Aplicacions segures

Xavier Perramon

1 crèdit
P03/05070/02095

Universitat Oberta de Catalunya

Índex

Introducció

En aquest mòdul es descriuen aplicacions que fan ús de tècniques de seguretat a les comunicacions per protegir les dades intercanviades.

Un exemple d'aplicacions a protegir són les que permeten establir sessions de treball interactives amb un servidor remot. En la primera part del mòdul veurem una aplicació d'aquest tipus, anomenada **SSH**, que defineix el seu propi protocol per tal que les dades de l'aplicació es transmetin xifrades. El mateix protocol proporciona també mecanismes d'autenticació del servidor davant l'usuari, i de l'usuari davant el servidor. Com veurem, el protocol SSH es pot utilitzar per a altres aplicacions a part de les sessions interactives, ja que permet l'encapsulació de connexions TCP a qualsevol port dins d'una connexió SSH.

La segona part d'aquest mòdul la dedicarem a una altra de les principals aplicacions que sol ser necessari protegir: el **correu electrònic**. Amb els mecanismes de protecció adients es pot garantir la **confidencialitat**, és a dir que ningú més que el destinatari o destinataris legítims puguin veure el contingut dels missatges, i l'**autenticitat**, és a dir que els destinataris puguin comprovar que ningú ha falsificat un missatge. Els sistemes actuals de correu electrònic típicament fan servir les **signatures digitals** per proporcionar el servei d'autenticació de missatge.

Una particularitat del correu electrònic segur respecte a altres aplicacions com SSH, és que la protecció es realitza preferentment sobre els missatges enviats, més que no pas sobre els protocols de comunicació utilitzats. Això és així perquè la confidencialitat i l'autenticitat s'han de garantir no només durant la transmissió dels missatges, sinó en qualsevol altre moment posterior, ja que el destinatari pot guardar els missatges que rep i tornar-los a llegir quan li convingui.

En aquest mòdul veurem dos dels principals sistemes utilitzats actualment per protegir els missatges de correu electrònic, com són **S/MIME** i **PGP**.

Objectius

Amb els materials associats a aquest mòdul didàctic assolireu els objectius següents:

1. Conèixer el mecanisme general de funcionament del protocol SSH i les principals aplicacions que en fan ús.

2. Comprendre les funcions de seguretat que poden proporcionar els sistemes de correu electrònic segur i els mecanismes per a assolir-les.

3. Identificar l'estàndard S/MIME com a aplicació de l'especificació MIME al correu segur, i conèixer l'ús que fa de l'estàndard PKCS #7 i els certificats digitals X.509.

4. Conèixer el mètode de representació d'informació segura en PGP i el model de confiança mútua del sistema PGP.

1. El protocol SSH

SSH és una aplicació dissenyada per substituir determinades eines d'accés remot usades tradicionalment en els sistemes Unix, com `rsh` (*Remote Shell*), `rlogin` (*Remote Login*) o `rcp` (*Remote Copy*), per noves versions amb serveis de seguretat.

El nom d'aquesta aplicació, SSH, és l'abreviatura de *Secure Shell*, que vindria a significar "versió segura del programa *Remote Shell*".

L'aplicació defineix un protocol propi per a la transmissió segura de les dades, el **protocol SSH**. Aquest protocol està directament per sobre de la capa de transport, concretament del transport TCP, i com veurem en aquest apartat proporciona serveis anàlegs als del protocol SSL/TLS. A part d'establir connexions segures, el protocol SSH també ofereix altres funcionalitats, com per exemple la redirecció de ports TCP, o la comunicació entre clients i servidors de finestres X, a través d'una connexió SSH.

L'autor de la primera implementació de l'SSH, Tatu Ylönen, de la Universitat de Tecnologia d'Hèlsinki, va publicar l'any 1995 l'especificació de la versió 1 del protocol. Des de llavors s'ha treballat en l'especificació d'una nova versió del protocol, la 2.0, actualment en fase d'esborrany a l'espera de la seva publicació oficial com a RFC. Encara que la funcionalitat que proporciona és bàsicament la mateixa, la nova versió incorpora moltes millores i és substancialment diferent de l'anterior. La versió antiga i la nova del protocol són habitualment anomenades SSH1 i SSH2, respectivament. En aquest apartat ens centrarem sobretot en el protocol SSH2.

1.1. Característiques del protocol SSH

SSH proporciona serveis de seguretat equivalents als del protocol SSL/TLS.

Confidencialitat. SSH serveix per comunicar dades, que típicament són l'entrada d'una aplicació remota i la sortida que genera, o bé la informació que es transmet per un port redirigit, i la confidencialitat d'aquestes dades es garanteix mitjançant el xifratge.

Com en el cas del protocol SSL/TLS, en SSH s'aplica un xifratge simètric a les dades, i per tant cal realitzar prèviament un intercanvi segur de claus entre

client i servidor. Una diferència respecte a SSL/TLS és que en SSH2 es poden utilitzar algorismes de xifratge diferents en els dos sentits de la comunicació.

Un servei addicional que proporciona SSH és la confidencialitat de la identitat de l'usuari. Mentre que en SSL 3.0 i TLS 1.0, si s'opta per autenticar el client, aquest ha d'enviar el seu certificat en clar, en SSH (i també en SSL 2.0) l'autenticació de l'usuari es realitza quan els paquets ja s'envien xifrats.

D'altra banda, SSH2 també permet ocultar certes característiques del tràfic, com per exemple la longitud real dels paquets.

Autenticació d'entitat. El protocol SSH proporciona mecanismes per autenticar tant l'ordinador servidor com l'usuari que s'hi vol connectar. L'autenticació del servidor sol realitzar-se conjuntament amb l'intercanvi de claus. En SSH2 el mètode d'intercanvi de claus és negociat per client i servidor, encara que actualment només n'hi ha definit un, basat en l'algorisme Diffie-Hellman.

Per autenticar l'usuari hi ha diferents mètodes; depenent de quin s'utilitzi, pot ser necessària també l'autenticació de l'ordinador client, mentre que altres mètodes permeten que l'usuari degudament autenticat accedeixi al servidor des de qualsevol ordinador client.

Autenticació de missatge. Igual que en SSL/TLS, en SSH2 l'autenticitat de les dades es garanteix afegint a cada paquet un codi MAC calculat amb una clau secreta. També hi ha la possibilitat d'utilitzar algorismes MAC diferents en cada sentit de la comunicació.

Igual que SSL/TLS, SSH també està dissenyat amb els següents criteris addicionals:

Eficiència. SSH contempla la compressió de les dades intercanviades per reduir la longitud dels paquets. SSH2 permet negociar l'algorisme a utilitzar en cada sentit de la comunicació, tot i que només n'hi ha un de definit a l'especificació del protocol. Aquest algorisme és compatible amb el que fan servir programes com `gzip` (RFC 1950–1952).

A diferència d'SSL/TLS, en SSH no hi ha prevista la reutilització de claus de sessions anteriors: a cada nova connexió es tornen a calcular les claus. Això és així perquè SSH està pensat per a connexions que tenen una durada més o menys llarga, com solen ser les sessions de treball interactives amb un ordinador remot, i no per a les connexions curtes però consecutives que són més típiques del protocol d'aplicació HTTP (que és el que inicialment es pretenia protegir amb SSL). De totes maneres, SSH2 defineix mecanismes per intentar escurçar el procés de negociació.

Extensibilitat. En SSH2 també es negocien els algorismes de xifratge, d'autenticació d'usuari, de MAC, de compressió, i d'intercanvi de claus. Cada algorisme s'identifica amb una cadena de caràcters que representa el seu nom. Els noms poden correspondre a algorismes oficialment registrats, o bé a algorismes proposats experimentalment o definits localment.

Adaptació d'SSH als idiomes locals

D'altra banda, SSH2 facilita l'adaptació de les implementacions als idiomes locals. Allà on el protocol preveu la transmissió d'un missatge d'error o informatiu que pugui ser mostrat a l'usuari humà, s'inclou una etiqueta que identifica l'idioma del missatge, d'acord amb RFC 1766.

Tant aquests missatges com els noms d'usuari es representen amb el joc de caràcters universal ISO/IEC 10646 mitjançant la codificació UTF-8, d'acord amb RFC 2279 (el codi ASCII n'és un subconjunt perquè en UTF-8 els caràcters amb codi menor que 128 es representen amb un sol byte, de valor igual al codi).

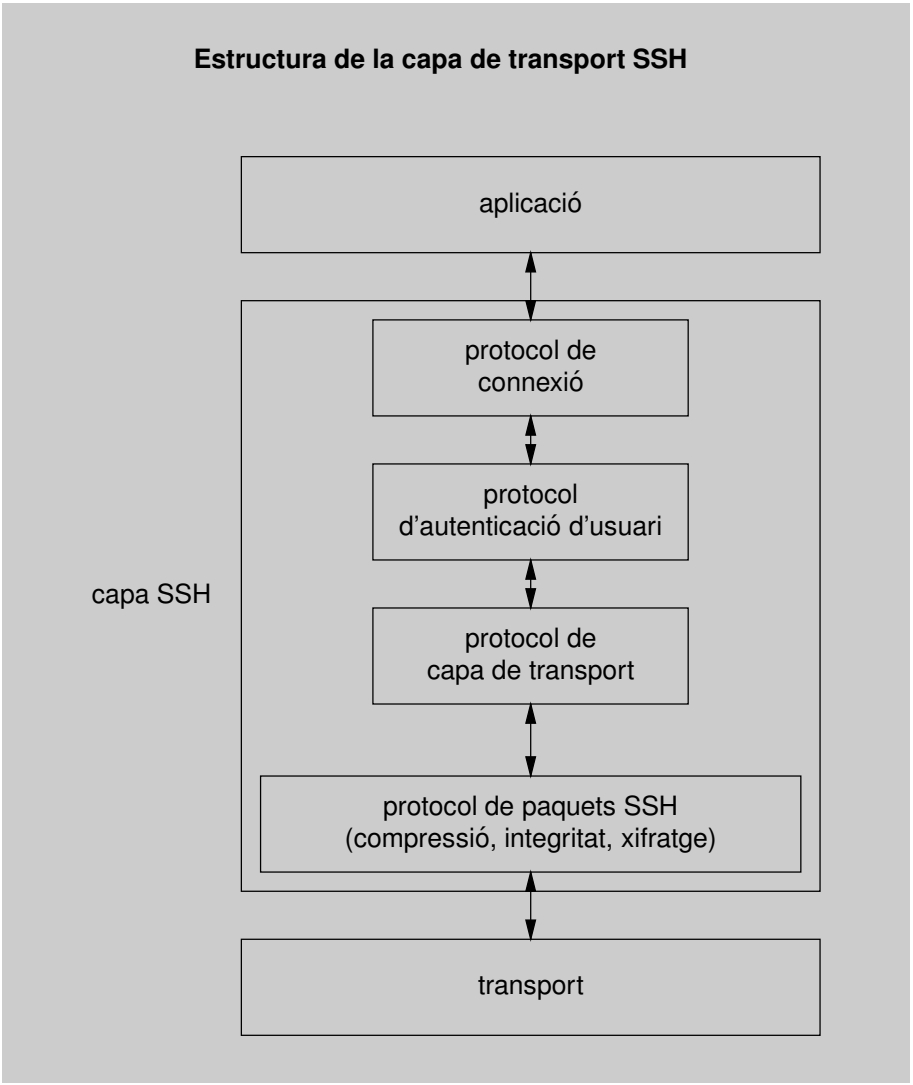
Algorismes no oficials

Els noms dels algorismes no oficials han de ser de la forma "*nom*@*domini*", on *domini* és un domini DNS controlat per l'organització que defineix l'algorisme (per exemple "xifratge-fort@uoc.edu").

1.2. La capa de transport SSH

De la mateixa manera que en SSL/TLS es defineixen dues sub-capes en el nivell de transport segur, en SSH també es pot considerar una divisió en dos sub-nivells. A més, en SSH2 el nivell superior està estructurat en 3 protocols, un per sobre de l'altre, com mostra la següent figura.

Estructura de la capa de transport SSH



Al nivell inferior de la capa SSH hi ha el **protocol de paquets SSH**. Els tres protocols que hi ha per sobre d'aquest són:

- El **protocol de capa de transport**, que s'encarrega de l'intercanvi de claus.
- El **protocol d'autenticació d'usuari**.
- El **protocol de gestió de les connexions**.

1.2.1. El protocol de paquets SSH

El protocol de paquets SSH s'encarrega de construir i intercanviar les unitats del protocol, que són els **paquets SSH**.

A l'hora d'enviar dades, als missatges dels nivells superiors se'ls aplica (per aquest ordre):

- La compressió.
- El codi d'autenticació MAC.
- El xifratge.

En recepció, a cada paquet se li aplica el processament invers (desxifratge, verificació d'autenticitat i descompressió).

El format dels paquets SSH2 és el següent:

Format dels paquets SSH2

Aquests són els camps que hi ha en un paquet SSH2:

- El primer és la longitud de la resta del paquet, exclòs el MAC (per tant, és igual a $1 + L_m + L_p$).
- El segon camp indica quants bytes de *padding* hi ha. Aquest nombre de bytes ha de ser tal que la longitud total del paquet, exclòs el MAC, sigui múltiple de 8 (o de la longitud de bloc en els xifratges de bloc, si és més gran que 8).

- El tercer camp és el contingut del missatge, comprimit si és el cas. El primer byte del contingut sempre indica de quin tipus de missatge es tracta, i l'estructura de la resta de bytes depèn del tipus.
- El quart camp són els bytes aleatoris de *padding*. Sempre hi són presents, fins i tot quan el xifratge utilitzat sigui de flux, i la seva longitud ha de ser almenys igual a 4. Per tant, la longitud mínima d'un paquet, sense comptar el MAC, és de 16 bytes.
- El cinquè camp és el codi d'autenticació MAC, obtingut mitjançant la tècnica HMAC a partir d'una clau secreta, un número de seqüència implícit de 32 bits, i el valor dels altres quatre camps del paquet. La longitud del MAC depèn de l'algorisme acordat, i pot ser 0 si s'utilitza l'algorisme nul.

Quan es xifren els paquets, s'aplica el xifratge a tots els camps tret del MAC, però incloent la longitud. Això vol dir que el receptor ha de desxifrar els 8 primers bytes de cada paquet per conèixer la longitud total de la part xifrada.

Bytes de *padding*

Els bytes de *padding* asseguren que la longitud de les dades a xifrar sigui apropiada per als xifrats de bloc.

1.2.2. El protocol de capa de transport SSH

El protocol de capa de transport s'encarrega de l'establiment de la connexió de transport, l'autenticació de servidor i intercanvi de claus, i les peticions de servei dels altres protocols.

El client es connecta al servidor mitjançant el protocol TCP. El servidor ha d'estar escoltant peticions de connexió en el port assignat al servei SSH, que és el 22.

El primer pas un cop establerta la connexió és negociar la versió del protocol SSH que es farà servir. Tant el client com el servidor envien una línia que conté el text "`SSH-`*`x.y-implementació`*", on *`x.y`* és el número de versió del protocol (per exemple, `2.0`) i *`implementació`* és una cadena identificativa del programari del client o servidor. Si els números de versió no concorden, el servidor decideix si pot continuar o no: si no pot, simplement tanca la connexió. Abans d'aquesta línia de text, el servidor també pot enviar-ne d'altres amb missatges informatius, mentre no comencin amb "`SSH-`".

Compatibilitat amb la versió 1

En SSH2 es defineix un mode de compatibilitat amb SSH1, en el qual el servidor identifica la seva versió amb el número 1.99: els clients SSH2 han de considerar aquest número equivalent a 2.0, mentre que els clients SSH1 respondran amb el seu número de versió real.

Quan s'han posat d'acord en la versió, client i servidor passen a intercanviar missatges amb el protocol de paquets SSH vist abans, inicialment sense xifrar i sense MAC. Per estalviar temps, el primer paquet SSH es pot enviar juntament amb la línia que indica la versió, sense esperar rebre la línia de l'altra part. Si les versions coincideixen, el protocol continua normalment; si no, pot ser necessari reiniciar-lo.

Primer de tot, es procedeix a l'intercanvi de claus. En SSH2 cada part envia

un missatge KEXINIT que conté una cadena de 16 bytes aleatoris anomenada *"cookie"*, i les llistes d'algorismes suportats per ordre de preferència: algorismes d'intercanvi de claus i, per a cada sentit de la comunicació, algorismes de xifratge simètric, de MAC i de compressió. També s'inclou una llista d'idiomes suportats per als missatges informatius. Per a cada tipus d'algorisme, s'escull el primer de la llista del client que estigui també a la llista del servidor.

Algorismes previstos en SSH2

Els algorismes criptogràfics que contempla SSH2 són els següents:

- Per a l'intercanvi de claus: Diffie-Hellman.
- Per al xifratge: RC4, Triple DES, Blowfish, Twofish, IDEA i CAST-128.
- Per al MAC: HMAC-SHA1 i HMAC-MD5 (amb tots els bytes o només amb els 12 primers).

Els paquets que vénen a continuació són els d'intercanvi de claus, i depenen de l'algorisme escollit (encara que SSH2 només preveu l'algorisme de Diffie-Hellman).

