Seguretat a la xarxa

Xavier Vilajosana Guillén

PID_00147703

Universitat Oberta de Catalunya
www.uoc.edu

Índex

Introducció

La seguretat a la Xarxa és un aspecte de cabdal importància. Cada dia apareixen noves aplicacions remotes o en xarxa, aquestes aplicacions emmagatzemen la informació en maquinari remot i requereixen intercanvis de dades constants entre els nostres ordinadors personals i els servidors remots. Tot aquest flux d'informació fa necessària l'existència de mecanismes per a evitar l'ús maliciós de la informació i per a assegurar la privadesa i protecció dels usuaris de la Xarxa.

A l'hora de protegir les xarxes de comunicacions, la **criptografia** és l'eina fonamental que ens permet evitar que algú intercepti, manipuli o falsifiqui les dades transmeses. Dedicarem una part d'aquest mòdul a introduir els conceptes de la criptografia necessaris per a entendre com s'aplica a la protecció de les comunicacions.

La finalitat bàsica de la criptografia és l'enviament d'informació secreta. Si apliquem una transformació, coneguda com a **xifratge**, a la informació que volem mantenir en privat, encara que un adversari aconsegueixi veure quines dades estem enviant li seran completament inintel·ligibles. Només el destinatari legítim serà capaç de fer la transformació inversa i recuperar les dades originals.

En el mòdul també estudiarem els sistemes tallafoc, que s'encarregaran de protegir l'accés a determinats punts de la Xarxa. Veurem que un sistema tallafoc actua com una barrera central per a reforçar el control d'accés als serveis que s'executen tant a l'interior com a l'exterior de la Xarxa. El tallafocs intenten prevenir els atacs de l'exterior contra les màquines internes d'una xarxa denegant peticions de connexió des de parts no autoritzades.

Veurem també que hi ha **xarxes privades virtuals** que es construeixen fent ús dels nodes d'una xarxa, però que mitjançant tècniques criptogràfiques i protocols segurs permeten aïllar els usuaris d'aquesta xarxa dels de la xarxa de comunicació.

Finalment farem una passada pels diferents protocols i aplicacions més rellevants per a assegurar les xarxes de comunicacions. Coneixerem les **galetes**, els continguts dinàmics i com aquests poden afectar la seguretat d'una lloc web. L'existència de protocols com **SSL/TLS** en el nivell de transport han permès que es pugui dotar de seguretat a les aplicacions en xarxa i als seus protocols. En aquest mòdul també coneixerem el funcionament d'aquests protocols i de passada com es poden fer transaccions segures a la Xarxa.

Objectius

L'estudi d'aquest mòdul us permetrà d'assolir els objectius següents:

1. Conèixer els principals mecanismes de seguretat a la xarxa.

2. Conèixer els fonaments de la criptografia i els certificats digitals.

3. Tenir una visió global dels elements de la xarxa que poden necessitar mecanismes de seguretat i conèixer els possibles punts febles d'una xarxa de computadors.

1. Tallafocs

Quan un equip és connectat a una xarxa de computadors, es poden identificar tres àrees de risc:

1) El nombre de punts que poden ser utilitzats com a origen per a dur a terme un atac contra qualsevol component de la xarxa s'incrementa. En un sistema aïllat (sense connexió), un requisit necessari per a ser atacat és forçosament l'existència d'un accés físic a l'equip. Però en el cas d'un sistema en xarxa, cadascun dels equips que pugui enviar informació cap a la víctima podrà ser utilitzat per un possible atacant.

Alguns serveis (com per exemple el Web i DNS) necessiten estar oberts públicament, de manera que qualsevol equip connectat a Internet podria ser l'origen d'una possible activitat maliciosa en contra. Això fa que sigui molt probable l'existència d'atacs regulars contra aquests sistemes.

2) La segona àrea de risc inclou l'expansió del perímetre físic del sistema telemàtic al qual l'equip acaba de ser connectat. Quan la màquina està aïllada, qualsevol activitat pot ser considerada com a interna a l'equip (i per tant, de confiança). El processador treballa amb les dades que troba a la memòria, que alhora han estat carregades des d'un mitjà d'emmagatzemament secundari. Aquestes dades estan realment ben protegides contra actes de modificació, eliminació, observació maliciosa, etc., en ser transferides entre diferents components de confiança.

Però aquesta mateixa premissa no és certa quan les dades són transferides a través d'una xarxa. La informació transmesa pel mitjà de comunicació és reenviada per dispositius que estan totalment fora del control del receptor. Aquesta informació podria ser llegida, emmagatzemada, modificada i més tard retransmesa cap al receptor legítim. Especialment en grans xarxes com Internet, no és trivial l'autenticació de l'origen que es presenta com l'emissor d'un missatge.

3) finalment, la tercera àrea de risc es deu a l'augment en el nombre de serveis d'autenticació (generalment un servei de *login-password*) que un sistema connectat a una xarxa ha d'oferir, respecte a un sistema aïllat. Aquests serveis no deixen de ser simples aplicacions (amb possibles errors de programació o de disseny) que protegeixen l'accés als recursos dels equips del sistema. Un error o vulnerabilitat en alguns d'aquests serveis pot representar el compromís del sistema al complet.

La prevenció d'atacs és la suma d'una sèrie de mecanismes de seguretat que proporcionen un primer nivell de defensa contra cert tipus d'atacs abans que aquests arribin al seu objectiu.

1.1. Sistemes tallafocs

Els sistemes tallafoc[1] són un mecanisme de control d'accés sobre la capa de xarxa. La idea bàsica és separar la nostra xarxa (en què els equips que intervenen són de confiança) dels equips de l'exterior (potencialment hostils).

(1)En anglès, *firewall*.

Un sistema tallafoc actua com una barrera central per a reforçar el control d'accés als serveis que s'executen tant a l'interior com a l'exterior de la xarxa. El tallafoc intentarà prevenir els atacs de l'exterior contra les màquines internes de la nostra xarxa denegant peticions de connexió des de parts no autoritzades.

Un tallafoc pot ser qualsevol dispositiu utilitzat com a mecanisme de control d'accés a nivell de xarxa per a protegir una xarxa en particular o un conjunt de xarxes. En la majoria dels casos, els sistemes tallafoc s'utilitzen per a prevenir accessos il·lícits a l'interior de la xarxa.

Un tallafoc és aquell sistema de xarxa expressament encarregat de separar xarxes de computadors, amb un control del trànsit existent entre aquestes. Aquest control consisteix, en última instància, a permetre o denegar el pas de la comunicació d'una xarxa a una altra mitjançant el control dels protocols Transmission Control Protocol/Internet Protocol (TCP/IP).

A l'hora d'instal·lar i configurar un sistema tallafoc a la nostra xarxa, s'ha de tenir present el següent:

1) Tot el trànsit que surt de l'interior cap a l'exterior de la xarxa per protegir, i viceversa, ha de passar pel tallafoc. Això es pot aconseguir bloquejant físicament tot l'accés a l'interior de la xarxa a través del sistema.

2) Només el trànsit autoritzat, definit en les polítiques de seguretat local del sistema, podrà traspassar el bloqueig.

3) El tallafoc mateix ha d'estar protegit contra possibles intrusions. Això implica l'ús d'un sistema operatiu de confiança amb suficients garanties de seguretat.

1.2. Construcció de sistemes tallafoc

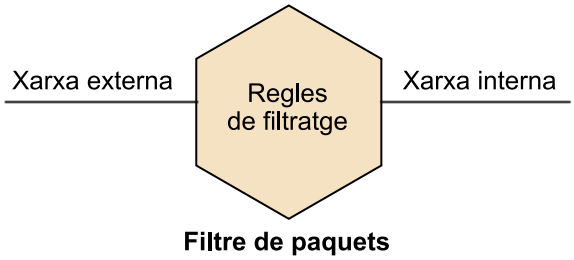
En el sentit més general, un sistema tallafoc consta de programari i maquinari. El programari pot ser de propietat, de prova, o gratuït. Per altra banda, el maquinari podrà ser qualsevol que pugui suportar aquest programari.

Actualment, dues de les tecnologies més utilitzades a l'hora de construir sistemes tallafoc són les següents:

- Encaminadors amb filtratge de paquets.
- Passarel·les a escala d'aplicació.

1.2.1. Encaminadors amb filtratge de paquets

Un encaminador amb filtratge de paquets és un dispositiu que encamina el trànsit TCP/IP (encaminador de TCP/IP) sobre la base d'una sèrie de regles de filtratge que decideixen quins paquets s'encaminen a través seu i quins són descartats.

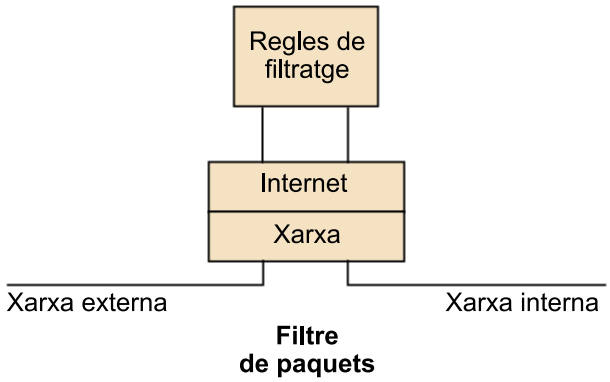


Les regles de filtratge s'encarreguen de determinar si a un paquet li està permès passar de la part interna de la xarxa a la part externa, i viceversa, i verifica el trànsit de dades legítim entre totes dues parts.

Els encaminadors amb filtratge de paquets, en treballar a escala de xarxa, poden acceptar o denegar paquets fixant-se en les capçaleres del protocol (tant IP, com UDP[2], TCP...), com poden ser:

[(2)] *UDP* és la sigla de *User Datagram Protocol.*

- Adreces d'origen i de destinació.
- Tipus de protocol i indicadors especials.
- Ports d'origen i de destinació o tipus de missatge (segons el protocol).
- Contingut dels paquets.
- Mida del paquet.



Les regles estan organitzades en conjunts de llistes amb una determinada política per defecte (denegar-ho tot, acceptar-ho tot...).

Cada paquet que arribi al dispositiu serà comparat amb les regles, començant pel principi de la llista fins que es trobi la primera coincidència. Si hi ha alguna coincidència, l'acció indicada en la regla serà activada (denegar, acceptar, redirigir...).

Per contra, si no és possible cap coincidència, serà consultada la política per defecte per a saber quina acció cal prendre (deixar passar el paquet, descartar-lo, redireccionar-lo...). Si es tracta, per exemple, d'una política de denegació per defecte, en cas de no haver-hi cap coincidència amb el paquet, aquest serà descartat.

Una política de denegació per defecte acostuma a ser més costosa de mantenir, ja que serà necessari que l'administrador indiqui explícitament tots els serveis que han de romandre oberts (la resta, per defecte, seran tots denegats).

En canvi, una política d'acceptació per defecte és més senzilla d'administrar, però incrementa el risc de permetre atacs contra la nostra xarxa, ja que requereix que l'administrador indiqui explícitament quins paquets cal descartar (la resta, per defecte, seran tots acceptats).

Avantatges i desavantatges dels encaminadors amb filtratge de paquets

La construcció d'un sistema tallafoc mitjançant un encaminador amb filtratge de paquets és realment econòmica, ja que generalment se sol fer amb maquinari ja disponible. A més, ofereix un alt rendiment en xarxes amb una càrrega de trànsit elevada.

Iptables

Un exemple d'encaminador amb filtre de paquets podria ser el programa **Iptables**, implementat com una part del programari d'encaminament del nucli Linux.

Addicionalment, aquesta tecnologia permet la implantació de la major part de les polítiques de seguretat necessàries.

Les polítiques de seguretat són el resultat de documentar les expectatives de seguretat, intentant plasmar en el món real els conceptes abstractes de seguretat. Poden ser definides de manera processal (plasmant de manera pràctica les idees o filosofies de l'empresa quant a seguretat) o de manera formal (utilitzant un model matemàtic que intenta abastar tots els estats i operacions possibles).

Tot i aquests avantatges, els encaminadors de xarxa amb filtratge de paquets poden presentar algunes deficiències, com per exemple:

- Molts dels encaminadors utilitzats poden ser vulnerables a atacs existents (tot i que la majoria dels distribuïdors tenen els paquets d'actualitzacions corresponents per a solucionar-ho). Per altra banda, no solen tenir capacitats de registre. Això provoca que a l'administrador li sigui difícil saber si l'encaminador mateix no està essent atacat.

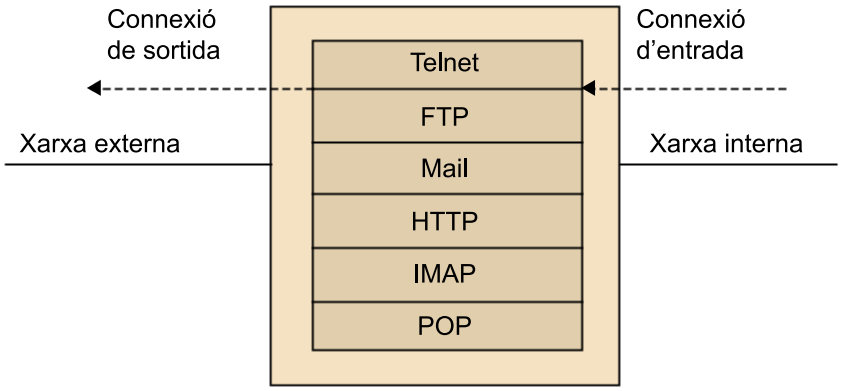
- La seva capacitat d'actuació es pot arribar a deteriorar a causa de la utilització d'un filtratge excessivament estricte, cosa que dificulta també el procés de gestió del dispositiu si aquest nombre de regles arriba a ser molt elevat.

- Les regles de filtratge poden arribar a ser molt complicades, i això provoca de vegades que les possibles distraccions en la configuració siguin aprofitades per un atacant per a fer una violació de la política de seguretat.

1.2.2. Passarel·la a escala d'aplicació

Una passarel·la a escala d'aplicació, coneguda també com a servidor intermediari[3], no encamina paquets a escala de xarxa sinó que actua com a retransmissor a escala d'aplicació. Els usuaris de la xarxa contactaran amb el servidor intermediari, que al seu torn estarà oferint un servei intermediari associat a una o més aplicacions determinades.

(3)En anglès, *proxy*.



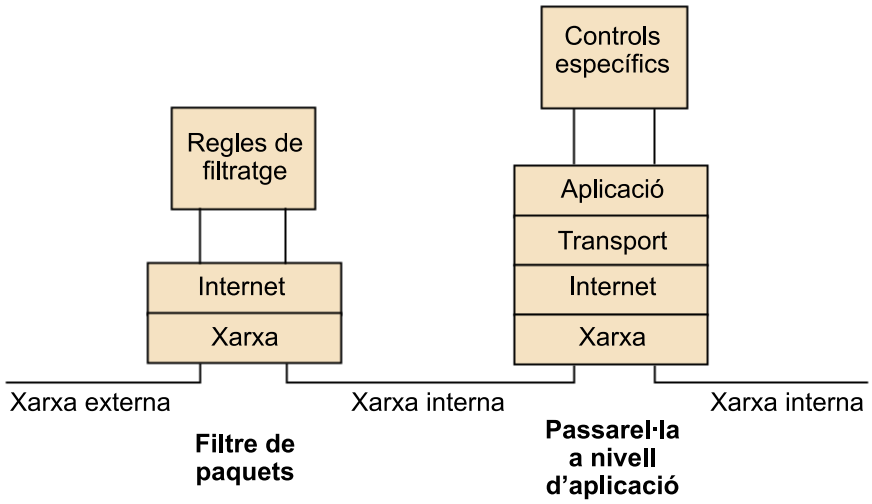
Passarel·la a nivell d'aplicació

El servei intermediari s'encarregarà de fer les connexions sol·licitades amb l'exterior i quan rebi una resposta s'encarregarà de retransmetre-la cap a l'equip que havia iniciat la connexió. Així, el servei intermediari executat en la passarel·la aplicarà les normes per a decidir si s'accepta o es rebutja una petició de connexió.

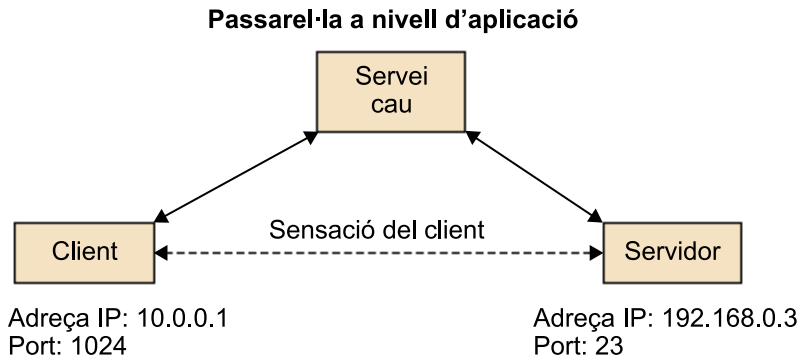
Una passarel·la separa completament l'interior de l'exterior de la xarxa en la capa d'enllaç, i ofereix únicament un conjunt de serveis a escala d'aplicació. Això permet l'autenticació dels usuaris que fan peticions de connexió i l'anàlisi de connexions a escala d'aplicació.

