Mecanismes de protecció

Xavier Perramon Tornil

PID_00187027

Universitat Oberta de Catalunya

Índex

Introducció

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 la primera part d'aquest mòdul a introduir els conceptes de la criptografia necessaris per 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.

Però a més de mantenir la informació en secret, hi ha altres serveis que poden ser igualment necessaris, com ara el de l'**autenticació**. Hem d'evitar, per exemple, que després de prendre totes les mesures necessàries perquè només el destinatari final pugui llegir la informació, resulti que aquest destinatari és un impostor que ha aconseguit fer-se passar per l'autèntic. A la segona part del mòdul veurem alguns sistemes per garantir l'autenticitat a les comunicacions, la majoria d'ells basats en tècniques criptogràfiques.

A la resta d'aquest mòdul didàctic estudiarem exemples de protocols de comunicació que, aplicant els mecanismes anteriors, permeten protegir la informació que es transmet entre ordinadors. Aquesta protecció es pot assolir a diferents nivells de l'arquitectura de comunicacions. En les xarxes sense fil, un dels nivells més vulnerables que hem de protegir és el **nivell d'enllaç**. Veurem les solucions proposades per a les especificacions Wi-Fi. A **nivell de xarxa**, el mecanisme principal en un entorn d'interconnexió basat en IP és el conjunt de protocols conegut com **IPsec**.

Alternativament, es pot implementar la protecció a **nivell de transport**, aprofitant així la infraestructura IP existent, principalment els encaminadors o *routers*. Com a exemple de protecció a nivell de transport veurem la família de protocols **SSL/TLS/ WTLS**.

Per cloure aquest mòdul, introduirem la tecnologia de **xarxes privades virtuals** o **VPN**, que permet utilitzar una xarxa pública àmpliament estesa com és Internet per a comunicacions segures, com si fos una xarxa privada dedicada.

La protecció al nivell més alt de la comunicació, el nivell d'aplicació, serà objecte d'estudi al mòdul didàctic següent.

Objectius

Els materials associats a aquest mòdul permetran a l'estudiant assolir els objectius següents:

1. Saber quines funcions ens ofereix la criptografia, tant les tècniques de clau simètrica com les de clau pública.

2. Conèixer els diferents algorismes de xifratge, integritat i autenticació disponibles, i els seus usos possibles.

3. Combinar les eines de criptografia pública amb les de criptografia simètrica per aconseguir diferents prestacions.

4. Conèixer l'ús dels certificats X.509 i les llistes de revocació, la seva estructura i la utilitat dels diferents camps.

5. Reconèixer la necessitat dels sistemes d'autenticació, quines tècniques concretes fan servir, i com aquestes tècniques permeten contrarestar els intents de suplantació.

6. Comprendre les possibilitats de protegir els protocols de comunicació a diferents nivells, i en particular el nivell d'enllaç, el de xarxa i el de transport.

7. Conèixer els mecanismes de protecció de les xarxes Wi-Fi.

8. Conèixer els protocols que formen l'arquitectura IPsec, i quines proteccions ofereix cadascun.

9. Conèixer el mecanisme general de funcionament dels protocols de transport segur SSL/TLS, i com aquests protocols poden ser utilitzats per altres de nivell superior, com HTTP o TELNET.

10. Introduir la tecnologia de les xarxes privades virtuals, i com es poden usar per connectar intranets de manera segura mitjançant una xarxa d'accés públic com és Internet.

1. Conceptes bàsics de criptografia

Al llarg de la història, s'han dissenyat diferents tècniques per ocultar el significat de la informació que no interessa que sigui coneguda per estranys. Algunes d'elles 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 enviar missatges xifrats que no poguessin ser interpretats per l'enemic.

Criptografia

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

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

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**.

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.

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 transformi en un altre missatge que anomenarem **text xifrat**, C. Això ho podem expressar com:

$$C = f(M)$$

Per tal que aquest xifratge sigui útil, ha d'existir 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

Per exemple, la "xifra del Cèsar" que abans hem esmentat 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 aprenguessin 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**.

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:

$$\begin{aligned} C &= e(k,M) \\ M &= d(x,C) = d(x,e(k,M)) \end{aligned}$$

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

Exemple d'ús d'una clau

L'algorisme de Juli Cèsar es pot generalitzar definint una clau k que indiqui quantes posicions 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 ($d \equiv e, x = -k$).

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.

Seguretat per ocultisme

Al llarg de la història hi ha hagut casos que han demostrat la perillositat de basar la protecció en 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 atacar-lo, com va passar amb l'algorisme A5/1 de la telefonia mòbil GSM.

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 desxifrar els missatges sense conèixer-ne la clau és molt més gran (uns quants ordres de magnitud) del que està 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 a 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.

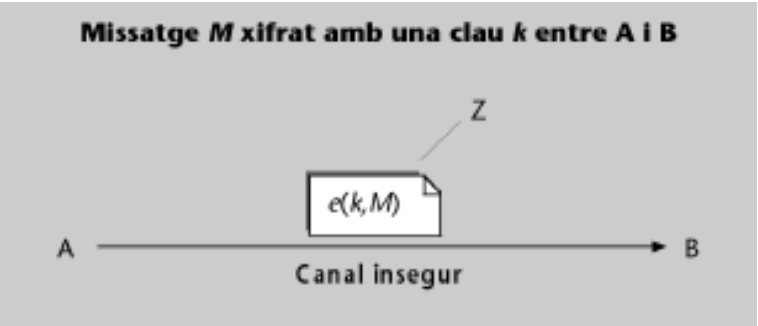
1.1. Criptografia de clau simètrica

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.

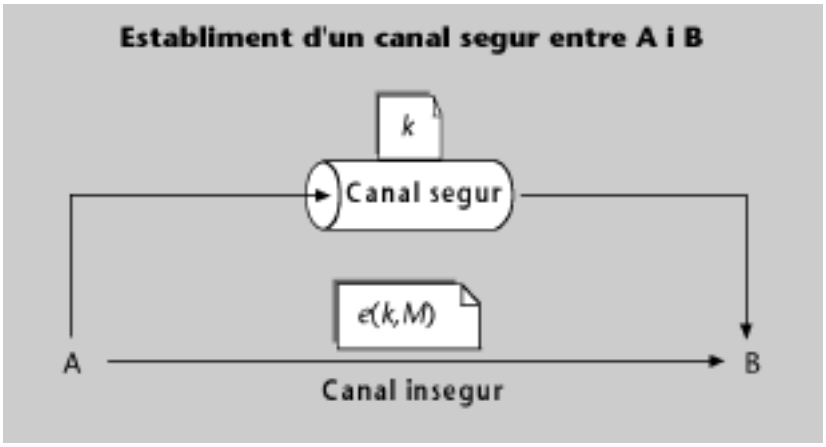
Per simplificar, suposarem que en aquest tipus de sistemes la clau de desxifratge és igual a la de xifratge: $x = k$ (si no, sempre podem considerar que en l'algorisme de desxifratge el primer pas és calcular la clau x a partir de la k). És per això que aquestes tècniques criptogràfiques s'anomenen de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

$$\begin{aligned} C &= e(k,M) \\ M &= d(k,C) = d(k,e(k,M)) \end{aligned}$$

La seguretat del sistema rau, doncs, a mantenir en secret la clau k. Quan els participants en una comunicació volen intercanviar-se missatges confidencials, han d'escollir una clau secreta i fer-la servir per xifrar els missatges. Llavors 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 serà capaç d'interpretar-los.



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? Està 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:



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 haguem 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.

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.

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.

1.1.1. 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 desxifrar només cal fer l'operació inversa amb el text xifrat i el mateix text de xifratge S.

Suma i resta de bits

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

$$0+0=0 \quad 0-0=0$$
$$0+1=1 \quad 0-1=1$$
$$1+0=1 \quad 1-0=1$$
$$1+1=0 \quad 1-1=0$$

$$0 \oplus 0 = 0$$
$$0 \oplus 1 = 1$$
$$1 \oplus 0 = 1$$
$$1 \oplus 1 = 0$$

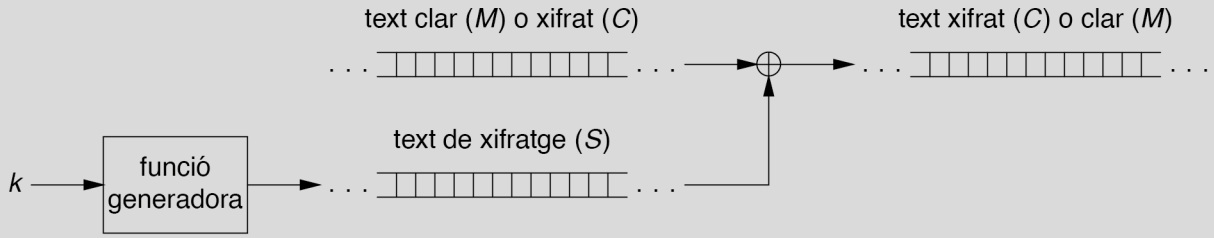
L'operació de combinació que s'utilitza típicament és la suma, i l'operació inversa, per tant, és la resta. Si el text està format per caràcters, aquest algorisme seria com una xifra del Cèsar en què la clau va canviant d'un caràcter a un altre. La clau que pertoca cada vegada ve donada pel text de xifratge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

Si considerem el text format per bits, la suma i la resta són equivalents. En efecte, quan s'apliquen bit a bit, totes dues són idèntiques a l'operació lògica "O exclusiva", denotada amb l'operador XOR (*"eXclusive OR"*) o el símbol $\oplus$. 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 seva 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 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 mateixa clau ha de ser tan llarga com el missatge a 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 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.

- El que es fa a 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.

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 ambdues parts comparteixen.

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 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 vindrà donat 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 a 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 desincronitzarà 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 "auto-sincronitzant"), 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 propis 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 el 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 on 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.

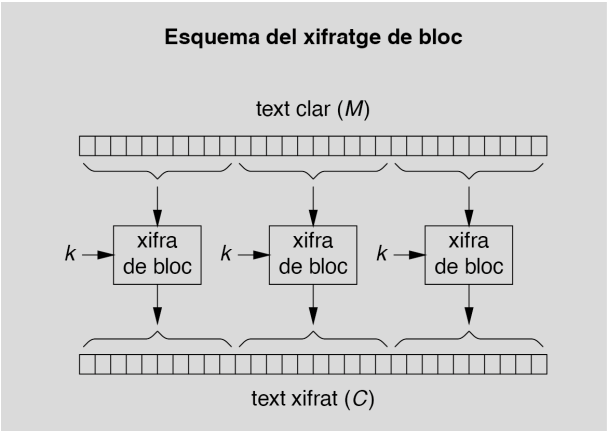
Un exemple d'algorisme de xifratge de flux és l'**RC4 (Ron's Code 4)**. Els algorismes següents en són exemples. Va ser dissenyat per Ronald Rivest el 1987 i publicat a Internet per un remitent anònim el 1994. És l'algorisme de xifratge de flux més utilitzat en moltes aplicacions gràcies a la seva simplicitat i rapidesa. Per exemple, el sistema de protecció WEP (*Wired Equivalent Privacy*) que incorpora l'estàndard IEEE 802.11 per a tecnologia LAN sense fils utilitza aquest criptosistema de xifratge de flux.

1.1.2. 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 d'ells el resultat és un bloc de la mateixa longitud.

Per xifrar un text en clar de L bits cal dividir-lo en blocs de b bits cadascun i xifrar aquests blocs un a 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 realitzar bloc a bloc.

La figura següent mostra l'esquema bàsic 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, on 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 tenen per què estar a una distància constant (la k de l'algorisme, tal com l'hem definit) de les lletres del text en clar. La clau es pot expressar llavors com la seqüència correlativa de lletres que corresponen a la A, la B, la C, etc. Per exemple:

  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

Clau de la substitució alfabètica

Està 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 **transposició** consisteix a reordenar la informació del text en clar segons un patró determinat. Un exemple podria ser formar grups de 5 lletres, inclosos 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 pot combinar-se amb altres operacions per afegir complexitat als algorismes de xifratge.

El **producte de xifres**, o combinació en cascada de diverses transformacions criptogràfiques, és una tècnica molt efectiva per 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 a l'entrada es propaguin per tota la sortida per "efecte allau".

Exemples d'algorismes de xifratge de bloc

DES (Data Encryption Standard). Durant molts anys ha estat l'algorisme més estudiat i alhora el més utilitzat. Desenvolupat per IBM durant els anys 70, va ser adoptat per l'NBS nord-americà (nom que tenia llavors l'actual NIST) com a estàndard per al xifratge de dades l'any 1977.

NBS i NIST

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, 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.

Bits addicionals de la clau DES

Un possible ús dels bits de la clau DES que no influeixen en l'algorisme és com a bits de paritat.

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 a bit (XOR) amb la sortida de la iteració anterior. Tal com està dissenyat l'algorisme, el desxifratge es realitza igual que el xifratge però fent les transposicions de la clau en l'ordre invers (començant per l'última).

Triple DES. 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 70 era molt costós fer una cerca entre les 2^{56} combinacions possibles, la tecnologia actual permet trencar l'algorisme en un temps cada vegada més curt.

Els reptes DES

A l'adreça `www.rsasecurity.com/rsalabs/challenges/` podeu trobar informació sobre els anomenats "reptes DES", que demostren com l'any 1999 ja era possible trencar una clau DES en menys de 24 hores.

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 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:

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

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

Repetició del DES

L'operació 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.

AES (Advanced Encryption Standard). Com que l'estàndard DES començava a quedar-se 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.

Doble DES

El motiu pel qual l'algorisme DES s'aplica tres vegades, i no dues, és perquè un atacant que tingués prou recursos podria trencar el "Doble DES" per força bruta, mitjançant un atac de text en clar conegut, amb menys de 2^{57} operacions de xifratge en lloc de les 2^{112} que caldria esperar.

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.

1.1.3. Ús dels algorismes de clau simètrica

Quan s'usa el xifratge simètric per 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.

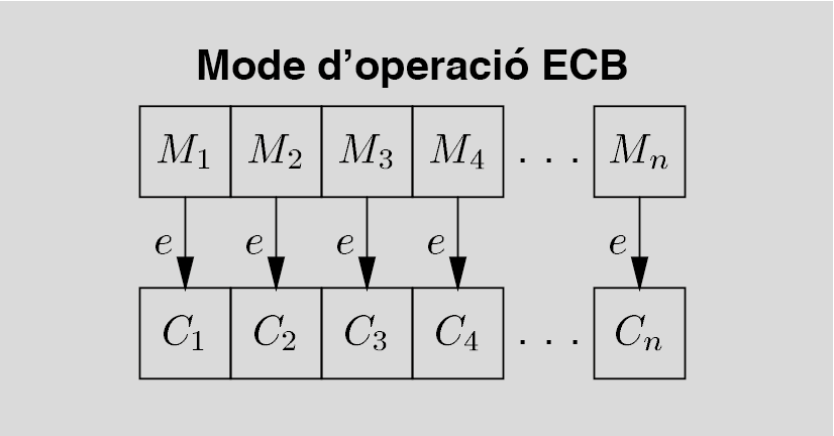
Un aspecte a tenir en compte és que si bé el xifratge pot fer que un atacant no descobreixi directament les dades transmeses, de vegades és possible que es pugui deduir informació indirectament. Per exemple, en un protocol que faci servir missatges amb una capçalera fixa, l'aparició de les mateixes dades 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 contrarestar aquesta propietat, es poden aplicar en diversos **modes d'operació** a les xifres de bloc.

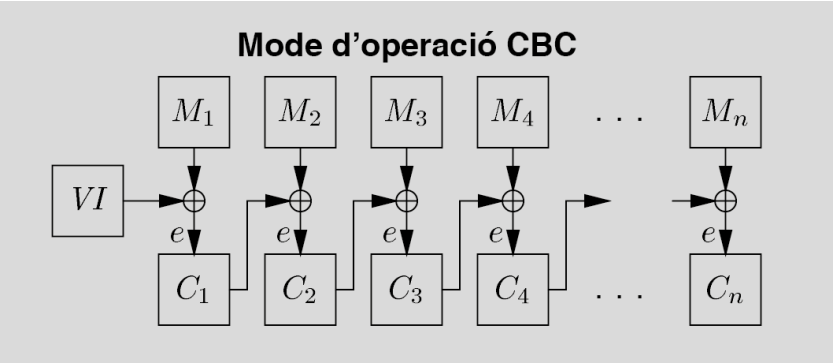
- El mode ECB (*Electronic Codebook*) és el més senzill, i consisteix simplement a dividir el text en blocs i xifrar cadascun de manera independent. Aquest mode pateix del problema de donar blocs iguals quan a l'entrada hi ha blocs iguals.

Mode ECB

El nom del mode ECB (*Electronic Codebook*) dóna la idea que es pot considerar com una simple substitució bloc a bloc, d'acord amb un codi o diccionari (amb moltes entrades, això sí) que ve donat per la clau.



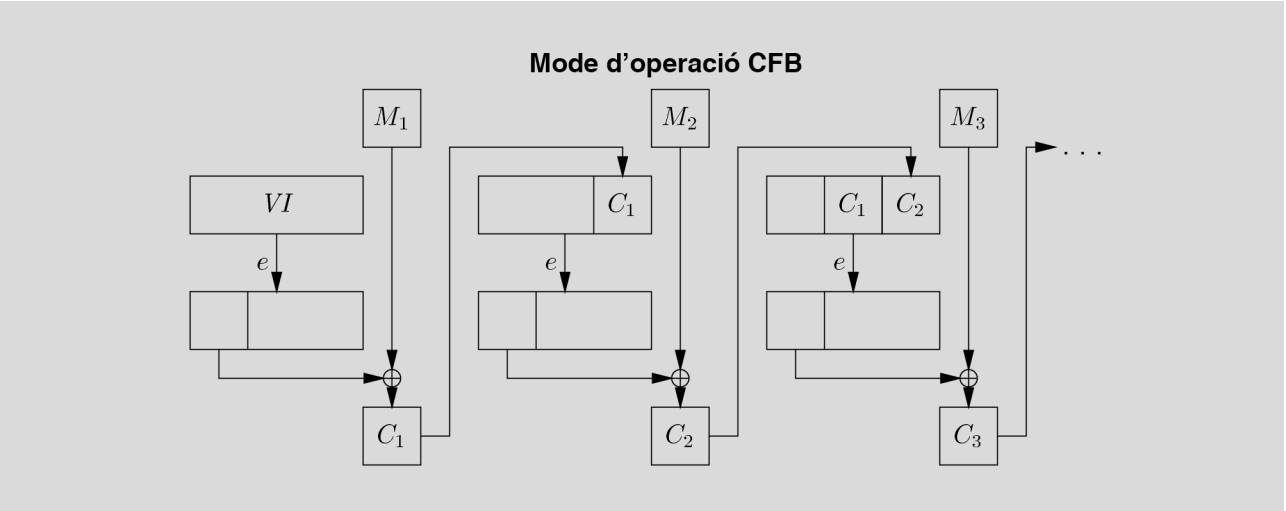
- En el mode CBC (*Cipher Block Chaining*), abans de xifrar cada bloc de text en clar se li suma (bit a bit, amb XOR) el bloc xifrat anterior. Al primer bloc se li suma un **vector d'inicialització** (VI), que és un conjunt de bits aleatoris de la mateixa longitud que un bloc. Escollint vectors diferents cada vegada, encara que el text en clar sigui el mateix les dades xifrades seran diferents. El receptor ha de saber 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.



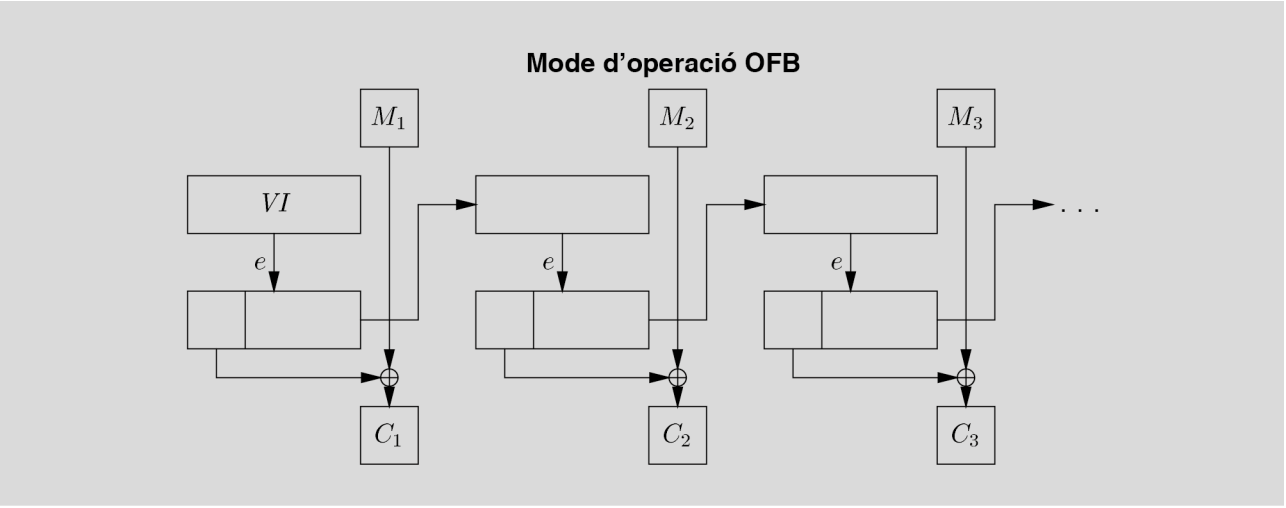
- En el mode CFB (*Cipher Feedback*), l'algorisme de xifratge no s'aplica directament al text en clar sinó a un vector auxiliar (inicialment igual al VI). Del resultat del xifratge es prenen n bits que se sumen a n bits del text en clar per obtenir n bits de text xifrat. Aquests bits xifrats s'usen alhora per actualitzar el vector auxiliar. El nombre n de bits generats a 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 poder-lo xifrar.

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.



- El mode OFB (*Output Feedback*) 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.



- 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 evitar que en textos d'entrada iguals s'obtinguin textos xifrats iguals, que és aplicable també als xifratges que no usen vector d'inicialització (incloses 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 donar un "sabor" diferent a la clau, se'ls sol anomenar **bits de sal**. Igual que el vector d'inicialització, els bits de sal s'envien en clar abans del text xifrat.

1.1.4. Funcions *hash* segures

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

En general, podem dir que una funció *hash* 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 poder-la aplicar en un sistema d'autenticació, la funció h ha de ser una funció *hash* segura.

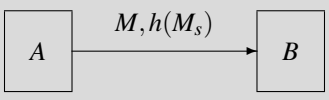
S'entén que una funció *hash* o de resum és **segura** si compleix aquestes condicions:

- És **unidireccional**, és a dir, si tenim $H = h(M)$ és computacionalment inviable trobar M a partir del resum H.
- És **resistent a col·lisions**, és a dir, donat un missatge M qualsevol és computacionalment inviable trobar un missatge $M' \neq M$ tal que $h(M') = h(M)$.

Secret dels algorismes

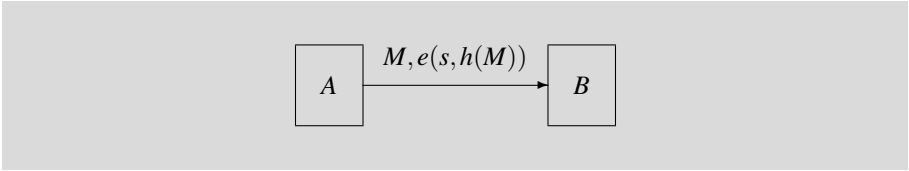
Noteu que les funcions *hash* són conegudes ja que tothom ha de poder calcular els resums de la mateixa manera.

Aquestes propietats permeten l'ús de les funcions *hash* segures per 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 ha modificat el missatge.

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



Per 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 podria intentar substituir-lo 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 de l'ordre de 2^{128} o 2^{160} missatges d'entrada per 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: de l'ordre de 2^{64} o 2^{80}, si el resum és de 128 o 160 bits, respectivament.

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 l'ordre de 200, però aquest resultat no és correcte. Podria ser-ho 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 l'ordre de $\sqrt{N}$.

Autenticitat i confidencialitat

Xifrar només el resum, en lloc del missatge sencer, és més eficient perquè hi ha menys bits a 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.

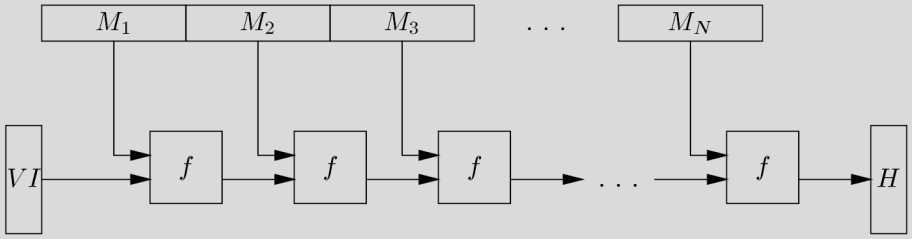
Resistència forta a les col·lisions

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

Exemples de funcions *hash* segures

L'esquema de la majoria de funcions *hash* 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 no fa gaire, l'algorisme de *hash* més usat era l'**MD5 (Message Digest 5)**. Però com que el resum que dóna és de només 128 bits, i a més s'han trobat maneres de generar col·lisions parcials en l'algorisme, actualment es recomana utilitzar algorismes més segurs, com el **SHA-1 (Secure Hash Algorithm-1)**. L'algorisme SHA-1, publicat el 1995 en un estàndard del NIST (com a revisió d'un algorisme anterior anomenat simplement SHA), dóna resums de 160 bits. L'any 2002 el NIST va publicar variants d'aquest algorisme que generen resums de 256, 384 i 512 bits.

Longitud del resum MD5

Com que la longitud del resum MD5 és de 128 bits, el nombre d'operacions per a un atac d'aniversari és de l'ordre de 2^{64}. Compareu aquesta magnitud amb la d'un atac per força bruta contra el DES (menys de 2^{56} operacions), de la qual no està gaire lluny.

1.2. Criptografia de clau pública

1.2.1. 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 **de clau asimètrica**.

En un algorisme criptogràfic de clau pública es fan servir claus diferents per al xifratge i el desxifratge. Una d'elles, 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}):

$$\begin{aligned} C &= e(k_{\text{pub}}, M) \\ M &= d(k_{\text{pr}}, C) \end{aligned}$$

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):

$$\begin{aligned} C &= e(k_{\text{pr}}, M) \\ M &= d(k_{\text{pub}}, C) \end{aligned}$$

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 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.

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.

Un exemple de problema fàcil de plantejar però difícil de resoldre és el dels logaritmes discrets. Si treballem amb aritmètica 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 trobar la solució augmenta proporcionalment a 2^n. Hi ha altres mètodes més eficients per calcular logaritmes discrets, però el millor algorisme conegut també necessita més temps del que es pot expressar polinòmicament.

Exemple d'operacions mòdul *m*

Per exemple, per obtenir $14^{11} \bmod 19$ 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 ($11 = 1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0$), i per tant $14^{11} = 14^8 \cdot 14^2 \cdot 14^1$, per obtenir el resultat amb menys multiplicacions:

$$\begin{aligned}
14^1 &= 14 && \equiv 14 \pmod{19} \rightarrow 14 \\
14^2 &= 14^1 \times 14^1 \equiv 14\times 14 = 196 \equiv 6 \pmod{19} \rightarrow 6 \\
14^4 &= 14^2 \times 14^2 \equiv 6\times 6 = 36 \equiv 17 \pmod{19} \\
14^8 &= 14^4 \times 14^4 \equiv 17\times 17 = 289 \equiv 4 \pmod{19} \rightarrow 4 \\
& && 336 \equiv 13 \pmod{19}
\end{aligned}$$

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 a un els exponents fins a trobar-ne un que doni com a resultat y. I si en comptes de tractar-se de nombres de 4 o 5 bits com aquests fossin nombres de més de 1000 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 realitzar en un temps relativament curt.

En la pràctica, els algorismes utilitzats permeten xifrar i desxifrar fàcilment, però tots ells són considerablement més lents que els equivalents amb criptografia simètrica. 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 amb no 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 criptografia simètrica.

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

  Per exemple, 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 xifrar i la clau pública per desxifrar. Si A envia un missatge xifrat amb la seva clau privada, tothom podrà desxifrar-lo amb la clau pública de 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

Intercanvi de claus Diffie-Hellman. És un mecanisme que permet que dues parts es posin d'acord de forma segura sobre una clau secreta. L'algorisme es basa en la dificultat de calcular logaritmes discrets.

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 forma que el xifratge i el desxifratge es poden realitzar així:

$$\begin{aligned} \text{Xifratge:} \quad C &= M^e \bmod n \\ \text{Desxifratge:} \quad M &= C^d \bmod n \end{aligned}$$

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

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 1024 bits. Com a exponent públic e típicament es fan servir valors senzills com 3 o 65537 ($2^{16}+1$) perquè fan més ràpid el xifratge.

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 del problemes considerats difícils).

ElGamal. Un altre esquema de xifratge, en aquest cas basat en el problema de Diffie-Hellman.

DSA (Digital Signature Algorithm). Publicat pel NIST en diverses versions del *Digital Signature Standard* (DSS), la primera d'elles l'any 1991. És una variant de l'algorisme de signatura ElGamal, que al seu torn segueix l'esquema de xifratge ElGamal. El DSA no és un algorisme de xifratge, sinó que només permet generar signatures i verificar-les.

Corbes el·líptiques. Mentre que els algorismes anteriors treballen bàsicament amb productes de nombres molt grans en aritmètica modular, aquesta nova branca de la criptografia de clau pública utilitza operacions definides sobre punts de corbes el·líptiques en un espai de coordenades enteres bidimensionals. L'avantatge d'aquesta tècnica és que proporciona una seguretat equivalent a la dels altres algorismes amb molts menys bits de clau, i amb un temps de càlcul també molt menor.

Seguretat amb corbes el·líptiques

La seguretat que proporciona una clau RSA de 1024 bits es pot assolir amb menys de 170 bits de clau quan s'utilitza criptografia de corbes el·líptiques.

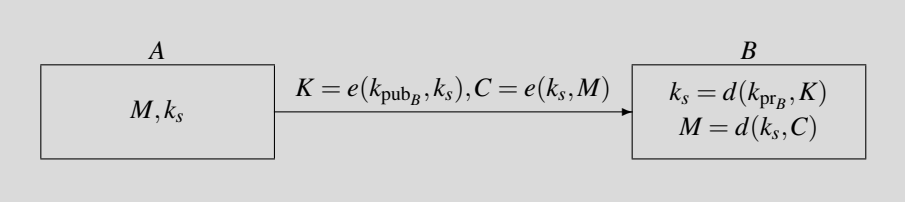
1.2.2. Ús de la criptografia de clau pública

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

- El problema de la confidencialitat entre dues parts que només disposen d'un canal insegur per 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 recuperar la clau de sessió k_s, i llavors ja pot desxifrar el missatge xifrat.

Sobre digital

Com veurem més endavant, aquesta tècnica per proporcionar confidencialitat amb criptografia de clau pública se sol anomenar "sobre digital" en el context del correu electrònic segur.

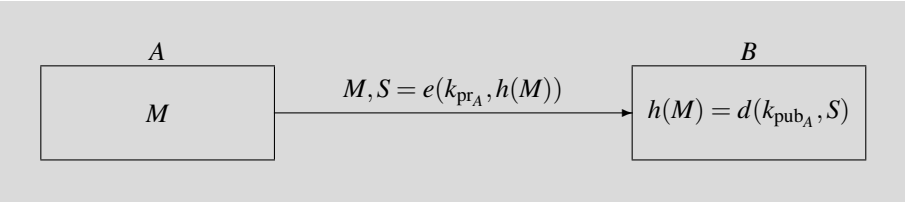


Ja que la clau de sessió és un missatge relativament curt (per exemple, si és una clau DES només tindrà 56 bits), un atacant podria intentar trencar el xifratge per força bruta, però no provant de desxifrar el missatge amb els possibles valors de la clau privada k_{pr_B}, sinó xifrant els possibles valors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

Per 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 a signar, sinó només el seu resum calculat amb una funció *hash* segura.

  Quan A vulgui enviar un missatge signat, haurà d'obtenir el seu resum i xifrar-lo amb la clau privada k_{pr_A}. Per verificar la signatura, el receptor ha de desxifrar-la 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 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.



1.3. Infraestructura de clau pública (PKI)

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 de 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 de 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

La sigla PKI correspon al nom en anglès de la infraestructura de clau pública (*Public Key Infrastructure*).

1.3.1. Certificats de clau pública

Un certificat de clau pública consta de tres parts bàsiques:

- Una identificació d'usuari, com per exemple el seu nom.
- El valor de la clau pública d'aquest usuari.
- 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** (CA).

CA

CA és la sigla anglesa de *Certification Authority*.

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

El directori X.500

L'especificació del directori X.500 està publicada a 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 (a cada node de l'arbre hi ha un objecte) i, dins de cada nivell, els objectes s'identifiquen mitjançant un **atribut distintiu**. A nivell global, cada objecte s'identifica amb un **nom distintiu** (DN), que no és més que la concatenació dels atributs distintius que hi ha entre l'arrel de l'arbre i l'objecte en qüestió. 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.

DN

DN és la sigla anglesa de *Distinguished Name*.

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"), *stateOrProvin-*

ceName (“st”), *localityName* (“l”), *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 realitzar 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 a la Recomanació X.509. Hi ha un mecanisme bàsic que fa servir contrasenyes, i un mecanisme més avançat que fa servir certificats.

A continuació podeu veure l’estructura d’un certificat X.509, amb els seus camps i subcamps. En la notació usada aquí, “rep.” vol dir que el camp es pot repetir una o més vegades, i “opc.” vol dir que el camp és opcional (i per tant “opc. rep.” vol dir que el camp es pot repetir zero o més vegades).

Camp	**Tipus**
toBeSigned	
version (opc.)	enter
serialNumber	enter
signature	
algorithm	identificador únic
parameters	(depèn de l’algorisme)
issuer	DN
validity	
notBefore	data i hora
notAfter	data i hora
subject	DN
subjectPublicKeyInfo	
algorithm	
algorithm	identificador únic
parameters	(depèn de l’algorisme)
subjectPublicKey	cadena de bits
issuerUniqueIdentifier (opc.)	cadena de bits
subjectUniqueIdentifier (opc.)	cadena de bits
extensions (opc. rep.)	
extnId	identificador únic
critical (opc.)	booleà
extnValue	(depèn de l’extensió)
algorithmIdentifier	
algorithm	identificador únic
parameters	(depèn de l’algorisme)
encrypted	cadena de bits

El cos del certificat ("`toBeSigned`" en la taula anterior) està format pels següents camps:

- El camp `version` és el número de versió del format: pot ser 1, 2 o 3. Encara que és un camp opcional, és obligatori si la versió és 2 o 3 (així, en absència d'aquest camp se sap que la versió és 1).
- El camp `serialNumber` és el número de sèrie del certificat, i serveix per distingir-lo de tots els altres que hagi generat la mateixa CA.
- El camp `signature` indica l'algorisme utilitzat per la CA per signar el certificat.
- El camp `issuer` és el nom distintiu (DN) del signant o **emissor** del certificat, és a dir, de la CA que l'ha generat.
- El camp `validity` indica el període de validesa del certificat, entre un instant inicial i un instant de caducitat. Quan parlem més endavant de les llistes de revocació veurem la utilitat de la data de caducitat en els certificats.
- El camp `subject` és el nom distintiu (DN) del **subjecte** a nom del qual està emès el certificat, és a dir, el titular de la clau pública.
- Al camp `subjectPublicKeyInfo` hi ha la clau pública del subjecte: a quin algorisme correspon, i el seu valor.
- Els camps `issuerUniqueIdentifier` i `subjectUniqueIdentifier` es van introduir a la versió 2, però no se solen fer servir perquè amb el camp `extensions` són innecessaris.
- El camp `extensions` es va introduir a la versió 3, i és una llista d'atributs addicionals del certificat. El subcamp `extnId` indica de quin tipus d'extensió es tracta. El subcamp `critical` indica si l'extensió és crítica o no: si ho és, les aplicacions que no reconeguin el tipus d'extensió han de considerar el certificat com a no vàlid (aquest subcamp és opcional perquè per defecte les extensions no són crítiques). El subcamp `extnValue` és el valor concret de l'extensió.

