT CAN SOLVE PROBLEMS FOR
PEOPLE BY CARRYING OUT INSTRUCTIONS GIVE TO IT
```

Rellenos de una sola vez.

La construcción de un cifrado inviolable en realidad es bastante sencilla. La técnica se conoce desde hace décadas y consiste en escoger una cadena de bits al azar como clave. Luego se convierte el texto normal en una cadena de bits, por ejemplo usando su representación ASCII. Por último, se calcula el or exclusivo, conocido como XOR y cuya tabla de valores lógicos puede verse a continuación.

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

El texto cifrado resultante no puede descifrarse porque cada texto normal posible es un candidato igualmente probable. El texto cifrado no proporciona al criptoanalista ninguna información en absoluto. En una muestra suficientemente grande de texto cifrado, cada letra ocurrirá con la misma frecuencia, al igual que cada digrama (combinación de dos letras) y cada trigrama (combinación de tres letras). Como ejemplo de cifrado basado en relleno de una sóla vez, cifremos el mensaje "texto cifrado" con la cadena "En un lugar de la Mancha de cuyo nombre…"

Texto original	t	e	x	t	o		c	i	f	r	a	d	o
Codificación ASCII (hex)	74	65	78	74	6F	20	63	69	66	72	61	64	6F
Texto de cifrado	E	n		u	n		l	u	g	a	r		d
Codificación ASCII (hex)	45	6E	20	75	6E	20	6C	75	67	61	72	20	64
Codificación cifrada (hex)	*31*	*0B*	*58*	*01*	*01*	*00*	*0F*	*1C*	*01*	*13*	*13*	*44*	*08*

Si procedemos ahora a descifrarlo con la clave de codificación, obtenemos el mensaje original ya que aplicamos la función XOR 2 veces:

Codificación cifrada (hex)	*31*	*0B*	*58*	*01*	*01*	*00*	*0F*	*1C*	*01*	*13*	*13*	*44*	*08*
Texto de cifrado	E	n		u	n		l	u	g	a	r		d
Codificación ASCII (hex)	45	6E	20	75	6E	20	6C	75	67	61	72	20	64
Codificación ASCII (hex)	74	65	78	74	6F	20	63	69	66	72	61	64	6F
Texto original	t	e	x	t	o		c	i	f	r	a	d	o

Sin embargo, este método tiene varias desventajas prácticas. En primer lugar, la clave no puede memorizarse, por lo que tanto el transmisor como el receptor deben llevar una copia por escrito consigo. Además, la cantidad total de datos que pueden transmitirse está limitada a la cantidad de clave disponible. Otro problema es la sensibilidad del método a la pérdida o inserción de caracteres. Si el transmisor y el receptor pierden la sincronía, todos los datos a partir de ahí aparecerán alterados.

Criptografía moderna.

La criptografía moderna usa las mismas ideas básicas que la criptografía tradicional, la transposición y la sustitución, pero su orientación es distinta. Mientras la criptografía tradicional usaba algoritmos sencillos y claves muy largas para la seguridad, hoy en día es cierto la afirmación contraria: el objetivo es hacer algoritmos de cifrado tan complicados y rebuscados que incluso si el criptoanalista obtiene cantidades enormes de texto cifrado a su gusto, no será capaz de entender nada y por tanto de descifrarlo.

Las transposiciones y sustituciones pueden implantarse mediante circuitos sencillos. En la figura siguiente se muestran dos dispositivos conocidos como caja P, que se usa para efectuar una transposición de una entrada de 12 bits; y otro dispositivo conocido como caja S, en el cual ingresa un texto normal de 3 bits y sale un texto cifrado de 3 bits. La potencia real de estos elementos básicos sólo se hace aparente cuando ponemos en cascada una serie completa de estas cajas para formar un cifrado de producto como podemos ver en la figura siguiente.

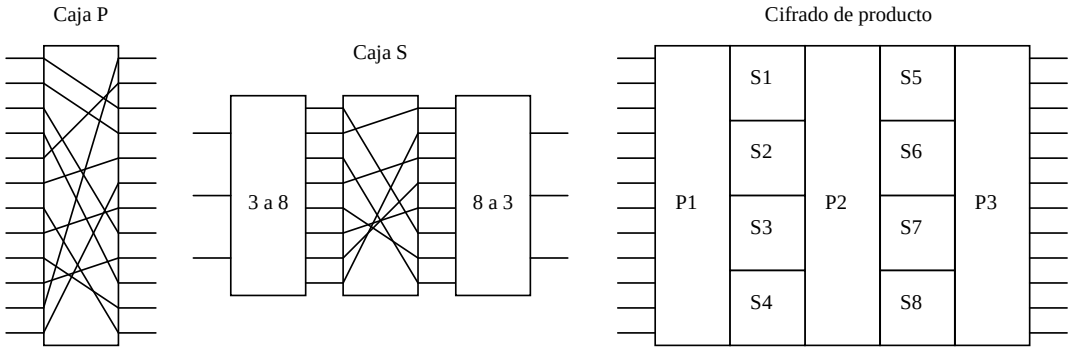


Figura 2: Ejemplo de cifrado de producto mediante cajas P y cajas S.

El cifrado moderno se divide actualmente en cifrado de clave privada y cifrado de clave pública:

- en el cifrado de clave privada las claves de cifrado y descifrado son la misma (o bien se deriva de forma directa una de la otra), debiendo mantenerse en secreto dicha clave. **Ejemplo**: DES (Data Encryption Standar), DES triple e IDEA (International Data Encryption Algorithm). El cifrado de clave privada, es más rápido que el de clave pública (de 100 a 1000 veces), y por tanto se utiliza generalmente en el *intercambio de información dentro de una sesión*. Estas claves también son conocidas como claves de sesión o de cifrado simétricas, ya que en ambos extremos se posee la misma clave.

- en el cifrado de clave pública, las claves de cifrado y descifrado son independientes, no derivándose una de la otra, por lo cual puede hacerse pública la clave de cifrado siempre que se mantenga en secreto la clave de descifrado. **Ejemplo**: Cifrado RSA (Rivest, Shamir, Adleman).El cifrado de clave pública es más lento y por tanto *se utiliza para intercambiar las claves de sesión*. Como este algoritmo utiliza dos claves diferentes, una privada y otra pública el cifrado se conoce como cifrado asimétrico

Veremos a continuación algunos algoritmos de cifrado modernos.

Cifrado con clave privada (simétrica)

Los métodos de cifrado de clave privada, también son conocidos como de clave simétrica, porque ambos extremos conocen la clave para poder descifrar.

Cifrado DES (Data Encryption Standar).

El algoritmo DES es un algoritmo de cifrado de clave privada desarrollado por IBM a principios de la década de los 70 a partir de otro algoritmo conocido como Lucifer que utilizaba claves de 112 bits y que

fueron reducidas a 56 bits en el algoritmo DES. La reducción del tamaño de las claves, propuesta por la NSA (Agencia Nacional de Seguridad) originó controversia debido a que sé penso que la NSA había debilitado intencionadamente el algoritmo del DES para poder desencriptarlo.

Dicha controversia llegó a su fin cuando en 1994 IBM publicó un artículo describiendo los criterios del desarrollo del algoritmo DES. El artículo indica que el diseño se realizó de forma que el algoritmo DES fuera resistente a criptoanálisis, pero lo suficientemente sencillo como para poder ser implementado en un circuito electrónico con la tecnología de principios de los 70. Por tanto, el algoritmo DES se diseño de forma que no pudiera ser descifrado por criptoanálisis, pero si que puede ser descifrado probando todas las claves posibles, asumiendo que se cuenta con el hardware adecuado.

En la siguiente figura se muestra un esbozo del algoritmo DES. El texto normal se cifra en bloques de 64 bits, produciendo 64 bits de texto cifrado. El algoritmo, que se parametriza con la clave de 56 bits, tiene 19 etapas diferentes.

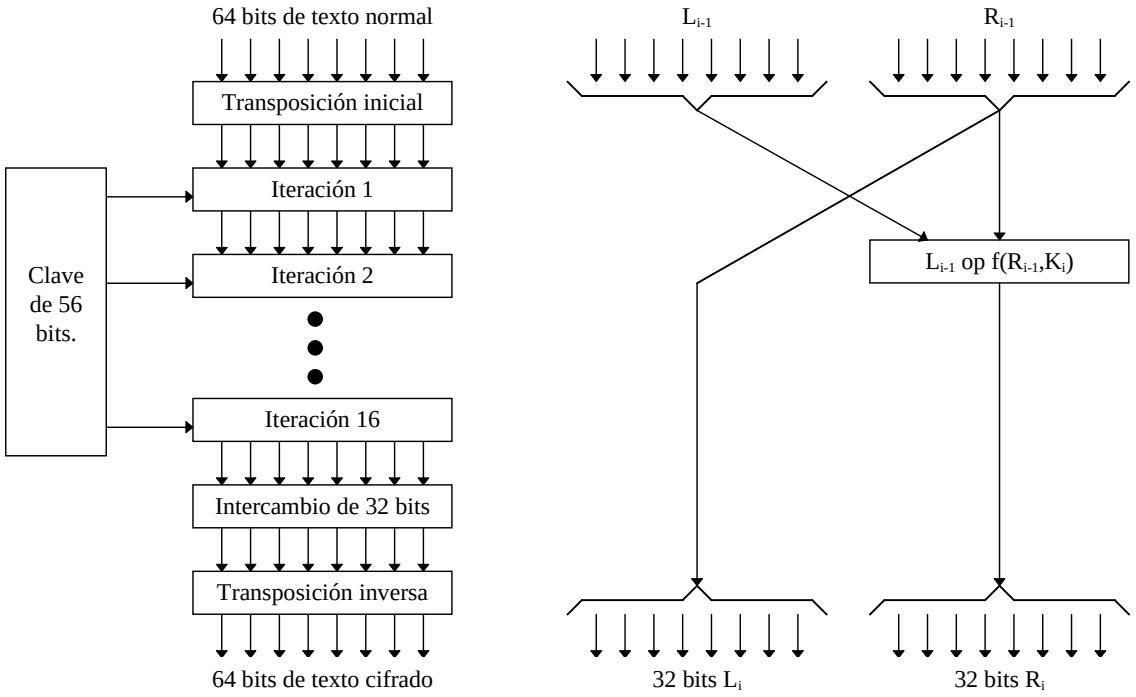


Figura 3: Algoritmo del cifrado DES.

La primera etapa es una transposición, independiente de la clave, del texto normal de 64 bits. La última etapa es el inverso exacto de esta transposición. La etapa previa a la última intercambia los 32 bits de la izquierda y los 32 bits de la derecha. Las 16 etapas restantes son funcionalmente idénticas, pero se parametrizan mediante diferentes funciones de la clave. El algoritmo se ha diseñado para permitir que el descifrado se haga con la misma clave que el cifrado, simplemente ejecutando los pasos en orden inverso. Cada una de las 16 etapas intermedias toma dos entradas de 32 bits y produce dos salidas de 32 bits. La salida de la izquierda es simplemente una copia de la entrada de la derecha. La salida de la derecha es el or exclusivo a nivel de bit de la entrada izquierda y una función de la entrada derecha y la clave de esta etapa K_i. Toda la complejidad reside en esta función.

La función consiste en cuatro pasos, ejecutados en secuencia. Primero se construye un número de 48 bits, E, expandiendo el R_{i-1} de 32 bits según una regla fija de transposición y duplicación. Después, se aplica un or exclusivo a E y K_i. Esta salida entonces se divide en ocho grupos de 6 bits, cada uno de los cuales se alimenta a una caja S distinta. Cada una de las 64 entradas posibles a la caja S se transforma en una salida de 4 bits. Por último estos 8x4 bits se pasan a través de una caja P.

En cada una de las 16 iteraciones, se usa una clave diferente. Antes de iniciarse el algoritmo, se aplica una transposición de 56 bits a la clave. Justo antes de cada iteración, la clave se divide en dos unidades de 28

bits, cada una de las cuales se gira hacia la izquierda una cantidad de bits dependiente del número de iteración. K_i se deriva de esta clave girada aplicándole otra transposición de 56 bits. En cada vuelta (o iteración) se extrae y permuta de los 56 bits un subgrupo de 48 bits diferente.

Descifrado del DES.

Aunque en un principio el DES parecía inviolable, dos investigadores de criptografía de Standford diseñaron en 1977 una máquina para violar el DES y estimaron que podría construirse por unos 20 millones de dólares. Dado un trozo pequeño de texto normal y el texto cifrado correspondiente, esta máquina podría encontrar la clave mediante una búsqueda exhaustiva del espacio de claves de 2^{56} en menos de un día. Otros autores han diseñado máquinas capaces de descifrar el DES mediante búsqueda exhaustiva en unas cuatro horas.

Otra estrategia similar es descifrar el DES por software. Aunque el cifrado es 1.000 veces más lento por software que por hardware, una computadora casera de alto nivel aún puede efectuar unos 250.000 cifrados/segundo en software, y es probable que este ociosa unos 2 millones de segundos al mes. Este tiempo de inactividad podría usarse para descifrar el DES. Si alguien pusiera un mensaje en uno de los grupos de noticias de Internet, no debería ser difícil conseguir las 140.000 personas necesarias para revisar las $7x10^{16}$ claves en un mes.

Probablemente la idea más innovadora para descifrar el DES es la lotería china. En este diseño, cada radio y televisión tiene que equiparse con un chip DES barato capaz de realizar 1 millón de cifrados/segundo en hardware. Suponiendo que cada una de los 1200 millones de personas de China tiene una radio o televisor, cada vez que el gobierno quiera descifrar un mensaje codificado DES, simplemente difunde el par texto normal/texto cifrado y cada uno de los 1200 millones de chips comienzan a buscar en su sección preasignada del espacio de claves. En 60 segundos se encontrarán una o más correspondencias. Para asegurar que se informe, los chips podrían programarse para anunciar un mensaje de que se ha ganado un premio informado a la autoridad competente.

La conclusión que se puede obtener de estos argumentos es que el DES no es seguro. Sin embargo, surge la idea de ejecutar el DES dos veces, con dos claves de 56 bits distintas. Esto proporciona un espacio de 2^{112}, esto es, $5x10^{33}$, lo cual provocaría que el ataque de la lotería china requiriese 100 millones de años. Sin embargo dos investigadores han desarrollado un método que hace sospechoso al doble cifrado. El método de ataque se llama encuentro a la mitad y funciona como sigue. Supóngase que alguien ha cifrado doblemente una serie de bloques de texto normal usando el modo de libro de código electrónico. Para unos pocos valores de i, el criptoanalista tiene pares igualados (P_i,C_i) donde:

$$C_i=E_{k2}(E_{k1}(P_i))$$

Si ahora aplicamos la función de descifrado D_{k2} a cada lado de la ecuación, obtenemos:

$$D_{k2}(C_i)=E_{k1}(P_i)$$

Porque el cifrado de x y su descifrado posterior con la misma clave produce x.

El ataque de encuentro a la mitad usa esta ecuación para encontrar las claves DES, K_1 y K_2, como sigue:

1. Calcular $R_i=E(P_1)$ para los 2^{56} valores de i, donde E es la función de cifrado DES. Ordenar esta tabla en orden ascendente según R_i.

2. Calcular $S_j=Dj(C_1)$ para todos los 2^{56} valores de j, donde D es la función de descifrado DES. Ordenar esta tabla en orden ascendente según S_j.

3. Barrer la primera tabla en busca de un R_i igual a algún S_j de la segunda tabla. Al encontrar un par, tenemos un par de claves (i,j) tal que $D_j(C_1)=E_i(P_1)$. Potencialmente i es K1 y j es K2.

4. Comprobar si $E_j(E_i(P_2))$ es igual a C_2. Si lo es, intentar todos los demás pares (texto normal, texto cifrado). De no serlo, continuar buscando pares en las dos tablas.

Ciertamente ocurrirán muchas falsas alarmas antes de encontrar las claves reales, pero tarde o temprano se encontrarán. Este ataque requiere sólo 2^{57} operaciones de cifrado o descifrado (para construir las dos tablas), solamente el doble que 2^{56}. Sin embargo requiere un total de 2^{60} bytes de almacenamiento para las dos tablas, por lo que actualmente no es factible en su forma básica, aunque se han desarrollado varias optimizaciones y concesiones que permiten menos almacenamiento a expensas de más computo. En conclusión, el cifrado doble usando el DES tampoco es mucho más seguro que el cifrado sencillo.

Cifrado DES triple.

Como hemos visto, el cifrado realizado con el algoritmo DES es descifrable mediante diversos tipo de ataques de fuerza bruta, hecho que corroboro IBM en su artículo de 1994, indicando de forma explícita que la NSA decidió que así fuera con el fin de poder desencriptar los mensajes que deseara.

Parece, por tanto, que el cifrado con el algoritmo DES no es seguro, sin embargo, el cifrado triple con el algoritmo DES es otro asunto. El método seleccionado, que se ha incorporado al estándar internacional 8732, se ilustra en la siguiente figura.

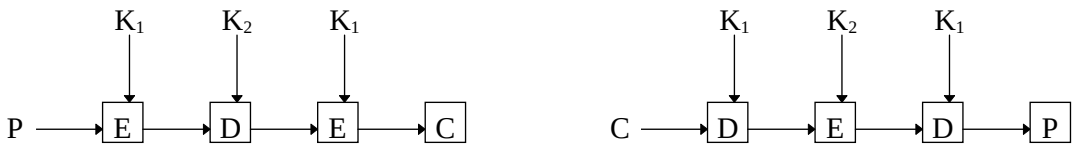


Figura 4: Esquema del cifrado DES triple.

Aquí se usan dos claves y tres etapas. En la primera etapa el texto normal se cifra con K_1. En la segunda etapa el algoritmo DES se ejecuta en modo de descifrado, usando K_2 como clave. Por último, se hace otro cifrado usando K_1. El hecho de que se usen dos claves y en modo EDE (cifrado-descifrado-cifrado) en lugar de EEE (cifrado-cifrado-cifrado) es debido a dos motivos:

1. En primer lugar, los criptógrafos admiten que 112 bits son suficientes para las aplicaciones comerciales por ahora. Subir a 168 bits (3 claves) simplemente agregaría carga extra innecesaria de administrar y transportar otra clave.

2. En segundo lugar, la razón de cifrar, descifrar y luego cifrar de nuevo es la compatibilidad con los sistemas DES de una sola clave. Tanto las funciones de cifrado como de descifrado son correspondencias entre números de 64 bits. Desde el punto de vista criptográfico, las dos correspondencias son igualmente robustas. Sin embargo, usando EDE en lugar de EEE, una computadora que usa cifrado triple puede hablar con otra que usa cifrado sencillo simplemente estableciendo $K_1=K_2$. Esta propiedad permite la introducción gradual del cifrado triple.

Cifrado IDEA (International Data Encryption Algorithm).

Después de comprobar la debilidad del algoritmo DES en su forma simple, diversos trabajos propusieron nuevos métodos de cifrados de bloques (BLOWFISH, Crab, FEAL, KHAFRE, LOKI91, NEWDES, REDOCII, SAFER K64). Sin embargo el más interesante e importante de los cifrados posteriores al algoritmo DES es el algoritmo IDEA, algoritmo internacional de cifrado de datos.

El algoritmo IDEA es un algoritmo de clave privada que fue diseñado por dos investigadores en Suiza, usa una clave de 128 bits, lo que lo hará inmune durante décadas a los ataques de la fuerza bruta, la lotería china y a los ataques de encuentro a la mitad. No hay ninguna técnica o máquina conocida actualmente que se crea que puede descifrar el algoritmo IDEA.

La estructura básica del algoritmo se asemeja al algoritmo DES en cuanto a que se alteran bloques de entrada de texto normal de 64 bits en una secuencia de iteraciones parametrizadas para producir bloques de salida de texto cifrado de 64 bits, como se puede ver en la figura siguiente.

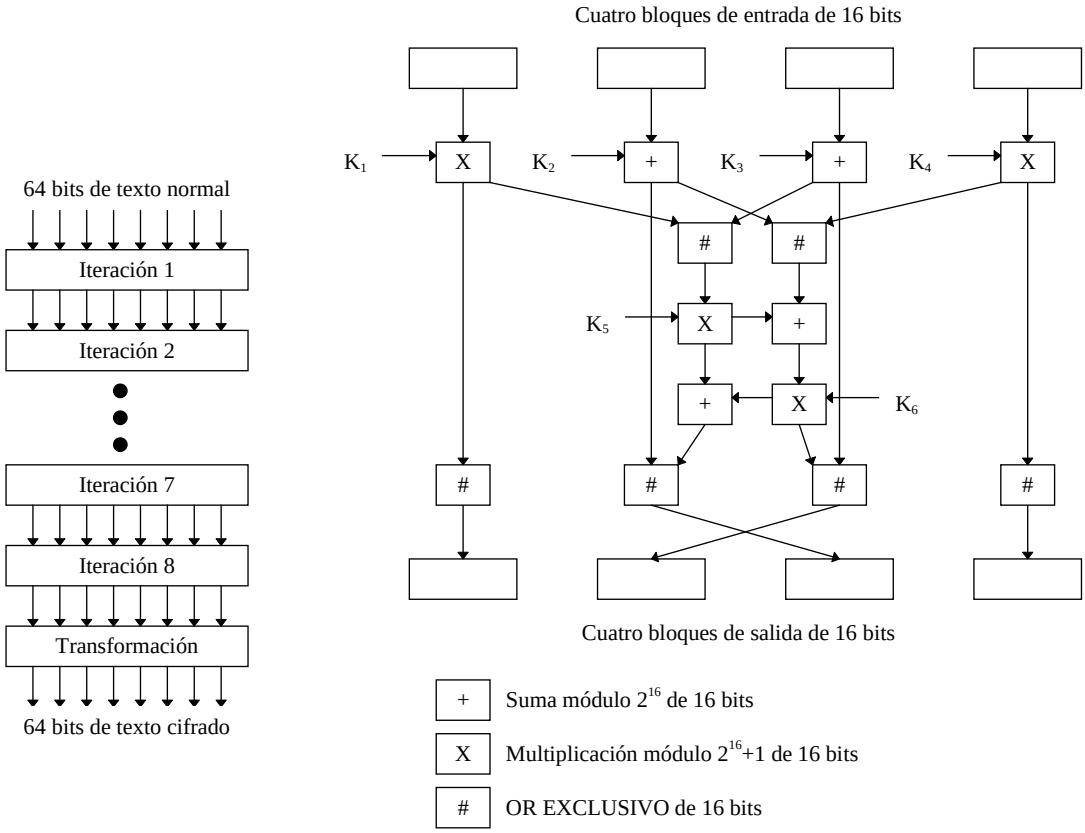


Figura 5: Algoritmo del cifrado IDEA.

Dada la extensa alteración de bits (por cada iteración, cada uno de los bits de salida depende de cada uno de los bits de entrada), basta con ocho iteraciones. Como con todos los cifrados de bloque, el algoritmo IDEA también puede usarse en el modo de realimentación de cifrado y en los demás modos del algoritmo DES. El algoritmo IDEA usa tres operaciones, todas sobre números sin signo de 16 bits. Estas operaciones son un or exclusivo, suma módulo 2^{16} y multiplicación módulo $2^{16}+1$. Las tres operaciones se pueden efectuar fácilmente en una microcomputadora de 16 bits ignorando las partes de orden mayor de los resultados. Las operaciones tienen la propiedad de que ningunos dos pares obedecen la ley asociativa ni la ley distributiva, dificultando el criptoanálisis. La clave de 128 bits se usa para generar 52 subclaves de 16 bits cada una, 6 por cada una de las ocho iteraciones y 4 para la transformación final. El descifrado usa el mismo algoritmo que el cifrado, sólo que con subclaves diferentes.

Cifrado AES (Advanced Encryption Estándar) o Rijndael.

Es considerado el sucesor de DES. Este algoritmo se adoptó oficialmente en octubre del 2000 como nuevo Estándar Avanzado de Cifrado (AES) por el NIST (National Institute for Standards and Technology) para su empleo en aplicaciones criptográficas.

Sus autores son dos, los belgas Joan Daemen y Vincent Rijmen, de ahí su nombre Rijndael. Tiene como peculiaridad que todo el proceso de selección, revisión y estudio tanto de este algoritmo como de los restantes candidatos, se efectuó de forma pública y abierta, por lo que, toda la comunidad criptográfica mundial ha participado en su análisis, lo cual convierte a Rijndael en un algoritmo perfectamente digno de la confianza de todos.

AES es un sistema de cifrado por bloques, diseñado para manejar longitudes de clave y de bloque variables, ambas comprendidas entre los 128 y los 256 bits.

AES es un algoritmo que se basa en aplicar un número determinado de rondas a un valor intermedio que se denomina estado. Dicho estado puede representarse mediante una matriz rectangular de bytes, que posee cuatro filas, y Nb columnas. Así, por ejemplo, si nuestro bloque tiene 160 bits (tabla 5), Nb será igual a 5.

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$

Tabla 5: Ejemplo de matriz de estado con N_b=5 (160 bits).

La clave tiene una estructura análoga a la del estado, y se representará mediante una tabla con cuatro filas y N_k columnas. Si nuestra clave tiene, por ejemplo, 128 bits, N_k será igual a 4 (tabla 6).

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$

Tabla 6: Ejemplo de clave con N_k=4 (128 bits).

El bloque que se pretende cifrar o descifrar se traslada directamente byte a byte sobre la matriz de estado, siguiendo la secuencia $a_{0,0}$, $a_{1,0}$, $a_{2,0}$, $a_{3,0}$, $a_{0,1}$…, y análogamente, los bytes de la clave se copian sobre la matriz de clave en el mismo orden, a saber, $k_{0,0}$, $k_{1,0}$, $k_{2,0}$, $k_{3,0}$, $k_{0,1}$….

Siendo *B* el bloque que queremos cifrar, y *S* la matriz de estado, el algoritmo AES con *n* rondas queda como sigue:

1. Calcular K_0,K_1, …K_n subclaves a partir de la clave K.
2. $S \leftarrow B \oplus K_0$
3. Para i = 1 hasta n hacer
4. Aplicar ronda i-ésima del algoritmo con la subclave Ki.

El *algoritmo de descifrado* consistirá en aplicar las inversas de cada una de las funciones en el orden contrario, y utilizar los mismos Ki que en el cifrado, sólo que comenzando por el último.

Es un algoritmo resistente al criptoanálisis tanto lineal como diferencial y uno de los más seguros en la actualidad ya que para recuperar la clave a partir de un par texto cifrado-texto claro hay que realizar una búsqueda exhaustiva.

Las Rondas de AES

Puesto que AES permite emplear diferentes longitudes tanto de bloque como de clave, el número de rondas requerido en cada caso es variable. En la tabla 7 se especifica cuantas rondas son necesarias en función de Nb y Nk.

	N_b=4 (128 bits)	N_b=6 (192 bits)	N_b=8 (256 bits)
N_k=4 (128 bits)	10	12	14
N_k=6 (192 bits)	12	12	14
N_k=8 (256 bits)	14	14	14

Tabla 7: Número de rondas para AES en función de los tamaños de clave y bloque.

Siendo S la matriz de estado, y K_i la subclave correspondiente a la ronda i-ésima, cada una de las rondas posee la siguiente estructura:

1. S ← ByteSub(S)

2. S ← DesplazarFila(S)

3. S ← MezclarColumnas(S)

4. S ← Ki ⊕ S

Estas cuatro funciones están diseñadas para proporcionar resistencia frente a criptoanálisis lineal y diferencial. Cada una de las funciones tiene un propósito:

- Funciones DesplazarFila y MezclarColumnas permiten obtener un alto nivel de difusión a lo largo de varias rondas.
- Función ByteSub consiste en la aplicación paralela de s-cajas.
- La capa de adición de clave es un simple or-exclusivo entre el estado intermedio y la subclave correspondiente a cada ronda.

La última ronda es igual a las anteriores, pero eliminando el paso 3.

Función ByteSub

La transformación ByteSub es una sustitución no lineal que se aplica a cada byte de la matriz de estado, mediante una s-caja 8*8 invertible, que se obtiene componiendo dos transformaciones:

1. Cada byte (2^8) genera el polinomio irreducible $m(x) = x^8 + x^4 + x^3 + x + 1$, y sustituido por su inversa multiplicativa. El valor cero queda inalterado.

2. Después se aplica la siguiente transformación afín en GF(2), siendo $x_0, x_1, \ldots, x_7$ los bits del byte correspondiente, e $y_0, y_1, \ldots, y_7$ los del resultado:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

La función inversa de ByteSub sería la aplicación de la inversa de la s-caja correspondiente a cada byte de la matriz de estado.

Función DesplazarFila

Esta transformación consiste en desplazar a la izquierda cíclicamente las filas de la matriz de estado. Cada fila f_i se desplaza un número de posiciones c_i diferente. Mientras que c_0 siempre es igual a cero (esta fila siempre permanece inalterada), el resto de valores viene en función de N_b y se refleja en la tabla 8.

N_b	c_1	c_2	c_3
4	1	2	3

6	1	2	3
8	1	3	4

Tabla 8: Valores de c_i según el tamaño de bloque N_b

La función inversa de DesplazarFila será, obviamente, un desplazamiento de las filas de la matriz de estado el mismo número de posiciones que en la tabla 2.5, pero a la derecha.

Función MezclarColumnas

Para esta función, cada columna del vector de estado se considera un polinomio cuyos coeficientes pertenecen a GF(28) y se multiplica módulo $x^4 + 1$ por:

$$c(x) = 03x^4 + 01x^2 + 01x + 02$$

donde 03 es el valor hexadecimal que se obtiene concatenando los coeficientes binarios del polinomio correspondiente en GF(28), en este caso 00000011, o sea, x+1, y así sucesivamente.

La inversa de MezclarColumnas se obtiene multiplicando cada columna de la matriz de estado por el polinomio:

$$d(x) = 0Bx^4 + 0Dx^2 + 09x + 0E$$

Cálculo de las Subclaves

Las diferentes subclaves Ki se derivan de la clave principal K mediante el uso de dos funciones: una de expansión y otra de selección. Siendo n el número de rondas que se van a aplicar, la función de expansión permite obtener, a partir del valor de K, una secuencia de $4*(n+1)*N_b$ bytes. La selección simplemente toma consecutivamente de la secuencia obtenida bloques del mismo tamaño que la matriz de estado, y los va asignando a cada Ki.

Sea K(i) un vector de bytes de tamaño $4*N_k$, conteniendo la clave, y sea W(i) un vector de $N_b*(n+1)$ registros de 4 bytes, siendo n el número de rondas. La función de expansión tiene dos versiones, según el valor de N_k:

a) Si Nk ≤ 6:

1. Para i desde 0 hasta N_k - 1 hacer
2. W(i) ← (K(4 * i), K(4 * i + 1), K(4 * i + 2), K(4 * i + 3))
3. Para i desde Nk hasta Nb * (n + 1) hacer
4. tmp ← W(i - 1)
5. Si i mod Nk = 0
6. tmp ← Sub(Rot(tmp)) ⊕ R(i/Nk)
7. W(i) ← W(i - Nk) ⊕ tmp

b) Si Nk > 6:

1. Para i desde 0 hasta Nk - 1 hacer
2. W(i) ← (K(4 * i), K(4 * i + 1), K(4 * i + 2), K(4 * i + 3))
3. Para i desde Nk hasta Nb * (n + 1) hacer
4. tmp ← W(i - 1)
5. Si i mod Nk = 0
6. tmp ←Sub(Rot(tmp)) ⊕ $R_c(i/N_k)$
7. Si i mod Nk = 4
8. tmp ← Sub(tmp)
9. W(i) ← W(i - N_k) ⊕ tmp

En los algoritmos anteriores, la función Sub devuelve el resultado de aplicar la s-caja de AES a cada uno de los bytes del registro de cuatro que se le pasa como parámetro. La función Rot desplaza a la izquierda una posición los bytes del registro, de tal forma que si le pasamos como parámetro el valor (a, b, c, d) nos devuelve (b, c, d, a). Finalmente, $R_c(j)$ es una constante definida de la siguiente forma:

- $R_c(j) = (R(j), 0, 0, 0)$

- Cada R(i) es el elemento de $GF(2^8)$ correspondiente al valor $x^{(i-1)}$.

Otros cifrados simétricos: métodos de bloque y flujo
En general, estos métodos simétricos tienen el inconveniente de la distribución de claves. La ventaja es que son rápidos.

Otro problema que aparece, es el procesamiento de la información por bloques de un tamaño conocido. Por ejemplo, DES procesa a bloques de 64 bits (8 bytes), lo cual con un intruso avispado, podría hacer movimiento de bloques dentro del mismo cifrado, sin levantar alarma. Otros cifrados de bloque que también sufren este problema son RC5, Blowfish

El problema de trabajar con bloques de 64 bytes se resuelve por **diferentes métodos**:
•*basados en **flujo**, que operan por bloques, pero convolucionando (por ejemplo con una XOR) la salida actual con salidas anteriores. Ejemplos: RC2, RC4 y CAST.*
•*con **rellenos variables** por bloque, por ejemplo insertando 0s e indicando la cantidad de rellenos*
•*con **algoritmos de bloque y clave variable** como el AES (Advanced Encryption Standard)*

Circuitos y hardware asociado.

En la actualidad existe hardware utilizado para cifrar comunicaciones a nivel de red y/o enlace. Ejemplo de ello son las Tarjetas Advancde Integration Module (AIM) para diferente gama de routers Cisco.