Es pot suposar que la majoria d'implementacions tindran un mateix algorisme preferit de cada tipus. Així, per reduir el temps de resposta es pot enviar el primer missatge d'intercanvi de claus després del KEXINIT sense esperar el de l'altra part, fent servir aquests algorismes preferits. Si la suposició resulta encertada l'intercanvi de claus continua normalment, i si no, els paquets enviats anticipadament són ignorats i es tornen a enviar amb els algorismes correctes.

Sigui quin sigui l'algorisme, com a resultat de l'intercanvi de claus s'obté un secret compartit i un identificador de sessió. Amb l'algorisme Diffie-Hellman, aquest identificador és el *hash* d'una cadena formada, entre d'altres, per les *cookies* del client i el servidor. Les claus de xifrat i de MAC i els vectors d'inicialització es calculen aplicant funcions *hash* de diverses maneres a diferents combinacions del secret compartit i l'identificador de sessió.

Per finalitzar l'intercanvi de claus cada part envia un missatge NEWKEYS, que indica que el següent paquet serà el primer que farà servir els nous algorismes i claus.

Tot aquest procés es pot repetir quan sigui convenient per regenerar les claus. L'especificació SSH2 recomana fer-ho després de cada gigabyte transferit o de cada hora de temps de connexió.

Si es produeix algun error en l'intercanvi de claus, o en qualsevol altre punt del protocol, es genera un missatge DISCONNECT, que pot contenir un text explicatiu de l'error, i es tanca la connexió.

Altres missatges que es poden intercanviar en qualsevol moment són:

- IGNORE: el seu contingut ha de ser ignorat, però pot servir per contrarestar anàlisis de flux de tràfic.
- DEBUG: serveixen per enviar missatges informatius.
- UNIMPLEMENTED: s'envien en resposta a missatges de tipus desconegut.

En SSH2, després de finalitzat l'intercanvi de claus el client envia un missatge SERVICE_REQUEST per sol·licitar un servei, que pot ser autenticació d'usuari, o bé accés directe al protocol de connexió si no cal aquesta autenticació. El servidor respon amb SERVICE_ACCEPT si permet l'accés al servei sol·licitat, o amb DISCONNECT en cas contrari.

1.2.3. El protocol d'autenticació d'usuari

En SSH es contemplen diferents mètodes d'autenticació d'usuari:

1) **Autenticació nul·la.** El servidor permet que l'usuari accedeixi directament, sense cap comprovació, al servei sol·licitat. Un exemple seria l'accés a un servei anònim.
2) **Autenticació basada en llistes d'accés.** A partir de l'adreça del sistema client i el nom de l'usuari d'aquest sistema que sol·licita l'accés, el servidor consulta una llista per determinar si l'usuari està autoritzat a accedir-hi. Aquesta és la mateixa autenticació que fa servir el programa rsh de Unix, en el qual el servidor consulta els fitxers .rhosts i /etc/hosts.equiv. Donada la seva vulnerabilitat a atacs de falsificació d'adreça IP, aquest mètode només es pot utilitzar en SSH1: en SSH2 no està suportat.
3) **Autenticació basada en llistes d'accés amb autenticació de client.** Igual que l'anterior, però el servidor verifica que el sistema client és efectivament qui diu ser, per evitar els atacs de falsificació d'adreça.
4) **Autenticació basada en contrasenya.** El servidor permet l'accés si l'usuari dóna una contrasenya correcta. Aquest és el mètode que segueix normalment el procés login en els sistemes Unix.
5) **Autenticació basada en clau pública.** En lloc de donar una contrasenya, l'usuari s'autentica demostrant que posseeix la clau privada corresponent a una clau pública reconeguda pel servidor.

En SSH2 el client va enviant missatges USERAUTH_REQUEST, que inclouen el nom d'usuari (es pot anar canviant d'un missatge a un altre), el mètode d'autenticació sol·licitat, i el servei al qual es vol accedir. Si el servidor permet l'accés respondrà amb un missatge USERAUTH_SUCCESS; si no, enviarà un missatge USERAUTH_FAILURE, que conté la llista de mètodes d'autenticació que es poden continuar intentant, o bé tancarà la connexió si ja hi ha hagut massa intents o ha passat massa temps. El servidor pot enviar opcionalment

un missatge informatiu USERAUTH_BANNER abans de l'autenticació.

Missatge USERAUTH_BANNER

El missatge USERAUTH_BANNER pot incloure, per exemple, text identificatiu del sistema servidor, avisos sobre restriccions d'ús, etc. Aquesta és la mena d'informació que típicament mostren els sistemes Unix abans del *prompt* per introduir el nom d'usuari, i que sol estar al fitxer /etc/issue.

Els missatges de sol·licitud d'autenticació contenen la següent informació segons el mètode:

1) Per a l'autenticació nul·la no es necessita cap informació addicional.

2) En l'autenticació basada en llistes d'accés (només aplicable a SSH1) cal donar el nom de l'usuari en el sistema client. L'adreça d'aquest sistema se suposa disponible a través dels protocols subjacents (TCP/IP).

3) Quan s'utilitzen llistes d'accés amb autenticació de client, en SSH2 el client envia el seu nom DNS complet, el nom de l'usuari local, la clau pública del sistema client (i certificats, si en té), la signatura d'una cadena de bytes que inclou l'identificador de sessió, i l'algorisme amb què s'ha generat aquesta signatura. El servidor ha de validar la clau pública i la signatura per completar l'autenticació.

4) En l'autenticació amb contrasenya només cal enviar directament la contrasenya. Per motius obvis, aquest mètode no hauria d'estar permès si el protocol de la sub-capa de transport SSH està utilitzant l'algorisme de xifratge nul.

Contrasenya caducada

SSH2 preveu el cas que la contrasenya de l'usuari en el sistema servidor estigui caducada i calgui canviar-la abans de continuar. No s'hauria de permetre el canvi, però, si no s'està utilitzant cap MAC (algorisme nul), perquè un atacant podria modificar el missatge que conté la nova contrasenya.

5) L'autenticació basada en clau pública és semblant a la de llistes d'accés amb autenticació de client. En SSH2 el client ha d'enviar un missatge que contingui la clau pública de l'usuari (i els certificats, si en té), l'algorisme que correspon a aquesta clau, i una signatura en la qual intervé l'identificador de sessió. El servidor donarà per bona l'autenticació si verifica correctament la clau i la signatura.

   Opcionalment, per tal d'evitar càlculs i interaccions innecessàries amb l'usuari, el client pot enviar abans un missatge amb la mateixa informació però sense la signatura, perquè el servidor respongui si la clau pública que se li ofereix és acceptable.

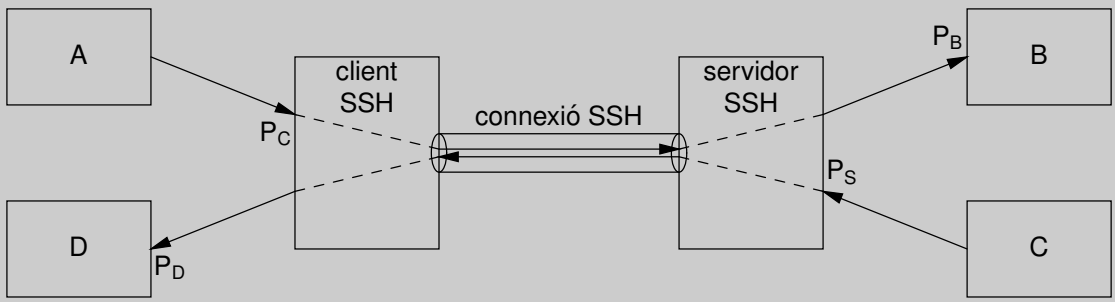
Quan el procés d'autenticació s'hagi completat satisfactòriament, en SSH2 es passa al servei que el client hagi sol·licitat en el seu últim missatge USERAUTH_REQUEST (el que ha donat lloc a l'autenticació correcta). Actualment, però, només hi ha un servei definit, que és el de connexió.

1.2.4. El protocol de connexió

El protocol de connexió gestiona les sessions interactives per a l'execució remota de comandes, enviant les dades d'entrada de client a servidor i les de sortida en sentit invers. També s'encarrega de la redirecció de ports TCP.

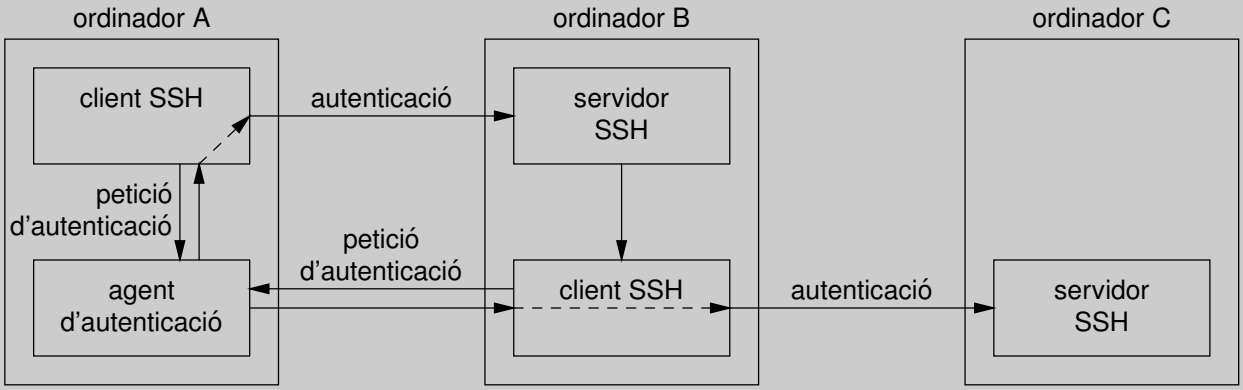
Tal com mostra la següent figura, amb la redirecció TCP és possible fer que les connexions que es realitzin a un determinat port P_C del client siguin redirigides a un port P_B d'un ordinador B des del servidor, o que les connexions que es realitzin a un determinat port P_S del servidor siguin redirigides a un port P_D d'un ordinador D des del client. D'aquesta manera la connexió SSH es pot utilitzar, per exemple, com a túnel d'altres connexions a través d'un tallafocs que estigui situat entre el client i el servidor SSH.

Redirecció de ports TCP amb SSH



A més, SSH contempla la possibilitat d'utilitzar el que s'anomena **agent d'autenticació**. Aquest agent és un procés que permet automatitzar l'autenticació de l'usuari basada en claus públiques quan cal realitzar-la des d'un ordinador remot. Per exemple, suposem la situació de la següent figura:

Ús de l'agent d'autenticació SSH



L'usuari de l'ordinador A utilitza un client SSH per connectar-se a l'ordinador B i treballar-hi amb una sessió interactiva. L'ordinador A pot ser, per exemple, un PC portàtil on l'usuari té guardada la seva clau privada i d'on no desitja que aquesta clau hagi de sortir mai. Llavors resulta que necessita establir una connexió SSH (per exemple, una altra sessió interactiva) des de l'ordinador B a l'ordinador C, i s'ha d'autenticar amb la seva clau personal. El client de l'ordinador B, en lloc de realitzar directament l'autenticació, per a la qual cosa necessitaria la clau privada de l'usuari, demana a l'agent de

l'ordinador A que signi el missatge adient per demostrar que posseeix la clau privada. Aquest esquema també es pot fer servir localment per part dels clients del mateix ordinador A.

Cada sessió interactiva, connexió TCP redirigida o connexió a un agent és un **canal**. Hi pot haver diversos canals oberts en una mateixa connexió SSH, cadascun identificat amb un número a cada extrem (els números assignats a un canal en el client i en el servidor poden ser diferents).

SSH2 preveu 14 tipus de missatges que es poden enviar durant la fase de connexió. Aquestes són algunes de les funcions que permeten realitzar els missatges:

Obrir un canal. Es poden obrir canals de diferents tipus: sessió interactiva, canal de finestres X, connexió TCP redirigida, o connexió amb un agent d'autenticació.

Configurar paràmetres del canal. Abans de començar una sessió interactiva el client pot especificar si necessita que se li assigni un pseudo-terminal en el servidor, com fa el programa `rlogin` de Unix (en canvi, el programa `rsh` no el necessita), i si és així, amb quins paràmetres (tipus de terminal, dimensions, caràcters de control, etc.). Hi ha altres missatges per indicar si es vol connexió amb l'agent d'autenticació o redirecció de connexions de finestres X.

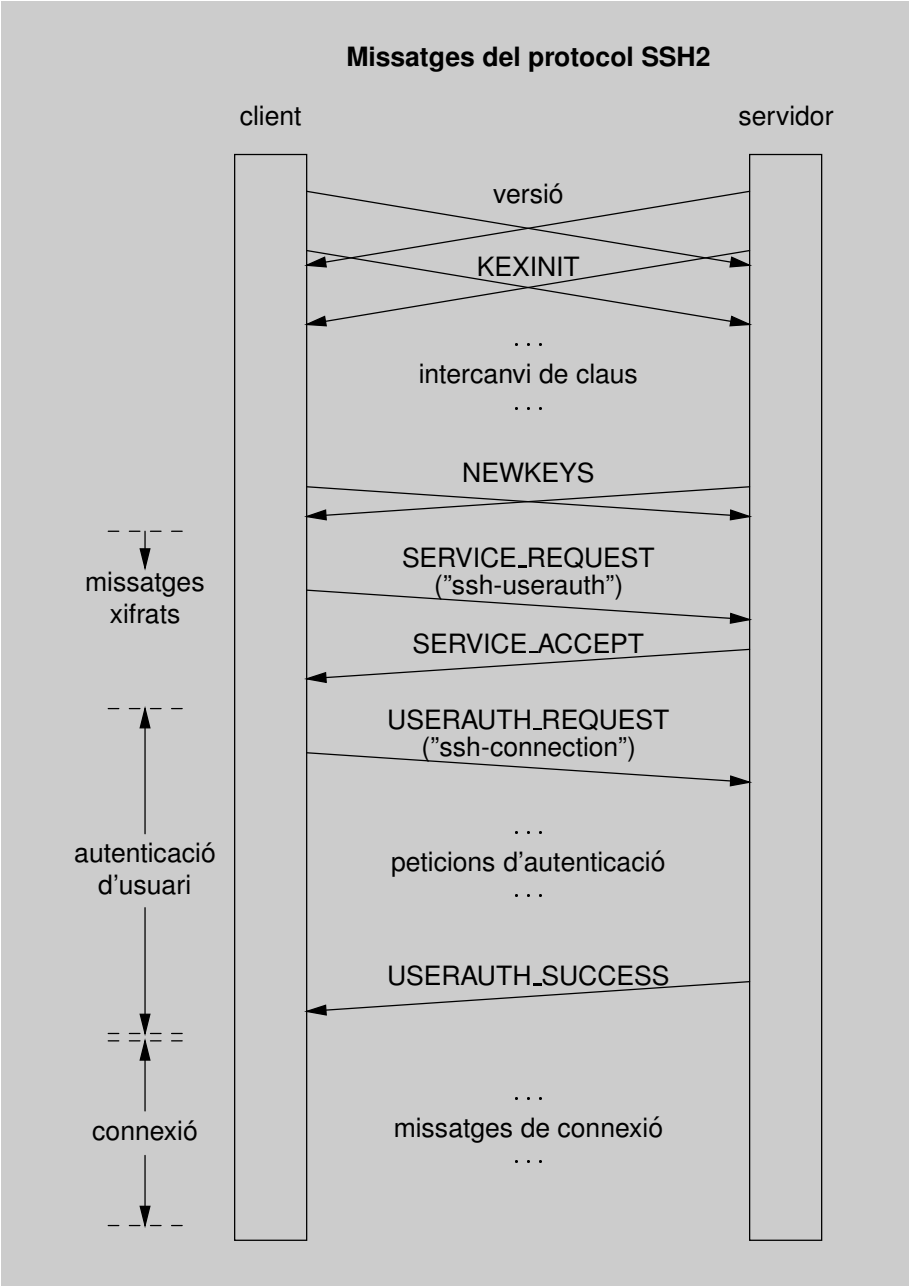
Començar sessió interactiva. Un cop configurats els paràmetres necessaris, el client pot donar el nom d'una comanda a executar en el servidor (com en `rsh`), o bé indicar que vol executar un intèrpret de comandes (com en `rlogin`). A més d'un procés remot, en SSH2 també hi ha la possibilitat d'iniciar un "subsistema", com pot ser, per exemple, una transferència de fitxers.

Enviar dades. En SSH2 hi ha dos tipus de missatge per aquest fi: un per enviar dades normals en qualsevol sentit i per qualsevol canal (incloent les sessions interactives), i un altre per enviar dades especials (per exemple, les de la sortida d'error `stderr`). A més de les dades de la sessió, el client també pot enviar un missatge per indicar que ha rebut un senyal o que s'ha produït un canvi en les dimensions del terminal.