Números de sèrie

Els números de sèrie dels certificats generats per una CA no tenen per què ser consecutius, n'hi ha prou que cadascun sigui diferent de tots els anteriors.

Tipus d'extensions

Hi ha un cert nombre d'extensions estàndard, però se'n poden definir de noves, segons les necessitats de les aplicacions, mentre cadascuna s'identifiqui amb un valor inambigu del subcamp `extnId`.

Després del cos hi ha el camp `algorithmIdentifier` (que en realitat és redundant amb el camp `signature`), i finalment hi ha el camp `encrypted`, que és la signatura del certificat, és a dir, el resultat de xifrar amb la clau privada de la CA el resum (*hash*) del cos del certificat.

Per tal de calcular el resum, el cos del certificat s'ha de representar com una seqüència de bytes mitjançant una codificació no ambigua, que garanteixi que qualsevol que vulgui verificar la signatura pugui reconstruir exactament la mateixa seqüència de bytes. Les regles per obtenir aquesta codificació s'anomenen DER, i formen part de la notació ASN.1, que és la notació en què esta definit el format dels certificats X.509.

ASN.1 i DER

ASN.1 és la sigla de *Abstract Syntax Notation One*, i DER és la sigla de *Distinguished Encoding Rules*.

1.3.2. 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?

Existeix 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, on 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 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.

1.3.3. Llistes de revocació de certificats (CRL)

La Recomanació X.509, a més de definir el format dels certificats, també defineix una altra estructura anomenada **llista de revocació de certificats** o CRL. Una llista d'aquest tipus serveix per 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 a l'empresa on estava), li han robat la clau privada, etc.

CRL

CRL és la sigla de *Certificate Revocation List*.

Així, si volem assegurar-nos completament de la validesa d'un certificat, no n'hi ha prou de verificar la seva signatura, 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 forma periòdica, afegint-hi cada vegada els certificats que hagin estat revocats. Quan arribi la data de caducitat que

constava al certificat, ja no caldrà tornar-lo a incloure a la CRL. Això permet que les CRL no creixin indefinidament.

Aquests són els camps d'una CRL:

Camp	**Tipus**
`toBeSigned`	
`version` (opc.)	enter
`signature`	
`algorithm`	identificador únic
`parameters`	(depèn de l'algorisme)
`issuer`	DN
`thisUpdate`	data i hora
`nextUpdate` (opc.)	data i hora
`revokedCertificates` (opc. rep.)	
`userCertificate`	enter
`revocationDate`	data i hora
`crlEntryExtensions` (opc. rep.)	
`extnId`	identificador únic
`critical` (opc.)	booleà
`extnValue`	(depèn de l'extensió)
`crlExtensions` (opc. rep.)	
`extnId`	identificador únic
`critical` (opc.)	booleà
`extnValue`	(depèn de l'extensió)
`algorithmIdentifier`	
`algorithm`	identificador únic
`parameters`	(depèn de l'algorisme)
`encrypted`	cadena de bits

El camp `issuer` identifica la CA que emet la CRL, i al camp `revokedCertificates` hi ha la llista dels certificats revocats (cadascun identificat pel seu número de sèrie). A les extensions de cada element de la llista hi pot haver, per exemple, el motiu de la revocació.

2. Sistemes d'autenticació

Un dels serveis de seguretat que es necessita en moltes aplicacions és el de l'**autenticació**. Aquest servei permet garantir que ningú ha falsificat la comunicació.

Podem distingir dos tipus d'autenticació:

- L'**autenticació de missatge** o **autenticació d'origen de dades** permet confirmar que l'originador *A* d'un missatge és autèntic, és a dir, que el missatge no ha estat generat per un tercer *Z* que vol fer creure que l'ha generat *A*.

  Com a efecte addicional, l'autenticació de missatge implícitament proporciona el servei d'**integritat de dades**, que permet confirmar que ningú ha modificat un missatge enviat per *A*.

- L'**autenticació d'entitat** permet confirmar la identitat d'un participant *A* en una comunicació, és a dir, que no es tracta d'un tercer *Z* que diu ser *A*.

A continuació veurem com es pot aconseguir cadascun d'aquests dos tipus d'autenticació.

2.1. Autenticació de missatge

Hi ha dos grups de tècniques per proporcionar autenticació de missatge:

- Els **codis d'autenticació de missatge** o **MAC**, basats en claus simètriques.
- Les **signatures digitals**, que es basen en la criptografia de clau pública.

MAC

MAC és la sigla de *Message Authentication Code*.

2.1.1. Codis d'autenticació de missatge (MAC)

Un **codi d'autenticació de missatge** o **MAC** s'obté amb un algorisme a que té dues entrades: un missatge M de longitud arbitrària, i una clau secreta k compartida per l'originador i el destinatari del missatge. Com a resultat dóna un codi $C_{\mathrm{MAC}} = a(k,M)$ de longitud fixa. L'algorisme MAC ha de garantir que sigui computacionalment inviable trobar un missatge $M' \neq M$ que doni el mateix codi que M, i també obtenir el codi d'un missatge qualsevol sense conèixer-ne la clau.

Les propietats dels algorismes MAC fan que donat un parell (M, C_{MAC}) un atacant no pugui obtenir un altre parell (M', C_{MAC}), ni en general qualsevol parell (M', C'_{MAC}). Per tant, el codi MAC serveix com a prova d'autenticitat del missatge.

Un possible algorisme MAC consisteix a aplicar al missatge M un xifratge de bloc en mode CBC amb la clau k, i prendre l'últim bloc xifrat com a codi C_{MAC}. Els algorismes MAC usats actualment, però, solen estar basats en una funció *hash*. Per exemple, la tècnica de calcular el resum a partir de la concatenació del missatge i la clau, o la de calcular el resum del missatge i xifrar-lo amb la clau, podrien servir com a algorismes MAC. Per millorar la seguretat contra certs atacs, molts protocols usen una tècnica d'autenticació de missatges una mica més sofisticada, anomenada HMAC.

Codi HMAC

Per calcular el codi HMAC, al missatge se li afegeix com a prefix una cadena de bits derivada de la clau, es calcula el seu *hash*, al resultat se li prefixa una altra cadena de bits derivada de la clau, i es torna a calcular el *hash*. (Encara que s'hagi de cridar dues vegades la funció *hash*, la segona serà molt més ràpida perquè les dades a resumir seran més curtes.)

2.1.2. Signatures digitals

Els codis MAC, com que es basen en una clau secreta, només tenen significat per a qui conegui aquesta clau. Si A envia missatges a B autenticats amb una clau compartida, només B podrà verificar l'autenticitat d'aquests missatges.

D'altra banda, en cas d'un conflicte en què A denegués l'autoria d'un missatge autenticat, B no podria demostrar davant d'un tercer imparcial (per exemple, un àrbitre o un jutge) que el missatge el va generar A. Revelar la clau secreta no seria prova suficient ja que, pel fet de ser coneguda per les dues parts, sempre hi hauria la possibilitat que el missatge en disputa i el seu codi d'autenticació els hagués generat B.

En canvi, si A autentica els missatges adjuntant-hi la signatura digital calculada amb la seva clau privada, tothom podrà verificar-los amb la seva clau pública. Aquesta tècnica d'autenticació proporciona, com a efecte addicional, el servei de **no repudi**. Això vol dir que un destinatari B pot demostrar fefaentment davant d'un tercer que un missatge ha estat generat per A.

Com hem vist al subapartat 1.2., els algorismes de signatura digital usats normalment es basen en el càlcul d'un *hash* i un xifratge mitjançant una clau privada. Són exemples

d'algorismes de signatura l'RSA, el d'ElGamal, i l'estàndard DSA (*Digital Signature Algorithm*).

2.2. Autenticació d'entitat

L'autenticació d'entitat es fa servir quan en una comunicació una de les parts vol assegurar-se de la identitat de l'altra part. Típicament, aquesta autenticació és un requisit per permetre l'accés a un recurs restringit, com per exemple un compte d'usuari en un ordinador, diners en efectiu en un caixer automàtic, accés físic a un lloc, etc.

En general, les tècniques per a la identificació d'un usuari *A* poden estar basades en:

- Alguna cosa que *A sap*, com per exemple una contrasenya o una clau privada.
- Alguna cosa que *A té*, com per exemple una targeta amb banda magnètica o amb xip.
- Alguna cosa que *A és* o, dit d'una altra manera, alguna propietat inherent a *A*, com per exemple les seves característiques biomètriques.

La biometria és una disciplina relativament moderna que és possible que tingui una certa implantació en un futur proper. En aquest mòdul, però, ens centrarem en les tècniques basades en l'intercanvi d'informació per mitjans electrònics.

Tècniques biomètriques

Les tècniques biomètriques fan ús de característiques fisiològiques humanes (com l'empremta dactilar, l'iris, la retina, la cara o la mà) o característiques del comportament humà (com la parla, la signatura manual o la pulsació de tecles).

Una diferència entre l'autenticació de missatge i l'autenticació d'entitat és que la primera pot ser intemporal (és possible verificar l'autenticitat d'un document signat, per exemple, deu anys enrere), mentre que la segona normalment es realitza en temps real. Això vol dir que per a l'autenticació d'entitat es pot dur a terme un protocol interactiu, en què les dues parts s'intercanvien missatges fins que la identitat en qüestió queda confirmada.

A continuació veurem dos grups de tècniques que es poden utilitzar per a l'autenticació d'entitat:

- Les basades en **contrasenyes** (o *"passwords"*, en anglès), també anomenades tècniques d'autenticació feble.
- Les basades en **protocols de repte-resposta** (*"challenge-response"*, en anglès), també anomenades tècniques d'autenticació forta.

2.2.1. Contrasenyes

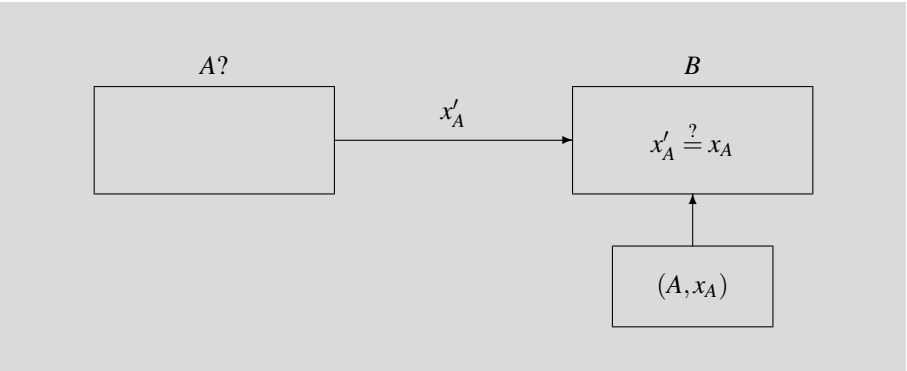
La idea bàsica de l'autenticació basada en contrasenyes és que l'usuari *A* envia la seva identitat (el seu identificador d'usuari, el seu nom de *login*, etc.) seguida d'una contrasenya secreta x_A (una paraula o combinació de caràcters que l'usuari pugui me-

moritzar). El verificador B comprova que la contrasenya sigui vàlida, i si ho és dóna per bona la identitat de A.

Hi ha diferents maneres de dur a terme aquesta autenticació, i també hi ha maneres diverses d'intentar atacar-la. Tot seguit veurem algunes variants de l'autenticació amb contrasenyes que intenten evitar determinats tipus d'atacs.

Llista de contrasenyes en clar

La manera més simple de comprovar una contrasenya és que el verificador tingui una llista de les contrasenyes associades als usuaris, és a dir, una llista de parells (A, x_A). Quan A envia la seva contrasenya, el verificador compara directament el valor rebut x'_A amb el que figura a la llista, x_A.



Si es tracta de les contrasenyes corresponents a usuaris d'un sistema informàtic, és evident que no poden estar guardades en un fitxer d'accés públic: és necessari que la llista estigui protegida contra lectura. Però si algú troba la manera de saltar-se aquesta protecció (cosa que de vegades passa en els sistemes multiusuari), automàticament tindrà accés a les contrasenyes de tots els usuaris.

A més, en aquests sistemes normalment hi ha un usuari administrador o "superusuari" que té accés a tots els fitxers, i per tant a les contrasenyes dels usuaris. Un superusuari que fos mal intencionat podria fer un mal ús d'aquest privilegi. O un superusuari que tingués un moment de descuit podria donar peu que un altre usuari se n'aprofités.

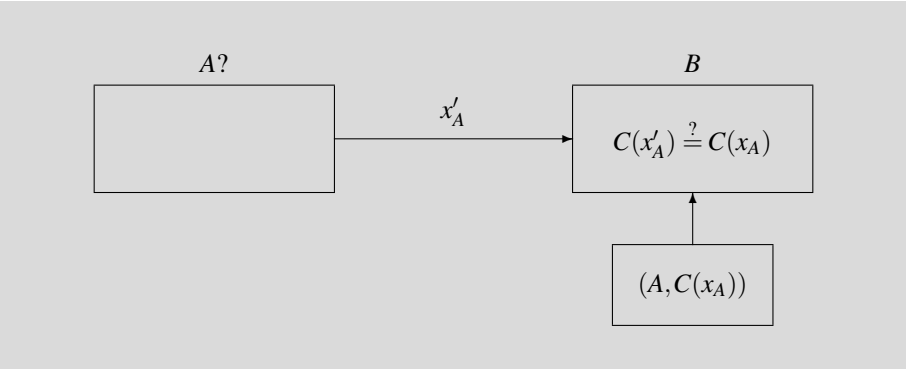
Llista de contrasenyes codificades

Una segona opció és que a la llista de contrasenyes, en comptes d'estar guardades en clar en parells (A, x_A), cadascuna estigui codificada amb alguna transformació C de manera que no se'n pugui deduir el valor real. Així, la llista ha de contenir parells $(A, C(x_A))$.

Aquesta codificació C podria ser per exemple un xifratge, però llavors caldria prendre les precaucions oportunes amb la clau de desxifratge (qui aconseguís accedir a aques-

ta clau podria obtenir totes les contrasenyes). El més habitual és que la codificació consisteixi a aplicar una funció unidireccional, com pot ser una funció *hash*.

Per a la verificació, B ha de calcular el valor transformat $C(x'_A)$ a partir de la contrasenya rebuda i comparar-lo amb el que consta a la llista.



Codificació de contrasenyes en Unix

Un cas molt conegut és el de la gestió de les contrasenyes en el sistema operatiu Unix, ja que la majoria de les variants de Unix usades actualment fan servir tècniques molt semblants, basades en les versions originals d'aquest sistema.

En Unix hi ha un fitxer on es guarden les contrasenyes dels usuaris codificades amb una funció unidireccional. Però en comptes d'una funció *hash*, el que es fa servir és un xifratge simètric: el text en clar és fix (una cadena de bits tots iguals a 0), com a clau de xifratge es fa servir la contrasenya, i el que es guarda a la llista és el text xifrat. Així s'aprofita la propietat dels algorismes criptogràfics de no poder deduir la clau a partir del text xifrat ni que es conegui el text en clar.

Com que la codificació de les contrasenyes és unidireccional, ningú (ni tan sols el superusuari) pot saber quina és la contrasenya de cada usuari encara que tingui accés a la llista. Això faria innecessària la protecció contra lectura del fitxer on hi ha la llista.

Però, encara que la codificació no sigui reversible, pot ser vulnerable a un altre tipus d'atac: la força bruta. Si l'atacant coneix la contrasenya codificada $C(x_A)$ i coneix l'algorisme de codificació C, pot provar de codificar possibles contrasenyes fins a trobar un resultat que coincideixi amb $C(x_A)$.

Espai de contrasenyes

Si considerem les possibles combinacions de, per exemple, 8 caràcters (suposant cada caràcter comprès entre els codis ASCII 32 i 126, és a dir, 95 caràcters diferents), n'hi ha 95^8 (aprox. $6.6 \cdot 10^{15}$). En canvi, hi ha estudis que indiquen que una gran part de les contrasenyes que escullen els usuaris es poden trobar en un diccionari de 150000 paraules, un espai 10^{10} vegades menor.

Hi ha un fet que afavoreix l'atacant en aquest cas: si els usuaris poden escollir les seves contrasenyes, normalment no faran servir combinacions arbitràries de caràcters sinó paraules fàcils de recordar. Per tant, l'espai de cerca es redueix considerablement. L'atacant pot, doncs, limitar-se a provar paraules d'un diccionari o combinacions derivades a partir d'aquestes paraules. Per això a aquest tipus d'atac se l'anomena **atac de diccionari**.

Programes "trencacontrasenyes"

Des de fa molts anys han existit els programes "trenca-contrasenyes", o *"password crackers"* en anglès. A un programa d'aquest tipus se li dóna una llista de contrasenyes codificades i un diccionari, i va provant cadascuna de les paraules, tant directament com amb diferents variacions: tot minúscules, tot majúscules, la primera lletra majúscula, amb les lletres a l'inrevés, afegint-hi una xifra numèrica,

afegint-n'hi dues, canviant certes lletres per xifres, etc. També pot provar combinacions amb dades addicionals com els identificadors dels usuaris, els seus noms, cognoms, etc.

Pot ser sorprenent la quantitat de contrasenyes que aquests programes poden arribar a endevinar en només unes poques hores.

Tècniques per dificultar els atacs de diccionari

A continuació veurem algunes possibles tècniques per fer més difícils els atacs de diccionari.

1) Ocultar la llista de contrasenyes codificades

Una primera solució és restringir l'accés a la llista de contrasenyes, protegint-la contra lectura. Encara que a la llista hi hagi les contrasenyes codificades, l'accés a aquesta llista per part d'un atacant li permet realitzar còmodament un atac de diccionari.

Llista de contrasenyes codificades en Unix

En les versions antigues del sistema Unix, les contrasenyes codificades estaven al fitxer `/etc/passwd`. Aquest mateix fitxer s'aprofitava per guardar-hi altres dades sobre el compte de cada usuari: el seu directori inicial, el seu *shell*, el nom de l'usuari, etc. Com que aquesta informació és d'accés públic, el fitxer `/etc/passwd` tenia permís de lectura per a tots els usuaris.

A les versions modernes de Unix continua havent-hi un fitxer `/etc/passwd` de lectura pública amb informació sobre els comptes dels usuaris, però les contrasenyes codificades ja no es guarden en aquest fitxer sinó en un altre, `/etc/shadow`, sobre el qual cap usuari (tret del superusuari) té permís de lectura.

Si l'atacant no té accés a la llista de contrasenyes, ja no podrà realitzar un atac "fora de línia" (*off-line*), és a dir, un atac realitzat en un ordinador escollit per l'atacant a la seva conveniència (per exemple, un ordinador potent, on ningú pugui veure què està fent, etc.). No li quedarà més remei que realitzar l'atac "en línia" (*on-line*), és a dir, directament amb el verificador real (per exemple, el programa `login` del sistema que vol atacar), enviant-li contrasenyes per veure si les accepta o no.

Si el verificador veu que hi ha algú que està fent proves d'autenticació fallides, això és un senyal que pot tractar-se d'un atac en línia. Llavors pot prendre certes mesures per contrarestar aquest atac. Per exemple, es pot fer que si es rep un cert nombre de contrasenyes invàlides consecutives, el recurs associat quedi bloquejat.

Protecció contra atacs a les targetes amb PIN

El mecanisme del bloqueig s'utilitza típicament en els mètodes d'autenticació basats en dispositius físics, com les targetes bancàries o els mòduls SIM (*Subscriber Identity Module*) dels telèfons mòbils. Aquests dispositius requereixen que l'usuari introdueixi un PIN (*Personal Identification Number*), que fa les funcions de contrasenya.

Per motius històrics i de conveniència dels usuaris, aquest PIN és un nombre de longitud curta, per exemple de 4 xifres. Llavors, per tal d'evitar els atacs de força bruta (que només necessitarien 10000 intents com a màxim), el dispositiu es bloqueja, per exemple, al tercer intent equivocat.

La protecció que proporciona el bloqueig pot ser un inconvenient per a l'usuari legítim que no té cap culpa que hi hagi algú intentant descobrir la seva contrasenya. Per evitar-li aquest inconvenient, una possibilitat és que cada usuari tingui

associada una altra contrasenya de desbloqueig, molt més difícil d'endevinar (per exemple, un PIN de 8 xifres per desbloquejar el PIN de 4 xifres).

Una altra possibilitat és permetre múltiples intents d'autenticació, però en comptes de respondre immediatament amb un missatge de "contrasenya incorrecta", demorar aquesta resposta amb un temps de retard, que anirà creixent a mesura que el mateix usuari vagi enviant més contrasenyes errònies. Això alentiria tant un atac que el faria inviable a partir d'uns quants intents. I quan l'usuari legítim faci servir la seva contrasenya correcta, el temps de resposta serà el normal.

2) Regles per evitar contrasenyes fàcils

La solució d'ocultar la llista de contrasenyes codificades dóna una seguretat semblant a la de la llista de contrasenyes en clar. Si algú descobreix la llista (malgrat la protecció contra lectura), pot realitzar sense més problemes un atac de diccionari.

Una altra manera de dificultar aquest atac és obligar els usuaris a escollir contrasenyes que compleixin unes determinades regles perquè no siguin fàcils d'endevinar. Per exemple:

- Que la contrasenya tingui una longitud mínima.
- Que no sigui tot lletres ni tot nombres.
- Que les lletres no coincideixin amb cap paraula de diccionari ni amb combinacions trivials de les paraules (com escriure les lletres a l'inrevés, etc.).
- Que la contrasenya no es derivi de l'identificador de l'usuari, del seu nom, cognom, etc.
- Etc.

Amb això s'aconsegueix que l'espai de contrasenyes on fer la cerca sigui més gran del que és habitual, i l'atac de diccionari necessiti més temps.

D'altra banda, també és cert que com més temps mantingui un usuari la seva contrasenya sense canviar-la, més gran serà la probabilitat que un atacant aconsegueixi descobrir-la. Per això és recomanable que els usuaris renovin les contrasenyes periòdicament. Hi ha sistemes que implanten una política de canvi forçat de contrasenyes. A partir d'un cert període (per exemple, 90 dies) la contrasenya es considera caducada i l'usuari l'ha de canviar per una de diferent.

3) Afegir complexitat a la codificació de les contrasenyes

Una altra solució per dificultar els atacs de diccionari és alentir-los fent que cada contrasenya costi més de codificar. Per exemple, si l'algorisme de codificació no és directament una funció *hash* sinó un bucle de N crides a la funció *hash*, provar cada contrasenya costa N vegades més.

El valor N es pot escollir tan gran com es vulgui, però tenint en compte que si és massa gran els usuaris legítims també podran notar que l'autenticació normal és més lenta.

Caducitat de les contrasenyes

Un temps de caducitat de les contrasenyes massa curt també podria ser contraproduent, perquè si els usuaris han de pensar moltes contrasenyes en poc temps, poden acabar oblidant quina era l'última, o anotant-la en un paper, amb la qual cosa la contrasenya és molt més vulnerable.

Complexitat de la codificació de les contrasenyes en Unix

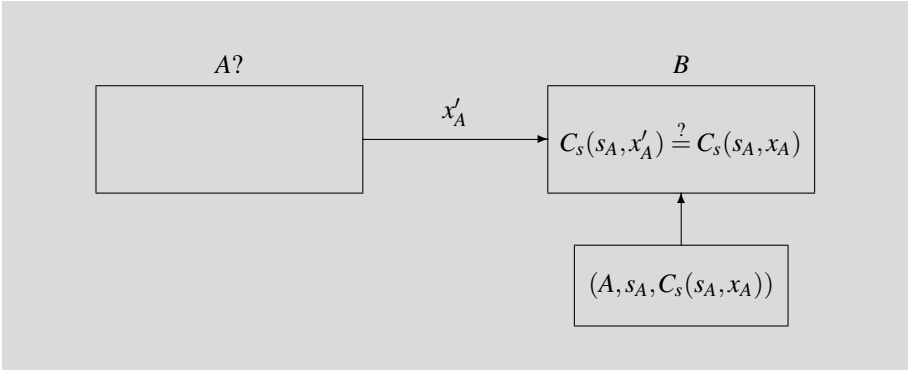
La solució que es va adoptar en Unix va ser repetir N vegades el xifratge per obtenir la contrasenya codificada, i es va escollir el valor $N = 25$.

Per tant, encara que l'atacant disposi d'una implementació ràpida de l'algorisme de xifratge, el temps esperat per trencar una contrasenya es multiplica per 25.

4) Afegir bits de sal a la codificació de les contrasenyes

Al subapartat 1.1. hem vist que, per tal de variar el resultat del xifratge, encara que es faci servir la mateixa clau amb el mateix text en clar, hi ha la possibilitat d'introduir els anomenats "bits de sal" per modificar la clau. De la mateixa manera, es pot definir un algorisme de codificació C_s que, a més de la contrasenya, tingui com a entrada uns bits de sal.

Així, cada vegada que un usuari A canviï la seva contrasenya x_A, es generen aleatòriament els bits de sal s_A i es guarden aquests bits juntament amb la contrasenya codificada $C_s(s_A, x_A)$. A l'hora de verificar, es torna a calcular aquest valor a partir dels mateixos bits de sal i es compara amb el valor guardat.



Aquesta tècnica no complica la verificació d'una contrasenya ni un atac de diccionari contra aquesta contrasenya específica. Però si l'atac de diccionari es realitza sobre diverses contrasenyes alhora (per exemple, sobre la llista sencera de tots els usuaris, com fan normalment els programes trencacontrasenyes), el temps de càlcul es multiplica pel nombre total de contrasenyes, suposant que totes estiguin codificades amb bits de sal diferents.

Això és així perquè, sense bits de sal, es pot agafar cada paraula del diccionari, codificar-la, i comparar el resultat amb totes les contrasenyes codificades. En canvi, si hi ha bits de sal, cal fer una codificació separada per cada contrasenya, cadascuna amb els seus bits de sal corresponents.

Amb els bits de sal també s'evita una variant més eficient de l'atac de diccionari, en la qual l'atacant no va provant una a una les paraules, sinó que fa servir una llista preconstruïda amb les paraules del diccionari ja codificades. Si per exemple el diccionari conté 150000 paraules, quan no es fan servir bits de sal l'atacant només ha de buscar la contrasenya codificada en una llista de 150000 paraules codificades. Però si es fan servir 12 bits de sal, la mida de la llista es multiplicaria per més de 4000 ($2^{12} = 4096$), i l'atacant hauria de treballar amb una llista de més de 600 milions de paraules codificades.

Un altre avantatge dels bits de sal és que si, per casualitat, dos usuaris escullen la mateixa contrasenya, els valors codificats de les seves contrasenyes seran diferents (si els bits de sal també són diferents), i encara que veiessin la llista de contrasenyes no sabrien que tots dos tenen la mateixa.

Bits de sal a les contrasenyes en Unix

En Unix es fan servir precisament 12 bits aleatoris de sal (que s'obtenen a partir dels bits de menys pes de l'hora en el moment en què l'usuari es canvia la contrasenya). Això vol dir que hi ha 4096 valors possibles d'aquests bits. Una llista completa de contrasenyes codificades a partir d'un diccionari de 150000 paraules hauria de contenir, doncs, més de 600 milions d'entrades. Això podria ser una quantitat descomunal en l'època en què es va desenvolupar el sistema Unix, però actualment una llista d'aquestes dimensions cap perfectament en el disc dur de qualsevol ordinador personal.

D'altra banda, l'algorisme de xifratge que es fa servir en Unix per obtenir la codificació de les contrasenyes és el DES. Com hem vist abans, per codificar una contrasenya s'aplica un xifratge a un text format per tots els bits a 0, i s'usa la contrasenya com a clau de xifratge (això explica per què les versions estàndard de Unix només fan servir contrasenyes de fins a 8 caràcters). En aquest cas, els bits de sal no s'utilitzen per modificar la clau, sinó que serveixen per permutar certs bits dels resultats intermedis que s'obtenen a cada iteració de l'algorisme DES.

Amb això s'aconsegueix un altre objectiu: que un atacant no pugui fer servir directament una implementació eficient de l'algorisme DES (les més eficients són els xips DES, és a dir, les implementacions amb maquinari), sinó que hagi d'introduir-hi modificacions (si es tracta d'un xip DES, calen uns certs recursos que no estan a l'abast de qualsevol).

5) Ús de *passphrases*

La propietat que aprofiten els atacs de diccionari és que el conjunt de contrasenyes que utilitzen normalment els usuaris és un subconjunt molt petit de tot l'espai de contrasenyes possibles.

Per ampliar aquest espai es pot fer que l'usuari no utilitzi paraules relativament curtes, d'uns 8 caràcters, sinó frases més llargues, anomenades *"passphrases"*. La codificació d'aquestes *passphrases* pot continuar sent una funció unidireccional, com per exemple una funció *hash* (amb la mateixa longitud que en el cas de les contrasenyes). La diferència és que, si abans la llista de contrasenyes codificades contenia valors d'entre un total d'uns 150000 diferents, amb *passphrases* contindrà moltíssims més valors possibles, i la cerca exhaustiva de l'atac de diccionari serà molt més llarga.

Contrasenyes d'un sol ús

Tots els esquemes d'autenticació amb contrasenyes que hem vist fins ara, a part dels atacs de diccionari, són vulnerables als anomenats **atacs de repetició** o de *"replay"*. Per dur a terme aquest atac, cal interceptar la comunicació entre A i B durant una autenticació, per exemple utilitzant un *sniffer*, i veure quin és el valor x_A enviat. Llavors l'atacant Z només ha de realitzar una autenticació davant B enviant-li aquest mateix valor x_A, i B es pensarà que es tracta de l'autèntic usuari A.

Sniffers

Vegeu els *sniffers* al mòdul didàctic "Vulnerabilitat en xarxes TCP/IP" d'aquesta assignatura.

Aquest tipus d'atac s'evita amb els protocols d'autenticació forta que veurem tot seguit. Però hi ha una altra tècnica que, tot i que es pot considerar basada en contrasenyes, té algunes propietats de l'autenticació forta, entre elles la resistència als atacs de repetició. Aquesta tècnica és la de les anomenades **contrasenyes d'un sol ús** (*"one-time passwords"*).

Com el seu nom indica, les contrasenyes d'un sol ús són contrasenyes que només són vàlides una vegada, i la vegada següent cal fer servir una contrasenya diferent. Així,

encara que l'atacant vegi quin valor s'està enviant per a l'autenticació, no li servirà de res perquè ja no el podrà tornar a fer servir.

Hi ha diferents possibilitats per implementar l'autenticació amb contrasenyes d'un sol ús. Per exemple, podem considerar les següents:

- Llista de contrasenyes compartida: A i B es posen d'acord, de manera segura (és a dir, no a través d'un canal que pugui ser interceptat), en una llista de N contrasenyes. Si es tracta d'una llista ordenada, cada vegada que A hagi d'autenticar farà servir la següent contrasenya de la llista. Alternativament, les contrasenyes poden tenir associat un identificador: a cada autenticació B escull una contrasenya de les que no s'hagin fet servir encara, envia a A el seu identificador, i A ha de respondre amb la contrasenya corresponent.

- Contrasenyes basades en una funció unidireccional (típicament una funció *hash*): A escull un valor secret x_0 i fa servir una funció unidireccional h per calcular els següents valors:

$$\begin{aligned} x_1 &= h(x_0) \\ x_2 &= h(x_1) = h^2(x_0) \\ &\vdots \\ x_N &= h(x_{N-1}) = h^N(x_0) \end{aligned}$$

  i envia el valor x_N a B (B ha d'assegurar-se que ha rebut aquest valor de l'usuari A autèntic).

  Llavors A inicialitza un comptador i a N. Cada vegada que s'hagi d'autenticar, A enviarà el valor x_{i-1}, i B aplicarà la funció unidireccional a aquest valor, el compararà amb el que té guardat, i comprovarà si $h(x_{i-1}) = x_i$. Si es compleix aquesta igualtat, la contrasenya és vàlida. La següent vegada, B substitueix el valor que tenia guardat, x_i, pel que acaba de rebre, x_{i-1}, i A actualitza el comptador i decrementant-lo en 1.

  Com que la funció h és unidireccional, ningú que vegi el valor x_{i-1} sabrà quin és el valor que tocarà la propera vegada (x_{i-2}), ja que A va obtenir x_{i-1} com a $h(x_{i-2})$, i fer el càlcul invers (és a dir, obtenir x_{i-2} a partir de x_{i-1}) ha de ser computacionalment inviable.

La tècnica de les contrasenyes basades en una funció *hash* és més eficient que la de la llista compartida, perquè B només ha de recordar l'últim valor x_i rebut.

Implementacions de les contrasenyes d'un sol ús

Existeixen diverses implementacions de contrasenyes basades en una funció *hash*, com per exemple **S/KEY** o **OPIE**. Amb aquests programes, quan l'usuari A està connectat de manera segura (per exemple localment, és a dir, davant la consola) al sistema servidor on es vol autenticar, pot generar una llista de N contrasenyes, imprimir-la, i portar-la a sobre cada vegada que hagi de fer una autenticació des d'un sistema remot. Per facilitar la introducció de les contrasenyes, el programa converteix els bits en una seqüència de paraules curtes, per exemple:

```
...
90: BAND RISE LOWE OVEN ADEN CURB
91: JUTE WONT MEEK GIFT OWL PEG
92: KNOB QUOD PAW SEAM FEUD LANE
...
```

La traducció es fa a partir d'una llista de 2048 paraules de fins a 4 caràcters, i a cadascuna se li assigna una combinació d'11 bits.

L'usuari *A* no necessita recordar el valor del comptador *i*, ja que aquest valor es guarda en el servidor. A cada autenticació, el servidor envia el valor actual del comptador, l'usuari el fa servir per consultar la seva llista, i respon amb la contrasenya corresponent. La següent vegada, el servidor actualitzarà el comptador decrementant-lo en 1.

2.2.2. Protocols de repte-resposta

El problema que tenen els esquemes d'autenticació basats en contrasenyes és que cada vegada que es vol realitzar l'autenticació s'ha d'enviar el mateix valor al verificador (tret de les contrasenyes d'un sol ús, com acabem de veure). Qualsevol atacant que aconsegueixi interceptar aquest valor fix podrà suplantar la identitat de l'usuari a qui correspon la contrasenya.

Hi ha un altre grup de mecanismes en què el valor que s'envia per a l'autenticació no és fix, sinó que depèn d'un altre, generat pel verificador. Aquest últim valor s'anomena **repte**, i s'ha d'enviar a l'usuari *A* com a primer pas per a la seva autenticació. Llavors *A*, fent ús d'una clau secreta, calcula una **resposta** a partir d'aquest repte, i l'envia al verificador *B*. Per això aquests mecanismes d'autenticació reben el nom de **protocols de repte-resposta**.

L'algorisme per calcular la resposta ha de garantir que no es pugui obtenir sense saber la clau secreta. Això permet al verificador confirmar que la resposta només ha pogut enviar-la *A*. Si es fa servir un repte diferent cada vegada, un atacant no pot treure profit de la informació que descobreixi interceptant la comunicació.

Depenent del protocol, el verificador pot generar els reptes de diverses maneres:

- Seqüencialment: en aquest cas el repte és simplement un nombre que es va incrementant cada vegada (el més normal és incrementar-lo d'un en un), i que per tant no es repetirà mai.

- Aleatòriament: el repte pot ser generat amb un algorisme pseudoaleatori, però la propietat que té en aquest cas és que no és predictible per als atacants.

- Cronològicament: el repte s'obté a partir de la data i hora actuals (amb la precisió que sigui adequada per al protocol). Aquest tipus de repte també s'anomena **marca d'hora** o *"timestamp"*. El receptor, és a dir, qui vol validar la identitat, pot utilitzar la marca d'hora per saber si es tracta d'una nova autenticació, o si algú vol reutilitzar els missatges d'una altra autenticació per intentar un atac de repetició.

L'avantatge dels reptes cronològics és que per obtenir-los només cal un rellotge, que segurament estarà disponible en qualsevol sistema, mentre que amb els seqüencials i els aleatoris cal mantenir informació d'estat (el nombre de seqüència o l'entrada per calcular el següent nombre pseudoaleatori) per evitar repeticions. L'inconvenient és que cal una certa sincronització entre els rellotges. Si es fan servir tècniques com un servidor de temps que dóna l'hora actual, també cal verificar que l'hora obtinguda sigui autèntica, és a dir, que un atacant no ens està enganyant fent-nos creure que és l'hora que l'interessa.