Aquestes dues característiques fan que les passarel·les ofereixin més seguretat respecte als filtres de paquets, i presenten un rang de possibilitats molt elevat. Per contra, la penalització introduïda per aquests dispositius és molt més gran. En cas d'una gran càrrega de trànsit en la xarxa, el rendiment es pot arribar a reduir dràsticament.

En la pràctica, les passarel·les i els dispositius de xarxa amb filtratge de paquets són complementaris. Així, aquests dos sistemes es poden combinar perquè proporcionin més seguretat i flexibilitat que si només se'n fes servir un, tal com es mostra a la figura següent.



Quan la passarel·la autentica el client, obre una connexió al servidor intermediari, i és el responsable de transmetre les dades que el client rep del servidor intermediari.



Aquest funcionament particular provoca que les passarel·les a escala d'aplicació presentin un rendiment inferior que els filtres de paquets. Per tal d'evitar-ho, els servidors intermediaris fan una còpia de les dades transmeses per a lliurar-les a un altre quan aquest les sol·liciti.

L'ús de les passarel·les proporciona diversos beneficis. D'entrada, les passarel·les permeten només aquells serveis per als quals hi ha un servidor intermediari habilitat. Així, si una passarel·la conté serveis intermediaris tan sols per als serveis HTTP i DNS, llavors només HTTP i DNS seran permesos en la xarxa interna. La resta de serveis seran completament rebutjats.

Un altre benefici de l'ús de passarel·les és que el protocol també pot ser filtrat, i es pot prohibir així l'ús de diferents subserveis dins d'un mateix servei permès. Per exemple, mitjançant una passarel·la que filtrés connexions FTP, seria possible prohibir únicament l'ús de l'ordre *put* d'FTP i deixar habilitada la resta d'ordres. Aquesta característica no és possible obtenir-la mitjançant l'ús de filtres de paquets.

Addicionalment, els servidors intermediaris també poden implantar el filtre de connexions per adreça IP de la mateixa manera que els filtres de paquets, ja que l'adreça IP està disponible en l'àmbit d'aplicació en el qual es farà el filtratge.

Tot i obtenir més control global sobre els serveis vigilats, les passarel·les també presenten algunes problemàtiques. Un dels primers inconvenients a destacar és la necessitat d'haver de configurar un servidor intermediari per a cada servei de la xarxa que cal vigilar (HTTP, DNS, Telnet, FTP...). A més, en el cas de protocols client-servidor, com per exemple Telnet, poden arribar a ser necessaris alguns passos addicionals per a connectar amb el punt final de la comunicació.

2. Xarxes privades virtuals

Els protocols segurs que hem vist fins ara permeten protegir les comunicacions, per exemple, d'una aplicació implementada com un procés client que s'executa en un ordinador i un procés servidor que s'executa en un altre ordinador. Si hi ha altres aplicacions que també necessiten una comunicació segura entre aquests dos ordinadors, o entre ordinadors situats en les mateixes xarxes locals, poden fer ús d'altres instàncies dels protocols segurs: noves associacions de seguretat IPsec, noves connexions SSL TLS, etc.

Una possibilitat alternativa és establir una xarxa privada virtual[4] entre aquests ordinadors o les xarxes locals on estan situats.

(4)En anglès, *virtual private network* (VPN).

2.1. Definició i tipus de xarxes privades virtuals

Una **xarxa privada virtual** (VPN) és una configuració que combina l'ús de dos tipus de tecnologies:

- Les tecnologies de seguretat que permeten la definició d'una **xarxa privada**, és a dir, un mitjà de comunicació confidencial que no pot ser interceptat per usuaris aliens a la xarxa.
- Les tecnologies d'encapçalament de protocols que permeten que, en comptes d'una connexió física dedicada per a la xarxa privada, es pugui utilitzar una infraestructura de xarxa pública, com és Internet, per a definir per sobre d'aquesta una **xarxa virtual**.

Per tant, una VPN és una xarxa lògica o virtual creada sobre una infraestructura compartida, però que proporciona els serveis de protecció necessaris per a una comunicació segura.

Depenent de la situació dels nodes que fan ús d'aquesta xarxa, podem considerar tres tipus de VPN:

1) Aquest és el cas habitual en què una empresa disposa de xarxes locals en diferents seus, geogràficament separades, en cadascuna de les quals hi ha una xarxa privada o **intranet**, d'accés restringit als seus empleats. Si interessa que des d'una de les seus es pugui accedir a les intranets d'altres seus, es pot usar una VPN per a interconnectar aquestes xarxes privades i formar una intranet única.

2) Quan un empleat de l'empresa vol accedir a la intranet des d'un ordinador remot, pot establir una VPN d'aquest tipus entre aquest ordinador i la intranet de l'empresa. L'ordinador remot pot ser, per exemple, un PC que l'empleat té a casa seva, o un ordinador portàtil des del qual es connecta a la xarxa de l'empresa quan està de viatge.

3) De vegades a una empresa li interessa compartir una part dels recursos de la seva intranet amb determinats usuaris externs, com per exemple proveïdors o clients de l'empresa. La xarxa que permet aquests accessos externs a una intranet s'anomena **extranet**, i la protecció s'assoleix mitjançant una VPN extranet.

2.2. Configuracions i protocols utilitzats en VPN

A cadascun dels tipus de VPN li sol correspondre una configuració específica.

- En les VPN entre intranets, la situació més habitual és que en cada intranet hi ha una **passarel·la VPN**, que connecta la xarxa local amb Internet. Aquesta passarel·la es comunica amb la de les altres intranets, aplicant el xifratge i les proteccions que siguin necessàries a les comunicacions de passarel·la a passarel·la a través d'Internet. Quan els paquets arriben a la intranet de destinació, la passarel·la corresponent els desxifra i els reenvia per la xarxa local fins a l'ordinador que els hagi de rebre. D'aquesta manera es fa servir la infraestructura pública d'Internet, en lloc d'establir línies privades dedicades, que representarien un cost més elevat. També s'aprofita la fiabilitat i redundància que proporciona Internet, ja que si una ruta no està disponible sempre es poden encaminar els paquets per un altre lloc, mentre que amb una línia dedicada la redundància implicaria un cost encara més gran.

- En les VPN d'accés remot, de vegades anomenades VPDN[5], un usuari es pot comunicar amb una intranet per mitjà d'un proveïdor d'accés a Internet, fent servir tecnologia convencional com per exemple un mòdem ADSL. L'ordinador de l'usuari ha de disposar de programari **client VPN** per a comunicar-se amb la passarel·la VPN de la intranet i dur a terme l'autenticació necessària, el xifratge, etc. Així també s'aprofita la infraestructura dels proveïdors d'Internet per a l'accés a la intranet, sense necessitat de trucades a un mòdem de l'empresa, que poden arribar a tenir un cost considerable.

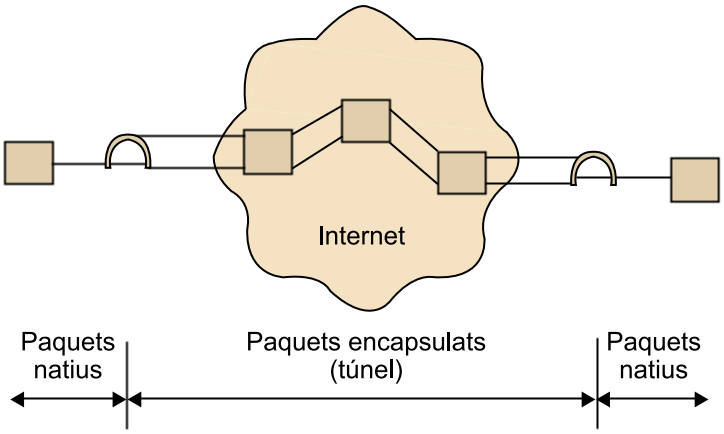
- El cas de les VPN extranet pot ser com el de les VPN entre intranets, en què la comunicació segura s'estableix entre passarel·les VPN, o bé com el de les VPN d'accés remot, en què un client VPN es comunica amb la passarel·la de la intranet. La diferència, però, és que en aquest cas normalment el control d'accés és més restrictiu per a permetre només l'accés als recursos autoritzats.

(5) *VPDN* és la sigla de *virtual private dial network*.

La definició d'una xarxa virtual es duu a terme mitjançant l'establiment de **túnels**, que permeten encapsular paquets de la xarxa virtual, amb els seus protocols, dins de paquets d'una altra xarxa, que normalment és Internet, amb el seu protocol, és a dir, IP.

Per a la comunicació entre les diferents intranets, o entre l'ordinador que accedeix remotament i la intranet, es poden utilitzar els protocols que siguin més convenients. Els paquets d'aquests protocols, per a poder-los fer arribar a la seva destinació a través d'Internet, es poden encapsular en datagrames IP, que a dins seu contindran els paquets originals. Quan arriben a la seva destinació, es desencapsulen aquests datagrames per a recuperar els paquets amb el format "natiu" del protocol corresponent.

Ús de túnels en una VPN



Hi ha diversos protocols que poden ser utilitzats per a establir els túnels, depenent del nivell de la comunicació en el qual es vulgui fer la protecció.

El protocol utilitzat en la gran majoria de configuracions VPN és IPsec en mode túnel, generalment amb ESP per a xifrar les dades, i opcionalment amb AH per a autenticar els paquets encapsulats. Les passarel·les VPN són, en aquest cas, passarel·les segures IPsec.

En el cas de les VPN d'accés remot o VPDN, hi ha la possibilitat d'encapsular trames PPP, que són les que transmet normalment un client VPN d'aquest tipus, sobre datagrames IP. Hi ha diverses opcions per a encapsular PPP (que al seu torn pot encapsular altres protocols de xarxa, com IPX, etc., o possiblement IP) sobre IP:

- El protocol PPTP[6] (RFC 2637) especifica una tècnica per a l'encapsulació de trames PPP però no afegeix serveis d'autenticació. Aquests serveis es poden fer amb els mateixos protocols que fa servir PPP, com Password Authentication Protocol (PAP) o Challenge Handshake Authentication Protocol (CHAP).

(6) *PPTP* és la sigla de *Point-to-Point Tunneling Protocol.*

- El protocol L2F[7] (RFC 2637) és semblant al PPTP però també pot treballar amb SLIP a més de PPP. Per a l'autenticació pot fer servir protocols auxiliars com Remote Authentication Dial-In User Service (RADIUS).

(7)*L2F* és la sigla de *Layer Two Forwarding*.

- El protocol L2TP[8] (RFC 2661) combina les funcionalitats que ofereixen PPTP i L2F.

(8)*L2TP* és la sigla de *Layer Two Tunneling Protocol*.

- El protocol SSH[9] ofereix la possibilitat de redirigir ports TCP sobre un canal segur, que podem considerar com un túnel a escala de transport. Des d'aquest punt de vista, també es podria considerar una connexió SSL TLS com un túnel a escala de transport que proporciona confidencialitat i autenticació. Habitualment, però, aquest últim tipus de túnel no serveix per a qualsevol tipus de trànsit sinó només per a dades TCP, i per tant, no es considera part integrant d'una VPN.

(9)*SSH* és la sigla de *Secure Shell*.

3. Introducció a la criptografia

Criptografia

Els termes *criptografia*, *criptologia*, etc., provenen de l'arrel grega *kryptós*, que vol dir 'amagat'.

Al llarg de la història, s'han dissenyat diferents tècniques per a ocultar el significat de la informació que no interessa que sigui coneguda per estranys. Algunes ja es feien servir en temps de l'antiga Grècia o de l'Imperi Romà: per exemple, s'atribueix a Juli Cèsar la invenció d'un codi per a enviar missatges xifrats que no poguessin ser interpretats per l'enemic.

La **criptografia** estudia, des d'un punt de vista matemàtic, els mètodes per a protegir la informació. D'altra banda, la **criptoanàlisi** estudia les possibles tècniques per a contrarestar els mètodes criptogràfics, i és de gran utilitat per a ajudar a fer-los més robustos i difícils d'atacar. El conjunt format per aquestes dues disciplines, criptografia i criptoanàlisi, s'anomena **criptologia**.

Ús del xifratge

El fet d'usar el xifratge parteix de l'assumpció que intentar evitar la intercepció de la informació per part d'un intrús (espia) és molt costós. Enviar missatges xifrats és més fàcil, i així, encara que un espia els pugui veure, no podrà interpretar la informació que contenen.

Quan la protecció que volem obtenir consisteix a garantir el secret de la informació, és a dir, la **confidencialitat**, utilitzem el mètode criptogràfic conegut com a **xifratge**.

Si M és el missatge que volem protegir o **text en clar**, xifrar-lo consisteix a aplicar-li un **algorisme de xifratge** f, que el transforma en un altre missatge que anomenarem **text xifrat**, C. Això ho podem expressar com:

$$C = f(M)$$

Per tal que aquest xifratge sigui útil, hi ha d'haver una altra transformació o **algorisme de desxifratge** f^{-1}, que permeti recuperar el missatge original a partir del text xifrat:

$$M - f^{-1}(C)$$

La xifra del Cèsar

La **xifra del Cèsar** consistia a substituir cada lletra del missatge per la que hi ha 3 posicions més endavant en l'alfabet (tornant a començar per la lletra A si arribem a la Z). Així, si apliquem aquest algorisme de xifratge al text en clar *alea jacta est* (i fent servir l'alfabet llatí actual, perquè en temps del Cèsar no hi havia lletres com la *w*), obtenim el text xifrat *dohd mdfwd hvw*. El desxifratge en aquest cas és ben senzill: només cal substituir cada lletra per la que hi ha 3 posicions abans en l'alfabet.

Un esquema com el de la xifra del Cèsar té l'inconvenient que si l'enemic descobreix quin és l'algorisme de xifratge (i a partir d'aquí dedueix l'algorisme invers), serà capaç d'interpretar tots els missatges xifrats que capturi. Llavors caldria instruir tots els "oficials de comunicacions" de l'exèrcit perquè aprenguin un nou algorisme, la qual cosa podria resultar complexa. En comptes d'això, el que es fa avui és utilitzar com a algorisme una funció amb un paràmetre anomenat **clau**.

Exemple d'ús d'una clau

L'algorisme de Juli Cèsar es pot generalitzar definint una clau k que indiqui quantes posicions cal avançar cada lletra en l'alfabet. La xifra del Cèsar, doncs, fa servir $k = 3$. Per al desxifratge es pot utilitzar el mateix algorisme però amb la clau invertida ($d \equiv e$, $x = -k$).

Llavors podem parlar d'una funció de xifratge e amb una **clau de xifratge** k, i una funció de desxifratge d amb una **clau de desxifratge** x:

$$C = e(k,M)$$
$$M = d(x,C) = d(x,e(k,M))$$

Així, una solució al problema de l'espia que s'assabenta de com cal desxifrar els missatges podria ser continuar fent servir el mateix algorisme, però amb una clau diferent.

Seguretat per ocultisme

Al llarg de la història hi ha hagut casos que han demostrat la perillositat de basar la protecció a mantenir els algorismes en secret (el que es coneix com a *seguretat per ocultisme*). Si l'algorisme és conegut per molts, és més fàcil que se'n detectin les febleses o vulnerabilitats i es puguin corregir ràpidament. Si no, un expert podria deduir l'algorisme per enginyeria inversa, i acabar-se descobrint que té punts febles per on es pot atacar, com va passar amb l'algorisme A5/1 de la telefonia mòbil GSM.

Una premissa fonamental en la criptografia moderna és l'anomenada **suposició de Kerckhoffs**, d'acord amb la qual els algorismes han de ser coneguts públicament i la seva seguretat només ha de dependre de la clau. En lloc d'intentar ocultar el funcionament dels algorismes, és molt més segur i efectiu mantenir en secret només les claus.

Un algorisme es considera **segur** si a un adversari li és impossible obtenir el text en clar M encara que conegui l'algorisme e i el text xifrat C. És a dir, és impossible desxifrar el missatge sense saber quina és la clau de desxifratge. La paraula *impossible*, però, l'hem de considerar amb diferents matisos. Un algorisme criptogràfic és **computacionalment segur** si, aplicant el millor mètode conegut, la quantitat de recursos necessaris (temps de càlcul, nombre de processadors, etc.) per a desxifrar els missatges sense conèixer la clau és molt més gran (uns quants ordres de magnitud) del que és a l'abast de ningú. En el límit, un algorisme és **incondicionalment segur** si no pot ser invertit ni amb recursos infinits. Els algorismes que es fan servir a la pràctica són (o intenten ser) computacionalment segurs.

L'acció d'intentar desxifrar missatges sense conèixer la clau de desxifratge s'anomena *atac*. Si l'atac té èxit, se sol dir col·loquialment que s'ha aconseguit "trencar" l'algorisme. Hi ha dues maneres de dur a terme un atac:

- Mitjançant la **criptoanàlisi**, és a dir, estudiant matemàticament la manera de deduir el text en clar a partir del text xifrat.

