OpenSSL

OpenSSL (https://www.openssl.org) és una llibreria lliure que ofereix una "API" criptogràfica de propòsit general als desenvolupadors d'aplicacions. És a dir, els permet incloure un conjunt de funcions criptogràfiques (ja provades i confiables) dins del codi font dels seus propis programes i d'aquesta manera no haver-se de preocupar dels detalls més tècnics sobre la seguretat de la informació, en delegar aquesta tasca en les funcions oferides per l'OpenSSL.

També és una utilitat de línia de comandes que permet implementar directament aquesta API des del terminal, interactivament. En aquest sentit, amb OpenSSL podem realitzar totes les tasques típiques de la criptografia (xifrat/desxifrat -tant simètric com asimètric-, creació/verificació de signatures, etc) així com de les relacionades amb la infraestructura PKI (creació/revocació de certificats, creació de CAs, etc) i altres funcionalitats extra, ja estiguin relacionades amb els protocols i algoritmes pròpiament TLS (estudi del "handshake") o més generalistes ("hashing", codificacions, etc).

> NOTA: Una eina alternativa a OpenSSL és **GnuTLS** (http://www.gnutls.org), la qual inclou igualment una llibreria i una utilitat de terminal (un petit tutorial d'aquesta es pot consultar a https://raysnotebook.info/computing/crypto-tls.html) . Altres eines alternatives també són **LibreSSL** (http://www.libressl.org), **BoringSSL** (https://boringssl.googlesource.com) o **WolfSSL** (https://www.wolfssl.com; aquesta darrera especialitzada en funcionar sobre dispositius embeguts de poca capacitat de càlcul), entre d'altres. D'altra banda, també existeix la llibreria particular **NSS** (https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS; la qual només és emprada per Firefox i obvia l'existència de les altres llibreries que puguin estar instal.lades al sistema). Totes elles són lliures i multiplataforma.

Un dels àmbits on l'ús de la llibreria OpenSSL és més visible és a l'hora d'implementar servidors segurs (normalment de tipus web) que xifrin i mantinguin la integritat de la comunicació amb els seus clients (i aconseguir així que aquest sigui segur pels seus usuaris, de manera que, per exemple, les seves visites no puguin ser "esnifades" per hackers, o que tinguin la garantia de no patir un atac "man-in-the-middle", etc). Això, a la pràctica, vol dir implementar servidors que facin servir el protocol TLS (el qual, recordem, "se situa per sobre" del protocol TCP -el nivell de transport al model TCP/IP- i "per sota" del nivell d'aplicació que correspongui) de forma que un servidor HTTP (per exemple) passi a ser HTTPS (HTTP + TLS).

> NOTA: Tal com de seguida veurem en aquest document, per poder aconseguir el descrit al paràgraf anterior, una primera passa bàsica és crear un certificat (és a dir, una clau pública certificada per una autoritat) per després associar-lo a un servidor (normalment de tipus web però no té perquè) de la forma que aquest disposi a la seva configuració particular. Els certificats (creats mitjançant OpenSSL o alguna eina similar) poden ser directament autosignats o bé signats per una CA externa a partir d'un CSR generat per nosaltres. De fet, amb OpenSSL fins i tot podríem implementar una CA pròpia per signar de forma centralitzada i autònoma els certificats dels diferents servidors que eventualment poguem gestionar.

> NOTA: Existeixen també diversos “wrappers” que faciliten la creació de certificats sense haver d'utilitzar les comandes que s'expliquen tot seguit. Exemples són els programes que vénen amb **OpenVPN** (https://github.com/OpenVPN/easy-rsa) , servidors de correu, servidor Apache, etc. També són dignes de menció les següents eines similars de consola, encara que no es basin en OpenSSL: **Cfssl** (https://github.com/cloudflare/cfssl), Lemur (https://github.com/Netflix/lemur) o la comanda *keytool* que proporciona el paquet "openjdk-11-jre-headless"

Estructura de les comandes *openssl*

La comanda *openssl* pot realitzar moltes tasques diferents, i és per això que incorpora un seguit de subcomandes que serveixen per classificar aquestes tasques segons els seu àmbit. Per veure totes aquestes subcomandes es pot executar *openssl list -commands* Per veure els paràmetres que admet una subcomanda concreta, es pot executar *openssl help subcomanda* Però moltes vegades això no és suficient per saber totes les opcions que tenim disponibles perquè algunes subcomandes incorporen al seu torn vàries sub-subcomandes. Concretament *openssl list -digest-commands* mostra les sub-subcomandes relacionades amb la subcomanda *dgst* i *openssl list -cipher-commands* mostra les sub-subcomandes relacionades amb la subcomanda *enc* , entre d'altres.