Tancar canal. Quan acaba l'execució normal del procés o intèrpret de comandes, el servidor envia un missatge indicant el codi de sortida (el valor numèric que retorna el procés). Si ha acabat a causa d'un senyal, en SSH2 envia un missatge amb el número de senyal. Hi ha altres missatges que serveixen per indicar que ja no hi ha més dades d'entrada, per sol·licitar el tancament d'un canal des d'un extrem, i per confirmar el tancament des de l'altre extrem.

Altres operacions (no associades a un canal obert). El client pot demanar que les connexions que arribin a un determinat port TCP del servidor li siguin redirigides, per poder-les reenviar a una altra adreça.

La següent figura resumeix l'intercanvi de missatges en SSH2.



1.3. Atacs contra el protocol SSH

Moltes de les consideracions sobre la protecció que proporciona SSL/TLS són aplicables també al protocol SSH. Aquest protocol està dissenyat perquè un atacant no pugui llegir el contingut dels missatges ni alterar-lo, i tampoc canviar-ne la seqüència.

Millores en SSH2

El protocol SSH1 era vulnerable a atacs de repetició, eliminació o reordenació de paquets perquè no utilitzava números de seqüència, i també al reenviament de paquets en sentit contrari si s'utilitzava una sola clau de xifratge per als dos sentits. Aquests problemes ja no són presents en SSH2.

La confidencialitat queda garantida amb el mètode d'intercanvi de claus basat en criptografia de clau pública, que protegeix contra els atacs "d'home al mig" que vam veure a l'apartat sobre SSL/TLS. A més, aquest mètode permet que el client s'asseguri que s'està connectant al servidor autèntic. Per comprovar

que la clau pública que envia el servidor és realment la seva, es poden fer servir certificats, o bé una base de dades local del client on hi hagi guardades les claus dels servidors reconeguts. I per autenticar l'usuari mitjançant clau pública (la seva o la del client des d'on es connecta, depenent del mètode d'autenticació), també hi ha les dues opcions: certificats o una base de dades de claus en el servidor.

Si no es fan servir certificats, el protocol contempla la possibilitat (encara que no és recomanada) de donar per bona la clau pública d'un servidor la primera vegada que s'hi estableixi una connexió, sense necessitat de cap comunicació prèvia. Això no és apropiat en un entorn on la seguretat sigui crítica perquè representa una vulnerabilitat a atacs "d'home al mig". En altres entorns, però, i mentre no hi hagi una infraestructura de claus àmpliament estesa, acceptar directament claus rebudes per primera vegada pot suposar un equilibri entre comoditat d'ús i seguretat.

Una característica interessant afegida en SSH2 és que les longituds dels paquets s'envien xifrades. Un atacant que vegi les dades intercanviades com un flux de bytes no pot saber on comença i acaba cada paquet SSH2 (si té accés al nivell de paquets TCP pot intentar fer deduccions, però sense total certesa). Això, juntament amb la possibilitat d'incloure *padding* arbitrari (fins a 255 bytes) i enviar missatges `IGNORE`, pot servir per ocultar les característiques del tràfic i dificultar els atacs amb text clar conegut.

D'altra banda, val la pena remarcar que els mètodes d'autenticació d'usuari mitjançant llistes d'accés es basen en la confiança del servidor en l'administrador del sistema client (de la mateixa manera que els protocols `rsh` i `rlogin`):

- Quan no s'autentica el sistema client (possibilitat contemplada solament en SSH1), el servidor només ha d'acceptar connexions que provinguin d'un port TCP privilegiat (menor que 1024) perquè a un usuari qualsevol no li sigui fàcil enviar paquets suplantant la identitat d'un altre.
- Quan hi ha autenticació del sistema client, el servidor confia que els usuaris no tindran accés a la clau privada d'aquest sistema, perquè si no podrien fer-la servir per generar missatges d'autenticació amb la identitat d'un altre usuari.

Finalment, tal com passa amb SSL/TLS, el protocol SSH està dissenyat per oferir determinades proteccions, però el nivell de seguretat que proporcioni en cada cas vindrà donat per les implementacions i l'ús que se'n faci. Es recomana que es puguin deshabilitar o restringir les característiques (mètodes d'autenticació d'usuari, redirecció de ports TCP, etc.) que en un determinat entorn impliquin alguna vulnerabilitat o possibilitat d'abús.

1.4. Aplicacions que fan ús del protocol SSH

Com que l'objectiu principal d'SSH és permetre l'execució remota de processos a l'estil dels programes `rsh` i `rlogin`, es poden implementar, i de fet s'han implementat, altres programes que es diguin per exemple `ssh` i `slogin` que facin el mateix però utilitzant el protocol SSH.

Els arguments poden ser els mateixos: el nom del servidor, "`-l` *usuari*" per especificar el nom d'usuari en el servidor, etc. Els programes també poden estar configurats per utilitzar diferents mètodes d'autenticació: el basat en els fitxers `.rhosts` i `/etc/hosts.equiv` funciona com en `rsh`/`rlogin`, i el basat en contrasenya funciona com en `rlogin`. Si s'utilitza autenticació del sistema client caldrà guardar la seva clau privada en algun lloc d'accés restringit. I si l'autenticació d'usuari està basada en la seva clau pública, la clau privada corresponent s'hauria de guardar protegida, típicament xifrada amb una clau simètrica derivada d'una contrasenya o d'una *passphrase*.

La implementació original del programa `ssh`, en les seves diferents versions, admet arguments de la forma "`-L` *p1*`:`*adr*`:`*p2*" i "`-R` *p1*`:`*adr*`:`*p2*" per especificar redireccions TCP del port local (del client) o remot (del servidor) *p1*, respectivament, al port *p2* de l'ordinador *adr*.

Exemples de redirecció de ports TCP amb SSH

1) Des d'un ordinador anomenat `prop` podem fer:

```
ssh -L 5555:srv.lluny.com:23 -l admin mig
```

Si ens autentiquem correctament, haurem establert una connexió interactiva amb l'ordinador `mig` com a usuari `admin`, i a més, qualsevol connexió al port 5555 de `prop`, com aquesta:

```
telnet prop 5555
```

serà redirigida al port 23 (TELNET) de l'ordinador `srv.lluny.com` (passant per `mig`, i amb el tram entre `prop` i `mig` protegit amb SSH).

2) Aquesta seria una manera de protegir una connexió a un servidor HTTP mitjançant un túnel SSH, suposant que puguem autenticar-nos davant d'aquest servidor:

```
ssh -L 5678:localhost:80 www.lluny.com
```

Un cop hem fet això, podem introduir l'adreça `http://localhost:5678/` en qualsevol navegador web de l'ordinador local, i ens portarà automàticament a l'adreça `http://www.lluny.com/` amb una connexió xifrada (i si en aquesta adreça hi ha una pàgina HTML amb referències no absolutes a altres pàgines del mateix servidor, aquestes altres pàgines també ens arribaran a través d'SSH).

2. Correu electrònic segur

El correu electrònic és una de les aplicacions més utilitzades en les xarxes de computadors. En el cas d'Internet, el protocol sobre el qual es fonamenta la transferència de missatges, l'SMTP, va ser publicat a l'estàndard RFC 821 l'any 1982. Com el seu nom indica, la principal característica d'aquest protocol és la seva simplicitat. Això ha permès que l'SMTP, juntament amb l'estàndard per al format dels missatges (RFC 822) i l'especificació MIME, sigui la base tecnològica de la gran majoria dels sistemes de correu actuals.

SMTP

SMTP és la sigla de *Simple Mail Transfer Protocol*.

MIME

MIME és la sigla de *Multipurpose Internet Mail Extensions*.

Aquesta gran virtut de l'SMTP, la simplicitat, és alhora una font de molts problemes de seguretat, ja que a un atacant pot resultar-li sorprenentment fàcil capturar missatges o enviar-ne de falsos en nom d'altres. En aquest apartat veurem algunes tècniques per afegir seguretat al servei de correu electrònic.

Si considerem el correu electrònic com un protocol de la capa d'aplicació, una possibilitat per protegir els missatges de correu seria utilitzar la seguretat que poden oferir les capes inferiors, com la de xarxa o la de transport. Per exemple, amb el protocol SMTP es pot negociar l'ús del transport segur SSL/TLS, mitjançant una comanda especial anomenada `STARTTLS` (RFC 2487).

En la transferència dels missatges, però, poden intervenir diversos agents intermedis, i per fer-la segura d'extrem a extrem caldria protegir tots els enllaços o intentar fer una connexió directa entre el sistema de l'originador i els dels destinataris. D'altra banda, la naturalesa *store-and-forward* (emmagatzematge i reenviament) del servei de correu electrònic fa que els missatges siguin vulnerables no solament quan es transfereixen d'un node intermedi a un altre, sinó també mentre estan emmagatzemats en aquests nodes. Això inclou el sistema de destinació final: un cop el missatge ha arribat a la bústia de l'usuari, el seu contingut pot ser inspeccionat o modificat per tercers abans que el llegeixi el destinatari legítim, o fins i tot després que ja l'hagi llegit.

És per això que s'han desenvolupat mètodes per protegir el correu electrònic en el mateix nivell d'aplicació, independentment del sistema de transport utilitzat. La idea és aplicar les funcions criptogràfiques necessàries al missatge abans de lliurar-lo als agents de transferència del servei de correu, i aquests només han de fer-lo arribar a la seva destinació de la manera habitual.

Així, d'una banda es pot aprofitar la infraestructura de correu electrònic ja existent, sense necessitat de canviar els servidors, etc., i d'altra banda la protecció és efectiva durant tot el procés, fins i tot mentre el missatge estigui emmagatzemat en la bústia del destinatari.

La majoria dels sistemes de correu electrònic segur que s'han proposat segueixen aquest model d'incorporar la seguretat dins dels propis missatges sense modificar el protocol de transferència. Alguns d'aquests sistemes són:

- **PEM** (*Privacy Enhanced Mail*)

  Va ser un dels primers sistemes de correu segur que es van desenvolupar: la primera versió es va publicar en l'especificació RFC 989. Estava basat directament en l'estàndard RFC 822, i només contemplava l'enviament de missatges de text ASCII. Actualment està en desús, però algunes de les tècniques que feia servir es continuen utilitzant actualment en els sistemes més moderns.

- **MOSS** (*MIME Object Security Services*)

  Va ser la primera especificació que va utilitzar el format MIME per representar la informació relacionada amb la seguretat. Estava basada en el sistema PEM, i es va publicar en el document RFC 1848.

- **PGP** (*Pretty Good Privacy*)

  Un dels sistemes més populars per afegir confidencialitat i autenticació, no solament al correu electrònic sinó a qualsevol tipus de dades. En molts entorns és l'estàndard *de facto* per a l'intercanvi segur d'informació. Ha anat evolucionant i actualment hi ha disponibles diverses versions, que inclouen variants com PGP/MIME, OpenPGP i GnuPG.

- **S/MIME** (*Secure MIME*)

  Un altre sistema que utilitza la tecnologia MIME, en aquest cas basat en la sintaxi definida per l'estàndard PKCS #7. Hi ha també una gran varietat d'implementacions disponibles.

En aquest apartat analitzarem alguns detalls dels sistemes S/MIME i PGP, però abans veurem les característiques generals del correu electrònic segur.

2.1. Seguretat en el correu electrònic

Quan s'afegeixen serveis de seguretat al correu electrònic cal tenir en compte les següents consideracions.

- En el correu electrònic la tramesa de missatges es duu a terme de manera no interactiva, i per tant no hi pot haver negociació d'algorismes o intercanvi

de claus quan s'envia un missatge. Això vol dir que possiblement caldrà un pas addicional per obtenir la clau necessària per comunicar-se amb un determinat corresponsal (consultar una base de dades de claus públiques, demanar a l'usuari que enviï la seva clau, etc.). Tots els sistemes de correu segur han de preveure algun mecanisme de **distribució de claus**.

- Una funcionalitat bàsica del correu electrònic és l'enviament d'un mateix missatge a **múltiples destinataris**. Amb el correu segur, si cal fer ús de paràmetres criptogràfics diferents per a cada destinatari, una solució poc eficient seria fer enviaments separats. Però si volem aprofitar les capacitats dels sistemes de correu existents, hem d'utilitzar una tècnica que permeti combinar en un sol missatge tota la informació necessària perquè pugui ser processat per cadascun dels destinataris.

Els serveis que proporcionen els sistemes de correu electrònic segur són bàsicament dos:

Confidencialitat. Mitjançant les tècniques de xifratge, es pot garantir que un missatge només podrà ser llegit pels seus destinataris legítims.

Autenticació de missatge. Els missatges poden incloure un codi d'autenticació (un codi MAC o una signatura digital) perquè els destinataris puguin verificar que han estat generats per l'originador autèntic, i ningú els ha modificat o falsificat.

Cadascun d'aquests dos serveis pot estar basat en tècniques criptogràfiques de clau simètrica o de clau pública.

Altres serveis

Hi ha altres serveis que no tots els sistemes de correu segur poden oferir: confidencialitat del flux de tràfic (que no se sàpiga qui envia missatges a qui i quan, i com de llargs), protecció contra negació de recepció (que un usuari no pugui dir que no ha rebut un missatge, o que un tercer no pugui esborrar missatges perquè el destinatari no els llegeixi) o contra atacs de repetició de missatges, etc.

2.1.1. Confidencialitat

Per tal que un originador A pugui enviar un missatge xifrat a un destinatari B, cal que tots dos hagin acordat l'ús d'una determinada **clau d'intercanvi** k_{AB}. Això es pot fer amb una comunicació segura "fora de línia" (per exemple cara a cara), o bé amb un mecanisme de distribució de claus.

La clau d'intercanvi k_{AB} pot ser una clau simètrica o una clau pública. Si és simètrica, es pot fer servir la mateixa en tots dos sentits de la comunicació, de A a B i de B a A ($k_{AB} = k_{BA}$). L'ús de claus d'intercanvi simètriques estava contemplat en l'estàndard PEM, però avui no és gaire habitual.

La situació més normal és que la clau d'intercanvi sigui una clau pública, i llavors les claus corresponents a un destinatari B són totes la mateixa: $k_{1B} = k_{2B} = k_{3B} = \ldots = k_{\mathrm{pub}_B}$.

Per xifrar un missatge de correu es fa servir sempre un algorisme de xifratge de clau simètrica, donat que els de clau pública són molt més costosos, especialment si el missatge és llarg.

El mètode que es fa servir doncs per enviar un missatge xifrat és el de l'anomenat **sobre digital**, que consisteix en:

1) Generar aleatòriament una clau de xifratge simètrica k_S, diferent per a cada missatge. Aquesta clau k_S s'anomena **clau de xifratge de contingut** o bé, per analogia amb els protocols de transport, **clau de sessió**.

2) Xifrar el missatge M amb aquesta clau simètrica i obtenir $C = E(k_S, M)$.

3) Per a cada destinatari B_i del missatge, xifrar la clau de sessió amb la clau pública d'aquest destinatari i obtenir $K_{B_i} = E(k_{\text{pub}_{B_i}}, k_S)$.

4) Construir un nou missatge afegint al missatge xifrat C totes les claus xifrades K_{B_i}.

5) Enviar als destinataris aquest missatge (el mateix missatge per a tots).

Sobre digital

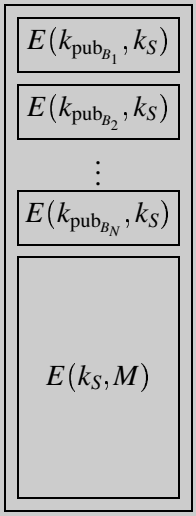
El nom de "sobre digital" prové de l'analogia amb el correu tradicional: un missatge de correu electrònic sense xifrar és com una postal, de la qual tothom en pot llegir el contingut, mentre que un missatge xifrat d'aquesta manera és com una carta amb sobre, que només pot obrir la persona que hi figura com a destinatària.

Missatges a un únic destinatari

La tècnica del sobre digital s'utilitza per als missatges adreçats a qualsevol nombre de destinataris, que també pot ser només un.

Per tant, si un missatge confidencial s'ha de trametre a N destinataris, no cal enviar N còpies del missatge xifrades amb la clau de cadascun d'ells. Es pot fer servir la mateixa còpia per a tots, amb la qual cosa és possible aprofitar els mecanismes ja existents per enviar un missatge a múltiples destinataris.

Missatge amb sobre digital



En recepció, cada destinatari B_i haurà de seleccionar la K_{B_i} corresponent a la seva clau pública, desxifrar-la amb la seva clau privada $k_{\text{priv}_{B_i}}$ per obtenir la clau de sessió k_S, i finalment desxifrar C amb aquesta clau de sessió per recuperar el missatge M.

Llistes de distribució de correu

Les llistes de distribució de correu són un cas especial, perquè l'originador d'un missatge pot no saber a quins destinataris arribarà. Una possibilitat és utilitzar una única clau d'intercanvi simètrica, coneguda per tots els membres de la llista (amb el problema que això comporta de no garantir l'autenticitat). Una altra possibilitat és que l'agent que expandeix la llista rebi els missatges xifrats amb una clau pròpia de la llista, i els reenviï xifrats amb les claus de cada destinatari.

2.1.2. Autenticació de missatge

Per a l'autenticació dels missatges també es poden fer servir tècniques simètriques o de clau pública.