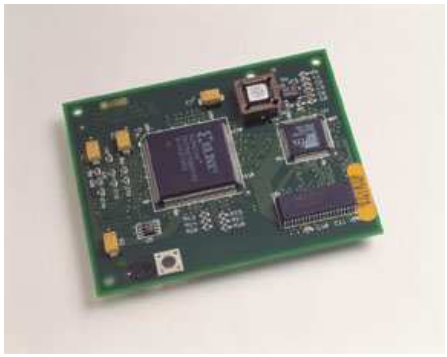


Figura 6: Tarjeta 2600 AIM 2 Mbps

Cifrado con clave pública (asimétrica)

Los métodos de cifrado de clave pública, también son conocidos como de clave asimétrica, porque son algoritmos que se basan en un par de claves, una pública que dispone todo el mundo y una clave asociada a dicha clave pública, que es privada y que guarda el usuario encarecidamente.

Cifrado con clave pública RSA (Rivest, Shamir, Adleman)

Históricamente, el problema de distribución de claves siempre ha sido la parte débil de la mayoría de criptosistemas. Sin importar lo robusto que sea el criptosistema, si un intruso puede robar la clave, el sistema no vale nada.

En 1976, dos investigadores de la Universidad de Stanford propusieron una clase nueva de criptosistema, en el que las claves de cifrado y descifrado eran diferentes y la clave de descifrado no podía derivarse de la clave de cifrado. En su propuesta, el algoritmo de cifrado (con clave), E, y el algoritmo de descifrado (con clave), D, tenían que cumplir los tres requisitos siguientes:

1. D(E(P))=P.

2. Es excesivamente difícil deducir D de E.

3. E no puede descifrarse mediante un ataque de texto normal seleccionado.

El método funciona como sigue. Una persona A que quiera recibir mensajes secretos, primero diseña dos algoritmos E y D, que cumplan los requisitos anteriores. El algoritmo de cifrado y la clave, E_A de cifrado, se hacen públicos, de ahí el nombre de criptografía de clave pública. Esto podría hacerse poniéndolos en un archivo accesible a cualquiera que quiera leerlo. A publica también el algoritmo de descifrado, pero mantiene secreta la clave de descifrado D_A. Por tanto E_A es pública, pero D_A es secreta.

Ahora veamos si podemos resolver el problema de establecer un canal seguro entre A y B, que nunca han tenido contacto previo. Se supone que tanto la clave de cifrado de A, E_A, como la clave de cifrado de B, E_B, están en un archivo de lectura pública. Ahora, A toma su primer mensaje P, calcula $E_B(P)$ y lo envía a B. B entonces lo descifra aplicando su clave secreta D_B (es decir, calcula $D_B(E_B(P))=P$). Nadie más puede leer el mensaje cifrado, $E_B(P)$, porque se supone que el sistema de cifrado es robusto y porque es demasiado difícil derivar D_B de la E_B públicamente conocida. A y B ahora pueden comunicarse con seguridad.

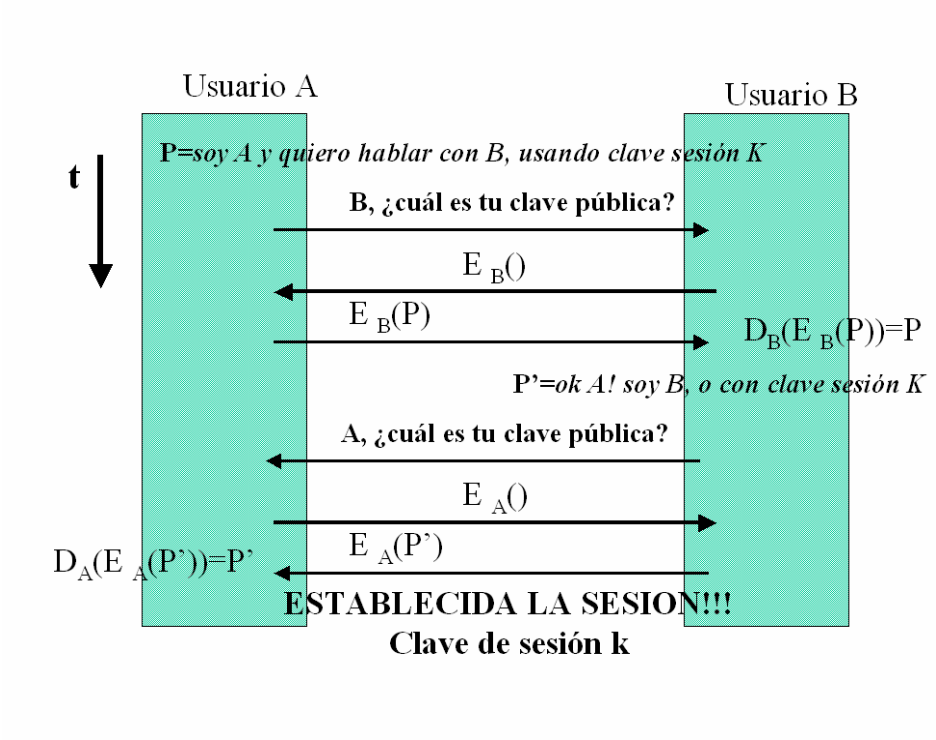


Figura 7: Escenario de establecimiento de sesión con clave asimétrica o pública

La única dificultad del método anterior estriba en que necesitamos encontrar algoritmos que realmente satisfagan los tres requisitos. Debido a las ventajas potenciales de la criptografía de clave pública, muchos investigadores están trabajando en este tema, y ya se han publicado algunos algoritmos. Un buen método fue descubierto por un grupo del M.I.T y es conocido como RSA por las iniciales de sus descubridores (Rivest, Shamir, Adleman). Su método se basa en:

1. Seleccionar dos números primos grandes, *p* y *q* (generalmente mayores que 10^{100}).

2. Calcular $n=pxq$ y $z=(p-1)x(q-1)$.

3. Seleccionar un número primo con respecto a *z* (es decir, un número sin ningún factor común con *z*), llamándolo *d*.

4. Encontrar e tal que *exd=1 mod z*

Con estos parámetros calculados por adelantado, estamos listos para comenzar el cifrado. Dividimos el texto normal (considerado como una cadena de bits) en bloques, para que cada mensaje de texto normal, P, caiga en el intervalo $0<P<n$, Esto puede hacerse agrupando el texto normal en bloques de k bits, donde k es el entero más grande para el que $2^k<n$ es verdad.

Para cifrar un mensaje, P, calculamos $C=P^e \pmod n$. Para descifrar C, calculamos $P=C^d \pmod n$. Puede demostrarse que, para todos los P del intervalo especificado, las funciones de cifrado y descifrado son inversas. Para ejecutar el cifrado, se necesitan e y n. Para llevar a cabo el descifrado se requieren d y n. **Por tanto, la clave pública consiste en el par (e,n) y la clave privada consiste en (d,n).**

La seguridad del método se basa en la dificultad de factorizar números grandes. Si el criptoanalista pudiera factorizar n (conocido públicamente), podría encontrar p y q, y a partir de éstos, z. Equipado con el conocimiento de z y de e, puede encontrar d usando el algoritmo de Euclides. Afortunadamente, los matemáticos han estado tratando de factorizar números grandes durante los últimos 300 años y las pruebas acumuladas sugieren que se trata de un problema excesivamente difícil. De acuerdo con los descubridores del RSA, la factorización de un número de 200 dígitos requiere 4 mil millones de años de tiempo de cómputo; la factorización de un número de 500 dígitos requiere 10^{25} años. En ambos casos se supone el mejor algoritmo conocido y una computadora con un tiempo de instrucción de 1 microsegundo. Aun si las computadoras continúan aumentando su velocidad, pasarán siglos antes de que sea factible la factorización de un número de 500 dígitos, y entonces simplemente se puede escoger un p y un q todavía más grandes.

Un ejemplo pedagógico trivial del algoritmo RSA se puede ver en la figura siguiente, en la cual se cifra un texto en el que la letra "a" se representa por el valor 1, la "b" por el valor 2, etc.

Texto normal (P)		Texto cifrado (C)			Después del descifrado	
Simbólico	*Numérico*	P^3	$P^3 \pmod{33}$	C^7	$C^7 \pmod{33}$	*Simbólico*
s	19	6859	28	13492928512	19	S
u	21	9261	21	1801088541	21	U
z	26	17576	20	1280000000	26	Z
a	01	1	01	1	01	A
n	14	2744	05	78125	14	N
n	14	2744	05	78125	14	N
e	05	125	26	8031810176	05	E
Cálculo del transmisor				Cálculo del receptor		

Para este ejemplo hemos seleccionado p=3 y q=11, dando n=33 y z=20. Un valor adecuado de d es d=7, puesto que 7 y 20 no tienen factores comunes. Con estas selecciones, e puede encontrarse resolviendo la ecuación $7e=1 \pmod{20}$, que produce e=3. El texto cifrado C, de un mensaje de texto normal, P, se da por la regla $C=P^3 \pmod{33}$. El texto cifrado lo descifra el receptor de acuerdo con la regla $P=C^7 \pmod{33}$. En la figura se muestra como ejemplo el cifrado del texto normal "*suzanne*".

Dado que los números primos escogidos para el ejemplo son tan pequeños, P debe ser menor que 33, por lo que cada bloque de texto normal puede contener sólo un carácter. El resultado es un cifrado por sustitución monoalfabética, no muy impresionante. En cambio, si hubiéramos seleccionado p y q del orden de 10^{100} podríamos tener n del orden de 10^{200}, por lo que cada bloque podría ser de hasta 664 bits (83 caracteres de 8 bits), contra 64 bits (8 caracteres de 8 bits) para el algoritmo DES.

Sin embargo, el algoritmo RSA es demasiado lento para poder cifrar grandes volúmenes de datos, por lo cual suele usarse para distribuir claves de sesión de una sola vez para su uso con los algoritmos DES, IDEA u otros semejantes.

El tamaño de las claves del RSA puede ser variable, mínimo 500 bits, pero lo habitual suele ser utilizar claves de 512 bits (32 dígitos hexadecimales), 768 o 1024 bits.

Se puede demostrar y comprobar que si invertimos el orden de las claves, el proceso es exactamente igual de seguro, es decir, es lo mismo $D_B(E_B(P))=P$ que $E_B(D_B(P))=P$, y según el algoritmo $C=P^d(\text{mod } n)$ y $P=C^e(\text{mod } n)$. Con ello este algoritmo puede tener utilidad para 2 tipos de situaciones. En el primer caso $D_B(E_B(P))=P$ para cifrar algo cuando se manda a B, porque utilizo su clave pública y sólo B puede descifrar con su clave privada. En el segundo caso, $E_B(D_B(P))=P$, B puede hacer uso de la clave privada para mandar información, y que TODO el mundo pueda leer si pide la clave pública de B y asegura que sólo B puede haber cifrado P, porque sólo B dispone de la clave privada correspondiente.

Otros algoritmos de cifrado con clave pública

Cabe comentar finalmente que existen otros algoritmos de clave pública (asimétrica), basados siempre en resolución de problemas matemáticos de cálculo costoso, que tienen un procedimiento directo fácil, pero difícil en inverso. Estos métodos son de resolución computacional imposible, para poder resolver la función matemática inversa.

Algoritmo de ElGamal

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

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

$$y = p^x \text{ (mod n)}$$

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

Escogiendo n primo, garantizamos que sea cual sea el valor de p, el conjunto $\{p, p^2, p^3 \ldots\}$ es una permutación del conjunto {1, 2,… n-1} lo suficientemente grande.

Firmas Digitales de ElGamal

Para firmar un mensaje m basta con escoger un número k aleatorio, tal que mcd(k, n-1) =1, y calcular

$a = p^k \text{ (mod n)}$

$b = (m - xa)k^{-1} \text{ (mod (n - 1))}$

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

$y^a a^b = p^m \text{ (mod n)}$

Codificación de ElGamal

Para codificar el mensaje m se escoge primero un número aleatorio k primo relativo con (n-1), que también será mantenido en secreto. Calculamos entonces las siguientes expresiones

$a = p^k \text{ (mod n)}$

$b = y^k m \text{ (mod n)}$

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

$m = b * a^{-x} \text{ (mod n)}$

Algoritmo de Rabin

El sistema de clave asimétrica de Rabin se basa en el problema de calcular raíces cuadradas modulo un número compuesto. Este problema se ha demostrado que es equivalente al de la factorización de dicho número.

En primer lugar escogemos dos números primos, p y q, ambos congruentes con 3 módulo 4 (los dos últimos bits a 1). Estos primos son la clave privada. La clave pública es su producto, $n = pq$.

Para codificar un mensaje m, simplemente se calcula

$c = m^2 \pmod n$

La decodificación del mensaje se hace calculando lo siguiente:

$m_1 = c^{(p+1)/4} \pmod p$
$m_2 = (p - c^{(p+1)/4}) \pmod p$
$m_3 = c^{(q+1)/4} \pmod q$
$m_4 = (q - c^{(q+1)/4}) \pmod q$

Luego se escogen a y b tales que $a = q(q^{-1} \pmod p)$ y $b = p(p^{-1} \pmod q)$. Los cuatro posibles mensajes originales son

$m_a = (am_1 + bm_3) \pmod n$
$m_b = (am_1 + bm_4) \pmod n$
$m_c = (am_2 + bm_3) \pmod n$
$m_d = (am_2 + bm_4) \pmod n$

Desgraciadamente, no existe ningún mecanismo para decidir cuál de los cuatro es el auténtico, por lo que el mensaje deberá incluir algún tipo de información para que el receptor pueda distinguirlo de los otros.

Algoritmo DSA

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

Creación del par clave pública-clave privada

El algoritmo de generación de claves es el siguiente:

1. Seleccionar un número primo *q* tal que $2^{159} < q < 2^{160}$.
2. Escoger *t* tal que $0 \leq t \leq 8$, y seleccionar un número primo *p* tal que $2^{511+64t} < p < 2^{512+64t}$, y que además q sea divisor de (p - 1).
3. Seleccionar un elemento $g \in Z^*_p$ y calcular $\alpha = g^{(p-1)/q} \bmod p$.
4. Si $\alpha = 1$ volver al paso 3.
5. Seleccionar un número entero aleatorio a, tal que $1 \leq a \leq q - 1$
6. Calcular $y = \alpha^a \bmod p$.
7. La clave pública es (p, q, α, y). La clave privada es *a*.

Generación y verificación de la firma

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

1. Seleccionar un número aleatorio k tal que $0 < k < q$.
2. Calcular $r = (\alpha^k \bmod p) \bmod q$.
3. Calcular $k^{-1} \bmod q$.
4. Calcular $s = k^{-1}(h + ar) \bmod q$.
5. La firma del mensaje m es el par (r, s).

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

1. Verificar que $0 < r < q$ y $0 < s < q$. En caso contrario, rechazar la firma.
2. Calcular el valor de h a partir de m.
3. Calcular $\omega = s^{-1} \bmod q$.
4. Calcular $u_1 = \omega * h \bmod q$ y $u_2 = \omega * r \bmod q$.
5. Calcular $v = (\alpha^{u1} y^{u2} \bmod p) \bmod q$.
6. 6. Aceptar la firma si y sólo si v = r**Localización de los dispositivos de cifrado**

Existen dos alternativas para la ubicación del cifrado, y vamos a analizarla desde el punto de vista de una sesión TCP/IP:

-cifrado de enlace a enlace, donde cada nodo intermedio (router) debe descifrar los paquetes
-cifrado extremo a extremo, donde sólo los extremos pueden descifrar la información, pero las cabeceras de los paquetes han de viajar descifradas para que los routers puedan encaminar

y una combinación de los anteriores, **cifrado mixto enlace-enlace, extremo-extremo**, donde las cabeceras van cifradas enlace a enlace y los datos extremo a extremo

Con esta estrategia de localización de los cifradores, podemos ocultar la información, pero realmente a un intruso sólo se le oculta las direcciones y los contenidos, pero no el **volumen de información** intercambiado. Para ocultar dicha información, se integra **tráfico de relleno**, para ocultar el volumen de información real.

Por ejemplo, si un usuario trata de acceder a un servidor de banca electrónica y la información intercambiada es mínima, frente a otro usuario que el volumen de información intercambiado es grande, da a entender que el segundo tiene más trato con el banco y el volumen de dinero debería ser mayor.

Comentario clave pública vs clave privada

Dada la lentitud y coste computacional de los métodos de las claves asimétricas, del orden de 1000 veces mayor, aunque es sí más seguros, en la realidad toda conexión remota constar de 2 partes. Un primer establecimiento para validar y autentificar, además de negociar una clave secreta para la sesión, utilizando para ello la clave pública y una segunda parte, que consiste en el cifrado de forma simétrica (DES, 3DES, IDEA, ...) con la clave llave secreta negociado.

9.2 PROTOCOLOS DE SEGURIDAD: AUTENTICACIÓN Y VALIDACIÓN

La validación de identificación es la técnica mediante la cual un proceso comprueba que su compañero de comunicación es quien se supone que es y no un impostor. La verificación de la identidad de un proceso remoto ante un intruso activo malicioso es sorprendentemente difícil y requiere protocolos complejos basados en criptografía.

Los métodos de autenticación o validación pueden estar basados en una clasificación más amplia como podemos ver a continuación:

- biomédicas, por huellas dactilares, retina del ojo, ...son muy seguros pero muy costosos
- tarjetas inteligentes que guardan información de los certificados de un usuario
- métodos clásicos basados en contraseña, son sistemas basados en esta tecnología están:
  - –Método tradicional o comprobación local
  - –Método distribuido o en red

Los métodos basados en protocolo, también son conocidos como métodos distribuidos o basados en red.

El modelo general que usan todos los protocolos de validación de identificación basado en red es éste. Un usuario (en realidad, un proceso) iniciador, digamos A, quiere establecer una conexión segura con un segundo usuario, B. A y B se llaman principales, los personajes centrales de nuestra historia. B es un banquero con el que A quiere hacer negocios. A comienza por enviar un mensaje a B o a un centro de distribución de claves (KDC) confiable, que siempre es honesto. Siguen varios otros intercambios de mensajes en diferentes direcciones. A medida que se envían estos mensajes, un intruso malicioso, C, puede interceptarlos, modificarlos o reproducirlos para engañar a A y a B, o simplemente para estropear sus actividades.

No obstante, una vez que se ha completado el **protocolo**, A está segura de que está hablando con B, y B está seguro de que está hablando con A. Es más, en la mayoría de los protocolos, los dos también habrán establecido una clave de sesión secreta para usarla durante toda la conversación subsiguiente. En la práctica, como hemos dicho en la sección anterior, por razones de prestaciones, todo el tráfico de datos se cifra usando criptografía de clave secreta, aunque la criptografía de clave pública se usa ampliamente con los protocolos de validación de identificación mismos y para establecer la clave de sesión. Por esto, la clave pública es utilizada únicamente para identificarse inicialmente y conseguir una clave de sesión. Obviamente esta clave debe ser mucho más confidencial que la clave de sesión, dado que si un intruso la obtuviera, podría descifrar y obtener desde ese momento TODAS las claves de TODAS las sesiones que realizáramos.

El objetivo es usar una clave de sesión nueva, seleccionada aleatoriamente, para cada nueva conexión con tal de reducir al mínimo la cantidad de tráfico que es enviado y así la cantidad de texto cifrado que puede obtener un intruso está limitado sólo al de la sesión. En el caso de que el intruso descifrara y obtuviera la clave, con suerte, la única clave presente en ese momento sería la clave de la sesión. Las claves públicas o secretas compartidas siempre se guardan bajo fuerte custodia, ya que son la clave para obtener todas las demás..

Los métodos distribuidos o en red además se clasifican en:

clave secreta (*privada*)

- basados en clave compartida, Diffie Hellman
- basado en servidores con registro de todos los usuarios. Ejemplo TACACS+ y Radius
- basados en NIS, NIS+ (*Network Information Service*)
- basados en centros de distribución de clavesbasados en criptografía, Kerberos

clave pública (que veremos en mayor detalle en el siguiente apartado)

- basados en servidores de directorios X.500, LDAP (*Lighweight Directory Access Protocol)*
- basados en Certificados

Pasemos a ver con detalle cada uno de ellos.

Validación de identificación de clave secreta.

Veremos a continuación distintos protocolos de validación basados en el uso de una clave secreta compartida por los usuarios A y B o por los usuarios y por un centro de distribución de claves (KDC).

Validación de identificación basada en clave secreta compartida.

Para nuestro primer protocolo de validación de identificación, supondremos que A y B ya comparten una clave secreta K_{AB}. Esta clave compartida podría haberse acordado telefónicamente o en persona pero, en cualquier caso, no a través de la (insegura) red.

Este protocolo se basa en un principio encontrado en muchos protocolos de validación de identificación: una parte envía un número aleatorio a la otra, que entonces lo transforma de una manera especial y devuelve el resultado. Tales protocolos se llaman protocolos de reto-respuesta. En éste, y en los subsiguientes protocolos de validación de identificación se usara la siguiente notación.

- A y B son las identidades de las procesos que desean verificar su identidad y C es la identidad del intruso.
- R_i son los retos, donde el subíndice identifica el retador.
- K_i son las claves, donde i indica el dueño; K_s es la clave de la sesión.

La secuencia de mensajes de nuestro primer protocolo de validación de identificación de clave compartida se muestra en la figura siguiente.

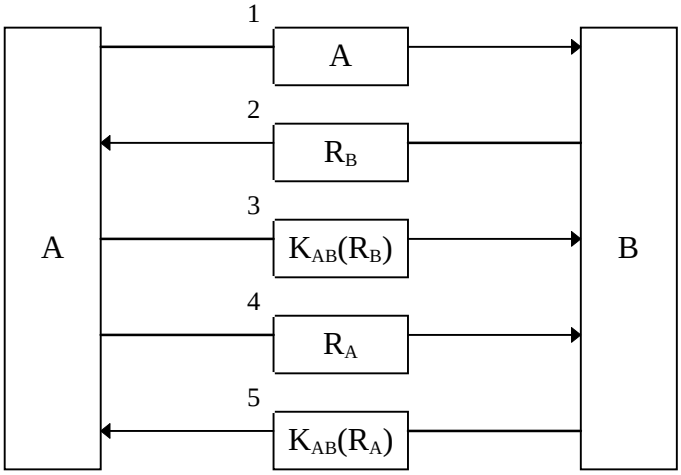


Figura 8: Validación de identificación de clave compartida.

En el mensaje 1, A envía su identidad, A, a B de manera que la entienda B. B, por supuesto, no tiene manera de saber si este mensaje vino de A o de C, por lo que escoge un reto, un número aleatorio grande, R_B, y lo envía a A como mensaje 2, en texto normal. A entonces cifra el mensaje con la clave que comparte con B y envía el texto cifrado $K_{AB}(R_B)$, en el mensaje 3. Cuando B ve este mensaje, de inmediato sabe que vino de A porque C no conoce K_{AB} y, por tanto, no pudo haberlo generado. Es más, dado que se selecciono R_B al azar de un espacio grande (digamos, números aleatorios de 128 bits), es muy poco probable que C haya visto R_B y su respuesta en una sesión previa.

En este punto, B está seguro que está hablando con A, pero A no está segura de nada; hasta donde sabe, C podría haber interceptado el mensaje 1 y enviado R_B como respuesta. Para saber a quién le está hablando, A selecciona un número al azar, R_A, y lo envía a B como texto normal, en el mensaje 4. Cuando B responde con $R_{AB}(K_A)$, A sabe que está hablando con B. Si desean establecer ahora una clave de sesión, A puede seleccionar una K_S, y enviársela a B cifrada con K_{AB}.

Aunque el protocolo anterior funciona, contiene mensajes de más. Estos pueden eliminarse combinando información y reduciendo el protocolo a tres mensajes, como puede verse en la siguiente figura:

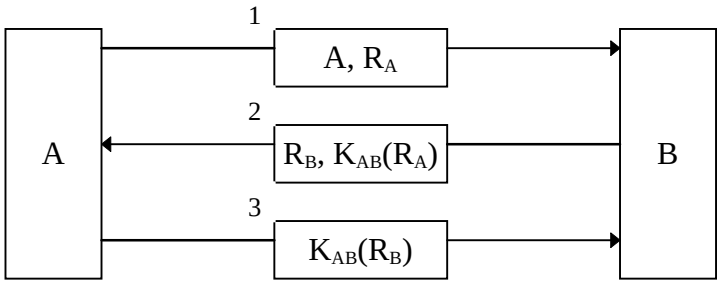


Figura 9: Mejora del algoritmo de validación de identificación de clave compartida.

Sin embargo, estos protocolos poseen un fallo. Un intruso C puede burlar este protocolo usando lo que se conoce como ataque por reflexión. En particular, C puede inutilizarlo si es posible abrir al mismo tiempo varias sesiones con B. Esta situación sería cierta si, por ejemplo, B es un banco y está preparado para aceptar muchas conexiones simultáneas de los cajeros bancarios.

El ataque por reflexión de C se muestra en la figura siguiente; comienza cuando C indica que es A y envía R_C. B responde, como de costumbre, con su propio reto, R_B. Ahora C está atorado, pues no conoce $K_{AB}(R_B)$.

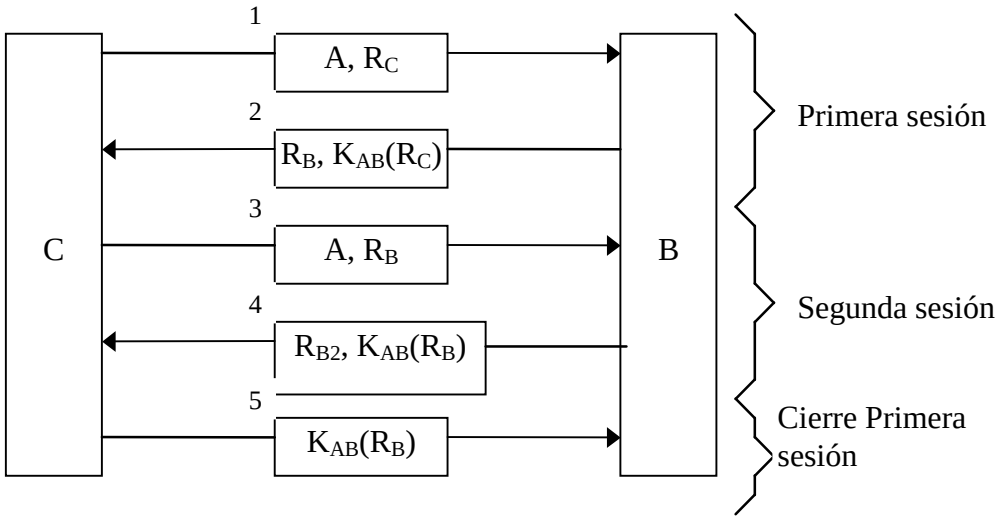


Figura 10: Ataque por reflexión al algoritmo de validación de identificación de clave compartida.

C puede abrir una segunda sesión con el mensaje 3, proporcionando como reto el R_B tomado del mensaje 2. B lo cifra y envía de regreso $K_{AB}(R_B)$ en el mensaje 4. Ahora C tiene toda la información faltante, por lo que puede completar la primera sesión y abortar la segunda. B ahora está convencido de que C es A, por lo que le autoriza cualquier operación que desee hacer como si fuera A.

Tres reglas generales que frecuentemente son de ayuda son las siguientes:

1. Hacer que el iniciador demuestre quién es antes de que lo tenga que hacer el respondedor. En este caso, B regala información valiosa antes de que C dé cualquier prueba de su identidad.

2. Hacer que el iniciador y el respondedor usen diferentes claves para comprobación, incluso si esto significa tener dos claves compartidas, K_{AB} y K'_{AB}.

3. Hacer que el iniciador y el respondedor tomen sus retos de diferentes conjuntos. Por ejemplo, el iniciador debe usar números pares y el respondedor números nones.

Las tres reglas se violaron aquí, con resultados desastrosos. Nótese que nuestro primer protocolo de validación de identificación (de cinco mensajes) requiere que A compruebe primero su identidad, por lo que ese protocolo no está a merced del ataque por reflexión.

Establecimiento de una clave compartida: intercambio de claves Diffie-Hellman.

Hasta ahora hemos supuesto que A y B comparten una clave secreta. Pero puede no ser así, sin embargo, y afortunadamente, existe una manera de que completos desconocidos establezcan una clave secreta a plena luz del día, aún con C registrando cuidadosamente cada mensaje.

El protocolo que permite que dos extraños establezcan una clave secreta compartida se llama intercambio de claves Diffie-Hellman, y funciona como sigue. A y B tiene que acordar dos números primos grandes, n y g, donde (n-1)/2 también es primo y g debe cumplir ciertas condiciones. Esto números pueden ser públicos, por lo que cualquiera de ellos puede escoger n y g y decírselo al otro abiertamente. Ahora, A escoge un número grande (digamos de 512 bits), x, y lo mantiene en secreto. De la misma manera, B escoge un número secreto grande y.

A inicia el protocolo de intercambio de claves enviando a B un mensaje que contiene (n, g, g^x mod n), como se muestra en la figura siguiente. B responde enviando a A un mensaje que contiene g^y mod n. Ahora A toma el número que B le envió y lo eleva a la potencia x para obtener $(g^y \text{ mod } n)^x$. B lleva a cabo una tarea parecida para obtener $(g^x \text{ mod } n)^y$. Por las leyes de la aritmética modular, ambos cálculos arrojan g^{xy} mod n. Ahora A y B comparten una clave secreta g^{xy} mod n.

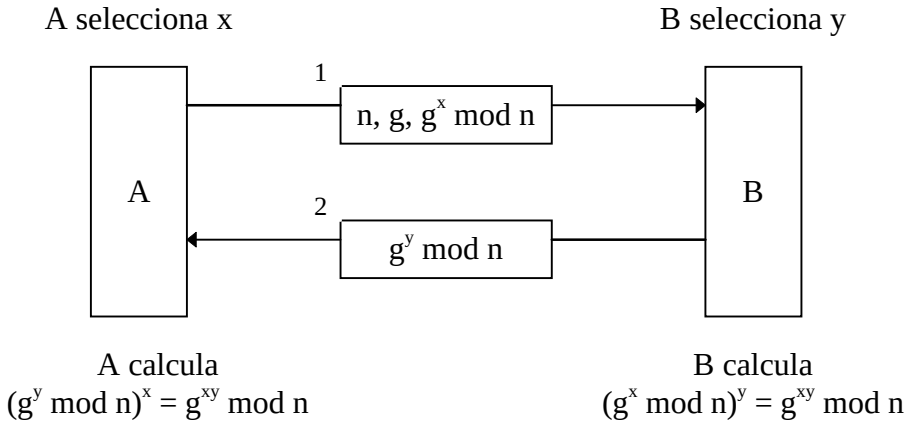


Figura 11:Intercambio de claves Diffie-Hellman.

Por supuesto, C ha visto ambos mensajes, así que conoce g y n del mensaje 1. Si C pudiera calcular x y y, podría averiguar la clave secreta. El problema es que, dado sólo g^x mod n, no es posible encontrar x. No se conoce un algoritmo práctico para calcular logaritmos discretos módulo un número primo muy grande.

A pesar de la elegancia del algoritmo Diffie-Hellman, hay un problema: cuando B recibe la tripleta (n, g, g^x mod n), no sabe si es A o C quien se la ha enviado. No hay manera de saberlo. Desgraciadamente, C puede explotar este hecho para engañar tanto a A como a B, como se ilustra en la siguiente figura.

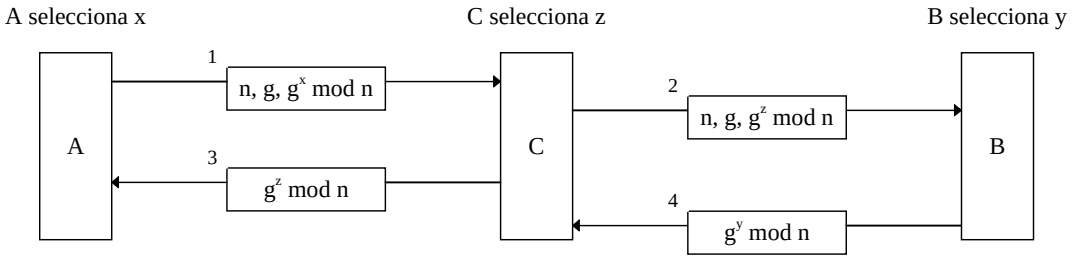


Figura 12: Ataque de brigada de cubetas al intercambio de claves Diffie-Hellman.

Ahora todos hacen la aritmética modular. A calcula la clave secreta como g^{xz} mod n, y también lo hace C (para los mensajes de A). B calcula g^{yz} mod n, al igual que C (para los mensajes de B). A piensa que esta hablando con B, por lo que establece una clave de sesión con C. Lo mismo hace B. Cada mensaje que A envía durante la sesión cifrada es capturado por C, almacenado, modificado y pasado (opcionalmente) a B. Lo mismo ocurre en la otra dirección. C ve todo y puede modificar los mensajes si lo desea, mientras que tanto A como B están pensando equivocadamente que tienen un canal seguro entre ambos. Este ataque se conoce como ataque de brigada de cubetas o ataque del hombre (mujer) en medio.

Validación de identificación usando un centro de distribución de claves.