Sincronització de rellotges

Si el repte s'obté a partir d'un rellotge, depenent del protocol pot ser necessari que emissor i receptor tinguin els seus rellotges sincronitzats, o bé que es permeti una certa tolerància entre l'hora que ha enviat l'emissor i la que diu el rellotge del receptor. La tolerància ha de ser tal que permeti diferències entre els rellotges però que no doni temps a un tercer a realitzar un atac de repetició.

Dispositius per al càlcul de les respostes

El càlcul de la resposta es pot fer amb algun programari dissenyat a aquest efecte, o també es pot utilitzar un dispositiu, com una mena de petita calculadora, que guarda la clau secreta. L'usuari introdueix el repte pel teclat d'aquesta calculadora, i per la pantalla apareix la resposta.

Alternativament, si el repte és cronològic, el dispositiu no té teclat sinó que va mostrant per la pantalla les respostes en funció de l'hora actual (per exemple, cada minut), en sincronia aproximada amb el rellotge del verificador.

Per a més seguretat, el dispositiu també pot estar protegit amb un PIN.

Els protocols de repte-resposta es poden classificar en dos grups:

- Els basats en tècniques simètriques, en les quals la clau secreta és compartida per l'usuari A i el verificador B.
- Els basats en tècniques de clau pública, en les quals A fa servir una clau privada per calcular la resposta.

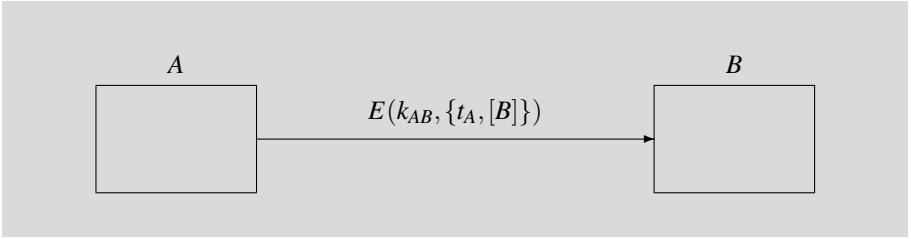
En la descripció dels protocols que veurem a continuació, farem servir la notació $\{a,b,\ldots\}$ per representar un missatge que conté els components $a,b,\ldots$, i si algun d'aquests components és opcional el representarem entre claudàtors, com per exemple $\{a,[b]\}$.

Protocols de repte-resposta amb clau simètrica

Si el protocol de repte-resposta es basa en una clau k_{AB} compartida per A i B, hi ha diverses maneres d'obtenir aquesta clau. Per exemple, la poden haver acordat directament A i B d'una manera segura, o bé la poden demanar a un servidor de claus centralitzat.

1) Autenticació amb marca de temps

El protocol més simple és el que fa servir com a repte implícit (no cal l'enviament del repte) l'hora actual. L'usuari A obté una marca de temps t_A i l'envia al verificador B xifrada amb la clau compartida.



El verificador desxifra el missatge rebut i comprova si la marca de temps és acceptable, és a dir, si està dins la tolerància de sincronització i no està repetida.

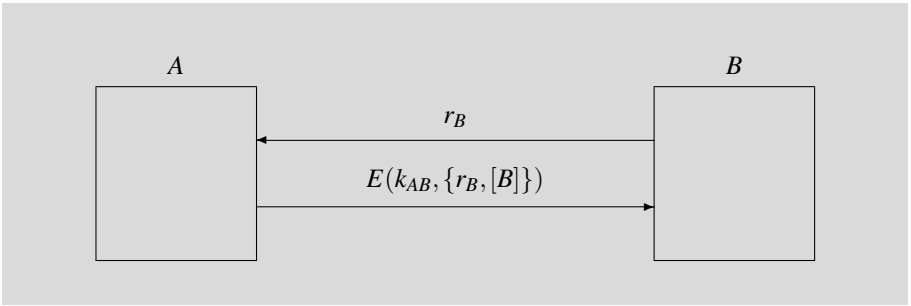
Si es fa servir la mateixa clau en els dos sentits (de A a B i de B a A), el fet d'incloure la identitat del verificador B evita que, en un procés d'autenticació mútua, un atacant intenti fer-se passar per B davant de A repetint el missatge en sentit contrari.

Detecció de repeticions

Per saber si s'està reutilitzant una marca de temps, el verificador necessita recordar quines s'han fet servir ja, però només les que estiguin dins de l'interval de tolerància. Passat aquest interval ja es pot esborrar la marca de la llista.

2) Autenticació amb números aleatoris

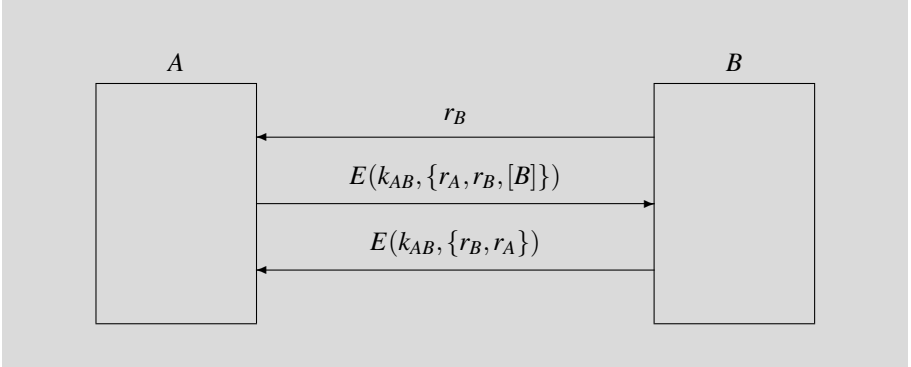
En aquest cas cal enviar explícitament el repte, que consisteix en un número aleatori r_B generat per B. La resposta és el repte xifrat amb la clau compartida.



El verificador desxifra el missatge i comprova que el nombre aleatori que conté és igual al repte. El fet d'incloure la identitat del verificador B evita que el missatge es pugui utilitzar en sentit contrari si alguna vegada A genera com a repte el mateix nombre r_B.

3) Autenticació mútua amb nombres aleatoris

Amb tres missatges, el protocol anterior es pot convertir en un protocol d'autenticació mútua, és a dir, de A davant B i de B davant A. En aquest cas, A ha de generar un altre nombre aleatori r_A, que actuarà com a repte en sentit invers, i incloure'l a la seva resposta.



Quan B desxifra la resposta, a més de comprovar que el valor r_B sigui l'esperat, també obté r_A, que haurà d'utilitzar per enviar la seva resposta a A. Llavors A comprova que els dos nombres aleatoris r_A i r_B tinguin els valors correctes.

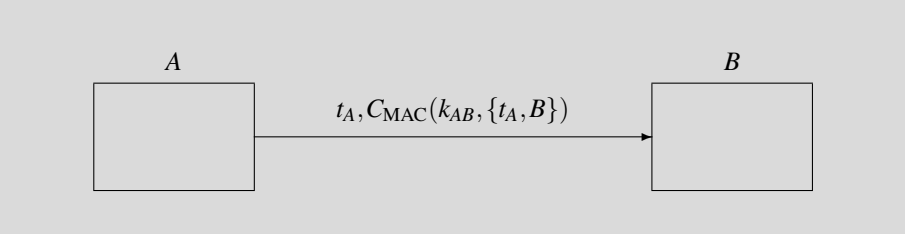
Com abans, el fet d'incloure l'identificador de B en el segon missatge evita atacs de repetició en sentit contrari.

4) Autenticació amb funció unidireccional

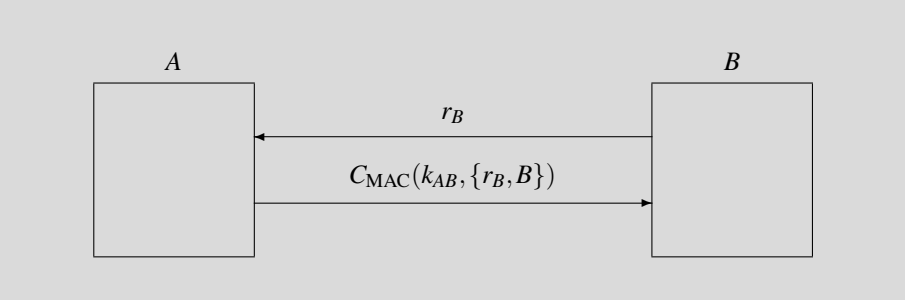
Els protocols anteriors fan ús d'un algorisme de xifratge simètric, però també és possible utilitzar funcions unidireccionals amb clau secreta, com per exemple els algorismes de MAC. Per a la verificació, en comptes de desxifrar cal comprovar que el MAC sigui correcte, i per tant els valors necessaris per calcular-lo s'han d'enviar en clar.

A continuació hi ha les variants dels protocols anteriors quan es fa servir un algorisme de MAC en lloc d'un de xifratge.

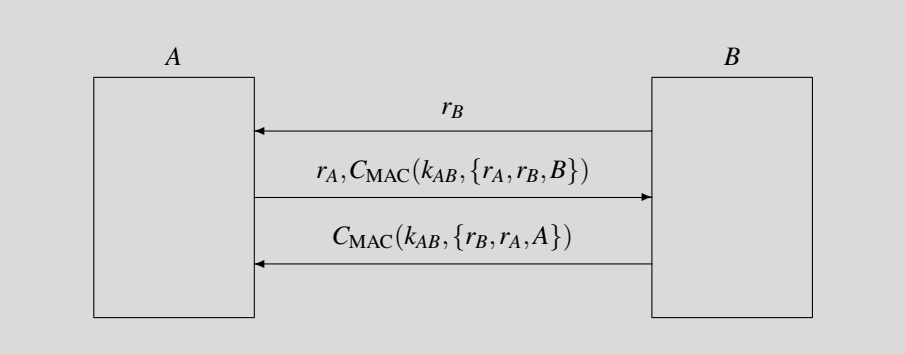
- Autenticació amb marca de temps:



- Autenticació amb nombres aleatoris:



- Autenticació mútua amb nombres aleatoris:



En aquest cas, cal afegir l'identificador del destinatari A en el càlcul de l'últim missatge per evitar atacs de repetició en sentit contrari, ja que ara els atacants poden veure el valor r_A.

Protocols de repte-resposta amb clau pública

Hi ha dues maneres d'utilitzar les tècniques de clau pública en els protocols de repte-resposta:

- El repte s'envia xifrat amb clau pública i la resposta és el repte desxifrat amb la corresponent clau privada.
- El repte s'envia en clar i la resposta és la signatura del repte.

Com que l'usuari A ha de fer servir la seva clau privada sobre un missatge que li han enviat, ha de prendre certes precaucions per tal d'evitar que l'altra part faci un ús il·legítim de la resposta. Això ho veurem en cadascun dels dos tipus de protocols de repte-resposta amb clau pública.

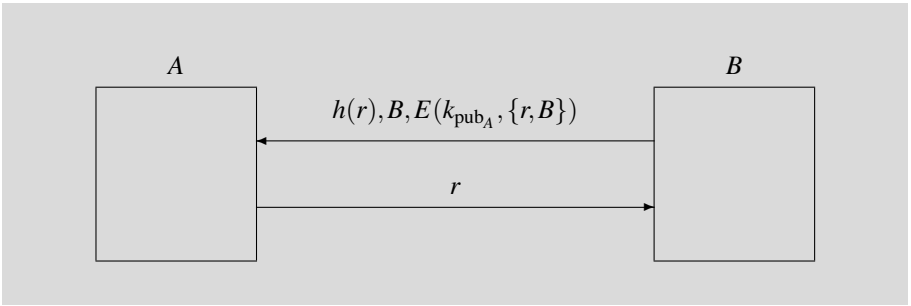
Claus per a aplicacions diferents

Per evitar els abusos que es poden cometre contra una clau pública, és molt recomanable fer servir claus públiques diferents per a cada aplicació, per exemple una clau per rebre missatges confidencials i una altra per a l'autenticació.

1) Desxifratge del repte

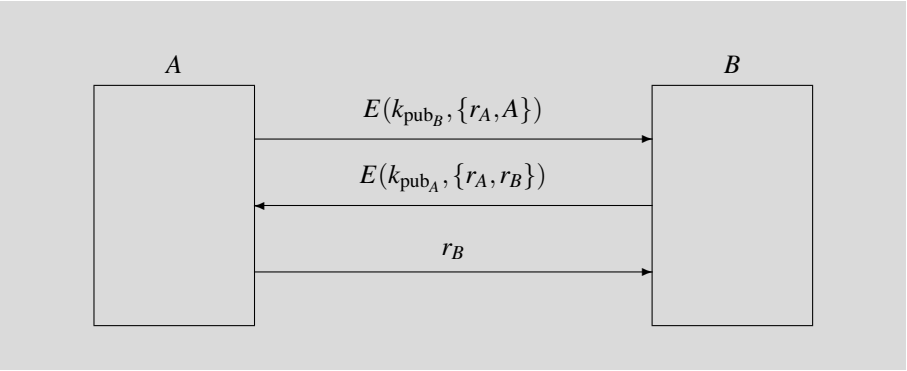
Si A rep un repte xifrat amb la seva clau pública, abans d'enviar la resposta ha d'assegurar-se que qui ha enviat el repte coneix el seu valor. Si no, un atacant pot enviar-li a A un repte xifrat, fent veure que l'ha generat aleatòriament, però que en realitat ha tret d'un altre missatge confidencial que algú havia enviat a A xifrat amb la seva clau pública. Si A respon a aquest repte, està donant a l'atacant el missatge confidencial desxifrat.

Una possibilitat és que A rebi, juntament amb el repte r xifrat, un resum o *hash* d'aquest repte.



Quan A rep el missatge, fa servir la seva clau privada per obtenir r i B, i comprova que aquests valors siguin correctes. Si el *hash* del repte r coincideix amb el valor rebut $h(r)$, vol dir que B coneix aquest repte. Per tant, se li pot enviar a B com a resposta el valor desxifrat. Si no, no se li envia res perquè pot tractar-se d'un defraudador que vol obtenir il·legítimament aquest valor desxifrat.

Una altra possibilitat és que A pugui escollir una part del repte. Aquest és el principi de l'anomenat protocol modificat de clau pública de Needham-Schroeder, que a més proporciona autenticació mútua.



Per a A, el repte està format per r_A i r_B: el fet que la primera part l'hagi generat A evita que el repte hagi estat maliciosament escollit per B. Per a B, el repte inclou l'identificador de A, amb la qual cosa tampoc pot ser un missatge xifrat que algú altre havia enviat a B i que A vol desxifrar il·legítimament.

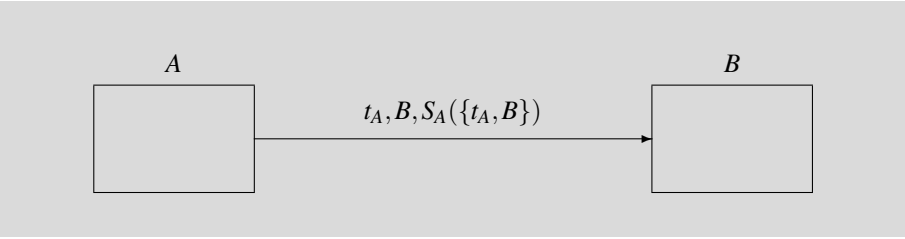
2) Signatura del repte

La Recomanació X.509 no solament especifica els formats de certificats i llistes de revocació que hem vist al subapartat 1.3., sinó que també defineix uns protocols d'autenticació forta basats en signatures digitals que fan ús de marques de temps i nombres aleatoris. Aquí mostrarem uns altres protocols, equivalents als basats en claus simètriques que hem vist anteriorment.

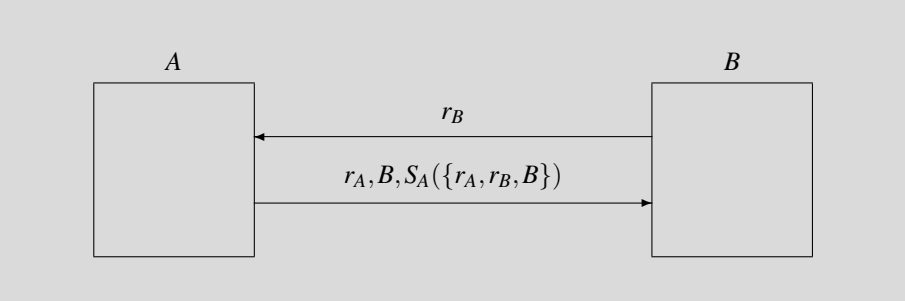
Com en el cas del desxifratge amb clau privada, A ha d'anar amb compte amb el que signa: mai no ha de signar directament un repte que li hagin enviat, sinó que en la signatura sempre ha d'intervenir com a mínim una part de text que hagi escollit el mateix A.

En la descripció d'aquests protocols, $S_A(M)$ vol dir la signatura digital del missatge M amb la clau privada de A.

- Autenticació amb marca de temps:

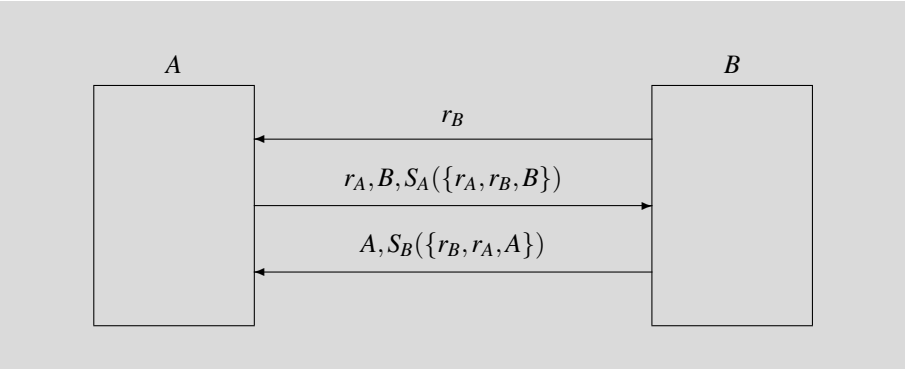


- Autenticació amb nombres aleatoris:



Aquí la inclusió del valor r_A evita que B hagi escollit r_B maliciosament per fer signar a A un missatge sense que se n'adoni.

- Autenticació mútua amb nombres aleatoris:



3. Protecció del nivell d'enllaç: xarxes sense fil

En els apartats anteriors hem vist els mecanismes bàsics de protecció, que proporcionen serveis com ara la confidencialitat o l'autenticació.

A l'hora d'aplicar aquests mecanismes a les xarxes de computadors, hi ha diverses opcions pel que fa al nivell de les comunicacions on s'introdueixin les funcions de seguretat.

- La protecció a nivell **físic** consisteix a assegurar que els atacants no puguin interferir en el medi de comunicació o en els equips de transmissió (cables, repetidors, etc.). Això inclouria, per exemple, impedir que es pugui "punxar" físicament el cable per capturar o injectar trànsit, o tancar els equips de commutació en una sala amb accés físic restringit. En aquest mòdul no entrarem a considerar aquest tipus de protecció.

- La protecció del **nivell d'enllaç** té per objectiu evitar que les trames que s'envien en els nivells baixos de la comunicació puguin ser interceptades o modificades. Dins d'aquesta categoria prenen especial rellevància els mecanismes per a protegir el trànsit de les xarxes sense fil, ja que el medi de transmissió utilitzat en aquest cas, és a dir l'aire, no es pot "protegir" físicament.

- La protecció a **nivell de xarxa** garanteix que les dades que s'enviïn als protocols de nivell superior, com TCP o UDP, es transmetran protegides. L'inconvenient és que pot ser necessari adaptar la infraestructura de la xarxa, i en particular els encaminadors (*routers*), perquè entenguin les extensions que cal afegir al protocol de xarxa (IP) per proporcionar aquesta seguretat.

- La protecció a **nivell de transport**, per la seva banda, té l'avantatge que només cal adaptar les implementacions dels protocols (TCP, UDP, etc.) que hi hagi en els nodes extrems de la comunicació, que típicament solen estar incorporades en el sistema operatiu o en biblioteques especialitzades. En aquest cas, doncs, només seria necessari un canvi en el programari.

- La protecció a **nivell d'aplicació** pot respondre millor a les necessitats de certs protocols. Un exemple concret és el del correu electrònic, on interessa protegir les dades d'aplicació, és a dir, els missatges de correu, més que no pas els paquets a nivell de transport o de xarxa. Això és així perquè un missatge és vulnerable a atacs d'accés il·legítim o de falsificació no només quan s'està transmetent per la xarxa sinó també quan està emmagatzemat a la bústia del destinatari.

En aquesta secció ens centrarem en la protecció de les trames que s'envien en les xarxes de comunicació sense fil. El problema que s'ha de resoldre és específic d'aquest tipus de xarxes, ja que, a diferència de les xarxes amb fil, l'accés al medi de transmissió és lliure, en el sentit que no cal fer res especial per a connectar-s'hi físicament. Per exemple, qualsevol usuari que tingui un dispositiu Wi-Fi en mode monitor pot veure les trames que es transmeten al seu entorn, sense cap limitació més que la distància a l'estació emissora.

3.1. Conceptes bàsics de les xarxes Wi-Fi

L'estàndard més utilitzat actualment per a les comunicacions en xarxes locals sense fil (WLAN) és l'anomenat *IEEE 802.11*, més conegut com a *Wi-Fi*. La primera versió de l'especificació, que data del 1997, permetia comunicacions de fins a 2 Mbit/s. Des de llavors s'hi han anat afegint extensions que accepten velocitats màximes de transmissió cada vegada més altes: 11 Mbit/s (802.11b), 54 Mbit/s (802.11a i 802.11g) i 600 Mbit/s (802.11n).

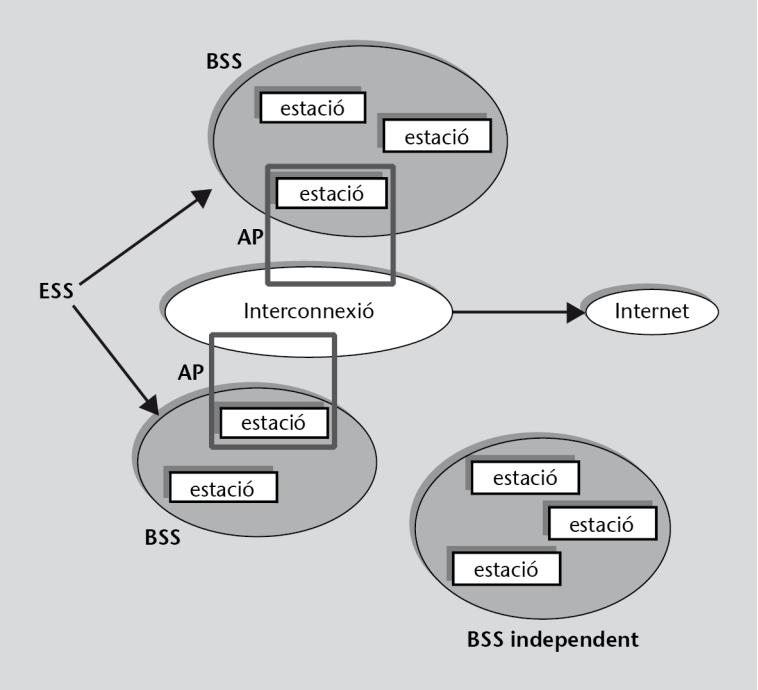
Un dels criteris de disseny inicials d'aquest estàndard era facilitar la interoperabilitat, cosa que no sempre ha estat del tot compatible amb la seguretat. Les primeres versions basaven la protecció de les comunicacions en el protocol WEP, que aviat es va demostrar que era massa feble. L'any 2004 es va publicar la norma IEEE 802.11i amb l'objectiu de corregir les deficiències del sistema de seguretat WEP.

WLAN

WLAN és la sigla de *wireless local area network*.

IEEE 802.11 permet la comunicació entre dispositius, anomenats **estacions**, que tinguin una interfície de xarxa sense fil. Cada interfície té una adreça MAC de 48 bits, amb el mateix format que les adreces Ethernet. Dues estacions o més que, per proximitat, poden comunicar-se entre si formen un *basic service set* (BSS). Es distingeixen dos tipus de BSS: els independents i els infraestructurals. Un **BSS independent**, també conegut com a xarxa *ad hoc*, és una xarxa aïllada en què les úniques comunicacions possibles són les directes d'una estació a una altra. En un **BSS infraestructural**, en canvi, hi ha una estació específica anomenada **punt d'accés** (AP) que permet la interconnexió amb altres xarxes, amb fil o sense. Un *extended service set* (ESS) és un conjunt d'un BSS infraestructural, o més, interconnectats mitjançant els seus AP. Des del punt de vista de les estacions, l'ESS funciona com si fos un únic BSS.

Components de les xarxes Wi-Fi



Els BSS s'identifiquen amb un BSSID, que en els infraestructurals és l'adreça MAC del seu AP. Els ESS tenen un identificador de format lliure de fins a 32 bytes. S'utilitza el terme genèric *service set identifier* (SSID) per a referir-se a l'identificador d'un BSS independent o d'un ESS.

Les estacions poden entrar i sortir d'un BSS de manera dinàmica. En un BSS infraestructural l'AP anuncia la seva presència enviant periòdicament **trames balisa**, típicament cada 100 ms. Els diferents camps d'una balisa indiquen l'SSID de la xarxa, les velocitats de transmissió que permet, etc.

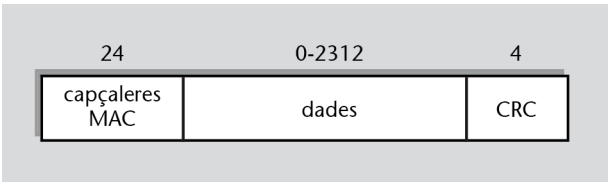
Una estació passa a ser membre d'un BSS infraestructural quan estableix una **associació** amb l'AP corresponent. En cada moment una estació només pot estar associada a un AP. Un cop establerta l'associació, l'estació normalment envia i rep totes les seves trames per mitjà d'aquest AP. Abans, però, per a poder fer l'associació i entrar al BSS cal que prèviament l'estació hagi fet una **autenticació** davant l'AP.

Configuracions Wi-Fi

La configuració típica de les xarxes Wi-Fi domèstiques és la d'un encaminador que d'una banda dóna accés a Internet via ADSL, i d'altra banda actua com a AP permetent la connexió des de les estacions que es trobin en el seu radi d'abast. En aquesta configuració hi ha un ESS format per un únic BSS. En una xarxa Wi-Fi corporativa, en canvi, és habitual tenir diversos AP en diferents parts d'un edifici: en aquest cas tots els BSS normalment pertanyen a un mateix ESS.

Les trames que poden enviar i rebre les estacions Wi-Fi pertanyen a un d'aquests tres tipus: trames de **gestió**, trames de **dades**, i trames de **control**. Les trames de gestió inclouen entre d'altres les balises, les trames d'autenticació i desautenticació, i les d'associació i dissociació.

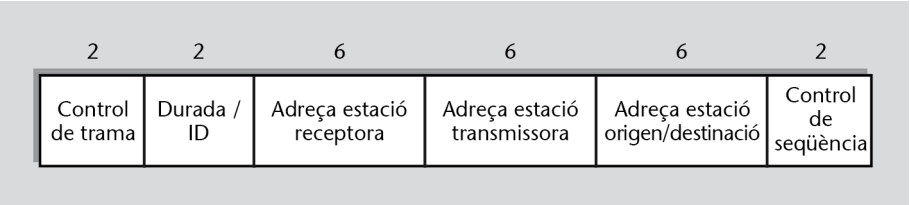
Format de les trames de dades



Les trames de dades que es transmeten en un BSS infraestructural consten de les parts següents:

- La capçalera MAC, de 24 bytes, amb l'estructura que es descriu a continuació.
- Les dades que s'envien en la trama, amb una longitud de fins a 2.304 bytes, o 2.312 si les dades inclouen encapsulament WEP.
- Un codi de comprovació d'errors, que és un codi CRC de 32 bits calculat sobre la capçalera i les dades.

Camps de la capçalera MAC d'una trama de dades



La capçalera MAC d'aquestes trames està formada pels camps següents:

- **Control de trama**: inclou diversos subcamps com ara la versió del protocol, tipus i subtipus de trama, un indicador (*flag*) per a indicar si és l'últim fragment d'una trama fragmentada, etc.
- **Durada/ID**: en una trama de dades aquest camp s'utilitza per a indicar el temps en què s'ha de transmetre una trama de control ACK.
- **Adreça de l'estació receptora**: indica l'estació a la qual s'envia directament la trama.
- **Adreça de l'estació transmissora**: indica l'estació que ha enviat aquesta trama.
- **Adreça de l'estació origen/destinació**. Si és una trama enviada des d'una estació a l'AP, aquest camp indica la destinació final, que pot ser el mateix AP (i llavors aquesta tercera adreça coincideix amb la primera) o bé una altra estació a la qual l'AP retransmetrà la trama. Per contra, si és una trama enviada per l'AP a una estació, aquest camp indica l'origen de la trama, que pot ser el mateix AP (i la tercera adreça coincidirà amb la segona) o bé una altra estació que ha enviat la trama per mitjà de l'AP.
- **Control de seqüència**: inclou un número de seqüència de la trama i un número de fragment.

3.2. Mètodes d'autenticació de les estacions Wi-Fi

Les primeres versions de l'estàndard Wi-Fi anteriors a l'IEEE 802.11i, seguint el criteri de simplicitat i de facilitar al màxim la interoperabilitat, preveien dos tipus d'autenticació de les estacions Wi-Fi davant l'AP.

- Autenticació de **sistema obert**. Aquesta autenticació, si l'AP està configurat per a permetre-la, és molt simple: cada estació que sol·licita l'autenticació rep automàticament la confirmació. Per això també és conegut com a *algorisme d'autenticació nul*.

  L'avantatge d'aquesta autenticació és que no cal que les estacions facin res especial per a completar-la. Així s'assoleix l'objectiu de facilitar la connexió de les estacions que s'incorporin a la xarxa.

- Autenticació de **clau compartida**. Aquest mètode d'autenticació s'utilitza juntament amb el sistema de xifratge WEP. En aquest cas l'AP fa ús d'una clau WEP que té preconfigurada i que només haurien de conèixer les estacions que s'hi hagin d'autenticar. Quan una estació sol·licita l'autenticació, l'AP li envia un missatge amb 128 bytes aleatoris, i l'estació ha de respondre amb una trama que contingui aquest mateix missatge, encriptada amb la clau WEP.

  Es tracta, doncs, d'un protocol de repte-resposta amb clau simètrica. El problema que té és que està basat en el xifratge WEP i, com veurem més endavant, descobrir una clau WEP no és excessivament complicat per a un atacant que pugui capturar una quantitat suficient de trames xifrades.

  Però, a més, aquest protocol d'autenticació té un altre problema, i és que la resposta encriptada no inclou cap identificador de l'estació que es vol autenticar. Com veurem també més endavant, això permet que un atacant que capturi un sol intercanvi de missatges utilitzi les dades capturades per a autenticar-se amb èxit davant l'AP.

A partir de la publicació de l'estàndard IEEE 802.11i, l'ús de l'autenticació de clau compartida està desaconsellat, i només es preveu en situacions en què es vulgui mantenir la compatibilitat amb sistemes anteriors. Per a fer una autenticació més segura IEEE 802.11i introdueix el concepte de *robust security network association* (RSNA), basat en l'*extensible authentication protocol* (EAP) d'acord amb l'estàndard IEEE 802.1X.

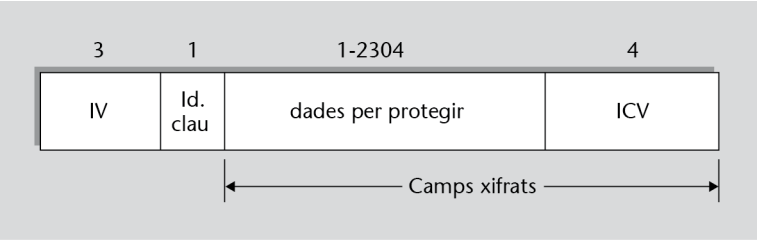
3.3. Protecció de trames amb WEP

La primera versió de l'estàndard IEEE 802.11 definia un mecanisme de seguretat per a protegir les trames enviades via ràdio: es tracta del protocol *wired equivalent privacy* (WEP). Tal com indica el nom, aquest protocol té per objectiu principal aconseguir una privacitat de les dades transmeses, davant dels simples curiosos que hi hagi pels voltants, que sigui semblant a la de les xarxes amb fil. De la mateixa manera que en un lloc amb una xarxa amb fil, els tafaners no poden veure res de la comunicació si no és que disposen d'algun mitjà per a interceptar el cable, en una xarxa sense fil amb WEP tampoc poden veure les dades que es transmeten. El mètode per a assolir aquesta **privacitat** és xifrar les dades. A més del servei de privacitat, el protocol WEP proporciona també el servei d'**integritat** per mitjà d'un codi d'integritat de les dades.

L'AP pot tenir configurades fins a quatre claus secretes WEP. Això permet per exemple utilitzar claus diferents amb grups d'estacions diferents, o anar canviant periòdicament la clau, però la gran majoria de vegades només es fa servir una sola clau WEP.

Cada trama WEP se xifra amb una clau de xifratge independent. Així es dificulta, entre d'altres, que un atacant pugui detectar dades repetides. La clau de xifratge utilitzada en una trama concreta s'obté a partir de la clau WEP en ús més un vector d'inicialització diferent per a cada trama. El valor d'aquest vector d'inicialització s'ha d'incloure en la trama mateix perquè el receptor sàpiga com desxifrar-la.

Dades d'una trama WEP



Una trama de dades xifrada amb WEP té el mateix format que una trama de dades normal, però la part corresponent a les dades s'estructura en quatre camps.

- El primer camp és el vector d'inicialització (IV) utilitzat per a xifrar la trama.
- El segon camp és l'identificador de clau, que serveix per a indicar quina clau WEP, de les quatre que pot tenir configurades l'AP, s'ha utilitzat en el xifratge.
- Al tercer camp hi ha les dades protegides.
- El quart camp és l'*integrity check value* (ICV), que és un CRC de 32 bits calculat sobre el tercer camp (abans de xifrar).

El tercer i quart camp, és a dir les dades protegides i l'ICV, es transmeten xifrats.

La inclusió de l'ICV permet comprovar que la trama xifrada no ha estat manipulada. Si un atacant que no coneix la clau vol modificar les dades d'una trama WEP enviada, o injectar una trama WEP amb les dades inventades, molt probablement quan el receptor la desxifri veurà que l'ICV no concorda. De totes maneres, un CRC no té les propietats de seguretat que pot tenir un codi d'integritat criptogràfic (per exemple basat en funcions resum o *hash*), i d'altra banda la resta de camps de la trama no estan protegits, cosa que permet que sí que puguin ser manipulats. Un altre cop, el criteri de la simplicitat va prevaldre sobre la seguretat en el disseny del protocol (un CRC és molt més fàcil de calcular que un resum).

3.3.1. El xifratge WEP

L'algorisme criptogràfic utilitzat per a xifrar les trames WEP és una xifra de flux, concretament l'**RC4** (*Ron's code 4*), dissenyat per Ronald Rivest. Va ser escollit per la seva simplicitat i pel nivell de seguretat que proporciona en relació amb la poca complexitat dels càlculs que requereix.

Aquest criteri era especialment important tenint en compte que la majoria de dispositius Wi-Fi poden ser dispositius mòbils, en els quals un baix consum d'energia té un paper rellevant. Si s'hagués escollit un algorisme més sofisticat, que comportés més potència de càlcul per a xifrar i desxifrar les mateixes dades, i per tant consumís més energia, l'autonomia de les bateries dels dispositius mòbils es podria veure reduïda sensiblement.

La longitud de les claus RC4 en general no és fixa: hi pot haver claus RC4 de fins a 2.048 bits (encara que una clau de xifratge tan llarga no té gaire sentit per a un xifratge simètric).

El protocol WEP preveu principalment l'ús de dues longituds de clau de xifratge RC4: claus de **64 bits** o claus de **128 bits**.

En les claus de xifratge que s'utilitzen en un BSS per a xifrar cada trama WEP hi ha una part variable i una part fixa:

- La part variable són els primers 24 bits de la clau, i es coneixen com a **vector d'inicialització** (IV). L'IV és diferent per a cada trama que es transmet.

- La part fixa és la resta de la clau: 40 bits si la clau és de 64 en total, o 104 bits si la clau és de 128 en total. Aquesta part fixa es coneix també com a **clau arrel**.