> NOTA: Altres comandes *openssl list* interessants són *openssl list -cipher-algorithms*, *openssl list -digest-algorithms* o *openssl list -public-key-algorithms* Cal tenir en compte, en aquest sentit, que segons la versió que tinguem instal·lada de la suite OpenSSL (la qual es mostra a la sortida de la comanda *openssl version*) tindrem accés a l'ús de més o menys algoritmes.

En qualsevol cas, tota la informació es pot consultar a les pàgines del manual d'OpenSSL, les quals, a més de la pàgina d'entrada *man openssl*, estan separades segons subcomandes, així: *man openssl-dgst, man openssl-enc*, etc

Respecte les subcomandes que ens seran útils, podem destacar les següents...:

openssl enc ... : Tasques relacionades amb operacions de xifrat/desxifrat simètric
openssl genpkey ... : Tasques relacionades amb la creació de claus privades
openssl pkeyutl ... : Tasques relacionades amb la inspecció i manipulació de claus públiques
openssl pkey ... : Tasques relacionades amb la inspecció i formatatge de claus asimètriques clàssiques
openssl ec ... : Tasques relacionades amb la inspecció i formatatge de claus de corba el·líptica
openssl dgst ... : Tasques relacionades amb operacions de "hashing" i HMACs
openssl passwd ... : Tasques relacionades amb operacions de "hashing" amb sals (per contrasenyes)
openssl rand ... : Tasques relacionades amb la generació de números aleatoris

...i si ens centrem en la gestió d'una infraestructura PKI (és a dir, en tot allò relacionat amb certificats, autoritats de certificació, CSRs, etc), imprescindible per l'ús de protocols tan importants com TLS, podem destacar també...:

openssl req ... : Tasques relacionades amb la creació i manteniment de CSRs
openssl x509 ... : Tasques relacionades amb la creació i manteniment de certificats x509
openssl verify ... : Tasques relacionades amb la verificació de certificats
openssl ocsp ... : Tasques relacionades amb la revocació de certificats (via OCSP).
openssl ca ... : Tasques relacionades amb la creació i manteniment d'una CA
openssl ciphers ... : Llista el conjunt de ciphersuites TLS existents, suportades, etc

Tasques habituals

A continuació s'indiquen alguns exemples pràctics de les subcomandes anteriors, a mode de "xuleta":

*Xifrar el contingut d'un fitxer amb un algoritme simètric:

openssl enc -e -XXX [-in /ruta/fitxer] [-out /ruta/fitxer.enc]

NOTA: Si no s'especifica cap fitxer d'entrada, l'entrada serà *stdin* (i aquesta s'acabarà pulsant CTRL+D).
NOTA: Si no s'especifica cap fitxer de sortida, la sortida serà *stdout*
NOTA: El valor "-XXX" representa el nom de l'algoritme simètric escollit (precedit per un guió). Aquest nom pot ser un de la llista mostrada per *openssl enc -list* , com per exemple "**aes-256-cbc**"
NOTA: A la pàgina man openssl enc s'explica per què tècnicament la comanda openssl (no així la llibreria subjacent) no suporta els mode d'operació AE/AEAD com ara el GCM o el CCM
NOTA: Existeixen "àlies" de les comandes *openssl enc -e -XXX ...* amb la forma *openssl XXX -e ...*