El establecimiento de un secreto compartido con un extraño casi funcionó, pero no por completo. Por otra parte no valía la pena, pues para hablarle a n personas de esta manera se requerían n claves. Para la gente muy popular, la administración de claves se volvería una verdadera carga, especialmente si cada clave tuviera que almacenarse en un chip de una tarjeta de plástico individual.

Un enfoque diferente es introducir un centro de distribución de claves confiables (KDC). En este modelo, cada usuario tiene una sola clave compartida con el KDC. La validación de identificación y la administración de claves de sesión ahora pasan a través del KDC. El protocolo de validación e identificación más simple conocido es la rana de boca amplia, que puede verse en la siguiente figura:

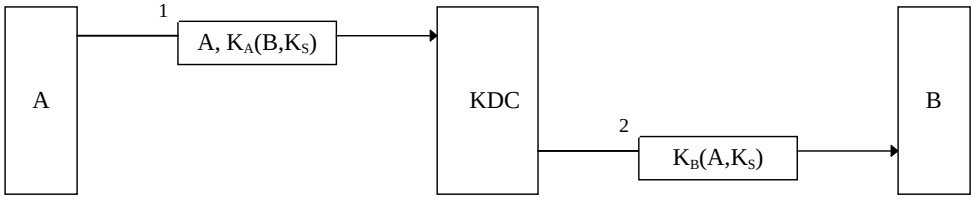


Figura 13: Protocolo de validación de rana de boca amplia.

El concepto en que se basa el protocolo de boca amplia es sencillo: A escoge una clave de sesión, K_S e indica al KDC que desea hablar con B usando K_S. Este mensaje se cifra con la clave secreta que comparte A (solamente) con el KDC, K_A. El KDC descifra este mensaje, extrayendo la identidad de B y la clave de sesión; luego construye un mensaje nuevo que contiene la identidad de A y la clave de sesión y envía este mensaje a B. Este cifrado se hace con K_B, la clave secreta que B comparte con el KDC. Cuando B descifra el mensaje, sabe que A quiere hablar con él, y también conoce la clave que desea usar.

La validación de identificación aquí es gratuita. El KDC sabe que el mensaje 1 debe haber venido de A, puesto que nadie más habría sido capaz de cifrarlo con la clave secreta de A. De la misma manera, B sabe que el mensaje 2 debe haber venido del KDC, en quien confía, puesto que nadie más sabe su clave secreta.

Sin embargo, este protocolo tiene un fallo, un intruso C puede copiar el mensaje 2 de la figura y todos los mensajes que le siguen y reproducirlos de nuevo para B, con lo cual B creerá estar hablando con A. Este problema se llama ataque por repetición, con lo cual, aunque no pueda modificar C los mensajes, puede hacer interferir a B y pueda generar a la larga en un ataque que se llama denegación de servicio.

Son posibles varias soluciones al ataque por repetición. La primera es incluir una marca de tiempo en cada mensaje. Entonces, si alguien recibe un mensaje obsoleto, puede descartarlo. El problema de este enfoque es que los relojes nunca están perfectamente sincronizados en toda una red, por lo que tiene que haber un intervalo durante el cual la marca de tiempo sea válida. C puede repetir el mensaje durante ese tiempo y salirse con la suya.

La segunda solución es poner un número de mensaje único, de una sola ocasión, a veces llamado núnico, en cada mensaje. Cada parte entonces tiene que recordar todos los núnicos y rechazar cualquier mensaje que contenga un núnico previamente usado. Pero los núnicos tienen que recordarse indefinidamente, no vaya a ser que C esté tratando de reproducir un mensaje de años de antigüedad. También, si una máquina se cae y pierde la lista de núnicos, nuevamente es vulnerable a un ataque por repetición. Las marcas de tiempo y los núnicos pueden combinarse para limitar el tiempo que pueden recordarse los núnicos, pero ciertamente el protocolo se volverá mucho más complicado.

Un enfoque más refinado para la validación de identificación es usar un protocolo multisentido de reto-respuesta. Un ejemplo bien conocido de tal protocolo es el de validación de identificación Needham-Schroeder, que puede verse en la siguiente figura:

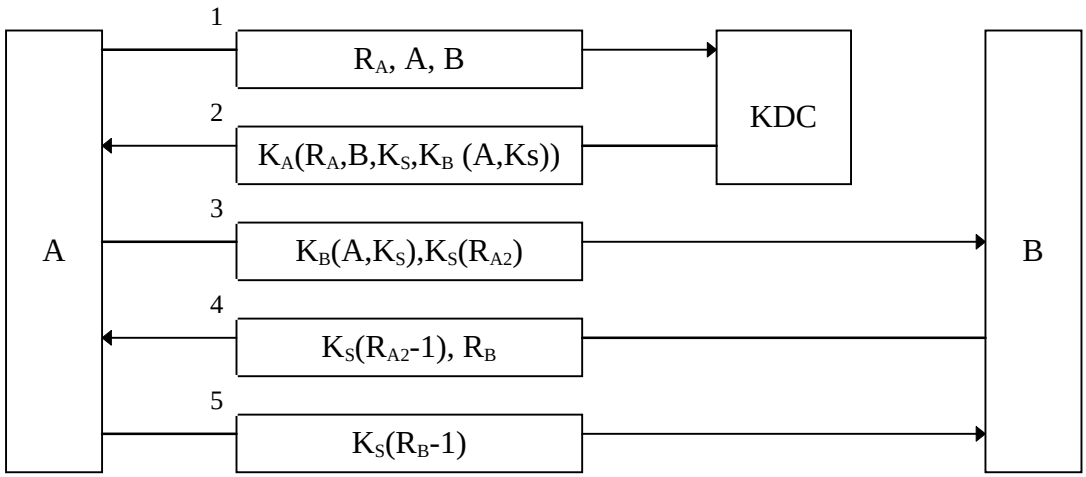


Figura 14: Protocolo de validación de identificación Needham-Schoeder.

El protocolo comienza cuando A indica al KDC que quiere hablar con B. Este mensaje contiene un número aleatorio grande, R_A, como núnico. El KDC devuelve el mensaje 2 que contiene el número aleatorio de A, una clave de sesión, y un billete que puede enviar a B. El objeto del número aleatorio, R_A, es asegurar a A que el mensaje 2 es reciente, y no una repetición. La identidad de B también se incluye por si a C se le ocurre reemplazar B del mensaje 1 por su propia identidad, para que el KDC cifre el billete al final del mensaje 2 con K_C en lugar de K_B. El billete cifrado con K_B se incluye dentro del mensaje cifrado para evitar que C lo reemplace por otra cosa en su camino hacia A.

A ahora envía el billete a B, junto con un nuevo número aleatorio, R_{A2}, cifrado con la clave de la sesión, K_S. En el mensaje 4, B devuelve $K_S(R_{A2}-1)$ para demostrar a A que está hablando con el verdadero B. El envío de regreso de $K_S(R_{A2})$ no habría funcionado, puesto que C lo podría haber robado del mensaje 3.

Tras recibir el mensaje 4, A está convencida de que está hablando con B, y de que no se pudieron haber usado repeticiones hasta el momento. A fin de cuentas, A acaba de generar R_{A2}, hace unos cuantos milisegundos. El propósito del mensaje 5 es convencer a B de que realmente está hablando con A, y de que tampoco se han usado repeticiones aquí. Al hacer que cada parte genere un reto y responda a otro, se elimina la posibilidad de un ataque por repetición.

Aunque este protocolo parece bastante sólido, tiene una ligera debilidad, si C llega a obtener una clave de sesión vieja en texto normal, puede iniciar una nueva sesión con B repitiendo el mensaje 3 correspondiente a la clave obtenida y convencerlo de que es A. Otway y Rees publicaron un protocolo que resuelve el problema. La figura siguiente muestra el protocolo de Otway-Rees:

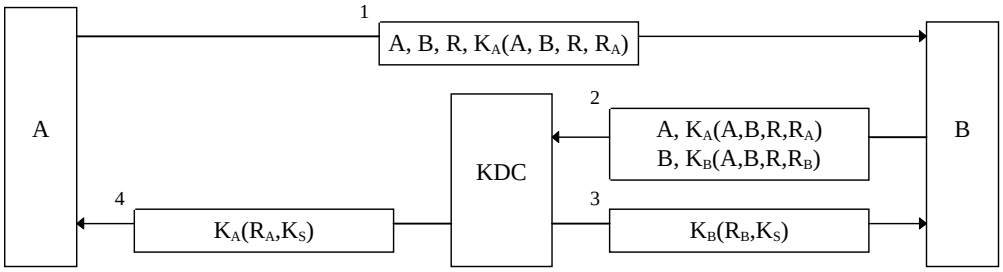


Figura 15: Protocolo de validación de identificación Otway-Ress.

En el protocolo Otway-Rees, A comienza por generar un par de números aleatorios: R, que se usará como identificador común, y R_A, que A usará para retar a B. Cuando B recibe este mensaje, construye un mensaje nuevo a partir de la parte cifrada del mensaje de A, y uno análogo propio. Ambas partes cifradas con K_A y K_B identifican a A y a B, contienen el identificador común y contienen un reto.

El KDC comprueba que el R de ambas partes es igual. Los R podrían no serlo porque C alteró el R del mensaje 1 y reemplazó parte del mensaje 2. Si los dos R son iguales, el KDC se convence de que el mensaje de solicitud de B es válido, y genera una clave de sesión y la cifra dos veces, una para A y otra para B. Cada mensaje contiene un número aleatorio del receptor como prueba de que el KDC, y no C, generó el mensaje. En este punto, tanto A como B poseen la misma clave de sesión y pueden comenzar a comunicarse. La primera vez que intercambien mensajes de datos, ambos podrán ver que el otro tiene una copia idéntica de K_S, por lo que la validación de identificación está completa.

Protocolo de autenticación Kerberos.

El protocolo de autenticación Kerberos fue diseñado por el MIT (Massachusetts Institute of Technology) para autentificar la identidad de los usuarios de una red digital de comunicaciones insegura, así como para distribuir las claves secretas de sesión que permitan a los usuarios de la red establecer comunicaciones seguras. Es conveniente resaltar que como usuario se engloba tanto a personas físicas que se comunican a través de ordenadores como a posibles servicios ofrecidos por la red.

El protocolo está basado en criptografía de clave secreta y básicamente actúa como un árbitro en quien los usuarios confían. Para desarrollar sus funciones, Kerberos comparte con cada usuario una clave secreta diferente intercambiada con éste a través de un canal seguro. El conocimiento de dicha clave se utiliza como prueba de identidad del usuario. En estas condiciones como Kerberos conoce las claves secretas de todos los usuarios, puede demostrar a cualquiera de ellos la autenticidad de la identidad de otro. Además Kerberos genera claves secretas transitorias, denominadas claves de sesión, las cuales son transmitidas exclusivamente a cada pareja de usuarios que desean entrar en comunicación.

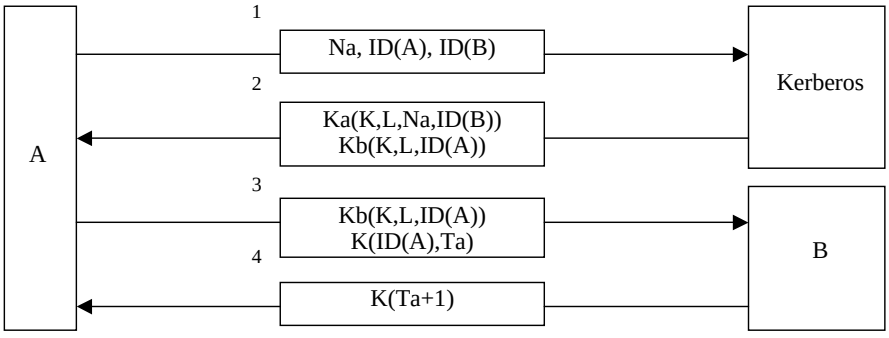


Figura 16: Protocolo de validación de identificación Kerberos.

Veamos el funcionamiento básico del protocolo. Para ello consideremos únicamente a los tres participante básicos del mismo: la autoridad de autenticación y servidor de claves Kerberos y dos usuarios de la red A y B. Al comienzo del protocolo A y B no comparten ninguna clave secreta. Sin embargo, cada uno de

ellos comparte con Kerberos una clave secreta de un criptosistema simétrico. Sea $E_d(\ldots)$ dicho criptosistema (clave simétrica *d*) y sean *Ka* y *Kb* las respectivas claves secretas compartidas con Kerberos por los usuarios A y B. Asimismo, sean *ID(A)* e *ID(B)* los respectivos identificadores de A y B en la red de comunicación. El funcionamiento del protocolo se describe a continuación:

1- El usuario A solicita a Kerberos una credencial para identificarse ante B, así como una clave de sesión que le permita establecer dicha comunicación. Para ello A envía a Kerberos un valor aleatorio *Na* y las identidades *ID(A)* e *ID(B)*.

2- Kerberos genera una clave de sesión aleatoria *K* y define el período de validez *L* de la credencial, cifrando a continuación los valores *K*, *L*, *Na* y la identidad *ID(B)* del usuario B con la clave secreta *Ka* del usuario A, obteniendo el valor *m* dado por $m = E_{Ka}(K, L, Na, ID(B))$.

   Seguidamente Kerberos calcula la credencial *c* cifrando *K*, *L* y la identidad *ID(A)* del usuario A con la clave secreta *Kb* del usuario B, es decir $c = E_{Kb}(K, L, ID(A))$.

   En estas condiciones Kerberos envía la pareja de valores *(m, c)* al usuario A.

3- El usuario A descifra m con su clave secreta *Ka* y recupera *K*, *L*, *Na* y la identifidad *ID(B)* del usuario B para el que fue emitida la credencial *c*. Entonces A verifica que el valor aleatorio *Na* corresponde con el que él previamente envió y guarda *L* como referencia. A continuación, A calcula el autentificador *a* para la credencial *c* cifrando su identidad *ID(A)* y su sello temporal *Ta* con la clave de sesión *K*, es decir $a = E_K(ID(A), Ta)$.

   Con todo ello el usuario A envía a B la pareja de valores *(c, a)*.

4- El usuario B recibe los valores *(c, a)* y descifra la credencial c con su clave secreta *Kb*, recuperando de esta forma *K*, *L* y la identidad *ID(A)*. Entonces utiliza la clave de sesión recuperada *K* para descifrar el autentificador y recuperar los valores *ID(A)* y *Ta* con el que comprueba que:

   - Las identidades de la credencial y el autentificador coinciden.
   - El sello *Ta* es válido y se encuentra en los límites de *L*.

5- Si las comprobaciones son satisfactorias, B se convence de la autenticidad de la identidad de A, así como de que la clave de sesión cifrada en la credencial *c* es la misma con la que se ha cifrado el autentificador *a*. En tal caso, B envía a A la conformidad *h* dada por $h = E_K(Ta + 1)$.

6- Por su parte el usuario A descifra *h* con la clave de sesión *K* y verifica que el valor recuperado es *Ta+1*, lo cual asegura a A que la clave de sesión *K* ha sido correctamente recibida por el usuario B.

Es conveniente hacer notar las diferentes funciones de los mensajes transmitidos en el protocolo anterior. Así por ejemplo, los valores cifrados *c* y *m* tienen la función de garantizar la confidencialidad en la transmisión de la clave secreta de sesión *K*, mientras que los valores *a* y *h* se utilizan para la confirmación de la clave, es decir, para convencer a los usuarios A y B de que ambos poseen la misma clave secreta de sesión *K*.

El propósito del sello temporal *Ta* y del período de validez *L* es doble. Por una parte, tienen como objetivo que el usuario A pueda utilizar la credencial c para realizar sucesivas autentificaciones ante B durante el período de validez de la clave sin necesidad de que intervenga Kerberos. El otro objetivo es el de prevenir un posible ataque activo al protocolo consistente en el almacenamiento de las claves para su posterior reutilización. Esto se consigue puesto que una clave no se acepta como válida si su sello temporal no se encuentra dentro de los límites del período de validez de la misma.

A continuación comentamos algunos puntos sobre Kerberos en plan resumen:

1. Kerberos viene en la mayoría de distribuciones UNIX y/o Linux, utilizado en procesos como login, rlogin o NFS (Network File System)
2. es un sistema centralizado y no distribuido en una red, lo cual si falla .o si se ve comprometido, se compromete toda la red, además las claves almacenadas son privadas y no públicas. En versión 4 y/o 5 incorpora servidores secundarios.requiere la kerberización (modificación) de todas las aplicaciones, así como de la sincronización de todas las máquinas
4. las versiones más actualizadas son v4 y v5, la diferencia estriba en que v5 solicita tanto login como password del usuario para conectar al servidor Kerberos, de forma que un intruso conociendo un usuario no pueda por fuerza bruta de contraseñas descifrar la contestación del servidor. Dicha información de contraseña junto con información de tiempo, va en el reto inicial R_A
5. cuando un usuario está más de 8 horas (por defecto) delante de una estación de trabajo kerberizada, debe identificarse otra vez
6. algunos inconvenientes citados se han resuelto con versiones mejoradas o con la utilización de otras aplicaciones, como SSH (Secure Shell)
7. en ningún momento los passwords viajan por la red ni guardados en memoria, sólo las credenciales
8. cabe destacar del protocolo visto, que realmente se distinguen dos partes, por una lado la autenticación y por otro la gestión de tickets para los diferentes servicios

Veamos un pequeño ejemplo de funcionamiento con Kerberos

Escenario: usuario A a través de login requiere de las credenciales necesarias para acceder a otros servicios

Pasos:

1.- al teclear el nombre, el login kerberizado envía el nombre al servidor Kerberos solicitando un ticket

2.- si el usuario es conocido, le manda un mensaje cifrado, donde utilizará su contraseña (password) para descifrarlo. De esta forma la contraseña nunca viaja por la red

3.- de dicho mensaje extrae el ticket para realizar la petición de servicio

Validación de identificación usando criptografía de clave pública.

La validación de identificación mutua también puede hacerse usando criptografía de clave pública. Para comenzar, supongamos que A y B ya conocen las claves públicas del otro (un asunto nada trivial). Ellos quieren establecer una sesión y luego usar criptografía de clave secreta en esa sesión, ya que típicamente es de 100 a 1000 veces más rápida que la criptografía de clave pública. El propósito de este intercambio inicial entonces es validar la identificación de ambos y acordar una clave de sesión secreta compartida.

Este procedimiento puede hacerse de varias maneras. Una típica se muestra en la figura siguiente:

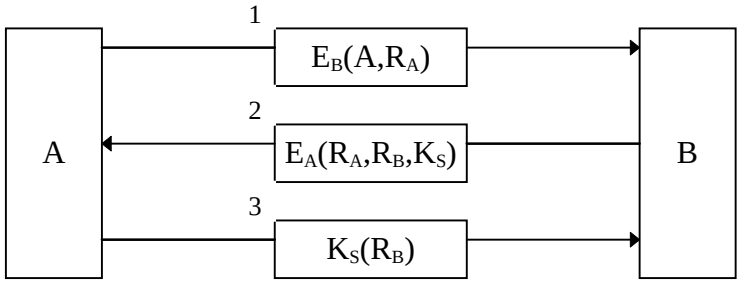


Figura 17 Protocolo de validación de identificación usando criptografía de clave pública.

Aquí, A comienza por cifrar su identidad y un número al azar, R_A, usando la clave pública (o de cifrado) de B, E_B. Cuando B recibe este mensaje, no sabe si vino de A o de C, pero entra en el juego y devuelve a A un mensaje que contiene el R_A de A, su propio número aleatorio, R_B, y la clave de sesión propuesta K_S.

Cuando A recibe el mensaje 2, lo descifra usando su clave privada y encuentra R_A en él, lo que le hace sentirse cómodo. El mensaje debió haber venido de B, puesto que C no tiene manera de determinar R_A. Es

más, el mensaje debe ser reciente y no una repetición, puesto que acaba de enviar R_A a B. A acepta la sesión devolviendo el mensaje 3. Cuando B ve R_B cifrado con la clave de sesión que él acaba de generar, sabe que A recibió el mensaje 2 y comprobó R_A.

El protocolo sin embargo tiene una debilidad, ya que supone que A y B conocen la clave pública del otro. Supongamos que no es así. A podría simplemente enviar a B su clave pública en el primer mensaje y pedir a B que devuelva la suya en el siguiente. El problema de este enfoque es que está sujeto a un ataque de brigada de cubetas. C puede capturar el mensaje de A a B y devolver su propia clave a A. A pensará que tiene una clave para hablar con B cuando, de hecho, tiene una clave para hablar con C. Ahora C puede leer todos los mensajes cifrados con lo que A piensa es la clave pública de B. El intercambio inicial de claves públicas puede evitarse almacenando todas las claves públicas en una base de datos pública. Así, A y B pueden obtener la clave pública del otro de la base de datos. Sin embargo, C aún puede poner en práctica el ataque de brigada de cubetas interceptando las solicitudes a la base de datos y enviando respuestas simuladas que contengan su propia clave.

Rivest y Shamir han diseñado un protocolo que fustra el ataque de brigada de cubetas de C. En su protocolo de interbloqueo, tras el intercambio de claves públicas, A envía solo la mitad de su mensaje a B al mismo tiempo que B espera la mitad de A, digamos por ejemplo que sólo los bits pares (después del cifrado). B entonces responde con sus bits pares y A también responde con sus bits pares. Tras recibir los bits pares de A y B, A envía sus bits nones, y luego lo hace B. El truco aquí es que, cuando C recibe los bits pares de A o de B, no puede descifrar el mensaje, aunque tiene la clave privada. En consecuencia, C es incapaz de volver a cifrar los bits pares usando la clave pública de A o de B. Si C envía basura a B o a A, el protocolo continuará, pero B o A pronto se dará cuenta de que el mensaje reensamblado no tiene sentido y de que ha sido engañado.

TACACS, XTACACS, TACACS+, RADIUS y otros

Todos son protocolos e implementaciones de control de acceso por validación y autenticación, que permiten que un servidor del acceso de red (**NAS** *Network Access Servers o* **RAS** *Remote Access Server*, por ejemplo un router Cisco 2511 o 5300) obtenga datos de administación del usuario a un servidor central y por tanto se descarge de la tarea administrativa de autenticación y comprobación de dicha información.

También, cabe considerar otro tipo de servidor para autenticación muy extendido en las redes Microsoft llamado Active Directory, que permite gestionar los usuarios y sus permisos (exportando incluso el escritorio) independientemente de la máquina conectada dentro de un dominio Microsoft. Los dominios Microsoft son gestionados por el Controlador de Dominio (Domain Controller, DC). Además, es importante resaltar que Active Directory incluye Kerberos y servidores de directorio como veremos a continuación, concretamente LDAP.

Todos ellos se pueden considerar el soporte a los sistemas de autenticación basado en contraseñas, no son considerados sistemas de clave simétrica compartida, porque en un principio no están pensados para intercambiar claves para cifrar, simplemente para poder autenticar.

Por tanto, como parte de estos protocolos de autenticación, hemos incluido estos protocolos muy utilizados en los sistemas de acceso en red, desde conexiones remotas.

Hay tres versiones del protocolo de autenticación "TACACS" (Terminal Access Controller Access Control System, sistema de control de acceso del Controlador de Acceso de Terminales). El primer es TACACS ordinario, fue el protocolo utilizado por Cisco en sus NAS, y ha estado en el uso por muchos años. El segundo es una extensión al primero, comúnmente llamado Extended Tacacs o XTACACS, introducido en 1990. Ambos se documentan en RFC1492. El tercero, TACACS+ que, a pesar del nombre, es totalmente un nuevo protocolo y no es compatible con TACACS o XTACACS.

TACACS y XTACACS carecen de muchas características de TACACS+ y RADIUS, y actualmente están fuera de mantenimiento. Debido a esto TACACS y XTACACS no serán discutidos.

RADIUS (Remote Authentication Dial In User Service) es un protocolo de control de accesos desarrollado por Livingston Enterprises y que la IETF ha recogido en los RFCs 2865 y 2866. Fue diseñado para autenticar usuarios y utiliza una arquitectura cliente/servidor. El servidor contiene información de los usuarios, almacenando sus contraseñas y sus perfiles, y el cliente es el encargado de pasar las peticiones de conexión de los usuarios al servidor para que éste las autentique y responda al cliente diciéndole si ese usuario está o no registrado.

Un ejemplo de uso de RADIUS se puede dar en un ISP, donde el NAS (Network Access Server) hace de cliente RADIUS y un host del ISP podría hacer de servidor RADIUS. El NAS recibe vía módem una petición de acceso y envía los datos que el usuario ha proporcionado al servidor RADIUS. Es éste el que consulta su base de datos y comprueba si el usuario que ha realizado la llamada es en realidad quien dice ser. En ese caso, responde al NAS con una respuesta de aceptación de la llamada y, opcionalmente, con datos sobre el perfil del usuario (tipo de servicio, protocolo de conexión, IP asignada, ...). En caso contrario notifica al NAS que debe rechazar la petición de conexión. Opcionalmente, el servidor RADIUS guardará **logs (*informes o históricos del sistema para auditoría***) de las conexiones en las que estemos interesados.

Además de estos logs, RADIUS también puede hacer de servidor para registrar y almacenar todos los logs que acontecen en el NAT o RAS. Este paso está descrito e implementado en los routers de Cisco Systems con la AAA Authentication, Authoring y Accounting, quién es, qué puede hacer y auditoría respectivamente.

9.3 APLICACIONES DE SEGURIDAD: FIRMA DIGITAL Y CERTIFICADOS

Firmar un documento en realidad de forma tradicional, con bolígrafo, consiste en afirmar que lo escrito en el documento es cierto y con la estampación a mano de la firma, estando yo físicamente como persona, corroboro que soy yo quien afirma dicho documento. En este proceso se genera una copia adicional que sirve como resguardo para posterior comprobación.

En el procedimiento electrónico hacemos igual, de forma que la integridad del documento se obtiene a través de unos compendios y acuses de recibo, involucrando a las personas interesadas, y la validación e identificación del firmante se realiza a través de claves, que corroboran y permiten afirmar que yo soy quien digo ser. En este proceso participan o terceros que certifican nuestra identidad o terceros en los cuales tenemos confianza y a su vez ellos confian en otros. Esto lo veremos más detalladamente.

Pasemos a ver a continuación dichos elementos.

Resolución del control de integridad.

En muchas situaciones se requiere conocer la validez de un mensaje, esto es, conocer que no ha sido modificado por el camino y que nos llega tal y como fue escrito originalmente. A continuación describiremos un esquema de validación de integridad. Este esquema se basa en la idea de una función de dispersión unidireccional que toma una parte arbitrariamente grande de texto común y a partir de ella calcula una cadena de bits de longitud fija. En algunos contextos también se llama "**huella digital**" a este proceso de resumen. La idea de integridad, viene a ser la misma idea de un CRC añadido al final de una trama, de forma que comprueba la integridad de las tramas recibidas.

En el caso de control de integridad, la función para generar el resumen del documento con el cual comprobar la integridad también se llama función de dispersión o compendio de mensaje (message digest), que tiene tres propiedades importantes:

1. Dado un texto P, es fácil calcular su compendio de mensaje MD(P).

2. Dado un compendio de mensaje MD(P), es imposible encontrar P.

3. Nadie puede generar dos mensajes que tengan el mismo compendio de mensaje.

Para cumplir con el tercer criterio, la dispersión debe ser de cuando menos de 128 bits de longitud, y preferentemente mayor.

Los compendios de mensaje funcionan tanto en clave privada como en clave pública, siendo los de mayor uso el MD5 y el SHA.

Compendio de mensaje MD5.

El MD5 es la quinta de una serie de funciones de dispersión diseñadas por Ron Rivest. Opera alterando los bits de una manera tan complicada que cada bit de salida es afectado por cada bit de entrada. Su algoritmo es el siguiente:

- Se coge el mensaje original y se rellena hasta alcanzar una longitud de 448 módulo 512 bits, esto es, el mensaje debe tener una longitud en bits tal que al dividirla por 512 proporcione como resto de la operación el valor 448.

- A continuación se añade al mensaje la longitud original del mismo como un entero de 64 bits, por lo cual el mensaje total a codificar es un múltiplo de 512 bits.

- Se inicializa un buffer de 128 bits con un valor fijo.

- Ahora empieza el cálculo del compendio. Cada ronda toma un bloque de 512 bits de entrada y lo mezcla por completo con el buffer de 128 bits y los valores de una tabla construida a partir de la función matemática seno. Este proceso continúa hasta que todos los bloques de entrada se han consumido.

Una vez terminado el cálculo, el buffer de 128 bits contiene el valor del compendio de mensaje.

Compendio de mensaje SHA.

El SHA (Secure Hash Algoritm), fue desarrollado por la NSA y procesa los datos de entrada en bloques de 512 bits, pero a diferencia del MD5 genera un compendio de mensaje de 160 bits. Su algoritmo es el siguiente:

- Se coge el mensaje original y se rellena hasta alcanzar una longitud de 448 módulo 512 bits, esto es, el mensaje debe tener una longitud en bits tal que al dividirla por 512 proporcione como resto de la operación el valor 448.

- A continuación se añade al mensaje la longitud original del mismo como un entero de 64 bits, por lo cual el mensaje total a codificar es un múltiplo de 512 bits.

- Se inicializa un buffer de 160 bits con un valor fijo.

- Ahora empieza el cálculo del compendio. Cada ronda toma un bloque de 512 bits de entrada y lo mezcla con el buffer de 160 bits, utilizando 80 rondas para cada bloque de entrada y modificando cada 20 rondas las funciones de mezcla del bloque y el buffer. Este proceso continúa hasta que todos los bloques de entrada se han consumido.

Una vez terminado el cálculo, el buffer de 160 bits contiene el valor del compendio de mensaje. El SHA es 2^{32} veces más seguro que el MD5, pero es más lento su cálculo que el cálculo del MD5.

Hasta la fecha, tanto el MD5 como el SHA se han mostrado inviolables, pues aún con ataques sofisticados ideados (como el ataque de cumpleaños), se tardarían más de 500 años en calcular los compendios de dos cartas con 64 variantes cada una, e incluso entonces no se garantiza una equivalencia.

Resolución del repudio: Firmas digitales.

La validación de identificación y autenticidad de muchos documentos legales, financieros y de otros tipos se determina por la presencia o ausencia de una firma manuscrita autorizada. Para que los sistemas computerizados de mensajes reemplacen el transporte físico de papel y tinta, debe encontrarse una solución a estos problemas.

El problema de inventar un reemplazo para las firmas manuscritas es difícil. Básicamente, lo que se requiere es un sistema mediante el cual una parte pueda enviar un mensaje “firmado” a otra parte de modo que:

1. El receptor pueda verificar la identidad proclamada del transmisor.

2. El transmisor no pueda repudiar después el contenido del mensaje.

3. El receptor no haya podido confeccionar el mensaje él mismo.

El primer requisito es necesario, por ejemplo, en los sistemas financieros. Cuando la computadora de un cliente ordena a la computadora de un banco que compre una tonelada de oro, la computadora del banco

necesita asegurarse de que la computadora que da la orden realmente pertenece a la compañía a la que se le aplicará el débito.

El segundo requisito es necesario para proteger al banco contra fraudes. Supongamos que el banco compra una tonelada de oro, e inmediatamente después cae el precio del oro. Un cliente deshonesto podría demandar al banco, alegando que nunca emitió una orden para comprar el oro. Cuando el banco presenta el mensaje ante el juez, el cliente niega haberlo enviado.

El tercer requisito es necesario para proteger al cliente en el caso de que el precio del oro suba y que el banco trate de falsificar un mensaje firmado en el que el cliente solicitó un lingote de oro en lugar de una tonelada.

Al igual que la criptografía, las firmas digitales se dividen en dos grandes grupos, firmas de clave secreta y firmas de clave pública.

Firmas de clave secreta.

Un enfoque de las firmas digitales sería tener una autoridad central que sepa todo y en quien todos confíen, digamos X. Cada usuario escoge entonces una clave secreta y la lleva personalmente a las oficinas de X. Por tanto, sólo A y X conocen la clave secreta de A, K_A, etc.

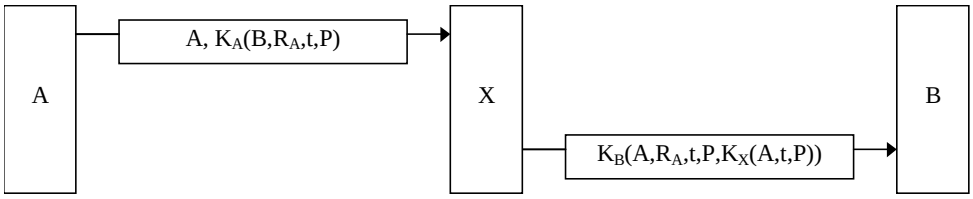


Figura 18: firma con clave secreta compartida con centro de distribución de claves

Cuando A quiere enviar un mensaje de texto normal firmado, P, a su banquero, B, genera $K_A(B,R_A,t,P)$ y lo envía como se muestra en la figura anterior. X ve que el mensaje es de A, lo descifra y envía un mensaje a B como se muestra. El mensaje a B contiene el texto normal del mensaje de A y también el mensaje firmado $K_X(A,t,P)$, donde t es una marca de tiempo. Ahora B atiende la solicitud de A.

Si ahora A niega el envío del mensaje, cuando el caso llega al juez y A niegue haber enviado a B el mensaje, B indica que el mensaje vino de A y no de un tercero C pues X no hubiera aceptado un mensaje de A a menos que estuviese cifrado con K_A, por lo que no hay posibilidad de que C enviará a X un mensaje falso en nombre de A. B además presenta la prueba $K_X(A,t,P)$. Entonces el juez pide a X (en quien todo el mundo confía) que descifre la prueba. Cuando X testifica que B dice la verdad el caso queda resuelto.