La clau arrel WEP de 40 o 104 bits és, doncs, la clau secreta (o les claus secretes, si se'n fan servir dues, tres o quatre, tot i que la situació més habitual és usar-ne només una) que té configurada l'AP, i que s'ha de configurar en les estacions Wi-Fi que s'hi vulguin associar.

Vector d'inicialització

El nom que se sol donar a la part variable de la clau és el de **vector d'inicialització**, tot i que aquest concepte està relacionat amb les xifres de bloc més que no pas amb les de flux. De fet el concepte de **bits de sal** s'acostaria més a la funció d'aquesta part variable de la clau.

Valor de la clau arrel

El valor de la clau arrel pot ser qualsevol combinació de bits, però per a facilitar la configuració manual de les estacions és habitual que una clau arrel de 104 bits tingui la forma de cadena de 13 caràcters ASCII.

L'algorisme que segueix una estació qualsevol, inclòs l'AP, per a generar una trama xifrada WEP és el següent:

1) Generar una cadena de 24 bits per a utilitzar com a IV, procurant que sigui diferent dels últims IV generats.

2) Concatenar els 24 bits de l'IV amb la clau WEP arrel per a formar la clau RC4 de xifratge de la trama.

3) Calcular el CRC de les dades que cal protegir. Amb aquesta operació s'obté l'ICV.

4) Concatenar les dades amb l'ICV, i xifrar aquesta seqüència amb l'algorisme RC4 utilitzant la clau de xifratge del punt 2. Com que es tracta d'una xifra de flux, aquesta operació consisteix a obtenir tants bits de text de xifratge (*keystream*) com tingui la seqüència a protegir, i sumar-los un a un amb els de la seqüència.

5) Omplir la part de dades de la trama WEP amb els camps que la componen: el VI, l'identificador de la clau WEP, i el resultat del xifratge obtingut al punt 4.

L'estació que rep la trama xifrada WEP ha de fer els passos inversos per a desxifrar-la:

1) Llegir el camp IV de la trama WEP.

2) Concatenar els 24 bits de l'IV amb la clau arrel indicada pel camp identificador de clau WEP. Així s'obté la clau RC4 de desxifratge de la trama.

3) Desxifrar la part xifrada de la trama amb l'algorisme RC4 utilitzant la clau de xifratge del punt 2. Com abans, el xifratge consisteix a sumar bit a bit les dades xifrades amb el text de xifratge (*keystream*) generat a partir de la clau.

4) Calcular el CRC de les dades desxifrades i comprovar que coincideix amb el camp ICV també acabat de desxifrar.

L'objectiu últim d'incloure un IV és tenir una clau de xifratge diferent per a cada trama, i per a això cada IV hauria de ser diferent, o almenys que no es repeteixin fins al cap d'un nombre molt gran de trames generades. Les implementacions Wi-Fi normalment segueixen una d'aquestes dues tècniques per a aconseguir-ho:

- Generar cada IV amb un generador de nombres pseudoaleatoris.
- Generar el primer IV com un nombre aleatori de 24 bits, i obtenir els següents sumant 1 cada vegada a l'anterior. Aquest mode de generar els VI s'anomena **mode comptador**.

Suma bit a bit

Recordeu que la suma bit a bit, que coincideix amb l'operació lògica XOR (*OR* exclusiva), és autocomplementària i que, per tant, el xifratge (suma) i el desxifratge (resta) es fan igual.

IV diferents

Com a màxim es poden generar 2^{24} IV diferents, és a dir uns 16 milions.

L'opció de l'algorisme RC4 per al xifratge WEP obeïa, com hem comentat abans, a la simplicitat de la seva implementació. Tot i que era un algorisme que es considerava raonablement segur, la manera com es va dissenyar el seu ús en el protocol WEP, especialment amb la introducció dels vectors d'inicialització, fa que presenti algunes vulnerabilitats importants.

Per comprendre les implicacions que tenen aquestes vulnerabilitats, abans veurem les característiques generals de l'algorisme RC4.

3.3.2. L'algorisme RC4

La simplicitat de l'algorisme RC4 ve donada d'una banda per les operacions en què es basa, i de l'altra per la poca memòria que requereix per a guardar la informació d'estat del xifratge.

- L'única operació aritmètica que es necessita per a implementar l'algorisme és la suma mòdul 256 o, el que és el mateix, la suma de 8 bits ignorant l'arrossegament (*carry*) generat. Aquesta és una operació senzillíssima d'implementar en maquinari, i molt ràpida de calcular. L'algorisme també fa servir l'operació d'intercanvi (*swap*) d'elements d'un vector, però no és una operació aritmètica sinó simplement de moviment de dades en memòria.

- La informació d'estat amb què treballa l'algorisme és un vector de 256 elements de 8 bits, més dos comptadors o índexs també de 8 bits cadascun. En total, doncs, es necessiten 258 bytes de memòria per a guardar aquesta informació d'estat (a part de l'espai que ocupi la clau, que només es necessita en la fase inicial de l'algorisme).

Per a la descripció de l'algorisme farem servir la notació següent:

- $K[0]$, $K[1]$, $K[2]$, etc. són el primer, segon, tercer, etc. bytes de la clau de xifratge. Si la clau és, per exemple, de 128 bits els elements que la formen són $K[0]$ fins a $K[15]$.

- $S[0]$, $S[1]$, $S[2]$, etc. són el primer, segon, tercer, etc. bytes del text de xifratge (*keystream*) generat.

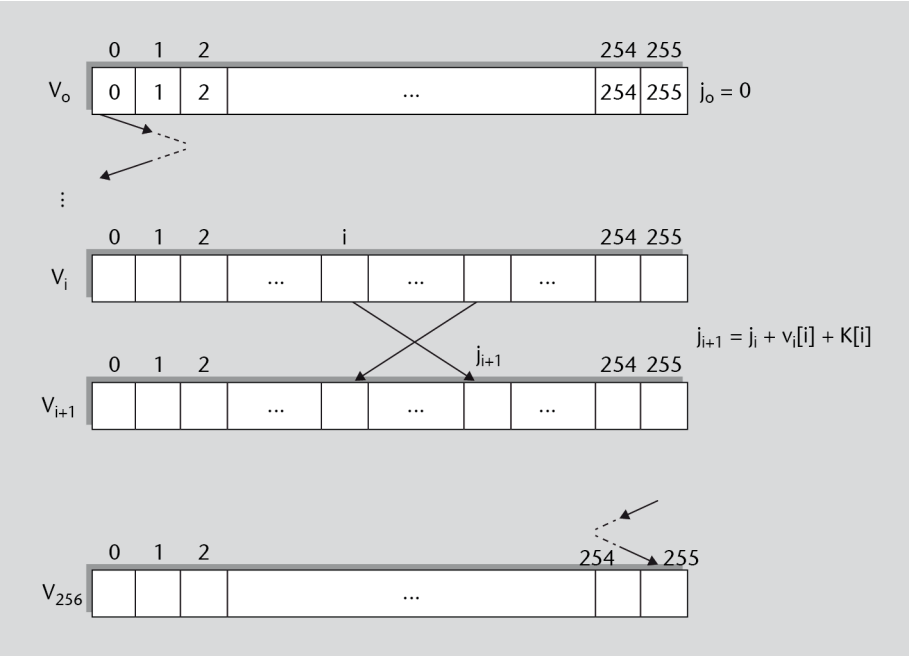
- $V[0], \ldots, V[255]$ són els elements del vector d'estat de l'algorisme.

- i, j són els dos comptadors interns amb què treballa l'algorisme.

- El símbol de suma + representa la suma mòdul 256, és a dir la suma de 8 bits.

El funcionament de l'algorisme es divideix en dues fases. La primera és el *key schedule algorithm* (KSA) o programació de la clau, i la segona és el *pseudo-random generation algorithm* (PRGA) o generació del text de xifratge pròpiament dit.

Esquema de l'algorisme KSA de xifratge RC4



L'objectiu de l'algorisme KSA és obtenir, a partir del valor de la clau K, un vector d'estat V que serveixi perquè l'algorisme PRGA comenci a generar el text de xifratge S. Els passos que segueix el KSA són aquests:

1) El vector V parteix d'un estat inicial (V_0) en què l'element $V_0[0]$ té el valor 0, l'element $V_0[1]$ té el valor 1, ... i l'element $V_0[255]$ té el valor 255. Els comptadors i i j valen 0.

2) Si la clau té una longitud de L bytes, es concatena amb si mateixa tantes vegades com calgui fins que ocupi 256 bytes. (De fet no cal ocupar físicament aquests bytes de memòria, n'hi ha prou de substituir els accessos a $K[i]$ per $K[i \bmod L]$.)

3) Es repeteix 256 vegades la seqüència següent:

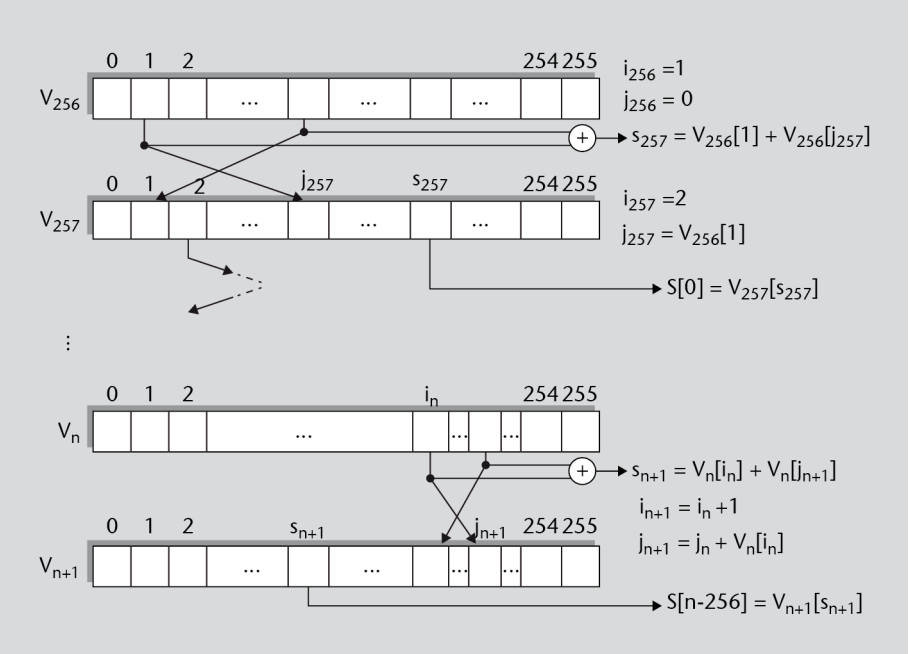
a) A l'índex j s'hi suma (mòdul 256) $V[i] + K[i]$.

b) S'intercanvien els elements $V[i]$ i $V[j]$ del vector d'estat.

c) A l'índex i s'hi suma 1.

Al final d'aquests passos tenim un vector d'estat V_{256} que conté una permutació aparentment aleatòria dels elements $0, \ldots, 255$.

Esquema de l'algorisme PRGA de xifratge RC4



Un cop obtingut el vector V_{256}, que és el vector d'estat inicial per a començar la generació del text de xifratge, s'aplica l'algorisme PRGA. En aquest algorisme primer s'inicialitza l'índex i a 1 i l'índex j a 0. A continuació, per a cada byte del text de xifratge S que es vulgui obtenir, es fa una iteració que consta dels passos següents:

1) A l'índex j s'hi suma (mòdul 256) $V[i]$.

2) S'intercanvien els elements $V[i]$ i $V[j]$ del vector d'estat.

3) Es calcula l'índex s com la suma dels dos elements intercanviats ($s = V[i] + V[j]$).

4) A l'índex i s'hi suma (mòdul 256) 1.

5) El resultat obtingut en aquesta iteració, és a dir el següent byte del text de xifratge S, és igual a l'element $V[s]$.

3.4. Vulnerabilitats del protocol WEP

Com ja hem comentat abans, el protocol WEP té una sèrie de vulnerabilitats, algunes de les quals són independents de l'elecció de l'RC4 com a algorisme de xifratge, i d'altres que estan directament relacionades amb la manera com s'usa aquest algorisme.

3.4.1. Vulnerabilitats no relacionades amb l'algorisme RC4

Aquest conjunt de vulnerabilitats és conseqüència de certes decisions de disseny del protocol WEP independents de l'ús de l'algorisme RC4.

Injecció de trames

Un atacant que capturi una trama WEP corresponent a una associació determinada pot retransmetre-la tantes vegades com vulgui i, si l'associació continua existint, el receptor donarà la trama per vàlida. I si l'associació ja no existeix, l'atacant pot canviar les adreces de l'estació transmissora i/o receptora per les d'altres estacions que sí que estiguin associades, i el nou receptor també donarà la trama per vàlida.

Això és així, d'una banda, perquè ni el nivell d'enllaç IEEE 802.11 ni el protocol WEP preveuen res per a detectar trames duplicades. Les trames WEP injectades per l'atacant tindran l'IV repetit, però això no és problema del receptor. Encara que l'ús dels IV tingui per objectiu que les trames xifrades siguin sempre diferents, res no impedeix a una estació enviar trames idèntiques xifrades amb el mateix IV. I les estacions receptores no solen comprovar si els arriben trames amb IV repetit, perquè això implicaria haver de recordar els IV de les últimes trames rebudes i comparar cada trama nova que arribi, cosa que normalment no fan.

I d'altra banda, com que els camps de la capçalera MAC no estan protegits pel codi d'integritat ICV, no hi ha cap problema a canviar les adreces d'aquesta capçalera.

Falsificació de l'autenticació

La falsificació de l'autenticació només té sentit en el mètode de clau compartida, perquè en el mètode d'autenticació de sistema obert no hi ha res per falsificar.

Si un atacant captura les trames intercanviades durant una autenticació de clau compartida entre una estació i un AP, pot autenticar-se amb èxit davant el mateix AP sense necessitat de conèixer la clau WEP corresponent.

En el procés d'autenticació de clau compartida s'intercanvien quatre trames: petició d'autenticació, repte, resposta, i resultat. La tercera trama està xifrada amb WEP, però l'atacant sap quin ha de ser el contingut desxifrat perquè en la segona trama hi ha el repte en clar. Per tant, si del contingut xifrat de la tercera trama en resta bit a bit (és a dir, hi suma) el contingut desxifrat, obté el text de xifratge (*keystream*) que es genera amb la clau WEP i l'IV de la tercera trama.

Llavors l'atacant només ha d'enviar una petició d'autenticació a l'AP, rebre el repte, i construir una trama de resposta amb l'IV capturat prèviament i la suma bit a bit del repte més el *keystream* calculat. L'AP veurà que és una resposta correctament

xifrada perquè en desxifrar-la obtindrà el repte de la segona trama, i per tant donarà per autenticada l'estació de l'atacant.

La captura de les trames d'autenticació, en general, permet obtenir una certa quantitat de *keystream* corresponent a un determinat IV. A la trama de resposta hi ha 140 bytes xifrats dels quals es coneix el valor desxifrat (els 128 del repte més els d'altres camps), i per tant s'obtenen 140 bytes de *keystream*. Això pot ser útil per a generar altres trames xifrades a part de la de resposta a l'autenticació. I a més hi ha altres atacs que permeten calcular més bytes de *keystream*, per si cal falsificar trames més llargues.

Desxifratge de trames mitjançant la comprovació d'integritat o atac *chopchop*

Un atacant que hagi capturat una trama WEP pot desxifrar els n últims bytes de dades xifrades, sense necessitat de saber la clau de xifratge, enviant a l'AP una mitjana de $128 \times n$ trames.

El codi d'integritat ICV que incorporen les trames WEP es calcula amb l'algorisme CRC-32, que és un mètode molt bo per a detectar errors de transmissió, però no és un algorisme criptogràfic. Com que el CRC-32 està basat en operacions aritmètiques lineals com són les divisions de polinomis mòdul 2, és possible fer càlculs inversos.

A cada seqüència de bits li correspon un polinomi binari P, els coeficients del qual són els bits de la seqüència. El seu CRC és una altra seqüència, corresponent al polinomi R, tal que la concatenació $P \parallel R$, que anomenarem X, és múltipla del polinomi generador G, en aquest cas el polinomi CRC-32. Per tant, per a comprovar que el CRC és correcte només cal veure si es compleix

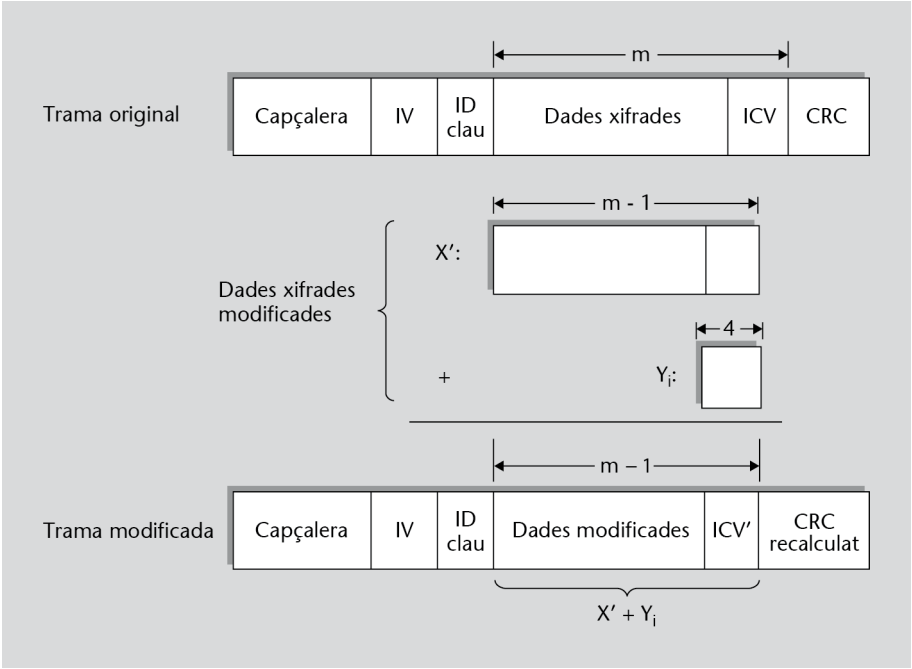
$$X \bmod G = P \parallel R \bmod G = 0$$

Si considerem X com a seqüència de m bytes $X[0] \parallel \cdots \parallel X[m-1]$, i X' és la mateixa seqüència sense l'últim byte $(X[0] \parallel \cdots \parallel X[m-2])$, és fàcil calcular quina és la seqüència Y que cal sumar a X' perquè continuï essent divisible per G, és a dir que es compleixi

$$(X' + Y) \bmod G = 0$$

i que, per tant, el CRC continuï essent correcte. Com que resulta que el valor Y només depèn de $X[m-1]$, és a dir de l'últim byte de la seqüència X, podem calcular els 256 valors $Y_0, \ldots, Y_{255}$ corresponents a cadascun dels possibles valors de l'últim byte de X.

Modificació de trames WEP per a l'atac *chopchop*



Amb aquesta tècnica, l'atac per a desxifrar una trama WEP consisteix a construir noves trames modificades, fins a 256 en total, seguint aquests passos:

- Suprimir l'últim byte xifrat de la trama original.
- Sumar cadascun dels valors $Y_0 \ldots, Y_{255}$, respectivament, a les dades xifrades que queden.
- Recalcular el CRC no xifrat per a cada nova trama.

Llavors l'atacant va enviant aquestes trames modificades a l'AP per tal d'esbrinar si l'ICV és correcte o no. Quan la resposta de l'AP indiqui que la trama és correcta, l'atacant sabrà quin és el valor desxifrat de l'últim byte xifrat de la trama: el que correspongui al valor Y_i utilitzat. I a partir del valor xifrat i el valor desxifrat, fent la resta (suma) obtindrà l'últim byte del *keystream* utilitzat per a xifrar la trama.

Suma de bits a les dades xifrades

Conceptualment, l'operació que caldria fer seria desxifrar les dades, sumar-hi la seqüència Y, i tornar-les a encriptar. Però com que el xifratge en realitat també és una suma (dels bits en clar amb el *keystream*), el resultat és el mateix si la suma es fa directament sobre les dades xifrades.

Com que hi ha 256 valors possibles diferents per provar, el nombre de trames que caldrà enviar abans de trobar la bona, de mitjana, serà de 128.

A partir de la trama bona es pot tornar a repetir l'atac per a trobar el penúltim byte del *keystream*, i així successivament. Amb aquest mètode es podrien obtenir tots els bytes del *keystream* (i per tant desxifrar tots els bytes xifrats de la trama original) excepte els 4 primers, perquè els valors Y són seqüències de 32 bits. Però com que entre els bytes desxifrats hi haurà els del codi ICV, és immediat deduir els bytes que falten perquè aquest codi sigui correcte.

Hi ha diferents maneres de fer que l'AP ens digui si l'ICV d'una trama WEP és correcte o no, entre les quals podem esmentar aquestes dues:

- Si disposem de dues estacions, des de cadascuna podem fer una autenticació (falsa, si no coneixem la clau WEP) i una associació amb l'AP. Llavors podem enviar les trames que es volen provar des de la primera estació amb destinació la segona per mitjà de l'AP. Si l'ICV d'una trama és correcte l'AP la retransmetrà a la segona estació, i si no, la descartarà.

- Una altra manera més senzilla és enviar les trames que es volen provar des d'una estació no associada. Si l'ICV d'una trama és correcte l'AP respondrà amb una trama de gestió que indicarà que l'estació no està associada, i si no, la descartarà.

Primers bytes xifrats de la trama

En la pràctica, és possible que l'AP descarti les trames WEP que no tinguin una longitud mínima, de manera que hi ha un punt a partir del qual no es poden obtenir més bytes de *keystream* amb l'atac *chopchop*.

Aquest atac que va escapçant byte a byte la trama capturada a mesura que la va desxifrant es coneix com a *atac chopchop de KoreK*, o simplement *atac chopchop*. Es basa en la tècnica d'un altre atac publicat anteriorment, anomenat *atac inductiu d'Arbaugh*. Aquest últim és de fet l'atac invers del *chopchop*: en comptes de retrocedir byte a byte, va "avançant" a còpia d'assaig i error i així va esbrinant els bytes següents del *keystream*. D'aquesta manera, amb una mitjana de 128 intents per byte es pot aconseguir una longitud arbitrària de *keystream*, útil per a construir trames WEP falsificades més o menys llargues sense saber la clau.

KoreK

KoreK és el pseudònim de l'autor que ha publicat diversos atacs contra els protocols Wi-Fi.

Atac de fragmentació

Un atacant que hagi descobert n bytes d'un *keystream* pot obtenir fins a $16 \times n - 60$ bytes d'un altre enviant fins a 16 trames fragmentades a l'AP.

IEEE 802.11 permet enviar una trama en fragments, fins a un màxim de 16. Si una estació envia a l'AP una trama fragmentada que hagi de ser retransmesa, l'AP normalment recompondrà la trama abans de retransmetre-la. I, si els fragments estan xifrats, la trama recombinada també estarà xifrada, possiblement amb un IV nou.

Així, un atacant que disposi de n bytes de *keystream* pot construir 16 fragments de trama WEP, cadascun amb $n-4$ bytes de dades més els 4 bytes de l'ICV, tots xifrats amb el mateix IV i *keystream*. Si envia aquests fragments a l'AP perquè els retransmeti, la trama resultant tindrà $16 \times (n-4)$ bytes de dades i 4 bytes d'ICV encriptats (sempre que no sobrepassi la longitud màxima de les dades d'una trama), en total $16 \times n - 60$ bytes dels quals el valor desxifrat serà conegut. Per tant, l'atacant podrà obtenir aquesta quantitat de bytes de *keystream*.

Si el *keystream* recuperat encara no arriba a la longitud necessària per a xifrar una trama llarga, es pot tornar a repetir aquest atac fins a obtenir la longitud màxima possible.

3.4.2. Vulnerabilitats relacionades amb l'algorisme RC4

Aquest conjunt de vulnerabilitats és conseqüència del mètode amb què s'utilitza l'algorisme RC4 en el protocol WEP.

L'atac FMS

Els criptoanalistes Scott Fluhrer, Itsik Mantin i Adi Shamir, en un article publicat l'any 2001, van detallar les bases teòriques d'un atac que permetia recuperar una clau arrel WEP a partir de les trames xifrades si es disposava d'un nombre suficient d'aquestes trames. Un altre equip d'investigadors va publicar l'any 2004 els resultats de la primera implementació d'aquest atac contra una xarxa Wi-Fi real.

Amb l'atac FMS, un atacant que conegui el primer byte de *keystream* ($S[0]$) d'aproximadament entre 4 i 9 milions de trames WEP, encriptades amb la mateixa clau arrel, pot recuperar el valor de la clau amb una probabilitat d'èxit del 50%.

Atac FMS

El nom amb què es coneix aquest atac prové de les inicials dels cognoms dels seus descobridors.

Aquest atac es basa en l'observació del funcionament de l'algorisme RC4. Tal com s'utilitza en el protocol WEP els 3 primers bytes de la clau RC4 són sempre coneguts, ja que formen el vector d'inicialització (IV) que s'envia en clar prefixat a les dades xifrades. Per a cada trama capturada amb l'IV corresponent, doncs, l'atacant disposa dels elements $K[0]$, $K[1]$ i $K[2]$ de la clau RC4. Amb aquesta informació pot reconstruir les 3 primeres iteracions de l'algorisme KSA i obtenir el vector d'estat V_3 i el valor de l'índex j_3.

Primer byte del *keystream*

El cas més habitual és que les dades xifrades d'una trama no fragmentada, o del primer fragment d'una trama fragmentada, comencin amb el primer byte de la capçalera LLC igual a AA (hexadecimal). Per tant, conegut el primer byte xifrat i el primer byte desxifrat, també es coneix el primer byte de *keystream*.

L'algorisme KSA es completa amb 253 iteracions més fins a arribar a obtenir el vector V_{256}, a partir del qual es calcula amb l'algorisme PRGA el primer byte de *keystream* $S[0]$:

$$S[0] = V_{257}[s_{257}] = V_{257}[V_{256}[1] + V_{256}[j_{257}]] = V_{257}[V_{256}[1] + V_{256}[V_{256}[1]]]$$

El contingut del vector V_{256} serà en general desconegut, però es pot seleccionar un subconjunt de les trames que tinguin certes propietats que afavoreixen l'atac. Per a fer l'atac, s'analitza cada trama i es comprova si amb el seu IV es compleixen les anomenades **condicions de resolució**:

i. $V_3[1] < 3$

ii. $V_3[1] + V_3[V_3[1]] = 3$

Si la trama no compleix aquestes condicions, es descarta i es passa a la següent.

Condicions de resolució

Estadísticament, una de cada 21675 trames complirà les condicions de resolució per a $n = 3$. Si creix n, també creix el nombre de trames que compleixen les condicions.

A continuació, l'atac consisteix a veure què passaria si es donés la feliç coincidència que tres elements determinats del vector d'estat no s'intercanviessin amb cap altre en les iteracions següents del KSA. Més concretament, el cas que es considera és que es donin aquestes tres condicions alhora:

i. Que l'element $V_3[1]$ no canviï de lloc entre les iteracions 4 i 256.

ii. Que l'element $V_3[V_3[1]]$ no canviï de lloc entre les iteracions 4 i 256.

iii. Que l'element $V_3[j_4]$ (desconegut, perquè si no tenim el quart byte de la clau, $K[3]$, no sabem quant val j_4), que a la iteració 4 passarà a ser $V_4[3]$, no canviï de lloc entre les iteracions 5 i 256.

Com que a cada iteració del KSA s'intercanvien $V[i]$ i $V[j]$, perquè es no malbarati aquesta sèrie de coincidències l'índex j no hauria de passar per cap dels valors "prohibits" 1, $V_3[1]$ i 3 (aquest últim, a partir de la iteració 5). Amb l'índex i no hi ha problema perquè a partir de la iteració 4 valdrà com a mínim 4 i a més sabem, per la primera condició de resolució, que $V_3[1] < 3$. Pel que fa a j, es pot calcular que la probabilitat que aquest índex no prengui cap dels 3 valors prohibits en cap de les 253 iteracions següents és $((256-3)/256)^{253}$, és a dir aproximadament un 5%.

Llavors sabem que, amb una probabilitat aproximada del 5%, aquests tres elements del vector d'estat no s'hauran mogut, i per tant $V_{256}[1] = V_3[1]$ i $V_{256}[V_{256}[1]] = V_3[V_3[1]]$. A la primera iteració del PRGA, s'intercanvien precisament els elements $V_{256}[1]$ i $V_{256}[V_{256}[1]]$, i es calcula l'índex s_{257} com la suma d'aquests dos valors. Per la segona condició de resolució aquesta suma és igual a 3, i això vol dir que el primer byte de *keystream* generat serà igual a $V_{257}[3]$. Si es compleix la nostra hipòtesi d'immobilitat, aquest element no s'haurà mogut entre les iteracions 5 i 256, i a més les condicions de resolució garanteixen que tampoc s'haurà mogut a la iteració 257.

En definitiva, si no s'han mogut els elements esmentats tindrem que

$$S[0] = V_{257}[3] = V_4[3] = V_3[j_4] = V_3[j_3 + V_3[3] + K[3]]$$

A partir d'aquí, com que j_3 i el vector V_3 són coneguts, i la hipòtesi inicial de l'atac és que $S[0]$ també és conegut, és immediat trobar el valor $K[3] = V_3^{-1}[S[0]] - j_3 - V_3[3]$ (fent les operacions en mòdul 256). Aquest serà el valor correcte del quart byte de la clau sempre que sigui veritat que els tres elements en qüestió no s'han mogut de lloc. Si no, el resultat obtingut amb aquesta fórmula serà un valor que podem considerar aleatori.

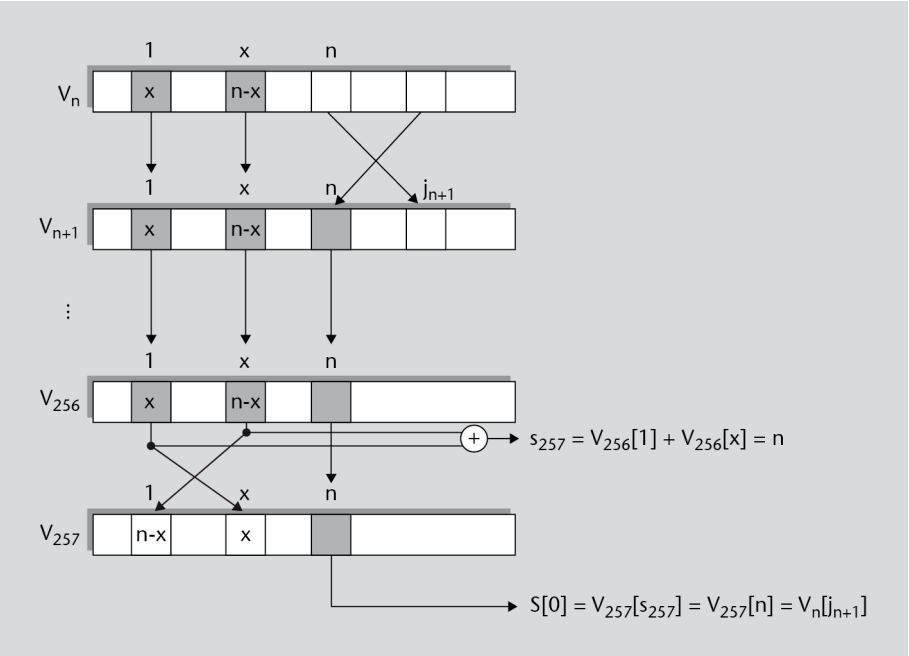
Inversió de la transformació $V[x]$

A partir de $y = V[x]$, sempre és possible trobar el valor únic $x = V^{-1}[y]$ perquè V és una permutació dels elements $0, \ldots, 255$: cada element apareix una i només una vegada en la permutació.

És a dir, si calculem el possible valor de $K[3]$ a partir d'un nombre de trames suficient perquè la distribució dels valors incorrectes sigui uniforme, trobarem que aproximadament 15 de cada 270 trames donaran un mateix valor, i les altres 255 donaran valors diferents entre si. Empíricament es pot comprovar que a partir d'unes 60 trames analitzades ja hi ha un valor que destaca sobre els altres perquè es repeteix almenys 3 o

4 vegades. Aquest valor candidat que té majoria de "vots" és el que en principi es pot considerar com a correcte.

Elements del vector d'estat que no s'haurien de modificar perquè sigui certa la hipòtesi de l'atac FMS



Un cop determinat el valor candidat a ser el quart byte de la clau, $K[3]$, es pot tornar a començar l'atac per tal d'esbrinar el byte següent, $K[4]$. En general, les condicions de resolució per a intentar obtenir el valor del byte $K[n]$ són les següents:

i. $V_n[1] < n$

ii. $V_n[1] + V_n[V_n[1]] = n$

i el valor es calcula amb la fórmula

$$K[n] = V_n^{-1}[S[0]] - j_n - V_n[n]$$

En l'obtenció de cada byte de la clau es poden trobar falsos positius. Això passarà si entre les trames que no compleixen la condició d'immobilitat n'hi ha diverses que coincideixen a donar un mateix valor (incorrecte) de $K[n]$, i superen en nombre les trames que donen el valor correcte, de manera que el valor fals té més vots que el bo. Quan un byte esbrinat $K[n]$ és incorrecte, tots els bytes següents $K[m]$ amb $m > n$ estaran mal calculats perquè l'algorisme KSA per a obtenir el vector V_m s'estarà aplicant amb valors erronis.

La comprovació per a saber si els bytes de la clau són correctes només es pot fer quan s'han esbrinat tots, des de $K[3]$ fins a $K[15]$ en el cas d'una clau WEP-104. Llavors es poden utilitzar uns quants VI per veure si la clau calculada dóna el byte de *keystream* $S[0]$ corresponent a cadascun dels VI. Si no és així, cal tornar enrere i refer els càlculs. Per exemple, es pot mirar quin dels bytes de la clau ha guanyat per una majoria més

Probabilitat de no intercanvi

A mesura que avança n també creix la probabilitat que els elements no siguin intercanviats perquè queden menys iteracions fins al final del KSA, però la variació no és gaire gran: des del 5,07% per a $n = 3$ fins al 5,84% per a $n = 15$.

Obtenció de l'últim byte de la clau

Si hi ha moltes trames per processar, en comptes d'obtenir l'últim byte de la clau $K[15]$ pel sistema de votació, pot ser més eficient obtenir només fins al penúltim byte i fer les comprovacions per força bruta amb cadascun dels 256 possibles valors de l'últim byte. En alguns casos fins i tot es pot aplicar la força bruta als dos últims bytes de la clau.

estreta, substituir-lo pel segon valor que hagi tingut més vots, i recalcular tots els bytes posteriors. I si la comprovació tampoc és satisfactòria, es poden repetir aquestes substitucions fins a trobar el valor correcte o fins que s'hagi ultrapassat un nombre màxim d'intents.

Experimentalment s'ha comprovat que a partir de 4 milions de trames, l'atac FMS permet obtenir el valor correcte de la clau WEP amb una probabilitat d'èxit del 50%. Aquest nombre de trames, però, puja fins a 9 milions si els IV estan generats en mode comptador en comptes d'haver-se generat aleatòriament. També s'ha comprovat que, a partir d'un cert punt, per molt que s'augmenti el nombre de trames processades difícilment s'ultrapassa el 75% de probabilitat d'èxit.

Èxit de l'atac FMS

Un criteri per a considerar que l'atac no ha tingut èxit és que el mètode d'assaig i error per a trobar la clau bona no doni cap resultat al cap de 2 o 3 minuts, amb la potència de càlcul mitjana dels ordinadors actuals.

El fet que el mode comptador requereixi més trames ve donat per la distribució dels IV que compleixen les condicions de resolució. En el mode aleatori estaran repartits uniformement entre les trames capturades, mentre que en el mode comptador estaran concentrades en sèries de trames consecutives o molt pròximes. Si l'atacant té la sort de topar aviat amb una d'aquestes sèries pot obtenir ràpidament els IV necessaris, però si no, necessitarà de mitjana moltes més trames.

El conjunt d'atacs KoreK

Es coneix amb el nom d'*atacs KoreK* una sèrie d'atacs al protocol WEP que exploten determinades correlacions entre la clau arrel i els primers bytes de text d'encriptació o *keystream*. L'any 2004 una persona que feia servir el pseudònim "KoreK" va publicar una eina que integrava tots aquests atacs, que en total eren 17. Alguns ja es coneixien prèviament, com és el cas de l'atac FMS, i els altres van ser descoberts per KoreK.