Les tècniques simètriques consisteixen en afegir al missatge un codi MAC, calculat amb una clau secreta compartida amb el destinatari. Això vol dir que s'ha de calcular un codi MAC diferent per a cada destinatari, i a més que no hi ha protecció contra el possible repudi per part de l'originador.

Per això, els sistemes de correu segur actuals fan servir les tècniques d'autenticació de clau pública, és a dir, les signatures digitals.

La signatura d'un missatge pot ser verificada per qualsevol que el rebi i conegui la clau pública del signant, i fins i tot es pot reenviar el missatge a altres usuaris, que també en podran comprovar l'autenticitat. I d'altra banda, les signatures proporcionen el servei de no repudi.

2.1.3. Compatibilitat amb els sistemes de correu no segur

Si volem fer servir la infraestructura SMTP existent per al correu segur, hem de tenir present que aquest protocol imposa certes restriccions per intentar maximitzar la interoperabilitat entre les implementacions, incloses les més antigues.

Algunes d'aquestes restriccions són, per exemple, que en principi els missatges només poden contenir caràcters ASCII, i que les línies dels missatges no poden tenir més de 1000 caràcters. Actualment molts agents SMTP són capaços de treballar sense aquestes restriccions, però de totes maneres hem de tenir-les en compte perquè no sabem per quines implementacions poden passar els nostres missatges.

A més, SMTP defineix una representació per als missatges que pot ser diferent de la representació local de cada sistema. El procés que s'encarrega d'enviar els missatges ha de transformar-los del format local al format SMTP a l'hora

de transferir-los. I a la inversa, quan arriba un missatge via SMTP normalment es converteix al format local abans d'emmagatzemar-lo en la bústia dels destinatari.

Exemples de transformació a format SMTP

Un exemple típic de transformació és el dels finals de línia: en SMTP es representen amb els caràcters <CR><LF>, mentre que en Unix es representen només amb <LF>.

Un altre exemple: alguns lectors de correu en Unix, especialment els més antics, interpreten que en una bústia de missatges la seqüència de caràcters "`From `" al començament de línia indica l'inici d'un nou missatge. En aquests sistemes, quan arriba un missatge que contingui aquesta seqüència al començament d'una línia, s'afegeix automàticament el caràcter ">" al davant de la línia. (Els lectors de correu més moderns fan servir el camp `Content-Length` de la capçalera per saber on acaba cada missatge i comença el següent.)

Per tant, quan s'han d'aplicar operacions criptogràfiques a un missatge, cal fer-ho sobre una **codificació canònica** que sigui convertible al format local de forma no ambigua.

Així, si hem d'enviar un missatge confidencial, xifrarem la forma canònica del missatge perquè quan el receptor la desxifri pugui convertir-la al seu format local. I si hem de calcular un codi d'autenticació o una signatura, ho farem també sobre la forma canònica, perquè el receptor sàpiga exactament a partir de quines dades s'ha generat i pugui realitzar la seva verificació.

Cada sistema de correu segur pot definir la seva codificació canònica. Per exemple, els sistemes basats en MIME fan servir el model de l'especificació RFC 2049. En el correu multimèdia les regles de codificació depenen del tipus de contingut, però en el cas del missatges de text, per simplicitat, el més normal és que la codificació canònica coincideixi amb el format SMTP, és a dir, amb caràcters ASCII i amb línies acabades en <CR><LF>.

D'altra banda, hi ha agents de correu que poden introduir modificacions en els missatges per adaptar-los a les seves restriccions. Alguns exemples són: posar a 0 el $8^{è}$ bit de cada caràcter, tallar les línies massa llargues inserint-hi finals de línia, eliminar els espais en blanc al final de línia, convertir els tabuladors en seqüències d'espais, etc.

Com que la informació criptogràfica constarà en general de dades arbitràries, hem d'utilitzar mecanismes de protecció del contingut per garantir que cap d'aquestes modificacions afectaran als missatges segurs. Aquest és el mateix problema que es va plantejar en el correu MIME per enviar continguts multimèdia, i la solució consisteix en utilitzar una **codificació de transferència**, com pot ser la codificació "base 64".

Codificació base 64

La codificació base 64 consisteix en representar cada grup de 6 bits amb un caràcter ASCII d'un joc de 64 ($2^6 = 64$), format per les lletres, els dígits, i els símbols "+" i "/".

2.2. S/MIME

S/MIME (*Secure MIME*) és una especificació de correu segur basada en l'estàndard PKCS #7, que va ser desenvolupada inicialment per RSA Data Security (el centre de recerca d'aquesta empresa és també l'origen de la sèrie d'estàndards PKCS).

El sistema de correu S/MIME fa servir la tècnica MIME per trametre missatges protegits criptogràficament segons el format PKCS #7.

Ús de MIME en S/MIME

Com que utilitza la representació MIME, S/MIME es pot integrar amb altres protocols d'aplicació a part del correu electrònic que també facin ús de MIME, com per exemple HTTP.

Durant els primers anys es van elaborar diferents esborranys de l'especificació S/MIME, que van ser adoptats per diversos implementadors independents. Les versions inicials definien dos perfils d'ús: el normal, que no era exportable fora dels Estats Units, i el restringit, que imposava un límit de 40 bits secrets en les claus simètriques. Per motius d'interoperabilitat, molts dels sistemes S/MIME que es van desenvolupar seguien el perfil restringit.

L'any 1998 es van publicar els documents informatius RFC 2311 i 2312, que recullen les característiques comunes a la majoria d'implementacions que existien llavors, en el que es coneix com versió 2 de S/MIME. En aquesta versió ja no es fa distinció entre perfil normal o restringit.

Ús de PKCS #7 en S/MIME v2

En la versió 2 d'S/MIME s'utilitza PKCS #7 versió 1.5 (RFC 2315).

Algorismes previstos en S/MIME versió 2

La versió 2 d'S/MIME preveu els algorismes criptogràfics següents.

- Per al xifratge simètric: Triple DES i RC2.
- Per als resums de missatge (*hash*): MD5 i SHA-1.
- Per al xifratge de clau pública: RSA.

L'estàndard oficial de correu segur S/MIME és l'anomenada versió 3, publicada en els documents RFC 2632–2634, de juny de 1999. Una diferència respecte a S/MIME versió 2 és que, en lloc de fer servir directament l'estàndard PKCS #7, es basa en el nou estàndard CMS (RFC 2630). El CMS manté la compatibilitat amb PKCS #7 però inclou nous mètodes per a l'intercanvi de claus, en particular el mètode Diffie-Hellman (descrit a l'RFC 2631).

CMS

CMS és la sigla de *Cryptographic Message Syntax*.

La versió 3 està dissenyada en general per oferir el màxim possible de compatibilitat amb la versió 2. De totes maneres, convé saber que la versió 3 suporta nous serveis, coneguts com ESS, que inclouen:

ESS

ESS és la sigla de *Enhanced Security Services*.

- Rebuts signats.
- Etiquetes de seguretat, que donen informació sobre el nivell de sensibili-

tat del contingut d'un missatge (segons una classificació definida per una determinada política de seguretat).

- Llistes de correu segures.
- Certificats de signatura, que permeten associar directament una signatura amb el certificat necessari per validar-la.

2.2.1. El format PKCS #7

PKCS #7 és un format per representar missatges protegits criptogràficament. Quan la protecció està basada en criptografia de clau pública, en PKCS #7 s'utilitzen certificats X.509 per garantir l'autenticitat de les claus.

L'estàndard PKCS #7 defineix unes estructures de dades per representar cadascun dels camps que formen part d'un missatge. A l'hora d'intercanviar aquestes dades s'han de codificar segons les regles especificades per la notació ASN.1 (la mateixa que s'utilitza per representar els certificats X.509 i les CRL).

Aquesta és l'estructura general d'un missatge PKCS #7, descrita amb la mateixa notació que vam fer servir per als certificats ("opc." vol dir opcional i "rep." vol dir repetible):

Camp	**Tipus**
`contentType`	identificador únic
`content` (opc.)	`Data`, `SignedData`, `EnvelopedData`, `SignedAndEnvelopedData`, `DigestedData`, o `EncryptedData`

El camp `contentType` és un identificador que indica quina de les sis estructures possibles hi ha al camp `content`. Aquestes estructures són:

1) `Data`: serveix per representar dades literals, sense aplicar-hi cap protecció criptogràfica.

2) `SignedData`: representa dades signades digitalment.

3) `EnvelopedData`: representa un missatge amb sobre digital (és a dir, un missatge xifrat simètricament al qual s'afegeix la clau simètrica xifrada

amb la clau pública de cada destinatari).

4) `SignedAndEnvelopedData`: representa dades signades i "tancades" en un sobre digital.

5) `DigestedData`: representa dades a les quals s'afegeix un resum o *hash*.

6) `EncryptedData`: representa dades xifrades amb clau secreta.

El camp `content` és opcional perquè en certs casos existeix la possibilitat que les dades d'un missatge no estiguin dins del propi missatge sinó en algun altre lloc.

D'aquests sis possibles tipus de contingut, els tres últims no es fan servir en S/MIME: per a dades signades i amb sobre es fa servir una combinació de `SignedData` i `EnvelopedData`, i els missatges que només contenen dades amb *hash* o dades xifrades simètricament no s'envien mai amb correu electrònic segur.

Per tant, els tipus de contingut PKCS #7 que pot haver-hi en un missatge S/MIME són `Data`, `SignedData` o `EnvelopedData`.

1) El tipus `Data`

Si el contingut PKCS #7 és de tipus `Data`, no està estructurat de cap manera especial, sinó que simplement consisteix en una seqüència de bytes. Quan s'utilitza en S/MIME, el seu contingut ha de ser una **part de missatge MIME**, amb les seves capçaleres i el seu cos en forma canònica.

El tipus `Data`, però, no apareix mai sol en un missatge S/MIME, sinó que sempre s'usa en combinació amb algun dels altres tipus PKCS #7, és a dir, amb `SignedData` o amb `EnvelopedData`.

2) El tipus `SignedData`

El tipus `SignedData` bàsicament conté dades, representades recursivament amb un altre missatge PKCS #7, i la signatura d'aquestes dades generada per un o més signants. L'estructura del contingut de tipus `SignedData` és aquesta:

Camp	Tipus
`version`	enter
`digestAlgorithms` (rep.)	
`algorithm`	identificador únic
`parameters`	(depèn de l'algorisme)
`contentInfo`	missatge PKCS #7
`certificates` (opc. rep.)	certificat X.509
`crls` (opc. rep.)	CRL
`signerInfos` (rep.)	
`version`	enter
`issuerAndSerialNumber`	
`issuer`	DN
`serialNumber`	enter
`digestAlgorithm`	
`algorithm`	identificador únic
`parameters`	(depèn de l'algorisme)
`authenticatedAttributes` (opc. rep.)	atribut X.501
`digestEncryptionAlgorithm`	
`algorithm`	identificador únic
`parameters`	(depèn de l'algorisme)
`encryptedDigest`	cadena de bytes
`unauthenticatedAttributes` (opc. rep.)	atribut X.501

El significat de cada camp és el següent:

- El camp `version` indica la versió del format de l'estructura `SignedData`.
- El camp `digestAlgorithms` és una llista dels algorismes de resum o *hash* que han fet servir els signants per signar les dades.
- El camp `contentInfo` conté les dades a signar, representades en forma de missatge PKCS #7. Aquest missatge normalment serà de tipus `Data` (per signar dades literals) o de tipus `EnvelopedData` (per signar un missatge confidencial).
- Al camp `certificates` hi ha certificats o cadenes de certificats que poden ser útils per verificar l'autenticitat de les claus públiques dels signants.
- Al camp `crls` hi ha una llista de CRL que es poden usar juntament amb els certificats del camp anterior.
- El camp `signerInfos` conté una estructura per cada signant, amb els següents sub-camps:

– `version` indica la versió del format d'aquesta estructura.

– `issuerAndSerialNumber` serveix per saber quina és la clau pública del signant. En lloc d'especificar la clau directament, es dóna el nom d'una CA i un número de sèrie de certificat. Aquestes dades iden-

Algorismes de *hash*

El camp `digestAlgorithms` apareix abans que les dades per facilitar el processament de l'estructura `SignedData` en una sola passada: sabent quins són els algorismes de *hash*, a mesura que es llegeixen les dades es poden anar calculant els resums.

tifiquen de manera única un certificat (una mateixa CA no pot generar dos certificats amb el mateix número de sèrie), i en aquest certificat, que pot ser un dels que hi ha al camp `certificates`, hi ha d'haver la clau pública del signant.

Identificació de les claus públiques

El mètode que usa PKCS #7 per identificar una clau pública, a partir d'un nom de CA i un número de sèrie, es va definir així per compatibilitat amb el sistema de correu segur PEM.

- `digestAlgorithm` és l'algorisme de *hash* que ha usat aquest signant (ha de ser un dels que hi havia al camp `digestAlgorithms` del començament).
- `authenticatedAttributes` és un conjunt d'atributs que s'afegeixen a les dades sobre les quals es calcula la signatura.
- `digestEncryptionAlgorithm` és l'algorisme amb què el signant ha xifrat el *hash* per calcular la signatura.
- `encryptedDigest` és la signatura, és a dir, el *hash* xifrat amb la clau privada del signant.
- `unauthenticatedAttributes` és un conjunt addicional d'atributs que no intervenen en la signatura.

Com podeu veure, PKCS #7 permet que cada signant afegeixi atributs a la seva signatura, que poden ser autenticats o no autenticats. Cadascun d'aquests atributs segueix l'estructura definida a la Recomanació X.501, que simplement consta d'un nom d'atribut i un o més valors.

Quan hi ha atributs autenticats, un d'ells ha de ser obligatòriament un atribut anomenat `messageDigest`, que té per valor el *hash* del camp `contentInfo`. Llavors, la signatura es calcula a partir del *hash* de tot el sub-camp `authenticatedAttributes`. Així es pot comprovar l'autenticitat de les dades del missatge (`contentInfo`) i de la resta d'atributs autenticats.

Exemple d'atribut autenticat

Un exemple típic d'atribut autenticat és l'atribut `signingTime`, que indica quan es va generar la signatura.

Quan no hi ha atributs autenticats, la signatura es calcula simplement a partir del *hash* del camp `contentInfo`.

3) El tipus `EnvelopedData`

El tipus `EnvelopedData` conté dades amb sobre digital. Les dades corresponen recursivament a un altre missatge PKCS #7. L'estructura del contingut de tipus `EnvelopedData` és aquesta:

Camp	Tipus
version	enter
recipientInfos (rep.)	
version	enter
issuerAndSerialNumber	
issuer	DN
serialNumber	enter
keyEncryptionAlgorithm	
algorithm	identificador únic
parameters	(depèn de l'algorisme)
encryptedKey	cadena de bytes
encryptedContentInfo	
contentType	identificador únic
contentEncryptionAlgorithm	
algorithm	identificador únic
parameters	(depèn de l'algorisme)
encryptedContent (opc.)	cadena de bytes

El significat de cada camp és el següent:

- El camp `version` indica la versió del format de l'estructura `EnvelopedData`.
- El camp `recipientInfos` conté una estructura per cada destinatari del missatge, amb els següents sub-camps:

– `version` indica la versió del format d'aquesta estructura.

– `issuerAndSerialNumber` serveix per saber a quin destinatari correspon aquesta estructura. Cada destinatari ha de buscar entre els elements del camp `recipientInfos` el que tingui aquest sub-camp igual a la CA i el número de sèrie del seu certificat.

– `keyEncryptionAlgorithm` indica amb quin algorisme de clau pública s'ha xifrat la clau de sessió.

– `encryptedKey` és la clau de sessió xifrada amb la clau pública d'aquest destinatari.

- Al camp `encryptedContentInfo` hi ha la informació sobre les dades xifrades, amb aquests sub-camps:

– `contentType` indica quin tipus de missatge PKCS #7 hi ha a les dades xifrades: normalment serà `Data` (quan es xifren dades literals) o `SignedData` (quan es xifra un missatge signat).

– `contentEncryptionAlgorithm` és l'algorisme simètric amb què s'han xifrat les dades (fent servir la clau de sessió).

– `encryptedContent` són les dades (és a dir, un missatge PKCS #7) xifrades.

Com hem vist, els continguts de tipus `SignedData` i `EnvelopedData` contenen recursivament altres missatges PKCS #7. La combinació que es pot trobar en un missatge S/MIME serà una d'aquestes quatre:

- `SignedData`[`Data`] (missatge amb dades signades).
- `SignedData`[`EnvelopedData`[`Data`]] (missatge amb dades xifrades i signades).
- `EnvelopedData`[`Data`] (missatge amb dades xifrades).
- `EnvelopedData`[`SignedData`[`Data`]] (missatge amb dades signades i xifrades).

Podem veure, doncs, que si un missatge s'ha de signar i xifrar, es poden fer les dues operacions en qualsevol ordre, segons interessi. Per exemple, signar primer i xifrar després permet que el destinatari guardi el missatge desxifrat per a verificacions posteriors, possiblement per part de tercers. Xifrar primer i signar després permet verificar l'autenticitat d'un missatge sense necessitat de desxifrar-lo.