Un problema potencial del protocolo de firma anterior es que C repita cualesquiera de los dos mensajes. Para minimizar este problema, se usan en todos los intercambios marcas de tiempo. Es más, B puede revisar todos los mensajes recientes para ver si se usó R_A en cualquiera de ellos. De ser así, el mensaje se descarta como repetición. Nótese que B rechazará los mensajes muy viejos con base en la marca de tiempo. Para protegerse contra ataques de repetición instantánea, B simplemente examina el R_A de cada mensaje de entrada para ver si un mensaje igual se recibió de A durante el tiempo de validez de la marca temporal. Si no, B puede suponer con seguridad que ésta es una solicitud nueva.

Firmas de clave pública.

Un problema estructural del uso de la criptografía de clave secreta para las firmas digitales es que todos tienen que confiar en X. Es más, X lee todos los mensajes firmados. Los candidatos más lógicos para operar el servidor X son el gobierno, los bancos y los abogados. Estas organizaciones no tiene porqué inspirar confianza completa a todos los ciudadanos. Por tanto, sería bueno si la firma de documentos no requiriese una autoridad confiable.

Afortunadamente, la criptografía de clave pública puede hacer una contribución importante aquí. Supongamos que los algoritmos públicos de cifrado y descifrado tienen la propiedad de que E(D(P))=P además de la propiedad normal de D(E(P))=P (el RSA tiene esta propiedad por lo que el supuesto es razonable). Suponiendo que éste es el caso, A puede enviar un mensaje de texto normal firmado y cifrado, P, a B transmitiendo $E_B(D_A(P))$. Firmado por $D_A(P)$ y cifrado por $E_B()$, de forma que sólo B podrá leerlos. Nótese que A conoce su propia clave de descifrado (privada), D_A, así como la clave pública de B, E_B, por lo cual la construcción de este mensaje es algo que A puede hacer.

Cuando B recibe el mensaje, lo transforma usando su clave privada, como es normal, produciendo $D_A(P)$, como se muestra en la figura siguiente:

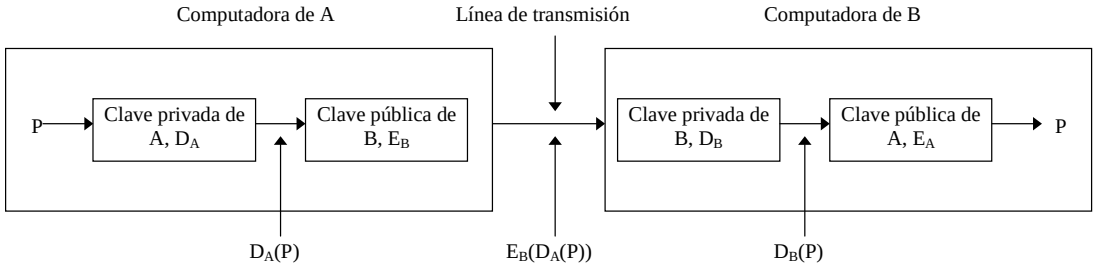


Figura 19: firma digital con clave pública

B almacena este texto en un lugar seguro y lo descifra usando E_A para obtener el texto normal original.

En todo el proceso, en resumen lo que se ha realizado es $E_A(D_B(E_B(D_A(P))))$.

Para ver cómo funciona la propiedad de firma, supongamos que A niega haber enviado el mensaje P a B. Cuando el caso llega al juez, B puede presentar tanto P como $D_A(P)$. El juez puede comprobar fácilmente que B tiene un mensaje válido cifrado por D_A con solo aplicarle E_A. Puesto que B no conoce la clave privada de A, la única forma en que B pudo haber adquirido el mensaje cifrado con ella sería que A en efecto lo hubiera enviado.

Sin embargo existen dos problemas, por un lado B puede demostrar que un mensaje fue enviado por A siempre y cuando D_A permanezca en secreto. Si A divulga su clave secreta, el argumento ya no se mantiene. Por otro lado, si A decide cambiar su clave, algo legal y probablemente buena idea de forma periódica, cuando se aplique la actual E_A a $D_A(P)$ no se obtiene P. En consecuencia, parece que sí que se requiere alguna autoridad para registrar todos los cambios de clave y sus fechas.

En principio cualquier algoritmo de clave pública puede usarse para firmas digitales. El estándar de facto de la industria es el algoritmo RSA y muchos productos de seguridad lo usan.

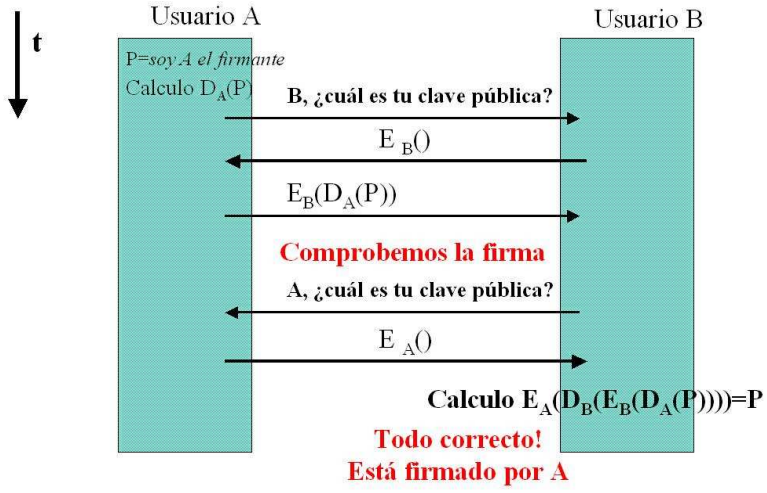


Figura 20: protocolo de firma digital con clave pública

Con todo ello, el protocolo quedaría tal como se ve en la figura 20.

Un ejemplo de comprobación de utilización de firma digital lo podemos ver en la comprobación de paquetes software "rpm" *en algunas distribuciones de Linux. Para comprobar la firma digital la aplicación* ***rpm*** *posee la opción "checksig": "rpm –checksig X.i386.rpm*" siendo *X.i386.rpm* el paquete a instalar. Esta opción en *rpm* permite comprobar la integridad del paquete y el origen. La **integridad** se comprueba con la firma basada con **md5.** La **comprobación del origen** (validación) se realiza con la clave pública de GNU (si se dispone). Si no disponemos de dicha firma, la comprobación de origen no puede realizarse y con ello, la GPG (*Gnu Privacy Guard*) no podrá efectuarse:

$$\mathbf{E_{GNU}(D_{GNU}(}\textit{compendio-rpm}\mathbf{))}=\textit{compendio-rpm}$$

Aplicaciones seguras

Visto todo el conjunto de protocolos, tanto para intercambiar información cifrada, autentificar y validar, generar soluciones al no repudio y el proceso de firma digital, pasemos a ver aplicaciones que catalogamos como seguras que hacen uso de estos elementos o están vinculadas con ellas.

Aplicaciones seguras y uso de certificados

Uno de los problemas que parecen en la distribución de claves públicas, es la propia distribución de dicha clave, que como hemos visto puede sufrir del ataque de alguien en medio. Una solución es la aplicación de métodos de interbloqueo, pero la solución más adoptada y estandarizada en la utilización de certificados.

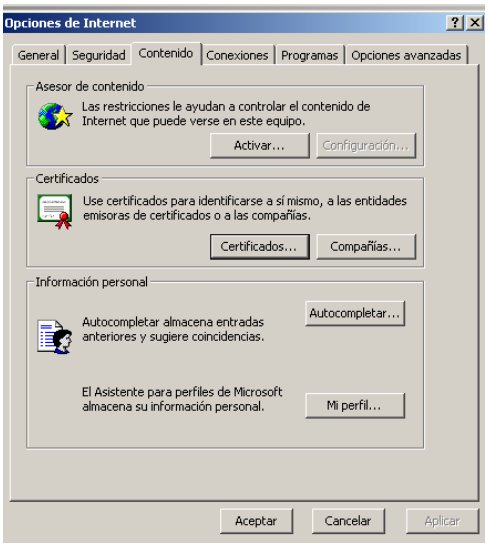


Figura 20: Configuración de Certificados en Herramientas: Opciones de Internet: Contenidos: Certificados en Internet Explorer 5.0

El **certificado digital** es un vínculo entre una clave pública y una identidad, que se consigue mediante una firma digital por una tercera parte o autoridad de certificación (***CA****: Certificaciont Authority o Autoridad de Certificación*) comprobando la integridad. Se considera como un objeto firmado con: *identidad del sujeto, clave, periodo de validez, identidad emisor*, ... La Fábrica Nacional de Moneda y Timbre (FNMT), Verisign, Deutsche Telecom., Microsoft, Xcert,... son diferentes autoridades de certificación.

A modo de ejemplo, en el caso de pedir un certificado para un ciudadano español, su certificado sería un archivo, firmado con la clave privada de CA con la identidad, la clave pública del dicha identidad, atributos varios y compendio de dicha información:

$$\mathbf{D_{CA}(identidad}, \mathbf{clave}, \mathbf{atributos}, \textit{compendio}\{\textit{identidad, clave, atributos}\})$$

La mayoría de navegadores disponen ya en la distribución, en los propios CDs de instalación, un directorio lleno de **certificados raiz**, es decir, certificados de las propias autoridades de certificación firmados por ellas mismas. Vemos una lista de los certificados raiz más habituales en los navegadores:

Enviado a	***Emitido por***
ABA.ECOM Root CA	ABA.ECOM Root CA
AC del Colegio Nacional de Correduria Publica Mexicana, A.C.	AC del Colegio Nacional de Correduria Publica Mexicana, A.C.
Baltimore EZ by DST	Baltimore EZ by DST
Belgacom E-Trust Primary CA	Belgacom E-Trust Primary CA
C&W HKT SecureNet CA Class A	C&W HKT SecureNet CA Class A
CA 1	CA 1
Certiposte Classe A Personne	Certiposte Classe A Personne
Certiposte Serveur	Certiposte Serveur
Certisign - Autoridade Certificadora - AC2	Certisign - Autoridade Certificadora - AC2
Class 1 Primary CA	Class 1 Primary CA
Copyright (c) 1997 Microsoft Corp.	Copyright (c) 1997 Microsoft Corp.
Deutsche Telekom Root CA 1	Deutsche Telekom Root CA 1
DST (ANX Network) CA	DST (ANX Network) CA
DST (NRF) RootCA	DST (NRF) RootCA
DST-Entrust GTI CA	DST-Entrust GTI CA
Entrust.net Secure Server Certification Authority	Entrust.net Secure Server Certification Authority
Equifax Secure Certificate Authority	Equifax Secure Certificate Authority
FNMT Clase 2 CA	**FNMT Clase 2 CA**
GlobalSign Root CA	GlobalSign Root CA
http://www.valicert.com/	http://www.valicert.com/
IPS SERVIDORES	IPS SERVIDORES
Microsoft Authenticode(tm) Root Authority	Microsoft Authenticode(tm) Root Authority
Microsoft Root Authority	Microsoft Root Authority
NetLock Uzleti (Class B) Tanusitvanykiado	NetLock Uzleti (Class B) Tanusitvanykiado
PTT Post Root CA	PTT Post Root CA
Saunalahden Serveri CA	Saunalahden Serveri CA
Secure Server Certification Authority	Secure Server Certification Authority
SecureNet CA Class A	SecureNet CA Class A
SecureSign RootCA	SecureSign RootCA
SERVICIOS DE CERTIFICACION - A.N.C.	SERVICIOS DE CERTIFICACION - A.N.C.
SIA Secure Client CA	SIA Secure Client CA
SIA Secure Server CA	SIA Secure Server CA
Swisskey Root CA	Swisskey Root CA
TC TrustCenter Class CA	TC TrustCenter Class CA
TC TrustCenter Time Stamping CA	TC TrustCenter Time Stamping CA
Thawte Personal CA	Thawte Personal CA
UTN - DATACorp SGC	UTN - DATACorp SGC
UTN-USERFirst-Client Authentication and Email	UTN-USERFirst-Client Authentication and Email
VeriSign Commercial Software Publishers CA	VeriSign Commercial Software Publishers CA
VeriSign Individual Software Publishers CA	VeriSign Individual Software Publishers CA
VeriSign Trust Network	VeriSign Trust Network
Xcert EZ by DST	Xcert EZ by DST

La **autoridad de certificación** es reconocida y aceptada por todos, e imposible de suplantar. Por regla general, por seguridad no se trabaja directamente con la autoridad de certificación, si no con un intermediario o autoridad de registro.

La **autoridad de registro** atiende las peticiones de certificado y proporciona diferentes métodos de petición de certificados.

Para implementar el sistema de autenticación en base a certificados a través de Autoridad de Certificación (CA) han de considerarse los siguientes elementos:

- Una política de certificación, incluyendo el ámbito de actuación (identificando los elementos a certificar, tanto personas como procesos, como servidores y/o clientes) así como la relación de la CA con otras CA, por regla general de forma jerárquica
- Un certificado de la CA, de otra CA superior u homóloga que asegure quien dice ser y valida su clave pública
- Los certificados de los usuarios (p.ej X.509), así como el procedimiento de certificación y de revocación. La revocación permite consultar directamente qué certificados no son válidos
- Los protocolos de autenticación, gestión y obtención de certificados, así como de distribución de ellos son como veremos a continuación:

–o por bases de datos (p.ej directorio X.500) o comúnmente llamados servidores de directorios, que veremos a continuación

–o bien directamente del usuario en tiempo de conexión (WWW con SSL, Secure Socket Layer)

Por tanto para la puesta en marcha de una CA se requiere generar un par de claves (tanto privada como pública), proteger la clave privada y generar el certificado de la propia CA a través de otra CA.

El **certificado raíz** es un certificado emitido de la CA para sí misma con su clave pública, para comprobar certificados emitidos por ella. Se suele instalar previamente dicho certificado en el navegador para poder utilizar los certificados de dicha CA. Los navegadores llevan por defecto muchos de ellos.

Lista de certificados revocados (o CRL *Certificate Revocation List***)** es una lista donde se recogen todos los certificados de la CA *dados de baja por caducidad o por problemas varios* como que se haya hecho pública la clave privada de un usuario, y por tanto cualquier firma emitida con posterioridad a la revocación no tiene validez. Este documento también es firmado por la propia CA.

Pero, esta infraestructura de autoridades en muchas ocasiones no se dispone, ni de Autoridades de Certificación ni de Registro. Con este planteamiento inicial una solución tomada estriba en la confianza de los propios usuarios entre ellos. Este tipo de redes se llaman **redes de confianza**. Por ejemplo, si Juan confía plenamente en Luis y Luis ha aceptado la identificación de Pedro, Juan podría inicialmente aceptar a Pedro, porque Luis es de su confianza. Una implementación de este tipo de certificados lo ofrece PGP como veremos en el apartado de correo seguro.

Pasemos a ver ahora la estructura de los certificados X.509, ampliamente utilizados actualmente**Certificado X.509**

X.509 es el protocolo que se utiliza para certificar las claves públicas, con lo que los usuarios pueden intercambiar datos de manera segura. X.509 está basado en criptografía asimétrica y firma digital y se emplea para autentificar la información en redes externas, en redes internas, en el correo electrónico y en general en aplicaciones que requieran autenticación.

Para ello, el certificado se cifra con la clave privada de la CA y todos los usuarios poseen la clave pública del CA.

El protocolo X.509 fue creado por la UIT para servir al X.400 (correo electrónico de OSI) y su origen se encuentra en el servicio de directorio X.500. En X.509 se define un framework (una capa de abstracción) para suministrar servicios de autenticación a los usuarios del directorio X.500.

El objetivo de este protocolo es asegurar la integridad, la privacidad y el no repudio de los mensajes.

X.509 se utiliza en SSL en los navegadores (https), PEM en el correo electrónico seguro, S/MIME, SET (Secure Electronic Transaction) que define una arquitectura para E-CommerceLos campos del X.509 escritos en ASN1 son:Versión: La del protocolo X.509 (actualmente versión 3)Número de serie: Es un número asignado por el CA y que identifica de manera única el certificado.Algoritmo de la firma del certificado: Identifica el algoritmo utilizado para firmar el certificado.Autoridad de certificación (CA): Es el nombre de la CA.Fecha de inicio y final: tiempo de validezUsuario: Es el nombre del usuario.Clave pública: Es la clave del usuario.Identificador único del CA: Es el número que identifica al CA. Es único en el mundo.Identificador único del usuario: Es el número que identifica al usuario para todos sus certificados.Extensiones: Si hay extensiones de la información

- Firma de la CA: Firma todos los campos anteriores empleando, para ello, su clave privada.

Servidor de directorios.

La idea de un directorio es el de un listín de teléfonos. Pero si queremos utilizarlo en el ámbito para organizar la información de una institutción etc, requiere que sea electónico, para permitir modificaciones en tiempo real, búsquedas aproximadas, etc.

Los servidores de directorio están basados en el protocolo X.500, primer protocolo de servidor de directorios, conocido como padre, y está pensado para guardar todo tipo de información referente a una institución, tanto de personas (datos personales, correo, teléfono), como de objetos (salones de actos, sala

de reuniones,...), pero X.500 debido a que es complicado, se pensó en una simplificación: LDAP (Lighweight Directory Access Protocol)(RFC 1478 y 2251).

La relación de los servidores de directorios con los certificados, es que tras largos debates, este tipo de servidores cumplen las características idóneas para albergar los certificados de los usuarios en una corporación.

LDAP es una simplificación de X.500 y se considera como una base de datos jerárquica orientada a objetos, a la cual se realizan consultas (búsquedas e inserciones) de registros. Estos registros pueden contener cualquier tipo de información, pero generalmente en una red incluyen información con usuarios y perfiles, para autentificar de forma muy rápida. Los mecanismos de consulta de esta base de datos es propietario, es decir no tiene similitud con SQL (Structured Queueing Language).

LDAP y en general los servidores de directorios han aprovechado la rapidez en las lecturas, frente a las bases de datos. Las bases de datos de propósito general están optimizadas para lecturas/escrituras, mientras que los servidores de directorios fundamentalmente las peticiones que van a atender son de lectura. Otra opción a las bases de datos para poder gestionar esta información, el servicio de directorios es el servicio DNS. Sin embargo hubo detractores a utilizar DNS como servidor de directorios y por tanto se optó por crear un servidor específico e independiente.

LDAP está basado en arquitectura de cliente/servidor y puede utilizar Kerberos en el proceso de autenticación. Los servicios ofrecidos por los servidores se sirven en el puerto 389 normalmente, aunque se puede configurar, además los servidores permiten varias peticiones en curso.

LDAP se utiliza principalmente para realizar consultas que permitan averiguar qué permisos puede tener un usuario dentro de la red controlada por servidores LDAP para ejecutar ciertos servicios. Veamos un ejemplo de validación de permisos. En este caso vamos a considerar el acceso de los profesores a las actas por red de la Universitat de Valencia, acceso al control de actas. El protocolo implementado es como sigue:

1. el cliente (usuario o profesor) quiere conectarse al gestor de actas
2. tras la identificación del cliente (usuario), el gestor de actas consulta al servidor LDAP para averiguar los permisos del cliente (usuario) en el acceso a la gestión de actas.
3. según los permisos, atiende al cliente

En esta base de datos, las clases no tienen métodos, sólo atributos, y los atributos son propiedades de una clase y cada atributo es un nombre (tipo) y una lista de valores. El nombre del atributo es una cadena de caracteres o un OID (Object Identifier) y se pueden estructurar jerárquicamente permitiendo herencias. Ejemplo de estos atributos son:

CN	Nombre habitual o canónic name
O	Organización
OU	Departamento
C	País
MAIL	Dirección de correo electrónico
PHONE	Número de teléfono
DC	Componente de dominio
DN	Nombre Distinguido y cada entrada tiene uno, único y propio

Las aplicaciones más frecuentes de LDAP son muy variadas, desde directorio de correo (email) para encontrar la dirección de correo de alguien o encontrar el certificado digital de alguien, pasando por una libreta de direcciones y teléfonos, o haciendo una sustitución de NIS (difusión de ficheros de contraseñas).

Otra aplicación de LDAP puede ser la integración de todos los recursos de la organización: personal, utilización y disponibilidad de salas de Reuniones, cuentas de usuario. Por ejemplo una forma de identificar de forma unívoca a una asignatura, es a través de la cadena observada en la figura.

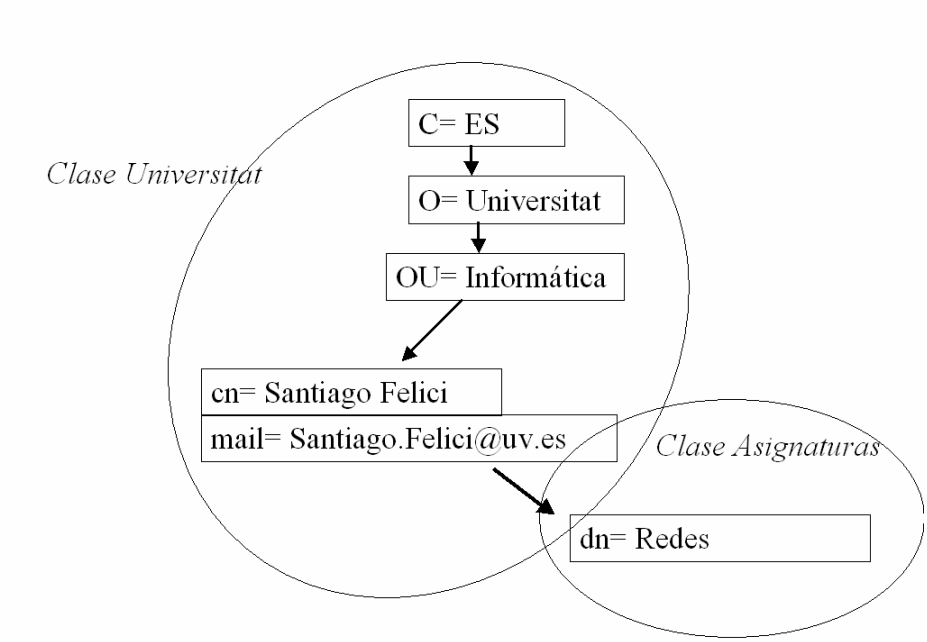


Figura 21: Ejemplo de jerarquía de clases y sus atributos en LDAP

El acceso de LDAP mediante JAVA, se realiza por JNDI (Java Naming Directory Interface) y como hemos dicho es una sintaxis propia, diferente a SQL. Existe un formato para exportar (a X.500) información entre servidores de directorios, conocido como LDIF.

Las distribuciones de LDAP más populares son:
-OpenLDAP (versión gratuita http://www.openldap.org)
-Iplanet Directory Server (versión comercial http://www.iplanet.com)

Aplicaciones seguras para confidencialidad del correo electrónico.

Cuando un mensaje de correo electrónico se envía entre dos sitios distantes, generalmente transitará por docenas de máquinas en el camino. Cualquiera de éstas puede leer y registrar el mensaje para su uso posterior, por lo que la confidencialidad es inexistente. No obstante, mucha gente quiere poder enviar correo electrónico que el destinatario pueda leer, pero nadie más. Este deseo ha estimulado a varias personas y grupos a aplicar principios criptográficos al correo electrónico para producir correo seguro. Veremos a continuación los compendios de mensajes, que permiten validar un mensaje, pasando a continuación a explicar dos de sistemas de correo electrónico seguro de amplio uso en Internet: PGP y PEM.

PGP: Bastante buena confidencialidad.

El primer sistema de correo electrónico seguro que veremos es PGP. PGP (Pretty Good Privacy, bastante buena confidencialidad) es el producto de una sola persona, Phil Zimmermann. PGP es un paquete completo de seguridad de correo electrónico que proporciona confidencialidad, validación de identificación, firmas digitales y compresión basado en el **envoltorio digital**, es decir un procesamiento completo que se le realiza al correo a mandar para que cumpla los requisitos fijados por PGP. El paquete completo, incluido código fuente, se distribuye por Internet de forma gratuita, por lo que es el de más amplio uso hoy en día.

El PGP intencionadamente usa algoritmos criptográficos existentes en lugar de inventar nuevos; PGP se basa en los algoritmos de encriptación RSA e IDEA y el compendio de mensaje MD5, algoritmos que han resistido análisis extensos de otras personas y no fueron diseñados ni influidos por ninguna agencia gubernamental que estuviera tratando de debilitarlos. Para ver como funciona PGP, consideremos la siguiente figura:

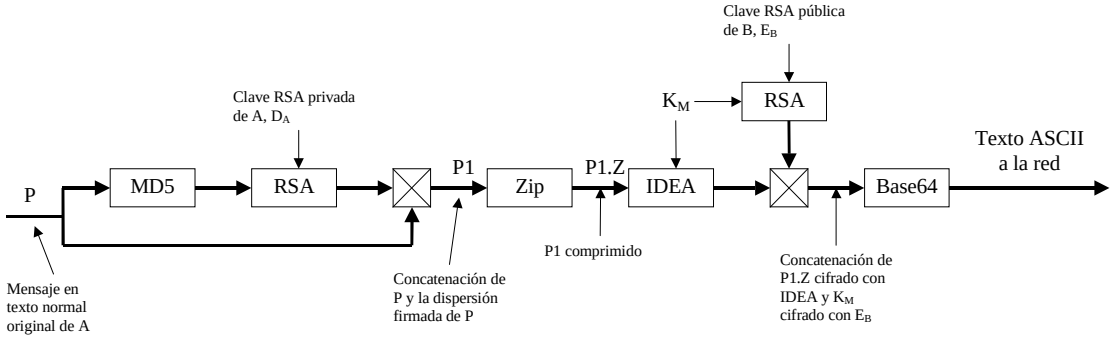


Figura 22: diagrama de bloques de cifrado en PGP

Aquí, A quiere enviar un mensaje de texto normal firmado, P, a B de una manera segura. Tanto A como B tienen claves RSA privadas (D_X) y públicas (E_X). Supongamos que cada uno conoce la clave pública del otro.

A comienza por invocar el programa PGP en su computadora. El PGP primero dispersa su mensaje, P, usando MD5, y luego cifra la dispersión resultante con su clave RSA privada, D_A. Cuando B recibe el mensaje, puede descifrar la dispersión con la clave pública de A y comprobar que es correcta.

La dispersión cifrada y el mensaje original ahora están concatenados en un solo mensaje P1, y comprimidos mediante el programa ZIP. Llamemos a la salida de este paso P1.Z.

A continuación, el PGP solicita a A una entrada al azar. Tanto el contenido como la velocidad de tecleo se usan para generar una clave de mensaje IDEA de 128 bits, K_M (llamada clave de sesión). K_M se usa ahora para cifrar P1.Z con IDEA en modo de realimentación de cifrado. Además, K_M se cifra con la clave pública de B, E_B. Estos dos componentes se concatenan y convierten a base64. El mensaje resultante contiene entonces sólo letras, dígitos y los símbolos +, / e =, lo que significa que puede ponerse en un cuerpo RFC 822 y esperar que llegue sin modificación.

Llamaremos todo este proceso realizado con el correo como envoltorio digital.

Cuando B recibe el mensaje deshace el envoltorio y para ello, invierte la codificación base64 y descifra la clave IDEA usando su clave RSA privada. Con la clave así obtenida, B descifra el mensaje para obtener P1.Z. Tras descomprimir esto, B separa el texto normal de la dispersión cifrada y descifra la dispersión usando la clave pública de A. Si la dispersión del texto normal concuerda con su propio cálculo MD5, sabe que P es el mensaje correcto y que vino de A.

La administración de claves ha recibido mucha atención en el PGP, puesto que es el punto débil de todos los sistemas de seguridad. Cada usuario mantiene localmente dos estructuras de datos: un anillo con claves privadas y un anillo de claves públicas. Estos anillos se llaman **repositorios de llaves o llaveros**. El anillo de claves privadas contiene uno o más pares de clave privada-clave pública personales. La razón para reconocer varios pares por usuario es permitir a los usuarios cambiar sus claves públicas periódicamente o cuando se piensa que una está comprometida, sin invalidar los mensajes que actualmente están en preparación o en tránsito. Cada par asociado tiene un identificador, para que el remitente de un mensaje pueda indicar al destinatario la clave pública usada para cifrarlo. El anillo de claves públicas contiene claves públicas de los corresponsales del usuario; se requieren para cifrar las claves de mensaje asociadas a cada mensaje. Cada entrada del anillo de claves públicas contiene no sólo la clave pública, sino también su identificador y una indicación del nivel de confianza del usuario en la clave.

PEM: Correo con confidencialidad mejorada.

En contraste con el PGP, el PEM (Privaty Enhanced Mail, correo con confidencialidad mejorada), es un estándar oficial de Internet y se describe en cuatro RFC, del RFC 1421 al RFC 1424. De manera muy

general, el PEM cubre el mismo territorio que el PGP: confidencialidad y validación de identificación, para sistemas de correo electrónico basados en el RFC 822.

Los mensajes enviados usando PEM primero se convierten a una forma canónica para que puedan tener las mismas convenciones de espacios blancos (por ejemplo, tabuladores, espacios al final) y el uso de retornos de carro y avances de línea. Esta transformación se lleva a cabo para eliminar los efectos de los agentes de transferencia de mensajes que modifican los mensajes que no son de su gusto. Sin canonización, tales modificaciones pueden afectar las dispersiones de los mensajes en sus destinos.

A continuación, se calcula a dispersión del mensaje usando MD5. Después se cifra la concatenación con la dispersión y el mensaje usando DES. El mensaje cifrado puede entonces codificarse con codificación base64 y transmitirse al destinatario.

Como en el PGP, cada mensaje se cifra con una clave de una sola vez que se incorpora en el mensaje. La clave puede protegerse mediante RSA o con DES triple. En la práctica, se usa RSA pues el PEM no indica cómo se debe hacer la administración de claves con DES.

La administración de claves es más estructurada que en el PGP. Las claves se certifican mediante autoridades certificadoras que producen certificados indicando el nombre del usuario, su clave pública y la fecha de expiración de la clave. Cada certificado tiene un número de serie único que lo identifica. Los certificados incluyen una dispersión MD5 firmada por la clave privada de la autoridad certificadora.

Aplicaciones seguras basadas en tarjetas inteligentes y PKI (Public Key Infraestructure)

Utilización y tipo de tarjetas.

Las tarjetas son un elemento fundamental en el control de acceso a los edificios y permiten la identificación a través de ella, bien con códigos PIN (Personal Identification Number) o/y bien a través de fotografías escaneadas en su superficie. La tarjeta (genérica) es un dispositivo de plástico de dimensiones determinadas, capaz de almacenar y, en algunos casos, procesar información de manera segura. Fue inventado por *Roland Moreno* (periodista francés) a finales de los 70 y actualmente su complejidad es tal que utilizan *tecnología VLSI* con microprocesador y sistemas de ficheros cifrado.

Las utilidades que puede tener son almacenamiento y procesamiento de datos confidenciales, estado de las cuentas de crédito, historiales médicos (identificación de pacientes), números de identificación personal, claves privadas (control de acceso), dinero electrónico (micropagos)
Las propiedades que tienen las tarjetas son respecto a:
•seguridad, en el almacenamiento de la información (sistemas de fichero) y en las operaciones que realizan (clave simétrica y clave asimétrica)
•portabilidad de información, como historiales, claves, ...
•coste asequible
•utilizan protocolos propietarios para su programación, como para la comunicación lector-tarjeta, incluso la comunicación entre lectores entre sí, aunque en todos los protocolos existe una función básica para el procesado de revocación que consiste en la gestión de dos listas, *lista blanca* o de autorizados y *lista negra* o de personas excluidas (caducados, extraviados, revocados...)

El tipo de tarjetas existentes en el mercado hoy en día son:

- Magnéticas (ISO 7811): son muy extendidas, pero son de fácil fraude, con poco espacio de memoria 180 bytes. Son utilizada en accesos e identificación

- Con memoria (ISO 7816): con chip integrado que almacena información de forma “segura”. Son utilizadas como monederos, ejemplo tarjetas para teléfonos públicos. Se las conoce como *Chipcards*

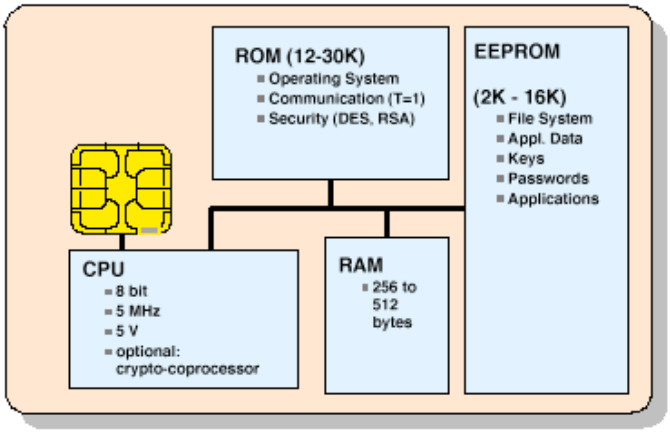


Figura 23: ejemplo de tarjeta inteligente con microprocesador, dónde la CPU está en la parte de acceso al bus interno de la tarjeta donde conecta la ROM, RAM y EEPROM

- **Con microprocesador** (ISO 7816/ISO 14443): son tarjetas seguras y pueden ser programadas, con memoria de entre 2-4kbytes. Se consideran multiaplicación, pero tienen el inconvenientes que sólo aceptan aplicaciones desarrolladas por el propio fabricante. Son utilizadas en módulos SIM, comercio electrónico, donde el usuario de identifica con PIN (personal identification number). Son conocidas como *SmartCards*

- **JAVA cards**: formada por capas, donde las aplicaciones (applets) y el S.O. Son independientes, utilizan un subconjunto de lenguaje JAVA, Java Card Api 2.1, permiten aplicaciones independiente del hardware escritas en lenguaje de alto nivel, son 10 a 200 veces mas lentas que las anteriores. Este tipo de tarjetas incorpora una máquina virtual JAVA y te permite la programación de Applets.