- Aplicant la **força bruta**, és a dir, provant un per un tots els valors possibles de la clau de desxifratge x fins a trobar-ne un que produeixi un text en clar amb sentit.

Atacs a la xifra del Cèsar

Continuant amb l'exemple de l'algorisme del Cèsar generalitzat (amb clau), un atac criptoanalític podria consistir a analitzar les propietats estadístiques del text xifrat. Les lletres xifrades que més es repeteixen probablement corresponen a vocals o a les consonants més freqüents, les combinacions més repetides de dues (o tres) lletres seguides probablement corresponen als dígrafs (o trígrafs) que típicament apareixen més vegades en un text, etc.

D'altra banda, l'atac per força bruta consistiria a intentar el desxifratge amb cadascun dels 25 possibles valors de la clau ($1 \leq x \leq 25$, si l'alfabet té 26 lletres) i mirar quin dóna un resultat intel·ligible. En aquest cas, la quantitat de recursos necessaris és tan modesta que fins i tot es pot fer l'atac a mà. Per tant, aquest xifratge seria un exemple d'algorisme insegur o feble.

A continuació veurem les característiques dels principals sistemes criptogràfics utilitzats en la protecció de les comunicacions. A partir d'ara, considerarem els missatges en clar, els missatges xifrats i les claus com a seqüències de bits.

3.1. Criptografia de clau simètrica

Els sistemes criptogràfics **de clau simètrica** es caracteritzen perquè la clau de desxifratge x és idèntica a la clau de xifratge k, o bé es pot deduir directament a partir d'aquesta.

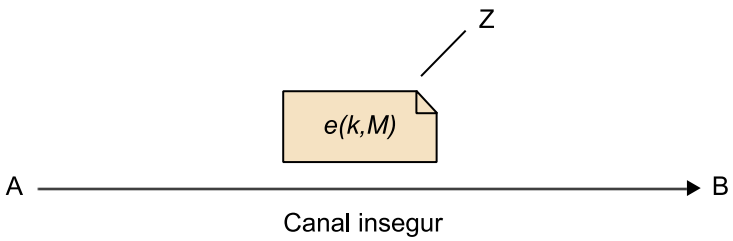
Per simplificar, suposarem que en aquest tipus de sistemes la clau de desxifratge és igual que la de xifratge: $x = k$ (si no, sempre podem considerar que en l'algorisme de desxifratge el primer pas és calcular la clau x a partir de la k). Per això aquestes tècniques criptogràfiques s'anomenen *de clau simètrica*, o de vegades també de **clau compartida**. Així, tenim:

$$C = e(k,M)$$
$$M = d(k,C) = d(k,e(k,M))$$

La seguretat del sistema està, doncs, a mantenir en secret la clau k. Quan els participants en una comunicació es volen intercanviar missatges confidencials, han d'escollir una clau secreta i fer-la servir per a xifrar els missatges. Lla-

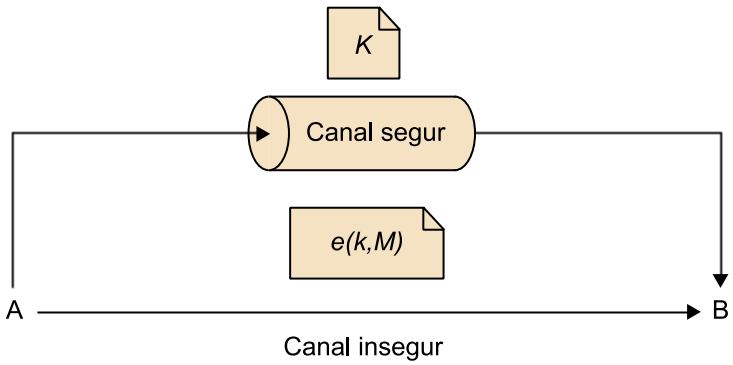
vors poden enviar aquests missatges per qualsevol canal de comunicació, amb la confiança que, encara que el canal sigui insegur i susceptible de ser inspeccionat per tercers, cap espia *Z* no serà capaç d'interpretar-los.

Missatge *M* xifrat amb una clau *k* entre A i B



Si el sistema és de clau compartida, és necessari que el valor de la clau secreta *k* que fan servir *A* i *B* sigui el mateix. Ara bé, com es poden assegurar que sigui així? És clar que no poden enviar la clau escollida a través del canal de comunicació de què disposen, perquè la hipòtesi inicial és que aquest canal és insegur i qualsevol podria descobrir la informació que s'hi transmet. Una possible solució és fer servir un canal a part, que pugui ser considerat suficientment segur:

Establiment d'un canal segur entre A i B



Canals segurs

Podrien ser exemples de "canals segurs": el correu tradicional (no electrònic) o un servei de missatgeria "física", una conversa telefònica, o cara a cara, etc.

Aquesta solució, però, té alguns inconvenients. D'una banda, se suposa que el canal segur no serà d'ús tan àgil com el canal insegur (si ho fos, seria molt millor enviar tots els missatges confidencials sense xifrar pel canal segur, i oblidar-nos del canal insegur!). Per tant, pot ser difícil anar canviant la clau. I d'altra banda, aquest esquema no és prou general: pot ser que hàgim d'enviar informació xifrada a algú amb qui no podem contactar de cap altra manera. Com veurem més endavant, aquests problemes relacionats amb l'**intercanvi de claus** se solucionen amb la criptografia asimètrica.

A continuació repassarem les característiques bàsiques dels principals algorismes criptogràfics de clau simètrica, els quals agruparem en dues categories: algorismes de flux i algorismes de bloc.

3.1.1. Algorismes de xifratge de flux

El funcionament d'una xifra de flux consisteix a combinar el text en clar *M* amb un text de xifratge *S* que s'obté a partir de la clau simètrica *k*. Per a desxifrar només cal fer l'operació inversa amb el text xifrat i el mateix text de xifratge *S*.

L'operació de combinació que s'utilitza típicament és la suma, i l'operació inversa, per tant, és la resta. Si el text està format per caràcters, aquest algorisme seria com una xifra del Cèsar en què la clau va canviant d'un caràcter a un altre. La clau que pertoca cada vegada està determinada pel text de xifratge *S* (anomenat *keystream* en anglès).

Suma i resta de bits

Quan treballem amb aritmètica binària o aritmètica de mòdul 2, es compleix:

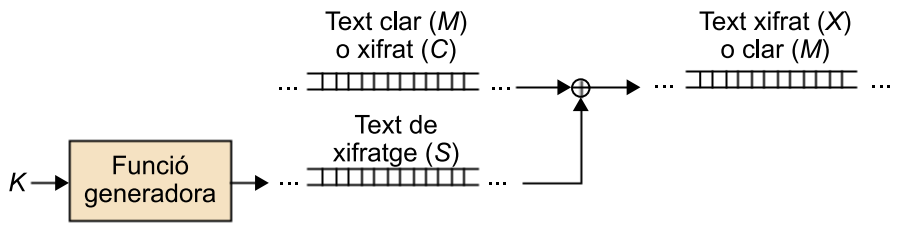
0 + 0 = 0	0 – 0 = 0	0 ⊕ 0 = 0
0 + 1 = 1	0 – 1 = 1	0 ⊕ 1 = 1
1 + 0 = 1	1 – 0 = 1	1 ⊕ 0 = 1
1 + 1 = 0	1 – 1 = 0	1 ⊕ 1 = 0

Si considerem el text format per bits, la suma i la resta són equivalents. En efecte, quan s'apliquen bit per bit, totes dues són idèntiques a l'operació lògica "O exclusiva", denotada amb l'operador XOR (*eXclusive OR*) o el símbol ⊕. Així doncs:

$$C = M \oplus S(k)$$
$$M = C \oplus S(k)$$

En els esquemes de xifratge de flux, el text en clar *M* pot ser de qualsevol longitud, i el text de xifratge *S* ha de ser almenys igual de llarg. De fet, no cal disposar del missatge sencer abans de començar a xifrar-lo o desxifrar-lo, ja que es pot implementar l'algorisme perquè treballi amb un "flux de dades" que va arribant (el text en clar o el text xifrat) i un altre "flux de dades" que es va generant a partir de la clau (el text de xifratge). D'aquí ve el nom d'aquest tipus d'algorismes. La figura següent il·lustra el mecanisme bàsic de la implementació.

Esquema del xifratge i desxifratge de flux



Hi ha diferents maneres d'obtenir el text de xifratge *S* en funció de la clau *k*:

- Si s'escull una seqüència *k* més curta que el missatge *M*, una possibilitat seria repetir-la cíclicament tantes vegades com calgui per a anar-la sumant al text en clar. L'inconvenient d'aquest mètode és que es pot trencar fàcilment, sobretot com més curta sigui la clau (en el cas mínim, l'algorisme seria equivalent a la xifra del Cèsar).

- En l'altre extrem, es podria fer directament $S(k) = k$. Això vol dir que la clau mateixa ha de ser tan llarga com el missatge per xifrar. Aquest és el principi de l'anomenada **xifra de Vernam**. Si *k* és una seqüència totalment aleatòria que no es repeteix cíclicament, estem davant d'un exemple de xifratge incondicionalment segur, tal com l'hem definit al començament d'aquest mòdul. Aquest xifratge s'anomena en anglès *one-time pad* ('quadern d'un sol ús'). El problema en aquest cas és que el receptor ha de disposar de la mateixa seqüència aleatòria per a poder fer el desxifratge, i si li ha d'arribar a través d'un canal segur, la pregunta és immediata: per què no enviar el missatge confidencial *M*, que és igual de llarg que la clau *k*, directament pel mateix canal segur? És evident, doncs, que aquest algorisme és molt segur però no és gaire pràctic en general.

  Ús del xifratge de Vernam

  Sovint les comunicacions entre els portaavions i els avions fan servir el xifratge de Vernam. En aquest cas, s'utilitza el fet que en un moment donat (abans d'enlairar-se) tant l'avió com el portaavions estan en el mateix lloc, i intercanviar-se, per exemple, un disc dur de 20 GB amb una seqüència aleatòria no és un problema. Posteriorment, quan l'avió s'enlaira pot establir una comunicació segura amb el portaavions utilitzant una xifra de Vernam amb la clau aleatòria que totes dues parts comparteixen.

- El que es fa en la pràctica és utilitzar funcions que generen **seqüències pseudoaleatòries** a partir d'una **llavor** (un nombre que actua com a paràmetre del generador), i el que s'intercanvia com a clau secreta *k* és només aquesta llavor.

  Exemple de funcions pseudoaleatòries

  Són exemples de funcions pseudoaleatòries les basades en registres de desplaçament realimentats (*feedback shift registers* o FSR). El valor inicial del registre és la llavor. Per a anar obtenint cada bit pseudoaleatori es desplacen tots els bits del registre una posició i s'agafa el que surt fora del registre. El bit que queda lliure a l'altre extrem s'omple amb un valor que és funció de la resta de bits.

Les seqüències pseudoaleatòries s'anomenen així perquè intenten semblar aleatòries però, és clar, són generades algorísmicament. A cada pas l'algorisme estarà en un determinat estat, que estarà determinat per les seves variables internes. Com que les variables seran finites, hi haurà un nombre màxim de possibles estats diferents. Això vol dir que al cap d'un cert període les dades generades es tornaran a repetir. Perquè l'algorisme sigui segur, interessa que el període de repetició sigui com més llarg millor (amb relació al missatge per

xifrar), a fi de dificultar la criptoanàlisi. Les seqüències pseudoaleatòries també han de tenir altres propietats estadístiques equivalents a les de les seqüències aleatòries pures.

Xifres síncrones i asíncrones

Si el text de xifratge *S* depèn exclusivament de la clau *k*, es diu que el xifratge és síncron. Aquest xifratge té el problema que si per algun error de transmissió es perden bits (o arriben repetits), el receptor es dessincronitzarà i sumarà bits del text *S* amb bits del text xifrat *C* que no toquen, amb la qual cosa el text desxifrat a partir de llavors serà incorrecte.

Això es pot evitar amb el xifratge asíncron (o "autosincronitzant"), en el qual el text *S* es calcula a partir de la clau *k* i el mateix text xifrat *C*. És a dir, en comptes de realimentar-se amb els seus bits d'estat, el generador es realimenta amb els *n* últims bits xifrats transmesos. D'aquesta manera, si es perden *m* bits consecutius en la comunicació, l'error afectarà com a màxim al desxifratge de $m + n$ bits del missatge original.

Exemples d'algorismes de xifratge de flux

Els algorismes de xifratge de flux actualment en ús tenen la propietat de ser poc costosos d'implementar. Les implementacions en maquinari són relativament simples i, per tant, eficients en el seu rendiment (en termes de bits xifrats per segon). Però també les implementacions en programari poden ser força eficients.

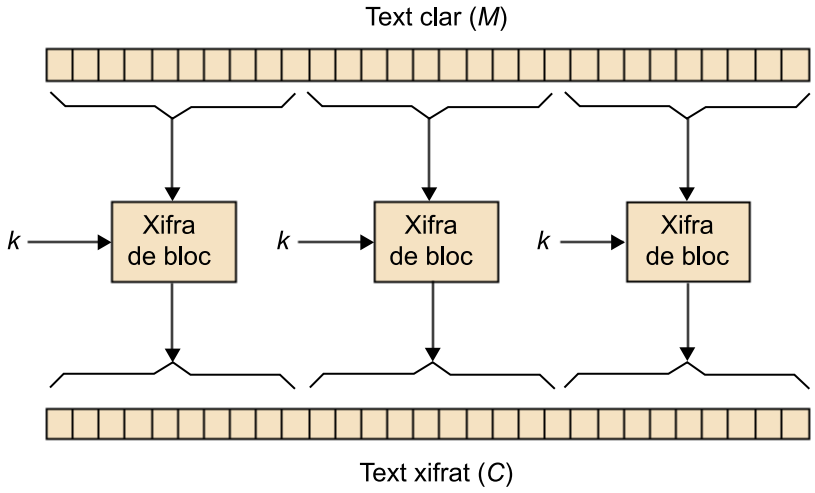
Les característiques del xifratge de flux el fan apropiat per a entorns en què calgui un rendiment alt i els recursos (capacitat de càlcul, consum d'energia) siguin limitats. Per això se solen utilitzar en comunicacions mòbils: xarxes locals sense fils, telefonia mòbil, etc.

3.1.2. Algorismes de xifratge de bloc

En una xifra de bloc, l'algorisme de xifratge o desxifratge s'aplica separadament a blocs d'entrada de longitud fixa *b*, i per a cadascun el resultat és un bloc de la mateixa longitud.

Per a xifrar un text en clar de *L* bits cal dividir-lo en blocs de *b* bits cadascun i xifrar aquests blocs un per un. Si *L* no és múltiple de *b*, es poden afegir bits addicionals fins a arribar a un nombre sencer de blocs, però llavors pot ser necessari indicar d'alguna manera quants bits hi havia realment en el missatge original. El desxifratge també s'ha de fer bloc per bloc. La figura següent mostra l'esquema bàsic del xifratge de bloc.

Esquema del xifratge de bloc



Molts dels algorismes de xifratge de bloc es basen en la combinació de dues operacions bàsiques: substitució i transposició.

La **substitució** consisteix a traduir cada grup de bits de l'entrada en un altre, d'acord amb una permutació determinada.

La xifra del Cèsar seria un exemple simple de substitució, en què cada grup de bits correspondria a una lletra. De fet, es tracta d'un cas particular de **substitució alfabètica**. En el cas més general, les lletres del text xifrat no ha d'estar necessàriament a una distància constant (la *k* de l'algorisme, tal com l'hem definit) de les lletres del text en clar.

Clau de la substitució alfabètica

És clar que la clau ha de ser una **permutació** de l'alfabet, és a dir, no hi pot haver lletres repetides ni faltar-ne cap. Si no, la transformació no seria invertible en general. La clau es pot expressar llavors com la seqüència correlativa de lletres que corresponen a la *A*, la *B*, la *C*, etc.

Exemple de substitució alfabètica

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

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

Text en clar: A L E A J A C T A E S T

Text xifrat: Q S T Q P Q E Z Q T L Z

La **transposició** consisteix a reordenar la informació del text en clar segons un patró determinat.

Exemple de transposició

Un exemple de transposició podria ser formar grups de 5 lletres, incloent-hi els espais en blanc, i reescriure cada grup (1,2,3,4,5) en l'ordre (3,1,5,4,2):

Text en clar: A L E A J A C T A E S T

Text xifrat: E A A L C J A T A S T E

La transposició per si sola no dificulta extraordinàriament la criptoanàlisi, però es pot combinar amb altres operacions per a afegir complexitat als algorismes de xifratge.

El **producte de xifres**, o combinació en cascada de diverses transformacions criptogràfiques, és una tècnica molt efectiva per a implementar algorismes força segurs de manera senzilla. Per exemple, molts algorismes de xifratge de bloc es basen en una sèrie d'iteracions de productes substitució-transposició.

Confusió i difusió

Dues propietats desitjables en un algorisme criptogràfic són la "confusió", que consisteix a amagar la relació entre la clau i les propietats estadístiques del text xifrat, i la "difusió", que propaga la redundància del text en clar al llarg del text xifrat perquè no sigui fàcilment reconeixible.