2.2.2. Format dels missatges S/MIME

Un missatge S/MIME és un missatge MIME amb les següents característiques:

- El seu tipus de contingut (camp `Content-Type` de la capçalera MIME) és "`application/pkcs7-mime`".
- En el seu cos hi ha una estructura PKCS #7 codificada segons la notació ASN.1.

Per als missatges S/MIME que només estiguin signats hi ha una representació alternativa, anomenada **signatura en clar**, que veurem més endavant.

Compatibilitat amb versions anteriors d'S/MIME

Per als tipus de contingut, com "`pkcs7-mime`" i altres que veurem a continuació, les primeres versions d'S/MIME feien servir noms començats per "`x-`".

En les versions experimentals d'alguns protocols és costum utilitzar un prefix com aquest per representar valors que encara no estan estandarditzats. Per compatibilitat amb aquestes versions, les aplicacions de correu S/MIME haurien de considerar els noms antics equivalents als noms sense prefix, com per exemple:

Nom antic	Nom actual
`x-pkcs7-mime`	`pkcs7-mime`
`x-pkcs7-signature`	`pkcs7-signature`
`x-pkcs10`	`pkcs10`

La capçalera MIME `Content-Type`, a més del valor "`application/`

pkcs7-mime", ha de tenir almenys un d'aquests dos paràmetres:

- smime-type: indica el tipus de contingut PKCS #7 que hi ha al cos del missatge.
- name: indica un nom de fitxer associat al contingut del missatge.

Fitxer associat a un missatge S/MIME

El paràmetre name serveix per mantenir la compatibilitat amb les primeres versions d'S/MIME, en les quals no estava definit el paràmetre smime-type. En aquestes versions, la manera d'especificar el tipus de contingut PKCS #7 era amb l'extensió d'un nom de fitxer. La part del nom que hi hagi abans de l'extensió és indiferent, però per conveni sol ser "smime".

Per especificar aquest nom de fitxer es pot fer servir el paràmetre name de la capçalera Content-Type, i també el paràmetre filename de la capçalera MIME Content-Disposition (definida a l'especificació RFC 2183), amb el valor d'aquesta capçalera igual a "attachment".

A més, com que el cos del missatge són dades binàries (la representació ASN.1 d'una estructura PKCS #7), normalment hi haurà una capçalera Content-Transfer-Encoding amb valor "base64" per indicar que aquestes dades estan codificades en base 64.

Hi ha tres formats bàsics de missatges S/MIME: els missatges amb sobre digital, els missatges signats, i els missatges signats en clar.

1) Missatges S/MIME amb sobre digital

Un missatge S/MIME amb sobre digital té aquestes característiques:

- Al cos del missatge hi ha una estructura PKCS #7 amb tipus de contingut EnvelopedData.
- El valor del paràmetre smime-type és "enveloped-data".
- Si s'especifica un nom de fitxer associat, la seva extensió és .p7m (per exemple, "smime.p7m").

Missatges signats i xifrats

El contingut EnvelopedData que hi ha en un missatge S/MIME amb sobre digital pot contenir una estructura SignedData: llavors es tracta d'un missatge signat i xifrat.

Aquest és un exemple de missatge S/MIME amb sobre digital:

```
Date: Mon, 1 Mar 2004 11:46:10 +0100
From: usuari-1@uoc.edu
Subject: Exemple 1
To: usuari-2@uoc.edu
MIME-Version: 1.0
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
        name="smime.p7m"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="smime.p7m"

MIAGCSqGSIb3DQEHA6CAMIACAQAxgDBzAgEAMC8wKjELMAkGA1UEBhMCRVMxDDAKBgNVBAoT
A1VPQzENMAsGA1UEAxMEQ0EtMQIBBjANBgkqhkiG9w0BAQEFAAQuCYHs970aZmmqKTr3gemZ
LzHVtB266O/TrIv4shSvos8Ko8mUSQGov0JSIugmeDBzAgEAMC8wKjELMAkGA1UEBhMCRVMx
DDAKBgNVBAoTA1VPQzENMAsGA1UEAxMEQ0EtMQIBBzANBgkqhkiG9w0BAQEFAAQuC14oIhps
+mh8Wxp79A81uv2ltG3vt6J9UdJQcrDL92wD/jpw1IKpoR224LT4PQAAMIAGCSqGSIb3DQEH
ATARBgUrDgMCBwQIZbTj6XqCRkGggARAF8K8apgPtK7JPS1OaxfHMDXYTdEG92QXfAdTPetA
FGuPfxpJrQwX2omWuodVxP7PnWT2N5KwE1oc6faJY/zG0AAAAAAAAAAAAAA=
```

2) Missatges S/MIME signats

Un missatge S/MIME signat té un format anàleg al dels missatges amb sobre digital. Les seves característiques són:

Missatges xifrats i signats

El contingut `SignedData` que hi ha en un missatge S/MIME signat pot contenir una estructura `Enveloped`: llavors es tracta d'un missatge xifrat i signat.

- Al cos del missatge hi ha una estructura PKCS #7 amb tipus de contingut `SignedData`.
- El valor del paràmetre `smime-type` és "`signed-data`".
- Si s'especifica un nom de fitxer associat, la seva extensió és la mateixa que en els missatges amb sobre digital, és a dir `.p7m` (per exemple, "`smime.p7m`").

Aquest és un exemple de missatge S/MIME signat:

```
Date: Mon, 1 Mar 2004 11:47:25 +0100
From: usuari-1@uoc.edu
Subject: Exemple 2
To: usuari-2@uoc.edu
MIME-Version: 1.0
Content-Type: application/pkcs7-mime; smime-type=signed-data;
        name="smime.p7m"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="smime.p7m"

MIAGCSqGSIb3DQEHAqCAMIACAQExDjAMBggqhkiG9w0CBQUAMIAGCSqGSIb3DQEHAaCAJIAE
OUNvbnRlbnQtVHlwZTogdGV4dC9wbGFpbg0KDQpFeGVtcGxlIGRlIG1pc3NhdGdlIHNpZ25h
dC4NCgAAAAAAAKCAMIIBSDCCAQWgAwIBAgIBBjANBgkqhkiG9w0BAQQFADAqMQswCQYDVQQG
EwJFUzEMMAoGA1UEChMDVU9DMQ0wCwYDVQQDEwRDQS0xMB4XDTAwMDkwMTAwMDAwMFoXDTEw
MDkwMTAwMDAwMFowVTELMAkGA1UEBhMCRVMxDDAKBgNVBAoTA1VPQzElMCMGCSqGSIb3DQEJ
ARYWdXN1YXJpLTFAY2FtcHVzLnVvYy5lczERMA8GA1UEAxMIdXN1YXJpLTEwSTANBgkqhkiG
9w0BAQEFAAM4ADA1Ai4MNRLOaI30lrRQjyyznTjQTs/vLveiFadRiGKlVNnsFkGx/EwHdFDy
7z4CpbtnAgMBAAEwDQYJKoZIhvcNAQEEBQADLgACRZrDsL/MJv9VZdbxNMpbjKcwwFPPVG9L
TqOZ8sTdAF09UnFsSj5jE0ABAPEAADGAMIGBAgEBMC8wKjELMAkGA1UEBhMCRVMxDDAKBgNV
BAoTA1VPQzENMAsGA1UEAxMEQ0EtMQIBBjAMBggqhkiG9w0CBQUAMA0GCSqGSIb3DQEBAQUA
BC4DMwI+4fvRqBPhFj/wB7gI+Or7nSYfkqP1fxbjdTqwu9B5jsnxDIs+PUYsboQIAAAAAAAA
AAA=
```

3) Missatges S/MIME signats en clar

Quan s'envia un missatge signat, els receptors que facin servir un lector de correu apropiat podran llegir el missatge i verificar la signatura. Moltes vegades, però, interessa que el missatge pugui ser llegit per tothom, encara que no es disposi d'un lector amb suport per a correu segur.

El que es fa en aquests casos és formar un missatge amb dues parts: la primera es representa com un missatge normal, que pot ser llegit amb qualsevol client de correu, i la segona és la signatura de la primera. Un missatge d'aquest tipus s'anomena **missatge signat en clar**.

Així, qui disposi d'un lector de correu segur podrà llegir el missatge i verificar la signatura, i qui faci servir un lector tradicional podrà igualment llegir el missatge, encara que no pugui comprovar si la signatura és autèntica.

Una de les especificacions de l'estàndard MIME, l'RFC 1847, defineix la manera d'afegir una signatura a un missatge MIME. El missatge resultant té les següents característiques:

- El tipus de contingut del missatge (capçalera MIME `Content-Type`) és "`multipart/signed`".
- La capçalera `Content-Type` té tres paràmetres obligatoris.

– `boundary`: com en tots els missatges de tipus `multipart`, aquest paràmetre indica el delimitador que es fa servir per separar les parts.

– `protocol`: indica el tipus de contingut que hi ha a la part del missatge que conté la signatura.

– `micalg`: indica l'algorisme o algorismes de *hash*, també anomenat MIC, amb què està calculada la signatura (per facilitar el processament del missatge en una sola passada).

- El cos del missatge consta de dues parts MIME.

– La primera part és el missatge sobre el qual es calcula la signatura. Té l'estructura d'una part MIME, amb les seves capçaleres i el seu cos. Com a cas particular, pot tractar-se d'un missatge MIME multipart si, per exemple, es vol signar un missatge amb imatges o documents annexos.

– La segona part conté la signatura, calculada a partir de la forma canònica de la part anterior. Aquesta segona part també ha de tenir l'estructura d'una part MIME, i el valor de la seva capçalera `Content-Type` ha de ser igual al del paràmetre `protocol` de la capçalera `Content-Type` del missatge principal.

MIC

MIC és la sigla de *Message Integrity Code*, que és la nomenclatura que feia servir el sistema PEM per referir-se al mètode d'autenticació de missatge (quan s'utilitzen claus públiques, aquest mètode és una signatura digital).

Quan s'aplica aquesta tècnica MIME de signatures en clar a S/MIME, les característiques dels missatges són les següents.

- El paràmetre `protocol`, i per tant la capçalera `Content-Type` de la segona part del missatge, ha de tenir el valor "`application/pkcs7-signature`".
- Si s'especifica un nom de fitxer associat a la signatura, és a dir, a la segona part, la seva extensió és `.p7s` (per exemple, "`smime.p7s`").

- Al cos de la segona part del missatge hi ha una estructura PKCS #7 amb tipus de contingut `SignedData`, però sense el camp `content` en el seu camp `contentInfo`.

Camp `content` no present

Recordeu que en un missatge PKCS #7 el camp `content` és opcional, i per tant hi ha la possibilitat que no hi sigui present. En els missatges S/MIME amb signatura en clar s'aprofita aquesta possibilitat.

A l'hora d'enviar un missatge S/MIME amb signatura en clar, a l'estructura `SignedData` de la segona part no es posen les dades signades perquè ja estan a la primera part del missatge. A l'hora de verificar la signatura, el receptor ha d'actuar igual que si en l'estructura `SignedData` de la segona part hi hagués les dades de la primera part.

En aquest cas, es diu que l'estructura PKCS #7 té dades signades **no incloses** (*"detached"*), per contrast amb l'altre format de missatges signats, en el qual l'estructura PKCS #7 té les dades signades **incloses** (*"attached"*).

Aquest és un exemple de missatge S/MIME signat en clar:

```
Date: Mon, 1 Mar 2004 11:47:40 +0100
From: usuari-1@uoc.edu
Subject: Exemple 3
To: usuari-2@uoc.edu
MIME-Version: 1.0
Content-Type: multipart/signed; boundary="20040301104740";
        protocol="application/pkcs7-signature"; micalg=md5

--20040301104740
Content-Type: text/plain

Exemple de missatge signat.

--20040301104740
Content-Type: application/pkcs7-signature; name="smime.p7s"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="smime.p7s"

MIAGCSqGSIb3DQEHAqCAMIACAQExDjAMBggqhkiG9w0CBQUAMIAGCSqGSIb3DQEHAQAAoIAw
ggFIMIIBBaADAgECAgEGMA0GCSqGSIb3DQEBBAUAMCoxCzAJBgNVBAYTAkVTMQwwCgYDVQQK
EwNVT0MxDTALBgNVBAMTBENBLTEwHhcNMDAwOTAxMDAwMDAwWhcNMTAwOTAxMDAwMDAwWjBV
MQswCQYDVQQGEwJFUzEMMAoGA1UEChMDVU9DMSUwIwYJKoZIhvcNAQkBFhZ1c3VhcmktMUBj
YW1wdXMudW9jLmVzMREwDwYDVQQDEwh1c3VhcmktMTBJMA0GCSqGSIb3DQEBAQUAAzgAMDUC
Lgw1Es5ojfSWtFCPLLOdONBOz+8u96IVp1GIYqVU2ewWQbH8TAd0UPLvPgKlu2cCAwEAATAN
BgkqhkiG9w0BAQQFAAMuAAJFmsOwv8wm/1Vl1vE0yluMpzDAU89Ub0tOo5nyxN0AXT1ScWxK
PmMTQAEA8QAAMYAwgYECAQEwLzAqMQswCQYDVQQGEwJFUzEMMAoGA1UEChMDVU9DMQ0wCwYD
VQQDEwRDQS0xAgEGMAwGCCqGSIb3DQIFBQAwDQYJKoZIhvcNAQEBBQAELgMzAj7h+9GoE+EW
P/AHuAj46vudJh+So/V/FuNlOrC70HmOyfEMiz49RixuhAgAAAAAAAAAAA==
--20040301104740--
```

2.2.3. Distribució de claus amb S/MIME

Com hem vist fins ara, el mètode que es fa servir en PKCS #7, i per tant en S/MIME, per identificar els usuaris i les seves claus públiques és a través dels seus certificats X.509.

Això vol dir que un usuari no necessita verificar les identitats dels altres perquè d'això ja s'encarreguen les autoritats de certificació. L'únic que ha de fer l'usuari és comprovar si el certificat (o cadena de certificats) del seu corresponsal està signat per una CA reconeguda i és un certificat vàlid, és a dir, no està caducat ni revocat.

Idealment, la distribució dels certificats dels usuaris s'hauria de poder fer mitjançant el servei de directori X.500, però si aquest servei no està disponible es poden utilitzar mètodes alternatius.

S/MIME defineix un tipus especial de missatge que serveix per transportar certificats o llistes de revocació. Es tracta d'un missatge S/MIME amb les següents característiques:

- Al cos del missatge hi ha una estructura PKCS #7 amb tipus de contingut `SignedData`, però sense dades signades (camp `content` de l'element `contentInfo`) ni signatures (camp `signerInfos` amb 0 elements). Per tant, els camps amb informació útil són `certificates` i/o `crls`.
- El valor del paràmetre `smime-type` és "`certs-only`".
- Si s'especifica un nom de fitxer associat, la seva extensió és `.p7c` (per exemple, "`smime.p7c`").

Aquest és un exemple de missatge S/MIME amb només certificats:

```
Date: Mon, 1 Mar 2004 11:48:05 +0100
From: usuari-1@uoc.edu
Subject: El meu certificat i el de la CA
To: usuari-2@uoc.edu
MIME-Version: 1.0
Content-Type: application/pkcs7-mime; smime-type=certs-only;
        name="smime.p7c"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="smime.p7c"

MIAGCSqGSIb3DQEHAqCAMIACAQExADALBgkqhkiG9w0BBwGggDCCAUgwggEFoAMCAQICAQYw
DQYJKoZIhvcNAQEEBQAwKjELMAkGA1UEBhMCRVMxDDAKBgNVBAoTA1VPQzENMAsGA1UEAxME
Q0EtMTAeFw0wMDA5MDEwMDAwMDBaFw0xMDA5MDEwMDAwMDBaMFUxCzAJBgNVBAYTAkVTMQww
CgYDVQQKEwNVT0MxJTAjBgkqhkiG9w0BCQEWFnVzdWFyaS0xQGNhbXB1cy51b2MuZXMxETAP
BgNVBAMTCHVzdWFyaS0xMEkwDQYJKoZIhvcNAQEBBQADOAAwNQIuDDUSzmiN9Ja0UI8ss504
0E7P7y73ohWnUYhipVTZ7BZBsfxMB3RQ8u8+AqW7ZwIDAQABMA0GCSqGSIb3DQEBBAUAAy4A
AkWaw7C/zCb/VWXW8TTKW4ynMMBTz1RvS06jmfLE3QBdPVJxbEo+YxNAAQDxMIIBGzCB2aAD
AgECAgEBMA0GCSqGSIb3DQEBBAUAMCoxCzAJBgNVBAYTAkVTMQwwCgYDVQQKEwNVT0MxDTAL
BgNVBAMTBENBLTEwHhcNMDAwOTAxMDAwMDAwWhcNMTAwOTAxMDAwMDAwWjAqMQswCQYDVQQG
EwJFUzEMMAoGA1UEChMDVU9DMQ0wCwYDVQQDEwRDQS0xMEgwDQYJKoZIhvcNAQEBBQADNwAw
NAItDgNwpzTW3wWW7che7zeVoV4DbqznSQfm5hnUe3kkZOelPU4o8DJqMav2JyxjAgMBAAEw
DQYJKoZIhvcNAQEEBQADLgACXnrqZYIk3CY+641wJs99q7mIC4bPK3O75IskUrvGxs1PvZRt
yoj8zZDJtcAAADGAAAAAAAAAAA=
```

Finalment, hi ha un altre tipus de missatge S/MIME per enviar **peticions de certificació** a una CA. Una petició de certificació és un missatge que conté les dades necessàries perquè la CA generi un certificat, bàsicament el nom del titular i la seva clau pública.