Amb els atacs KoreK, un atacant que conegui els dos primers bytes de *keystream* $(S[0], S[1])$ d'aproximadament entre 150.000 i 700.000 trames WEP, encriptades amb la mateixa clau arrel, pot recuperar el valor de la clau amb una probabilitat d'èxit del 50%.

Els atacs KoreK es poden dividir en tres grups:

Segon byte de *keystream*

Així com en la gran majoria de casos el primer byte de dades encriptades d'una trama WEP és el primer byte de la capçalera LLC, el segon byte encriptat serà el segon byte de la mateixa capçalera, que també és igual a AA (hexadecimal), i a partir d'aquest valor es pot saber el segon byte de *keystream* $S[1]$.

- Atacs que permeten esbrinar $K[n]$ a partir de $K[0], \ldots, K[n-1]$ i $S[0]$. L'atac FMS pertany a aquest grup.
- Atacs que permeten esbrinar $K[n]$ a partir de $K[0], \ldots, K[n-1]$, $S[0]$ i $S[1]$.
- Atacs "negatius" que, si V_n compleix certes condicions i $S[0]$ pren certs valors, permeten descartar determinats valors de $K[n]$.

Molts dels atacs es basen, igual que l'FMS, en la probabilitat que determinats elements del vector d'estat no siguin intercanviats a partir de la iteració n de l'algorisme KSA. Cadascun dels atacs individuals té condicions de resolució pròpies. L'eina que es va publicar anava comprovant, trama per trama, si es complien les condicions d'algun atac, i si era així el duia a terme i obtenia un vot a favor d'un candidat a $K[n]$, o un vot en contra, en cas que es tractés d'un atac negatiu. Els vots es ponderaven segons la probabilitat d'èxit de l'atac realitzat.

El fet d'implementar diversos atacs diferents en paral·lel facilita la tasca de descobrir la clau amb un nombre menor de trames analitzades. Experimentalment s'ha trobat que a partir de 150.000 trames els atacs KoreK permeten obtenir el valor correcte de la clau WEP amb una probabilitat d'èxit del 50% si els VI estan generats aleatòriament. En canvi, si els VI es generen en mode comptador el nombre de trames necessàries creix fins a 700.000 per a assolir el mateix 50% d'èxit. A partir de 270.000 trames en mode aleatori, i 1.700.000 en mode comptador, la taxa d'èxit és del 90%.

L'atac PTW

L'any 2007 es va publicar un nou atac, conegut com a PTW, que és una variant millorada d'un altre que va ser descobert el 2005, anomenat **atac Klein**.

Amb l'atac PTW, un atacant que conegui els bytes de *keystream* del tercer al quinzè ($S[2],\ldots,S[14]$) d'aproximadament 35.000 trames WEP, encriptades amb la mateixa clau arrel, pot recuperar el valor de la clau amb una probabilitat d'èxit del 50%.

Atac PTW

El nom amb què es coneix aquest atac prové de les inicials dels cognoms dels autors que el van publicar: Andrei Pyshkin, Erik Tews i Ralf-Philipp Weinmann.

L'atac Klein es basa en una anomalia de les propietats estadístiques de la programació de claus RC4. L'algorisme KSA genera un vector d'estat aparentment aleatori, i com que cada element pot tenir un de 256 valors possibles, és d'esperar que la probabilitat que un element $V[n]$ tingui un determinat valor x sigui $1/256$, és a dir aproximadament un 0,4%. Una combinació d'elements del vector, com ara $V[V[i]+V[j]]+V[j]$, en principi també hauria de ser igual a qualsevol valor $0 \leq x \leq 255$ de manera equiprobable. Però l'anomenada **correlació de Jenkins** demostra que hi ha un valor d'aquesta expressió que és més probable que els altres. Concretament:

$$\text{Prob}(V[V[i]+V[j]]+V[j]=i) = 2/256$$

Primers 15 bytes de *keystream*

Hi ha algunes trames, com és el cas de les que contenen paquets ARP, en què els 15 primers bytes de dades corresponen a camps de capçaleres amb valors constants. Per tant, si aquestes trames estan encriptades es poden obtenir 15 bytes de *keystream*.

Així, la combinació d'elements anterior pot prendre el valor i amb una probabilitat aproximada del 0,8%, en comptes del 0,4% que s'esperaria.

A més, la correlació de Jenkins també demostra que la resta de valors $x \neq i$ són equiprobables, de manera que la probabilitat de cadascun és $(1-2/256)/255 = 127/32640$.

Igual que en l'atac FMS, en l'atac Klein inicialment cal fer les 3 primeres iteracions de l'algorisme KSA per a obtenir V_3, i a la iteració 4 sabem que l'element $V_3[j_4]$ passarà a ser $V_4[3]$. En alguna de les iteracions següents l'índex j pot prendre el valor 3, i llavors aquest element canviarà de lloc. Però l'índex i no tornarà a valer 3 fins a la iteració 258, ja dins de l'algorisme PRGA, és a dir fins al cap de 254 iteracions.

Si considerem que les variacions de j tenen un comportament aleatori, la probabilitat que l'índex j no prengui el valor 3 en cap d'aquestes 254 iteracions és $(255/256)^{254}$, és a dir un 37%. Com que l'índex i tampoc valdrà 3 en cap de les 254 iteracions, aquesta és la probabilitat que l'element $V_4[3]$ no s'hagi mogut del seu lloc fins a la iteració 258. En l'altre 63% dels casos j haurà valgut 3 en algun moment i l'element $V_{258}[3]$ ja no serà el mateix que hi havia a $V_4[3]$.

A la iteració següent, la 259, s'obtindrà el tercer byte de *keystream* $S[2]$:

$$S[2] = V_{259}[s_{259}] = V_{259}[V_{258}[i_{258}] + V_{258}[j_{259}]] = V_{259}[V_{258}[3] + V_{258}[j_{259}]]$$

Com que en aquesta iteració s'hauran intercanviat els elements de les posicions 3 i j_{259}, la suma $V_{258}[3] + V_{258}[j_{259}]$ serà la mateixa que $V_{259}[j_{259}] + V_{259}[3]$, i per tant:

$$S[2] = V_{259}[V_{259}[3] + V_{259}[j_{259}]]$$

Sumant $V_{259}[j_{259}]$:

$$S[2] + V_{259}[j_{259}] = V_{259}[V_{259}[3] + V_{259}[j_{259}]] + V_{259}[j_{259}]$$

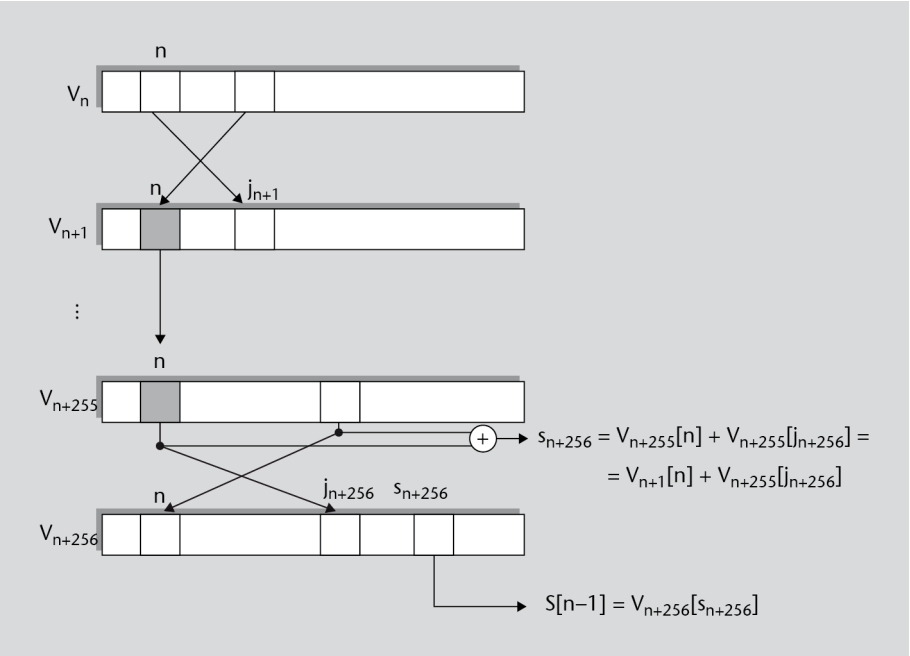
I per la correlació de Jenkins sabem que aquesta expressió té més probabilitat de valer 3 que qualsevol altre valor. Així tenim que hi ha una probabilitat de $2/256$ que es compleixi $S[2] + V_{259}[j_{259}] = 3$ o, ja que $V_{259}[j_{259}]$ és l'element que hi havia a $V_{258}[3]$ abans de l'últim intercanvi, que es compleixi $S[2] + V_{258}[3] = 3$.

Per a obtenir el valor del byte $K[3]$, l'atac Klein es basa en la probabilitat que la hipòtesi $S[2] + V_4[3] = 3$ sigui certa. Llavors, com que

$$V_4[3] = V_3[j_4] = V_3[j_3 + V_3[3] + K[3]]$$

aïllant $K[3]$ tenim que el valor candidat és $K[3] = V_3^{-1}[3 - S[2]] - j_3 - V_3[3]$ (amb totes les operacions en mòdul 256).

Element del vector d'estat que no s'hauria de modificar perquè sigui certa la hipòtesi de l'atac Klein



Com en l'atac FMS, per a cada trama s'obté un vot a un candidat a $K[3]$, i després d'analitzar totes les trames disponibles es determina el candidat més probable. Un cop decidit el valor de $K[3]$ es repeteix el procés per a la resta de bytes $K[n]$. La fórmula general per a obtenir cada byte és:

$$K[n] = V_n^{-1}[n - S[n-1]] - j_n - V_n[n]$$

Analitzem ara la probabilitat que els valors candidats trobats siguin correctes, i centrem-nos en el cas $n = 3$. Hi ha dues combinacions que fan que la hipòtesi de l'atac Klein $(S[2] + V_4[3] = 3)$ sigui certa:

- Que es donin simultàniament aquestes dues condicions:

  – L'element $V_{258}[3]$ és el mateix que $V_4[3]$. Això passa, com hem vist abans, amb una probabilitat del 37%.

  – Es compleix $S[2] + V_{258}[3] = 3$. Segons la correlació de Jenkins, la probabilitat és $2/256$.

  La probabilitat total d'aquesta primera combinació és $0,37 \times 2/256 = 0,74/256$.

- Que es donin simultàniament aquestes altres condicions:

  – L'element $V_{258}[3]$ és diferent de $V_4[3]$. La probabilitat és del 63%.

  – Coincideix que $V_4[3]$ és tal que $S[2] + V_4[3] = 3$. Segons la correlació de Jenkins, tenint en compte que ara no estem en el cas més probable, la probabilitat és $127/32640$.

Probabilitat de la hipòtesi de l'atac Klein

Mentre que la hipòtesi de l'atac FMS té una probabilitat que varia amb n, en l'atac Klein és la mateixa per a qualsevol n perquè el nombre d'iteracions considerades és constant (254).

La probabilitat total d'aquesta altra combinació és $0,63 \times 127/32640 = 0,63/256$.

Com que són combinacions disjuntes, podem sumar les probabilitats parcials i tenim que la probabilitat total que sigui certa la hipòtesi és $1,37/256$. Dit d'una altra manera, la probabilitat que es compleixi la hipòtesi de l'atac Klein és 1,37 vegades la "normal". Aquest resultat no és gaire espectacular, si el comparem per exemple amb la hipòtesi de l'atac FMS, que es complia amb una probabilitat aproximada del 5% (unes 14 vegades la normal).

La gran diferència, però, entre l'atac FMS i l'atac Klein és que en aquest últim no hi ha condicions de resolució, i totes les trames es poden usar per a obtenir un valor candidat per a cada $K[n]$. Així, encara que la probabilitat de complir-se la hipòtesi de l'atac **sigui 10 vegades menor**, en l'atac Klein es necessiten moltes menys trames per a recuperar la clau WEP.

Efecte de la correlació de Jenkins

Observeu que sense la correlació de Jenkins la probabilitat total de la hipòtesi seria $0,37 \times 1/256 + 0,63 \times 1/256 = 1/256$. És a dir, el cas $S[2] + V_4[3] = 3$ es donaria amb la mateixa probabilitat que qualsevol altre.

Experimentalment s'ha comprovat que a partir de 43.000 trames l'atac Klein permet obtenir el valor correcte de la clau WEP amb una probabilitat d'èxit del 50%. A més, a partir de 60.000 trames la probabilitat d'èxit ja és del 90%. D'altra banda, el nombre de trames necessàries en l'atac Klein és independent de com es generen els VI (mode aleatori o mode comptador), ja que no hi ha condicions de resolució i s'aprofiten totes les trames.

L'atac PTW pròpiament dit consisteix a afegir una sèrie de millores a l'atac Klein que permeten augmentar-ne l'eficiència. Aquests són alguns dels canvis introduïts en l'atac PTW:

- Mentre que en l'atac Klein, un cop determinat el valor de $K[3]$, es busca el de $K[4]$, el de $K[5]$, etc., en l'atac PTW es busquen les sumes acumulades dels bytes de la clau arrel: $\sigma_3 = K[3], \sigma_4 = K[3] + K[4], \sigma_5 = K[3] + K[4] + K[5]$, i així successivament. Aquestes sumes es poden obtenir si, en comptes de treballar per exemple amb $j_5 = j_4 + V_4[4] + K[4]$, es continua desenvolupant l'expressió i es treballa amb $j_5 = j_3 + V_3[3] + V_4[4] + K[3] + K[4]$.

  Amb aquesta modificació no es redueix el nombre de trames necessàries per a trobar la clau, i a més les probabilitats que es compleixin les hipòtesis sobre les sumes de bytes són inferiors a les de les hipòtesis de Klein. Però treballar amb les sumes té l'avantatge de fer molt més ràpida la cerca de claus alternatives quan es descobreix que una clau candidata és incorrecta. Això es deu al fet que, a diferència dels bytes $K[i]$, que s'han de calcular seqüencialment perquè cadascun depèn dels anteriors, cada suma σ_i es pot obtenir de manera independent de les altres.

  A partir de les sumes σ_3, σ_4, ... σ_{15} és immediat obtenir $K[3]$, $K[4]$, ... $K[15]$ fent unes simples restes. Si amb les sumes més votades s'obté una clau que no és la correcta, es canvia una de les sumes per la següent més votada i només cal tornar

a restar les σ_i necessàries per a obtenir una clau nova. Això és molt més ràpid que recalcular els vectors d'estat per a tornar a generar els valors nous dels bytes $K[i]$, com es fa en l'atac Klein.

- Aquesta rapidesa en l'obtenció de claus noves permet implementar algorismes més exhaustius i eficients per a trobar la clau correcta. Per exemple, si ordenem els candidats a cada σ_i de més a menys votat, podem seleccionar un nombre màxim m de candidats més votats per a utilitzar-los en les diferents combinacions de claus possibles. Treballant amb $m = 2$, hi pot haver fins a $2^{13} = 8.192$ combinacions diferents de claus per provar. Amb $m = 3$ el nombre de combinacions ja puja a $3^{13} = 1.594.323$, que pot ser molt elevat si a cada intent s'han de recalcular els vectors d'estat com en l'atac Klein.

  En canvi, l'atac PTW permet utilitzar valors m més grans, i fins i tot valors m_i que siguin diferents per a cada σ_i i que variïn dinàmicament. La tècnica dels nombres dinàmics de candidats consisteix a posar inicialment tots els valors m_i a 1, la qual cosa dóna una única combinació de bytes de la clau. Si aquesta clau no és la correcta, s'incrementa en 1 el màxim m_i corresponent a la suma σ_i en què el candidat de la posició $m_i + 1$ tingui menys diferència de vots respecte al de la posició m_i, i es tornen a provar les combinacions de claus en què intervingui el nou candidat. Si tampoc es troba la correcta, es torna a incrementar un altre m_i, i així successivament.

- L'ús de sumes de bytes σ_n en comptes de bytes $K[n]$ introdueix un problema que no existeix en l'atac Klein. Si en algun byte $K[n]$ es dóna el cas que a partir d'una certa posició $4 \leq p \leq n$ la suma $K[p]+K[p+1]+\cdots+K[n]+p+(p+1)+\cdots+n$ és igual a 0, és molt probable que l'índex j_p sigui igual a j_{n+1}. Això vol dir que a la iteració p del KSA es produirà un intercanvi que desfarà la hipòtesi de treball de l'atac PTW, i no serà aplicable la correlació de Jenkins. Com que aquesta condició és independent del VI, en aquest cas tots els valors de la suma σ_n seran més o menys equiprobables i serà molt més difícil encertar el valor correcte. Llavors es diu que $K[n]$ és un **byte fort** de la clau.

  Una solució consisteix a detectar que un byte és fort quan la distribució dels vots de les sumes candidates s'assembla a una distribució uniforme. Llavors es tracta de deduir, per a cada possible valor de p, quins són els valors de $K[n]$ que fan que es compleixi la condició de byte fort ($\sum_p^n K[i] + i = 0$), i considerar els resultats obtinguts com a candidats a $K[n]$. Altres solucions són provar per força bruta tots els valors de $K[n]$ o, si hi ha molts bytes forts a la clau i la força bruta no és viable, fer l'atac Klein en comptes del PTW.

La introducció d'aquestes millores permet a l'atac PTW rebaixar el nombre de trames necessàries per a tenir un 50% de probabilitat d'èxit fins a 35.000, i fins a 47.000 per a un 90% d'èxit.

3.4.3. Eines per a explotar les vulnerabilitats WEP

Després que es van publicar els diferents atacs contra el protocol WEP, es van desenvolupar eines que implementaven aquests atacs, moltes de codi obert o programari lliure. Una de les primeres va ser AirSnort (2001). Després, la publicació dels atacs FMS i KoreK va donar lloc al projecte Aircrack (2004). A partir d'aquest desenvolupament i d'altres, com ara l'eina Wesside (2004), es va crear Aircrack Next Generation o Aircrack-ng (2006). Des de la versió 0.9 (2007), Aircrack-ng implementa l'atac PTW.

Aircrack-ng és de fet un paquet que incorpora diverses eines, entre les quals hi ha `airmon-ng`, `aireplay-ng`, `airodump-ng` i `aircrack-ng` mateix.

L'eina `airmon-ng`

Aquesta utilitat serveix per a posar la targeta Wi-Fi en mode monitor, i d'aquesta manera poder capturar trames enviades per altres estacions. El mode monitor és equivalent al mode promiscu sense la necessitat d'establir una associació amb cap altra estació o AP.

L'eina `aireplay-ng`

Aquesta eina permet injectar trames noves o prèviament capturades. D'aquesta manera es pot forçar la generació de trames WEP de resposta en cas que no hi hagi estacions Wi-Fi actives pels voltants.

L'eina `aireplay-ng` pot treballar en diferents modes, entre els quals podem destacar els següents:

- **Mode desautenticació**. En aquest mode es generen trames falses de desautenticació destinades a una estació autenticada amb l'AP. L'objectiu és provocar que aquesta estació iniciï una nova autenticació i, depenent del sistema operatiu amb què treballi, enviï una petició ARP per tal d'esbrinar l'adreça IP de l'AP que fa d'encaminador.

- **Mode autenticació falsa**. En aquest mode es fa un atac de falsificació d'autenticació com el que hem vist anteriorment. Això pot ser necessari per a fer posteriorment altres atacs si no hi ha cap altra estació associada.

- **Mode injecció de paquets ARP**. Aquest és probablement el mode més efectiu per a forçar l'enviament de trames WEP i poder capturar així els VI necessaris per a obtenir la clau. En aquest mode, l'eina escolta el medi fins que detecta una petició ARP.

  El paquet ARP estarà xifrat, però és relativament fàcil detectar que una trama conté una petició ARP. En primer lloc perquè la longitud de les dades xifrades serà de 40

Longitud dels paquets ARP

En comptes de buscar només paquets amb 40 bytes de dades xifrades, `aireplay-ng` busca paquets que en tinguin 40 o 58 bytes. El motiu és que si l'origen és un ordinador d'una xarxa amb fil, haurà afegit bytes de farciment fins a arribar a la longitud mínima de les dades d'una trama Ethernet (46 bytes).

bytes: 8 de capçalera LLC, 28 del paquet ARP mateix, i 4 d'ICV. I després perquè l'adreça de destinació, que no està xifrada, serà l'adreça de difusió (*broadcast*).

Un cop capturada la petició ARP, l'eina `aireplay-ng` comença a retransmetre-la una vegada darrere l'altra, aprofitant la vulnerabilitat que ja hem vist d'injecció de trames. Si l'adreça IP sol·licitada és la de l'AP, aquest enviarà tantes trames WEP amb paquets ARP de resposta com peticions rebi, i cada resposta estarà xifrada amb un IV diferent. Si la petició original l'havia enviat una estació sol·licitant l'adreça d'una altra estació, l'AP retransmetrà la petició, l'estació sol·licitada enviarà la resposta a l'AP, i l'AP retransmetrà la resposta al sol·licitant original, de manera que per cada trama injectada se'n generaran 3 de noves, cadascuna amb IV propi.

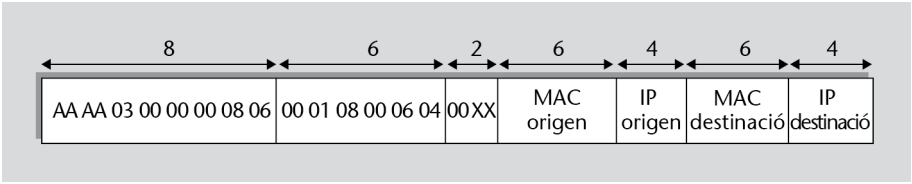
Aquest atac sol ser molt productiu, perquè normalment els filtres de paquets deixen passar sense restriccions el trànsit del protocol ARP, i els sistemes de detecció d'intrusions no prenen cap acció especial amb aquest tipus de paquets. L'estació que havia generat la petició original pot rebre moltíssimes respostes, però el més habitual és que les ignori. Amb aquesta tècnica, doncs, és fàcil obtenir uns quants centenars d'IV per segon, i aconseguir en menys d'un minut els necessaris per a fer un atac PTW.

L'eina `airodump-ng`

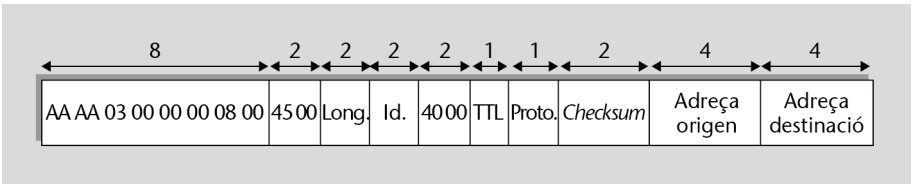
Aquesta eina és l'equivalent de la utilitat `Tcpdump` de les xarxes amb fil: captura les trames que detecta, i permet guardar-les en un fitxer. Les trames poden provenir d'un atac actiu provocat amb `aireplay-ng`, o bé ser capturades en un atac passiu simplement escoltant el medi. En aquest últim cas és molt probable que la majoria de trames capturades corresponguin a paquets IP.

Quan guarda les trames capturades en fitxer, `airodump-ng` té l'opció de guardar-hi només la informació útil per a l'eina `aircrack-ng`: l'IV i els primers bytes de *keystream* de cada trama.

Estructura d'un paquet ARP en una trama Wi-Fi



Estructura d'una capçalera IPv4 en una trama Wi-Fi



El *keystream* s'obté deduint el tipus de trama de què es tracta. Si per la seva longitud s'infereix que conté un paquet ARP, es poden conèixer com a mínim els primers 22 bytes de dades desxifrades, i restant les dades xifrades s'obté el *keystream*:

- Els primers 8 bytes són la capçalera LLC/SNAP, amb valor fix: en hexadecimal, AA:AA (punts d'accés origen i destinació), 03 (codi de control), 00:00:00 (codi d'organització) i 08:06 (tipus Ethernet: ARP).

LLC/SNAP

LLC és la sigla de *logical link control*, i SNAP és la sigla de *sub-network access protocol*.

- Els 6 bytes següents són la capçalera ARP, també amb valor fix: en hexadecimal, 00:01 (tipus de protocol: Ethernet), 08:00 (protocol de xarxa: IPv4), 06 (longitud d'adreça MAC), 04 (longitud d'adreça IP).

- Els 2 bytes següents són el codi d'operació ARP: 00:01 (petició) o 00:02 (resposta).

- Els 6 bytes següents són l'adreça MAC d'origen, que ha de coincidir amb la que hi ha a la capçalera MAC 802.11 (no xifrada).

A continuació hi ha l'adreça IP d'origen, que no es pot deduir a partir de la mateixa trama, però sí del context observant altres trames. L'adreça MAC de destinació, o bé és zero en les peticions, o es troba desxifrada a la capçalera 802.11 en les respostes. Finalment hi ha l'adreça IP de destinació, que també es pot deduir del context.

Altres tipus de trames poden contenir paquets del protocol STP o, per defecte, s'assumeix que contenen paquets IPv4. En aquest últim cas es poden obtenir els primers 12 bytes de *keystream* a partir dels valors dels bytes desxifrats, fent certes suposicions que es compleixen en la gran majoria dels casos, com per exemple que el paquet no està fragmentat:

- Els 8 primers bytes són la capçalera LLC/SNAP, igual que la de les trames amb paquets ARP, però canviant el tipus Ethernet a 08:00 (IPv4).

STP

STP és la sigla de *spanning tree protocol*.

- En els 2 bytes següents hi ha la versió del protocol, la longitud de la capçalera i el camp de serveis diferenciats. En la gran majoria de paquets IPv4 aquests dos bytes són, en hexadecimal, 45:00.

- Els 2 bytes següents són la longitud del paquet IP, que es pot deduir per la longitud de la trama.

A continuació hi ha els 2 bytes de l'identificador de datagrama, que en general tindran un valor desconegut. A l'hora de trobar la clau WEP amb l'atac PTW, Aircrack-ng fa una cerca de tots els valors possibles d'aquests dos bytes si només disposa de paquets IPv4. En els 2 bytes següents, si el paquet no està fragmentat, hi haurà els valors 40:00 o 00:00, depenent de si està activat l'indicador DF (*Don't fragment*) o no. El primer d'aquests dos bytes és l'últim que s'utilitza per a obtenir el *keystream* que necessita l'atac PTW. Aircrack-ng assumeix que un 85% dels paquets tindran l'indicador activat, i assigna el pes corresponent als vots que obtingui amb aquesta suposició. La

resta de camps de la capçalera IPv4 poden tenir valors més indeterminats, però ja no s'utilitzen en l'atac PTW.

L'eina `aircrack-ng`

Aquesta és l'eina que a partir dels IV obtinguts implementa l'atac per a recuperar la clau WEP. Per defecte aplica l'atac PTW i els atacs KoreK en paral·lel, que inclouen l'atac FMS, però té opcions per a deshabilitar els atacs que no interessi fer. També pot executar automàticament un atac Klein quan a la clau hi ha molts bytes forts resistents a l'atac PTW.

La figura següent mostra un exemple del resultat obtingut amb una execució de l'eina `aircrack-ng`. En aquest exemple l'atac ha tardat 9 segons a trobar la clau correcta a partir de poc més de 35.000 IV. Mentre prova possibles valors de la clau, l'eina va mostrant per pantalla el nombre de vots ponderats que obtenen els valors candidats de cada byte de la clau.

Exemple d'execució de l'eina Aircrack-ng

```
Starting PTW attack with 35374 ivs.

                              Aircrack-ng 1.1

                [00:00:09] Tested 147763 keys (got 35374 IVs)

   KB    depth   byte(vote)
    0    0/  1   31(45616) 71(44476) C1(44472) E0(43624) E6(42056)
    1    0/  1   43(45904) 69(43488) CE(43012) DE(42716) CF(42300)
    2    0/  1   33(45724) FF(43596) E7(42132) 44(41768) 6E(41760)
    3    0/  1   36(47692) 45(43780) F0(42432) CA(42132) 84(41952)
    4    0/  1   38(48140) D4(44004) 31(43224) CC(42204) 80(42016)
    5    0/  1   37(45680) 8C(42976) 8A(42836) C1(41428) C8(41372)
    6    0/  1   30(48868) 62(43596) 7B(42900) 18(41540) FA(41508)
    7    0/  1   42(45684) C1(43588) A8(42896) C6(42612) 24(42280)
    8    0/  1   34(45860) AD(43920) 78(43480) 10(41912) 9E(41904)
    9    0/  3   61(43124) A7(42932) 28(42824) C1(42712) 25(41736)
   10    5/  1   9E(41948) 39(41584) 87(41400) F7(41364) 78(40992)
   11    0/  1   0C(44696) C3(42636) A6(42344) 9A(41948) 12(41840)
   12    0/  1   35(45568) C4(42680) 11(41916) 02(41912) CA(41912)

     KEY FOUND! [ 31:43:33:36:38:37:30:42:34:36:41:31:35 ] (ASCII: 1C36870B46A15 )
        Decrypted correctly: 100%
```

3.5. Solucions a les vulnerabilitats WEP

Quan es va veure que el disseny inicial de la seguretat en l'estàndard IEEE 802.11 tenia deficiències importants, es van proposar diverses solucions per a intentar corregir aquests problemes.

Entre les propostes que es van fer podem destacar les següents:

- En l'article que descrivia l'atac FMS, de l'any 2001, els seus autors suggerien aplicar-lo a les trames en les quals l'IV comença per $K[0] = n$ i $K[1] = 255$ quan s'està buscant el valor $K[n](3 \leq n < 8)$, perquè en aquests casos és més fàcil que es compleixi la hipòtesi de l'atac. Per tant, una primera solució era no enviar trames WEP xifrades amb aquests IV, anomenats **vectors d'inicialització febles**. La realitat és que aquesta mesura només incrementa molt lleugerament el nombre de trames que necessita l'atacant per a descobrir la clau, però tot i així la majoria de sistemes operatius la van incorporar al seu nucli, i a les targetes Wi-Fi que la implementaven en el seu maquinari els van donar l'etiqueta comercial **WEPplus**.

- Una altra solució proposada l'any 2001, anomenada **WEP2**, es basava en l'ús de vectors d'inicialització de 128 bits i claus arrel secretes també de 128 bits, és a dir, claus RC4 de 256 bits en total. Això de fet no impedeix fer els atacs dissenyats per al protocol WEP original, però sí que allarga el temps necessari per a completar-los amb èxit, tot i que el creixement no és exponencial sinó només lineal.

- Alguns fabricants van optar per una solució més efectiva, coneguda com **Dynamic WEP**. Com el nom indica, aquesta tècnica consisteix a anar canviant dinàmicament les claus WEP, cosa que complica considerablement els atacs respecte a les claus estàtiques. Les diferents implementacions, però, no eren interoperables entre si perquè cada fabricant seguia les seves pròpies especificacions.

A partir de la proposta de les claus dinàmiques van començar els treballs d'estandardització que donarien lloc a la publicació de l'especificació **IEEE 802.11i** l'any 2004. En l'edició de 2007, aquesta extensió va deixar de ser una especificació separada i es va incorporar com un capítol de l'estàndard base IEEE 802.11.

WPA

WPA és la sigla de *Wi-Fi protected access*.

Mentre s'estava elaborant el text d'aquesta especificació, i atesa la urgència per a resoldre els problemes que presentava el protocol WEP, l'associació de fabricants Wi-Fi Alliance va desenvolupar una solució intermèdia anomenada **WPA**, amb la intenció que es pogués utilitzar amb el mateix maquinari de les targetes Wi-Fi existents, o introduint-hi només unes quantes modificacions al microprogramari (*firmware*). Aquesta solució estava basada en els esborranys que anava publicant el grup de treball IEEE 802.11i. Quan se'n va aprovar la versió oficial l'any 2004, l'estàndard IEEE 802.11i va ser incorporat a les especificacions de la Wi-Fi Alliance amb el nom de **WPA2**.

3.5.1. WPA

L'estàndard WPA introdueix canvis fonamentals tant en el mètode d'autenticació de les estacions, com en l'algorisme de xifratge de les trames.

A diferència del protocol WEP, en què normalment hi ha una sola clau secreta compartida per l'AP i les estacions, WPA preveu l'ús de claus diferents en cada **associació segura**, és a dir en cada RSNA, i defineix els mecanismes per a establir aquestes claus dinàmicament.

RSNA

RSNA és la sigla de *Robust Security Network Association*.

L'ús d'una clau única compartida entre l'AP i les estacions, com preveu el protocol WEP, pot ser apropiada per a una xarxa sense fil domèstica, però és més problemàtica en una xarxa corporativa mitjana o gran. Quan hi ha desenes o centenars d'estacions amb la mateixa clau, si un atacant accedeix a la clau en una de les estacions, automàticament les comunicacions de totes les altres queden compromeses. A més, canviar la clau pot requerir actualitzacions manuals en cadascuna de les estacions, cosa que pot ser poc pràctica.

Per això, WPA preveu l'ús del mètode de control d'accés a la xarxa definit en un altre estàndard de la sèrie IEEE 802, concretament l'**IEEE 802.1X**. Aquest estàndard facilita l'intercanvi segur de claus de sessió entre dos nodes de la xarxa, amb una autenticació mútua prèvia. Que l'autenticació sigui mútua en WPA implica que l'estació s'autentica davant l'AP, però l'AP també s'autentica davant l'estació, perquè aquesta es pugui assegurar que no està parlant amb un AP falsificat.

EAP

EAP és la sigla d'*extensible authentication protocol*. Aquest protocol està definit a l'especificació RFC 3748.

L'estàndard IEEE 802.1X, al seu torn, es basa en el protocol EAP, que permet dur a terme una autenticació treballant al nivell d'enllaç, és a dir, sense necessitat de tenir assignada encara una adreça de xarxa (IP). EAP preveu l'ús de diversos mètodes d'autenticació i, com el nom indica, se n'hi poden afegir altres de definits en altres especificacions. Així, per mitjà de l'EAP es pot fer una autenticació basada, per exemple, en noms d'usuari i contrasenyes, en claus públiques i certificats X.509, en dispositius físics com ara targetes amb xip, etc.

EAPOL

EAPOL és la sigla de *EAP over LANs*.

El que fa IEEE 802.1X és definir un format de trames anomenat EAPOL per a enviar els missatges del protocol EAP per una xarxa local. D'altra banda, en la terminologia IEEE 802.1X l'extrem de la comunicació EAP que sol·licita l'autenticació s'anomena **suplicant**, i l'altre extrem, el que la concedeix, s'anomena **autenticador**. L'autenticador pot concedir l'autenticació per si mateix, o bé pot comunicar-se amb un **servidor d'autenticació** que pren la decisió final. Típicament el servidor utilitzarà un protocol com ara RADIUS o Diameter per a dur a terme l'autenticació.

PSK

PSK és la sigla de *pre-shared key*.

WPA també continua permetent l'ús d'una clau compartida o PSK, per simplicitat en el cas de xarxes petites com solen ser les xarxes domèstiques. Però en aquest cas la clau de xifratge no és directament la clau compartida més un VI, com en el protocol WEP, sinó que la clau compartida s'utilitza per a derivar les corresponents claus de sessió per a cada associació.

En qualsevol cas, cada parell estació-AP utilitza les seves pròpies claus per a protegir les seves comunicacions. Així s'aconsegueix que una estació no pugui espiar les trames enviades entre l'AP i una altra estació del mateix BSS.

Aquest esquema, però, té un inconvenient: mentre que amb una clau compartida és fàcil enviar una trama xifrada simultàniament a més d'un node, com en el cas del trànsit de difusió (*broadcast*) o de difusió selectiva (*multicast*), amb claus independents seria necessari enviar tantes trames com destinataris, cadascuna xifrada amb la clau corresponent. Per a evitar aquesta ineficiència, en WPA es treballa amb dos tipus de claus:

- Les **claus entre parelles** són les que s'utilitzen per a les trames entre cada parell de nodes, és a dir entre l'AP i cada estació.

- Les **claus de grup** són conegudes per tots els membres del BSS i s'utilitzen per a les trames de difusió (*broadcast*) o de difusió selectiva (*multicast*). Es pot generar una clau de grup nova cada vegada que una estació abandona el BSS i es dissocia de l'AP, per a evitar que pugui continuar desxifrant el trànsit del grup.