La confusió fa que canviant un sol bit de la clau canviïn molts bits del text xifrat, i la difusió fa que el canvi d'un sol bit del text en clar afecti també molts bits del text xifrat.

En un bucle de productes de xifres bàsiques, la substitució contribueix a la confusió, mentre que la transposició contribueix a la difusió. La combinació d'aquestes transformacions simples, repetides diverses vegades, fa que els canvis en l'entrada es propaguin per tota la sortida per un "efecte d'allau".

Exemples d'algorismes de xifratge de bloc

Durant molts anys l'algorisme de xifratge de bloc ha estat l'algorisme més estudiat i alhora el més utilitzat. Desenvolupat per IBM durant els anys setanta, va ser adoptat pel NBS nord-americà (nom que tenia llavors l'actual NIST) com a estàndard per al xifratge de dades l'any 1977. *NBS* és la sigla de National Bureau of Standards, i *NIST* és la sigla de National Institute of Standards and Technology.

L'algorisme admet una clau de 64 bits, però només 7 de cada 8 intervenen en el xifratge. Un possible ús dels bits de la clau DES que no influeixen en l'algorisme és com a bits de paritat. de manera que la longitud efectiva de la clau és de 56 bits. Els blocs de text als quals s'aplica el DES han de ser de 64 bits cadascun.

La part central de l'algorisme consisteix a dividir l'entrada en grups de bits, fer una substitució diferent sobre cada grup, i a continuació una transposició de tots els bits. Aquesta transformació es repeteix 16 vegades: a cada iteració, l'entrada és una transposició diferent dels bits de la clau sumada bit per bit (XOR) amb la sortida de la iteració anterior. Tal com està dissenyat l'algorisme, el desxifratge es fa igual que el xifratge però fent les transposicions de la clau en l'ordre invers (començant per l'última).

Tot i que al llarg dels anys l'algorisme DES s'ha demostrat molt resistent a la criptoanàlisi, el seu principal problema actualment és la vulnerabilitat als atacs de força bruta, a causa

de la longitud de la clau, de només 56 bits. Si en els anys setanta era molt costós fer una cerca entre les 2^{56} combinacions possibles, la tecnologia actual permet trencar l'algorisme en un temps com més va més curt.

Per això, el 1999 el NIST va canviar l'algorisme DES pel Triple DES com a estàndard oficial, mentre no estigués disponible el nou estàndard AES. El Triple DES, com el seu nom indica, consisteix a aplicar el DES tres vegades consecutives. Això es pot fer amb tres claus (k_1, k_2, k_3), o bé amb només dues de diferents (k_1, k_2, i una altra vegada k_1). La longitud total de la clau amb la segona opció és de 112 bits (dues claus de 56 bits), que avui ja es considera prou segura; la primera opció proporciona més seguretat, però a costa de fer servir una clau total de 168 bits (3 claus de 56 bits), que pot costar una mica més de gestionar i intercanviar.

Per a fer el sistema adaptable a l'estàndard antic, en el Triple DES s'aplica una seqüència xifratge-desxifratge-xifratge (E-D-E) en lloc de tres xifratges:

$$C = e(k_3, d(k_2, e(k_1,M)))$$
$$\text{o bé: mod 1 } emC = e(k_1, d(k_2, e(k_1,M)))$$

Així, fent $k_2 = k_1$ tenim un sistema equivalent al DES simple.

Com que l'estàndard DES es començava a quedar antiquat, a causa sobretot de la longitud tan curta de les claus, i el Triple DES no és gaire eficient quan s'implementa amb programari, l'any 1997 el NIST va convocar la comunitat criptològica a presentar propostes per a un nou estàndard, l'AES, que substituís el DES. Dels quinze algorismes candidats que es van acceptar, se'n van triar cinc com a finalistes, i l'octubre del 2000 es va donar a conèixer el guanyador: l'algorisme Rijndael, proposat pels criptògrafs belgues Joan Daemen i Vincent Rijmen.

El Rijndael pot treballar amb blocs de 128, 192 o 256 bits (encara que l'estàndard AES només en preveu de 128), i la longitud de la clau també pot ser 128, 192 o 256 bits. Depenent d'aquesta última longitud, el nombre d'iteracions de l'algorisme és 10, 12 o 14, respectivament. Cada iteració inclou una substitució fixa byte a byte, una transposició, una transformació consistent en desplaçaments de bits i XOR i una suma binària (XOR) amb bits obtinguts a partir de la clau.

Repetició del DES

L'operació de d'aplicar el xifratge DES amb una clau, i el resultat tornar-lo a xifrar amb una altra, no és equivalent a un sol xifratge DES (no hi ha cap clau única que doni el mateix resultat que donen les altres dues juntes). Si no fos així, la repetició del DES no seria més segura que el DES simple.

3.1.3. Ús dels algorismes de clau simètrica

Quan s'usa el xifratge simètric per a protegir les comunicacions, es pot escollir l'algorisme que sigui més apropiat a les necessitats de cada aplicació: normalment, com més seguretat menys velocitat de xifratge, i viceversa.

Un aspecte que cal tenir en compte és que si bé el xifratge pot fer que un atacant no descobreixi directament les dades transmeses, de vegades és possible que es pugui deduir informació indirectament. Per exemple, en un protocol que faci servir missatges amb una capçalera fixa, l'aparició de les mateixes dades xifrades diverses vegades en una transmissió pot indicar on comencen els missatges.

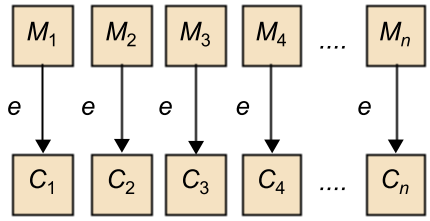
Això no passa amb les xifres de flux si el seu període és prou llarg, però en una xifra de bloc, si dos blocs de text en clar són iguals i es fa servir la mateixa clau, els blocs xifrats també seran iguals. Per a contrarestar aquesta propietat, es poden aplicar en diversos **modes d'operació** a les xifres de bloc:

1) Mode ECB (*electronic codebook*). És el més senzill, i consisteix simplement a dividir el text en blocs i xifrar cadascun dels blocs de manera independent. Aquest mode pateix del problema de donar blocs iguals quan en l'entrada hi ha blocs iguals.

Nom del mode ECB

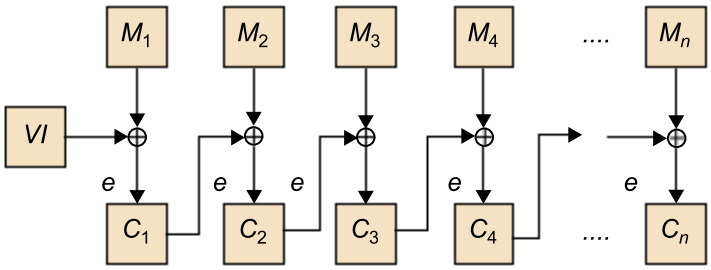
El nom *ECB* (*electronic codebook*) dóna la idea que es pot considerar com una simple substitució bloc per bloc, d'acord amb un codi o diccionari (amb moltes entrades, això sí) que està determinat per la clau.

Mode d'operació ECB



2) Mode CBC (*cipher block chaining*). En el mode CBC, abans de xifrar cada bloc de text en clar se li suma (bit per bit, amb XOR) el bloc xifrat anterior. Al primer bloc se li suma un **vector d'inicialització** (VI), que és un conjunt de bits aleatoris de la mateixa longitud que un bloc. Escollint vectors diferents cada vegada, encara que el text en clar sigui el mateix les dades xifrades seran diferents. El receptor ha de conèixer el valor del vector abans de començar a desxifrar, però no cal guardar aquest valor en secret, sinó que normalment es transmet com a encapçalament del text xifrat.

Mode d'operació CBC

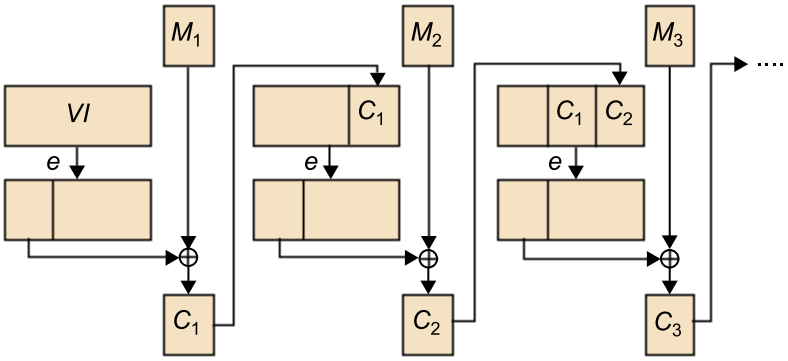


Mode CFB com a xifra de flux

És fàcil veure que el mode CFB (i també l'OFB) es pot considerar com una xifra de flux que fa servir com a funció generadora una xifra de bloc.

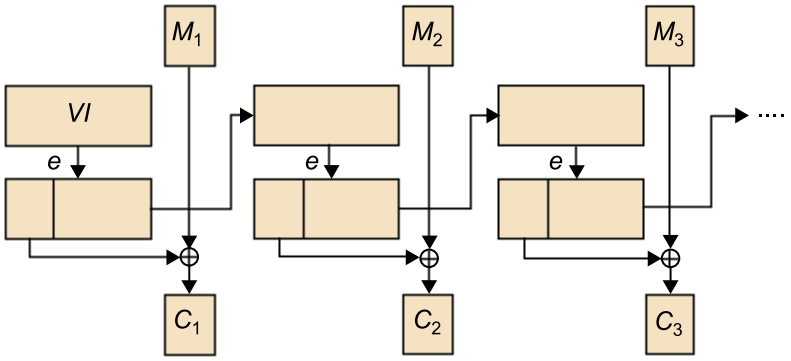
3) Mode CFB (*cipher feedback*). En el mode CFB, l'algorisme de xifratge no s'aplica directament al text en clar sinó a un vector auxiliar (inicialment igual que el VI). Del resultat del xifratge es prenen n bits que se sumen a n bits del text en clar per a obtenir n bits de text xifrat. Aquests bits xifrats s'usen alhora per a actualitzar el vector auxiliar. El nombre n de bits generats en cada iteració pot ser menor o igual que la longitud de bloc b. Prenent, per exemple, $n = 8$, tenim una xifra que genera un byte cada vegada sense haver d'esperar a tenir un bloc sencer per a poder-lo xifrar.

Mode d'operació CFB



4) Mode OFB (*output feedback*). El mode OFB és com el CFB però, en lloc d'actualitzar el vector auxiliar amb el text xifrat, s'actualitza amb el resultat obtingut de l'algorisme de xifratge. La propietat que diferencia aquest mode dels altres és que un error en la recepció d'un bit xifrat afecta només el desxifratge d'aquest bit.

Mode d'operació OFB



5) Altres modes. A partir dels modes anteriors es poden definir diverses variants. Per exemple, el mode CTR (*counter*) és com l'OFB però el vector auxiliar no es realimenta amb el xifratge anterior sinó que simplement és un comptador que es va incrementant.

Hi ha una altra tècnica per a evitar que amb textos d'entrada iguals s'obtinguin textos xifrats iguals, que és aplicable també als xifratges que no usen vector d'inicialització (incloent-hi les xifres de flux). Aquesta tècnica consisteix a modificar la clau secreta amb bits aleatoris abans d'usar-la en l'algorisme de xifratge (o en el de desxifratge). Com que aquests bits aleatoris serveixen per a donar un "sabor" diferent a la clau, se solen anomenar **bits de sal**. Igual que el vector d'inicialització, els bits de sal s'envien en clar abans del text xifrat.

3.1.4. Funcions resum segures

A part de xifrar dades, hi ha algorismes basats en tècniques criptogràfiques que s'usen per a garantir l'autenticitat dels missatges. Un tipus d'algorismes d'aquestes característiques són les anomenades **funcions resum segures**, també conegudes com funcions de **resum de missatge**[10].

[10]En anglès, *message digest*.

En general, podem dir que una funció resum ens permet obtenir una cadena de bits de longitud fixa, relativament curta, a partir d'un missatge de longitud arbitrària:

$$H = h(M)$$

Per a missatges M iguals, la funció h ha de donar resums H iguals. Però si dos missatges donen el mateix resum H no necessàriament han de ser iguals. Això és així perquè només hi ha un conjunt limitat de possibles valors H, ja que la seva longitud és fixa, i en canvi de missatges M n'hi pot haver molts més (si la longitud pot ser qualsevol, n'hi haurà infinits).

Per a poder-la aplicar en un sistema d'autenticació, la funció h ha de ser una funció resum segura.

Secret dels algorismes

Noteu que les funcions resum són conegudes, ja que tothom ha de poder calcular els resums de la mateixa manera.

S'entén que una funció resum és **segura** si compleix aquestes condicions:

1) És **unidireccional**, és a dir, si tenim $H = h(M)$ és computacionalment inviable trobar M a partir del resum H.

2) És **resistent a col·lisions**, és a dir, donat un missatge M qualsevol és computacionalment inviable trobar un missatge $M' \neq M$ tal que $h(M') = h(M)$.

Aquestes propietats permeten l'ús de les funcions resum segures per a donar un servei d'autenticitat basat en una clau secreta S compartida entre dues parts A i B. Aprofitant la unidireccionalitat, quan A vol enviar un missatge M a B pot preparar un altre missatge M_s, per exemple, concatenant l'original amb la clau: $M_s = (M,s)$. Llavors envia a B el missatge M i el resum del missatge M_s.

Per comprovar l'autenticitat del missatge rebut, B verifica que el resum correspon efectivament a M_s. Si és així, vol dir que l'ha generat algú que coneix la clau secreta s (que hauria de ser A), i també que ningú no ha modificat el missatge.

Una altra tècnica consistiria el calcular el resum del missatge M i xifrar-lo fent servir s com a clau de xifratge.

Autenticitat i confidencialitat

Xifrar només el resum, en lloc del missatge sencer, és més eficient perquè hi ha menys bits per xifrar. Això, és clar, suposant que només calgui autenticitat, i no confidencialitat. Si també interessa que el missatge sigui confidencial, llavors sí que cal xifrar-lo sencer.

Per a verificar l'autenticitat cal recuperar el resum enviat, desxifrant-lo amb la clau secreta *s*, i comparar-lo amb el resum del missatge *M*. Un atacant que volgués modificar el missatge sense conèixer la clau el podria intentar substituir per un altre que doni el mateix resum, amb la qual cosa *B* no detectaria la falsificació. Però si la funció de resum és resistent a col·lisions, això li hauria de ser impossible a l'atacant.

Per tal de dificultar els atacs contra les funcions de resum, d'una banda els algorismes han de definir una relació complexa entre els bits de l'entrada i cada bit de la sortida. D'altra banda, els atacs per força bruta es contraresten fent prou llarga la longitud del resum. Per exemple, els algorismes usats actualment generen resums de 128 o 160 bits. Això vol dir que un atacant podria haver de provar vora 2^{128} o 2^{160} missatges d'entrada per a trobar una col·lisió (és a dir, un missatge diferent que donés el mateix resum).

Però hi ha un altre tipus d'atac més avantatjós per a l'atacant, anomenat **atac de l'aniversari**. Un atac d'aquest tipus parteix del supòsit que l'atacant pot escollir el missatge que serà autenticat. La víctima llegeix el missatge i, si hi està d'acord, l'autentica amb la seva clau secreta. Però l'atacant ha presentat aquest missatge perquè n'ha trobat un altre que dóna el mateix resum, i per tant pot fer creure al destinatari que el missatge autèntic és aquest altre. I això es pot aconseguir fent una cerca per força bruta amb moltes menys operacions: vora 2^{64} o 2^{80}, si el resum és de 128 o 160 bits, respectivament.

Resistència forta a les col·lisions

La resistència dels algorismes de resum a les col·lisions, tal com l'hem definida, de vegades s'anomena *resistència feble*, mentre que la propietat de ser resistent a atacs de l'aniversari s'anomena *resistència forta*.

Paradoxa de l'aniversari

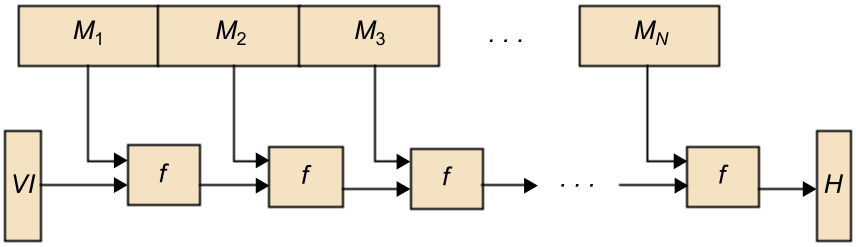
El nom d'aquest tipus d'atac ve d'un problema clàssic de probabilitats conegut com la "paradoxa de l'aniversari". El problema consisteix a trobar el nombre mínim de persones que hi ha d'haver en un grup per tal que la probabilitat que almenys dues d'elles celebrin el seu aniversari el mateix dia sigui superior al 50%. Una resposta intuïtiva pot ser que la solució és de vora 200, però aquest resultat no és correcte. Ho podria ser si es volgués obtenir el nombre de persones necessàries perquè hi hagi un 50% de probabilitat de coincidència amb una persona determinada. Si permetem que la coincidència sigui entre qualsevol parella de persones, la solució és un nombre molt menor: 23.