La utilización de las tarjetas y su aplicación dentro de una infraestructura de red, permite gestionar de forma segura muchas transacciones y el control de acceso a todos los servicios en una red. Este mecanismos o infraestructura se conoce como Infraestructura de Clave Pública (PKI).

PKI: Public Key Infraestructure

Una PKI se define como la infraestructura de red formada por servidores y servicios que en base a claves públicas gestionan de forma segura todas las transacciones realizadas a través de la red. Actualmente está en fase de estandarización con un RFC. Las operaciones básicas realizadas en una PKI son la certificación (medio por el cual las claves se publican) y la validación, a través de revocación y autenticación. Las claves públicas de los servicios, usuarios, aplicaciones, clientes, ... pueden ubicarse en servicios de directorios, en autoridades de certificación, en tarjetas inteligentes, ... Los fabricantes de PKI más representativos son RSA Labs, Verisign GTE Cyber Trust, Xcert, Netscape,....

La PKI como su nombre indica hace uso de los algoritmos de cifrado con clave pública y las aplicaciones (u operaciones) que se realizan con ellas son:

1. para cifrar, por ejemplo en el caso que A quiere mandar P a B, para ello utiliza la clave pública de B, enviando $E_B(P)$ y B utilizando su clave privada realiza el paso inverso D_B ($E_B(P)$)
2. para firmar y autentificar, por ejemplo en el caso que A quiere mandar P a B y firmarlo. Entonces para ello A manda {P, $D_A(P)$} utiliza su clave privada y B utilizando la clave pública de A, realiza la comprobación E_A ($D_A(P)$)

Una de las funciones importantes de la PKI es la validación de la información de los certificados. Para ello el usuario dispone de dos formas principales de validación, o preguntar por la validez a la CA (validación conocida como on line) y/o examinar el periodo de validez incluido en el certificado (off-line). El problema en esta última forma de validación, estriba en la complejidad de gestionar la revocación de certificados. La información del certificado es inválida si, la clave privada de la entidad se compromete, o si los datos de la entidad cambian.

Si la validación se realiza on-line, la revocación resulta sencilla. Si la validación se realiza off-line, la revocación se realiza mediante métodos como las listas de certificados revocados o CRLs (***Certificate Revocated List***), que es una lista de los certificados revocados, publicada y firmada periódicamente por la

CA. El usuario chequea esta lista para comprobar la validez de los certificados.Las distribución y gestión de las claves dentro de las PKI es lo que se llama repositorios (llaveros) de Certificados y son los elementos necesarios para almacenar los certificados de los usuarios. Para poder almacenarlos se suele utilizar servidores de directorios basados en X.500 y LDAP. Cabe destacar que los principales navegadores llevan soporte para servidores de directorio y también pueden gestionar las claves directamente con SSL (Secure Socket Layer) con total seguridad.
Las PKI pueden estar basadas en certificados X.509 y/o el soporte de repositorio de llaves (llaveros) de PGP, ya que éste incluye opción para clave pública, bien por confianza directa, es decir reside en los propios usuarios y confianza jerárquica, a través de una CA.

Aplicaciones seguras: Secure Socket Layer

Secure Socket Layer (SSL) es una pila de protocolos diseñada por Netscape y permite configurar al navegador en conexión segura. En el caso de una conexión web, aparece el protocolo ***https***. La pila de protocolos SSL está localizada sobre el nivel de transporte y es independiente del protocolo superior (interfaz similar a BSD Sockets). SSL protege del ataque de alguien al medio a través de certificados digitales X.509v3.

Obviamente el establecer sesiones seguras con SSL en la práctica reduce las prestaciones de un sistema normal, debido al cifrado en el establecimiento de la conexión

SSL es un conjunto de protocolos que se establecen entre la capa de transporte y aplicación en el modelo TCP/IP. Estos protocolos son Alert Protocol (AP), Handshake Protocol (HP), Change Cipher Spec (CCSP) y Application Data Protocol (ADP), que son protocolos que se ubican al mismo nivel, por debajo de la aplicación en sí. Por debajo de todos ellos, encargado de cifrar y fragmentar los paquetes se encuentra el Record Protocol (RP). Con esta pila de protocolos, cada sesión tiene sus parámetros criptográficos y no varían durante la sesión, salvo mensajes Change Cipher Spec (CCSP).

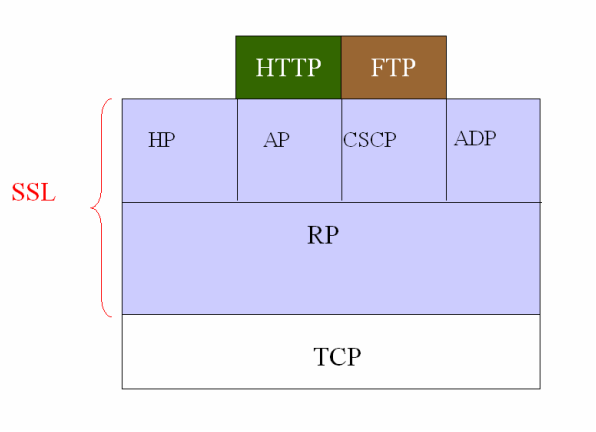


Figura 24. Pila de protocolos en SSL

Cuando el navegador ha negociado una conexión segura por SSL, es decir opera en modo de conexión segura, aparece el dibujo de un candado o llave en la barra inferior.

Los puertos utilizados para las aplicaciones que utilizan SSL son 443 para HTTPS, 465 para SSMTP, 563 para SNEWS, 636 para SS-LDAP, 995 para SPOP3.

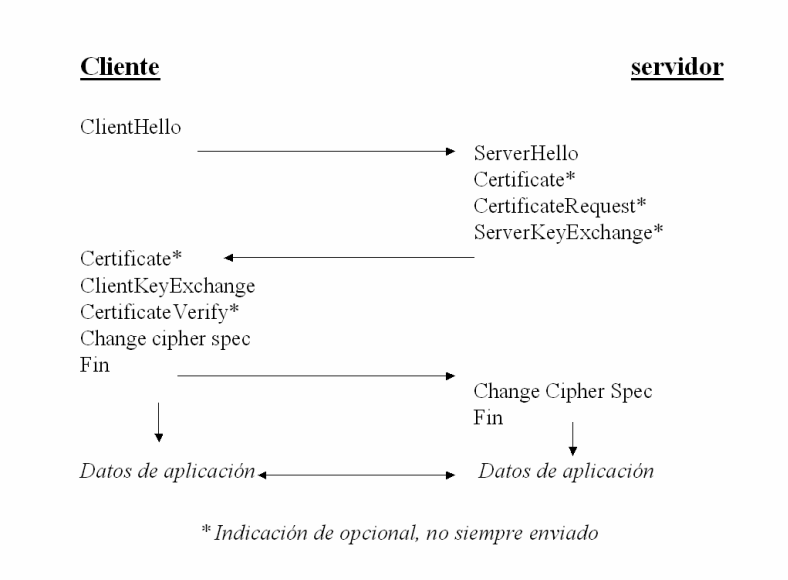


Figura 25. Figura de establecimiento de conexión con SSL utilizando HP y CSCP

El establecimiento de una sesión se realiza con *Handshake Protocol (HP) como podemos ver en la figura anterior y sigue los siguientes pasos:*

1. El **cliente inicia** la conexión y le **responde el servidor**
2. Se negocian los parámetros de conexión segura
3. El **servidor** *envía su certificado firmado* (opcional)
4. Certificado expedido por una CA, o validado *ad hoc*
5. Puede requerir un certificado del cliente
6. El **cliente** *envía una clave simétrica*
7. La genera y cifra (con la clave pública del servidor)
8. Se intercambian mensajes *Change Cipher Spec (CCSP)*
9. El cliente informa de los cifradores disponibles
10. El servidor elige el más fuerte común a ambos
11. Ya puede comenzar la comunicación cifrada

El protocolo AP *Alert Protocol es utilizado para el* intercambio de mensajes especiales, como Warning: para notificar avisos o Error (fatal) para indicar errores irrecuperables.

Una mejora del protocolo SSL es el Transport Layer Security (TLS), que es una generalización de SSL para ser un estándar no dependiente de un solo fabricante (Netscape). La v1.0 es el RFC-2246 del IETF (1.999), es una ligera modificación de SSL v3.0. Adicionalmente a *SSL*, tiene una extensibilidad del protocolo, ya que se pueden agregar nuevos protocolos por encima del *Record Protocol. Los protocolos que c*omprende son similares a los de SSL: Record Protocol (RP), Alert Protocol (AP), Handshake Protocol (HP), Change Cipher Spec (CCSP) y Application Data Protocol (ADP).

9.4 REDES Y SEGURIDAD

La seguridad es un concepto relativo, pues los mecanismos que minimizan la vulnerabilidad de un bien o recurso en un determinado caso, pueden ser insuficientes en otro caso. Esta suficiencia o insuficiencia vendrá determinada por la importancia de los bienes y recursos que posea, de forma que un ordenador que solo contenga contabilidad doméstica puede considerarse seguro sin la presencia de ningún mecanismo, mientras que un ordenador que contenga la contabilidad de una empresa debe poseer mecanismos para asegurar la imposibilidad de manipulación de la misma.

Las amenazas a la seguridad pueden clasificarse, atendiendo a la intencionalidad de las mismas en accidentales o intencionadas. Las amenazas accidentales son las que se producen sin necesidad de un intento premeditado, como por ejemplo una avería en el sistema. Las amenazas intencionadas pueden

variar desde el examen casual de la información de un ordenador hasta ataques sofisticados utilizando conocimientos especiales sobre el sistema.

Si en lugar de atender a la intencionalidad atendemos al daño ocasionado, las amenazas a la seguridad se clasifican en pasivas y activas. Las amenazas pasivas son las que no conllevan ninguna modificación en la información que posee el sistema y por tanto no se modifica ni su operación ni su estado. Las amenazas activas suponen una alteración del sistema y un cambio en su estado de operación. Aunque obviamente las amenazas activas son mucho más perjudiciales que las pasivas, estas deben ser tenidas en cuenta pues en muchos casos pueden convertirse en activas con la intervención de un agente distinto.

La seguridad adquiere cada vez más importancia a medida que la red informática se encuentra presente en más aspectos de la economía mundial, aspectos como el comercio electrónico, la transacción de información confidencial a través de la misma, etc.

Obviamente, la seguridad es un tema que debe inquietar a cualquier organización que hoy día decida conectar su red a otras sobre Internet. Basta echar un vistazo a las estadísticas para tomar conciencia del riesgo que se corre: el número de incidentes contra sistemas conectados casi se duplica cada año,según el Computer Emergency Response Team Coordination Center (CERT-CC). Y no debeextrañarnos, si tenemos en cuenta el vertiginoso crecimiento de Internet en los últimos años, queimplica, por una parte, nuevas redes susceptibles de ser atacadas, y por otra, nuevos atacantes en potencia.

Lo cierto es que tal y como están las cosas, atacar una red conectada a Internet que no haya sido protegida de un modo "especial" (es tan frecuente como erróneo creer que una filosofía de seguridad tradicional, basada en contraseñas y protección de ficheros, es suficiente para protegerse en Internet), es relativamente fácil si se sabe cómo, y mucho más aún si se utilizan sistemas operativos antiguos que no.han sido actualizados ni debidamente "parcheados". En la red es posible encontrar, sin mucho esfuerzo, listas de debilidades tanto de protocolos como de sistemas operativos, así como guías que señalan los pasos a seguir para explotar dichas debilidades. Incluso existen servidores de ftp anónimo con todo tipo de herramientas orientadas a tomar el control de cualquier máquina

Todas las líneas actuales de investigación en seguridad de redes comparten una idea: la concentración de la seguridad en un punto. Se obliga a que todo el tráfico entre la red que se pretende proteger y las redes externas pase por un mismo punto. Este punto se conoce con el nombre de cortafuego, y físicamente puede ser desde un simple host hasta un complejo conjunto de redes separadas por routers. El empleo de un cortafuego presenta enormes ventajas sobre los enfoques de seguridad en redes tradicionales (que requieren la seguridad individual de cada host conectado, y por tanto sólo pueden justificarse en entornos con un reducido número de máquinas), permitiendo concentrar todos los esfuerzos en el control de tráfico a su paso por el cortafuego.

Peligros y modos de ataque

El proceso de diseñar un sistema de seguridad podría decirse que es el encaminado a cerrar las posibles vías de ataque. Se hace imprescindible, por tanto, adquirir un profundo conocimiento acerca de las debilidades que los atacantes aprovechan, y del modo en que lo hacen. La variedad de ataques posibles contra un sistema es excesivamente amplia y variada a primera vista. Sin embargo, analizándolos con más detenimiento, observamos que la mayoría de ellos no aprovechan una única debilidad, sino una combinación de éstas, y que en el fondo, el tipo de debilidades es, afortunadamente, más reducido. Sin embargo, eso tampoco es tranquilizador si, como veremos, hay problemas de difícil solución.

Últimamente vemos aparecer en la red diversas taxonomías de vulnerabilidades y tipos de ataques. Pasemos a ver una lista con los tipos de ataques que actualmente se pueden realizar sobre Internet, explicando brevemente en qué consiste cada uno y qué debilidades aprovecha.

- *Sniffing* : este ataque consiste en escuchar los datos que atraviesan la red, sin interferir con la conexión a la que corresponden. Se utiliza principalmente para obtener contraseñas, y en algunos casos para obtener información confidencial. Para proteger los contraseñas contra el sniffing basta con emplear mecanismos de autenticación y encriptación.

- *Spoofing* **:** es el nombre que se le da a los intentos del atacante por ganar el acceso a un sistema haciéndose pasar por otro que dispone de los privilegios suficientes para realizar la conexión. El ataque que más se suele utilizar sobre conexiones TCP es el conocido como *adivinación del número de secuenci*a. Se basa en la idea de que si un atacante puede predecir el número inicial de secuencia de la conexión TCP generado por la máquina destino, entonces el atacante puede adoptar la identidad de máquina "confiada". . VER ANEXO 1.

- *Hijacking* **:** consiste en robar una conexión después de que el usuario ha superado con éxito el proceso de identificación ante el sistema. El ordenador desde el que se lanza el ataque ha de estar en alguna de las dos redes extremo de la conexión, o al menos en la ruta entre ambas. El único método seguro para protegerse contra este tipo de ataques es el uso de encriptación. . VER ANEXO 1.

- *Ingeniería social:* son ataques que aprovechan la buena voluntad de los usuarios de los sistemas atacados. Un ejemplo de ataque de este tipo es el siguiente: se envía un correo con el remite "root" a un usuario, en una gran red académica (donde frecuentemente los usuarios no conocen a los administradores), con el mensaje "por favor, cambie su contraseña a murcia1". El atacante entonces espera un poco, y entra con ese contraseña. A partir de ahí puede emplear otras técnicas de ataque (bugs del sistema para obtener un control total de la máquina, confianza transitiva para entrar en otras máquinas de la red, etc). Ante este tipo de ataques la mejor defensa es educar a los usuarios acerca de qué tareas no deben realizar jamás, y qué información no deben suministrar a nadie, salvo al administrador en persona.

- *Explotar bugs del software:* aprovechan errores del software. A la mayor parte del software se le ha añadido la seguridad demasiado tarde, cuando ya no era posible rediseñarlo todo. Además, muchos programas corren con demasiados privilegios, lo que les convierte en objetivo de los hackers, que únicamente han de hacerse con una copia del software a explotar y someterlo a una batería de pruebas para detectar alguna debilidad que puedan aprovechar. Entre los posibles ataques que se pueden efectuar está el desbordamiento de pila, consistente en introducir datos en la pila a través de funciones de entrada salida, de forma que permitan modificar las posiciones de retorno de las funciones y con ello ejecutar código que permite al atacante tomar control del sistema. Esta secuencia de ataques, se conoce como ***exploits***. La solución a este problema de desbordamiento se realiza a través de funciones de entrada salida limitada, Las funciones strcpy(), strcat(), gets(), son potencialmente vulnerables.

- *Confianza transitiva* **:** en sistemas Unix existen los conceptos de *confianza entre hosts y entre usuario*s. Se dice que un sistema es *confiado* para otro cuando desde el primero, cualquier usuario puede establecer un conexión al segundo sin necesidad de dar un contraseña. Se dice que un usuario sobre un sistema es *confiado* para otro sistema cuando ese usuario, desde el primer sistema, puede establecer un conexión al segundo sin necesidad de dar un contraseña. Así, cualquier atacante que tome el control de una máquina, probablemente podrá conectarse a otras gracias a la confianza entre hosts y/o entre usuarios

- *Ataques dirigidos por datos* **:** son ataques que tienen lugar en modo diferido, sin la participación activa por parte del atacante en el momento en el que se producen. El atacante se limita a hacer llegar a la víctima una serie de datos que al ser interpretados ejecutarán el ataque propiamente dicho.

- *Caballo de Troya* **:** un programa que se enmascara como algo que no es, normalmente con el propósito de conseguir acceso a una cuenta o ejecutar comandos con los privilegios de otro usuario.

- *Denegación de servicios* **:** estos ataques no buscan ninguna información contenida en las máquinas atacadas ni conseguir acceso a ellas. Únicamente van encaminados a impedir que sus usuarios legítimos puedan usarlas. El caso más típico es el *mail bombing:* envío de cantidades ingentes de correo a la máquina atacada hasta saturarla. Puesto que es casi imposible evitar todos los ataques de denegación de servicio, lo más importante es configurar los servicios para que si

uno de ellos es inundado, el resto permanezca funcionando mientras se encuentra y soluciona el problema. VER ANEXO 2.

- *Enrutamiento fuente:* los paquetes IP admiten opcionalmente el enrutamiento fuente, con el que la persona que inicia la conexión TCP puede especificar una ruta explícita hacia él. La máquina destino debe usar la inversa de esa ruta como ruta de retorno, tenga o no sentido, lo que significa que un atacante puede hacerse pasar por cualquier máquina en la que el destino confíe (obligando a que la ruta hacia la máquina real pase por la del atacante). Dado que el enrutamiento fuente es raramente usado, la forma más fácil de defenderse contra ésto es deshabilitarlo en el router.

- *Adivinación de contraseñas:* un elevado porcentaje de penetraciones en sistemas se deben al fallo del sistema de contraseñas. El fallo más común es la mala elección de contraseñas por parte de los usuarios. Este se suele llevar a cabo en dos formas básicas. La primera consiste en intentar entrar usando pares cuenta-contraseña conocidos o asumidos (muchos sistemas operativos disponen de cuentas administrativas con contraseñas por defecto, que pese a no ser comentadas en los manuales del sistema, son conocidas por los atacantes). El segundo modo en que los hackers obtienen los contraseñas es mediante el uso de crackers (programas que comparan un diccionario de términos contra ficheros de contraseñas robados). Para protegerse contra estos ataques es vital tanto la educación al usuario sobre cómo elegir su contraseña, como mantener asegurado el fichero de contraseñas, de modo que no pueda ser robado.

- *Icmp redirect y destination unreachable* **:** muchos mensajes ICMP recibidos en un host son específicos a una conexión particular o son disparados por un paquete enviado por ese host. La intención es limitar el alcance de los cambios dictados por ICMP. Desafortunadamente las viejas implementaciones de ICMP no usan esta información extra, y cuando llega uno de esos mensajes, todas las conexiones entre el par de hosts que intervienen en la conexión que propició el mensaje se ven afectadas. Además, con la opción redirect, alguien puede alterar la ruta a un destino para que las conexiones en las que esté interesado pasen por su máquina, de forma que pueda intervenirlas. Los mensajes redirect deben obedecerlos sólo los hosts, no los routers, y sólo cuando estos provengan de un router de una red directamente conectada.

Elementos de seguridad

Una vez conocidos los peligros a los que nos enfrentamos, necesitamos medios para pretegernos contra ellos. En principio, limitando el tráfico entre nuestra red y las externas, a aquel que se considere seguro, o al menos que esté justificado, limitaremos el número de ataques posibles. A continuación veremos que el *filtro de paquetes* y los *servidores proxy* nos permiten ésto. Además, si decidimos permitir que se pueda acceder a nuestras máquinas desde el exterior, habremos de asegurarnos de que los intentos de conexión provienen de quienes dicen provenir. Para ello no podemos fiarnos de los contraseñas convencionales, puesto que un ataque por sniffing daría el contraseña al atacante. Veremos a continuación métodos de *autenticación* que nos solucionan este problema. Por último, si creemos que nuestra red puede ser objeto de un ataque *hijackin*g, necesitamos alguna técnica para impedirlos. En este caso necesitaremos *encriptar* la conexión.

Los métodos a aplicar en una red para ofrecer barreras de seguridad son:

1. Métodos criptográficos: redes privadas virtuales, IPSec, SSH (Secure Shell)
2. Seguridad perimetral: cortafuegos, NAT (Network Address Traslation) e IDS (Intrusión Detection System)
3. Seguridad en el sistema centralizado (envolventos y/o proxies)

Descritos los métodos, los elementos utilizados por dichos métodos son:

Criptografía*:* mediante el uso de la criptografía se intenta proteger la información a base de codificarla de una forma desconocida a los elementos que no forman parte de la comunicación: *algoritmos de clave privada o simétricos* (DES, TDES, IDEA, RC4 y Skipjack), *algoritmos de clave pública o asimétricos* (RSA). Las aplicaciones básicas de los algoritmos criptográficos son: el *cifrado* es la encriptación de un

mensaje con una clave; la *firma digital* para protegerlos contra la falsificación, permitiendo al receptor probar la fuente y la integridad de los mismos, una *función hash segura*.

***Autenticación**:* la autenticación es el proceso seguido por una entidad para probar su identidad ante otra. Distinguimos dos tipos de autenticación: la de un usuario a una máquina durante la secuencia de login inicial, y la de máquina a máquina durante una operación. Los contraseñas tradicionales son demasiado débiles para usarlos sobre una red, y por tanto se usan *contraseñas no reusables.* Estos cambian cada vez que se usan, y por tanto no son sensibles al sniffing. El método de autenticación por dirección IP del host (o bien su nombre DNS) es susceptible de ser atacado mediante spoofing con relativa facilidad, y por tanto se usan técnicas de criptografía, contando con un Centro de Distribución de Claves (KDC) para la distribución y verificación de las mismas. El KDC más conocido es *Kerbero*s.

***Filtro de paquetes**:* los routers permiten realizar un filtrado de paquetes en base a la información contenida en sus cabeceras. Básicamente, la información que se suele examinar es: la dirección IP origen, la dirección IP destino, el tipo de protocolo (TCP, UDP o ICMP), el campo de opciones IP, el puerto origen TCP o UDP, el puerto destino TCP o UDP, el campo de banderas TCP y el tipo de mensaje ICMP. Además de la información contenida en el paquete, se puede tener en cuenta la interfaz de red por la que llega el paquete. El hecho de que los servidores de servicios Internet residan en ciertos números de puertos concretos, permite al router bloquear o permitir la conexión a esos servicios simplemente especificando el número de puerto apropiado en el conjunto de reglas especificado para el filtro de paquetes. El filtro de paquetes es transparente a los usuarios, es decir, no requiere conocimientos ni cooperación por su parte. Sin embargo, las reglas de filtro son difíciles de definir, y una vez definidas, duras de testear. Una relación de puertos y su asignación lo podemos ver en el ANEXO 3.

***Servidores proxy o pasarelas**:* son aplicaciones que nos permiten redirigir el tráfico del nivel de aplicación a través de un cortafuego en el acceso a una red (ejemplo con Socks, Sock-et-s) y/o puentear con otra aplicación, dentro de un sistema centralizado. En este último caso también se llama envolvente.. Al cliente le presentan la ilusión de que está tratando directamente con el servidor real. El servidor real cree que está tratando directamente con un usuario en el host donde está corriendo el proxy. Este sistema no siempre es transparente al usuario, puesto que algunos proxies requieren software cliente especial, o bien el software estándar utilizándolo con procedimientos especiales. Los servicios proxy sólo son efectivos usados en conjunción con un mecanismo que restrinja las comunicaciones directas entre los hosts internos y externos (bien con un dual-homed host, bien con filtro de paquetes).

Estos tres últimos medios son analizados o estudiados desde el punto de vista de pasarelas, como una clasificación más general. Cualquier dispositivo que permite la intercomunicación a niveles superiores al de red se denomina genéricamente pasarela (gateway en inglés). Así nos referimos a pasarelas del nivel de transporte o del de aplicación. Una pasarela del nivel de aplicación es un host que implementa dos (o mas) pilas completas de protocolos y que puede 'trasvasar' datos de una aplicación a su equivalente en la otra pila de protocolos, realizando las conversiones adecuadas. Un ejemplo de esto es el intercambio de mensajes de correo electrónico, por ejemplo entre SMTP (protocolo de correo en Internet) y X.400 (protocolo de correo OSI). Otro ejemplo es el servicio que permite enviar desde un formulario Web un mensaje corto a un teléfono GSM.

Los problemas que se plantean en la interconexión de redes diferentes a nivel de pasarelas de aplicación son en cierto modo parecidos a los que se dan a niveles inferiores, como ocurre al interconectar redes Ethernet y Token Ring. Por ejemplo, si se dispone una pasarela de correo electrónico entre dos redes diferentes, una de las cuales implementa acuse de recibo y la otra no, se plantea el problema de qué hacer en la pasarela cuando un mensaje es enviado de la primera a la segunda; a lo sumo se podría acusar recibo cuando el mensaje se recibe en la pasarela, pero eso no significa que el mensaje haya llegado a su destino; la otra alternativa sería simplemente no enviar ningún acuse de recibo en este caso.

Además de permitir la interconexión de protocolos distintos las pasarelas de aplicación también se emplean para conectar redes con el mismo protocolo o puenteando a servidores controladn su acceso. Existen diversas razones por las que esto puede ser conveniente, como por ejemplo las siguientes:

- **Seguridad (cortafuegos):** si se quiere tener un control riguroso del tráfico entre una determinada red y el exterior se dispone un cortafuegos que aísle dicha red; a menudo los cortafuegos (o cortafuegos) actúan como pasarelas a nivel de transporte o de aplicación.

- **Eficiencia (servidores proxy/cache):** los servidores cache permiten capturar una copia de la información que un usuario se trae del exterior para disponer de ella a<nte la eventualidad de que más tarde otro usuario requiera la misma información. Esto mejora el tiempo de respuesta al usuario ya que la información solicitada se le sirve localmente y reduce el nivel de ocupación de la línea WAN o la conexión a Internet, normalmente de elevado costo. Para realizar su función los servidores caché actúan como pasarelas del nivel de aplicación, normalmente para el protocolo http.

- **NAT (traducción de direcciones):** Como veremos más adelante existen situaciones en las que es necesario manejar un direccionamiento diferente en la red interna y la externa; en estos casos es preciso utilizar un dispositivo NAT (Network Address Translation) que nos permita el intercambio de paquetes entre ambas redes. Algunas aplicaciones requieren una complejidad mayor en el proceso de traducción de direcciones hasta el punto de que esto solo es posible mediante un programa que interprete y modifique la información contenida en el paquete a nivel de aplicación.

- **Envolventes o proxies de aplicación**: son programas sencillos, son pasarelas a nivel de aplicación, que gestionan todo el control de acceso a los servidores dentro de un sistema centralizado. Ej *tcpwrappers*.

Seguridad en red basada en criptografía

Túneles

En ocasiones se quiere intercambiar paquetes entre dos redes que utilizan el mismo protocolo, pero que están unidas por una red que utiliza un protocolo diferente. Por ejemplo supongamos que un banco dispone de una red de área extensa con protocolo SNA que une todas sus oficinas, y dos de éstas disponen además de redes locales TCP/IP. Si se desea que estas dos oficinas intercambien tráfico TCP/IP sin establecer para ello una nueva línea ni instalar routers multiprotocolo en todo el trayecto se puede establecer un túnel entre ambas oficinas. Los dos nodos SNA ubicados en los extremos del túnel serán los encargados de añadir a los paquetes IP la cabecera SNA adecuada para que lleguen al otro extremo. Los paquetes IP viajarán 'encapsulados' a través del túnel en paquetes SNA, de forma que los paquetes IP no sean vistos por la red SNA. Los conceptos de túnel y de encapsulado de paquetes van siempre asociados entre sí.

En Internet hay situaciones en las que los túneles se utilizan de forma habitual; por ejemplo la interconexión de routers multicast a través de routers unicast para constituir la red MBone, o la interconexión de 'islas' IPv6 para constituir el 6bone. En estos casos los túneles se utilizan para enviar paquetes del mismo protocolo (IP) pero de distinto tipo (multicast en unicast) o versión (IPv6 en IPv4). En algunos casos resulta útil encapsular incluso un paquete en otro del mismo tipo y versión; por ejemplo encapsulando un paquete IPv4 en otro podemos forzar la ruta que ha de seguir, cumpliendo así una función similar a la opción 'loose source routing' de la cabecera IP, pero sin las limitaciones que tiene ésta en el número de direcciones. Otro uso del encapsulado podría ser para superar el valor máximo del campo TTL; al utilizar encapsulado la cabecera interior empezaría a descontar su TTL sólo a partir del punto de salida del túnel.

Uno de los usos más habituales de los túneles actualmente es para la creación de redes privadas virtuales o VPNs, como veremos a continuación.

Los túneles no son una solución deseable en sí mismos, ya que a lo largo del túnel los paquetes han de llevar dos cabeceras, lo cual supone un mayor overhead; además los extremos del túnel se convierten en puntos simples de fallo y potenciales cuellos de botella en el rendimiento de la red.

Redes Privadas Virtuales (VPNs)

Entendemos por red privada virtual o VPN (Virtual Private Network) la interconexión de un conjunto de ordenadores haciendo uso de una infraestructura pública, normalmente compartida, para simular una infraestructura dedicada o privada. Entre las características propias de una red privada virtual se encuentra la posibilidad de utilizar direccionamiento no integrado en la red del proveedor del servicio.

Existen muchas maneras de crear y utilizar VPNs, tanto en IP como en otros protocolos. Aquí solo comentaremos algunas de sus posibilidades más habituales para que sirvan como ejemplo de su aplicación.

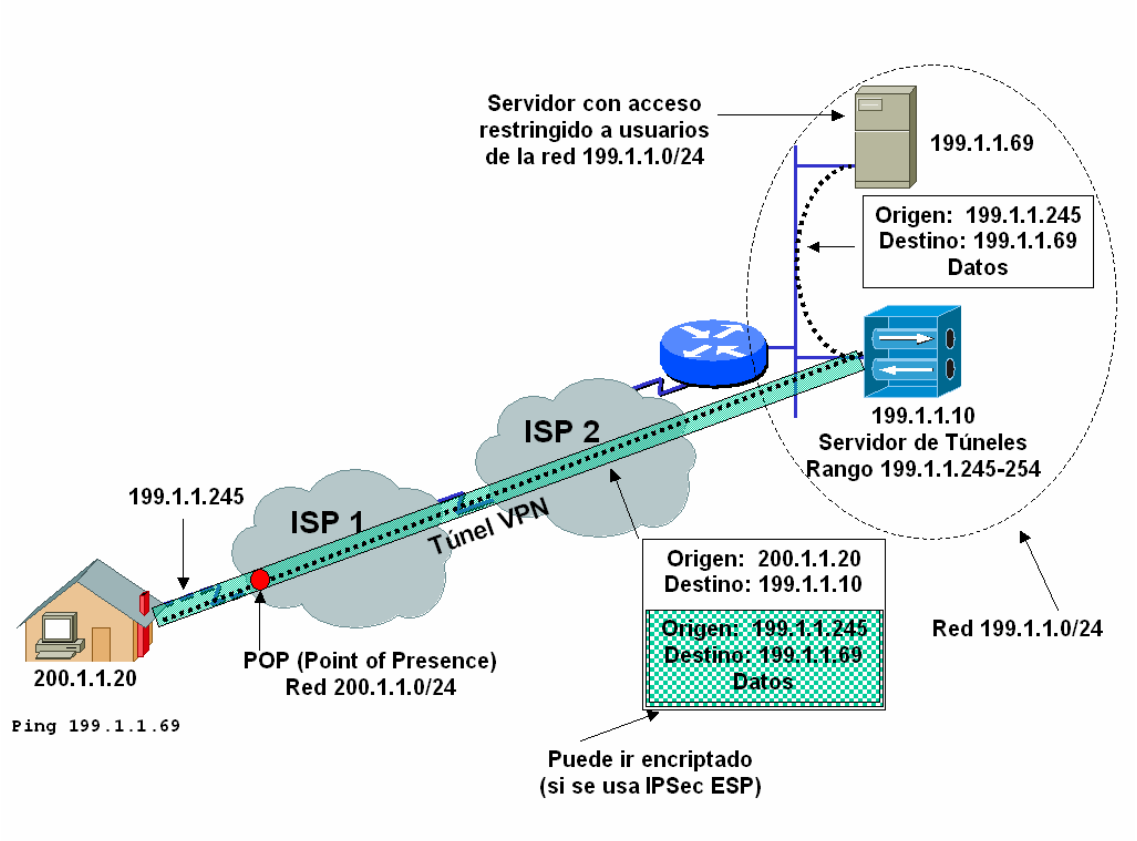


Figura 26: ejemplo de acceso a la Intranet 199.1.1.0/24 a través de VPN

Supongamos que un viajante quiere conectarse remotamente a la red de su empresa para consultar una base de datos y para ello emplea los servicios de un proveedor comercial cualquiera, ya que esto le permite conectarse a su red pagando tarifa metropolitana. Supongamos también que por razones de seguridad o de licencias de software la empresa se ha visto obligada a limitar el acceso a dicha base de datos únicamente a los ordenadores de la red de la empresa, es decir aquellos que tienen direcciones IP de su red (supongamos que se trata de la red 199.1.1.0/24). Como nuestro viajante al conectarse recibe una dirección IP de la red de su proveedor (supongamos que se trata de la red 200.1.1.0/24) no le es posible acceder a dichos servicios.