Autenticació WPA i gestió de claus

WPA defineix dos modes d'autenticació:

- El **mode WPA-PSK**. És el que treballa amb una clau mestra predefinida, compartida entre l'AP i les estacions. Com hem comentat abans, sol usar-se només en xarxes amb poques estacions. La Wi-Fi Alliance també va donar a aquest mode el nom **WPA-Personal**.

- El **mode WPA-802.1X**. És el que utilitza el control d'accés basat en IEEE 802.1X més EAP, juntament amb un servidor d'autenticació. La Wi-Fi Alliance també va donar a aquest mode el nom **WPA-Enterprise**.

Tant si s'utilitza un mode com l'altre, una estació que vol entrar en un ESS ha de seguir aquests passos:

1) L'estació ha d'identificar l'ESS al qual vol accedir. Mitjançant les trames balisa descobreix la informació que necessita sobre l'ESS i l'AP que el gestiona, com ara el BSSID (és a dir, l'adreça MAC de l'AP), les velocitats de transmissió suportades, etc.

Entre els camps de la trama balisa, també anomenats *IE*, n'hi pot haver un de tipus RSN (*robust security network*). Si aquest IE és present, vol dir que l'AP suporta l'establiment d'associacions segures WPA. Els diferents subcamps de l'IE RSN indiquen els algorismes d'autenticació i de xifratge suportats.

IE

IE és la sigla d'*information element*.

2) Per compatibilitat amb els sistemes que implementen la màquina d'estats 802.11, l'estació primer ha de fer una autenticació de sistema obert, com hem vist a l'apartat 3.2, seguida d'una associació 802.11.

La trama de gestió que conté la sol·licitud d'associació inclou un IE de tipus RSN en què s'especifiquen l'algorisme d'autenticació i el de xifratge que l'estació està disposada a utilitzar, entre els anunciats per l'AP en les trames balisa. L'algorisme d'autenticació escollit determina si aquesta es farà en mode WPA-PSK o en mode WPA-802.1X.

Autenticació de clau compartida

Recordeu que l'autenticació de clau compartida és totalment insegura, i per això l'estàndard WPA no la recull.

3) L'estació i l'AP estableixen de manera segura una **clau mestra entre parelles** o PMK de 256 bits.

- Si es treballa en mode WPA-PSK, la clau mestra PMK és directament la clau compartida PSK prèviament configurada.

  Moltes vegades, per a facilitar la configuració de les estacions, la PSK no s'especifica directament sinó com una frase de pas (*passphrase*). En aquests casos els bits de la PSK són el resultat d'aplicar una funció de generació de claus, definida en l'estàndard PKCS#5 i basada en funcions resum (*hash*), a partir de la frase de pas i l'SSID.

- Si es treballa en mode WPA-802.1X, s'inicia el protocol EAP per a dur a terme l'autenticació. L'estació es posa d'acord amb l'autenticador, que pot ser l'AP o un servidor d'autenticació, sobre el mètode EAP que s'utilitzarà. Aquest mètode ha de garantir que un espia que observi la comunicació no pugui obtenir cap contrasenya o altra informació secreta que li permeti fer una autenticació fraudulenta.

  Llavors suplicant i autenticador s'intercanvien els missatges EAP necessaris fins a completar l'autenticació. Si el procés acaba amb èxit, el resultat és que l'estació i l'AP s'han autenticat mútuament de manera satisfactòria, i a més el mètode EAP utilitzat també ha de proporcionar un valor de 512 bits que s'utilitzarà com a **clau mestra de sessió** o MSK.

  Finalment s'obté la PMK, que és igual als primers 256 bits de l'MSK.

PMK i MSK

PMK és la sigla de *pairwise master key*, i MSK és la sigla de *master session key*.

4) L'autenticació es completa executant una **negociació en 4 passos** o *4-way handshake*. D'una banda, aquesta negociació permet verificar que tant l'AP com l'estació han obtingut correctament la clau mestra PMK, i d'aquesta manera comprovar que són els autèntics. I d'altra banda, com a resultat de la negociació s'obtenen també les claus necessàries per a protegir les trames WPA, tant entre parelles com de grup.

Els missatges de la *4-way handshake* s'envien en un tipus especial de trama, anomenat *EAPOL-Key*, definit en l'estàndard IEEE 802.1X. Durant la negociació s'obtenen una clau de xifratge i una clau d'autenticació de missatge, anomenades *KEK* i *KCK* respectivament, per ser utilitzades exclusivament en la negociació. Els missatges de la negociació s'envien xifrats amb RC4, excepte els dos primers perquè la clau KEK encara no està disponible, i autenticats amb HMAC-MD5, excepte el primer perquè la clau KCK tampoc està disponible. A més, un dels camps de les trames EAPOL-Key és un comptador per a detectar atacs de repetició.

KEK, KCK, PTK i GTK

KEK és la sigla de *key encryption key*, KCK és la sigla de *key confirmation key*, PTK és la sigla de *pairwise transient key*, i GTK és la sigla de *group temporal key*.

L'intercanvi de missatges en la *4-way handshake* és el següent:

1) L'autenticador (AP) envia al suplicant (estació) un valor aleatori N_A.

2) El suplicant genera un altre valor aleatori N_S i calcula la **clau transitòria entre parelles** o PTK, de 512 bits. El càlcul es fa aplicant una funció unidireccional a la clau mestra PMK, els valors aleatoris N_A i N_S, i les adreces MAC d'autenticador i suplicant. Llavors s'obtenen les claus KCK i KEK prenent els primers 128 bits de la PTK i els 128 bits següents, respectivament.

Un cop obtingudes aquestes claus, el suplicant envia el valor N_S a l'autenticador.

3) L'autenticador fa els mateixos càlculs per a obtenir la clau PTK, i a partir d'aquesta, la KCK i la KEK. Llavors envia la **clau temporal de grup** o GTK al suplicant, xifrada amb la KEK.

4) El suplicant comprova que el missatge anterior és correcte. Si és així, l'autenticitat de l'autenticador (AP) haurà quedat confirmada. El suplicant envia llavors a l'autenticador un missatge que no cal que contingui res en el seu camp de dades. Si l'autenticador veu que és correcte, l'autenticitat del suplicant (estació) també haurà quedat confirmada.

Com a resultat del procés anterior, l'AP i l'estació han acordat una clau PTK que utilitzaran entre si. D'aquesta PTK, prenent els bits 256-511, s'obté una **clau temporal** o TK, que serà la que es farà servir per a xifrar i autenticar les trames WPA.

TK

TK és la sigla de *temporal key*.

En la negociació, a més, l'AP envia a l'estació la clau de grup GTK. Aquesta clau de grup la determina unilateralment l'AP. La manera d'obtenir-la és una qüestió interna de l'AP però, per analogia amb la PTK, es pot obtenir per exemple aplicant una funció unidireccional a una clau mestra de grup o GMK més un valor aleatori N_G.

Un cop establerta la sessió, el protocol *4-way handshake* es pot tornar a iniciar en qualsevol moment per a renegociar la clau transitòria PTK; per exemple quan la sessió és llarga i ja fa cert temps que s'està utilitzant la mateixa clau. En aquest cas, l'enviament de la clau GTK és opcional si no ha canviat.

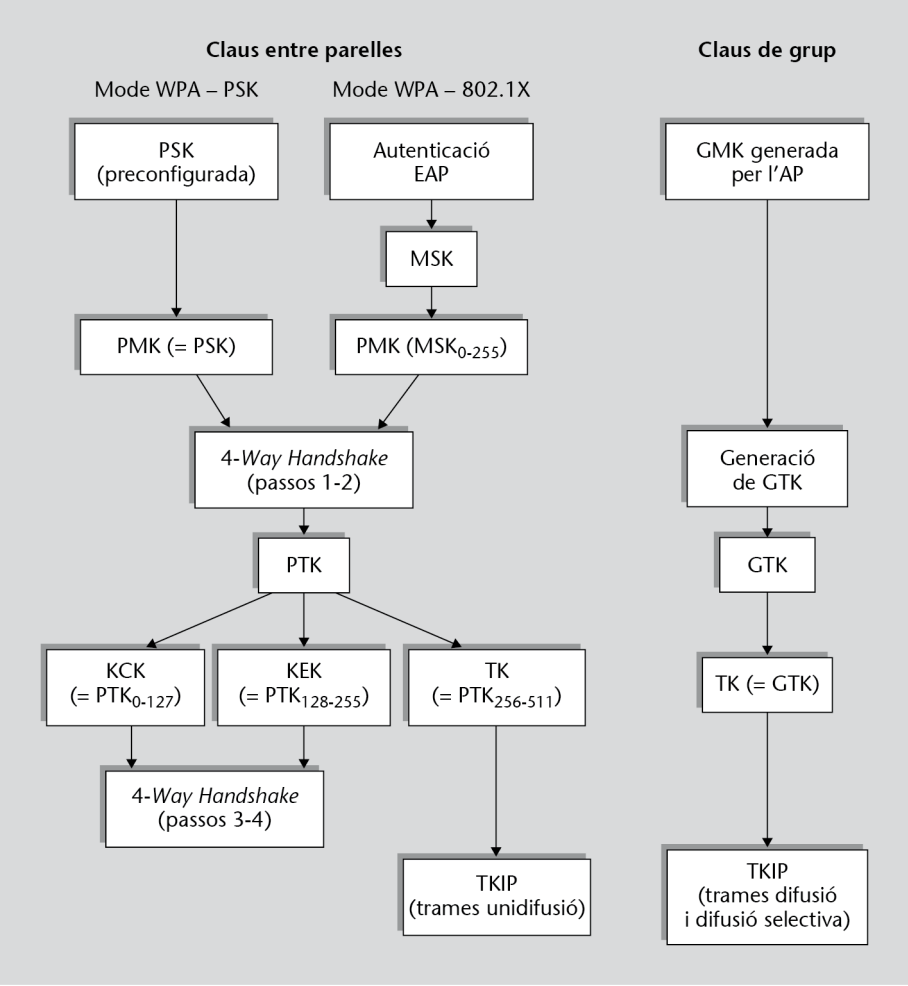
I en el moment en què canviï la GTK, per exemple perquè una estació ha sortit del BSS i ja no ha de continuar rebent trànsit de difusió, es fa un altre tipus de negociació anomenat *group key handshake* entre l'AP i cada estació. Aquesta negociació té dos passos perquè només cal enviar la nova clau GTK xifrada a l'estació, i que aquesta confirmi que l'ha rebut.

Com es pot comprovar, l'autenticació WPA introdueix fortes mesures de seguretat per a evitar qualsevol tipus d'atac: un mecanisme segur per a derivar una clau mestra PMK que és diferent per a cada sessió amb cada estació (excepte en el mode WPA-PSK), una clau transitòria PTK que es pot anar canviant periòdicament, i un protocol de generació de claus temporals en 4 passos afegit al mètode d'autenticació, amb l'ús de claus criptogràfiques independents de les de la comunicació normal, derivades de les adreces MAC i amb comptadors per a evitar atacs de repetició.

Una feblesa d'aquest esquema és l'ús dels algorismes RC4 i MD5 per al xifratge i l'autenticació de la negociació en 4 passos, que no són tan segurs com altres algorismes que s'han desenvolupat posteriorment. Però l'objectiu inicial de l'estàndard WPA era que es pogués utilitzar amb el maquinari disponible, i aquesta era una solució de compromís mentre no s'estenia la implementació del WPA2. D'altra banda, en el mode WPA-PSK una estació que capturi els valors N_A i N_S de la negociació d'una altra estació, que no s'envien xifrats, immediatament sabrà quina és la seva clau PTK i podrà desxifrar-ne el trànsit.

A mode de resum, el diagrama següent mostra les relacions entre les diferents claus que formen l'anomenada *jerarquia de claus WPA*.

Jerarquia de claus WPA



Mètodes d'autenticació EAP usats en WPA-802.1X

Actualment hi ha desenes de mètodes EAP, entre els estandarditzats per l'IETF i els definits per diversos fabricants. Alguns dels usats més habitualment en el mode WPA-802.1X són els següents:

- **EAP-TLS**. En aquest mètode la comunicació amb el servidor d'autenticació, per exemple un servidor RADIUS, es protegeix mitjançant el protocol TLS amb autenticació mútua basada en certificats de servidor i de client.

- **EAP-TTLS** (*EAP-Tunneled TLS*). És una variant simplificada del mètode anterior en què no són necessaris els certificats de client, la qual cosa el fa molt més pràctic. S'utilitza el protocol TLS per a crear un canal segur o "túnel" només amb certificat de servidor. Llavors es fa l'autenticació del client amb un altre mètode, que pot ser per exemple basat en contrasenya, per mitjà d'aquest canal segur.

TLS

El protocol de seguretat TLS (*transport layer security*) s'estudia en l'apartat 5 d'aquest mòdul.

- **PEAP** (*Protected EAP*). És un mètode genèric per encapsular l'autenticació de client dins un altre mètode amb autenticació de servidor, per exemple basat en TLS.

A més dels passos per a fer l'autenticació, cadascun d'aquests mètodes ha de definir també com es genera el valor que es farà servir com a clau mestra de sessió (MSK).

Els mètodes com EAP-TTLS o PEAP estableixen l'autenticació del servidor, però llavors cal usar un altre mètode per a l'autenticació del client. Aquest altre mètode pot ser, per exemple:

Generació de l'MSK

En els mètodes EAP basats en TLS, l'MSK es genera normalment aplicant una funció unidireccional a les cadenes de bits aleatòries utilitzades en la fase de negociació TLS (*handshake protocol*) i el secret mestre que se n'obté.

- **EAP-MD5**. És un mètode de repte-resposta. La resposta és un *hash* MD5 d'una cadena formada per la contrasenya del client més el repte.

- **EAP-MSCHAPv2** (*EAP-Microsoft challenge handshake authentication protocol version 2*). Utilitza el protocol MSCHAPv2, definit en l'especificació RFC 2759.

- **EAP-GTC** (*EAP-generic token card*). També és un mètode de repte-resposta en el qual la resposta és generada per un dispositiu físic, com pot ser una targeta amb xip.

El xifratge TKIP

A més del mètode d'autenticació, l'altre canvi fonamental introduït en WPA respecte a WEP és l'algorisme de xifratge. O més exactament, la generació de les claus de xifratge, ja que l'algorisme pròpiament dit és el mateix: RC4. Això es va decidir, com ja hem vist abans, per a intentar aprofitar el maquinari de les targetes de xarxa que hi havia llavors.

L'esquema de xifratge que s'utilitza en l'estàndard WPA s'anomena *TKIP*. Les diferències principals que presenta respecte a l'esquema WEP són les següents:

- La clau amb què s'encripten les dades de cada trama no s'obté a partir d'un vector d'inicialització variable i una part fixa, sinó que tots els bits de la clau RC4 es recalculen a cada trama.

TKIP

TKIP és la sigla de *temporal key integrity protocol*.

- Les trames TKIP incorporen un codi MIC calculat a partir d'una clau secreta, com a prevenció contra els atacs de modificació o truncament com el *chopchop*. El codi MIC no substitueix sinó que complementa el camp ICV. D'altra banda, quan es produeix fragmentació aquest codi es calcula sobre la trama original abans de fragmentar, en comptes d'haver-hi un codi MIC per cada fragment.

- Per a evitar atacs d'injecció, el codi MIC no es calcula només sobre les dades xifrades sinó que també s'hi afegeixen les adreces MAC de les estacions origen i destinació.

- Cada trama inclou un comptador de seqüència de 48 bits, anomenat *TSC*, com a mesura contra els atacs de repetició. Aquest comptador es reinicialitza a 1 cada vegada que es fa servir una clau temporal TK nova. El comptador N d'una trama enviada després que una altra amb comptador M ha de complir $N > M$ (si són trames consecutives, pot ser per exemple $N = M + 1$, però no necessàriament). Les trames rebudes que no segueixin aquesta regla són descartades.

MIC

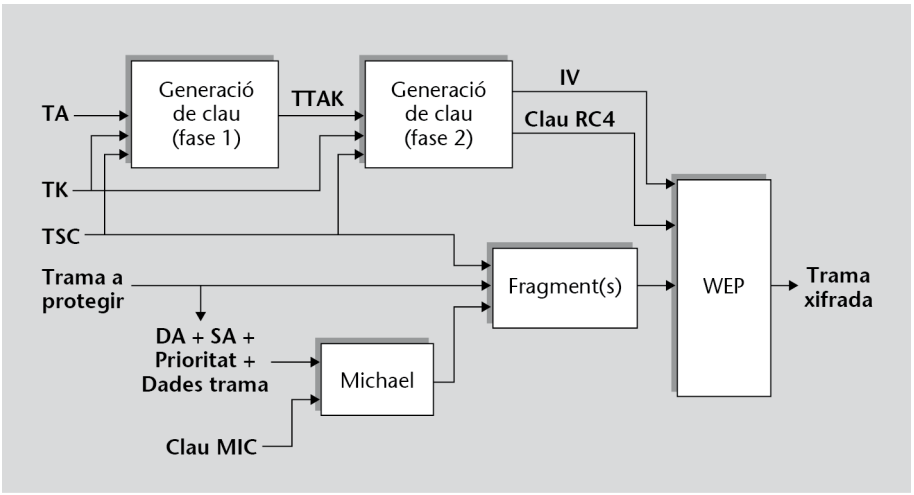
MIC és la sigla de *message integrity code*, que és la nomenclatura amb què IEEE 802.11 es refereix al codi d'autenticació de missatge, per a evitar confusions amb *medium access control* (MAC).

Algunes trames poden tenir assignada una prioritat, i pot passar que trames de prioritats diferents siguin rebudes en ordre diferent al d'enviament. Per tant, emissor i receptor han de mantenir un comptador TSC independent per cada prioritat utilitzada (n'hi pot haver com a màxim 8 de diferents).

TSC

TSC és la sigla de *TKIP sequence counter*.

Generació de trames xifrades TKIP



El procés que se segueix en l'algorisme TKIP per a generar una trama xifrada inclou els passos següents:

1) Es genera el codi MIC aplicant un algorisme anomenat *Michael* a les entrades següents:

- La informació següent de la trama que es vol protegir: l'adreça MAC de destinació (DA), l'adreça MAC d'origen (SA), la prioritat, i el camp de dades.

- La clau MIC de 64 bits. Per a evitar atacs de repetició en sentit contrari s'utilitzen dues claus MIC diferents per a les trames de l'AP a l'estació i per a les

trames de l'estació a l'AP. La primera s'obté dels bits 128-191 de la clau TK, i la segona dels bits 192-255 de la mateixa clau.

L'algorisme Michael és una funció resum senzilla amb operacions simples que es pot calcular molt ràpidament. No és, però, una funció resum segura perquè no té la propietat de la unidireccionalitat.

2) En cas que sigui necessari, s'aplica la fragmentació a la trama més el codi MIC. A cada fragment se li assigna un comptador TSC diferent, sempre respectant l'ordre creixent.

3) S'aplica una funció criptogràfica, anomenada **fase 1**, a les entrades següents:

- La clau temporal TK obtinguda en la *4-way handshake*. Com a clau per al xifratge s'utilitzen els primers 128 bits (0-127) de la clau TK.
- L'adreça MAC de l'estació transmissora, TA.
- El comptador TSC. Per a la fase 1 s'utilitzen els 24 bits de més pes del comptador.

El resultat de la fase 1 és un valor TTAK (*TKIP-mixed transmit address and key*) de 80 bits. Aquest valor serà el mateix per a totes les trames que tinguin els mateixos 24 bits de més pes del comptador TSC, i per tant no caldrà recalcular-lo cada vegada.

4) A continuació s'aplica una altra funció criptogràfica, anomenada **fase 2**, a les entrades següents:

- El resultat TTAK de la fase 1.
- La mateixa clau de xifratge que en la fase 1 (els bits 0-127 de la clau TK).
- El comptador TSC. Per a la fase 2 s'utilitzen els 24 bits de menys pes del comptador.

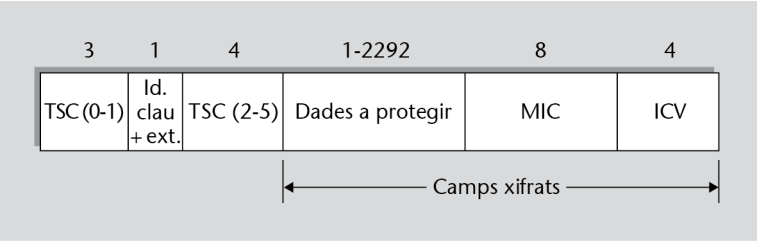
El resultat de la fase 2 és una clau de xifratge RC4 de 128 bits, amb 24 bits d'IV i 104 bits de clau arrel.

La propietat principal del xifratge TKIP és que els 104 bits de clau arrel són diferents per a cada trama, amb la qual cosa els atacs estadístics contra el xifratge WEP no són aplicables.

L'IV es construeix de manera que els bytes primer i tercer es copien dels 16 bits de menys pes del TSC, i el segon byte es deriva del primer amb la precaució que el resultat no sigui un VI feble, és a dir, que no tingui el segon byte igual a 255.

5) Finalment la trama, o cada fragment de trama si és el cas, se xifra igual que en el protocol WEP.

Dades d'una trama TKIP



L'estructura de la trama xifrada TKIP que es genera és lleugerament diferent de la de les trames WEP normals.

- El primer camp conté l'IV, igual que en les trames WEP, en aquest cas obtingut a partir dels dos bytes de menys pes del TSC.

- El camp següent té activat un bit d'extensió per a indicar que a continuació hi ha un camp addicional.

- El camp addicional d'extensió s'aprofita per a incloure-hi la resta de bytes del TSC (2-5).

- A continuació de les dades de la trama, i abans del camp ICV, s'insereixen els 64 bits del codi MIC.

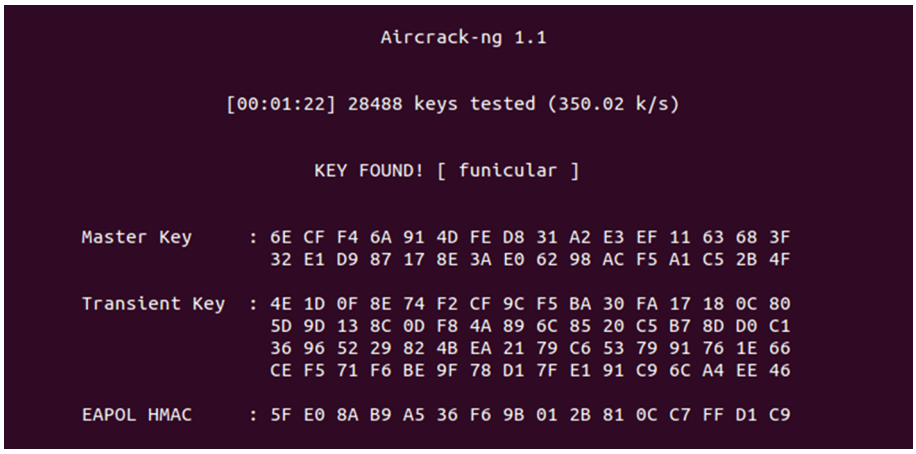
Vulnerabilitats i contramesures

La vulnerabilitat principal del sistema WPA es troba en l'ús del mode d'autenticació de clau compartida, WPA-PSK. Encara que es treballi amb una clau mestra PMK de 256 bits, si aquesta clau prové exclusivament d'una paraula més o menys fàcil de recordar, l'espai de claus possibles queda molt reduït i fa viable un atac per força bruta. Per exemple, l'eina `aircrack-ng` permet fer un atac de diccionari sobre els paquets de la negociació *4-way handshake* entre una estació i l'AP. A diferència dels atacs WEP, que com més trames tinguin disponibles més ràpidament poden trobar la clau, l'atac de diccionari WPA només necessita les trames d'una sola negociació. De fet, amb dues de les quatre trames n'hi ha prou. Aquestes trames es poden obtenir amb `airodump-ng`, esperant de manera passiva que alguna estació estableixi una associació amb l'AP o bé provocant de manera activa l'associació amb el mode desautenticació de l'eina `aireplay-ng`. L'atac consisteix a anar provant, per a cada paraula del diccionari, si les claus que se'n deriven quadren amb el contingut xifrat i autenticat de les trames capturades.

La conclusió és que si s'utilitza el mode WPA-PSK cal escollir una clau que no sigui cap paraula de diccionari en cap idioma ni una combinació trivial de paraules (per exemple, una paraula escrita del revés). S'aconsella utilitzar frases llargues, de 20 caràcters com a mínim, que no continguin paraules de diccionari.

La figura següent mostra l'exemple d'una execució de l'eina `aircrack-ng` per a descobrir una clau WPA-PSK amb l'atac de diccionari. En menys d'un minut i mig, i després de provar poc més de 28.000 paraules de diccionari, en aquest exemple l'eina ha trobat que la clau és la paraula *funicular*.

Exemple d'execució d'un atac de diccionari WPA



Pel que fa al protocol TKIP, com hem vist abans, el seu disseny inclou un seguit de proteccions criptogràfiques per a evitar els atacs d'injecció, de repetició, de modificació, de truncament, etc. Però a més l'especificació inclou també una mesura en el funcionament del protocol per a intentar contrarestar els atacs contra el codi MIC. L'objectiu és evitar atacs d'assaig i error, de l'estil de l'atac *chopchop* sobre l'ICV. Si un atacant aconseguís trencar el codi MIC podria injectar trames correctes.

Per a evitar-ho, el protocol TKIP està dissenyat per a alentir la velocitat a la qual es poden fer els intents d'atac. Concretament, l'especificació estableix que les trames amb els camps CRC, ICV o TSC incorrectes han de ser ignorades. Però si aquests camps són correctes i el codi MIC és erroni, això ha de ser considerat com un possible atac i senyalitzat com a tal en els registres (*logs*) de seguretat. L'estació que detecti l'error ha d'enviar a l'AP un tipus especial de trama EAPOL-Key, anomenat *Michael MIC failure report*. I si es detecten dues trames errònies en menys de 60 segons, s'ha de deshabilitar la recepció de trames TKIP durant un minut. Quan es restableixi, les claus haurien de ser renegociades.

Això implica que un atacant no podrà fer més de dos intents per minut. Tot i així s'han proposat atacs, com l'anomenat *atac Beck-Tews*, que en certes condicions teòricament permetrien a un atacant desxifrar els últims 12 bytes d'una trama (MIC i ICV) en poc més de 12 minuts. Si és una trama ARP amb contingut conegut, es pot obtenir fàcilment la clau MIC ja que l'algorisme Michael no està dissenyat per a ser unidireccional. I en 4 o 5 minuts més es podria obtenir prou *keystream* per a poder injectar determinats tipus de trames.

3.5.2. WPA2

L'especificació WPA2 incorpora tota la funcionalitat de l'estàndard IEEE 802.11i. Els canvis que introdueix respecte a WPA són de dos tipus:

- D'una banda defineix mecanismes com la preautenticació i l'emmagatzemament de claus mestres (*PMK caching*), que fan més ràpida i eficient la reautenticació d'una estació mòbil quan surt d'un BSS i entra en un BSS adjacent del mateix ESS (*roaming*).

- D'altra banda introdueix un algorisme de xifratge nou, anomenat **CCMP**, que no està basat en l'RC4 sinó en la xifra AES-128. Aquest mètode de xifratge és molt més segur, ja que actualment no se'n coneixen vulnerabilitats significatives, i tot i que no és tan senzill d'implementar com l'RC4, és força més eficient que la majoria d'altres xifres de bloc existents.

CCMP

CCMP és la sigla de *CTR with CBC-MAC protocol*.

Els sistemes WPA2 han de suportar obligatòriament el xifratge CCMP. L'ús del xifratge TKIP és opcional, per compatibilitat amb els sistemes WPA. D'altra banda, quan es treballa amb CCMP els algorismes criptogràfics utilitzats en la *4-way handshake* són AES (segons l'estàndard RFC 3394) per al xifratge i HMAC-SHA1 per a l'autenticació de missatge, en comptes de RC4 i HMAC-MD5 respectivament.

Bits de la clau TK

Els primers 128 bits de la clau TK són els únics que es necessiten en el protocol CCMP perquè s'utilitzen tant per a xifrar com per a autenticar. Quan es treballa amb CCMP, doncs, no cal generar 512 bits de PTK en els passos 2 i 3 de la *4-way handshake* sinó que n'hi ha prou amb 384.

El xifratge CCMP consisteix a aplicar el mode CCM, definit a l'especificació RFC 3610, a la xifra de bloc AES amb clau de 128 bits. El mode CCM proporciona alhora autenticació de missatge i confidencialitat, tot amb la mateixa clau. La clau CCMP és de 128 bits i s'obté, com en TKIP, dels bits 0-127 de la clau temporal TK.

- El codi d'autenticació MAC es genera fent un xifratge AES-128 en mode CBC, amb la tècnica coneguda com a *CBC-MAC*. El vector d'inicialització, segons l'especificació CCM, té 1 byte d'indicadors, 2 bytes que indiquen la longitud de les dades, i els altres 13 bytes han de ser únics per a cada trama. Per a aconseguir-ho, aquests 13 bytes es construeixen de la manera següent:

  – 1 byte codifica la prioritat.

  – Els 6 bytes següents són l'adreça MAC (*medium access control*) de l'estació transmissora.

  – Els últims 6 bytes són un **número de paquet** o PN (*packet number*) de 48 bits que s'incrementa a cada trama.

  Després d'aquest vector d'inicialització, els 2 blocs següents que es xifren contenen una combinació dels camps invariants de la capçalera MAC (*medium access control*) de la trama, que són bàsicament tots excepte el camp Durada/ID.

  A continuació d'aquests 2 blocs es xifren les dades de la trama, completades al final amb bytes iguals a 0 si la seva longitud no és múltipla de 16. Del resultat de

xifrar l'últim bloc es prenen els primers 64 bits, i aquest serà el codi MAC que autenticarà la trama.

- Les dades xifrades s'obtenen aplicant un xifratge en el mode comptador (*CTR mode*) segons la terminologia CCM. Per a cada bloc de dades que es vol xifrar, es construeix un bloc auxiliar que té 1 byte d'indicadors, 2 bytes que contenen un comptador igual a 1 per al primer bloc i que s'incrementa a cadascun dels blocs següents, i els altres bytes són iguals als 13 bytes únics que s'utilitzen per a generar el codi MAC.

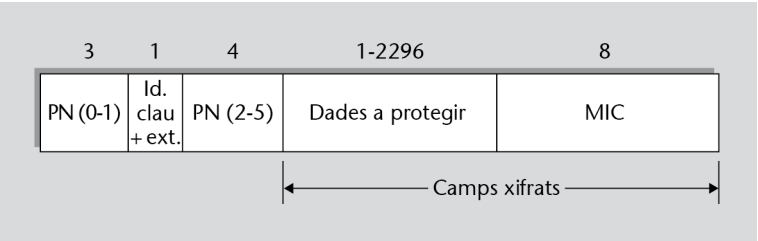
  Llavors se xifra aquest bloc auxiliar amb AES-128, i el resultat se suma bit a bit (amb l'operació XOR) amb el bloc corresponent que es vol xifrar, com si fos una xifra de flux. Si la longitud de l'últim bloc és menor de 16 bytes, només s'utilitzen els bytes del bloc auxiliar xifrat que calguin.

Terminologia CCM

En el context de l'especificació CCM, MAC significa *message authentication code*, i vector d'inicialització té el sentit que se li dóna normalment en les xifres de bloc, és a dir, el bloc aleatori que s'utilitza com si fos l'anterior al primer bloc que es vol xifrar.

El codi MAC obtingut en el primer pas també se xifra sumant-lo amb un altre bloc auxiliar xifrat, en el qual el comptador és igual a 0. El resultat és el codi MIC, segons la terminologia WPA.

Dades d'una trama CCMP



Després d'aplicar l'autenticació de missatge i el xifratge ja es pot generar la trama xifrada CCMP. La seva estructura és semblant a la de les trames TKIP, substituint els bytes del comptador TSC pels del número de paquet PN, i amb la diferència principal que en CCMP no s'inclou el camp ICV. Aquest camp, que en TKIP serveix per a reforçar el codi MIC basat en l'algorisme Michael, no es considera necessari en CCMP atesa la fortalesa de l'autenticació CBC-MAC amb la xifra AES.

Els únics atacs pràctics que es coneixen sobre el sistema WPA2 són els mateixos que afecten el sistema WPA quan s'utilitza el mode PSK. És a dir, WPA2-PSK és exactament igual de vulnerable a atacs de diccionari que WPA-PSK, ja que aquests atacs actuen sobre la negociació *4-way handshake* i són independents de l'algorisme de xifratge de les trames.

4. Protecció del nivell de xarxa: IPsec

En l'apartat anterior hem vist la protecció bàsica de les xarxes de comunicació sense fil. En aquest apartat i posteriors veurem la protecció de protocols de comunicacions de nivells més alts, aplicables tant a xarxes cablejades com sense fil.

En aquesta secció veurem l'arquitectura IPsec, dissenyada per protegir el protocol de xarxa usat a Internet, és a dir, el protocol IP. A la secció següent veurem mecanismes per protegir les comunicacions a nivell de transport, i en el mòdul d'aplicacions segures veurem exemples de protecció dels protocols a nivell d'aplicació.

4.1. L'arquitectura IPsec

L'anomenada **arquitectura IPsec** (RFC 2401) afegeix serveis de seguretat al protocol IP (versió 4 i versió 6), que poden ser usats pels protocols de nivells superiors (TCP, UDP, ICMP, etc.).

IPsec es basa en l'ús d'una sèrie de protocols segurs, dels quals n'hi ha dos que proporcionen la major part dels serveis:

- El **protocol AH** (*Authentication Header*, RFC 2402) ofereix el servei d'autenticació d'origen dels datagrames IP (incloent la capçalera i les dades dels datagrames).

- El **protocol ESP** (*Encapsulating Security Payload*, RFC 2406) pot oferir el servei de confidencialitat, el d'autenticació d'origen de les dades dels datagrames IP (sense incloure la capçalera), o tots dos alhora.

Opcionalment, cadascun d'aquests dos protocols també pot proporcionar un altre servei, el de protecció contra repetició de datagrames.

Per a l'autenticació i la confidencialitat és necessari utilitzar unes determinades claus, corresponents als algorismes criptogràfics que s'apliquin. Una possibilitat és configurar aquestes claus de forma manual en els nodes IPsec. Normalment, però, es faran servir uns altres protocols per a la **distribució de claus**, com poden ser:

- ISAKMP (*Internet Security Association and Key Management Protocol*, RFC 2408)

- IKE (*Internet Key Exchange*, RFC 2409)

- El protocol d'intercanvi de claus OAKLEY (RFC 2412)

Els agents que intervenen en l'arquitectura IPsec són:

- Els nodes extrems de la comunicació: l'origen i la destinació final dels datagrames.
- Els nodes intermedis que suporten IPsec, anomenats **passarel·les segures**, com per exemple els encaminadors o tallafocs amb IPsec.

Passarel·les segures

Un encaminador IPsec normalment actua com a passarel·la, però pot passar que en alguna comunicació actuï com a node extrem, com per exemple quan se li envien ordres de configuració amb el protocol de gestió SNMP.

El trànsit IPsec també pot passar per nodes intermedis que no suporten IPsec. Aquests nodes són transparents al protocol perquè per a ells els datagrames IPsec són com qualsevol altre datagrama IP.

La relació que s'estableix entre dos nodes que s'envien datagrames IPsec l'un a l'altre s'anomena **associació de seguretat** (SA). Aquests dos nodes poden ser o no els extrems de la comunicació, és a dir, l'origen dels datagrames o la seva destinació final. Per tant, podem distingir dos tipus de SA:

SA

SA és la sigla de *Security Association*.

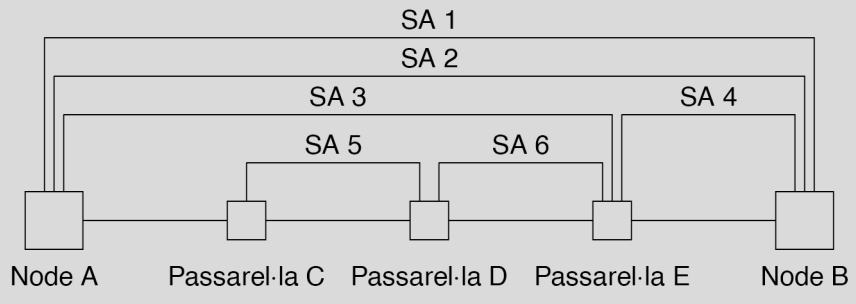
- Les SA extrem a extrem: s'estableixen entre el node que origina els datagrames i el node al qual van destinats.
- Les SA amb una passarel·la segura: almenys un dels nodes és una passarel·la segura (també poden ser-ho tots dos). Per tant, els datagrames vénen d'un altre node i/o van cap a un altre node.