La conclusió és que si una funció de resum pot donar *N* valors diferents, perquè la probabilitat de trobar dos missatges amb el mateix resum sigui del 50% el nombre de missatges que cal provar és de vora: $\sqrt{N}$.

Exemples de funcions resum segures

L'esquema de la majoria de funcions resum usades actualment és semblant al dels algorismes de xifratge de bloc: el missatge d'entrada es divideix en blocs de la mateixa longitud, i a cadascun se li aplica una sèrie d'operacions juntament amb el resultat obtingut del bloc anterior. El resultat que queda després de processar l'últim bloc és el resum del missatge.

Esquema de les funcions de resum



L'objectiu d'aquests algorismes és que cada bit de la sortida depengui de tots els bits de l'entrada. Això s'aconsegueix amb diverses iteracions d'operacions que "barregen" els bits entre ells, de manera semblant a com la successió de transposicions en els xifratges de bloc provoca un "efecte allau" que garanteix la difusió dels bits.

Fins fa poc, l'algoritme de resum més usat era l'MD5 (Message Digest 5). Però com el resum que dóna és de només 128 bits, i a part s'han trobat altres maneres de generar col·lisions parcials en l'algoritme, actualment es recomana utilitzar algoritmes més segurs, com l'SHA-1 (Secure Hash Algorithm-1). L'algoritme SHA-1, publicat el 1995 en un estàndard del NIST (com a revisió d'un algoritme anterior anomenat simplement *SHA*), dóna resums de 160 bits. L'any 2002 el NIST va publicar variants d'aquest algoritme que generen resums de 256, 384 i 512 bits.

3.2. Criptografia de clau pública

Tractarem ara els conceptes fonamentals de la criptografia de clau pública.

3.2.1. Algorismes de clau pública

Com hem vist al subapartat anterior, un dels problemes de la criptografia de clau simètrica és el de la distribució de les claus. Aquest problema es pot solucionar si fem servir **algorismes de clau pública**, també anomenats **algorismes de clau asimètrica**.

En un algorisme criptogràfic de clau pública es fan servir claus diferents per al xifratge i el desxifratge. Una de les claus, la **clau pública**, es pot obtenir fàcilment a partir de l'altra, la **clau privada**, però en canvi és computacionalment molt difícil obtenir la clau privada a partir de la clau pública.

Els algorismes de clau pública típicament permeten xifrar amb la clau pública (k_{pub}) i desxifrar amb la clau privada (k_{pr}):

$$C = e(k_{pub},M)$$
$$M = d(k_{pr},C)$$

Però també hi pot haver algorismes que permetin xifrar amb la clau privada i desxifrar amb la pública (més endavant veurem com es pot utilitzar aquesta propietat):

$$C = e(k_{pr},M)$$

$$M = d(k_{pub}, C)$$

Adaptació dels problemes difícils

Si l'avanç de la tecnologia redueix el temps de resolució, es pot augmentar la longitud *n*, amb la qual cosa caldran unes quantes operacions més per al plantejament, però la complexitat de la solució creixerà exponencialment.

Els algorismes de clau pública es basen en problemes matemàtics "fàcils" de plantejar a partir de la solució, però "difícils" de resoldre. En aquest context, s'entén que un problema és fàcil si el temps per a resoldre'l, en funció de la longitud *n* de les dades, es pot expressar en forma polinòmica, com per exemple $n^2 + 2n$ (en teoria de la complexitat, es diu que aquests problemes són de la "classe P"). Si el temps de resolució creix més ràpidament, com per exemple amb 2^n, el problema es considera difícil. Així, es pot escollir un valor de *n* tal que el plantejament sigui viable però la resolució sigui computacionalment intractable.

Un exemple de problema fàcil de plantejar però difícil de resoldre és el dels logaritmes discrets. Si treballem amb aritmètica de mòdul *m*, és fàcil calcular aquesta expressió:

$$y = b^x \bmod m$$

El valor x s'anomena *logaritme discret de y en base b mòdul m*. Escollint convenientment *b* i *m*, pot ser difícil calcular el logaritme discret de qualsevol *y*. Una possibilitat és anar provant tots els valors de x: si *m* és un nombre de *n* bits, el temps per a trobar la solució augmenta proporcionalment amb 2^n. Hi ha altres mètodes més eficients per a calcular logaritmes discrets, però el millor algorisme conegut també necessita més temps del que es pot expressar polinòmicament.

Exemple d'operacions mòdul *m*

Per a obtenir 14^{11} mod 19 podem multiplicar 11 vegades el nombre 14, dividir el resultat entre 19 i prendre el residu de la divisió, que és igual a 13. Però també es pot aprofitar que l'exponent 11 és 1011 en binari ($11 = 1\cdot2^3 + 0\cdot2^2 + 1\cdot2^1 + 1\cdot2^0$), i per tant $14^{11} = 14^8 \cdot 14^2 \cdot 14^1$, per a obtenir el resultat amb menys multiplicacions:

$$14^1 = 14 \equiv 14 \pmod{19} \rightarrow 14$$
$$14^2 = 14^1 \times 14^1 \equiv 14 \times 14 \equiv 196 \equiv 6 \pmod{19} \rightarrow 6$$
$$14^4 14^2 \times 14^2 \equiv 6 \times 6 \equiv 36 \equiv 17 \pmod{19}$$
$$14^8 = 14^4 \times 14^4 \equiv 17 \times 17 \equiv 289 \equiv 4 \pmod{19} \rightarrow 4$$
$$9 - 9 \qquad 336 \equiv 13 \pmod{19}$$

Així, sabem que $\log_{14} 13 = 11 \pmod{19}$. Però si haguéssim d'obtenir el logaritme de qualsevol altre nombre *y* hauríem d'anar provant un per un els exponents fins a trobar-ne un que doni com a resultat *y*. I en comptes de tractar-se de nombres de 4 o 5 bits, si fossin nombres de més de 1.000 bits, el problema seria intractable.

Així, doncs, els algorismes de clau pública s'han de dissenyar de manera que sigui inviable calcular la clau privada a partir de la pública, i lògicament també ha de ser inviable invertir-los sense saber la clau privada, però el xifratge i el desxifratge s'han de poder fer en un temps relativament curt.

En la pràctica, els algorismes utilitzats permeten xifrar i desxifrar fàcilment, però tots són considerablement més lents. Per això, la criptografia de clau pública se sol emprar només en els problemes que la criptografia simètrica no pot resoldre: l'intercanvi de claus i l'autenticació sense repudi (signatures digitals).

Velocitat de la criptografia de clau pública

El xifratge i desxifratge amb algorismes de clau pública pot arribar a ser dos o tres ordres de magnitud més lent que amb els equivalents amb criptografia simètrica.

Els mecanismes d'**intercanvi de claus** permeten que dues parts es posin d'acord en les claus simètriques que faran servir per a comunicar-se, sense que un tercer que estigui escoltant el diàleg pugui deduir quines són aquestes claus.

Exemple de mecanismes d'intercanvi de claus

A pot escollir una clau simètrica *k*, xifrar-la amb la clau pública de *B*, i enviar el resultat a *B*. Llavors *B* desxifrarà amb la seva clau privada el valor rebut, i sabrà quina és la clau *k* que ha escollit *A*. La resta de la comunicació anirà xifrada amb un algorisme simètric (molt més ràpid), fent servir aquesta clau *k*. Els atacants, com que no coneixeran la clau privada de *B*, no podran deduir el valor de *k*.

L'**autenticació** basada en clau pública es pot dur a terme si l'algorisme permet utilitzar les claus a la inversa: la clau privada per a xifrar i la clau pública per a desxifrar.

Si *A* envia un missatge xifrat amb la seva clau privada, tothom podrà desxifrar-lo amb la clau pública d'*A*, i al mateix temps tothom sabrà que el missatge només el pot haver generat qui conegui la clau privada associada (que hauria de ser *A*). Aquesta és la base de les **signatures digitals**.

Exemples d'algorismes de clau pública

L'RSA és l'algorisme més utilitzat en la història de la criptografia de clau pública. El seu nom ve de les inicials dels qui el van dissenyar l'any 1977: Ronald Rivest, Adi Shamir i Leonard Adleman. La clau pública està formada per un nombre *n*, calculat com a producte de dos factors primers molt grans ($n = p \cdot q$), i un exponent *e*. La clau privada és un altre exponent *d* calculat a partir de *p*, *q* i *e*, de tal manera que el xifratge i el desxifratge es poden fer així:

$$\text{Xifratge: } C = M^e \bmod n$$

$$\text{Desxifratge: } M = C^d \bmod n$$

Com es pot veure, la clau pública i la privada són intercanviables: si s'usa qualsevol per a xifrar, caldrà usar l'altra per a desxifrar.

La fortalesa de l'algorisme RSA es basa, d'una banda, en la dificultat d'obtenir *M* a partir de *C* sense saber *d* (problema del logaritme discret), i d'altra banda, en la dificultat d'obtenir *p* i *q* (i per tant *d*) a partir de *n* (problema de la factorització de nombres grans, que és un altre dels problemes considerats difícils).

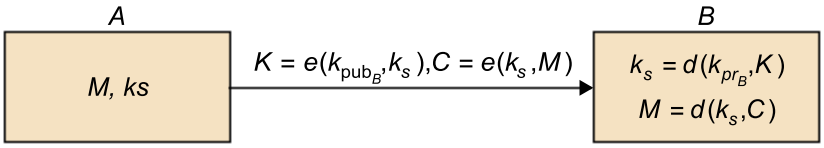
Valors usats en l'RSA

Actualment el problema de factoritzar nombres de 512 bits és molt complex, però abordable si es disposa de prou recursos. Per tant, es recomana fer servir claus públiques amb un valor n a partir de 1.024 bits. Com a exponent públic e típicament es fan servir valors senzills com 3 o 65.537 ($12^{16} + 1$) perquè fan més ràpid el xifratge.

3.2.2. Ús de la criptografia de clau pública

Hem vist abans que les principals aplicacions de la criptografia de clau pública són l'intercanvi de claus per a proporcionar confidencialitat, i la signatura digital per a proporcionar autenticitat i no-repudi.

El problema de la confidencialitat entre dues parts que només disposen d'un canal insegur per a comunicar-se es resol amb la criptografia de clau pública. Quan A vol enviar un missatge secret M a B, no cal xifrar tot el missatge amb un algorisme de clau pública (això podria resultar molt lent), sinó que s'escull una clau simètrica k_s, de vegades anomenada **clau de sessió** o **clau de transport**, i es xifra el missatge amb un algorisme simètric fent servir aquesta clau. L'únic que cal xifrar amb la clau pública de B (k_{pub_B}) és la clau de sessió. En recepció, B fa servir la seva clau privada (k_{pr_B}) per a recuperar la clau de sessió k_s, i llavors ja pot desxifrar el missatge xifrat.

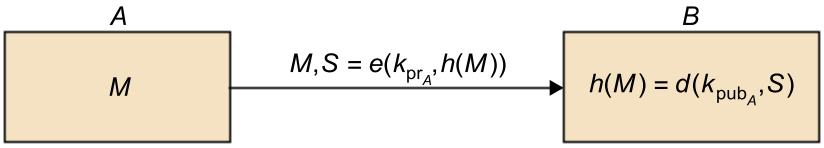


Ja que la clau de sessió és un missatge relativament curt (per exemple, si és una clau DES només tindrà 56 bits), un atacant podria intentar trencar el xifratge per força bruta, però no provant de desxifrar el missatge amb els possibles valors de la clau privada k_{pr_B}, sinó xifrant els possibles valors de la clau de sessió k_s amb la clau pública k_{pub_B}. En el cas d'una clau de sessió DES, independentment del nombre de bits de la clau pública, l'atacant només necessitaria un esforç de vora 2^{56} operacions.

Per a evitar aquest tipus d'atac, el que es xifra realment amb la clau pública no és directament el valor secret (en aquest cas k_s), sinó que a aquest valor se li afegeix una cadena més o menys llarga de bits aleatoris. Al receptor només li cal descartar aquests bits aleatoris del resultat que obtingui del desxifratge.

Una **signatura digital** és bàsicament un missatge xifrat amb la clau privada del signant. Però, per qüestió d'eficiència, el que es xifra no és directament el missatge per signar, sinó només el seu resum calculat amb una funció resum segura.

Quan *A* vulgui enviar un missatge signat, n'haurà d'obtenir el resum i xifrar-lo amb la clau privada k_{pr_A}. Per a verificar la signatura, el receptor l'ha de desxifrar amb la clau pública k_{pub_A} i comparar el resultat amb el resum del missatge: si són iguals, vol dir que el missatge l'ha generat *A* i ningú no l'ha modificat. Com que se suposa que la funció de resum és resistent a col·lisions, un atacant no podrà modificar el missatge sense que la signatura deixi de ser vàlida.



3.2.3. Infraestructura de clau pública

Tal com hem vist fins ara, la criptografia de clau pública permet resoldre el problema de l'intercanvi de claus, fent ús de les claus públiques dels participants. Ara, però, es planteja un altre problema: si algú ens diu que és *A* i la seva clau pública és k_{pub}, com podem saber que realment k_{pub} és la clau pública d'*A*? Perquè és perfectament possible que un atacant Z generi el seu parell de claus (k'_{pr},k'_{pub}) i ens digui "jo sóc A, i la meva clau pública és k'_{pub} ".

Una possible solució a aquest problema és que hi hagi algú de confiança que ens asseguri que efectivament les claus públiques pertanyen als seus suposats propietaris. Aquest algú pot signar un document que digui "la clau pública d'*A* és k_{pub_A}", i publicar-lo perquè tothom en tingui coneixement. Aquest tipus de document s'anomena **certificat de clau pública**, i és la base del que es coneix com a **infraestructura de clau pública (PKI)**.

4. Certificats digitals

Un certificat de clau pública consta de tres parts bàsiques:

1) Una identificació d'usuari, com per exemple el seu nom.
2) El valor de la clau pública d'aquest usuari.
3) La signatura de les dues parts anteriors.

Si l'autor de la signatura és algú en qui confiem, el certificat ens serveix com a garantia que la clau pública pertany a l'usuari que hi figura. Qui signa el certificat pot ser una autoritat que es responsabilitzi de verificar de manera fefaent l'autenticitat de les claus públiques. En aquest cas, es diu que el certificat ha estat generat per una **autoritat de certificació**[11].

[11]En anglès, *certification authority* (CA).

Hi pot haver diferents formats de certificats, però el més usat és el dels **certificats X.509**, especificat en la definició del **servei de directori X.500**.

El directori X.500

L'especificació del directori X.500 està publicada en la Sèrie de recomanacions ITU-T X.500, una de les quals és la Recomanació X.509.

El directori X.500 permet emmagatzemar i recuperar informació, expressada com a **atributs**, d'un conjunt d'**objectes**. Els objectes X.500 poden representar, per exemple, països, ciutats, o bé empreses, universitats (en general, organitzacions), departaments, facultats (en general, unitats organitzatives), persones, etc. Tots aquests objectes estan organitzats jeràrquicament en forma d'arbre (en cada node de l'arbre hi ha un objecte) i, dins de cada nivell, els objectes s'identifiquen mitjançant un **atribut distintiu**. Globalment, cada objecte s'identifica amb un **nom distintiu**[12], que no és més que la concatenació dels atributs distintius que hi ha entre l'arrel de l'arbre i l'objecte en qüestió.

[12]En anglès, *distinguished name* (DN).

El sistema de noms és, doncs, semblant al DNS d'Internet, amb la diferència que els components d'un nom DNS són simples cadenes de caràcters, i els d'un DN X.500 són atributs, cadascun amb un tipus i un valor.

Exemples de nom distintiu

Alguns exemples de tipus d'atributs que es poden usar com a atributs distintius en un DN són: *countryName* (habitualment denotat amb l'abreviatura *c*), *stateOrProvinceName* (*st*), *localityName* (*i*), *organizationName* (*o*), *organizationalUnitName* (*ou*), *commonName* (*cn*), *surname* (*sn*), etc. Un exemple de DN és "c=ES, st=Barcelona, l=Barcelona, o=Universitat Oberta de Catalunya, ou=SI, cn=cv.uoc.edu".

X.500 defineix un protocol d'accés al servei que permet fer operacions de consulta, i també operacions de modificació de la informació dels objectes. Aquestes últimes operacions, però, normalment només estan permeses a certs usuaris autoritzats, i per tant calen mecanismes d'autenticació dels usuaris, i aquests

mecanismes estan definits en la Recomanació X.509. Hi ha un mecanisme bàsic que fa servir contrasenyes, i un mecanisme més avançat que fa servir certificats.

4.1. Cadenes de certificats i jerarquies de certificació

Un certificat ens soluciona el problema de l'autenticitat de la clau pública si està signat per una CA en la qual confiem. Però què passa si ens comuniquem amb un usuari que té un certificat emès per una CA que no coneixem?