De vegades es fa servir com a petició de certificació un certificat X.500 autosignat per l'interessat. Altres vegades es fa servir una estructura de dades definida expressament per a aquest fi, especificada en l'estàndard PKCS #10.

El tipus de missatge S/MIME per enviar peticions de certificació té les següents característiques.

- Al cos del missatge hi ha una estructura PKCS #10.
- El valor de la capçalera `Content-Type` és "`application/pkcs10`".
- Si s'especifica un nom de fitxer associat, la seva extensió és `.p10` (per exemple, "`smime.p10`").

2.3. PGP i OpenPGP

PGP (*Pretty Good Privacy*) és un programari que proporciona funcions criptogràfiques i de gestió de claus, desenvolupat inicialment per Philip Zimmermann l'any 1990. Es pot utilitzar per protegir qualsevol mena de dades, però el seu ús més habitual és per enviar missatges de correu electrònic xifrats i/o signats.

Una de les característiques distintives de PGP és el mètode que utilitza per certificar l'autenticitat de les claus públiques. En lloc de recórrer a autoritats de certificació, com fa S/MIME, cada usuari pot certificar directament les claus que està convençut que són autèntiques. I també pot prendre decisions respecte a una clau desconeguda en funció de quins siguin els usuaris que han certificat aquesta clau.

Una altra característica pròpia de PGP és l'eficiència en l'intercanvi dels missatges ja que, sempre que sigui possible, les dades es comprimeixen abans de xifrar-les i/o després de signar-les.

Amb el pas del temps han anat apareixent diverses versions de PGP, algunes d'elles amb diferents variants, com per exemple versions "internacionals" per complir les restriccions d'exportació de programari criptogràfic fora dels Estats Units, o versions que per poder ser distribuïdes gratuïtament no inclouen algorismes subjectes a patents en certs països.

- Les versions de PGP fins a la 2.3 es consideren obsoletes: el format de les signatures és incompatible amb el de les versions posteriors.
- Després de la versió 2.5 es va introduir un altre canvi de format, a causa de les condicions d'ús d'algorismes patentats als Estats Units (en concret, l'algorisme RSA).
- Les versions 2.6.x i les seves variants han estat durant molt temps les més difoses. El format dels missatges, anomenat indistintament "versió 2" o "versió 3" (V3), està documentat en l'especificació RFC 1991.
- En les versions 4.x es van introduir, entre altres novetats, les claus d'una sola funció: claus només per xifrar o només per signar.
- El que es va començar a dissenyar amb el nom de "PGP 3.0" finalment va donar lloc a les versions 5.x, que utilitzen un nou format per als missatges, conegut com "versió 4" (V4) o "OpenPGP" i documentat a l'especificació

Algorismes en PGP 2.6.x

Els algorismes que utilitzen les versions 2.6.x de PGP són: IDEA per al xifratge simètric, RSA per al xifratge de clau pública, i MD5 per al *hash*.

RFC 2440.

- Les noves versions (PGP 6 i posteriors) afegeixen millores en la funcionalitat però mantenint la compatibilitat amb les anteriors.
- També s'està desenvolupant en paral·lel, com a part del projecte GNU, el programari GnuPG (*GNU Privacy Guard*), basat en OpenPGP i de distribució totalment lliure (no fa ús d'algorismes patentats com RSA o IDEA).

Algorismes en PGP 5.x

PGP 5.x contempla l'ús de nous algorismes, com Triple DES i CAST-128 per al xifratge simètric, DSA i ElGamal per a les signatures, i SHA-1 i RIPEMD-160 per al *hash*. Si treballa només amb IDEA, RSA i MD5, les versions 5.x poden generar missatges totalment compatibles amb les versions 2.6.x.

2.3.1. Format dels missatges PGP

GnuPG

Podeu trobar informació sobre el projecte GnuPG a `www.gnupg.org`.

Les dades que processa PGP es codifiquen amb unes estructures de dades anomenades **paquets PGP**. Un missatge PGP està format, doncs, per un o més paquets PGP.

Un paquet PGP és una seqüència de bytes, amb una capçalera que indica de quin tipus de paquet es tracta i la seva longitud, i a continuació un seguit de camps de dades que depenen del tipus de paquet.

El format V3 defineix 10 tipus de paquets, mentre que el format V4 en defineix 14. A continuació veurem els principals tipus de paquets PGP.

1) Paquet de dades literals

Serveix per representar dades en clar, sense xifrar (seria l'anàleg del contingut `Data` en PKCS #7).

En un paquet d'aquest tipus hi ha un camp que dóna el valor de les dades, i un altre que indica si aquest valor s'ha de processar com a text o com a dades binàries. En el primer cas, les seqüències <CR><LF> que hi hagi en el text corresponen a finals de línia i es poden convertir a la representació local a l'hora de visualitzar-les o guardar-les en fitxer, mentre que en el segon cas no s'han de modificar.

2) Paquet de dades comprimides

En aquest tipus de paquet hi ha un camp que indica l'algorisme de compressió, i un altre que conté una seqüència de bytes comprimida. Quan es descomprimeixen aquest bytes, el resultat ha de ser un o més paquets PGP.

Algorismes de compressió

L'algorisme de compressió que usa PGP és el ZIP (RFC 1951), i OpenPGP usa també l'algorisme ZLIB (RFC 1950).

Típicament el que es comprimeix és un paquet de dades literals, opcionalment precedit per un paquet de signatura (que veurem a continuació).

3) Paquet de dades xifrades amb clau simètrica

El contingut d'aquest paquet és directament una seqüència de bytes xifrats

amb un algorisme simètric. El resultat de desxifrar-los ha de ser un o més paquets PGP.

Típicament el que es xifra simètricament són paquets de dades en clar o paquets de dades comprimides.

Els paquets d'aquest tipus s'usen per enviar un missatge de correu xifrat amb sobre digital, o bé quan l'usuari vol simplement xifrar un fitxer. En el primer cas, cal adjuntar al missatge la clau simètrica xifrada de tal forma que només la pugui desxifrar el destinatari o destinataris. Això es fa amb paquets xifrats amb clau pública (el tipus de paquet PGP que veurem a continuació). En el segon cas, la clau no es guarda enlloc sinó que l'usuari l'ha de recordar quan vulgui desxifrar el fitxer. De fet, l'usuari no dóna directament la clau de xifratge sinó una *passphrase*, a partir de la qual s'obté la clau simètrica aplicant-hi una funció de *hash*.

Xifratge simètric en PGP

PGP utilitza el mode CFB per al xifratge simètric. En lloc d'especificar a part el vector d'inicialització (VI) s'usa un VI igual a zero, però a les dades a xifrar se'ls anteposa una cadena de 10 bytes: els 8 primers són aleatoris, i els altres 2 són redundants (iguals al 7è i 8è) i serveixen perquè el destinatari comprovi si ha usat la clau correcta per desxifrar.

4) Paquet de dades xifrades amb clau pública

En aquest tipus de paquet hi ha un camp que serveix per identificar la clau pública utilitzada, un altre que indica l'algorisme de xifratge, i un altre amb les dades xifrades.

L'ús típic d'aquest paquet és per xifrar una clau de sessió, amb la qual s'haurà generat un paquet de dades xifrades simètricament, per enviar un missatge amb sobre digital. La clau pública utilitzada en aquest cas és la de cadascun dels destinataris del missatge.

5) Paquet de signatura

Un paquet d'aquest tipus conté camps amb la informació següent.

- Classe de signatura, que pot ser:
  - Signatura de dades binàries.
  - Signatura de text canònic.
  - Certificat, és a dir, associació de clau pública amb nom d'usuari.
  - Revocació de clau pública.
  - Revocació de certificat.
  - Datació (*"timestamp"*).
- Data i hora en què s'ha creat la signatura.
- Identificador de la clau amb què s'ha creat.
- Algorismes utilitzats per al *hash* i el xifratge asimètric.
- La signatura, que s'obté aplicant els algorismes especificats a les dades a signar, concatenades amb els camps autenticats. En el format V3 aquests camps autenticats són la classe de signatura i la data de creació. En el format V4 es pot especificar quins altres camps s'autentiquen.
- Altres camps afegits en el format V4, entre els quals hi ha:

– Data de caducitat de la signatura i de la clau.

– Comentaris de l'autor de la signatura.

– Motiu de revocació (en el cas de les signatures de revocació).

La manera de saber a quines dades correspon una signatura depèn del context. Si és la signatura d'un missatge de correu, en el mateix missatge hi ha d'haver el paquet amb les dades (normalment dades literals) a continuació del de signatura. Si és un certificat o una revocació, la signatura ha d'anar després dels paquets de clau pública i de nom d'usuari corresponents (tot seguit veurem aquests dos tipus de paquets PGP).

També és possible signar el contingut d'un fitxer i deixar el paquet de signatura en un fitxer separat. Això se sol fer quan es distribueixen programes (com per exemple el mateix PGP) per garantir que una versió és autèntica i no és cap "cavall de Troia". En aquest cas l'associació entre dades i signatura es pot establir mitjançant els noms dels fitxers: per exemple, el fitxer amb la signatura es pot dir com l'original però amb l'extensió `.sig`.

6) Paquet de clau pública

Aquest tipus de paquet conté la següent informació relativa a una clau pública:

- La data de creació de la clau.
- L'algorisme a què correspon la clau.
- Els valors dels components de la clau. Si l'algorisme és RSA, aquests valors són el mòdul n i l'exponent públic e.

La clau pública d'un usuari es fa servir per enviar-li dades xifrades o per verificar les signatures que generi. Però els paquets corresponents (dades xifrades amb clau pública o signatura, respectivament) no contenen el valor de la clau pública utilitzada, sinó només el seu **identificador de clau**.

L'identificador d'una clau pública és un número de 8 bytes que es pot usar per buscar el valor de la clau en una base de dades.

Identificadors de claus PGP repetits

Les implementacions no han de suposar que els identificadors de clau són únics: podria haver-hi dues claus PGP diferents amb el mateix identificador. La probabilitat que passi això, però, és molt baixa (tret que es faci expressament), perquè hi pot haver 2^{64} ($> 10^{19}$) identificadors diferents.

Si per exemple una signatura està generada amb una clau que té un determinat identificador, i resulta que hi ha dues claus amb aquest identificador, cal verificar-la amb cadascuna de les claus per comprovar si és vàlida.

7) Paquet de nom d'usuari

El contingut d'un paquet d'aquest tipus és simplement una cadena de caràcters, que s'utilitza per identificar el propietari d'una clau pública. Per tant, té la mateixa funció que el DN del subjecte en els certificats X.509,

Altres camps del paquet de clau pública

En el format V3 hi ha un camp que indica el període de validesa de la clau, però totes les implementacions el posen a 0, que vol dir que les claus són vàlides per sempre. En el format V4 aquesta informació s'especifica en els paquets de signatura. D'altra banda, en V4 hi ha previstos components de la clau pública per a altres algorismes a més de l'RSA.

Valor de l'identificador de clau

En les claus V3 (que són sempre claus RSA) l'identificador és igual als 8 bytes de menys pes del mòdul públic n. En les claus V4 és igual als 8 bytes de menys pes de l'empremta (més endavant s'expliquen les empremtes PGP).

però sense una estructura predefinida.

Encara que el seu format és lliure, se sol seguir el conveni d'identificar els usuaris amb adreces de correu electrònic RFC 822, com per exemple:

```
Philip R. Zimmermann <prz@acm.org>
```

8) Paquet de clau privada

Aquest tipus de paquet serveix per guardar els components de la clau privada d'un usuari. No hi ha cap motiu per enviar-lo mai a cap altre usuari, i per tant el format exacte del paquet pot dependre de la implementació.

Per assegurar la confidencialitat, en el fitxer on es guardi aquest paquet els components secrets de la clau haurien d'estar xifrats, normalment amb una clau simètrica derivada d'una *passphrase*. Així, cada vegada que l'usuari vulgui desxifrar o signar un missatge amb la seva clau privada, haurà de donar aquesta *passphrase* per poder obtenir els valors necessaris.

Un usuari pot tenir diverses claus, associades al mateix o a diferents noms. En el fitxer on hi hagi els paquets de clau privada, a continuació de cadascun hi haurà el paquet o paquets de nom d'usuari corresponents.

9) Paquet de nivell de confiança en una clau

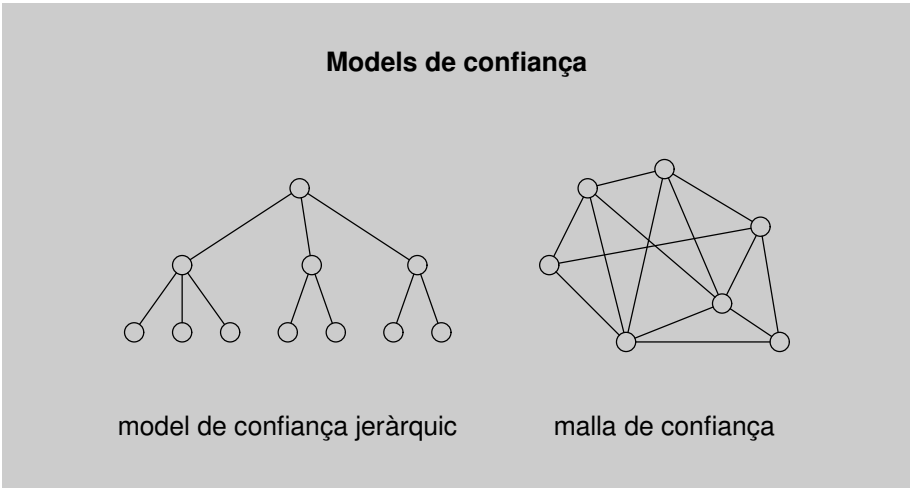
Aquest tipus de paquet tampoc s'envia mai sinó que només es guarda en el magatzem de claus propi de cada usuari, ja que únicament té significat per a qui l'ha generat. Serveix per indicar com és de fiable una clau com a certificadora, és a dir, a l'hora d'associar altres claus amb noms d'usuaris.

En el format V3 hi ha un altre tipus de paquet que serveix per incloure comentaris, però s'ha suprimit en el format V4 perquè cap implementació l'utilitzava.

El format V4 també fa servir altres 5 tipus de paquets, entre ells els relacionats amb les anomenades **sub-claus**. Una clau pot tenir associades una o més sub-claus: típicament, la clau principal s'utilitza per signar i les sub-claus per xifrar.

2.3.2. Distribució de claus PGP

Com hem esmentat al començament d'aquest apartat, la certificació de claus en PGP no segueix un model jeràrquic, com el de les autoritats de certificació X.509, sinó un model descentralitzat de confiança mútua, de vegades anomenat **malla de confiança** (*"web of trust"*).



Quan un usuari genera el seu parell de claus PGP (pública i privada), a la clau pública li ha d'associar un nom d'usuari, i a continuació ha d'auto-certificar aquesta associació, és a dir, signar amb la seva clau privada la concatenació de la clau pública i el nom. El paquet de la clau pública, el del nom d'usuari i el de la signatura formen un **bloc de clau**.

Lectura complementària

Al document `www.cl.cam.ac.uk/Research/Security/Trust-Register/gtr1998introduction.pdf` podeu trobar una comparació dels diferents models de confiança.

Opcionalment, l'usuari pot associar més d'un nom a la clau (per exemple, si té diverses adreces de correu electrònic), i llavors el bloc estarà format per la clau pública i tantes parelles nom–signatura com calgui.

Auto-certificació de claus

Quan es genera un nou parell de claus, les versions modernes de PGP automàticament signen la clau pública i el nom d'usuari amb la pròpia clau privada, però les versions més antigues no ho feien així i s'havia de generar l'auto-certificat expressament.

Llavors l'usuari pot enviar aquest bloc a altres amb qui hagi de mantenir correspondència. Cada receptor, si està convençut que la clau efectivament correspon a l'usuari originador, la certificarà afegint un altre paquet de signatura al bloc (o un per cada nom, si n'hi ha més d'un i també els considera autèntics). Així, cada usuari disposarà d'un magatzem de claus públiques certificades o **clauer** (*"keyring"*) que podrà fer servir per xifrar missatges i verificar signatures dels seus corresponsals.

A més de paquets de claus públiques, noms d'usuari i signatures de certificació, en un clauer també hi pot haver signatures de revocació per invalidar una clau o per anul·lar un certificat. La revocació ha d'estar signada per la pròpia clau (la que es vol invalidar o la que va generar el certificat que es vol anul·lar), i un cop emesa és irrevocable.

Claus de revocació

En OpenPGP també està prevista l'existència de claus autoritzades a revocar altres claus.