Este problema puede resolverse creando un túnel VPN (Virtual Private Network) entre el ordenador del viajante y un ordenador de la red de su empresa. El túnel le va a permitir a nuestro usuario remoto acceder a la red de la empresa como si estuviera conectado en la red local (aunque con una velocidad que dependerá obviamente de la conexión física que utilice). El túnel se creará entre su ordenador y un equipo que denominaremos ‘servidor de túneles’ ubicado en la red local de la empresa, el cual se encargará de encapuslar y desencapsular los paquetes de este usuario. Veamos en detalle como funcionaría el túnel VPN en este caso concreto.

En primer lugar, la empresa deberá tener el servidor de túneles conectado a su red local; dicho servidor puede ser un equipo específicamente diseñado para este fin, un router o un servidor de propósito general Linux o Windows 2000 Server por ejemplo (el software necesario para actuar como servidor de túneles VPN está incluido en Windows 2000 server). Supongamos que el servidor de túneles está conectado a la red local mediante una interfaz Fast Ethernet y que tiene la dirección IP 199.1.1.10. Además de disponer de una dirección propia el servidor de túneles debe tener asignado un rango de direcciones que utilizará para establecer los túneles. Supongamos que no esperamos más de 10 usuarios simultáneos de este servicio, por lo que reservamos para este fin las últimas 10 direcciones útiles de la red de la empresa, es decir el rango que va de la 199.1.1.245 a la 199.1.1.254 ambas inclusive.

Supongamos ahora que nuestro viajante se conecta por red telefónica a su proveedor habitual y que mediante la asignación dinámica de direcciones de PPP recibe de éste la dirección 200.1.1.20. A continuación se conecta con el servidor de túneles y, previa autenticación mediante usuario/contraseña, el servidor crea un túnel para él y le asigna la dirección 199.1.1.245. A partir de ese momento los datagramas que envíe el viajante llevarán dos cabeceras, una exterior y otra interior. Si el viajante hace desde su PC un ping a una dirección cualquiera de la empresa (por ejemplo la 199.1.1.69) enviará un datagrama cuya cabecera exterior llevará como dirección de origen 200.1.1.20 y de destino 199.1.1.10 (el servidor de túneles) y en la cabecera interior llevará como dirección de origen 199.1.1.245 y como destino 199.1.1.69; este datagrama hará su viaje en dos etapas; en la primera llegará al servidor de túneles (199.1.1.10), que lo despojará de su cabecera exterior y procesará a continuación la cabecera interior, enviando el datagrama al destino deseado en la red de la empresa. El datagrama de respuesta al ping seguirá un proceso simétrico inverso, es decir hará la primera parte de su viaje con una sola cabecera que contendrá 199.1.1.69 como dirección de origen y 199.1.1.245 como destino; ese datagrama será entregado al servidor de túneles, que se encargará entonces de incorporarle la cabecera exterior con dirección de origen 199.1.1.10 y de destino 200.1.1.20; cuando llegue al ordenador del viajante el software cliente VPN le despojará de la cabecera exterior y lo procesará como un datagrama dirigido a él proveniente de 199.1.1.69. Así pues el uso del túnel VPN permite a nuestro viajante acceder a la red de su empresa como si estuviera conectado en la red local detrás del servidor de túneles. El software necesario para crear túneles VPN como el descrito en este ejemplo está integrado en sistemas tales como Windows 98 o Windows 2000 profesional.

Obsérvese que mientras está operativo el túnel todo el tráfico del usuario remoto con cualquier destino de Internet (pertenezca o no a la empresa) se realiza a través del servidor de túneles. Esto significa que el usuario sufrirá las limitaciones asociadas a la capacidad del servidor de túneles y, si accede a destinos ubicados fuera de la red local de su empresa sufrirá también las limitaciones que le imponga la conexión a Internet que tenga su empresa; por tanto lo lógico sería utilizar la conexión a través del túnel únicamente cuando se requiera acceder a servicios de acceso restringido dentro de la red de la empresa.

Además de conectar un ordenador aislado como en el ejemplo anterior es posible establecer un túnel VPN entre routers. Esto nos permite conectar toda una red local a través de un solo equipo, y además no requiere soporte de túneles en el host final. En este caso los routers serán los encargados de realizar la labor de encapsulado/desencapsulado de datagramas. Las VPNs permiten conectar por ejemplo una oficina remota a una oficina principal utilizando Internet, con lo que los costes se pueden reducir considerablemente, especialmente si se trata de conexiones de larga distancia. Además, la reciente aparición de conexiones a Internet de alta velocidad y bajo costo (fundamentalmente ADSL y cable módems) hace especialmente atractivo el uso de túneles VPN sobre este tipo de servicios para constituir enlaces internos en una red sin incurrir en los elevados costos de constituir una infraestructura dedicada mediante enlaces punto a punto o circuitos frame relay o ATM.

IPSEC

Con bastante frecuencia cuando se establecen túneles VPN como hemos visto en los ejemplos anteriores, además de la integración de las direcciones de red se plantea un requerimiento desde el punto de vista de la seguridad, dado que los datagramas viajan por una infraestructura pública en la que la información puede ser capturada y/o modificada por usuarios externos. Por este motivo la constitución de túneles VPN viene acompañada a menudo de un requerimiento de seguridad, constituyendo lo que podríamos denominar redes privadas virtuales seguras.

El problema de una comunicación segura a través de una red se resuelve normalmente a nivel de enlace, a nivel de red o a nivel de aplicación. Cada una de estas tres alternativas tiene sus ventajas e inconvenientes:

- **Nivel de enlace:** La seguridad a nivel de enlace se implementa en los dispositivos que se conectan al medio de transmisión, por ejemplo dos routers que se comunican mediante una línea punto a punto. En este caso los mecanismos de seguridad se pueden aplicar de forma transparente al protocolo utilizado a nivel de red. Sin embargo en una red grande requiere encriptar y desencriptar la información en cada salto que da el paquete; aun en el caso de utilizar dispositivos que realicen esta tarea por hardware el retardo que esto puede introducir cuando el número de saltos es elevado puede hacer inviable el uso de aplicaciones en tiempo real que requieren cierto nivel de Calidad de Servicio. Además implementar seguridad a nivel de enlace requiere controlar la infraestructura de la red, cosa que no es factible cuando se utilizan los servicios de un operador o un ISP. Un ejemplo de seguridad a nivel de enlace se da en las redes de televisión por cable (CATV) en que la información viaja encriptada (DES) entre el Cable Módem y el CMTS (Cable Modem Termination System); al ser las redes CATV un medio broadcast es necesario el uso de encriptación para evitar la captación de información por parte de usuarios que no son los destinatarios de la misma.

- **Nivel de red:** Esta es la aproximación adoptada por los estándares IPSec. En este caso la seguridad se limita al protocolo IP y otros protocolos sólo podrán aprovecharla si se encapsulan previamente en paquetes IP. La seguridad a nivel de red puede aplicarla el usuario de forma transparente al proveedor del servicio y encaja de forma muy adecuada con el concepto de VPNs pudiendo crear lo que denominamos redes privadas virtuales seguras.

- **Nivel de aplicación:** esta es la aproximación adoptada por ejemplo en correo electrónico por PEM (Privacy Enhanced Mail) o PGP (Pretty Good Privacy), en HTTP por Secure HTTP o SSL (Secure Sockets Layer), o por SNMP versión 3. El principal inconveniente de abordar la seguridad a nivel de aplicación estriba precisamente en la necesidad de incorporar funcionalidades similares en cada uno de los protocolos del nivel de aplicación que deban utilizarlas, replicando así gran cantidad de tareas en diferentes partes de código. La ventaja es que la seguridad se puede desarrollar de forma selectiva, aplicándola únicamente en el intercambio de información confidencial o importante. Además, al aplicarse los mecanismos de encriptado o validación de la información en el nivel más alto posible el nivel de seguridad obtenido es máximo ya que se reduce el riesgo de que la información pueda ser interceptada o modificada por otros usuario o procesos distintos del destinatario de ésta.

Denominamos IPSec al conjunto de estándares que trata todo lo relacionado con el intercambio seguro de información a través de Internet. En realidad IPSec es una arquitectura (descrita en el RFC 2401, de 66 páginas), y un conjunto de protocolos y mecanismos de autenticación y encriptado.

Las principales funcionalidades que incorpora IPSec son las siguientes:

- AH (Autentication Header): en este caso no se pretende garantizar la confidencialidad de la información sino únicamente su veracidad. La cabecera de autenticación asegura al receptor que el datagrama no ha sido alterado durante su viaje por la red, ni en su contenido ni en su cabecera (salvo por la modificación del campo TTL y por consiguiente del checksum, que se han de realizar en cada salto); por consiguiente además de garantizarse el contenido se garantiza la autenticidad del remitente. AH se describe en el RFC 2402, de 22 páginas.

- ESP (Encapsulating Security Payload): garantiza la confidencialidad de la información. La parte de datos del datagrama (incluida la cabecera de transporte) viaja encriptada de forma que sólo el destinatatrio pueda descifrar su contenido. Opcionalmente ESP puede incluir la funcionalidad de AH, es decir garantizar que el contenido no ha podido ser alterado por terceros. ESP se describe n el RFC 2406, de 22 páginas.

- ISAKMP (Internet Security Association and Key Management Protocol): consiste en una serie de mecanismos seguros para realizar, de forma manual o automática, el intercambio de claves necesarias para las labores de encriptado y autenticación realizadas por AH y ESP. ISAKMP se describe en el RFC 2408, de 86 páginas.

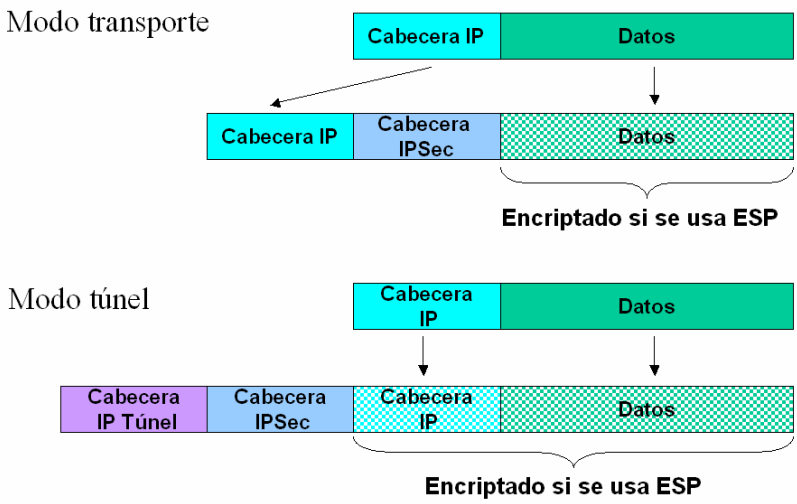


Figura 27: modos de funcionamiento de IPSec: modo transporte y túnel

Podemos distinguir dos modos de funcionamiento de IPSec:

a) Modo transporte: la encriptación se realiza extremo a extremo, es decir del host de origen al host de destino. Para extender en una empresa el uso de IPSec en modo transporte es necesario que todos los hosts tengan una implementación de IPSec.

b) Modo túnel: el encriptado se efectúa únicamente entre los routers de acceso a los hosts implicados. En este caso la información viaja no encriptada en la parte de la red local. El funcionamiento de IPSec en modo túnel permite integrar de forma elegante IPSec en una VPN, ya que el mismo dispositivo que realiza el túnel VPN se encarga de realizar las labores correspondientes al túnel IPSec.

Evidentemente el modo transporte es más fiable puesto que ofrece comunicación segura host a host. Sin embargo el modo túnel tiene la ventaja de que permite incorporar la seguridad sin necesidad de incorporar IPSec en los hosts; aunque la seguridad que se obtiene en este caso no es tan alta la sencillez de implantación es mucho mayor y se consiguen la mayor parte de los beneficios del modo transporte, ya que se protege la parte más expuesta del trayecto, que corresponde precisamente a la infraestructura pública o del operador. En función de las circunstancias que rodeen cada caso se deberá optar por una u otra, pudiendo haber incluso situaciones híbridas en las que en una misma empresa determinado tipo de información baste protegerla con el modo túnel mientras que para algún host concreto, que maneje información de mayor importancia, se deba utilizar el modo transporte.

Aunque en IPSec se prevé la posibilidad de utilizar una amplia diversidad de algoritmos de autenticación y encriptado el único exigido para todas las implementaciones es el DES (Data Encription Standard) que utiliza claves de 56 bits. Desde hace unos años se sabe que DES es relativamente poco seguro, ya que el código puede ser descifrado utilizando fuerza bruta en un tiempo no demasiado grande con un ordenador potente actual, por lo que también se suele utilizar bastante Triple DES, que es mucho más seguro aunque también algo más costoso de calcular. En un futuro próximo se prevé utilizar el algoritmo AES (Advanced Encryption Standard) del cual aún no existen implementaciones comerciales.

Uno de los problemas que plantea la encriptación es el consumo intensivo de CPU. Esto suele ser un problema especialmente en los routers y servidores de túneles que atienden túneles IPSec con la función ESP activada (la función AH no requiere encriptación). En estos casos se pueden concentrar cientos de usuarios y flujos de varios Megabits por segundo en un mismo equipo, que ha de realizar las labores de

encriptado/desencriptado para todos ellos, pudiendo verse limitado el rendimiento de las comunicaciones por la capacidad de la CPU del equipo utilizado. Para evitar este problema muchos servidores de túneles y routers permiten incorporar módulos que realizan los algoritmos de encriptación por hardware para hacerlos con mucha mayor rapidez.

Conexión segura, SSH: Secure Shell

Otra forma de proteger la información transmitida por la red desde la capa de aplicación a través de túneles extremo a extremo a nivel de aplicación, son las aplicaciones SSH, Secure Shell (SSH), por Tatu Ylonen (1.995). La interfaz de usuario es una línea de comandos segura de la capa de aplicación, inicialmente pensado para evitar el paso de contraseñas en las conexiones de telnet. Se considera el sustituto de los protocolos "r" (rsh, rcp, rlogin, ...) SSH es una aplicación especificada en drafts del IETF e implementa la parte de control a través del puerto 22.

Las funciones de SSH son, autenticación de equipos y usuarios (utilizando diferentes métodos de autenticación: RSA, Kerberos, ...), envío de contraseñas cifrados, permite generar canales X11 seguros para exportar pantalla por el uso de conexión cifrada, permite incorporar servidores externos de autenticación como Radius y Tacacs+, realiza el intercambio de claves públicas RSA, además que puede utilizar servidores de gestión de claves tanto con enfoque distribuido (claves en los equipos) como con enfoque centralizado (claves en una CA): certificados X.509

Actualmente, a modo de ejemplo SSH es utilizado como protocolo de conexión desde el exterior del Dpto de Informática de la Universitat de Valencia. Se puede consultar más información y descargar aplicaciones SSH en http://informatica.uv.es/~carlos/adm/comun/conexion_remota.htmlSSH comprende un conjunto de protocolos diferentes:

Transport Layer Protocol (SSH-TRANS) que implementa la parte autenticación del servidor y cliente, confidencialidad, integridad, compresión. Este protocolo funciona de la siguiente manera:

1. El cliente inicia una conexión contra el servidor y se verifica la versión del protocolo SSH (1.0 ó 2.0, incompatibles)
2. Negociación de los servicios requeridos, utilizando Diffie-Hellman para intercambio de la clave aleatoria. Si no se pueden soportar todos los servicios, se aborta
3. Intercambio de las claves de sesión que permiten cifrar y autentificar la sesión. Este intercambio de claves pueden ser renegociadas posteriormente
4. Finalmente intercambio de datos cifrados utilizando algoritmo simétrico negociado para cifrado, algoritmo hash negociado para la integridad y algoritmo de compresión negociado.

User Authentication Protocol (SSH-USERAUTH) es un protocolo por encima de SSH-TRANS y funciona de la siguiente manera

1. El servidor informa de los tipos de autenticación y crea una lista de métodos propuestos
2. El cliente escoge el método de autenticación en cualquier orden
3. El servidor acepta el método de autenticación y tiene la libertad de no aceptarlo

Los métodos de autenticación utilizados normalmente son contraseña tradicional, clave pública con RSA, y confianza basada en equipos.

Connection Protocol (SSH-CONN) es un protocolo por encima de SSH-TRANS y se encarga de establecer canales (multiplexados en una sesión) y a cada canal se les asigna un identificador único. En estos canales se aplica control de flujo por ventana y funciona de la siguiente manera

1. El iniciador abre el canal y especifica los parámetros de conexión (id, ventana, etc)
2. Intercambian datos en el canal
3. Un participante cierra el canal

Los tipos de canales son interactive Session como por ejemplo en consola remota o ejecución remota, para exportar consolas con X11 Redirect de forma que se redirigen todas las conexiones X11 por el canal y el canal tipo TCP/IP Redirect, que permite capturar un par (puerto TCP, IP) y se envía en modo túnel.

Seguridad en red perimetral

Cortafuegos (firewalls)

Especialmente con el crecimiento comercial de la Internet muchas empresas se enfrentan ante la necesidad de conectar sus redes al exterior; sin embargo esto plantea varios problemas de seguridad por diversos motivos, por ejemplo:

- Los ordenadores de la red local contienen información de carácter confidencial cuya salvaguardia resulta vital para la empresa.
- Del exterior pueden llegar virus u otro tipo de programas que perjudicarían seriamente los ordenadores de la empresa.
- Los empleados pueden utilizar la conexión a Internet para salir a lugares no autorizados (por ejemplo servidores de información no relacionados con su actividad en la empresa).

Para resolver los problemas de acceso seguro hacia/desde el exterior se ha creado el concepto de *cortafuego* o *firewall*, que consiste en un dispositivo formado por uno o varios equipos que se sitúan entre la red de la empresa y la red exterior (normalmente la Internet); el cortafuego analiza todos los paquetes que transitan entre ambas redes y filtra los que no deben ser reenviados, de acuerdo con un criterio establecido de antemano.

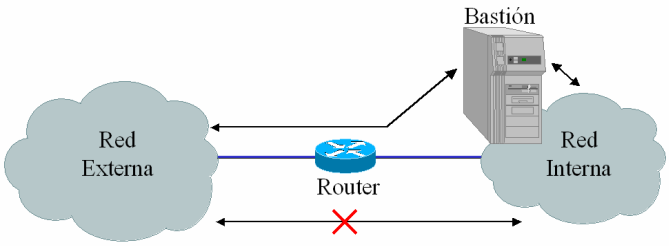


Figura 28: cortafuegos basado en router

En su versión más simple el cortafuego consiste únicamente en un router en el que se han configurado diversos filtros, por ejemplo impidiendo o limitando el acceso a determinadas direcciones de red, o el tráfico de ciertas aplicaciones o una combinación de ambos criterios, como vemos en la figura anterior. Dado que los usuarios siguen teniendo conexión directa a nivel de red con el exterior esta solución no es muy fiable; además las posibilidades de definir filtros en los routers son limitadas y el rendimiento se resiente de forma considerable si al router se le carga con una tarea de filtrado compleja.

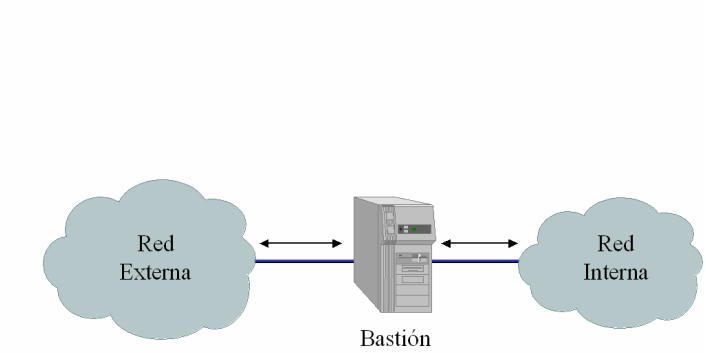


Figura 29: cortafuegos basadon en host bastión (*dual homed gateway*)

El siguiente nivel de cortafuego está formado por un host (*dual homed gateway*) que conecta por una parte a la Internet y por otra a la red corporativa, actuando él como router, como vemos en la figura anterior. El host implementa un servidor Web proxy que actúa como pasarela de aplicación para los servicios que se quieren permitir, limitado por las restricciones, filtros o reglas que se han especificado. El ordenador que actúa como barrera entre la red interna y la Internet se denomina host bastión ('bastion host') en terminología de cortafuegos. Esta solución ofrece una seguridad mayor que la anterior ya que el servidor proxy , al ser un host, puede procesar filtros mas complejos que un router; pero si un usuario malintencionado (hacker o cracker) consiguiera instalar un programa 'espía' (sniffer) en el host bastión podría capturar tráfico de la red interna de la empresa, ya que está directamente conectado a la LAN.

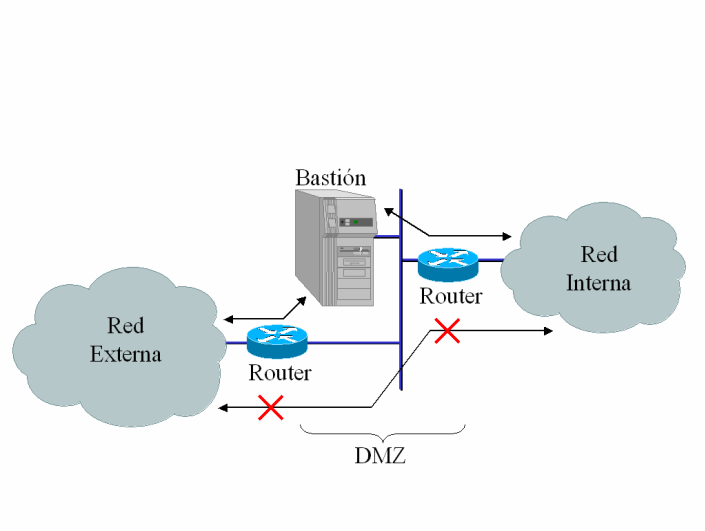


Figura 30: cortafuegos con zona desmiliatarizada, subred apantallada (*screened subnet*)

En un nivel mayor de seguridad el cortafuego se implementa mediante un host y dos routers, conectados entre sí por una pequeña red local; uno de los routers conecta a su vez con la red local de la empresa y el otro con la Internet; el host implementa una pasarela multiaplicación (servidor proxy) como antes, pero al no estar el directamente conectado a la red local de la empresa incluso en el caso de que un hacker consiguiera instalar en él un sniffer no podría capturar tráfico confidencial, pues solo podrá ver los paquetes dirigidos a él. Esta configuración se llama subred apantallada o *screened subnet*. En este modelo la red que une los routers con el host se denomina *zona desmilitarizada* o zona DMZ (Demilitarized Zone,

algo así como una zona neutra de seguridad). En ocasiones se utilizan otras variantes de arquitectura de cortafuego aún mas complejas.

En seguridad de redes, además de una buena arquitectura de cortafuego es importante comprobar que no hay 'agujeros' que permitan el acceso no controlado por otras vías. Por ejemplo, en una empresa con una red altamente protegida podría una oficina regional haber establecido una conexión local con otra institución sin conocimiento de los responsables de seguridad, o disponer de un sistema no seguro de autenticación de usuarios en el acceso por red conmutada para teletrabajo. En suma, que además de un buen cortafuego hay que tener una política de seguridad rigurosa si se quiere que las medidas sean efectivas, de lo contrario habremos puesto una puerta blindada, pero nos pueden entrar ladrones en casa por el 'trastero' con gran facilidad.

Traducción de direcciones (NAT)

Hace unos años cuando una organización deseaba conectar su red de forma permanente a la Internet lo normal era que solicitara al NIC una red adecuada a sus necesidades (clase B o C normalmente) y asignara números IP de dicha red a todas sus máquinas. De esta forma todos los ordenadores tenían acceso directo a Internet con todas las funcionalidades. Pero debido al crecimiento exponencial de Internet cada vez resulta más difícil obtener números IP del NIC correspondiente; por otro lado, en muchas ocasiones no se necesita, o incluso no se desea, disponer de un acceso directo a Internet con completa funcionalidad, por razones de seguridad fundamentalmente. Estos dos motivos, la seguridad y la dificultad para conseguir direcciones públicas, han impulsado a las organizaciones a hacer un mayor uso de las redes privadas según los rangos especificados en el RFC 1918 (10.0.0.0, 172.16.0.0 a 172.31.0.0, y 192.168.0.0 a 192.168.255.0). Estas redes privadas no pueden en principio intercambiar datagramas directamente con el exterior, por lo que han de utilizar un equipo intermedio que se ocupe de realizar la traducción de direcciones o NAT (Network Address Translation).

El NAT puede realizarse en un host (por ejemplo en Linux la función NAT se denomina ‘IP Masquerade’) aunque también se implementa en muchos routers. Dado que generalmente la función de NAT se realiza en la frontera entre una red local y el exterior la opción del router es bastante popular.

Normalmente NAT solo traduce paquetes IP que en el campo protocolo tienen el valor TCP, UDP o ICMP. El resto de protocolos (fundamentalmente protocolos de routing) no se traducen pues no tendría sentido intercambiar información de routing a través de un NAT.

Los tipos de NAT, inicialmente los NAT solo permitían iniciar conexiones desde ‘dentro’, es decir desde la red privada hacia el exterior. Este modo de funcionamiento, que se consideraba una medida de seguridad, se denomina **NAT tradicional**. Mas recientemente se ha extendido el uso de NATs que también permiten el inicio de la sesión desde fuera, por lo que a este tipo de NAT se le denomina **NAT bidireccional**.

La traducción se puede hacer de dos maneras: modificando únicamente la dirección IP, a lo que denominamos **NAT básico**, o cambiando también el número de puerto TCP o UDP, que llamaremos **NAPT (Network Address Port Translation)**.

Por último, si la traducción de la dirección pública y la privada se realiza de acuerdo con una tabla de equivalencia que se carga en la configuración del dispositivo NAT y que no se modifica dinámicamente decimos que se trata de **NAT Estático**. En cambio si dicha tabla de equivalencia es gestionada dinámicamente por el dispositivo NAT de forma que las direcciones y/o números de puerto se puedan reutilizar decimos que tenemos un **NAT dinámico**.

Combinando el NAT Básico o el NAPT con las modalidades estática y dinámica obtenemos cuatro combinaciones de NAT que pasamos a describir a continuación:

- NAT básico estático. La tabla de equivalencia IP privada – IP pública está incluida en la configuración del dispositivo NAT y solo se modifica por decisión del administrador de la red. La correspondencia es biunívoca, es decir a cada dirección privada le corresponde una pública y

viceversa. Los números de puerto no se modifican. Con este tipo de NAT es preciso disponer de un número de direcciones públicas igual al de direcciones privadas. Este NAT tiene interés fundamentalmente en situaciones en las que se desea máxima sencillez y no se necesita reducir el número de direcciones públicas, por ejemplo se quiere conectar a Internet una red clase C privada y se dispone para ello de una clase C pública.

- NAT básico dinámico. La tabla de equivalencias de IP privada a IP pública se construye de forma dinámica, a medida que lo requieren los hosts. Las entradas caducan al terminar la conexión (TCP) o pasado un tiempo de inactividad (UDP), lo cual permite su reutilización. Los números de puerto no se modifican. El número de direcciones públicas puede ser inferior al de direcciones privadas, pero ha de ser suficiente para el número de ordenadores que se quieren conectar simultáneamente al exterior. Permite un ahorro de direcciones frente al NAT estático, pero al no modificar el número de puerto es mas fácil de implementar y tiene menos restricciones que el NAPT.

- NAPT (Network Address Port Translation) estático. En este caso la tabla de equivalencia se carga de forma estática en la configuración del equipo, pero las entradas incluyen no solo la dirección IP sino también el número de puerto TCP o UDP. El NAPT estático permite virtualizar servidores: por ejemplo es posible asignar el puerto 21 (servidor FTP) de una dirección pública del NAT al puerto 21 de un host en la red privada, y el puerto 80 (servidor Web) de la misma dirección a otro host de la red privada.

- NAPT (Network Address Port Translation) dinámico. En este caso la tabla de equivalencias se construye dinámicamente a medida que los hosts lo requieren, como en el NAT básico dinámico. Sin embargo, a diferencia de aquel las entradas de la tabla incluyen no solo la dirección IP, sino también el número de puerto. Las entradas caducan al terminar la conexión (TCP) o pasado un tiempo de inactividad (UDP), lo cual permite su reutilización. En este caso es posible aprovechar una misma dirección IP pública para conectar al exterior diversos ordenadores simultáneamente, ya que se aprovecha el número de puerto UDP o TCP para multiplexar conexiones de hosts diferentes. Este NAT permite un aprovechamiento máximo de las direcciones públicas, pero es el que tiene mayor complejidad de implementación.

Las cuatro modalidades antes descritas pueden coexistir en una misma red, por ejemplo utilizando NAT o NAPT estático para los servidores que deban ser accesibles desde el exterior y NAT o NAPT dinámico para el resto de los hosts.

Las modificaciones que NAT introduce en el paquete IP son las siguientes:

- Cabecera IP: Al modificar las direcciones de origen y/o destino el valor del campo checksum en la cabecera del datagrama cambia y por tanto ha de recalcularse.

- Cabecera de transporte (TCP/UDP): El campo checksum en la cabecera de transporte (TCP o UDP) ha de recalcularse, ya que la pseudocabecera incluye las direcciones IP de origen y destino. Además en el caso de NAPT se ha de modificar el valor del puerto de origen o destino.

- Mensajes ICMP: Los mensajes ICMP suelen incluir embebida la cabecera del datagrama IP y el principio de la cabecera TCP/UDP que originó el mensaje ICMP. El dispositivo NAT ha de localizar en el mensaje ICMP la dirección IP y modificarla; además ha de modificar consecuentemente el checksum de la cabecera IP embebida. En el caso de hacer NAPT se ha de modificar también el número de puerto TCP/UDP que aparece en la cabecera embebida.

- Mensajes SNMP. Los mensajes de gestión, que notifican cambios en la situación de los diferentes dispositivos, suelen llevar en su parte de datos direcciones IP que el NAT debe localizar y modificar.

Limitaciones de los NAT, a medida que aumenta el nivel de sofisticación aumenta como es lógico la complejidad del NAT. En el caso del NAT o NAPT dinámico el router ha de conservar una información de estado, ya que ha de mantener control de las conexiones existentes. Esta información de estado hace mas difícil establecer un router de backup ya que esta información de estado ha de trasladarse a dicho

router en caso de caída del principal. Además el router ha de tomar decisiones respecto a cuando termina una conexión para liberar la correspondiente entrada en su tabla (la de dirección IP en el caso de NAT o la de dirección IP + puerto TCP/UDP en el caso de NAPT). En TCP es bastante fácil detectar el cierre de la conexión a través de los segmentos con el bit FIN, pero en UDP no existe un procedimiento explícito de desconexión, por lo que en estos casos normalmente se fija un tiempo de inactividad a partir del cual una conexión se considera inexistente, o bien se espera a tener que liberar recursos y entonces se cierra la conexión que lleva más tiempo inactiva.

Incluso en el caso más sencillo del NAT básico estático se plantean problemas de difícil solución que hacen que determinadas aplicaciones no funcionen a través de estos dispositivos. Algunos ejemplos de situaciones en las que el NAT tiene dificultades o no funciona son las siguientes:

- Protocolo FTP. En el momento de establecer una conexión FTP los hosts intercambian una serie de mensajes que contienen entre otra información sus direcciones IP. Lógicamente estas direcciones han de modificarse. El problema es que esas direcciones se encuentran codificadas como caracteres ASCII, no en binario como ocurre normalmente. Así, si la dirección de origen es por ejemplo “200.200.200.1” (13 caracteres) y la de destino es “192.168.1.1” (11 caracteres) el proceso de traducción puede optar por rellenar el campo con ceros (“192.168.001.1”) para mantener constante el número de octetos. Pero cuando la traducción se realiza en sentido inverso es inevitable aumentar en dos octetos la longitud del segmento TCP correspondiente; como consecuencia de ello el NAT a partir de ese momento ha de incrementar en dos octetos los contadores de bytes que aparecen en los cmapos número de secuencia y número de ACK para esa conexión TCP; esto supone que el NAT ha de mantener una cierta *información de estado* para esa conexión mientras exista. Incluso puede darse el caso de que el aumento en dos octetos del segmento provoque la fragmentación del datagrama correspondiente.