Hi ha la possibilitat que una CA tingui un certificat que garanteixi l'autenticitat de la seva clau pública, signat per una altra CA. Aquesta altra CA potser sí que la coneixem, o potser té al seu torn un certificat signat per una tercera CA, i així successivament. D'aquesta manera, es pot establir una jerarquia d'autoritats de certificació, en què les CA de nivell més baix emeten els certificats d'usuari, i les CA de cada nivell són certificades per una de nivell superior.

En un món ideal, hi podria haver una CA arrel que estigués a dalt de tot de la jerarquia i tingués la màxima autoritat. En la pràctica, aquesta CA arrel global no existeix, i segurament no existirà mai (seria difícil que fos acceptada per tothom). En lloc d'haver-hi un sol arbre que comprèn totes les CA del món, el que hi ha en la realitat són arbres independents (alguns possiblement amb una sola CA), cadascun amb la seva CA arrel.

Perquè puguem verificar l'autenticitat de la seva clau pública, un usuari ens pot enviar el seu certificat, més el certificat de la CA que l'ha emès, més el de la CA que ha emès aquest altre certificat, etc., fins a arribar al certificat d'una CA arrel. Això és el que s'anomena una **cadena de certificats**. Els certificats de les CA arrel tenen la propietat de ser autosignats, és a dir, estan signats per elles mateixes.

Un possible tipus d'extensió dels certificats X.509 és *basicConstraints*, i un camp del seu valor indica si el certificat és de CA (es pot usar la seva clau per a emetre altres certificats) o no. En cas que ho sigui, un altre subcamp (*pathLenConstraint*) permet indicar el nombre màxim de nivells de la jerarquia per sota d'aquesta CA.

4.2. Llistes de revocació de certificats

La Recomanació X.509, a més de definir el format dels certificats, també defineix una altra estructura anomenada **llista de revocació de certificats**[13].

(13)En anglès, *certificate revocation list* (CRL).

Una llista d'aquest tipus serveix per a publicar els certificats que han deixat de ser vàlids abans de la seva data de caducitat. Els motius poden ser diversos: s'ha emès un altre certificat que substitueix el revocat, ha canviat el DN del titular (per exemple, ha deixat de treballar en l'empresa on estava), li han robat la seva clau privada, etc.

Així, si ens volem assegurar completament de la validesa d'un certificat, no n'hi ha prou de verificar-ne la seva, sinó que haurem d'obtenir la versió actual de la CRL (publicada per la CA que va emetre el certificat) i comprovar que el certificat no aparegui en aquesta llista.

Una CA normalment actualitzarà la seva CRL de manera periòdica, afegint-hi cada vegada els certificats que hagin estat revocats. Quan arribi la data de caducitat que constava en el certificat, ja no caldrà tornar-lo a incloure a la CRL. Això permet que les CRL no creixin indefinidament.

5. Seguretat a la Xarxa

Fins al moment hem vist aspectes teòrics i pràctics relacionats amb l'arquitectura de la Xarxa i els protocols de comunicació. En aquest apartat es presenta una visió més aplicada dels conceptes de seguretat als nivells d'aplicació i transport de la Xarxa. Tractarem des d'aspectes per considerar en el desenvolupament d'aplicacions fins als protocols de seguretat més usats avui dia.

5.1. Galetes

Les galetes (*cookies*) són un mecanisme per a facilitar informació sobre la navegació efectuada. En definitiva, són una eina emprada pels servidors web per a emmagatzemar informació sobre els seus visitants.

> Una galeta és un fitxer de text que alguns servidors web graven en el nostre ordinador amb informació sobre la navegació que hem fet en les seves pàgines. Aquest fitxer es desa en el nostre disc dur, i serà retornat posteriorment al servidor quan aquest el sol·licita.

La caducitat d'aquestes galetes, moment en el qual deixen de ser actives, la determina el dissenyador de la web del servidor, i pot variar entre el temps de la sessió mateixa o fins a una data especificada. Les galetes no causen danys en el sistema, ja que només permeten reconèixer l'usuari quan es connecta a un lloc web, registrant les visites. En concret poden arribar a desar la contrasenya utilitzada per a accedir a aquella pàgina, dades personals de l'usuari, etc.; encara així, molts usuaris no volen aquest tipus de control no autoritzat i prefereixen l'anonimat, perquè aquest és gairebé l'únic problema real de les galetes. Destaquem també que hi ha virus que busquen aquesta informació "sensible" dins de les galetes.

Per mitjà de les opcions de configuració del navegador es pot habilitar o deshabilitar l'acceptació de galetes. També el podem configurar perquè ens avisi de l'arribada d'alguna galeta. Amb les últimes versions dels navegadors s'ha anat millorant el sistema de gestió de les galetes.

Tècnicament, les galetes són trossos de dades arbitràries definits pel servidor web i enviats al navegador. El navegador les torna sense modificar al servidor, i això reflecteix un estat (memòria d'esdeveniments anteriors) en les transaccions HTTP, que d'altra manera serien independents d'estat. Sense les galetes,

cada petició d'una pàgina web o un component d'una pàgina web seria un esdeveniment aïllat, sense cap relació amb la resta de peticions d'altres pàgines del mateix lloc.

Retornant una galeta al servidor web, el navegador li proporciona un mitjà per a relacionar la sol·licitud de la pàgina actual amb sol·licituds de pàgines anteriors. A més de ser definides per un servidor web, les galetes també poden ser definides per un *script* en un llenguatge com JavaScript, si està suportat i habilitat en el navegador web.

Navegadors i galetes

Les especificacions de galetes suggereixen que els navegadors han de suportar un nombre mínim de galetes o una quantitat mínima de memòria per a emmagatzemar-les. En concret, s'espera que un navegador sigui capaç d'emmagatzemar almenys 300 galetes de 4 kilobytes cada una i almenys 20 galetes per servidor o domini.

El servidor que estableix la galeta pot especificar una data d'esborrament, i la galeta serà esborrada en aquesta data. Un lloc de compres podria voler ajudar clients potencials recordant les coses que hi havia a la seva cistella de la compra, fins i tot si tanquen el navegador sense fer la compra i tornen més tard, per evitar que hagin de buscar els productes de nou. En aquest cas, el servidor crearia una galeta amb data d'esborrament segons el desig del dissenyador del lloc web. Si no es defineix una data d'esborrament, la galeta és esborrada quan l'usuari tanca el seu navegador. Per tant, definir una data d'esborrament és una manera de fer que la galeta sobrevisqui entre sessions. Per aquesta raó, a les galetes amb data esborrament s'anomenen *persistents*.

5.2. Continguts actius

En els darrers anys el Web ha evolucionat des del contingut estàtic que ens permetia crear HTML fins a continguts del tot dinàmics i que constitueixen ja avui dia la majoria de les pàgines web. Un contingut dinàmic és creat en temps d'execució per un conjunt de processos que s'executen tant en el client com en el servidor, i això depèn de la tecnologia emprada. En aquest sentit la seguretat del Web com més va requereix tècniques més especifiques per a assegurar els continguts i protegir les màquines de codis maliciosos. En aquest sentit introduirem algunes de les tecnologies usades per a la creació de contingut dinàmic.

5.2.1. Miniaplicacions

Una miniaplicació és un component de programari que corre en el context d'un altre programa, per exemple un navegador web. La miniaplicació ha de córrer en un contenidor, que el proporciona un programa amfitrió, mitjançant un connector, o en aplicacions com telèfons mòbils que suporten el model de

programació per miniaplicacions. Les miniaplicacions s'executen en el context del client, i per tant requereixen tècniques que assegurin que l'execució no danyarà la màquina client.

A diferència d'un programa, una miniaplicació no pot córrer de manera independent, ofereix informació gràfica i de vegades interactua amb l'usuari, típicament manca de sessió i té privilegis de seguretat restringits. Una miniaplicació normalment porta a terme una funció molt específica que manca d'ús independent.

Les miniaplicacions tenen restriccions de seguretat fortes a l'hora d'accedir a l'ordinador client que les està executant. Les polítiques de seguretat són, doncs, un factor molt important que cal tenir en compte quan es desenvolupa aquesta mena de programari.

5.2.2. Miniaplicacions de servidor/JSP

Les miniaplicacions de servidor són objectes Java executats per un servidor d'aplicacions i que responen a invocacions HTTP, servint pàgines dinàmiques el contingut de les quals generalment és un fitxer HTML generat dinàmicament.

Una miniaplicació de servidor és capaç de rebre una invocació i generar una resposta en funció de les dades de la invocació, de l'estat del sistema i de les dades a què pugui accedir. Les miniaplicacions de servidor poden estar empaquetades dintre d'un fitxer de format WAR[14], com a aplicació web, dintre d'un contenidor.

(14)En anglès, *web application archive*.

L'execució d'una miniaplicació de servidor es fa dintre d'un o més processos del servidor d'aplicacions, de manera que es genera un nou flux. No generar un nou procés (com acostumen a fer els CGI) implica un estalvi de recursos que es tradueix en un rendiment millor del sistema però comporta altres problemes de concurrència.

Les miniaplicacions de servidor poden ser objectes java precompilats o JSP compilats en temps d'execució (o en un altre moment després de l'arrencada del servidor d'aplicacions).

Per altra banda, JavaServer Pages (JSP) és una tecnologia que permet als desenvolupadors de pàgines web generar dinàmicament respostes a peticions HTTP. La tecnologia permet que codi Java i certes accions predefinides siguin incrustades en un context estàtic, és a dir, dins de l'HTML mateix. La sintaxi de JSP incorpora etiquetes XML addicionals, anomenades *accions de JSP*, per a usar-les per a invocar altres funcions.

5.2.3. CGI

La interfície d'entrada comuna[15] és una important tecnologia del World Wide Web que permet a un client (navegador web) sol·licitar dades d'un programa executat en un servidor web. CGI especifica un estàndard per a transferir dades entre el client i el programa. És un mecanisme de comunicació entre el servidor web i una aplicació externa, el resultat final de l'execució de la qual són objectes MIME. Les aplicacions que s'executen en el servidor reben el nom de *CGI*.

(15)En anglès, Common Gateway Interface (CGI).

Les aplicacions CGI van ser una de les primeres maneres pràctiques de crear contingut dinàmic per a les pàgines web. En una aplicació CGI, el servidor web passa les sol·licituds del client a un programa extern. Aquest programa pot estar escrit en qualsevol llenguatge que suporti el servidor, encara que per raons de portabilitat se solen usar llenguatges de *script*. La sortida d'aquest programa és enviada al client en lloc de l'arxiu estàtic tradicional. CGI ha fet possible la implementació de funcions noves i variades en les pàgines web, de tal manera que aquesta interfície ràpidament es va tornar un estàndard, i va ser implementada en tot tipus de servidors web.

Els CGI són programes que podrien introduir problemes de seguretat si han estat programats malintencionadament. Per aquest motiu molts dels proveïdors d'hostatge de pàgines web no permeten l'execució d'aquest tipus de serveis web.

5.2.4. ASP/PHP

El servidor de pàgines actives[16] correspon a la tecnologia introduïda per Microsoft en l'any 1996, i permet l'ús de diferents scripts i components ActiveX al costat del tradicional HTML per a mostrar pàgines generades dinàmicament. Es basa en el VBScript, però hi ha diversos llenguatges de programació que es poden utilitzar, com per exemple Perl, Javascript, etc. ASP és una tecnologia dinàmica que funciona en el costat del servidor, cosa que significa que quan l'usuari sol·licita un document ASP, les instruccions de programació dintre de l'*script* s'executen en el servidor per a enviar al navegador únicament el codi HTML resultant. A l'usuari només se li envia el que sol·licita, i no pot accedir a cap altre servei del servidor. Així, una de les seves aplicacions més importants és la de l'accés a bases de dades.

(16)En anglès, Active Server Pages (ASP).

Hi ha altres llenguatges amb funcionalitats semblants. Entre aquests destacarem PHP (Hypertext Preprocessor), que és un llenguatge de programació creat per a desenvolupar aplicacions web, molt similar als llenguatges de programació C o C++. El seu codi s'insereix en les pàgines HTML, i s'executa en el servidor.

Adreça web recomanada

Per a saber més sobre PHP podeu consultar l'adreça següent: http://www.php.net.

Cal destacar l'aparició de molts entorns de treball que faciliten el desenvolupament d'aplicacions web tot integrant les funcionalitats de generació de contingut i l'emmagatzematge. Per exemple, Cake, Rails, etc.

5.2.5. RIA

RIA (*rich Internet applications*) és un acrònim que engloba una gran multitud de termes que defineixen un seguit d'aplicacions en què els continguts són dinàmics i que són carregats en el temps d'inicialització. L'aplicació només fa consultes al servidor per a obtenir dades de les bases de dades, mentre que les eines (reproductors de vídeo, processament d'imatges) ja estan carregades en la banda del client. Això fa que es puguin oferir funcionalitats molt més riques i entorns multimèdia més aconseguits.

Es pot dir que les RIA són la nova generació de les aplicacions i és una tendència ja imposada per empreses com Macromedia, Magic Software, Sun o Microsoft, que es troben desenvolupant recursos per a fer d'aquest tipus d'aplicacions una realitat. Aquestes aplicacions estan basades en plataformes J2EE o .NET, amb una interfície Flash, Java Swing, Java FX o Google Web Toolkit, i utilitzen una arquitectura client/servidor asíncrona, segura i escalable, juntament amb una interfície d'usuari web. Entre els beneficis principals de les aplicacions RIA tenim una millora important en l'experiència visual, que fan de l'ús de l'aplicació quelcom molt senzill, ofereixen millores en la connectivitat i el desplegament instantani de l'aplicació, agilitzant-ne l'accés i garanteixen la desvinculació de la capa de presentació, és a dir, l'accés a l'aplicació des de qualsevol computador en qualsevol lloc del món. Un entorn de treball que permet desenvolupar de manera senzilla aquesta mena d'aplicacions és el Google Web Toolkit.

5.3. Protocols de seguretat

Els protocols defineixen les regles i les normes que utilitzen els ordinadors per a comunicar-se amb la Xarxa. Internet és un canal insegur per a enviar informació. Quan s'ha d'enviar un missatge per Internet, aquest passa per nombrosos nodes intermedis abans d'arribar a la seva destinació. Algun d'aquests nodes intermedis podria interceptar, llegir, destruir o modificar la informació enviada.

Moltes vegades durant el procés de disseny d'una aplicació la seguretat és un aspecte que es deixa de banda i s'acaba afegint amb posterioritat. En molts casos aquests afegits no han previst tots els possibles problemes, i per tant fan que l'aplicació pugui ser vulnerable.

En un principi, en el model OSI[17] de comunicacions, es va decidir posar tot el que es refereix a seguretat en el nivell de presentació. D'aquesta manera, es podria oferir aquest servei a totes les aplicacions. No obstant això, el model OSI no va tenir èxit, i per tant aquesta solució no ha estat usada.

(17)*OSI* és la sigla d'*Open Systems Interconnection*; en català, 'interconnexió de sistemes oberts'.

Si analitzem el model Internet, basat en el protocol TCP/IP, veurem que no hi ha cap nivell de presentació, però podem arribar a establir una certa equivalència entre tots dos models que ens serà útil per a assenyalar els sistemes de seguretat que podem establir en cadascun dels nivells. En el nivell d'aplicació podem implementar aquells serveis de seguretat que siguin específics de cada aplicació.

Comparació del model OSI i l'arquitectura TCP/IP

El servei de seguretat passa a través d'aplicacions intermèdies. Per exemple: Pretty Good Privacy (PGP), Secure Electronic Transactions (SET), S/MIME, etc. Per sota del nivell d'aplicació, podem arribar a intercalar serveis de seguretat com Secure Sockets Layer (SSL), Transport Layer Secure (TLS), etc.

Entre els nivells TCP i IP, podríem establir mesures de seguretat transparents a les aplicacions. Exemple: IPSEC. Fins i tot per sota del nivell IP, podem arribar a xifrar les capçaleres IP, però té l'inconvenient que si no es fa correctament el desxifratge, les dades no arribaran a la seva destinació.

5.3.1. PGP

Hi ha eines basades en algoritmes criptogràfics que permeten protegir la informació que s'intercanvia a través de la Xarxa. Pretty Good Privacy (PGP) és un protocol que permet xifrar fitxers i missatges de correu electrònic, de manera que només hi puguin accedir els usuaris que determinem. De manera addicional, PGP permet generar una firma digital dels fitxers i missatges, cosa que en permet garantir la integritat. Mitjançant aquest programa podem portar la gestió de claus, xifrar i firmar digitalment missatges i fitxers, l'esborrament segur de fitxers, etc.

Resumint, els seus dos usos més comuns són els següents:

- Garantir que un fitxer informàtic (missatge de correu electrònic, fitxer de text, full de càlcul, imatge, etc.) ha estat creat per qui diu que n'és el creador (firma digital).

- Impedir que persones sense autorització llegeixin un fitxer informàtic (xifratge).