Aquesta comanda demana una "passphrase" interactivament, la qual pot ser curta i senzilla si es vol perquè, a partir d'ella, es generarà automàtica i internament la clau real utilitzada pel xifratge (la qual, aquesta sí, serà complexa perquè tindrà la longitud de bits corresponent, 128 bits, 256 bits, etc). En relació amb això (i per evitar el "warning" que es mostra en executar la comanda anterior), a la comanda anterior caldrà afegir, a més, o bé el paràmetre *-pbkdf2* o bé el paràmetre *-iter nº*. El primer indica que es farà servir l'algoritme PBKDF2 (un dels possibles algoritmes de tipus "derivació de clau", o KDF, de "Key Derivation Function") per generar la clau real a partir de la "passphrase" i que es repetirà iterativament el seu ús un nombre fixe de 10000 repeticions per fer el resultat final més segur (aquest valor ve indicat dins del codi font de l'arxiu "enc.c" d'OpenSSL); el segon indica que també es farà servir l'algoritme PBKDF2 però ens permet indicar el nombre d'iteracions desitjades que volem que "pateixi" la "passphrase" per derivar la clau d'encriptació real. Com major sigui el valor indicat en *-iter* més segura serà la clau a atacs de força bruta però més costosa serà computacionalment parlant.

NOTA: L'algoritme PBKDF2 necessita usar una "sal" (és a dir, un valor aleatori, tot i que no cal que sigui secret) per a què les claus generades a partir de les passphrases introduïdes no siguin sempre les mateixes si la passphrase és la mateixa. Aquesta sal la genera automàticament la comanda *openssl enc*, així que nosaltres no ens haurem de preocupar en aquest sentit, però és bo de totes formes saber per què és necessària aquesta "sal" internament. La idea és que la clau criptogràfica es genera sempre a partir de la passphrase introduïda més una sal, que serà un valor diferent cada vegada, aconseguint així que la clau també sigui diferent cada vegada, tot i que la passphrase sigui la mateixa; amb això s'evita que un criptoanalista en pugui deduir cap patró entre passphrases repetides i les seves claus corresponents. Cal tenir en compte, que la sal, generada en el moment del xifrat del fitxer, s'haurà d'emmagatzemar dins d'aquest (a la pràctica, al principi) per tal de permetre posteriorment el seu desxifrat, ja que en aquell moment s'haurà de realitzar el mateix procés (deduïr-ne la clau criptogràfica a partir de la passphrase introduïda més la sal, en aquest cas emmagatzemada prèviament). D'altra banda, si es volgués especificar explícitament el valor fixe de la sal a utilitzar per la funció PBKDF2 (tot i que no té gaire sentit), es pot afegir el paràmetre *-S valorEnHex* En tot cas, si es vol veure la sal i la clau criptogràfica generades per OpenSSL que es fan servir per realitzar un (des)xifratge concret es pot afegir el paràmetre *-p*

NOTA: Mencionar que la comanda *openssl enc* també ens permet, amb el paràmetre *-iv valorEnHex* triar un valor concret pel "vector d'inicialització" (IV) en els modes d'operació dels algoritmes d'encriptació que l'utilitzin internament, tot i que, en general, també ens interessarà que aquest valor sigui aleatori (i generat igualment per la pròpia comanda sense haver de fer-hi res). Indicar que el valor IV no cal que sigui secret, només cal que sigui el més aleatori possible en cada procés de xifrat diferent; el que no pot és repetir-se. Aquest valor no s'emmagatzema juntament amb el fitxer xifrat perquè es pot derivar a partir de la pròpia passphrase (tal com la clau). En tot cas, també podem conèixer quin és aquest valor mitjançant el paràmetre *-p* Consulteu l'article https://crypto.stackexchange.com/questions/31825/need-for-salt-with-iv/31831#31831 per llegir una explicació més exhaustiva de quina diferència hi ha entre sals i IVs,

NOTA: Mencionar també que, en lloc d'indicar una "passphrase" (a partir de la qual OpenSSL generarà, com ja sabem, la clau simètrica mitjançant l'ús de PBKDF2, a més del valor IV) és possible indicar directament la clau tal qual (en format hexadecimal), fent ús del paràmetre *-K valorEnHex* En aquest cas, però, caldrà indicar obligatòriament,a més, el paràmetre *-iv valorEnHex* perquè la comanda *openssl* no calcularà automàticament el valor de IV (com fa quan se li proporciona una "passphrase") sinó que se li ha de donar ja indicat. Per generar aquests dos valors es pot fer ús de la utilitat *openssl rand*, com es detalla al final d'aquest document