- En general cualquier protocolo del nivel de aplicación que incluya en la parte de datos información sobre direcciones IP o números de puerto TCP/UDP supone un reto para un NAT, ya que la detección y modificación de dichas direcciones es difícil, compleja y requiere que el NAT analice información perteneciente al nivel de aplicación. Algunos ejemplos de esto son el protocolo de la ITU-T H.323 utilizado en videoconferencia y en telefonía por Internet (voz sobre IP); también se encuentran en este caso los juegos interactivos de Internet, las aplicaciones tipo Napster, etc. Normalmente el funcionamiento de estas aplicaciones a través de un NAT sólo se consigue cuando se dispone de una pasarela a nivel de aplicación, programa que realiza las modificaciones necesarias en los datos para que la aplicación pueda utilizarse a través de un NAT. Las pasarelas a nivel de aplicación sólo están disponibles para algunas aplicaciones estándar.

- La comunicación de aplicaciones NetBIOS sobre TCP/IP tiene problemas parecidos a las aplicaciones del apartado anterior: la información sobre direcciones IP aparece de forma no consistente y con desplazamientos variables en la información intercambiada por los hosts, lo cual impide que los NAT las modifiquen; como consecuencia no es posible utilizar TCP para transportar NetBIOS cuando se atraviesa un NAT.

- IPSec solo puede utilizarse de forma limitada a través de un NAT. Esto se debe a que IPSec incorpora una cabecera de autenticación (AH, Autentification Header) que permite al receptor detectar si el paquete IP ha sido modificado en ruta. Evidentemente la cabecera AH no incluye el campo TTL ni el checksum, ya que se sabe que estos campos cambian de valor durante el camino de un datagrama, pero si incluyen las direcciones IP de origen y destino. Como estas direcciones se modifican en el NAT el receptor cuando comprueba la cabecera AH detecta que el datagrama ha sido alterado y lo descarta. En el caso de utilizar IPSec en modo túnel y realizar el túnel en el mismo dispositivo que realiza el NAT las limitaciones se reducen.

A pesar de todas las limitaciones reseñadas NAT resulta un mecanismo muy útil en un amplio abanico de situaciones, por lo que es seguro que se seguirá utilizando NAT en Internet durante bastantes años, al menos hasta que se generalice el uso de IPv6. Con ese nuevo protocolo y su abundante espacio de direcciones parece difícil concebir un escenario en el que sea aconsejable utilizar NAT.

Los aspectos más relevantes de NAT se discuten en los RFC 1631 y 2663.

Detección de intrusos o IDS (Intrusión Detecction System)

Las vulnerabilidades de los diferentes sistemas dentro de una red son los caminos para realizar los ataques. En muchas ocasiones, el atacante enmascara el ataque en tráfico permitido por el cortafuegos y por tanto para delatarlo se necesita un IDS, y ambos son complementarios.

Las características deseables para un IDS son que esté continuamente en ejecución y debe poderse analizar él mismo y detectar si ha sido modificado por un atacante, utilizar los mínimos recursos posibles y adaptarse fácilmente a los cambios de sistemas y usuarios, por lo que en ocasiones poseen inteligencia para adaptarse (aprender por su experiencia) y configurarse.

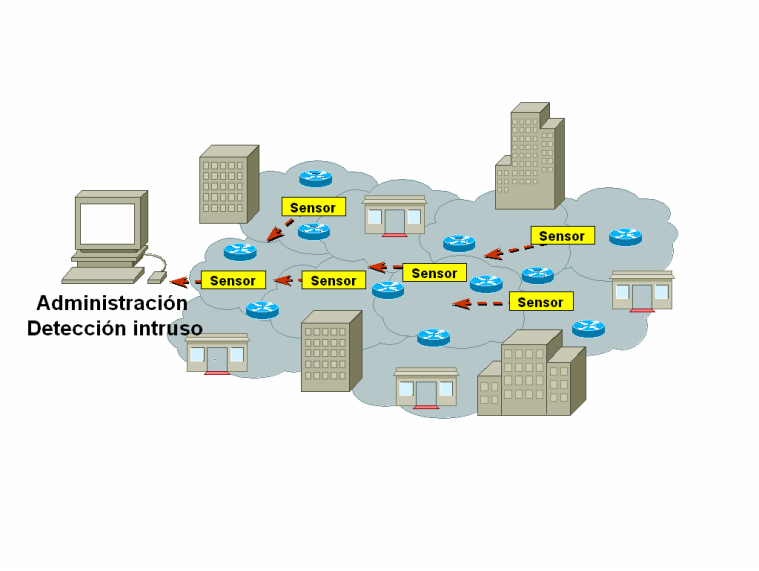


Figura 31: ejemplo de detección de intrusos en red

Una clasificación según su localización es:

- NIDS (Network Intrusion Detection System): detecta los paquetes armados maliciosamente y diseñados para no ser detectados por los cortafuegos. Consta de un sensor situado en un segmento de la red y una consola. La ventaja que tiene este tipo de cortafuegos es que no se requiere instalar software adicional en ningún servidor. Inconveniente: es local al segmento, si la información cifrada no puede procesarla
- HIDS (Host Intrusion Detection System): analiza el tráfico sobre un servidor. Las ventajas que tiene es que registra comandos utilizados, es más fiable, mayor probabilidad de acierto que NIDS.

Clasificación según modelos de detección:

- Detección de mal uso: verifica sobre tipos ilegales de tráfico, secuencias que previamente se sabe se utilizan para realizar ataques (conocidas como exploits)

- Detección de uso anómalo: verifica diferencias estadísticas del comportamiento normal de una red, según franjas horarias, según la utilización de puertos (evitaría el rastreo de puertos)

Clasificación según naturaleza

- Pasivos: registran violación y genearan una alerta

- Reactivos: responden ante la situación, anulando sesión, rechazando conexión por el cortafuegos, etc

Honey Pot (jarrones de miel)

En ocasiones es interesante aprender de los propios atacantes. Para ello, en las redes se ubican servidores puestos adrede para que los intrusos los saboteen y son monitorizados por sistemas que actúan como puentes a los servidores, registrando de forma transparente los paquetes que acceden a dichos servidores.

De esta forma, detectado un ataque (por modificación de la estructura de archivos), se recompone la traza del atacante (secuencia de paquetes registrados en el monitor puente) y se pasa a un análisis forense. Este análisis forense concluye, en caso de detectar un nuevo ataque, en una nueva regla de detección.

Seguridad en red basada en sistema centralizado

Encapsuladores (proxies) y pasarelas

Los envolventes son un invento reciente de la seguridad en UNIX. Aunque nacieron por la necesidad de modificar el funcionamiento de los sistemas operativos sin acceso a su código fuente, su uso como herramienta de seguridad se ha extendido por numerosas razones:

- La lógica de seguridad se encapsula dentro de un programa sencillo y fácil de verificar.
- El programa envuelto permanece separado del envolvente, lo cual permite actualizar el envolvente o el programa envuelto de forma individual.
- Como los envolventes llaman al programa envuelto mediante la llamada al sistema *exec()*, un mismo envolvente puede utilizarse para controlar el acceso a un enorme grupo de programas envueltos.

Envolvente de correo.

El correo electrónico es una fuente de problemas de seguridad. Para solucionarlo, se ha diseñado un envolvente de sendmail que puede obtenerse en la dirección de FTP ftp.tis.com como usuario anonymous. El envolvente de sendmail consiste en dos programas, smap y smapd.

El programa smap acepta mensajes de la red y los escribe en un directorio del disco. Aunque el programa se ejecuta con los permisos de *root*, lo realiza en un sistema de archivos restringido con *chroot*, desde el cual no se puede acceder al resto del disco. El programa es llamado por el gestor de demonios *inetd* y termina al recoger el correo.

El programa smapd revisa periódicamente el directorio donde escribe smap y envía los mensajes recibidos mediante el *sendmail* o algún otro programa.

Envolvente de acceso.

Un superservidor es un demonio o programa en ejecución en segundo plano en UNIX, que está pendiente de las peticiones externas a servicios determinados escuchando en los puertos de servicio, de forma que una vez establecida la conexión, lanza el proceso o demonio que atiende dicha petición. Así de esta forma se reduce el número de procesos en ejecución en el sistema.

Los envolventes participan en la parte lógica de control de acceso, que forma parte de la decisión al lanzar o no el servicio solicitado.

En *Unix*, el superservidor más utilizado es "*inetd*" cuya configuración se encuentra en el fichero "*inetd.conf*". Pero "*inetd*" no realiza ningún control de seguridad en el momento de lanzar un cierto servicio y por tanto, es necesario encapsular los servicios lanzados a través un programa, que controle el acceso a los servicios. De esta forma, el programa encapsulador puede englobar de forma sencilla toda la lógica de seguridad.

Uno de los encapsuladores más utilizados y de propósito general es *tcpwrappers (*y su demonio es conocido como *tcpd)*. El programa es de libre distribución, desarrollado por Wietse Venema y forma parte de la distribución de la mayoría de sistema operativos *Linux*. El programa *tcpwrappers* permite un alto nivel de control sobre las conexiones TCP y permite enviar un mensaje al cliente que se conecta, pudiendo hacerle advertencias legales y avisos (*banners*), ejecutar consulta doble de la dirección IP, comparar el nombre del cliente y el servicio solicitado con una lista de control de acceso, ejecutar órdenes o comandos, registrar la información mediante "*syslog*", emplear "*ident*" (RFC 1413) para averiguar el nombre del usuario, ...

El envolvente *tcpwrapper* posee dos formas de configuración, o bien modificando los ficheros de configuración, */etc/hosts.allow* y */etc/hosts.deny*, o bien a través de sólo */etc/hosts.allow* permitiendo un mayor número de opciones en su configuración, siendo este último el modo utilizado finalmente.

Al utilizarse conjuntamente "*inetd*" y *tcpwrappers* se ha desarrollado un nuevo programa superservidor extendido llamado "x*inetd*" (*Extended Internet Service Daemon*). Este programa, integra el encapsulador *tcpwrappers* a través de las librerías "*libwrap*". Adicionalmente a las funcionalidades de *tcpwrappers* incluye además accesos por franja horaria, evita el ataque DOS (*Deny of Service*), ... Estas librerías hacen uso de la configuración del fichero */etc/hosts.allow*, cuya estructura son entradas con la siguiente sintaxis está es:

lista_de_Demonios: lista_de_clientes : option : option : option: ...

donde:

-lista_de_Demonios: contiene servicios especificados en el fichero */etc/services* y cuyos programas generalmente se localizan en */usr/sbin*

-lista_de_clientes: patrones por nombres ".uv.es"(para todo el dominio de la *Universitat*), por direcciones "147.156." (para especificar 147.156.x.x), utiliza los comodines ALL (siempre cumple), LOCAL (perteneciente al dominio del servidor), KNOWN/UNKNOWN (usuarios cuyo nombre se conoce/desconoce), PARANOID (clientes cuyo nombre no encaja con su dirección IP). Estos comodines pueden modificarse con el operador *EXCEPT*.

-options : puede ser {*allow*|*deny*}, ejecución de órdenes ({*spawn*|*twist*}[1] *comando_de_shell*), emitir *banner*[2]*s*, ...

Dichos ficheros son evaluados y en el momento de la primera coincidencia de servicio y cliente, se ejecutan las opciones anexas. Si entre dichas opciones no aparece la opción {*allow*|*deny*}, se sobreentiende que se está permitido para la ejecución del servicio solicitado. Para ver con detalle las diferentes opciones, consultar el manual "*hosts.allow*" y "*hosts_options*".

El envolvente de acceso más conocido es tcpwrapper. El programa puede obtenerse usando FTP anónimo desde la dirección ftp.porcupine.org/pub/security/tcp_wrappers_XXX.tar.gz, donde XXX es la versión. El envolvente forma parte de la distribución de la mayoría de sistema operativos Linux.

El programa tcpwrapper permite un alto nivel de control sobre las conexiones TCP. El programa se inicia por el demonio *inetd* y entonces lleva a cabo una o más de las siguientes acciones:

- Envía un mensaje al cliente que se conecta, pudiendo hacerle advertencias legales y avisos.
- Ejecuta una consulta doble de la dirección IP. Con ello se asegura de que la dirección y el nombre del ordenador remoto coinciden. En caso de no ser así se almacena la información en un fichero de registro y puede opcionalmente cortar la conexión.

---

[1] Con "*spawn"* se lanza un proceso hijo que atiende la línea de comandos pasado, donde los *stdin, stdout, stderr* están conectados a /dev/null, para evitar conflicto con la comunicación con el cliente. Previo a la ejecución del hijo, se evalúan los parámetros y expansiones que contiene. Con "*twist"* los *stdin, stdout, stderr* están conectados al cliente remoto, con lo cual toma control del proceso que llevaba la evaluación de control de acceso. Este comando por tanto ha de ser el último en ejecutarse.

[2] El comando *banners* tiene como argumento el directorio donde se encuentran los ficheros que permite visualizar en el cliente un mensaje. El nombre de los ficheros coincide con el nombre del servicio solicitado

- Compara el nombre del cliente y el servicio solicitado con una lista de control de acceso, comprobando si el cliente o el cliente y el servicio particular están autorizados o denegados.

El envolvente tcpwrapper posee dos ficheros de configuración, */etc/hosts.allow* y */etc/hosts.deny*. Cuando sucede una conexión se realiza lo siguiente:

1. Se examina el archivo /etc/hosts.allow para comprobar si el par (ordenador, servicio) están permitidos.

2. Si no se encontró, se examina el archivo /etc/hosts.deny para comprobar si el par (ordenador, servicio) debe ser rechazado.

3. Si no se encontró el par (ordenador, servicio) en los ficheros anteriores, la conexión se permite.

Tcpwrapper posee un lenguaje de configuración muy simple y flexible, pudiendo configurar el envolvente como se desee.

SOCKS: Sock-et-S

Los Socks son proxies que actúan como representante local (apoderado) de un servidor remoto para atravesar el cortafuegos. Permite controlar de forma rigurosa el acceso a Internet de las diferentes conexiones realizadas entre elementos en ambas partes de un cortafuegos.

Para configurar SOCKS, hay que compilar las aplicaciones para modificar las llamadas a los sockets. Los ficheros de configuración en el servidor /etc/sockd.conf y en el cliente /etc/socks.conf.

Más información en http://www.socks.nec.com
uncionamiento de Socks es el siguiente, cuando un cliente SOCKS conecta a un servidor de Internet, el servidor SOCKS intercepta la conexión y la filtra en base a número de puerto, origen, destino, clave de usuario, ... En el caso, que sea admisible, puentea la conexión directamente con el cliente.

ANEXO 1: SPOOFING Y HIJACKING, SUPLANTACIÓN DE IPS Y ROBO DE SESIONES[3]

En redes de computadoras podemos distinguir dos tipos de ataques : pasivos y activos. En el primero, el atacante se dedica únicamente a escuchar de la red. Ésta es quizá la forma más habitual de penetración: un intruso que haya logrado acceso a un host de nuestra red, podría lanzar un programa sniffer y dedicarse a escuchar los 100 primeros bytes de todas las conexiones dirigidas a los puertos telnet, ftp, pop3, login y demás protocolos que pasen por la red nuestros contraseñas sin encriptar.

Para evitar esto han aparecido mejoras, como por ejemplo los contraseñas de un solo uso (one-time contraseñas), Kerberos, ... en donde el usuario tiene una lista de claves de acceso y usa una cada vez para registrarse en el sistema. De esta forma, si alguien logra ver nuestra clave de acceso circulando por la red no podrá utilizarla, ya que habrá caducado.

Con esta solución hacemos más robusta la autenticación por medio de contraseñas. Pero ¿qué ocurre si por debajo de ésta se realiza una autenticación por dirección IP ?, es decir, ¿ qué ocurre si dependiendo de la dirección IP de nuestro interlocutor actuamos de una forma u otra, pidiendo contraseña o no?. Esto puede llegar a ser un problema grave y lo veremos en el siguiente texto.

El spoofing es una técnica que consiste en engañar a un computador haciéndole creer que esta dialogando con un host cualquiera, cuando en realidad lo está haciendo con nosotros. Esto puede que no parezca una amenaza, ya que normalmente la autenticación se hace mediante contraseña, por lo que aunque en nuestra conexión con un servidor suplantemos la identidad de otro host, tendremos que conocer la clave de acceso para hacer uso de un servicio dado.

El problema aparece cuando algunos de nuestros hosts establecen una relación de confianza : entonces, si alguien es capaz de engañarles diciendo que es uno del grupo, esto podría causar estragos en la seguridad de nuestra red. Pero, ¿cómo se plasman esas relaciones de confianza en la realidad ?.

En UNIX existen un par de ficheros que permiten, por medio del comando rlogin, hacer conexiones sin necesidad de entrar ningún contraseña. Estos ficheros son los siguientes:

- $HOME/.rhosts
- /etc/host.equiv

El fichero .rhosts se encuentra en el directorio raíz de cada usuario y permite añadir pares 'máquina-nombre de usuario' que indican máquinas y usuarios en los que se puede confiar. Por ejemplo, supongamos que tenemos dos cuentas llamadas satomi en dos computadoras distintas, vega.domain.cxm y antares.domain.cxm.

Nos gustaría conectarnos de una a otra sin necesidad de pasar contraseñas por la red, por razones de rapidez y de seguridad, frustrando así a los sniffers. La forma de hacerlo sería añadir en /home/satomi/.rhosts de vega.domain.cxm la siguiente línea:

antares.domain.cxm satomi

y en antares.domain.cxm :

vega.domain.cxm satomi

Si además quisiésemos que el usuario adela de altair.algo.cxm pudiese acceder también a nuestra cuenta de antares.algo.cxm sin entrar contraseña, deberíamos añadir en el .rhosts de antares lo siguiente:

[3] Esta parte ha sido elaborado como trabajo de la asignatura de Redes por el alumno Emilio Mira.

altair.domain.cxm adela

El fichero /etc/host.equiv tiene un uso más general y tan solo puede ser con½gurado por el administrador del equipo. En él aparecen los nombres de las máquinas de las que aceptaremos conexiones sin pedir contraseñas, pero tan solo cuando el nombre de la cuenta del host remoto coincida con el nombre de la cuenta en el nuestro.

En nuestro ejemplo de antes, si el fichero /etc/host.equiv de antares contuviese :

altair.domain.cxm
vega.domain.cxm

cualquier usuario de altair y vega podría hacer rlogin a antares sobre una cuenta del mismo nombre que la remota sin necesidad de contraseña.

Cuando el comando rlogin envía una solicitud de conexión, el destino primero mira si el host que laoriginó se encuentra en /etc/host.equiv. De ser así iniciaría una sesión sin pedir contraseña sobre una cuenta con su mismo nombre de usuario. En caso contrario, buscaría en el .rhosts del usuario donde queremos iniciar una sesión el nombre de la máquina origen y el usuario que originó la conexión. Si existe, no pediría contraseña. Solo en caso contrario necesitaríamos entrar la clave de acceso.

Es ahora cuando nos damos cuenta de lo siguiente : ¾ Dónde recae la autenticación de este servicio ?. En la dirección IP del origen. En general, los llamados 'servicios R' de U nix , el sistema X Window y los RPC, basan su autenticación en la dirección IP del host origen, y es ahí donde recae el problema. Veamos ahora las posibles variantes del spoofing.

IP spoofing

El IP spoofing es sin duda el más habitual. En 1985, Robert Morris, que por entonces trabajaba en AT&T, ya dio cuenta de él en su artículo _A weakness in the 4.2BSD UNIX TCP/IP software_. Diez años más tarde, Kevin Mitnick utilizó esta técnica para asaltar sistemas informáticos del Gobierno de EEUU e importantes empresas y universidades, convirtiéndose en el cracker más buscado de la historia. La base del IP spoo½ng es la siguiente : un host puede enviar paquetes con una dirección IP distinta a la suya, haciéndole creer al receptor que está dialogando con otro cualquiera, y en el peor de los casos, con un host de confianza.

'Hijacking' o robo de sesión

Supongamos que estamos en una red Ethernet, donde varias estaciones de trabajo realizan conexiones TCP a un servidor que se encuentra en nuestro mismo segmento, y pongámonos en la piel de un intruso. .Si tenemos suficientes privilegios en nuestra máquina, podríamos utilizar un programa sniffer y escuchar todo lo que circula por este tramo, por ejemplo, una sesión telnet de una estación de trabajo con el servidor. En nuestra escucha pasiva a nivel de transporte podremos conocer el número de puerto y de secuencia en ambas partes con tan solo fijarnos en un segmento TCP. Esto es así porque en un solo segmento viajan los puertos origen y destino, el número de secuencia del emisor y el siguiente número de secuencia que se esperaba de la otra parte. Veamos el siguiente ejemplo :

Supongamos que somos el atacante y nuestro sniffer es el programa UNIX tcpdump. Estamos escuchando una conexión telnet entre una estación de trabajo con dirección IP 192.168.23.30, y un servidor con 192.168.23.1. Aparecen entonces dos paquetes :

192.168.23.30.1140 > 192.168.23.1.telnet: P 1320825536:1320825537(1) ack 1036160984 win 5840 <nop,nop,timestamp 4356776 390625> (DF)

192.168.23.1.telnet > 192.168.23.30.1140: P 1036160984:1036160985(1) ack 1320825537 win 5840 <nop,nop,timestamp 390646 4356776> (DF)

Esta sintaxis es la que utiliza el programa tcpdump. Los campos que nos interesan son los siguientes :

> dirección_origen.puerto > dirección_destino.puerto : flags_TCP num_de_secuencia_inicial : num_de_secuencia_final (tamaño_del_los_datos_TCP) num_de_ack opciones

Vemos como el primer segmento va del cliente al servidor y es, seguramente, una tecla pulsada. El segundo es la respuesta del servidor reconociendo al cliente su último envío y enviándole la misma tecla para que pueda hacer el eco, tal y como lo establece el protocolo telnet.

Es entonces cuando el atacante podría entrar en juego: podría enviar un paquete IP con la dirección origen de la estación de trabajo, dirección destino del servidor, puerto origen y destino el que corresponda, y número de secuencia y ACK el calculado a partir de lo que escuchó. De esta forma, el servidor reconocería este paquete como válido y lo aceptaría aumentando el número de secuencia que espera de la estación de trabajo:

> 192.168.23.30.1140 > 192.168.23.1.telnet: P 1320825537:1320825560(23) ack 1036160985 win 5840 <nop,nop,timestamp 286213 6365981> (DF)

> 192.168.23.1.telnet > 192.168.23.30.1140: P 1036160985:1036161018(23) ack 1320825560 win 5840 <nop,nop,timestamp 6412341 286245> (DF)

¿ Qué ocurrirá con la estación de trabajo ?. Simplemente, que cuando envíe paquetes al servidor, éstos serán rechazados, ya que sus números de secuencia no corresponderán con el esperado; el servidor pensará que son paquetes duplicados. El propietario legítimo de esa conexión verá como, sin causa aparente, su sesión se ha colgado, y normalmente iniciará una nueva sin darle mayor importancia. Al fin y al cabo esas cosas ocurren.

Mientras tanto, el intruso podría haber enviado un paquete que contuviese un comando a su elección, con el consiguiente riesgo para el usuario legítimo.

Spoofing simple

Veamos ahora otra forma por la cual un atacante podría penetrar en un sistema. Supongamos el mismo escenario que en el caso del hijacking : cliente, servidor y atacante en el mismo segmento Ethernet. Supongamos que el usuario satomi de la máquina cliente tiene otra cuenta con el mismo nombre en el servidor, y para facilitar su conexión desde su lugar de trabajo habitual ha añadido en su .rhosts del servidor la siguiente linea :

> *cliente.algo.cxm satomi*

El atacante conoce esto e intenta lo siguiente: hace un programa que emule el funcionamiento del comando rlogin, pero que en la dirección origen de los paquetes que envía ponga la dirección IP de cliente.algo.cxm y en los campos 'usuario local' y 'usuario remoto' que el protocolo rlogin utiliza escriba satomi. Además, este programa deberá poner la interfaz en modo promiscuo para ver las respuestas que el servidor le da al verdadero cliente. Una vez está listo, el atacante empieza la prueba.

Su programa comienza con una conexión TCP al puerto 513 del servidor. Para ello, envía el siguiente segmento :

> 192.168.23.30.1022 > 192.168.23.1.login: S 915208618:915208618(0) win 32120 <mss 1460,sackOK,timestamp 963912 0,nop,wscale 0> (DF)

Vemos como el segmento enviado usa el flag SYN y el número de secuencia 915208618 para establecer la conexión.

Cuando el servidor lo recibe, envía su respuesta a la solicitud de conexión con un número de secuencia elegido por él y aceptando el número de secuencia del cliente incrementándolo en una unidad :

192.168.23.1.login > 192.168.23.30.1022: S 184994651:184994651(0) ack 915208619 win 32120 <mss 1460,sackOK,timestamp 53591392 963912,nop,wscale 0> (DF)

Para finalizar el saludo a tres vías de TCP, el intruso envía el último paquete al servidor :

192.168.23.30.1022 > 192.168.23.1.login: . ack 184994652 win 32120 <nop,nop,timestamp 963931 53591392> (DF)

Debemos darnos cuenta que, para construirlo, ha necesitado ver el número de secuencia que el servidor ha elegido, el 184994651, y lo ha incrementado en uno en señal de aprobación. Esto no habría sido posible si la interfaz no hubiese estado en modo promiscuo, ya que ese paquete no iba dirigido a él, sino al verdadero cliente. Es más, para que esta técnica funcione, el atacante debe estar situado en el mismo segmento Ethernet del servidor o del cliente, en el caso de que sean distintos.

Pero antes de enviar este último paquete, el atacante se da cuenta que un paquete viajó desde el verdadero cliente hasta el servidor.

192.168.23.30.1022 > 192.168.23.1.login: R 0:0(0) ack 184994652 win 0

Éste es un paquete que lleva el flag RESET . ¿Qué ocurrió?. Simplemente, que la computadora cliente estaba encendida, por lo que respondió al servidor con lo siguiente : _No quiero hacer ninguna conexión al puerto 513_. Cuando esto llega al servidor se aborta el intento de conexión.

Obviamente el spoofing simple no acaba aquí. ¾ Qué habría ocurrido si la máquina cliente hubiese estado apagada ?. Jamás habría respondido al servidor por lo que nuestro intruso podría haber seguido con el ataque. Pero ahora aparece un nuevo problema : si el cliente esta apagado no podrá responder a la consulta ARP que el servidor realiza para conocer su dirección MAC (es bastante probable que se tenga que hacer esta consulta si el cliente estaba apagado), por lo que jamás responderá a una solicitud de conexión por parte del falso cliente. Para evitarlo, es el intruso el que responde a la consulta ARP complicando un poco más su software.

Una vez se ha establecido la conexión, al intruso tan solo le resta seguir el protocolo del rlogin enviando el nombre de usuario local y el remoto, y acabará teniendo una sesión sin el consentimiento de su propietaria.

Por si esto fuese poco, para el caso en el que el verdadero cliente esté encendido, en algunos sistemas operativos es posible bloquear momentáneamente la escucha sobre un puerto. A esta técnica se le conoce como SYN flooding. Consiste en enviar en un breve periodo de tiempo muchas solicitudes de conexión (paquetes SYN) sin enviar el tercer paquete del saludo a tres vías. De esta forma se consigue llenar la cola de conexiones entrantes con conexiones medio abiertas, provocando que, hasta que no caduquen, ninguna solicitud de conexión será tramitada. Después del SYN flooding sobre el cliente, el intruso tendría tiempo para lanzar la conexión sobre el servidor sin que el cliente interfiriese en este proceso.

Para finalizar, comentar que hemos supuesto que el atacante conocía la existencia y el contenido del .rhosts de satomi en el servidor. Esto no es tan fácil, puesto que este fichero se halla dentro de $HOME. Pero es posible probar esta técnica de forma indiscriminada sobre todos los usuarios del servidor usando como IP origen la de sus respectivos puestos de trabajo, o cualquier computadora desde donde hallan hecho conexiones al servidor.

Spoofing a ciegas

Este tipo de spoofing es más complicado y difícil que el anterior, aunque igual de peligroso. El escenario ahora es distinto : supongamos que el intruso intenta llevar a cabo el ataque que vimos antes pero desde fuera del segmento del cliente y del servidor y fuera también de la ruta que los une. ¾ Cual es el problema ahora?. Esta vez, cuando su programa envíe el primer paquete para la conexión TCP, se quedará esperando la respuesta del servidor para conocer qué número de secuencia eligió, pero esta respuesta nunca llegará, ya que ahora este paquete de respuesta no pasará por él. Si no conoce el número de secuencia que el servidor eligió no será posible establecer una conexión TCP.

Pero el protocolo IP puede hacerle la vida más fácil a nuestro intruso. Existen un par de opciones denominadas 'enrutamiento estricto desde el origen' y 'enrutamiento libre desde el origen', que básicamente sirven para lo mismo : indicarle a los routers por donde encaminar nuestros paquetes y los paquetes de vuelta. De esta forma el atacante incluirá en sus paquetes esta opción con los routers que debe atravesar para que los paquetes devueltos por el servidor no vayan al verdadero cliente, sino que lo hagan a su máquina. Afortunadamente, es raro que los routes actuales hagan caso a esta opción de un paquete IP, por lo que no tendremos en cuenta esta posibilidad.

Otra forma de conseguir un rerouting sería modificar la información de los routers. Esto puede ser complicado en caso de un routing estático, pero cuando en nuestra red usamos routing dinámico, el atacante podría enviar información de routing falsa al router, dependiendo claro está del protocolo que utilice (RIP, EGP, ...). De todas formas no trataremos este caso aquí.

Volvamos a cual era la traba que le aparecía al intruso : le era imposible ver el número de secuencia que el servidor elegía para iniciar la conexión. Pero, ¿de qué forma genera un host ese número de secuencia?.La especificación advertía originalmente que el número inicial de secuencia debería generarse a partir de un reloj interno de 32 bits que se suponía se incrementaba aproximadamente cada 4 microsegundos, aunque la realidad es muy distinta. En la mayoría de sistemas actuales podemos encontrar tres formas de generar estos números :

- La regla 64K : Esta regla es sorprendentemente simple. Se elige un número inicial cuando arranca el host, y se incrementa cada segundo por una constante, normalmente 128,000. Cuando una conexión es iniciada este contador aumenta en 64,000.

- Incremento por unidades de tiempo : esta es la regla más parecida a la especificación, en donde cada x unidades de tiempo se incrementa el contador, pero cada fabricante la implementa de una forma distinta, eligiendo unidades de tiempo distintas. Además, dependiendo de la carga del sistema este incremento no será ni mucho menos perfecto. Como ejemplo, en los antiguos núcleos de Linux el generador de números de secuencia se incrementaba en 1 cada microsegundo.

- Aleatorio : un ejemplo lo tenemos en los núcleos de Linux actuales.

Visto esto, el intruso podría optar por intentar adivinar el número de secuencia que va a elegir el servidor. Esto no parece muy difícil si el objetivo usa la regla 64K., pero es casi imposible si su generador de números de secuencia es aleatorio.

Supongamos ahora que el servidor utiliza la regla 64K. Esto se puede saber haciendo varias conexiones consecutivas : si los números de secuencia que recibimos difieren en 64,000 o en 192,000 sin duda estamos en este caso. Es entonces cuando el intruso empieza el ataque. Ahora su software se complica, debe averiguar, mediante la forma que hemos comentado antes, cual va a ser el número de secuencia que el servidor elegirá.

Recibe los siguientes números de secuencia como respuesta a sus conexiones :

1. 1217261284 2. 1217325284 3. 1217389284 4. 1217581284

Vemos como la diferencia entre 1-2 , 2-3 es de 64,000 y entre 3-4 es de 192,000 por lo que es probable que el siguiente número de secuencia elegido por el servidor sea 1217581284 + 64000, siempre y cuando el tiempo que tarda el paquete SYN en llegar al servidor sea de menos de un segundo. Así que el intruso prueba este número y envía estos paquetes separados por un pequeño instante de tiempo :

192.168.23.30.1177 > 192.168.23.1.login: S 712432833:712432833(0) win 32120 <mss1460, sackOK, timestamp 395512 0,nop,wscale 0> (DF)

192.168.23.30.1177 > 192.168.23.1.login: . ack 1217645284 win 32120 <nop,nop,timestamp 963931 53591392> (DF)

El primero inicia una conexión y el segundo confirma el número de secuencia elegido por el servidor. Ahora bien, él cómo sabe el atacante si su intento de conexión ha tenido éxito ?. Simplemente no lo sabe, ya que nunca va a recibir ningún paquete procedente del servidor. El único que los recibirá será el legítimo cliente, el cual habrá sido adormilado mediante un SYN flooding o algo similar que consiga el mismo efecto. Después de esto, al cracker solo le resta enviar la misma información que usaba para emular el protocolo rlogin y esperar a que funcione.

Pero a primera vista puede parecer que algo se nos ha pasado por alto : ¿ qué ocurre con la información que envía el servidor ?. Si el atacante no puede ver ningún mensaje del servidor, acabará perdiendo la pista del número de ACK que debe enviarle para reconocerle que han llegado sus segmentos TCP. Pero esto no tiene demasiada importancia, ya que cuando el servidor vea que no ha recibido ACK tras un tiempo dado, comenzará a retransmitir. Aunque el tipo de retransmisión depende del algoritmo utilizado, esto es algo que no interfiere con los segmentos del intruso, que serán todos aceptados.