No obstant això, encara que té altres funcions ens centrarem en les dues formes més comunes d'ús de PGP, la firma digital i el xifratge. Per a usar qualsevol d'aquestes funcions, l'usuari de PGP ha de crear una parella de claus (una de pública i una altra de privada), que és la base sobre la qual se sosté la criptografia de claus públiques, basada en els algoritmes de clau asimètrica. Crear la parella de claus és molt senzill, ja que és l'aplicació PGP la que s'encarrega de pràcticament tot.

Perquè es pugui utilitzar PGP els interlocutors hauran de tenir instal·lat el programa. En instal·lar-se el programa es generen unes claus. Una de les claus que formen la parella és la clau privada, que cal protegir sempre d'accessos no autoritzats, ja que tota la nostra seguretat es basa en el fet que ningú més no hi tingui accés. Per a reduir-ne la vulnerabilitat, la clau privada està protegida per una contrasenya complexa en forma de frase que és molt més segura que una contrasenya típica de menys de deu caràcters. La seva utilitat és desxifrar els missatges que ens siguin enviats de manera segura. Al contrari, la clau pública cal distribuir-la a totes aquelles persones amb les quals vulguem mantenir comunicacions segures. Podria semblar que distribuir lliurement la nostra clau pública és una manera de baixar les nostres defenses, però no és així: la clau pública està xifrada i, a més, la seva única utilitat és xifrar els missatges i fitxers

que els altres ens vulguin enviar. Amb la clau és impossible desxifrar els missatges o fitxers que algú ens hagi enviat; tampoc no és possible fer-se passar per nosaltres firmant un fitxer. Per a això caldria disposar de la clau privada.

PGP es pot integrar amb els programes de correu electrònic més usuals per a fer que xifrar i firmar missatges no consisteixi en res més que fer clic sobre un botó. En xifrar un missatge, ens demanarà que indiquem a qui l'enviarem, per a triar així la clau pública correcta amb què cal xifrar-lo. Per a firmar, en canvi, ens demanarà que teclegem la contrasenya de la nostra clau privada, amb la qual cosa evitem també que algú que usi el nostre ordinador en la nostra absència es pugui fer passar per nosaltres.

Per a xifrar les dades s'empra un algoritme de clau simètrica, la clau del qual es xifra amb un algoritme de clau asimètrica. D'aquesta manera es combinen les millors propietats de tots dos: la seguretat d'un algoritme asimètric (en què clau pública i privada són diferents) amb la rapidesa i robustesa d'un algoritme simètric (la clau del qual és única i, per tant, vulnerable). Un tercer algoritme s'empra per a firmar documents: s'extreu un conjunt de bits del missatge (resum) amb una funció resum i es xifra amb la clau privada de l'emissor. Així, pel seu mode de funcionament, PGP (i altres programes similars) consta dels tres subsistemes: xifratge del document amb clau asimètrica, xifratge amb clau simètrica i firma digital.

Quan es vol enviar un correu o un fitxer xifrat de *B* a *A*, PGP el xifra usant un sistema simètric (generalment IDEA o DES), usant una clau aleatòria –clau DES–, que posteriorment es xifra (per exemple amb RSA, amb la clau pública de *A*: KAp). S'envia el document xifrat amb la clau aleatòria, i aquesta clau xifrada amb la clau RSA privada del destinatari.

Quan *A* rep el correu i el vol desxifrar, el seu programa PGP primer desxifra la clau simètrica amb la seva clau privada RSA (Kas), i després desxifra el document usant la clau desxifrada –clau DES.

5.4. SSL

Secure Socket Layer és un sistema de protocols de caràcter general dissenyat el 1994 per l'empresa Nestcape Communications Corporation, basat en l'aplicació conjunta de criptografia simètrica, criptografia asimètrica (de clau pública), certificats digitals i firmes digitals, per a oferir connexions segures mitjançant Internet. Aquest grup de protocols comprèn:

- El protocol de transport Secure Sockets Layer (SSL), desenvolupat per Netscape Communications al començament dels anys noranta. La primera versió d'aquest protocol àmpliament difosa i implementada va ser la 2.0. Poc després Netscape va publicar la versió 3.0, amb molts canvis respecte a l'anterior, que avui ja gairebé no s'utilitza.

- L'especificació Transport Layer Security (TLS), elaborada per l'Internet Engineering Task Force (IETF). La versió 1.0 del protocol TLS està publicada en el document RFC 2246. És pràcticament equivalent a SSL 3.0 amb algunes petites diferències, per la qual cosa en certs contextos es considera TLS 1.0 com si fos el protocol "SSL 3.1".

- El protocol Wireless Transport Layer Security (WTLS), pertanyent a la família de protocols Wireless Application Protocol (WAP) per a l'accés a la Xarxa des de dispositius mòbils. La majoria dels protocols WAP són adaptacions dels ja existents a les característiques de les comunicacions sense fils, i en particular el WTLS està basat en el TLS 1.0. Les diferències se centren principalment en aspectes relatius a l'ús eficient de l'amplada de banda i de la capacitat de càlcul dels dispositius, que pot ser limitada.

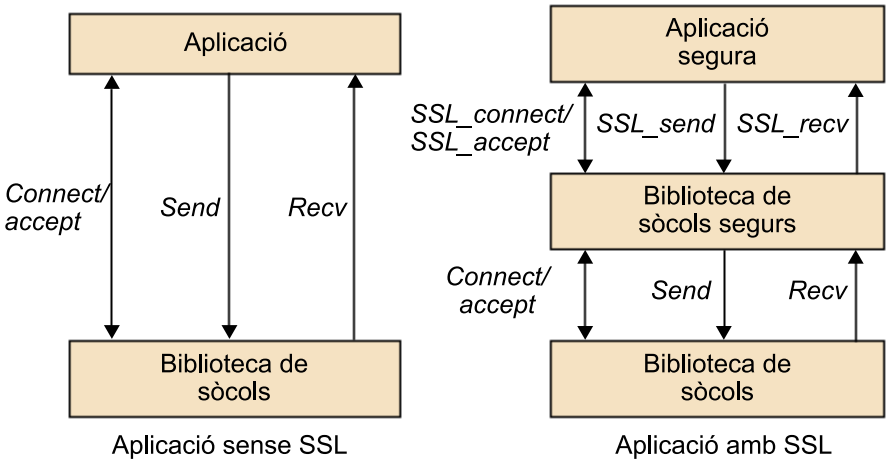
5.4.1. Característiques del protocol SSL/TLS

L'objectiu inicial de disseny del protocol SSL va ser protegir les connexions entre clients i servidors web amb el protocol HTTP. Aquesta protecció havia de permetre al client assegurar-se que s'havia connectat al servidor autèntic, i enviar-li dades confidencials, com per exemple un número de targeta de crèdit, amb la confiança que ningú més que el servidor seria capaç de veure aquestes dades.

Les funcions de seguretat, però, no es van implementar directament en la protecció d'aplicació HTTP, sinó que es va optar per introduir-les en el nivell de transport. Així hi podria haver moltes més aplicacions que fessin ús d'aquesta funcionalitat.

Per a això es va desenvolupar una interfície d'accés als serveis del nivell de transport basada en la interfície estàndard dels sòcols. En aquesta nova interfície, funcions com *connect*, *accept*, *send* o *recv* van ser substituïdes per altres equivalents però que utilitzaven un protocol de transport segur: *SSL_connect*, *SSL_accept*, *SSL_send*, *SSL_recv*, etc. El disseny es va fer de tal manera que qualsevol aplicació que utilitzés TCP per mitjà de les crides dels sòcols podia fer ús del protocol SSL només canviant aquestes crides. D'aquí ve el nom del protocol.

Disseny del protocol SSL



Datagrames en WTLS

Una característica distintiva del WTLS és que no solament permet protegir connexions TCP, com fan SSL i TLS, sinó que també defineix un mecanisme de protecció per a les comunicacions en mode datagrama, usat en diverses aplicacions mòbils.

Els serveis de seguretat que proporcionen els protocols SSL/TLS són:

- **Confidencialitat**. El flux normal d'informació en una connexió SSL/TLS consisteix a intercanviar paquets amb dades xifrades mitjançant claus simètriques (per motius d'eficiència i rapidesa). En l'inici de cada sessió, client i servidor es posen d'acord en quines claus utilitzaran per a xifrar les dades. Sempre es fan servir dues claus diferents: una per als paquets enviats del client al servidor, i l'altra per als paquets enviats en sentit contrari. Per evitar que un intrús que estigui escoltant el diàleg inicial pugui saber quines són les claus acordades, se segueix un mecanisme segur d'intercanvi de claus, basat en criptografia de clau pública. L'algorisme concret per a aquest intercanvi també es negocia durant l'establiment de la connexió.

- **Autenticació d'entitat**. Amb un protocol basat en signatures digitals el client pot confirmar la identitat del servidor al qual s'ha connectat. Per a validar les signatures el client necessita conèixer la clau pública del servidor, i això normalment es fa per mitjà de certificats digitals. SSL/TLS també preveu l'autenticació del client davant el servidor. Aquesta possibilitat, però, no s'usa tan sovint perquè moltes vegades, en comptes d'autenticar automàticament el client a escala de transport, les mateixes aplicacions utilitzen el seu mètode d'autenticació propi.

Autenticació de client

Un exemple d'autenticació de client a escala d'aplicació són les contrasenyes que poden introduir els usuaris en formularis HTML. Si l'aplicació fa servir aquest mètode, al servidor ja no li cal autenticar el client a escala de transport.

- **Autenticació de missatge**. Cada paquet enviat en una connexió SSL/TLS, a més d'estar xifrat, pot incorporar un codi MAC perquè el destinatari comprovi que ningú no ha modificat el paquet. Les claus secretes per al càlcul

dels codis MAC (una per a cada sentit) també s'acorden de manera segura en el diàleg inicial.

A més, els protocols SSL/TLS estan dissenyats amb aquests criteris addicionals:

- **Eficiència**. Dues de les característiques de SSL/TLS, la definició de sessions i la compressió de les dades, permeten millorar l'eficiència de la comunicació.
  - Si el client demana dues o més connexions simultànies o molt seguides, en lloc de repetir l'autenticació i l'intercanvi de claus (operacions computacionalment costoses perquè hi intervenen algorismes de clau pública), hi ha l'opció de reutilitzar els paràmetres prèviament acordats. Si es fa ús d'aquesta opció, es considera que la nova connexió pertany a la mateixa **sessió** que l'anterior. En l'establiment de cada connexió s'especifica un **identificador de sessió**, que permet saber si la connexió comença una sessió nova o és continuació d'una altra.

  - SSL/TLS preveu la negociació d'algorismes de **compressió** per a les dades intercanviades, per compensar el trànsit addicional que introdueix la seguretat. Ni SSL 3.0 ni TLS 1.0, però, especifiquen cap algorisme concret de compressió.

Connexions consecutives o simultànies

Una situació típica en què s'usa SSL/TLS és la d'un navegador web que accedeix a una pàgina HTML que conté imatges: amb HTTP "no persistent" (l'únic mode definit en HTTP 1.0), això requereix una primera connexió per a la pàgina i a continuació tantes connexions com imatges hi hagi. Si les connexions pertanyen a la mateixa sessió SSL/TLS, només cal fer la negociació una vegada.

- **Extensibilitat**. Al començament de cada sessió, client i servidor negocien els algorismes que faran servir per a l'intercanvi de claus, l'autenticació i el xifratge (a més de l'algorisme de compressió). Les especificacions dels protocols inclouen unes combinacions predefinides d'algorismes criptogràfics, però deixen oberta la possibilitat d'afegir-hi nous algorismes si se'n descobreixen altres que siguin més eficients o més segurs.

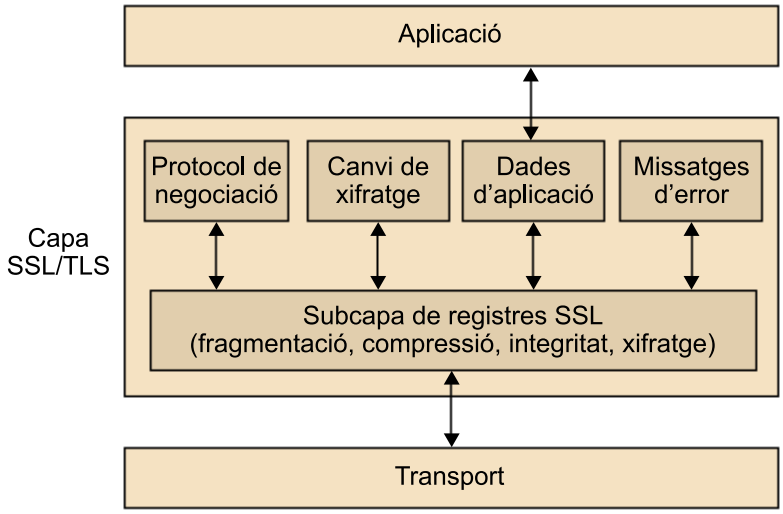
5.4.2. El transport segur SSL/TLS

La capa de transport segur que proporciona SSL/TLS es pot considerar dividida en dues subcapes.

- La subcapa superior s'encarrega bàsicament de negociar els paràmetres de seguretat i de transferir les dades de l'aplicació. Tant les dades de negociació com les d'aplicació s'intercanvien en **missatges**.

- En la subcapa inferior, aquests missatges són estructurats en **registres** als quals s'aplica, segons correspongui, la compressió, l'autenticació i el xifratge.

Estructura de la capa SSL/TLS



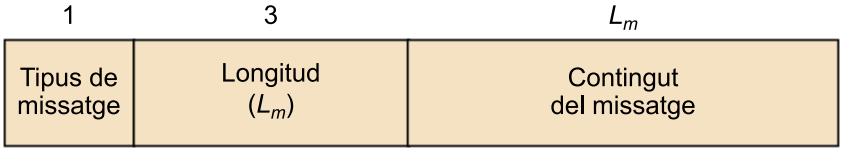
El **protocol de registres SSL/TLS** és el que permet que les dades protegides siguin convenientment codificades per l'emissor i interpretades pel receptor. Els paràmetres necessaris per a la protecció, com ara els algorismes i les claus, s'estableixen de manera segura a l'inici de la connexió mitjançant el **protocol de negociació SSL/TLS**.

El protocol de negociació SSL/TLS

El protocol de negociació SSL/TLS, també anomenat **protocol d'encaixada de mans** (*handshake protocol*), té per finalitat autenticar el client o el servidor, i acordar els algorismes i claus que faran servir d'una manera segura, és a dir, garantint la confidencialitat i la integritat de la negociació.

Com tots els missatges SSL/TLS, els missatges del protocol de negociació s'inclouen dins del camp de dades dels registres SSL/TLS per a ser transmesos al destinatari. L'estructura d'un missatge de negociació és la que es mostra a la figura següent.

Format dels missatges de negociació SSL/TLS



El contingut del missatge tindrà uns determinats camps depenent del tipus de missatge de negociació de què es tracti. En total hi ha 10 tipus diferents, que veurem a continuació en l'ordre en què s'han d'enviar.

1) Petició de salutació (*hello request*). Quan s'estableix una connexió, el servidor normalment espera que el client iniciï la negociació. Alternativament, pot optar per enviar un missatge *hello request* per a indicar al client que està preparat per a començar. Si durant la sessió el servidor vol iniciar una renegociació, també ho pot indicar al client enviant-li un missatge d'aquest tipus.

2) Salutació de client (*client hello*). El client envia un missatge *client hello* a l'inici de la connexió o com a resposta a un *hello request*. Aquest missatge conté la informació següent:

- La versió del protocol que el client vol fer servir.
- Una cadena de 32 bytes aleatoris[18].
- Opcionalment, l'identificador d'una sessió anterior, si el client vol tornar a utilitzar els paràmetres que s'hi van acordar.
- La llista de les combinacions d'algorismes criptogràfics que el client ofereix utilitzar, per ordre de preferència. Cada combinació inclou l'algorisme de xifratge, l'algorisme de MAC i el mètode d'intercanvi de claus.
- La llista dels algorismes de compressió oferts, per ordre de preferència (com a mínim n'hi ha d'haver un, encara que sigui l'algorisme nul).

Algorismes criptogràfics previstos en SSL/TLS

SSL/TLS preveu els algorismes criptogràfics següents:

- Xifratge: RC4, DES, Triple DES, RC2, IDEA i FORTEZZA (aquest últim només en SSL 3.0).
- MAC: MD5 i SHA-1.
- Intercanvi de claus: RSA, Diffie-Hellman i FORTEZZA KEA (aquest últim només en SSL 3.0).

Si només interessa autenticar la connexió, sense confidencialitat, també es pot usar l'algorisme de xifratge nul.

3) Salutació de servidor (*server hello*). Com a resposta, el servidor envia un missatge *server hello*, que conté aquesta informació:

- La versió del protocol que es farà servir en la connexió. La versió serà igual que la que va enviar el client, o inferior si aquesta no és suportada pel servidor.
- Una altra cadena de 32 bytes aleatoris.

(18) Dels 32 bytes aleatoris que s'envien en els missatges de salutació, els 4 primers han de ser una marca de temps, amb precisió de segons.

Algorismes de compressió

L'únic algorisme de compressió previst en SSL/TLS és l'algorisme nul, és a dir, cap compressió.