Una altra possibilitat per fer la distribució és a través d'un **servidor de claus PGP**, que gestiona un magatzem global de claus públiques amb els seus certificats (i revocacions, si és el cas). Hi ha diversos d'aquests servidors sincronitzats entre si, de manera que les actualitzacions del magatzem que es facin en un d'ells es propaguen automàticament a tots els altres.

Servidors de claus PGP

A l'adreça `www.rediris.es/cert/servicios/keyserver/` hi ha un d'aquests servidors de claus PGP.

Qualsevol usuari pot enviar a un servidor PGP els certificats de la seva clau, o

de les claus d'altres usuaris, perquè els afegeixi al clauer global. Llavors es poden fer consultes als servidors per obtenir la clau i els certificats associats a un determinat nom d'usuari (o als noms que continguin una determinada sub-cadena, si no es coneix el seu valor exacte).

Clauer global

La mida del "clauer global" emmagatzemat en els servidors de claus PGP és actualment de més de 2 Gbytes. A l'adreça `bcn.boulder.co.us/~neal/pgpstat/` podeu trobar un interessant estudi estadístic sobre les claus d'aquest magatzem.

És important remarcar que els servidors PGP no són autoritats de certificació: qualsevol clau pública que se'ls enviï serà afegida al magatzem sense fer cap verificació respecte a la identitat del propietari.

És responsabilitat de cada usuari que vulgui fer servir una clau d'un servidor PGP assegurar-se de la seva autenticitat. Per a això es pot tenir en compte quins altres usuaris han certificat aquesta clau.

Claus PGP falses

Són famoses, per exemple, les claus amb nom `<president@whitehouse.gov>` que han estat enviades als servidors PGP per persones que no tenen cap relació amb aquesta adreça de correu.

2.3.3. El procés de certificació PGP

Per facilitar l'intercanvi i la certificació de claus, PGP assigna a cada clau pública una **empremta** (*"fingerprint"*), que és simplement un *hash* del valor de la clau.

Aquesta empremta s'utilitza perquè un usuari pugui comprovar que la clau que ha rebut d'un altre, o d'un servidor de claus, sigui efectivament la que volia rebre i no una falsificació. L'identificador de clau no és suficient per a aquest fi, ja que és possible per a un impostor construir una clau pública de la qual conegui la clau privada i que tingui el mateix identificador que una altra clau pública. En canvi, construir una clau amb la mateixa empremta que una altra és pràcticament impossible.

Càlcul de l'empremta

Per calcular l'empremta de les claus V3, la funció *hash* que s'aplica és MD5 (16 bytes), mentre que per a les claus V4 és SHA-1 (20 bytes).

Col·lisions d'empremtes en PGP versió 3

Degut a la manera com es calcula l'empremta en les claus V3, sí que és possible tenir dues claus amb la mateixa empremta, però sempre amb diferent nombre de bits. Per tant, a l'hora d'autenticar una clau V3 el nombre de bits de la clau és una dada fonamental.

L'ús de l'empremta facilita la comprovació de l'autenticitat de la clau. L'alternativa seria comprovar-ne tots els bytes un per un, cosa que pot ser incòmoda i feixuga tenint en compte que actualment se solen fer servir claus de 1024 bits (256 dígits hexadecimals), o fins i tot de 2048 bits.

Suposem, per exemple, que un usuari *A* necessita certificar la clau d'un altre usuari *B* perquè ha d'intercanviar-hi missatges segurs. L'usuari *A* pot demanar a *B* que li enviï la seva clau pública per correu electrònic tradicional, o bé obtenir-la d'un servidor de claus. Llavors *A* ha de comprovar que ningú ha manipulat el missatge de resposta, obtenint de *B* la següent informació per un canal a part (no per correu electrònic):

- L'identificador de la clau pública de *B*.
- L'empremta d'aquesta clau.

- L'algorisme i el nombre de bits de la clau (el nombre de bits pot ser necessari per evitar col·lisions en l'empremta).

Mètodes per comunicar la informació sobre una clau pública

Un canal segur per enviar la informació anterior pot ser, per exemple, a través de comunicació directa "cara a cara" o de conversa telefònica. També hi ha qui es fa imprimir el valor de l'empremta PGP en la seva targeta, o qui el fa accessible via `finger` o HTTP (aquest mètode, però, no és tan segur).

Una altra possibilitat és organitzar una *"key-signing party"* o "reunió de signatura de claus".

Encara que PGP treballi internament amb identificadors de clau de 8 bytes, quan ha de mostrar els seus valors a l'usuari només en mostra els 4 bytes de menys pes. Per exemple, si una clau té per identificador el valor hexadecimal 657984B8C7A966DD, l'usuari només veu el valor C7A966DD. Això dóna més comoditat sense augmentar significativament, en la pràctica, les possibilitats de repetició.

Identificadors de 4 bytes repetits

En el cas límit, més de dos terços dels habitants de la Terra podrien tenir clau PGP sense que hi hagués cap identificador de 4 bytes repetit.

Aquest seria, doncs, un exemple de tota la informació necessària per certificar una clau:

```
bits /keyID    User ID
1024R/C7A966DD Philip R. Zimmermann <prz@acm.org>
Key fingerprint =  9E 94 45 13 39 83 5F 70  7B E7 D8 ED C4 BE 5A A6
```

Quan l'usuari *A* ha comprovat que els valors que li ha comunicat *B* coincideixen amb els calculats a partir de la clau pública rebuda electrònicament, ja pot certificar que aquesta clau correspon al nom o noms d'usuari que identifiquen l'usuari *B*.

2.3.4. Integració de PGP amb el correu electrònic

Tal com hem vist anteriorment, un missatge PGP consta d'una seqüència de paquets PGP. Hi pot haver diferents combinacions:

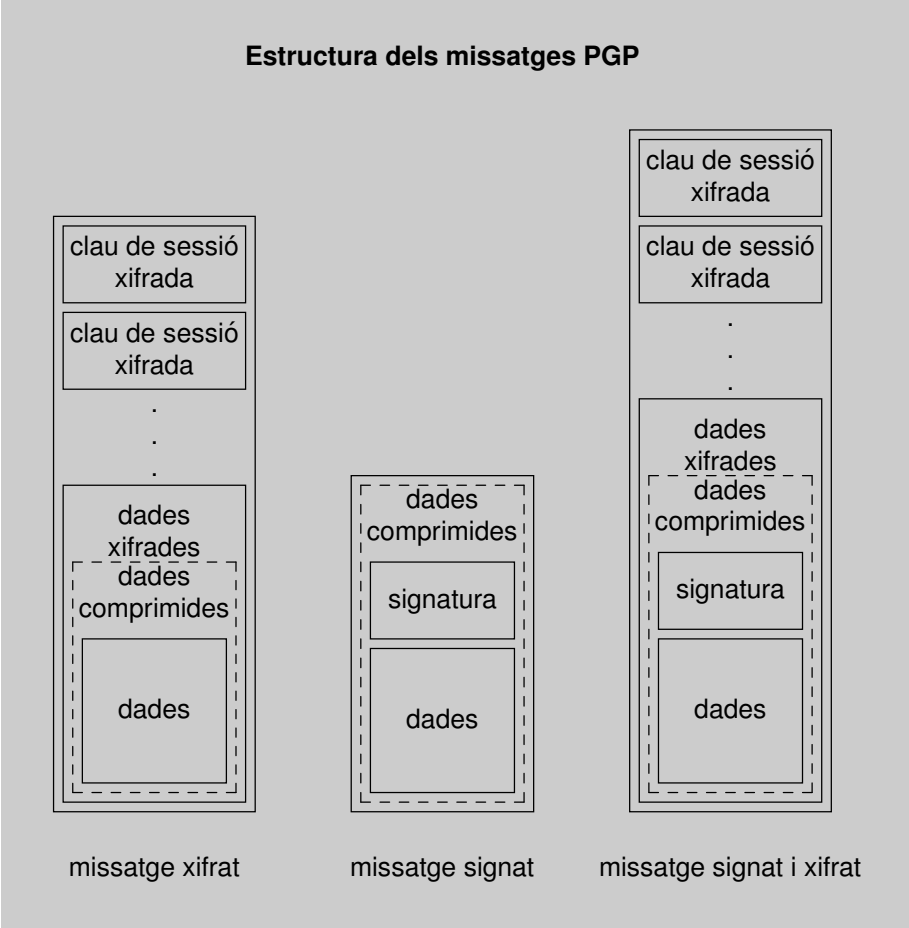
- Si és un missatge xifrat amb sobre digital, primer hi ha tants paquets com destinataris, cadascun amb la clau de sessió xifrada amb la clau pública del destinatari corresponent. A continuació hi ha el cos del missatge, possiblement comprimit, en un paquet xifrat simètricament amb la clau de sessió.

Xifratge de la clau de sessió en OpenPGP

OpenPGP també contempla la possibilitat de xifrar la clau de sessió amb claus simètriques, fent servir un nou tipus de paquet.

- Si és un missatge signat, primer hi ha el paquet de signatura, i després el cos del missatge en un paquet de dades literals. Opcionalment, aquests dos paquets es poden incloure dins d'un paquet comprimit.
- Si és un missatge signat i xifrat, l'estructura és com la dels missatges xi-

frats, excepte que quan es desxifra el paquet de dades xifrades el resultat és un missatge signat, és a dir, un paquet de signatura seguit d'un paquet de dades literals, o bé un paquet comprimit que quan es descomprimeix dóna els dos paquets anteriors.



Els missatges construïts d'aquesta manera contindran dades binàries arbitràries. Per enviar-los a través dels sistemes de correu electrònic tradicionals es poden utilitzar dues tècniques: encapsulació RFC 934 i MIME (amb el mètode anomenat PGP/MIME). Amb l'encapsulació RFC 934 es poden representar missatges xifrats i/o signats, missatges signats en clar, o blocs de claus públiques.

La tècnica d'encapsulació RFC 934

L'especificació RFC 934 defineix una tècnica per combinar un o més missatges en un únic cos. Aquesta és la tècnica que fa servir MIME per ajuntar diverses parts en un missatge multipart.

L'encapsulació RFC 934 consisteix en concatenar els missatges a combinar, posant-los simplement un a continuació de l'altre, i fent servir delimitadors d'encapsulació per indicar on comença cadascun i on acaba l'últim.

El text que hi hagi abans del primer delimitador es considera com un "pròleg" i el que hi hagi després de l'últim delimitador es considera com un "epíleg", però ni un ni l'altre formen part del missatges encapsulats.

Com a delimitadors d'encapsulació es fan servir línies que comencin amb un guió

seguit d'un altre caràcter que no sigui espai. Si dins dels missatges a encapsular hi ha alguna línia que comenci per guió, simplement se li anteposa un guió i un espai. A l'hora de desencapsular, a les línies que comencin per guió i espai se'ls suprimeixen aquests dos caràcters. Les que comencin per guió i un altre caràcter seran considerades com a delimitadors. Això permet l'encapsulació recursiva de missatges compostos dins d'un altre missatge compost.

1) Missatges PGP xifrats i/o signats

Aquest és un exemple de missatge PGP, codificat amb el mètode d'encapsulació RFC 934.

```
Date: Mon, 1 Mar 2004 11:35:40 +0100
From: usuari-1@uoc.edu
Subject: Exemple 4
To: usuari-2@uoc.edu

-----BEGIN PGP MESSAGE-----
Version: 2.6.3i

iQBDAwUBObNQzFDy7z4CpbtnAQF9aQFrBtyRK8bdaPFlht7KeFzO/N0lJTcnYhbS
TvlZsTwr6+iQJqHP5nKnYr0W/Q9mo60AI3QAAAAAAEV4ZW1wbGUgZGUgbWlzc2F0
Z2Ugc2lnbmF0Lg0K
=8gbQ
-----END PGP MESSAGE-----
```

Com podem veure a l'exemple, l'estructura del missatge és la següent.

- Com a delimitador inicial d'encapsulació es fa servir la cadena "`BEGIN PGP MESSAGE`" entre dues seqüències de 5 guions, i el delimitador final és igual però canviant "`BEGIN`" per "`END`".
- A continuació del delimitador inicial hi pot haver un seguit de capçaleres, amb camps com `Version` per indicar quina versió de PGP ha generat el missatge, `Comment` per introduir comentaris de l'usuari, o `Charset` per especificar el joc de caràcters utilitzat en el text del missatge. .
- Després de les capçaleres hi ha una línia en blanc, i el paquet o paquets PGP que formen el missatge codificats en base 64.
- Just després dels paquets PGP i abans del delimitador de final, hi ha una línia de 5 caràcters: el primer és el signe ''=" i els altres 4 són la codificació en base 64 d'un CRC de 24 bits de tots els bytes dels paquets. Aquest CRC serveix per comprovar que no s'hagin produït modificacions en el missatge que hagin pogut afectar la descodificació.

Joc de caràcters en OpenPGP

En OpenPGP, el joc de caràcters per defecte és Unicode (ISO/IEC 10646), codificat amb UTF-8 (RFC 2279)

En la terminologia PGP, la seqüència de línies des del delimitador d'encapsulació d'inici fins al de final s'anomena **armadura ASCII** del missatge.

2) Missatges PGP signats en clar

Igual que S/MIME, PGP també defineix un format per enviar missatges signats en clar, que permet llegir-ne el contingut als usuaris que no disposin de PGP. Aquest n'és un exemple:

```
Date: Mon, 1 Mar 2004 11:35:55 +0100
From: usuari-1@uoc.edu
Subject: Exemple 5
To: usuari-2@uoc.edu

-----BEGIN PGP SIGNED MESSAGE-----
Hash: MD5

Exemple de missatge signat.

-----BEGIN PGP SIGNATURE-----
Version: 2.6.3i

iQBDAwUBObNQzFDy7z4CpbtnAQF9aQFrBtyRK8bdaPFlht7KeFzO/N0lJTcnYhbS
TvlZsTwr6+iQJqHP5nKnYr0W/Q9mow==
=5TnX
-----END PGP SIGNATURE-----
```

En aquest cas hi ha dos sub-missatges encapsulats, amb la següent estructura.

- El delimitador d'inici de la primera part és la cadena "`BEGIN PGP SIGNED MESSAGE`", amb una seqüència de 5 guions al davant i al darrere.
- En el primer sub-missatge hi ha zero o més capçaleres `Hash`, que indiquen l'algorisme (o algorismes) de *hash* utilitzats per calcular la signatura (o signatures), seguits d'una línia en blanc i el cos del missatge. L'especificació de l'algorisme al començament permet processar el missatge en una sola passada. En absència d'aquest camp, per defecte s'entén que la funció de *hash* utilitzada és MD5.
- A continuació del primer sub-missatge hi ha l'armadura ASCII d'un o més paquets de signatura, amb un delimitador d'inici format per la cadena "`BEGIN PGP SIGNATURE`", també amb 5 guions al davant i al darrere, i amb un delimitador de final igual però canviant "`BEGIN`" per "`END`".

  Les signatures es calculen a partir del cos del missatge en forma canònica, és a dir, representant els finals de línia amb <CR><LF>. A més, PGP sempre elimina els espais en blanc i tabuladors que hi hagi abans d'un final de línia a l'hora d'obtenir les signatures.

Línies que comencen amb guió

Si en el primer sub-missatge hi ha línies que comencen per guió, cal afegir-los la seqüència "`- `" d'acord amb RFC 934. La signatura, però, s'obté de les línies originals.

3) Missatges de blocs de claus públiques

Hi ha un altre format d'armadura PGP que serveix per enviar blocs de claus públiques i certificats. El delimitador d'inici consta de la cadena "`BEGIN PGP PUBLIC KEY BLOCK`" envoltada per dues seqüències de 5 guions, i el de final és igual però canviant "`BEGIN`" per "`END`". Aquest és un exemple:

```
Date: Mon, 1 Mar 2004 11:38:20 +0100
From: usuari-1@uoc.edu
Subject: La meva clau PGP
To: usuari-2@uoc.edu

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.6.3i

mQA9AzmvN9AAAAEBbAw1Es5ojfSWtFCPLLOdONBOz+8u96IVp1GIYqVU2ewWQbH8
TAd0UPLvPgK1u2cAEQEAAbQhVXN1YXJpIDEgPHVzdWFyaS0xQGNhbXB1cy51b2Mu
ZXM+iQBCAwUQOa830FDy7z4CpbtnAQFdoQFox7LHdl8wdIA69f4REn14bVYxBaxw
4I35PJwRWqI2c+8T75vqUdBhiydsZ2Fo
=Ninr
-----END PGP PUBLIC KEY BLOCK-----
```

Qualsevol dels tipus d'armadura que hem vist es pot utilitzar per intercanviar informació PGP amb altres mitjans de transferència a més del correu electrònic: FTP, HTTP, etc.

4) PGP/MIME

Per incorporar PGP a MIME, inicialment es va definir el tipus de contingut MIME `application/pgp`. Més tard, però, aquest tipus de contingut es va abandonar en favor del mètode RFC 1847, que és el mateix que fa servir S/MIME per als missatges signats en clar. A més del tipus de contingut `multipart/signed` corresponent als missatges signats, RFC 1847 també defineix el tipus `multipart/encrypted` corresponent als missatges xifrats.