Detectar y evitar el IP spoofing

La detección de estos ataques no es fácil, pero se pueden hacer algunas cosas. Por ejemplo, en la shell sh de UNIX (bash en Linux) existe un fichero de historia que almacena los últimos comandos ejecutados. Con la ayuda de este fichero podemos descubrir si hemos sido víctimas de un ataque hijacking : si en algún momento se cuelga nuestra sesión telnet con otro computador, podemos comprobar qué comandos se ejecutaron por última vez viendo este fichero de historia. Si vemos comandos que no ejecutamos nosotros del tipo :

echo atacante.domain.com evil >> $HOME/.rhosts

sin duda hemos sido víctimas de un robo de sesión.

Para detectar ataques del tipo spoofing 'a ciegas' podemos incorporar al software servidor un tcpwrapper o envolvente que se encargue de registrar las conexiones anómalas, como por ejemplo las que no trasmiten datos (consisten en el establecimiento de la conexión mediante el saludo a tres vías y la inmediata desconexión), las que se quedan medio abiertas (tan solo envían el primer segmento SYN), etc. De esta forma estaremos registrando parte de los intentos de adivinar el siguiente número de secuencia, como ya se explicó antes.

Pero esto no es infalible, ya que si nuestro host fuese un servidor ftp público o un servidor web, cualquiera podría hacer conexiones consecutivas aparentemente normales a esos servicios pero con el fín de conocer el algoritmo de generación de números de secuencia.

Para evitar estos ataques podemos adoptar algunas medidas. El uso de conmutadores en nuestra red soluciona completamente el problema del robo de sesión y el spoo½ng simple, ya que evitan las escuchas de conversaciones ajenas. El spoofing 'a ciegas' requiere otras medidas como pueden ser :

- Configuración de los routers que dan acceso al exterior para que no permitan la entrada de paquetes con una dirección origen perteneciente a nuestra red.
- Uso de sistemas con generadores de números de secuencia aleatorios para evitar que puedan ser adivinados.
- Uso de sistemas protegidos al ataque de SYN flooding.
- Evitar en lo posible el uso de los ½cheros .rhosts y /etc/host.equiv .

Pese a que este ataque es tan conocido, algunos sistemas operativos como el IRIX 6.2 ó 6.5 usan generadores de números de secuencia fáciles de adivinar, y otros como el HP-UX 10.20 implementan la regla 64K.

Existen escaneadores de puertos como el nmap que además nos dice qué sistema operativo se ejecuta en el host que estamos escaneando y la dificultad de adivinar sus números de secuencia, por lo que un racker que encuentre una máquina que utilice la reglas 64K. no dudará en hacer un seguimiento de ésta. Si, además, el host en cuestión tiene abierto el puerto finger, el intruso podrá ver gratuitamente los nombres de usuario y las direcciones de las máquinas desde las que hicieron su conexión, y a partir de ahí construirse una base de datos para luego hacer una prueba masiva.

ARP spoofing

El Protocolo de Resolución de Direcciones (ARP) también puede ser utilizado para un ataque de spoofing. Veamos un ejemplo de como podría llevarse a cabo por un intruso :

Supongamos que en nuestra LAN tenemos dos computadoras : sirio, un servidor, y algol,un PC Linux usado por satomi. El usuario satomi, también tiene una cuenta con el mismo nombre en sirio, y ha añadido en el fichero .rhosts de éste último la siguiente entrada :

algol satomi

Esto facilitará sus conexiónes por rlogin. Además, como los administradores son un poco paranoicos, para evitar el IP spoofing toda la red está formada por conmutadores y el servidor tiene un generador de números de secuencia aleatorios.

En nuestra LAN también existen otros hosts, y uno de ellos está siendo utilizado por un trabajador descontento para lanzar un ataque sobre sirio. El intruso conoce la existencia en sirio del ½chero .rhosts de satomi, y también conoce las medidas de seguridad que los administradores han adoptado. Es entonces cuando se le ocurre lanzar un ataque de spoo½ng ARP contra sirio suplantando a algol.

El ataque consiste en enviar a sirio información ARP falsa diciendo que la dirección MAC del host del atacante corresponde con la dirección IP de algol. ¾ Cómo se puede hacer esto ?. Muchas implementaciones de ARP, cuando escuchan una consulta ARP (recordemos que la consulta es broadcast), fichan esa dirección MAC aunque ellos no sean los buscados. Lo que podría hacer el intruso sería enviar una consulta ARP buscando a un host inexistente y, dentro del paquete ARP, cambiar su dirección MAC por la dirección MAC dealgol. Tan solo tiene que hacerlo dentro del paquete y no en la cabecera MAC porque lo que recibe el software ARP de sirio es la consulta libre de la cabecera MAC, que fue eliminada por la capa de nivel de enlace.

Una vez hecha la consulta, sirio incorporará en su cache ARP esta información, y cuando se quiera comunicar con algol sus paquetes irán dirigidos a la dirección IP de algol pero a la dirección MAC del intruso, y éste se podría aprovechar la relación de con½anza que existe entre el sirio y algol.

Una forma de frustrar este ataque es establecer asignaciones estáticas de direcciones hardware, pero si se invento ARP fue precisamente para evitar esto. En redes pequeñas quizá pueda llevarse a cabo, pero cuando una LAN es demasiado grande esto puede requerir mucho tiempo y esfuerzo.

Otra solución menos incomoda sería el ARPWATCH, un programa que vigila los cambios en la cache ARP y nos informa a través del correo electrónico y ½cheros de log de posibles variaciones.

DNS spoofing

Este es quizá el menos habitual de los ataques de spoofing, pero conviene conocerlo. El ½n del intruso es engañar a un host con una correspondencia 'nombre del host-dirección IP' falsa. Esto se puede conseguir de tres formas :

1. Falsificando una respuesta del DNS al host que se quiere atacar, de tal forma que la información alterada quede en su cache.
2. Modificando una transferencia de zona de un servidor de nombres primario a uno secundario.
3. Logrando acceso al DNS primario y modificando su base de datos para que una consulta posterior por parte del host a atacar le devuelva información falsa.

Siguiendo con el ejemplo anterior, supongamos ahora que los administradores de sirio han detectado ataques ARP spoofing con la ayuda del ARPWATCH y deciden hacer permanente la cache ARP solamente en este servidor. El intruso se da cuenta de esto y, conociendo una vulnerabilidad del DNS, logra acceso a él. Entonces modifica la entrada de algol cambiando la dirección IP que había por la suya.

Cuando sirio reciba una conexión del intruso como usuario satomi, mirará el fichero .rhosts de satomi y vera que si la conexión se hace desde algol no requerirá passsword. Hace una consulta al DNS para

conocer la dirección IP de algol y el servidor de nombres le responde con la IP del intruso. sirio comprueba el origen de la conexión rlogin y ve que coincide con la IP devuelta por el DNS, por lo que acepta su conexión y no le pide ningún tipo de contraseña.

Este ataque también se podría haber llevado a cabo sin haber conseguido acceso al servidor de nombres, simplemente falsificando la respuesta del DNS ante la consulta que hace sirio o modificando una transferencia de zona de un primario a un secundario en el caso de que la consulta de sirio fuese al secundario.

Si el DNS tiene contacto directo con la red externa, para evitar que un intruso pueda explotar un fallo en su implementación o en su configuración podríamos aislarlo de la red. Podemos conseguir este aislamiento utilizando dos servidores de nombres y un cortafuego. El cortafuego se situaría entre nuestra red y la red exterior, y colocaríamos un DNS al frente del cortafuego y otro detrás. El de detrás del cortafuegos almacenaría todos los nombres y direcciones IP de nuestra red, y sería el encargado de responder consultas a hosts internos. El que se encuentra al frente estaría vacío de información y se dedicaría a responder consultas de hosts externos preguntando al DNS interno. Así, nuestro cortafuego filtraría las conexiones externas al servidor de nombres interno y tan solo permitiría consultas de fuera que viniesen del DNS al frente, evitando cualquier intento de irrumpir en el servidor de nombres que contiene toda la información de nuestra red.

Si además configuramos nuestro router de entrada para que no deje pasar paquetes de fuera con una dirección origen perteneciente a nuestra red, evitaremos cualquier intento de spoo½ng sobre el DNS interno suplantando al externo.

ANEXO 2: ATAQUES DE DENEGACIÓN DE SERVICIO[4]

El DoS (Denial of Service), también conocido como Negación del Servicio, no es mas que una forma voluntaria o involuntaria (normalmente con propósito) de dejar sin servicio a un nodo de la red. Suele ser voluntaria (utilizada por los típicamente llamados hackers) y orientada hacia nodos servidores.

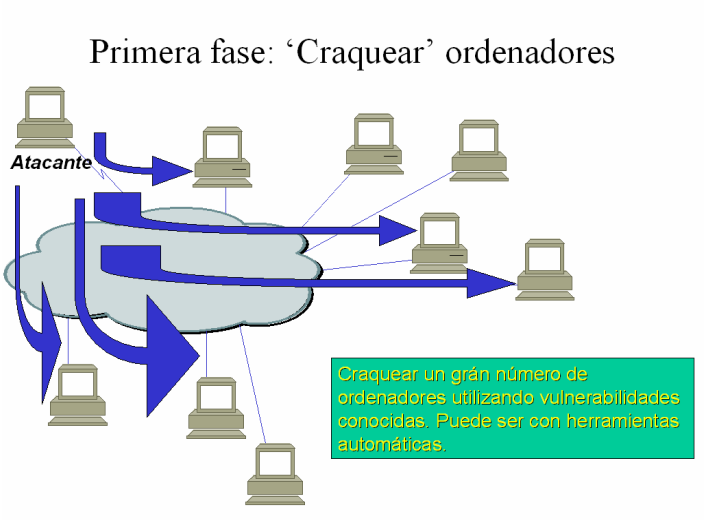
Esto se consigue mediante el uso abusivo de recursos de la máquina, mediante "inundación" de peticiones por red, etc... aunque hay muchas formas mas. El unico propósito de esto cuando es voluntario, puede ser la intención de aislar una red dejando sin servicio el nodo que conecta con el exterior, reconducir el tráfico actuando falsamente como dicho nodo, etc... aunque tambien es muy utilizado, como es lógico, por administradores de sistemas y demas operadores para comprovar la vulnerabilidad de su red.

El problema del DoS es que cualquier red es vulnerable a ello independientemente del Sistema Operativo utilizado (aunque muchas veces suele ser el centro del error). Esto sucede, ya que se basa en la ineficiencia del sistema frente a inundaciones (flooding) de paquetes, también debido a los desbordamientos de buffers, pilas y demas estructuras. Todo esto unido a fallos en los protocolos, como el IPv4 y aprovechamiento de las capacidades de los UDP, los ICMP, etc...

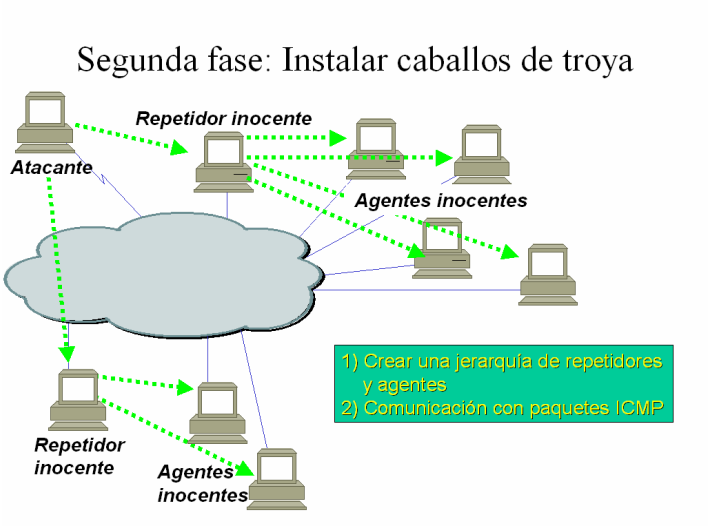
Las formas de crear un DoS son muy diversas. Las mas conocidas son: Syn Flooding, Echo/Chargen, Smurf, ataques DNS, TearDrop, Spool, Proquota, ICMP Flooding, UDP spoofing, Ping of Death, Snork, trinoo, TFN2, stacheldraht, etc... otros mas simples y normalmente accidentales son el nonblocking I/O, bucle fork(), Stacks & Buffers, etc... (estos se sulen basar en errores de programación).

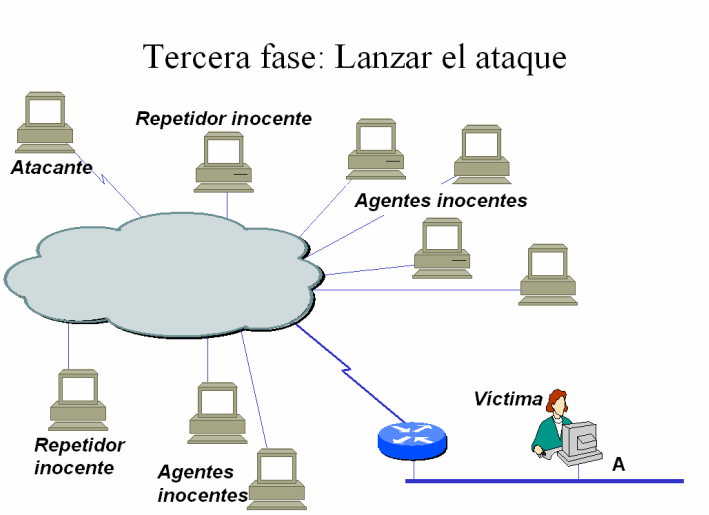
Los ataques DoS se podrian clasificar muy basicamente en: remotos y locales. Remotos serian aquellos realizados desde un host distinto a la red de la maquina, mientras que locales serian aquellos relizados en la propia máquina objetivo. Dentro de los remotos podemos incluir los DDoS (Distribuited Denial of Service)

Veamos un ejemplo en las siguientes figuras:



[4] Este anexo corresponde en parte a un trabajo de la asignatura de Redes, entregado por el alumno Alberto Chovares





Un DDoS es un DoS coordinado realizado desde multiples hosts hacia un objetivo. Utilizando la arquitectura Cliente/Servidor se puede iniciar un DoS a un objetivo desde cada host utilizando uno como "master" que a traves de dicha arquitectura de programacion controla a los demas. De este modo se puede conseguir inundar al objetivo con miles de peticiones provocándole una "negacion de servicio".

Vamos a ver los tipos básicos de DoS....

Errores de programacion

Estos suelen ser causados involuntariamente y su error provoca un DoS. En el primer ejemplo se produce un uso abusivo de los recursos y en el segundo un bloqueo. La gran ventaja de este tipo de DoS, es que son fácilmente detectables y corregibles.

EJEMPLO 1

```
main( )
{
        while (1)
                fork ( );
}
```

Aunque este simple programa esta escrito con proposito, muchas veces suele ocurrir la aparicion de un fork( ) reiterativo, en este caso por culpa de un bucle... Esto hace que la máquina consuma todos los recursos disponibles para estos procesos a medida que van aumentando exponencialmente. Normalmente esto se puede evitar limitando el numero de procesos permitidos a un usuario.

EJEMPLO 2

```
.
.
.
for (i=0; i<= maxi; i++)
{
        if ( (sockfd=client[i])<0)
                continue;
        if (FD_ISSET(sockfd,&rset))
        {
                if ( (n=Readline(sockfd,line,MAXLINE) )==0)
                {
.
.
.
```

Este es un fragmento de código extraido de un servidor TCP para mostrar otro de los problemas. Aquí el error se encuentra, en que cada vez que el servidor acepta un cliente, lee un linea (un numero determinado de bytes) del socket y sigue con los demás. El problema aparece cuando un cliente envia simplemente un byte y por las razones que sean ya no envia mas. Esto produce bloqueo DoS en el servidor, este problema de programación se conoce tambien como *nonblocking I/O*.

Consumo de ancho de banda

Este es el tipo de DoS mas utilizado, se suele utilizar en una red local o particular, pero es mas comun consumir recursos remotamente. Existen dos formas básicas de que se produzca:

Escenario 1
El host remoto que provoca el DoS en el servidor lo consigue porque tiene un ancho de banda mayor. Por ejemplo, una conexión T1(1’544Mbps) inunda un enlace de 128Kbps. Este tipo de negación de servicio se suele producir en redes de baja velocidad.

Escenario 2
Suponiendo un ancho de banda menor se puede “amplificar” uniendo varios sitios para inundar la red objetivo. Esto seria utilizando un DDoS. Por ejemplo inundar una línea T3(45Mbps) mediante la unión de muchas líneas de 56Kbps hasta superar el ancho de la T3.

Inanicion de recursos

Este difiere del consumo de ancho de banda, en que esta mas enfocado al consumo de recursos del sistema que de recursos de red. Generalmente esto se produce por la saturacion de la CPU, memoria, sistema cuotas, de archivos... Esto produce normalmente un fallo general del sistema.

Enrutamiento

Un ataque DoS basado en enrutamiento consiste en que los atacntes manipulan las tablas de distribucion o enrutamiento, produciendo así la negación del servicio a los sistemas legítimos. La mayoria de los

protocolos de enrutamiento, tales como RIPv1 (Routing Information Protocol) y BGPv4 (Border Gateway Protocol), carecen o tienen una autenticación muy sencilla.
De esta manera se puede crear un DoS y/o en su defecto enrutar o redirigir el tráfico de forma diferente, incluso a un "agujero negro" (una red que no existe).

DNS

Los DoS sobre servidores de nombres de dominio (DNS), son tan problemáticos como los anteriores, almacenando información falsa en la caché del servidor. Esto hace posible al igual que anteriormente redirigir al host que consulta el DNS a otras direcciones o incluso a ninguna. Hoy en dia esto es muy utilizado, haciendo que grandes sitios web queden inaccesibles durante cierto tiempo.

Explicando ya, formas concretas y para no comentar todas las condiciones DoS imaginables, comentaremos las mas relevantes, conocidas y utilizadas....

Smurf y Fraggle

Este tipo de ataque realiza una petición de ping difundida y dirigida una red de sistemas que responderá. Se puede enviar a un direccion de la red o una dirección de difusión. Este requiere un dispositivo que ejecute la función de difusión de la capa 3 (IP) a la capa 2 (red).

En una red de clase C (24 bits) tendriamos que la dirección de red sería .0 y la de difusion sería .255 (este tipo de difusión se utiliza para diagnósticos sobre el funcionamiento de las propias máquinas).
Este ataque envia paquetes ICMP ECHO falsificados a la red de difusión que actuará de red amplificadora. La dirección de origen del paquete ICMP ECHO debe ser la dirección del host/server objetivo, mediante *spoofing* (flasificando el paquete). Esto provocará la respuesta de toda la red hacia el objetivo, terminando en un DoS.

Esta es una variante del Smurf que en vez de utilizar paquetes ICMP utiliza UDP asignandoles a la direccion origen la direccion del objetivo igual que antes. La diferencia es que ha de recurrir a enviar a la red "amplificadora" y concretamente al puerto 7 (echo) de sus hosts. De esta manera devuelven un "echo" de lo recibido y se obtiene el mismo resultado anterior, DoS.

Para evitar esto, es posible desactivar la función de difusiones dirigidas de los routers, también se pueden configurar los sistemas operativos para rechazar paquetes echo de difusión. Ademas se puede limitar el trafico de ICMP, UDP en los routers.

SYN flooding

Hasta que llegaron tipos nuevos de DoS (smurf, fraggle...) este era la condición de DoS más utilizada. Este es conocido como inundación SYN.

Cuando se inicia una comunicación TCP, comienza un proceso de tres vias que consiste en: un paquete SYN del cliente al servidor (estado *listen*), un paquete SYN/ACK de respuesta del servidor (estado *syn_recv*) y por ultimo un ACK desde el cliente (estdo *established*). Cuando la conexión no se ha establecido totalmente, se dice que es una conexión potencial. Normalmente el protocolo designa pocos recursos a una conexión potencial (que aun no está establecida) y por tanto es facil provocar un DoS.

Supongamos un sistema A que envia un paquete SYN al sistema B (objetivo) pero mediante IP Spoofing (se conoce como spoofing a falsear la direccion origen, en este caso un host inexistente). En este momento B enviará un SYN/ACK a la dirección inexistente (si existiera, B recibiria un paquete RST de respuesta indicando que no ha pedido comunicación). Como no recibe ACK inmediatamente se crea una cola que según el protocolo dura cierto tiempo (estdo *syn_recv*). En este momento si B recibe mas peticiones del mismo modo (antes de terminar el tiempo para la cola) agotará sus recursos porque no podra eliminar las colas y caera en DoS.

Aumentar el tamaño de la cola es una de las soluciones posibles aunque tiene como desventaja un consumo de recursos que puede afectar al sistema. Otra solución podria ser disminuir el tiempo de establecimiento de conexión aunque no es realmente buena. Hoy en dia los sistemas operativos ya tienen soluciones diversas. Linux por ejemplo utiliza SYN "cookie", que contraresta el flood de SYN mediante criptografia y permite a los usuarios legítimos seguir conectandose. WinNT utiliza la asignación de recursos adicionales cuando la cola cae por debajo de un umbral.

TearDrop / NewTear / SynDrop / Bonk / Boink / SSPing / Targa / ...

Este tipo de ataque explota las vulnerabilidades del código de ensamblaje de paquetes de las implantaciones específicas de las pilas IP. Como los paquetes atraviesan diferentes redes, puede ser necesario fragmentarlos basandose en la maxima unidad de transmision en redes, MTU (Maximum Transmission Unit).

Algunos kernels Linux, Win95 y WinNT tenian el problema de comprobar si los paquetes eran demasiado grandes (para fragmentarlos), pero no si eran demasiado pequeños... Con esto, enviando paquetes de un tamaño específicamente pequeño podia caer el sistema en DoS debido a un error de ensamblaje.

Hoy en dia este problema, de momento, esta corregido para los nuevos kernels Linux y existen service packs para WinNT i patch para Windows 9x.

Ping of Death: ping de la muerte

Los paquetes IP (segun la RFC-791) pueden ser mayores de 65535 bytes de longitud incluida la cabecera (20 bytes si no se especifican opciones). Para los paquetes mayores la subcapa no puede reconstruirlos. Esto consiste en enviar paquetes ICMP tan grandes que al reconstruirlos, se desborden los buffers provocando DoS. (Actualmente este problema esta corregido y habian patch para Win95 y WinNT)

EJEMPLOS

ping -l 65510 direccionIP
ping -165527 -s 1 direccionIP

Snork

Este metodo consiste en enviar, mediante spoofing, paquetes UDP al puerto 135 de un servidor. (En este caso el spoofing consiste en incluir como direccion origen la de otro servidor). Cuando el servidor B recibe el paquete UDP parece que se ha equivocado y le envia un REJECT al servidor A , el cual responde con otro REJECT. Esto solo dura unos minutos hasta que "rompan" el paquete, pero si se utilizan muchos mas se creara un loop que consumira los recursos e incluso el ancho de banda.

KillWin / WinNuke

Enviando al puerto 139 (Netbios) datos especificando "Out of Band" se puede crear un DoS. Se consigue activando en la cabecera TCP el URGENT bit flag. De este modo el sistema operativo utiliza el URGENT POINTER para determinar donde terminan los datos urgentes. Cuando el puntero apunta al final de la trama y encuentra datos no esperados o no normales provoca el error del sistema.

Chargen y Chargen-Echo

Parecido al Fraggle, este se basa en enviar datagramas UDP mediante spoofing (con la direccion origen = direccion objetivo) al puerto 19 (chargen) de todos los hosts de una subred (con direccion broadcast) de forma que responderan inundando el objetivo con el generador de caracteres.

Combinando el puerto 19 (chargen) y el puerto 7 (echo) se puede conseguir un DoS si creamos un socket UDP al puerto 7 y enviamos un datagrama UDP al puerto 19. Esto provoca que el chargen genere caracteres que seran respondidos al puerto echo y que este a su vez devolvera al puerto 19 y asi sucesivamente.

Trinoo / Tribal Flood Network / Stacheldraht / ...

Es un DDoS que necesita de otros hosts. A partir de una lista de IPs uno actua de “master” iniciando un UDP flooding al objetivo con la ayuda de los demas hosts a través de un puerto. Se puede determinar viendo si existen conexiones TCP de tipo 6 en el puerto 27655 (normalmente utilizado).

El TFN y sus variantes (existen varias), es otra variante de Trinoo que necesita un master y hosts para realizar un flooding coordinado, pero a diferencia de Trinoo este puede realizarlo mediante UDP flood, TCP SYN flood, ICMP echo y ICMP broadcast. Además utiliza encriptacion “blowfish” para enviar su lista de IPs.

ANEXO 3: PUERTOS ASIGNADOS EN PROTOCOLOS TCP Y UDP. FICHERO /ETC/SERVICES

```
# Revision: 1.32.214.7 $ $Date: 97/09/10 14:50:42 $
#
# This file associates official service names and aliases with
# the port number and protocol the services use.
#
# Some of the services represented below are not supported on HP-UX.
# They are provided solely as a reference.
#
# The form for each entry is:
# <official service name>  <port number/protocol name>  <aliases>
#
# See the services(4) manual page for more information.
# Note: The entries cannot be preceded by a blank space.
#
tcpmux         1/tcp                # TCP port multiplexer (RFC 1078)
echo           7/tcp                # Echo
echo           7/udp                #
discard        9/tcp  sink null     # Discard
discard        9/udp  sink null     #
systat        11/tcp  users         # Active Users
daytime       13/tcp                # Daytime
daytime       13/udp                #
qotd          17/tcp  quote         # Quote of the Day
chargen       19/tcp  ttytst source  # Character Generator
chargen       19/udp  ttytst source  #
ftp-data      20/tcp                # File Transfer Protocol (Data)
ftp           21/tcp                # File Transfer Protocol (Control)
telnet        23/tcp                # Virtual Terminal Protocol
smtp          25/tcp                # Simple Mail Transfer Protocol
time          37/tcp  timeserver    # Time
time          37/udp  timeserver    #
rlp           39/udp  resource      # Resource Location Protocol
whois         43/tcp  nicname       # Who Is
domain        53/tcp  nameserver    # Domain Name Service
domain        53/udp  nameserver    #
bootps        67/udp                # Bootstrap Protocol Server
bootpc        68/udp                # Bootstrap Protocol Client
tftp          69/udp                # Trivial File Transfer Protocol
rje           77/tcp  netrjs        # private RJE Service
finger        79/tcp                # Finger
http          80/tcp  www           # World Wide Web HTTP
http          80/udp  www           # World Wide Web HTTP
link          87/tcp  ttylink       # private terminal link
supdup        95/tcp                #
hostnames    101/tcp  hostname      # NIC Host Name Server
tsap         102/tcp iso_tsap iso-tsap # ISO TSAP (part of ISODE)
pop          109/tcp postoffice pop2 # Post Office Protocol - Version 2
pop3         110/tcp  pop-3         # Post Office Protocol - Version 3
portmap      111/tcp  sunrpc        # SUN Remote Procedure Call
portmap      111/udp  sunrpc        #
ident        113/tcp  authentication # RFC1413
sftp         115/tcp                # Simple File Transfer Protocol
uucp-path    117/tcp                # UUCP Path Service
nntp         119/tcp  readnews untp  # Network News Transfer Protocol
ntp          123/udp                # Network Time Protocol
netbios_ns   137/tcp                # NetBIOS Name Service
netbios_ns   137/udp                #
netbios_dgm  138/tcp                # NetBIOS Datagram Service
netbios_dgm  138/udp                #
netbios_ssn  139/tcp                # NetBIOS Session Service
netbios_ssn  139/udp                #
bftp         152/tcp                # Background File Transfer Protocol
snmp         161/udp  snmpd         # Simple Network Management Protocol Agent
snmp-trap    162/udp  trapd         # Simple Network Management Protocol Traps
bgp          179/tcp                # Border Gateway Protocol
# PV performance tool services entries
pvserver     382/tcp    # PV server
pvalarm      383/tcp    # PV alarm management
#
# UNIX services
#
biff         512/udp  comsat         # mail notification
```

```
exec         512/tcp                 # remote execution, passwd required
login        513/tcp                 # remote login
who          513/udp  whod           # remote who and uptime
shell        514/tcp  cmd            # remote command, no passwd used
syslog       514/udp                 # remote system logging
printer      515/tcp  spooler        # remote print spooling
talk         517/udp                 # conversation
ntalk        518/udp                 # new talk, conversation
route        520/udp  router routed  # routing information protocol
efs          520/tcp                 # Extended file name server
timed        525/udp  timeserver     # remote clock synchronization
tempo        526/tcp  newdate        #
courier      530/tcp  rpc            #
conference   531/tcp  chat           #
netnews      532/tcp  readnews       #
netwall      533/udp                 # Emergency broadcasting
uucp         540/tcp  uucpd          # uucp daemon
remotefs     556/tcp  rfs_server rfs # Brunhoff remote filesystem
ingreslock  1524/tcp                 #
#
# Other HP-UX services
#
lansrm       570/udp                 # SRM/UX Server
DAServer     987/tcp                 # SQL distributed access
instl_boots 1067/udp                 # installation bootstrap protocol server
instl_bootc 1068/udp                 # installation bootstrap protocol client
nfsd-keepalive 1110/udp              # Client status info
nfsd-status    1110/tcp              # Cluster status info
msql        1111/tcp                 # Mini SQL database server
rlb         1260/tcp                 # remote loopback diagnostic
clvm-cfg    1476/tcp                 # HA LVM configuration
diagmond    1508/tcp                 # Diagnostic System Manager
nft         1536/tcp                 # NS network file transfer
sna-cs      1553/tcp                 # SNAplus client/server
sna-cs      1553/udp                 # SNAplus client/server
ncpm-pm     1591/udp                 # NCPM Policy Manager
ncpm-hip    1683/udp                 # NCPM Host Information Provider
cvmon       1686/udp                 # Clusterview cvmon-cvmap communication
registrar   1712/tcp                 # resource monitoring service
registrar   1712/udp                 # resource monitoring service
ncpm-ft     1744/udp                 # NCPM File Transfer
psmond      1788/tcp                 # Predictive Monitor
psmond      1788/udp                 # Hardware Predictive Monitor
pmlockd     1889/tcp                 # SynerVision locking daemon
pmlockd     1889/udp                 #
nfsd        2049/udp                 # NFS remote file system
netdist     2106/tcp                 # update(1m) network distribution service
rfa         4672/tcp                 # NS remote file access
veesm       4789/tcp                 # HP VEE service manager
hacl-hb     5300/tcp                 # High Availability (HA) Cluster heartbeat
hacl-gs     5301/tcp                 # HA Cluster General Services
hacl-cfg    5302/tcp                 # HA Cluster TCP configuration
hacl-cfg    5302/udp                 # HA Cluster UDP configuration
hacl-probe  5303/tcp                 # HA Cluster TCP probe
hacl-probe  5303/udp                 # HA Cluster UDP probe
hacl-local  5304/tcp                 # HA Cluster Commands
hacl-test   5305/tcp                 # HA Cluster Test
hacl-dlm    5408/tcp                 # HA Cluster distributed lock manager
lanmgrx.osB 5696/tcp                 # LAN Manager/X for B.00.00 OfficeShare
r4-sna-cs   5707/tcp                 # SNA client/server (up to Release 4.1)
SNAplus     5708/udp                 # SNA logical network A (up to Release 4.1)
r4-sna-ft   5709/tcp                 # SNA file transfer (up to Release 4.1)
hcserver    5710/tcp                 # HP Cooperative Services
grmd        5999/tcp                 # graphics resource manager
spc         6111/tcp                 # sub-process control
desmevt     6868/tcp                 # DE/ Services Monitor, Event Service
pdclientd   6874/tcp                 # Palladium print client daemon
pdeventd    6875/tcp                 # Palladium print event daemon
iasqlsvr    7489/tcp                 # Information Access
recserv     7815/tcp                 # SharedX Receiver Service
ftp-ftam    8868/tcp                 # FTP->FTAM Gateway
mcsemon     9999/tcp                 # MC/System Environment monitor
console     10000/tcp                # MC/System Environment console multiplexor
actcp       31766/tcp                # ACT Call Processing Server
#
# Kerberos (Project Athena/MIT) services
#
kerberos5    88/udp   kdc                 # Kerberos 5 kdc
klogin       543/tcp                  # Kerberos rlogin -kfall
kshell       544/tcp  krcmd            # Kerberos remote shell -kfall
ekshell      545/tcp  krcmd            # Kerberos encrypted remote shell -kfall
kerberos     750/udp  kdc              # Kerberos (server) udp -kfall
```

```
kerberos     750/tcp  kdc            # Kerberos (server) tcp -kfall
kerberos_master 751/tcp kadmin       # Kerberos kadmin
krbupdate    760/tcp  kreg           # Kerberos registration -kfall
kpasswd      761/tcp  kpwd           # Kerberos "passwd" -kfall
eklogin     2105/tcp                 # Kerberos encrypted rlogin -kfall
# The X10_LI server for each display listens on ports 5800 + display number.
# The X10_MI server for each display listens on ports 5900 + display number.
# The X11 server for each display listens on ports 6000 + display number.
# The X11 font server listens on port 7000.
# Do NOT associate other services with these ports.
# Refer to the X documentation for details.

hpoms-ci-lstn   5403/tcp      #SAP spooler support
hpoms-dps-lstn  5404/tcp      #SAP spooler support
samd            3275/tcp        # sam daemon

dtspc   6112/tcp       #subprocess control
```