- L'identificador de la sessió actual. Si el client n'ha enviat un i el servidor vol reprendre la sessió corresponent, ha de respondre amb el mateix identificador. Si el servidor no vol reprendre la sessió (o no pot perquè ja no guarda la informació necessària), l'identificador enviat serà diferent. Opcionalment, el servidor pot no enviar cap identificador per a indicar que la sessió actual mai no podrà ser represa.

- La combinació d'algorismes criptogràfics escollida pel servidor entre la llista dels enviats pel client. Si es reprèn una sessió anterior, aquesta combinació ha de ser la mateixa que es va fer servir llavors. L'algorisme de compressió escollit pel servidor, o el que es va fer servir en la sessió que es reprèn.

Si s'ha decidit continuar una sessió anterior, client i servidor ja poden començar a utilitzar els algorismes i claus prèviament acordats i se salten els missatges que vénen a continuació, passant directament als de finalització de la negociació (missatges *finished*).

4) Certificat de servidor (*certificate*) o intercanvi de claus de servidor (*server key exchange*). Si el servidor es pot autenticar davant el client, que és el cas més habitual, envia el missatge *certificate*. Aquest missatge normalment contindrà el certificat X.509 del servidor, o una cadena de certificats. Si el servidor no té certificat, o s'ha acordat un mètode d'intercanvi de claus que no en fa servir, ha d'enviar un missatge *server key exchange*, que conté els paràmetres necessaris per al mètode que cal seguir.

5) Petició de certificat (*certificate request*)

- Tipus de certificats: en SSL/TLS estan previstos els certificats de clau pública RSA, DSA o FORTEZZA KEA (aquest últim tipus només en SSL 3.0). En cas que s'hagi de fer també l'autenticació del client, el servidor li envia un missatge *certificate request*. Aquest missatge conté una llista dels possibles tipus de certificat que el servidor pot admetre, per ordre de preferència, i una llista dels DN de les autoritats de certificació que el servidor reconeix.

6) Fi de salutació de servidor (*server hello done*). Per acabar aquesta primera fase del diàleg, el servidor envia un missatge *server hello done*.

7) Certificat de client (*certificate*)

- Client sense certificat: si el client rep una petició de certificat però no en té cap d'apropiat, en SSL 3.0 ha d'enviar un missatge d'avís, però en TLS 1.0 ha d'enviar un missatge *certificate* buit. En qualsevol cas, el servidor pot respondre amb un error fatal, o bé continuar sense autenticar el client. Un cop el servidor ha enviat els seus missatges inicials, el client ja sap com ha de continuar el protocol de negociació. En primer lloc, si el servidor

li ha demanat un certificat i el client en té algun de les característiques sol·licitades, l'envia en un missatge *certificate*.

8) Intercanvi de claus de client (*client key exchange*). El client envia un missatge *client key exchange*, el contingut del qual depèn del mètode d'intercanvi de claus acordat. En cas de seguir el mètode RSA, en aquest missatge hi ha una cadena de 48 bytes que es farà servir com a **secret premestre**, xifrada amb la clau pública del servidor.

Un possible atac contra la negociació és modificar els missatges perquè les dues parts acordin utilitzar el protocol SSL 2.0, que és més vulnerable. Per a evitar aquest atac, en els dos primers bytes del secret premestre hi ha d'haver el número de versió que es va enviar en el missatge *client hello*. Llavors, client i servidor calculen l'anomenat **secret mestre**, que és una altra cadena de 48 bytes. Per a fer aquest càlcul, s'apliquen funcions resum al secret premestre i a les cadenes aleatòries que es van enviar en els missatges de salutació. A partir del secret mestre i les cadenes aleatòries, s'obtenen:

- Les dues claus per al xifratge simètric de les dades (una per a cada sentit: de client a servidor i de servidor a client).

- Les dues claus MAC (també una per a cada sentit).

- Els dos vectors d'inicialització per al xifratge, si s'utilitza un algorisme de bloc.

9) Verificació de certificat (*certificate verify*). Si el client ha enviat un certificat en resposta a un missatge *certificate request*, ja es pot autenticar demostrant que posseeix la clau privada corresponent mitjançant un missatge *certificate verify*. Aquest missatge conté una signatura, generada amb la clau privada del client, d'una cadena de bytes obtinguda a partir de la concatenació de tots els missatges de negociació intercanviats fins ara, des del *client hello* fins al *client key exchange*.

10) Finalització (*finished*). A partir d'aquest punt ja es poden fer servir els algorismes criptogràfics negociats. Cada part envia a l'altra una notificació de canvi de xifratge seguida d'un missatge *finished*. La notificació de canvi de xifratge serveix per a indicar que el missatge següent serà el primer enviat amb els nous algorismes i claus.

El missatge *finished* segueix immediatament la notificació de canvi de xifratge. El seu contingut s'obté aplicant funcions resum al secret mestre i a la concatenació de tots els missatges de negociació intercanviats, des del *client hello* fins a l'anterior a aquest (incloent-hi el missatge *finished* de l'altra part, si ja l'ha

enviat). Normalment serà el client el primer a enviar el missatge *finished*, però en el cas de reprendre una sessió anterior, serà el servidor qui l'enviarà primer, just després del *server hello*.

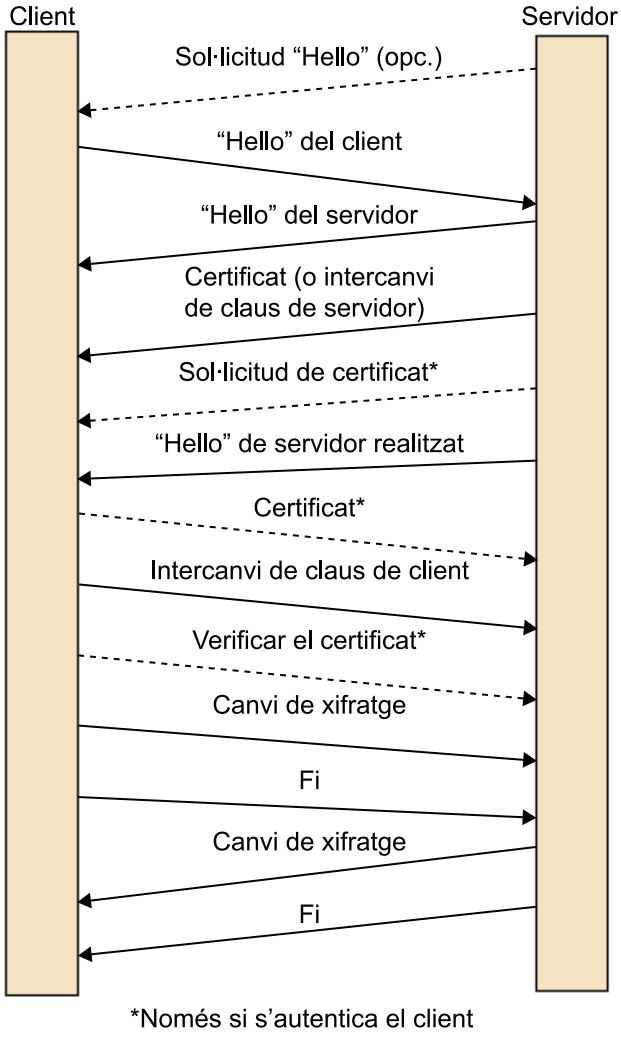
Una de les principals diferències entre SSL 3.0 i TLS 1.0 és la tècnica usada per a obtenir els codis de verificació dels missatges *finished*, i també per a calcular el secret mestre i per a obtenir les claus a partir d'aquest secret (en SSL s'utilitzen funcions resum directament, i en TLS s'utilitzen codis HMAC).

El contingut del missatge *finished* serveix per a verificar que la negociació s'ha dut a terme correctament. Aquest missatge també permet autenticar el servidor davant el client, ja que el primer necessita la seva clau privada per a desxifrar el missatge *client key exchange* i obtenir les claus que es faran servir en la comunicació.

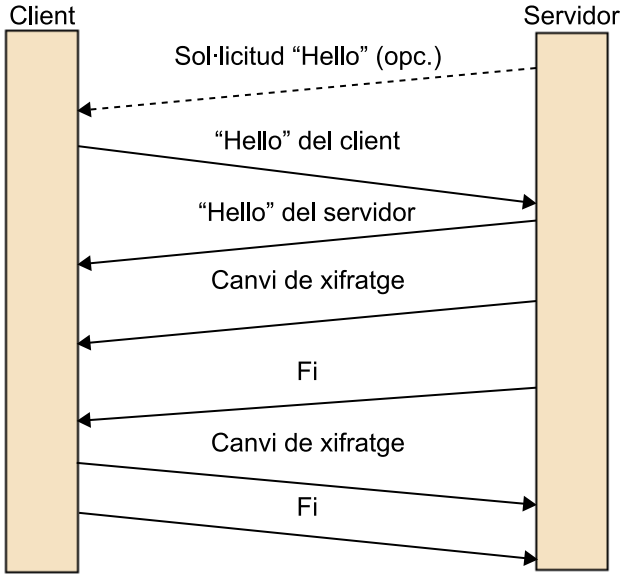
Un cop enviat el missatge *finished*, es dóna per acabada la negociació, i client i servidor poden començar a enviar les dades d'aplicació fent servir els algorismes i claus acordats.

Els diagrames següents resumeixen els missatges intercanviats durant la fase de negociació SSL/TLS.

Negociació d'una sessió SSL/TLS nova



Negociació d'una sessió SSL/TLS represa



A més dels missatges de negociació, notificacions de canvi de xifratge i dades d'aplicació, també es poden enviar missatges d'error. Aquests missatges contenen un codi de nivell de gravetat, que pot ser "missatge d'avís" o "error fatal", i un codi de descripció de l'error. Un error fatal provoca la fi de la connexió i la invalidació de l'identificador de sessió corresponent, és a dir, la sessió no podrà ser represa.

Exemples d'errors fatals

Són exemples d'errors fatals: MAC incorrecte, tipus de missatge inesperat, error de negociació, etc. (TLS 1.0 preveu més codis d'error que SSL 3.0).

També es pot enviar un missatge d'avís per a indicar la fi normal de la connexió. Per a evitar atacs de truncament, si una connexió acaba sense haver enviat aquest avís s'invalidarà l'identificador de sessió.

5.5. Transaccions segures en Xarxa

Cal conèixer també diferents iniciatives per a permetre transaccions segures a Internet. Presentem les més importants.

5.5.1. Secure Electronic Transaction

Arran de les mancances del protocol SSL, diferents empreses i organismes van buscar un sistema que permetés fer operacions sensibles per Internet de manera segura, com els pagaments, per tal d'estimular la confiança dels consumidors en el comerç electrònic. El 1996 un grup d'empreses del sector financer, informàtic i de seguretat (Visa International, MasterCard, Microsoft, Nestcape, IBM, RSA, etc.) va anunciar el desenvolupament d'aquesta nova tecnologia.

Secure Electronic Transaction[19] (SET) és un protocol que permet donar seguretat en les transaccions per Internet en què s'usin targetes de crèdit. Les seves especificacions es poden trobar al lloc web oficial de SETco, organisme encarregat d'homologar els mòduls de programació i els certificats desenvolupats per empreses privades que s'usin en implementacions del protocol SET.

[19]En català, 'transaccions electròniques segures'.

Adreça web recomanada

Podeu accedir a l'històric de la pàgina web de SETco a:
http://web.archive.org/web/*/www.setco.org

El seu funcionament es basa en l'ús de certificats digitals i sistemes criptogràfics. Els primers per a assegurar la perfecta identificació de totes aquelles parelles que intervenen en una transacció en línia basada en l'ús de targetes de pagament, i els segons per a protegir la tramesa de les dades sensibles en el seu viatge entre els diferents servidors que participen en el procés. Una de les seves característiques és la de separar el procés de compra del de pagament.

Com a inconvenients de la SET esmentarem la seva complexitat i lentitud. Normalment, amb SSL no es necessita certificat ni programari addicional, només seleccionar els productes que cal comprar i acceptar el pagament. La lentitud de la SET és deguda al fet que calen diferents verificacions d'identitat i integritat per part de diverses entitats al llarg d'una transacció.

5.5.2. 3D-Secure

3D-Secure és un sistema promogut per Visa/Mastercard amb la finalitat d'incentivar el comerç electrònic, que intenta evitar els fraus quan el pagament es fa amb targetes de crèdit.
Està basat en el model dels "tres dominis", en el qual es requereix:

1) Que l'entitat emissora de la targeta autentiqui el client o titular de la targeta (comprador): domini de l'emissor.

2) Que l'entitat adquirent autentiqui el comerç (venedor): domini de l'adquirent.

3) Que totes dues parts (entitat emissora i adquirent) siguin reconegudes mútuament com a legítimes per a efectuar la transacció, i que es completi de manera segura: domini d'intercanvi.

En aquest model es deixa a l'arbitri de cada una de les entitats l'elecció del "procediment d'autenticació". D'aquesta manera, tenen flexibilitat per a seleccionar el mètode d'autenticació més apropiat per als seus comerços i titulars, i substituir-lo per noves solucions quan ho considerin convenient. La confidencialitat i integritat de la informació de pagament s'aconsegueixen amb la utilització del protocol SSL. El procés de compra amb 3D Secure funciona com una compra normal en un comerç en línia, amb la particularitat que, a l'hora d'introduir la informació de pagament, el comprador haurà de facilitar la contrasenya 3D-Secure-Secure. D'aquesta manera, es confirma que és el titular mateix de la targeta i no un tercer sense autorització el que efectua la transacció.

Resum

El mòdul ha fet una introducció als conceptes bàsics de la seguretat en les xarxes de comunicació. Els **tallafocs** han estat presentats com a mesures passives de seguretat en els sistemes en xarxa. Hem vist que un tallafocs pot ser tant de maquinari com de programari i que fins hi tot es poden combinar diferents sistemes per a assegurar una xarxa.

S'ha presentat el concepte de *xarxa privada virtual* (VPN), que no és res més que una xarxa lògica o virtual creada sobre una infraestructura compartida, però que proporciona els serveis de protecció necessaris per a una comunicació segura. Les **xarxes privades virtuals** (VPN) permeten utilitzar la xarxa pública Internet com si fos una xarxa privada dedicada, per exemple, entre diverses intranets d'una mateixa organització. La tècnica bàsica que utilitzen les VPN són els **túnels**, en els quals els paquets protegits s'encapsulen dins de datagrames IP que circulen de manera normal per la xarxa Internet.

En aquest mòdul també hem vist que les **tècniques criptogràfiques** permeten xifrar un text mitjançant una **clau de xifratge**, i només qui conegui la **clau de desxifratge** corresponent serà capaç d'obtenir el text original.

Segons la relació que hi hagi entre totes dues claus, els algorismes criptogràfics es classifiquen en **algorismes simètrics** si la clau de xifratge i la de desxifratge són la mateixa, o **algorismes de clau pública** si les claus són diferents. Els algorismes simètrics, al seu torn, es poden classificar en **algorismes de flux**, si el xifratge consisteix a afegir al text dades pseudoaleatòries calculades a partir de la clau, o **algorismes de bloc**, si el xifratge es fa sobre blocs de mida fixa del text original.

La particularitat de la criptografia de clau pública és que a partir d'una de les claus, la **clau privada**, és pot deduir fàcilment l'altra, la **clau pública**, mentre que la deducció inversa és pràcticament impossible. Això permet que tothom que conegui la clau pública d'un usuari la pugui fer servir per a xifrar dades confidencials, amb la seguretat que només qui tingui la clau privada les podrà desxifrar, i sense necessitat d'acordar cap clau secreta a través d'un canal a part. L'ús de les claus a l'inrevés (la privada per a xifrar i la pública per a desxifrar) és la base de les **signatures digitals**.

Com que la criptografia de clau pública és computacionalment més costosa que la simètrica, no es fa servir mai directament per a obtenir confidencialitat, sinó sempre per mitjà d'una clau de sessió simètrica. De la mateixa manera, la

signatura d'un text no es calcula directament a partir del text, sinó aplicant-hi una funció resum segura. La propietat d'aquest tipus de funció és que és molt difícil trobar un missatge que doni el mateix resum que un altre.

Després d'aquesta aproximació més teòrica, i per concloure el mòdul, s'han presentat diferents conceptes de seguretat de la Xarxa a escala d'aplicació i transport. Hem aprofundit en el coneixement d'un protocol, l'SSL/TLS, de la capa de transport, que serveix per a dotar de seguretat a les comunicacions de les aplicacions en xarxa. L'ús típic dels protocols SSL/TLS és protegir de manera transparent un protocol d'aplicació com és HTTP. El protocol **HTTPS** és simplement la combinació d'HTTP amb el transport segur SSL/TLS.

Bibliografia

Menezes, A. J.; Oorschot, P. C. van; Vanstone, S. A. (1996). *Handbook of Applied Cryptography*. Boca Ratón: CRC Press.

SETco [documentació en línia]. <http://web.archive.org/web/*/www.setco.org>. [Data de consulta: 12 d'abril del 2010.]

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

Yuan, R.; Strayer, W. T. (2001). *Virtual Private Networks, Technologies and Solutions*. Boston: Addison-Wesley.