A més, es considera que les SA són unidireccionals, és a dir, si *A* envia datagrames IPsec a *B*, i *B* envia datagrames IPsec a *A*, tenim dues SA, una en cada sentit.

D'altra banda, quan s'estableix una SA entre dos nodes, es fa servir un dels dos protocols bàsics IPsec: o AH o ESP. Si es volen fer servir tots dos alhora, cal establir dues SA, una per cada protocol.

Per tant, pot passar que en una comunicació entre dos nodes extrems intervinguin diverses SA, cadascuna amb els seus nodes d'inici i de final i amb el seu protocol.

Exemple de combinació d'associacions de seguretat



Cada node ha de guardar informació sobre les seves SA, com per exemple els algorismes criptogràfics que utilitza cadascuna, les claus, etc. En la terminologia IPsec, al lloc on es guarda aquesta informació se li diu **base de dades d'associacions de seguretat** o SAD. A cada SA li correspon un nombre anomenat **índex de paràmetres de seguretat** o SPI. Totes les SA que un node tingui establertes amb un altre han de tenir SPI diferents. Per tant, cada SA en què participa un node queda identificada per l'adreça IP de destinació i el seu SPI.

SAD

SAD és la sigla de *Security Association Database*.

SPI

SPI és la sigla de *Security Parameters Index*.

SPD

SPD és la sigla de *Security Policy Database*.

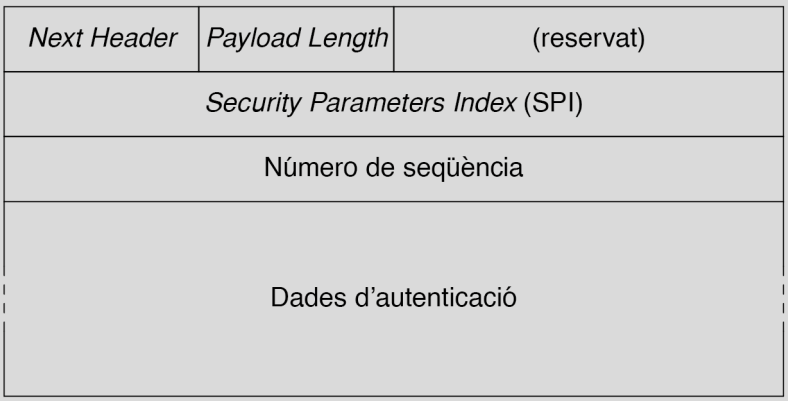
Per a cada datagrama que arriba a un node IPsec, es consulta una **base de dades de polítiques de seguretat** (SPD) on s'especifiquen criteris per determinar quina de les següents 3 accions cal realitzar:

- Aplicar serveis de seguretat IPsec al datagrama, és a dir, processar-lo segons AH i/o ESP.
- Processar-lo com un datagrama IP normal, és a dir, de forma transparent a IPsec.
- Descartar el datagrama.

4.2. El protocol AH

El protocol AH defineix una capçalera que conté la informació necessària per a l'autenticació d'origen d'un datagrama.

Format de la capçalera AH



El camp *Next Header* serveix per indicar a quin protocol corresponen les dades que vénen a continuació de la capçalera AH. El camp *Payload Length* indica la longitud de la capçalera (aquesta informació es necessita perquè l'últim camp és de longitud variable, ja que depèn de l'algorisme d'autenticació).

Camp *Next Header* en AH

Possibles valors del camp *Next Header* són, per exemple, 6 (TCP), 17 (UDP), 50 (si a continuació ve una capçalera ESP), etc.

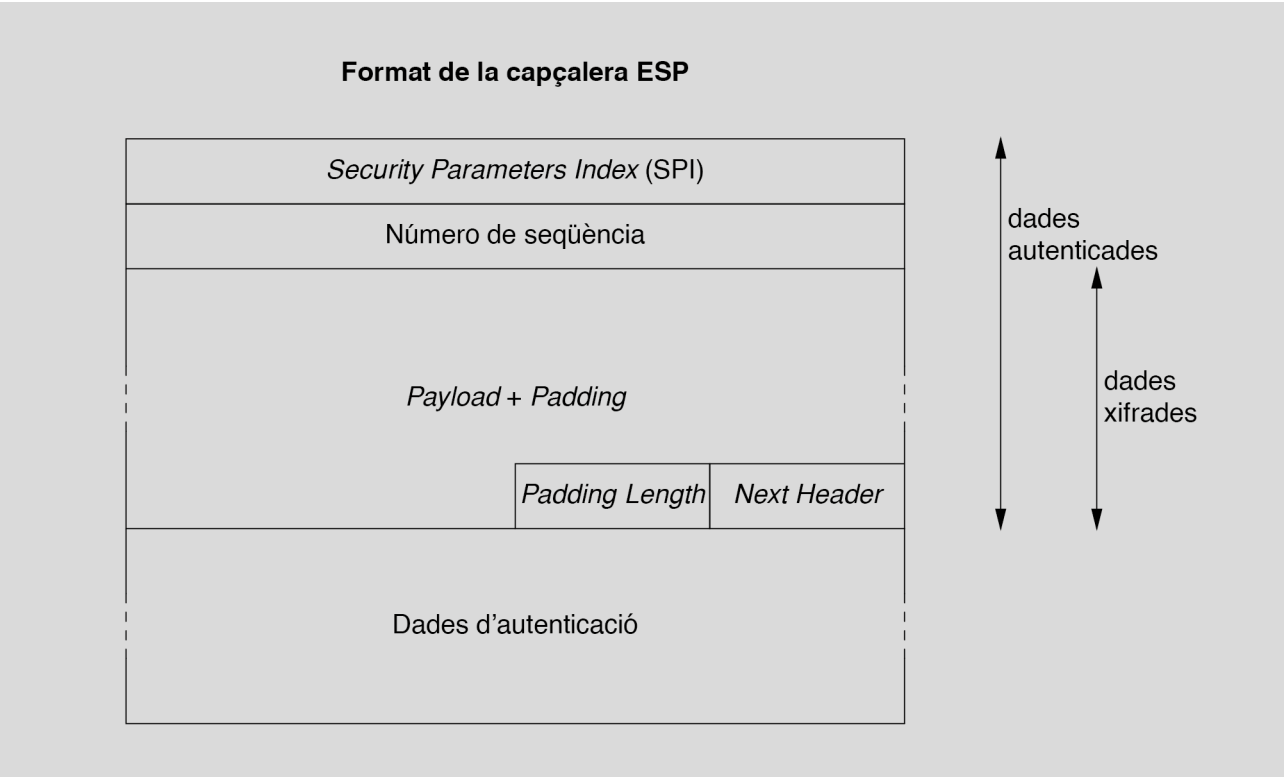
El camp SPI serveix per identificar a quina SA correspon aquesta capçalera AH, i el número de seqüència s'utilitza si es vol proporcionar el servei de protecció contra repetició de datagrames.

Finalment, l'últim camp conté un codi d'autenticació, per exemple un codi MAC, calculat segons l'algorisme que correspongui a aquesta SA. El codi s'obté a partir del datagrama sencer, tret dels camps que poden variar de node a node (com els camps TTL i *Checksum* de la capçalera IP), o els que tenen un valor desconegut *a priori* (com el propi camp de dades d'autenticació de la capçalera AH).

El node que rebi un datagrama amb capçalera AH verificarà el codi d'autenticació, i si és correcte donarà el datagrama per bo i el processarà normalment. Si a més s'utilitza el servei de protecció contra repeticions, cal comprovar que el número de seqüència no sigui repetit: si ho és, es descarta el datagrama.

4.3. El protocol ESP

El protocol ESP defineix una altra capçalera, que de fet inclou dintre seu totes les dades que vinguin a continuació en el datagrama (el que en anglès s'anomena *"payload"*).



Els camps SPI i número de seqüència són anàlegs als de la capçalera AH.

A continuació vénen les dades del datagrama (*payload*), a les quals pot ser necessari afegir bytes addicionals (*padding*) si s'utilitza un algorisme de xifratge de bloc, per fer que el nombre de bytes a xifrar sigui múltiple de la longitud de bloc. El camp *Padding Length* indica exactament el nombre de bytes que s'han afegit (pot ser 0). El camp *Next Header* indica de quin protocol són les dades del datagrama.

Dades xifrades en ESP

Depenent de l'algorisme de xifratge utilitzat, pot ser necessari incloure abans de les dades xifrades paràmetres com el vector d'inicialització, etc.

Depenent del servei o serveis que proporcioni aquesta capçalera ESP, pot ser que les dades estiguin xifrades (incloent el *padding* i els camps *Padding Length* i *Next Header*), que s'hi afegeixi un codi d'autenticació calculat a partir de la capçalera ESP (però no de les capçaleres que hi pugui haver abans), o totes dues coses alhora.

Camp *Next Header* en ESP

Els possibles valors del camp *Next Header* són els mateixos que en la capçalera AH: 6 (TCP), 17 (UDP), etc. Si les dades (*payload*) comencessin amb una capçalera AH, el valor seria 51, però no és habitual aplicar ESP a un datagrama amb AH, és més normal fer-ho a la inversa.

El node que rebi un datagrama amb capçalera ESP haurà de verificar el codi d'autenticació, o desxifrar les dades, o totes dues coses (per aquest ordre, perquè si s'apliquen els dos serveis primer es xifra i després s'autentiquen les dades xifrades). Si a més s'utilitza el servei de protecció contra repeticions, també cal comprovar el número de seqüència. Aquest últim servei, però, només es pot utilitzar quan la capçalera ESP està autenticada.

4.4. Modes d'ús dels protocols IPsec

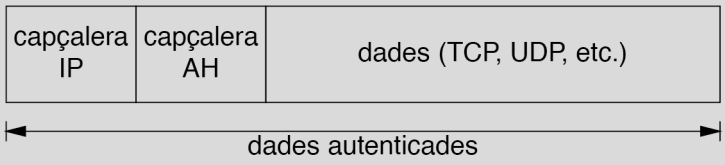
L'arquitectura IPsec defineix dos modes d'ús dels protocols AH i ESP, depenent de com s'incloguin les capçaleres corresponents en un datagrama IP.

- En el **mode transport**, la capçalera AH o ESP s'inclou després de la capçalera IP convencional, com si fos una capçalera d'un protocol de nivell superior, i a continuació van les dades del datagrama (per exemple, un segment TCP amb la seva capçalera corresponent, etc.).

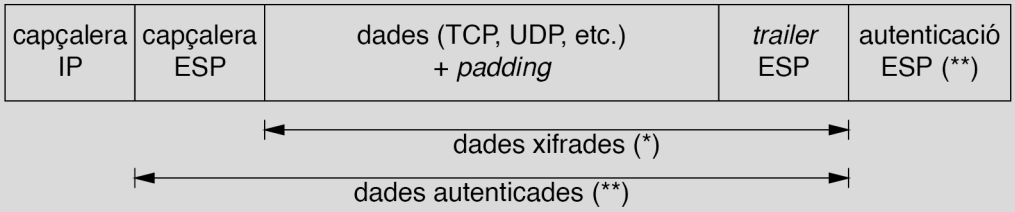
- En el **mode túnel**, el datagrama original s'encapsula tot sencer, amb la seva capçalera i les seves dades, dins d'un altre datagrama. Aquest altre datagrama tindrà una capçalera IP en la qual les adreces d'origen i de destinació seran les dels nodes inici i final de la SA. Per tant, es diu que entre aquests dos nodes hi ha un "túnel" dins del qual viatgen intactes els datagrames originals. A continuació de la capçalera IP del datagrama "extern" hi ha la capçalera AH o ESP.

Les següents figures mostren la disposició de les capçaleres IPsec en cada mode. En aquestes figures, "*trailer* ESP" es refereix als camps *Padding Length* i *Next Header* de la capçalera ESP.

AH en mode transport

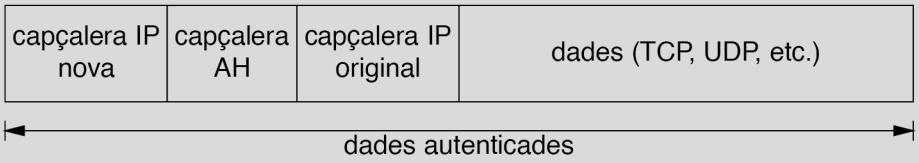


ESP en mode transport

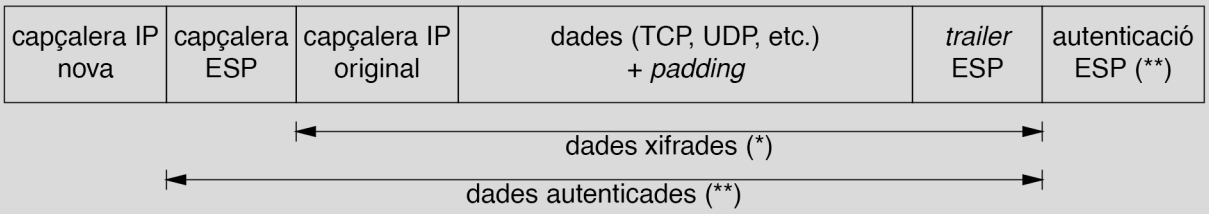


(*: només si s'usa confidencialitat)
(**: només si s'usa autenticació)

AH en mode túnel



ESP en mode túnel



(*: només si s'usa confidencialitat)
(**: només si s'usa autenticació)

El protocol IP preveu que un datagrama es pugui fragmentar, i es pot donar el cas que els fragments d'un mateix datagrama vagin per camins diferents fins a arribar a la seva

destinació final. Això representaria un problema en una SA entre passarel·les segures (o entre un node extrem i una passarel·la segura) si es fes servir el mode transport: per exemple, alguns fragments podrien quedar sense protegir, altres podrien resultar indesxifrables perquè no han passat per la passarel·la que els havia de desxifrar, etc. Per evitar aquestes situacions, en IPsec només es permet el mode transport en les SA extrem a extrem.

El mode túnel no té aquest problema perquè, encara que la SA sigui entre passarel·les, cada datagrama té com a adreça de destinació la del node que hi ha al final del túnel, i tots els fragments han d'acabar arribant a aquest node. Per tant, el mode túnel es pot fer servir en qualsevol SA, tant si és extrem a extrem com si hi intervé una passarel·la segura.

Com hem vist abans, hi pot haver diverses SA en el camí entre l'originador dels datagrames i la destinació final. Això vol dir que les disposicions de capçaleres AH i ESP que mostren les figures anteriors es poden combinar entre elles. Per exemple, hi pot haver un túnel dins d'un altre túnel, o un túnel dins d'una SA en mode transport, etc.

Un altre cas que es pot donar és el de dues SA entre els mateixos nodes d'origen i de destinació, una amb el protocol AH i l'altra amb el protocol ESP. En aquest cas, l'ordre més lògic és aplicar primer ESP amb servei de confidencialitat i després AH, ja que així la protecció que ofereix AH, és a dir, l'autenticació, s'estén a tot el datagrama resultant.

5. Protecció del nivell de transport: SSL/TLS/WTLS

Tal com hem vist a l'apartat anterior, l'ús d'un protocol segur a nivell de xarxa pot requerir l'adaptació de la infraestructura de comunicacions, per exemple canviar els encaminadors IP per altres que entenguin IPsec.

Un mètode alternatiu que no necessita modificacions als equips d'interconnexió és introduir la seguretat en els protocols de transport. La solució més usada actualment és l'ús del protocol SSL o d'altres de basats en SSL.

Aquest grup de protocols comprèn:

- El protocol de transport *Secure Sockets Layer* (SSL), desenvolupat per Netscape Communications a començaments dels anys 90. 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'IETF (*Internet Engineering Task Force*). La versió 1.0 del protocol TLS està publicada al document RFC 2246. És pràcticament equivalent a SSL 3.0 amb algunes petites diferències, per la qual cosa en certs contextos es considera el TLS 1.0 com si fos el protocol "SSL 3.1".

- El protocol *Wireless Transport Layer Security* (WTLS), pertanyent a la família de protocols WAP (*Wireless Application Protocol*) 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 fil, 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.

En aquest apartat parlarem de les característiques comunes a SSL 3.0 i TLS 1.0, amb algun esment particular a les diferències entre ells. La majoria de referències als "protocols SSL/TLS" s'han d'entendre aplicables també a WTLS.

5.1. Característiques del protocol SSL/TLS

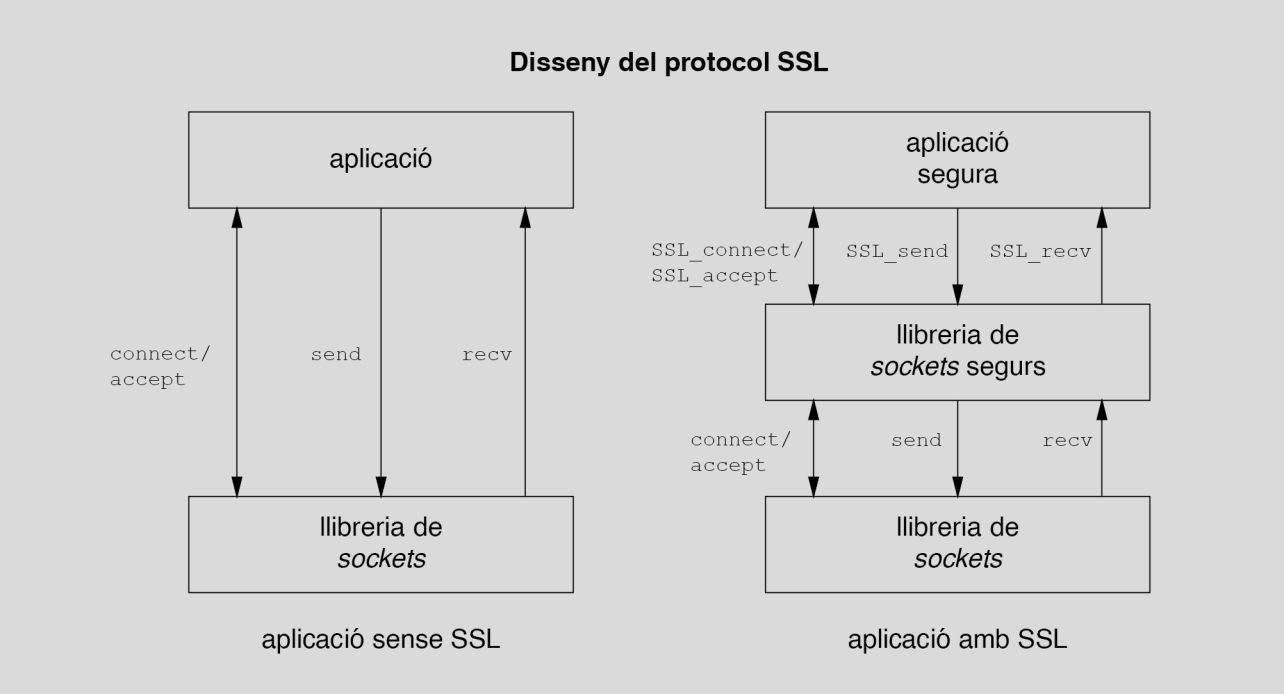
L'objectiu inicial de disseny del protocol SSL va ser protegir les connexions entre clients i servidors web amb el protocol HTTP. Aquesta protecció havia de permetre al client assegurar-se que s'havia connectat al servidor autèntic, i enviar-li dades confidencials, com per exemple un número de targeta 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 el protecció d'aplicació HTTP, sinó que es va optar per introduir-les a nivell de transport. Així podria haver-hi moltes més aplicacions que fessin ús d'aquesta funcionalitat.

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.

A tal fi es va desenvolupar una interfície d'accés als serveis del nivell de transport basada en la interfície estàndard dels *sockets*. 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 a través de les crides dels *sockets* podia fer ús del protocol SSL només canviant aquestes crides. D'aquí ve el nom del protocol.



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). A l'inici de cada sessió, client i servidor es posen d'acord en quines claus utilitzaran per 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 de repte-resposta basat en signatures digitals, el client pot confirmar la identitat del servidor al qual s'ha connectat. Per validar les signatures el client necessita conèixer la clau pública del servidor, i això normalment es fa a través 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 nivell de transport, les mateixes aplicacions utilitzen el seu propi mètode d'autenticació.

Autenticació de client

Un exemple d'autenticació de client a nivell 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 nivell de transport.

Autenticació de missatge. Cada paquet enviat en una connexió SSL/TLS, a més d'anar xifrat, pot incorporar un codi MAC perquè el destinatari comprovi que ningú ha modificat el paquet. Les claus secretes per al càlcul dels codis MAC (una per a cada sentit) també s'acorden de forma segura en el diàleg inicial.

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

Eficiència. Dues de les característiques d'SSL/TLS, la definició de sessions i la compressió de les dades, permeten millorar l'eficiència de la comunicació.

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.

- 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.

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 com-

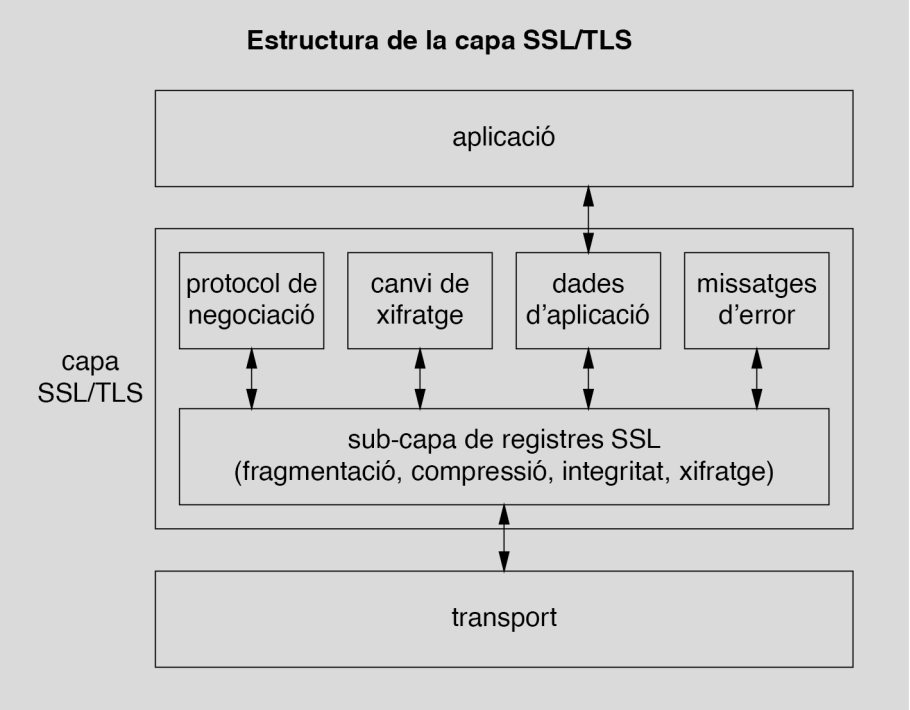
binacions predefinides d'algorismes criptogràfics, però deixen oberta la possibilitat d'afegir-hi nous algorismes si se'n descobreixen d'altres que siguin més eficients o més segurs.

5.2. 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**.

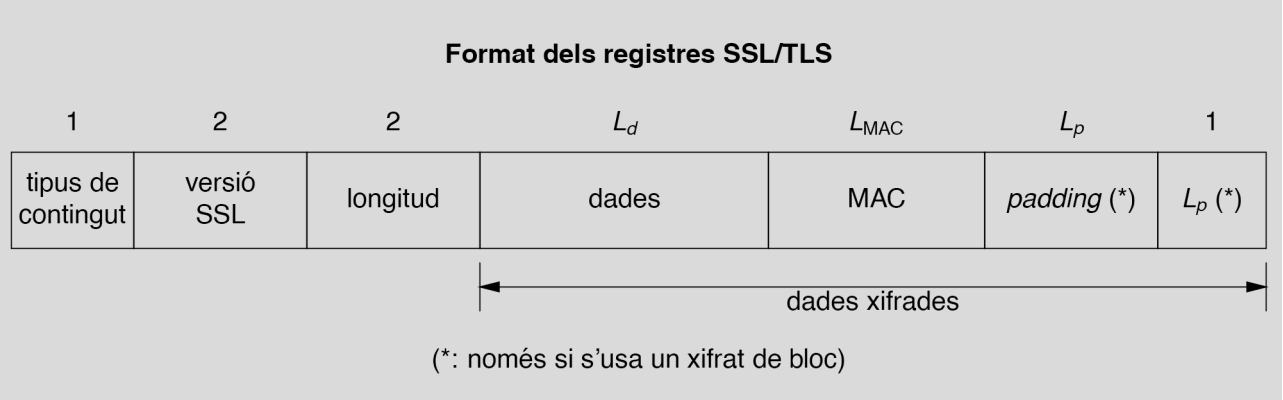
- En la subcapa inferior, aquests missatges són estructurats en **registres** als quals s'aplica, segons correspongui, la compressió, l'autenticació i el xifratge.



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 forma segura a l'inici de la connexió mitjançant el **protocol de negociació SSL/TLS**. A continuació veurem les característiques de cadascun d'aquests dos protocols.

5.2.1. El protocol de registres SSL/TLS

La informació que s'intercanvien client i servidor en una connexió SSL/TLS s'empaqueta en registres, que tenen aquest format:



El significat de cada camp és el següent:

- El primer camp indica quin és el tipus de contingut de les dades, que pot ser:
  - un missatge del protocol de negociació,
  - una notificació de canvi de xifratge,
  - un missatge d'error, o
  - dades d'aplicació.
- El segon camp són dos bytes que indiquen la versió del protocol: si són iguals a 3 i 0 el protocol és SSL 3.0, i si són iguals a 3 i 1 el protocol és TLS 1.0.
- El tercer camp indica la longitud de la resta del registre. Per tant, és igual a la suma de L_d i L_{MAC} i, si les dades estan xifrades amb un algorisme de bloc, $L_p + 1$.
- El quart camp són les dades, comprimides si s'ha acordat algun algorisme de compressió.
- El cinquè camp és el codi d'autenticació (MAC). En el càlcul d'aquest MAC intervenen la clau MAC, un número de seqüència implícit de 64 bits (que s'incrementa a cada registre però no s'inclou en cap camp) i, naturalment, el contingut del registre.

  La longitud d'aquest camp depèn de l'algorisme de MAC que s'hagi acordat utilitzar. Pot ser igual a 0 si es fa servir l'algorisme nul, que és el que s'utilitza al començament de la negociació mentre no se n'ha acordat cap altre.

Dades d'un registre SSL/TLS

Normalment les dades d'un registre corresponen a un missatge de la subcapa superior, però també és possible ajuntar en un mateix registre dos o més missatges, sempre que tots pertanyin al tipus indicat pel primer camp. També pot passar que un missatge es fragmenti en diversos registres, si la seva longitud és superior a un cert màxim (16384 bytes abans de comprimir).

- Si s'ha acordat utilitzar un algorisme de bloc per xifrar les dades, cal afegir bytes addicionals (*padding*) a cada registre per tenir-ne un nombre total que sigui múltiple de la longitud del bloc.

  La tècnica que s'usa per saber quants bytes addicionals hi ha és posar-ne almenys un, i el valor de l'últim byte sempre indica quants altres bytes de *padding* hi ha abans (aquest valor pot ser 0 si només faltava un byte per tenir un bloc sencer).

Padding en SSL i TLS

Una altra diferència entre SSL i TLS està en els bytes de *padding*. En SSL n'hi ha d'haver el mínim necessari, i el seu valor (tret de l'últim byte) és irrellevant. En TLS tots els bytes de *padding* han de tenir el mateix valor que l'últim.

El protocol de registres SSL/TLS s'encarrega de formar cada registre amb els seus camps corresponents, calcular el MAC, i xifrar les dades, el MAC i el *padding* amb els algorismes i les claus que pertoquin.

En la fase de negociació, mentre no s'hagin acordat els algorismes, els registres no es xifren ni s'autentiquen, és a dir, s'apliquen algorismes nuls. Com veurem després, però, tot el procés de negociació queda autenticat *a posteriori*.

5.2.2. 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 i/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 ser transmesos al destinatari. L'estructura d'un missatge de negociació és aquesta:

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 indi-

car al client que està preparat per 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 següent informació:

- La versió del protocol que el client vol fer servir.
- Una cadena de 32 bytes aleatoris.
- 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.

  Algorismes criptogràfics previstos en SSL/TLS

  SSL/TLS considera 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.

- 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).

Bytes aleatoris

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.

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 a la que va enviar el client, o inferior si aquesta no és suportada pel servidor.
- Una altra cadena de 32 bytes aleatoris.
- L'identificador de la sessió actual. Si el client va enviar-ne 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 indicar que la sessió actual mai no podrà ser represa.
- La combinació d'algorismes criptogràfics escollida pel servidor d'entre la llista de les enviades pel client. Si es reprèn una sessió anterior, aquesta combinació ha de ser la mateixa que es va fer servir llavors.

Algorismes de compressió

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

- 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 pot autenticar-se 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 a seguir.

5) Petició de certificat (*Certificate Request*)

En cas que s'hagi de realitzar 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.

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).

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*)

Un cop el servidor ha enviat els seus missatges inicials, el client ja sap com 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*.

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.

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.

Llavors, client i servidor calculen l'anomenat **secret mestre**, que és una altra cadena de 48 bytes. Per fer aquest càlcul, s'apliquen funcions *hash* al secret premestre i a les cadenes aleatòries que es van enviar en els missatges de salutació.

Atacs de versió del protocol

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 evitar aquest atac, als dos primers bytes del secret premestre hi ha d'haver el número de versió que es va enviar en el missatge *Client Hello*.

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 pot autenticar-se 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 indicar que el següent missatge 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 *hash* 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 el missatge *Finished* de l'altra part, si ja l'ha enviat). Normalment el client serà 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*.

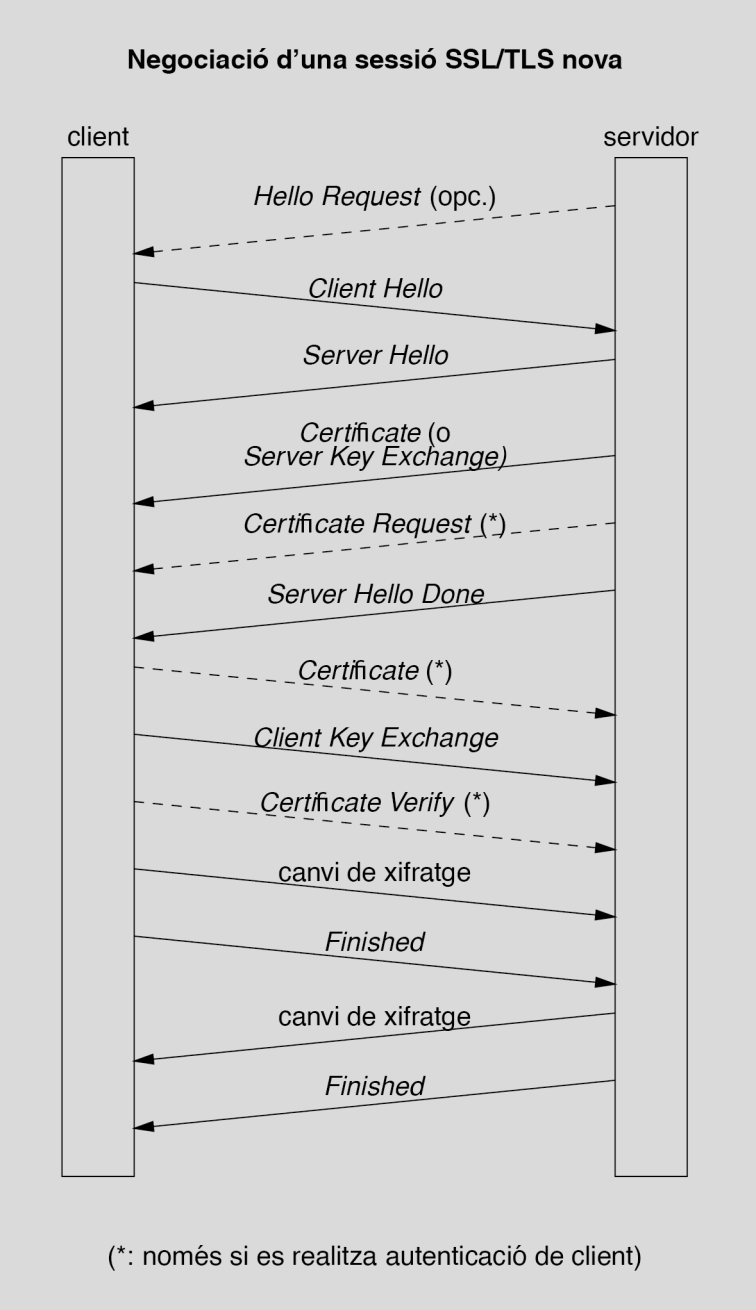
El contingut del missatge *Finished* serveix per verificar que la negociació s'ha dut a terme correctament. Aquest missatge també permet autenticar el servidor davant el client, ja que el primer necessita la seva clau privada per desxifrar el missatge *Client Key Exchange* i obtenir les claus que es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

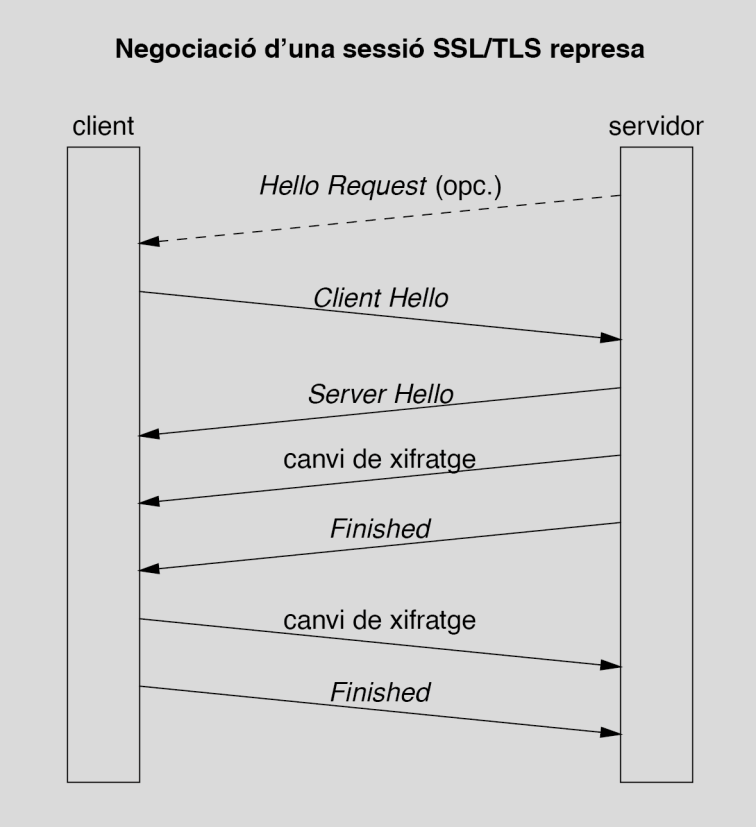
Un cop enviat el missatge *Finished*, es dóna per acabada la negociació, i client i servidor poden començar a enviar les dades d'aplicació fent servir els algorismes i claus acordats.

Verificació d'autenticitat en SSL i TLS

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

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


Negociació d'una sessió SSL/TLS nova
client
servidor
Hello Request (opc.)
Client Hello
Server Hello
Certificate (o
Server Key Exchange)
Certificate Request (*)
Server Hello Done
Certificate (*)
Client Key Exchange
Certificate Verify (*)
canvi de xifratge
Finished
canvi de xifratge
Finished
(*: només si es realitza autenticació de client)



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. 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 indicar la fi normal de la connexió. Per evitar atacs de truncament, si una connexió acaba sense haver enviat aquest avís s'invalidarà el seu identificador de sessió.

5.3. Atacs contra el protocol SSL/TLS

Els protocols SSL/TLS estan dissenyats per resistir els següents atacs:

Lectura dels paquets enviats per client i servidor. Quan les dades s'envien xifrades, un atacant que pugui llegir els paquets, per exemple utilitzant tècniques de *sniffing*, s'enfronta al problema de trencar el xifratge si vol interpretar el seu contingut. Les claus que s'utilitzen per al xifratge s'intercanvien amb mètodes de clau pública, que l'atacant hauria de trencar si vol saber quins són els valors acordats.