La tècnica per incloure missatges PGP en missatges MIME RFC 1847 s'anomena PGP/MIME, i està especificada a l'RFC 2015. PGP/MIME defineix tres tipus de contingut per a les parts MIME que representin missatges PGP: `application/pgp-encrypted`, `application/pgp-signature` i `application/pgp-keys`.

Actualment, però, és molt més habitual l'ús de les armadures ASCII per encapsular missatges PGP que no pas la tècnica PGP/MIME.

Resum

En aquest mòdul didàctic hem presentat dues aplicacions que fan ús dels mecanismes de protecció que hem vist al mòdul anterior. La primera d'elles és el *Secure Shell* **(SSH)**, que permet establir una connexió xifrada amb un servidor. SSH defineix el seu propi protocol per autenticar el servidor i l'usuari que s'hi vol connectar, i per determinar les claus per al xifratge simètric i per als codis MAC, de manera semblant a com ho fa SSL/TLS.

Un cop establerta la comunicació segura, el protocol SSH permet fer ús de diverses funcions, com són l'enviament de dades protegides per un canal, la **redirecció de ports TCP** des del client o des del servidor, etc.

L'altra aplicació que hem vist en aquest mòdul és el **correu electrònic segur**. Donat que en aquesta aplicació interessa protegir els missatges enviats més que la comunicació en si, es defineixen mecanismes per introduir la confidencialitat i l'autenticació en el cos dels missatges de correu. Això permet aprofitar la infraestructura d'agents de correu existents, mantenint la compatibilitat amb els sistemes de correu tradicionals.

Per a la confidencialitat normalment s'utilitza la tècnica del **sobre digital**, que consisteix en xifrar el missatge amb una clau de sessió simètrica, i afegir-hi aquesta clau de sessió xifrada amb la clau pública de cada destinatari. Així es pot enviar un mateix missatge a múltiples destinataris, com es fa amb el correu electrònic normal. Per a l'autenticació es fan servir les **signatures digitals**, que a més proporcionen el servei de no repudi.

Un dels principals sistemes de correu electrònic segur actualment en ús és **S/MIME**. Aquest sistema incorpora estructures de dades **PKCS #7** en els missatges de correu utilitzant la tecnologia MIME. L'estàndard PKCS #7 especifica com representar missatges xifrats amb sobre digital i/o signats, aplicant criptografia de clau pública i sobre una infraestructura de certificats X.509.

Un altre sistema per a l'intercanvi de dades protegides, i en particular missatges de correu electrònic, és **PGP**. Aquest sistema utilitza un format propi, publicat en especificacions com l'antiga "PGP versió 3" o la més moderna "OpenPGP", per representar les dades xifrades i/o signades. Per a les claus públiques no fa servir una infraestructura de certificats X.509, sinó que la certificació és descentralitzada: cada usuari pot certificar les claus públiques que cregui autèntiques, de manera que les relacions entre claus públiques d'usuaris que confien en altres formen una "malla de confiança".

Activitats

3-1 Una implementació lliure del protocol SSH prou coneguda és la del projecte OpenSSH. Visiteu la seva pàgina web (`www.openssh.com`) i comproveu quins protocols i quins algorismes criptogràfics suporta l'última versió.

3-2 Visiteu la pàgina web del projecte GnuPG (`www.gnupg.org`) i comproveu quins algorismes criptogràfics suporta l'última versió.

3-3 Accediu a un servidor de claus públiques PGP, per exemple `www.rediris.es/cert/servicios/keyserver/`. Quina informació cal donar-li per trobar una clau PGP? Quin tipus d'informació pot retornar?

Busqueu les claus públiques associades al nom "`Philip R. Zimmermann`". Hi ha algun motiu per pensar que alguna d'elles pugui ser falsa?

Exercicis d'autoavaluació

3-1 Una organització vol oferir un servei web segur, amb autenticació de servidor i confidencialitat de les dades, però no disposa d'un servidor HTTPS sinó de programari client i servidor del protocol SSH, que es pot instal·lar en qualsevol ordinador que hagi de fer ús d'aquest servei. Com es pot configurar el programari SSH per oferir el servei desitjat?

3-2 En els sistemes de correu electrònic segur com S/MIME o PGP:

a)Com s'aplica la protecció de confidencialitat a un mateix missatge de manera que pugui ser llegit per 5 destinataris diferents, i només aquests 5?

b)El remitent pot usar un client de correu que guardi còpia de cada missatge enviat. Si el missatge s'ha enviat xifrat, com pot fer el remitent per consultar més endavant el contingut original del missatge?

c)Si el remitent també vol signar el missatge, és necessari que el signi abans de xifrar-lo o després? Per què?

3-3 Observeu els exemples de missatges signats en clar S/MIME (pàgina 35) i PGP (pàgina 47). Sabríeu explicar per què la signatura del primer és més llarga que la del segon?

3-4 Si un atacant intenta un atac de repetició contra el correu S/MIME, reenviant per exemple un missatge com aquest amb el camp "`Date`" de la capçalera canviat:

```
Date: Mon, 14 Jun 2004 11:45:20 +0100
From: professor
Subject: Lliurament de la pràctica ajornat
To: estudiants
MIME-Version: 1.0
Content-Type: multipart/signed; boundary="20040614094520";
        protocol="application/pkcs7-signature"; micalg=md5

--20040614094520
Content-Type: text/plain

El lliurament de la pràctica que s'havia de realitzar aquest divendres
queda ajornat fins divendres de la setmana que ve.

El vostre professor.

--20040614094520
... (la signatura S/MIME del missatge) ...
--20040614094520--
```

hi hauria manera de detectar l'atac?

3-5 Si un atacant intenta un atac de repetició contra el correu PGP, reenviant per exemple un missatge com aquest amb el camp "`Date`" de la capçalera canviat:

```
Date: Mon, 14 Jun 2004 11:45:30 +0100
From: professor
Subject: Lliurament de la pràctica ajornat
To: estudiants

-----BEGIN PGP SIGNED MESSAGE-----
Hash: MD5

El lliurament de la pràctica que s'havia de realitzar aquest divendres
queda ajornat fins divendres de la setmana que ve.

El vostre professor.

-----BEGIN PGP SIGNATURE-----
... (la signatura PGP del missatge) ...
-----END PGP SIGNATURE-----
```

hi hauria manera de detectar l'atac?

3-6 La signatura d'un missatge PGP signat es calcula mitjançant l'algorisme de *hash* MD5, de 128 bits de sortida. El missatge signat inclou els primers 16 bits d'aquest *hash* en clar, per comprovar que les dades sobre les quals es vol verificar la signatura són correctes.

a)Fins a quin punt pot comprometre això la seguretat de l'algorisme de *hash*?

b)Fins a quin punt ajuden aquests bits a comprovar que les dades són correctes?

Solucionari

3-1 Es pot fer ús de la funcionalitat de redirecció de ports TCP que proporciona el protocol SSH. En l'ordinador on hi hagi el servidor web, que típicament admetrà connexions pel port 80, s'instal·la també el servidor SSH, configurat de manera que permeti l'autenticació nul·la (si no és necessària l'autenticació de client). Llavors, en cada ordinador que s'hagi de connectar al servidor, s'instal·la el client SSH configurant-lo de manera que les connexions que arribin a un port *P* local siguin redirigides a connexions fetes des del servidor al port 80 del propi servidor.

Un cop fet això, per accedir al servidor web des dels clients només cal donar URLs de la forma "`http://localhost:`*P*`/...`" al navegador, i automàticament s'establiran les connexions amb el servidor web mitjançant un canal SSH.

3-2

a)Amb la tècnica del sobre digital, és a dir, xifrant el missatge amb una clau de sessió simètrica, i afegint al missatge xifrat el resultat de xifrar la clau de sessió amb la clau pública de cadascun dels 5 destinataris.

b)Caldria afegir una 6ª còpia de la clau de sessió, aquesta xifrada amb la clau pública del remitent.

c)Depenent de l'ús que es vulgui fer del missatge en recepció, pot ser més interessant xifrar abans de signar, signar abans de xifrar, o pot ser indiferent.

3-3 En el missatge S/MIME hi ha una estructura PKCS #7 de tipus `SignedData`, que normalment inclourà almenys el certificat X.509 del signant (i possiblement també el de la CA emissora, el de la CA de nivell superior, etc.). En el missatge PGP només hi ha un identificador de la clau pública del signant: el valor sencer de la clau pública s'ha d'obtenir per altres mitjans (clauer local del receptor, servidor de claus públiques, etc.).

3-4 Es pot detectar la repetició perquè la signatura es representa mitjançant una estructura PKCS #7 `SignedData`, que pot incloure un atribut que indica quan d'ha generat la signatura: l'atribut `signingTime` (encara que aquest atribut no és obligatori en PKCS #7 ni CMS, se suposa que les implementacions S/MIME haurien d'incloure'l en els missatges signats).

3-5 Es pot detectar la repetició perquè el receptor pot saber quan s'ha generat la signatura: aquesta informació es troba en un dels camps del paquet de signatura (en el moment de verificar la signatura, les implementacions PGP normalment mostren la data en què es va crear).

3-6

a)Si el missatge és només signat no hi ha cap compromís, perquè tothom pot calcular el *hash* de les dades signades (si és signat i xifrat, primer es construeix el missatge signat i després es xifra el resultat).

b)La probabilitat que en un missatge incorrecte coincideixin els 16 primers bits del *hash* és 2^{-16}. Per tant, aquests 16 bits donen una confiança raonable que el missatge ha arribat correctament.

Glossari

Agent d'autenticació SSH: Aplicació que, a través de connexions SSH amb altres aplicacions, permet a aquestes altres aplicacions realitzar l'autenticació de client del protocol SSH, mitjançant les claus privades corresponents, i sense que aquestes claus hagin de sortir del sistema on s'està executant l'agent.

Armadura ASCII: Representació d'un missatge PGP apta per ser inclosa en el cos d'un missatge de correu electrònic, sense perill que els agents de correu la modifiquin fent irrecuperable el missatge PGP.

***Attached* (dades signades):** Vegeu *Signatura amb signades incloses*.

Base 64: Codificació que permet representar dades binàries arbitràries com a línies de caràcters ASCII, fent servir un caràcter per cada 6 bits.

Bloc de clau pública PGP: Conjunt format per una clau pública PGP, el nom o noms d'usuari associats, i els certificats que confirmen que la relació entre la clau i cada nom és autèntica.

Canal SSH: Cadascun dels diversos fluxos d'informació que es poden transmetre a través d'una connexió SSH (un canal pot ser una sessió interactiva, una connexió a un servidor de finestres X, una connexió TCP redirigida, o una connexió a un agent d'autenticació).

Clauer PGP: Base de dades que conté un conjunt de claus PGP.

CMS: Vegeu *Cryptographic Message Standard*.

Codi de redundància cíclica (CRC): Valor calculat a partir d'una seqüència de bits, per confirmar (dins d'un marge de probabilitat) que no s'ha produït un error de transmissió quan aquesta seqüència arriba al destinatari.

Codificació canònica: Representació dels missatges de correu electrònic que es fa servir per a la seva transferència, amb l'objectiu que tots els sistemes puguin convertir aquesta forma canònica, si cal, a la representació local que utilitzi cadascun.

Codificació de transferència: Representació del contingut dels missatges de correu electrònic que pot ser necessària per compatibilitat amb tots els possibles agents de correu que han de processar els missatges (per exemple, la codificació base 64).

CRC: Vegeu *Codi de redundància cíclica*.

***Cryptographic Message Standard* (CMS):** Versió del format PKCS #7 estandarditzada per l'IETF (*Internet Engineering Task Force*).

***Detached* (dades signades):** Vegeu *Signatura amb signades no incloses*.

Empremta (*fingerprint*): Valor resumit d'una clau pública PGP, obtingut amb una funció *hash*, que s'utilitza en comptes de la clau sencera quan s'ha de comparar amb un valor suposadament autèntic.

Encapsulació RFC 934: Tècnica per combinar diversos missatges de correu electrònic en un únic cos de missatge RFC 822.

***Fingerprint*:** Vegeu *Empremta*.

HMAC: Tècnica per calcular codis d'autenticació de missatge (MAC) basada en funcions *hash*.

Identificador de clau PGP: Número que serveix per identificar una clau pública dins d'un paquet PGP, per no haver d'incloure-hi el valor sencer de la clau, i que internament es representa amb 8 bytes, encara que a l'usuari normalment se li mostren només els 4 últims bytes.

Malla de confiança: Model de confiança mútua utilitzat en sistemes com PGP, on les claus públiques es poden autenticar mitjançant certificats generats per qualsevol usuari, en lloc de fer servir una estructura jeràrquica d'autoritats de certificació.

MIME: Vegeu *Multipurpose Internet Mail Extensions*.

Missatge MIME multipart: Missatge MIME que té el seu cos estructurat en diverses parts, cadascuna amb el seu contingut, que pot ser text, gràfics, dades, etc. o un altre missatge MIME (que al seu torn pot ser un missatge multipart).

***Multipurpose Internet Mail Extensions* (MIME):** Estàndard per a la inclusió d'altres tipus d'informació, diferents de simples línies de text, en el cos dels missatges de correu electrònic, de manera compatible amb l'estàndard RFC 822.

OpenPGP: Versió del format dels missatges PGP estandarditzada per l'IETF (*Internet Engineering Task Force*).

Paquet PGP: Estructura de dades utilitzada per representar els diversos tipus d'informació que hi ha en un missatge protegit amb PGP.

PEM: Vegeu *Privacy Enhanced Mail*.

PGP: Vegeu *Pretty Good Privacy*.

PKCS #7: Vegeu *Public Key Cryptographic Standard #7*.

PKCS #10: Vegeu *Public Key Cryptographic Standard #10*.

***Pretty Good Privacy* (PGP):** Aplicació per protegir dades, amb confidencialitat (sobre digital) i/o autenticitat (signatura digital), que fa servir claus públiques autenticades segons un esquema descentralitzat, i que s'utilitza típicament per protegir el correu electrònic.

***Privacy Enhanced Mail* (PEM):** Un dels primers sistemes de correu electrònic segur que es van desenvolupar, compatible directament amb el format RFC 822.

***Public Key Cryptographic Standard #7*:** Estàndard per representar missatges protegits criptogràficament (normalment amb sobre digital i/o signatura digital), a partir de claus públiques autenticades amb certificats X.509.

***Public Key Cryptographic Standard #10*:** Estàndard per representar peticions de certificació, que s'envien a una CA perquè aquesta generi un certificat a partir de les dades de la petició.

***Remote Shell*:** Aplicació que es va incorporar al sistema Unix BSD (amb el nom `rsh`), i actualment disponible en gairebé totes les versions de Unix, que permet als usuaris d'un sistema executar comandes en un altre sistema remot.

RFC 822: Estàndard per a la representació dels missatges de correu electrònic, estructurats en un conjunt de línies de capçalera, el cos del missatge, i una línia en blanc que separa les capçaleres del cos.

***Secure MIME* (S/MIME):** Sistema de correu electrònic segur que fa servir MIME per incloure dades PKCS #7 o CMS en els missatges, i per tant proporciona confidencialitat (sobre digital) i/o autenticitat (signatura digital) a partir de claus públiques autenticades amb certificats X.509.

***Secure Shell*:** Aplicació que proporciona un servei anàleg al del programa *Remote Shell* dels sistemes Unix, però amb la comunicació protegida mitjançant autenticació i xifratge, i amb funcionalitats afegides, com la redirecció de ports TCP a través de connexions segures, etc. També és el nom que rep el protocol utilitzat per aquesta aplicació per a la comunicació segura.

Signatura amb dades incloses (*attached*): Estructura de dades que representa una signatura i que inclou les dades signades.

Signatura amb dades no incloses (*detached*): Estructura de dades que representa una signatura però que no inclou les dades signades, que es troben en algun altre lloc (per exemple, una altra part del missatge).

Signatura en clar: Signatura (amb dades no incloses) que s'afegeix com a segona part d'un missatge signat, la primera part del qual conté les dades signades, i que permet llegir

el missatge encara que no es disposi del programari necessari per verificar la signatura.

***Simple Mail Transfer Protocol* (SMTP):** Protocol usat en Internet per a la transmissió de missatges de correu electrònic, especificat a l'estàndard RFC 821.

S/MIME: Vegeu *Secure MIME*.

SMTP: Vegeu *Simple Mail Transfer Protocol*.

Sobre digital: Tècnica per proporcionar confidencialitat, consistent en xifrar les dades amb una clau de sessió simètrica, i afegir al missatge aquesta clau de sessió xifrada amb la clau pública de cada destinatari.

SSH: Vegeu *Secure Shell*.

Sub-clau PGP: Clau PGP associada a una clau principal, de manera que típicament la clau principal es fa servir per signar i les seves sub-claus per xifrar (les sub-claus estan definides només en OpenPGP, no en el sistema PGP original).

Bibliografia

1. Barrett, D. J.; Silverman, R. (2001). *SSH, The Secure Shell: The Definitive Guide.* Sebastopol: O'Reilly.

2. Stallings, W. (2003). *Cryptography and Network Security, Principles and Practice, 3rd ed.* Upper Saddle River: Prentice Hall.