Cal advertir, però, que depenent de l'aplicació que l'utilitzi, el protocol SSL/TLS pot ser objecte d'atacs amb text clar conegut. Per exemple, quan s'utilitza juntament amb HTTP per accedir a servidors web amb continguts coneguts.

Si la comunicació és totalment anònima, és a dir sense autenticació de servidor ni client, sí que hi ha la possibilitat de capturar les claus secretes amb un atac anomenat "d'home al mig" (en anglès, *"man-in-the-middle attack"*). En aquest atac l'espia genera les seves pròpies claus públiques i privades, i quan una part envia a l'altra informació sobre la seva clau pública, tant en un sentit com en l'altre, l'atacant la intercepta i la substitueix per l'equivalent amb la clau pública fraudulenta. Com que l'intercanvi és anònim, el receptor no té manera de saber si la clau pública que rep és la de l'emissor autèntic o no.

En canvi, si es realitza l'autenticació de servidor i/o client, és necessari enviar un certificat on hi ha d'haver la clau pública de l'emissor signada per una autoritat de certificació que el receptor reconegui, i per tant no pot ser substituïda per una altra.

Suplantació de servidor o client. Quan es realitza l'autenticació de servidor o client, el certificat digital degudament signat per la CA serveix per verificar la identitat del seu propietari. Un atacant que vulgui fer-se passar pel servidor (o client) autèntic hauria d'obtenir la seva clau privada, o bé la de l'autoritat de certificació que ha emès el certificat per poder-ne generar un altre amb una clau pública diferent i que sembli autèntic.

Alteració dels paquets. Un atacant pot modificar els paquets perquè arribin al destinatari amb un contingut diferent de l'original (si estan xifrats no podrà controlar quin serà el contingut final desxifrat, només sabrà que serà diferent). Si passa això, el receptor detectarà que el paquet ha estat alterat perquè el codi d'autenticació (MAC) quasi amb total seguretat serà incorrecte.

Si l'alteració es realitza en els missatges de negociació quan encara no s'aplica cap codi MAC, amb la finalitat per exemple de forçar l'adopció d'algorismes criptogràfics més febles i vulnerables, aquesta manipulació serà detectada en la verificació dels missatges *Finished*.

Repetició, eliminació o reordenació de paquets. Si l'atacant torna a enviar un paquet correcte que ja havia estat enviat abans, o suprimeix algun paquet fent que no arribi a la seva destinació, o els canvia d'ordre, el receptor ho detectarà perquè els codis MAC no coincidiran amb el valor esperat. Això és així perquè en el càlcul del MAC es fa servir un número de seqüència que es va incrementant a cada paquet.

Tampoc es poden copiar els missatges enviats en un sentit (de client a servidor o de servidor a client) al sentit contrari, perquè en els dos fluxos de la comunicació es fan servir claus de xifratge i de MAC diferents.

Com a consideració final, cal destacar que la fortalesa dels protocols segurs rau no solament en el seu disseny sinó en el de les implementacions. Si una implementació només suporta algorismes criptogràfics febles (amb pocs bits de clau), o genera nombres pseudoaleatoris fàcilment predictibles, o guarda els valors secrets en emmagatzemament (memòria o disc) accessible a atacants, etc., no estarà garantint la seguretat del protocol.

5.4. Aplicacions que fan ús d'SSL/TLS

Com hem vist al començament d'aquest apartat, els protocols SSL/TLS van ser dissenyats per permetre la protecció de qualsevol aplicació basada en un protocol de transport com TCP. Algunes aplicacions que fan ús d'aquesta característica són:

- HTTPS (HTTP sobre SSL/TLS): el protocol més utilitzat actualment per a la navegació web segura.

- NNTPS (NNTP sobre SSL): per a l'accés segur al servei de News.

Aquestes aplicacions amb SSL/TLS funcionen exactament igual que les originals. Les úniques diferències són l'ús de la capa de transport segur que proporciona SSL/TLS i l'assignació de números de port TCP propis: 443 per a HTTPS i 563 per a NNTPS.

En molts altres casos, però, és preferible aprofitar els mecanismes d'extensió previstos en el mateix protocol d'aplicació, si n'hi ha, per negociar l'ús de SSL/TLS, a fi d'evitar la utilització innecessària de nous ports TCP. Així ho fan aplicacions com:

- TELNET, fent servir l'opció d'autenticació (RFC 1416).

- FTP, fent servir les extensions de seguretat (RFC 2228).

- SMTP, fent servir les seves extensions per a SSL/TLS (RFC 2487).

- POP3 i IMAP, també fent servir ordres específiques per a SSL/TLS (RFC 2595).

També hi ha definit un mecanisme per negociar l'ús d'SSL/TLS en HTTP (RFC 2817), com a alternativa a HTTPS.

6. Xarxes privades virtuals (VPN)

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** o VPN entre aquests ordinadors o les xarxes locals on estan situats. En aquest apartat veurem les característiques principals de les xarxes privades virtuals.

VPN

VPN és la sigla de *Virtual Private Network*.

6.1. Definició i tipus de VPN

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'encapsulació 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 definir per sobre d'ella 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:

VPN entre xarxes locals o intranets. 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 interconnectar aquestes xarxes privades i formar una intranet única.

VPN d'accés remot. 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.

VPN extranet. 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 seva protecció s'assoleix mitjançant una VPN extranet.

6.2. Configuracions i protocols utilitzats en VPN

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

- En les VPN entre intranets, la situació més habitual és que a 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 suposarien 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 suposaria un cost encara més gran.

- En les VPN d'accés remot, de vegades anomenades VPDN, un usuari es pot comunicar amb una intranet a través d'un proveïdor d'accés a Internet, fent servir tecnologia convencional com per exemple a través d'un mòdem ADSL. L'ordinador de l'usuari ha de disposar de programari **client VPN** per comunicar-se amb la passarel·la VPN de la intranet i dur a terme l'autenticació necessària, el xifratge, etc.

  VPDN

  VPDN és la sigla de *Virtual Private Dial Network*.

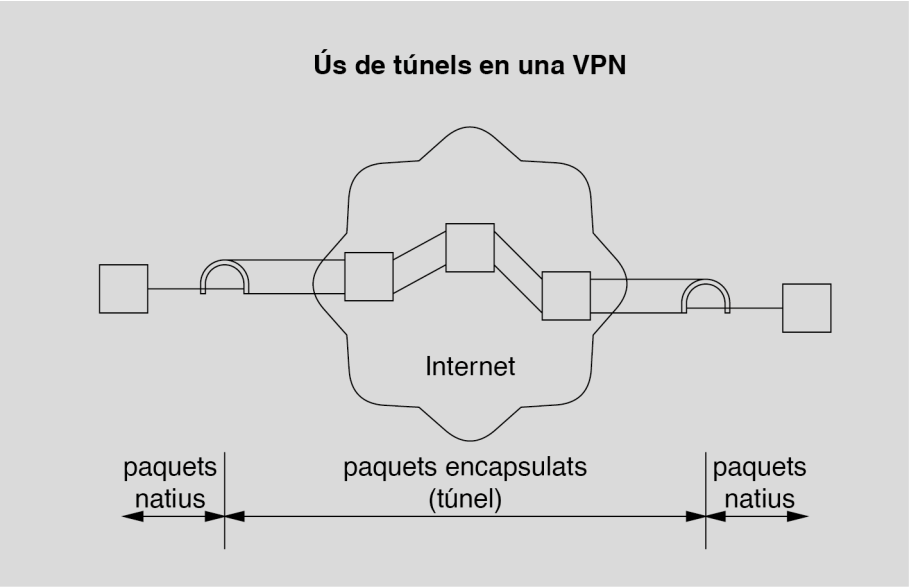
  Així també s'aprofita la infraestructura dels proveïdors d'Internet per a l'accés a la intranet, sense necessitat de trucades a un mòdem de l'empresa, que poden arribar a tenir un cost considerable.

- 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 permetre només l'accés als recursos autoritzats.

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 poder-los fer arribar a la seva destinació a través d'Internet, es poden encapsular en datagrames IP, que dins seu contindran els paquets originals. Quan arriben a la seva destinació, es desencapsulen aquests datagrames per recuperar els paquets amb el format "natiu" del protocol corresponent.



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

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

Túnels a nivell d'enllaç. 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 encapsular PPP (que al seu torn pot encapsular altres protocols de xarxa, com IPX, etc. o possiblement IP) sobre IP:

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

- El protocol L2F (*Layer Two Forwarding*, RFC 2341) és semblant al PPTP però també pot treballar amb SLIP a més de PPP. Per a l'autenticació pot fer servir protocols auxiliars com RADIUS (*Remote Authentication Dial-In User Service*).

- El protocol L2TP (*Layer Two Tunneling Protocol*, RFC 2661) combina les funcionalitats que ofereixen PPTP i L2F.

Túnels a nivell de transport. El protocol SSH (*Secure Shell*), com veurem en el mòdul sobre aplicacions segures, ofereix la possibilitat de redirigir ports TCP sobre un canal segur, que podem considerar com un túnel a nivell de transport. Des d'aquest punt de vista, també es podria considerar una connexió SSL/TLS com un túnel a nivell 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.

Resum

En aquest mòdul 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 les 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**, es 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 pugui fer-la servir per xifrar dades confidencials, amb la seguretat que només qui tingui la clau privada podrà desxifrar-les, 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 xifrar i la pública per 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 obtenir confidencialitat, sinó sempre a través 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 *hash* segura**. La propietat d'aquest tipus de funció és que és molt difícil trobar un missatge que doni el mateix *hash* que un altre.

Per tal de garantir que les claus públiques són autèntiques, i pertanyen a qui se suposa que han de pertànyer, es poden fer servir **certificats de clau pública**, com per exemple els certificats X.509. Quan una **autoritat de certificació** (CA) signa un certificat, està donant fe de l'autenticitat de la clau pública corresponent. Els certificats són un component bàsic de la **infraestructura de clau pública** (PKI), com també ho són les **llistes de revocació de certificats** (CRL).

Les signatures digitals proporcionen el servei d'**autenticació de missatge**. Els anomenats **codis MAC** també proporcionen aquest servei, però fent servir claus secretes compartides en lloc de claus públiques.

Un altre servei d'autenticació és el d'**identificació** o **autenticació d'entitat**. Aquest mecanisme permet comprovar que l'altra part de la comunicació és qui diu ser, i no un impostor. Això es pot aconseguir amb tècniques d'**autenticació feble** basades en contrasenyes o, si és necessari, amb tècniques d'**autenticació forta** basades en **proto-**

cols de repte-resposta, que a diferència de les anteriors no són vulnerables a atacs de repetició.

Quan s'apliquen els mecanismes de confidencialitat i autenticació als protocols de comunicació, és possible fer-ho a diferents nivells. Un exemple de protecció a nivell d'enllaç és el de les comunicacions sense fil, mitjançant els protocols **WEP**, **WPA** i **WPA2**. Per protegir les comunicacions a nivell de xarxa es pot fer servir l'**arquitectura IPsec**, que inclou els protocols **AH**, per autenticar datagrames IP, i **ESP** per xifrar i/o autenticar les dades dels datagrames IP. També hi ha protocols per a l'intercanvi segur de les claus necessàries.

Tota comunicació entre dos nodes de la xarxa mitjançant protocols IPsec pertany a una **associació de seguretat** (SA). Cada SA estableix el protocol a utilitzar, i en quin dels dos modes possibles treballa: el **mode transport**, en el qual la capçalera AH o ESP actua com si fos la capçalera de les dades de nivell superior (transport), o el **mode túnel**, en el qual es construeix un nou datagrama IP que té com a dades el datagrama original convenientment protegit. El mode transport només es pot usar en les SA que vagin d'extrem a extrem, és a dir, des del node que origina els datagrames fins al que els rep.

També hi ha la possibilitat de protegir les comunicacions a nivell de transport. En aquest cas es poden fer servir els protocols **SSL/TLS**, que utilitzen el servei de transport TCP estàndard. En aquests protocols hi ha una **negociació** inicial que permet autenticar el servidor i, si és el cas, el client, mitjançant els seus certificats. El mateix protocol de negociació serveix per establir les claus de sessió que es faran servir en la comunicació posterior, com les claus per al xifratge simètric de les dades o les claus per als codis MAC.

L'ús típic dels protocols SSL/TLS és per 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.

Finalment, 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.

Activitats

1. Visiteu pàgines que contenen llistes d'algorismes criptogràfics i els seus atacs coneguts (per exemple `www.ramkilde.com/bc.html` per als xifratges de bloc, `planeta.terra.com.br/informatica/paulobarreto/hflounge.html` per a les funcions *hash*, etc.), i comproveu quins algorismes es poden considerar actualment segurs i quins no.

2. Visiteu la pàgina `www.rsasecurity.com/rsalabs/challenges/`, a l'apartat *"RSA factoring challenge"*, i comproveu quants bits té l'últim nombre que s'ha aconseguit factoritzar.

3. El projecte EuroPKI pretén crear una infraestructura de clau pública a nivell europeu. Visiteu la seva pàgina web (`www.europki.org`) i examineu el certificat de la seva CA. És una CA arrel? De quants bits és la seva clau pública? Examineu també la llista de certificats emesos per aquesta CA i la seva CRL. Hi ha algun certificat revocat? Quan s'emetrà la pròxima CRL? Si hi ha certificats revocats, podeu esbrinar si tornaran a aparèixer a la pròxima CRL?
Navegueu també per les pàgines web d'altres CA de la jerarquia, com per exemple la de RedIRIS (`www.rediris.es/cert/iris-pca/`) o la de l'Anella Científica del CESCA (`www.cesca.es/comunicacions/scd/`). Alguna d'aquestes CA inclou el subcamp `pathLenConstraint` en el seu certificat?

4. Una implementació *"open source"* dels protocols SSL/TLS prou coneguda és la del projecte OpenSSL. Visiteu la seva pàgina web (`www.openssl.org`) i comproveu quins algorismes criptogràfics suporta l'última versió.

Exercicis d'autoavaluació

1. En el xifratge de bloc en mode ECB, si hi ha un error de transmissió en un bloc de text xifrat, només es veu afectat el bloc corresponent del text desxifrat. En mode CBC, però, l'error es propaga: un error en la transmissió de C_i afecta el desxifratge de M_i i M_{i+1}.
   a) Afectaria l'error algun altre bloc més enllà de M_{i+1}?

   b) Suposeu que hi ha un error en un bit de la versió original (abans de xifrar) de M_i. A quants blocs de text xifrat es propagarà aquest error? Quin serà l'efecte en recepció?

2. Si es produeix un error de transmissió en un bit de text xifrat en mode CFB de 8 bits (longitud de cada unitat de text xifrat C_i igual a 8 bits), fins on es propagarà l'error?

3. Considereu la següent proposta d'algorisme per verificar si, després d'un intercanvi de clau secreta, les dues parts A i B han obtingut el mateix valor de la clau k. Primer, A crea una cadena de bits aleatoris r de la mateixa longitud que la clau, calcula $a = k \oplus r$, i envia aquest valor a B. Llavors B dedueix r calculant $b = a \oplus k$ $(= k \oplus r \oplus k = r)$ i envia aquest valor b a A. Si A veu que el valor rebut b coincideix amb r, sabrà que B té el mateix valor de k, sense que cap dels dos hagi enviat aquest valor en clar. Té algun problema aquesta proposta?

4. L'estàndard PKCS #1 especifica com formatar les dades que es volen xifrar amb una clau pública RSA abans d'aplicar-los l'algorisme de xifratge. Segons la versió 1.5 de l'estàndard, cal crear una seqüència de L bytes (on L és la longitud en bytes del mòdul n):
   - El primer byte és igual a 0.
   - El segon byte indica el tipus de format, i en aquest cas és igual a 2.
   - Els següents bytes (com a mínim, 8) han de tenir valors aleatoris diferents de 0.
   - El següent byte és igual a 0.
   - La resta de bytes (com a màxim, $L-11$) són el missatge que es vol xifrar.

   a)Quina seguretat proporcionen el segon byte i els bytes aleatoris?

   b)Quina utilitat té el byte igual a 0 abans del missatge?

5. Mentre que una clau de xifratge IDEA de 128 bits actualment es considera força segura, una clau RSA de 512 bits es considera poc segura. Per què?

6. Si un certificat X.509 ha deixat de ser vàlid abans de la seva caducitat, la CA que el va emetre el pot incloure en la seva llista de certificats revocats (CRL). Sabent que en la CRL no hi ha el nom (DN) de l'usuari a qui es revoca el certificat, sinó només el número de sèrie del certificat, hi ha alguna manera que un atacant pugui manipular la CRL per fer creure que el certificat que s'està revocant és el d'un altre usuari?

7. La Recomanació X.509 descriu diversos protocols d'autenticació, un dels quals és l'anomenada "autenticació en dos passos", que es pot resumir així:

$$A \rightarrow B: \quad S_A(\{r_A, t_A, B\})$$
$$A \leftarrow B: \quad S_B(\{r_B, t_B, A, r_A\})$$

La mateixa Recomanació X.509 defineix un altre protocol, anomenat "autenticació en tres passos", en què les marques de temps són opcionals i per tant no requereix sincronització entre A i B. Aquest altre protocol en la seva versió original era equivalent al següent intercanvi:

$$A \rightarrow B: \quad S_A(\{r_A, B\})$$
$$A \leftarrow B: \quad S_B(\{r_B, A, r_A\})$$
$$A \rightarrow B: \quad S_A(\{r_B\})$$

Però aquest protocol té un problema potencial que pot ser explotat per un impostor C que es vulgui fer passar per A davant de B. L'impostor, d'una banda, pot repetir un missatge inicial prèviament capturat:

$$C \rightarrow B: \quad S_A(\{r_A, B\})$$
$$C \leftarrow B: \quad S_B(\{r'_B, A, r_A\})$$

i d'altra banda pot fer que A inicií una autenticació amb C:

$$A \rightarrow C: \quad S_A(\{r'_A, C\})$$
$$A \leftarrow C: \quad S_C(\{r'_B, A, r'_A\})$$
$$A \rightarrow C: \quad S_A(\{r'_B\})$$

Llavors, C només ha d'enviar a B aquest últim missatge, que és el que necessita perquè es convenci que està parlant amb A, quan en realitat està parlant amb C:

$$C \rightarrow B: \quad S_A(\{r'_B\})$$

Quina seria una possible solució senzilla per evitar aquest atac? (En la versió actual de la Recomanació X.509 aquest problema ja està solucionat.)

8. La signatura digital d'un missatge es calcula xifrant amb la clau privada del signant el *hash* del missatge. Per què no es xifra directament el missatge a signar?

9. Un possible atac contra les signatures digitals consisteix a fer creure que el signant ha calculat el *hash* amb un altre algorisme (p. ex. MD4 en lloc de MD5), i si aquest algorisme és menys segur que l'original, pot ser que l'atacant sigui capaç d'obtenir un missatge diferent que doni el mateix *hash* amb aquest altre algorisme, de manera que la signatura continuaria sent vàlida. Com es pot evitar aquest atac? (Un dels estàndards de criptografia de clau pública, el PKCS #7, inclou una mesura contra aquest atac.)

10. L'especificació IPsec indica que quan dues associacions de seguretat (SA) en mode transport es combinen per usar AH i ESP en una mateixa comunicació extrem a extrem, només un dels dos possibles ordres és apropiat: aplicar primer el protocol ESP i després el protocol AH. Per què?

11. Una organització té instal·lada una xarxa amb adreces IP privades, i connectada a Internet mitjançant un *router* NAT (*Network Address Translator*), que tradueix les adreces privades en una o més adreces públiques (assignades per un registrador oficial). Si es vol fer servir IPsec per connectar-se a servidors externs, sense fer cap canvi en la xarxa, quines combinacions de protocols (AH, ESP) i modes d'operació (transport, túnel) seran apropiades i quines no, i per què?

12. Com pot contrarestar el protocol HTTPS (HTTP sobre SSL/TLS) les següents amenaces a la seguretat del servei WWW?

a) Atac criptogràfic de força bruta: cerca exhaustiva en l'espai de claus per desxifrar els paquets xifrats simètricament.

b) Atac de text clar conegut, tenint en compte que molts missatges HTTP, com per exemple les peticions "`GET`", contenen text predictible.

c) Atac de repetició: reenviar missatges de negociació SSL/TLS capturats prèviament.

d) Atac "d'home al mig" (*"man-in-the-middle"*): interceptar una negociació SSL/TLS, reenviant paquets modificats al client com si fossin del servidor autèntic, i viceversa.

e)Obtenció de contrasenyes: captura de *passwords* HTTP o d'altres aplicacions.

f) Falsificació IP (*"IP spoofing"*): generar paquets amb adreces IP falses.

g) "Segrest" IP (*"IP hijacking"*): interrompre una connexió autenticada entre dos nodes i continuar-la fent-se passar per un d'ells.

h) "Inundació" de paquets SYN (*"SYN flooding"*): enviar paquets TCP amb el *flag* SYN per iniciar una connexió però no respondre al missatge final per acabar d'establir-la, amb la intenció de saturar el servidor amb connexions TCP mig obertes.

13. Un client C vol establir una connexió SSL/TLS amb un servidor S que té una clau pública K. Durant la fase inicial de negociació, un atacant A intercepta els missatges SSL/TLS i respon a C en nom de S fent veure que la clau pública del servidor és K' en lloc de K, i seguint tots els passos de la negociació utilitzant aquesta clau K'. Com pot C detectar aquest frau?

14. En el protocol SSL/TLS, li és possible al receptor reordenar registres SSL/TLS que li arribin desordenats? Per què?

Solucionari

1.

a) No.

b) L'error en M_i farà que tots els blocs a partir de C_i siguin diferents dels que s'haurien de transmetre. En recepció, però, tots els blocs a partir de M_{i+1} es recuperaran correctament (el bloc M_i es recuperarà amb el mateix error d'origen).

2. Es recuperaran incorrectament els $N+1$ bytes de text en clar a partir de l'error, on N és $L/8$ (L = longitud de bloc de l'algorisme de xifratge). Per exemple, en DES ($L = 64$), $N + 1 = 9$ bytes.

3. Un atacant que tingui accés a la comunicació veurà els valors $a = k \oplus r$ i $b = r$. Llavors només cal calcular $a \oplus b$ per obtenir k.

4.

a) El segon byte indica com interpretar les dades un cop desxifrades, i els bytes aleatoris asseguren que el nombre M a xifrar serà gran ($M > 2^{8 \cdot L-24}$, i amb el segon byte igual a 2, $M > 2^{8 \cdot L-17}$). Si M fos massa petit, el desxifratge podria ser trivial (especialment si $M^e < n$). A més, els bytes aleatoris dificulten els atacs per força bruta xifrant amb la mateixa clau pública: cal provar almenys 2^{64} combinacions per cada possible valor del missatge.

b) El byte igual a 0 serveix per saber on acaben els bytes aleatoris (que han de ser diferents de 0) i on comença el missatge.

5. Perquè en els algorismes simètrics qualsevol combinació de bits és una clau vàlida, i per tant l'esforç per trencar una clau de 128 bits ha de ser de l'ordre de 2^{128} operacions. En canvi, les claus públiques han de complir unes propietats (i per tant no qualsevol combinació de 512 bits és una clau RSA vàlida), i els mètodes per trencar claus públiques aprofiten aquestes propietats.

6. Els números de sèrie identifiquen de manera única els certificats que emet una CA, i la manera de fer creure que s'està revocant un altre certificat és modificant la CRL. Com que la CRL està signada per la CA que l'emet, l'atacant hauria de ser capaç de falsificar la signatura de la CA.

7. Una solució senzilla és que el missatge final de l'autenticació en tres passos inclogui l'identificador del destinatari:

$$A \rightarrow B: \quad S_A(\{r_B, B\})$$

(Aquesta és la solució que hi ha a la versió actual de la Recomanació X.509.)

8. Perquè xifrar amb clau privada un missatge de longitud arbitrària pot ser molt costós, ja que la criptografia de clau pública requereix molts més càlculs que la de criptografia simètrica. Per això es xifra només el *hash*, que és d'una longitud curta.

9. Una possible solució (la que fa servir l'estàndard PKCS #7) és que les dades que es xifren amb la clau privada no siguin únicament el *hash* del missatge, sinó una concatenació d'aquest *hash* amb un identificador de l'algorisme de *hash* utilitzat.

10. Perquè amb ESP es poden xifrar les dades dels paquets IP, i després amb AH es poden autenticar els paquets sencers, inclosa la capçalera. Si es fes a la inversa, s'estaria autenticant el paquet intern amb AH, però en el paquet ESP extern no s'estarien protegint les capçaleres (amb confidencialitat i/o autenticació).

11. Com que el NAT modifica les capçaleres IP (concretament les adreces d'origen o destinació), la combinació més apropiada és fer servir ESP en mode túnel, de manera que el *router* no modifiqui el paquet encapsulat. No es pot fer servir AH perquè autentica tot el paquet, incloses les capçaleres. El mode transport té el problema que els *routers* NAT també han de modificar el *checksum* de les capçaleres TCP i UDP, ja que en el seu càlcul intervenen les adreces de la capçalera IP. (Alternativament, es pot fer servir IPsec a la part externa de la xarxa, després del NAT, o, si les implementacions ho suporten, deshabilitar els *checksums* TCP i UDP).

12.

a) La protecció és la que doni l'algorisme de xifratge simètric escollit, i dependrà de la longitud de la clau de sessió.

b) Els atacs de text clar conegut són possibles, i poden reduir en part l'esforç necessari per al desxifratge per força bruta.

c) Dins d'una mateixa sessió es detectaria la repetició de les dades, perquè el MAC seria incorrecte. Encara que es fes servir un xifratge de bloc en mode ECB, el MAC continuaria sent invàlid perquè es calcula a partir d'un número de seqüència implícit. Tampoc es poden copiar dades en sentit contrari perquè es fan servir claus de xifratge i de MAC diferents en cada sentit.

d) Si l'intercanvi de claus és anònim, l'atac tindria èxit. Si s'utilitza autenticació (de servidor i/o client), l'atacant hauria de trencar l'algorisme d'autenticació.

e) Aquest problema en general es redueix a un atac al xifratge simètric de la comunicació.

f) Si no hi ha autenticació, aquest atac tindria èxit. Si hi ha autenticació, els certificats (que poden incloure l'adreça IP o nom DNS de servidor i/o client) serveixen per evitar aquest atac.

g) Sense conèixer les claus de sessió, l'atacant no pot continuar la comunicació.

h) El protocol SSL/TLS treballa sobre TCP, i no té accés als mecanismes d'establiment de la connexió TCP. Per tant, SSL/TLS no protegeix contra aquest atac.

13. Si la clau K està autenticada mitjançant un certificat, C descobrirà que K' és una clau falsa perquè no hi haurà un certificat vàlid per a aquesta clau.

14. Els registres SSL/TLS no haurien d'arribar desordenats perquè el protocol SSL/TLS s'usa sobre TCP, que garanteix la seqüència correcta de les dades. Si un atacant intencionadament desordenés els registres, el receptor no sabria en principi com reordenar-los perquè en el registre no hi ha cap número de seqüència explícit (però el MAC permetria detectar el canvi de seqüència).

Glossari

AH: Vegeu *Authentication Header*.

Associació de seguretat (SA): Relació entre un node origen i un node de destinació que fan servir un dels protocols IPsec (AH o ESP) per enviar datagrames IP protegits.

Atac: Acció realitzada per una tercera part, diferent de l'emissor i el receptor d'informació protegida, per intentar contrarestar aquesta protecció.

Atac d'aniversari: Atac contra les funcions *hash*, consistent en trobar dos missatges que donin el mateix resum, en lloc de trobar un missatge que doni el mateix resum que un altre determinat, que requereix moltes més operacions.

Atac de diccionari: Atac contra els mètodes d'autenticació d'entitat basats en contrasenyes, consistent a provar les paraules d'un diccionari fins a trobar la correcta.

Atac de força bruta: Atac contra les funcions criptogràfiques, consistent a provar tots els possibles valors de la clau fins a trobar el correcte.

Atac d'home al mig: Atac contra l'autenticació en els protocols de comunicació segurs, en què l'atacant intercepta els missatges d'autenticació i els substitueix per altres amb les claus públiques canviades, de manera que es pot produir una suplantació si no es comprova l'autenticitat d'aquestes claus.

Autenticació: Protecció de la informació contra falsificacions.

Autenticació d'entitat: Servei de seguretat que permet confirmar que un participant en una comunicació és autèntic, i no es tracta d'un impostor que està intentant suplantar-lo.

Autenticació de missatge: Servei de seguretat que permet confirmar que l'originador d'un missatge és autèntic, i que el missatge no ha estat creat o modificat per un falsificador.

Autenticació d'origen de dades: Nom amb què es coneix de vegades l'autenticació de missatge.

***Authentication Header* (AH):** Protocol de l'arquitectura IPsec que proporciona autenticació dels datagrames IP.

Autoritat de certificació (CA): Entitat que emet certificats de clau pública que serveixen perquè els usuaris que confien en aquesta autoritat es convencin de l'autenticitat de les claus públiques.

Autoritat de certificació (CA) arrel: CA que no en té cap altra de superior que certifiqui l'autenticitat de la seva clau pública, i que per tant té un certificat signat per ella mateixa.

CA: Vegeu *Autoritat de certificació*.

Cadena de certificats: Llista de certificats, cadascun dels quals permet verificar l'autenticitat de la clau pública de la CA que ha emès l'anterior, fins a arribar al certificat d'una CA arrel.

Certificat de clau pública: Estructura de dades que conté un nom d'usuari i una clau pública, i que està signada digitalment per una autoritat que dóna fe de l'autenticitat d'aquesta clau pública.

Clau: Paràmetre d'un algorisme de xifratge o de desxifratge que permet definir transformacions criptogràfiques diferents sense necessitat de canviar l'algorisme.

Clau de sessió: Clau simètrica generada *ad hoc* per protegir un determinat intercanvi d'informació i que és coneguda per les dues parts utilitzant criptografia de clau pública, perquè no pugui ser descoberta per un atacant.

Clau privada: Clau que permet realitzar la transformació criptogràfica inversa a la que s'obté amb una clau pública i que és computacionalment inviable obtenir a partir d'aquesta última.

Clau pública: Clau que permet realitzar la transformació criptogràfica inversa a la que s'obté amb una clau privada i que es pot obtenir fàcilment a partir d'aquesta última.

Clau simètrica: Clau que permet realitzar tant una transformació criptogràfica com la transformació inversa, és a dir, xifratge i desxifratge.

Codi d'autenticació de missatge (MAC): Valor calculat a partir d'un text amb una clau secreta i que pot ser utilitzat per qui conegui la clau per comprovar l'autenticitat del missatge.

Confidencialitat: Protecció de la informació contra lectura per part de tercers no autoritzats.

Contrasenya: Paraula (“*password*”) o cadena de caràcters secreta, de longitud relativament curta, usada per una entitat per autenticar-se.

Criptoanàlisi: Estudi de les tècniques matemàtiques per anul·lar la protecció que proporciona la criptografia.

Criptografia: Estudi de les tècniques matemàtiques per protegir la informació, de manera que no pugui ser llegida o modificada per parts no autoritzades.

Criptologia: Disciplina que engloba la criptografia i la criptoanàlisi.

CRL: Vegeu *Llista de revocació de certificats*.

Desxifratge: Transformació inversa al xifratge per obtenir el text en clar a partir del text xifrat i la clau de desxifratge.

***Digest*:** Nom que es dóna de vegades a un resum o *hash*.

***Encapsulating Security Payload* (ESP):** Protocol de l'arquitectura IPsec que proporciona autenticació i/o confidencialitat de les dades dels datagrames IP.

ESP: Vegeu *Encapsulating Security Payload*.

Extranet: Xarxa privada d'una organització en què una part dels seus recursos són accessibles a determinats usuaris externs a aquesta organització.

***Hash*:** Cadena de bits, de longitud predeterminada, que s'obté a partir d'una seqüència de bits de longitud arbitrària, com a “resum” d'aquesta seqüència.

Índex de paràmetres de seguretat (SPI): Nombre que, juntament amb l'adreça IP del node de destinació, permet a un node origen identificar una associació de seguretat IPsec.

Infraestructura de clau pública (PKI): Conjunt d'estructures de dades, procediments i agents que permeten garantir l'autenticitat de les claus públiques d'una comunitat d'usuaris.

Intranet: Xarxa privada corporativa d'una organització amb accés restringit als usuaris que pertanyen a aquesta organització.

IPsec: Conjunt de protocols a nivell de xarxa (AH, ESP, etc.) que afegeixen seguretat al protocol IP.

Llista de revocació de certificats (CRL): Llista de certificats que han deixat de ser vàlids abans de la seva data de caducitat, emesa i signada per la mateixa CA que va emetre aquests certificats.

MAC: Vegeu *Codi d'autenticació de missatge*.

No repudi: Protecció de la informació contra denegació d'autoria per part del seu originador.

***Padding*:** Dades addicionals que pot ser necessari afegir a un text en clar abans d'aplicar-li un algorisme de xifratge de bloc per tal que la seva longitud sigui múltipla de la longitud de bloc.

***Passphrase*:** Cadena de caràcters secreta, de longitud generalment més llarga que una contrasenya, usada per una entitat per autenticar-se.

***Password*:** Vegeu *Contrasenya*.

PKI: Vegeu *Infraestructura de clau pública*.

Repte-resposta: Tipus de mètode d'autenticació d'entitat basat en un valor secret que l'entitat a autenticar ha de fer servir per calcular una resposta vàlida a un repte que li envia el verificador.

Resum: Vegeu *Hash*.

SA: Vegeu *Associació de seguretat*.

Sal: Conjunt de bits aleatoris que es generen *ad hoc* per modificar una clau de xifratge i que permeten que un mateix text resulti en textos xifrats diferents encara que es xifri amb la mateixa clau.

***Secure Sockets Layer* (SSL):** Protocol per protegir les comunicacions a nivell de transport que ofereix uns serveis de comunicació segura anàlegs als que ofereix la interfície dels *sockets*.

Seguretat computacional: Seguretat que proporciona una tècnica criptogràfica la criptoanàlisi de la qual requeriria una quantitat de recursos computacionals molt més gran del que està a l'abast de ningú.

Seguretat incondicional: Seguretat que proporciona una tècnica criptogràfica que no permet obtenir cap informació sobre el text en clar, independentment de la quantitat de recursos disponibles per a la criptoanàlisi.

Signatura digital: Valor calculat a partir d'un text amb una clau privada i que pot ser comprovat amb la corresponent clau pública, la qual cosa permet confirmar que només el pot haver generat el posseïdor de la clau privada.

SPI: Vegeu *Índex de paràmetres de seguretat*.

SSL: Vegeu *Secure Sockets Layer*.

Text en clar: Informació directament intel·ligible.

Text xifrat: Resultat d'aplicar un xifratge a un text en clar.

TLS: Vegeu *Transport Layer Security*.

***Transport Layer Security* (TLS):** Versió del protocol SSL estanderditzada per l'IETF (*Internet Engineering Task Force*).

Túnel: Associació entre dos nodes d'una xarxa per intercanviar-se paquets d'un protocol determinat, possiblement amb origen i destinació final en altres nodes, encapsulats en paquets del protocol de comunicació que fa servir la xarxa (típicament, la xarxa és Internet i el protocol d'encapsulació és IP).

VPN: Vegeu *Xarxa privada virtual*.

***Wireless Transport Layer Security* (WTLS):** Versió del protocol TLS adaptada a les comunicacions sense fils en un entorn WAP (*Wireless Application Protocol*).

WTLS: Vegeu *Wireless Transport Layer Security*.

Xarxa privada virtual (VPN): Xarxa lògica (virtual) definida sobre una xarxa pública, com per exemple Internet, i que funciona, mitjançant túnels, com si fos una xarxa privada dedicada.

Xifratge: Transformació d'un text en clar, mitjançant un algorisme que té com a paràmetre una clau, en un text xifrat inintel·ligible per a qui no conegui la clau de desxifratge.

Xifratge de bloc: Transformació criptogràfica en què el text en clar es divideix en blocs i s'aplica un algorisme de xifratge a cadascun d'aquests blocs.

Xifratge de flux: Transformació criptogràfica en què el text en clar es combina amb una seqüència pseudoaleatòria obtinguda a partir de la clau.

Bibliografia

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

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

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