Si no es vol indicar la "passphrase" interactivament sinó que es vol indicar-la directament a la pròpia comanda, cal indicar el paràmetre *-pass pass:hola* (suposant que la "passphrase" sigui "hola"). També es pot indicar així: *-pass env:VAR* si la variable d'entorn VAR especificada té com a valor la "passphrase" adient, o també així: *-pass file:/ruta/fitxer.txt* si el fitxer especificat (el qual podria estar al seu torn xifrat amb algun algoritme simètric...) té com a primera línia el valor de la "passphrase" adient, o també així: *-pass stdin* si la "passphrase" es rebrà a través d'una canonada. Per saber més opcions pots consultar l'apartat "Pass Phrase Options" de *man openssl*

La sortida xifrada per defecte es genera en forma binària però es pot afegir el paràmetre *-a* per transformar el binari xifrat en text, que és més fàcil de gestionar. Cal tenir en compte, però, que si volguéssim desxifrar un arxiu determinat que estigués en format textual en comptes de en format binari, llavors serà obligatori afegir aquest paràmetre *-a* a la comanda de desxifrat (que de seguida veurem) per a què primer "decodifiqui" l'entrada de text a binari i llavors implementi el desxifrat pròpiament dit. En realitat, el que fa el paràmetre *-a* (en el cas del procés de xifrat) és transformar contingut binari a contingut textual no pas de qualsevol format sinó *codificat* en l'algoritme Base64 (o viceversa en el procés de desxifrat); de fet, un nom alternatiu al paràmetre *-a* és *-base64* ; teniu més informació sobre aquest algoritme de *codificació* a la taula següent. Cal tenir en compte, d'altra banda, que aquesta funcionalitat de la comanda *openssl* es pot utilitzar per *codificar* contingut binari en Base64 i viceversa sense que hagi d'existir cap procés d'encriptació/desencriptació; és a dir, podem transformar un contingut binari en Base64 (i prou, sense xifrar!) simplement executant la comanda *openssl enc -a [-in /ruta/fitxer] [-out /ruta/fitxer.b64]* i el procés invers (és a dir, "**d**ecodificar" de Base64 a binari) amb la comanda *openssl enc -d -a [-in /ruta/fitxer.b64] [-out /ruta/fitxer]*

"Base64" és un algoritme molt conegut però **no serveix per (des)xifrar** sinó per (des)codificar. La diferència està en que els algoritmes de codificació no empren cap clau per transformar el text original en un altre sinó que utilitzen uns passos clarament definits i coneguts (i, repetim, sense cap clau!) de manera que qualsevol pugui tornar al missatge "original" (és a dir, descodificar) a partir del missatge codificat sense cap misteri (de fet, només consultant una taula d'equivalència). Per exemple, el sistema ASCII o Unicode són algoritmes (o més aviat, taules) de codificació massivament utilitzats per (de)codificar combinacions de bits en símbols alfanumèrics. El sistema "base64" en concret, s'utilitza per transformar bytes (per natura, binaris) en una representació textual (amb un joc de 64 caràcters, d'aquí el nom) que permeti manipular aquests bytes amb eines pensades per treballar amb caracters i no pas bits; es fa servir molt, per exemple, per l'enviament de fitxers adjunts (per exemple, una foto) en correus

electrònics, ja que els primers servidors de correu no podien treballar amb contingut binari : el que fan els clients en aquest cas és transformar el contingut binari del fitxer a adjuntar en la seva representació textual basada en "base64" i enviar-ne aquesta com a part del missatge de correu (el destinatari haurà de realitzar la tasca contrària, és clar: identificar el text "base64" i procedir a la seva decodificació per tornar a obtenir el fitxer binari original). Concretament, la idea de l'algoritme "base64" es agafar tot el conjunt de bytes a codificar, separar-los en conjunts de 6 bits i associar a cadascun d'aquests 6 bits un determinat símbol, determinat per la taula https://en.wikipedia.org/wiki/Base64#Base64_table ; en el cas d'haver d'omplir forats al final, s'emprarà el símbol "="

NOTA: Existeix una comanda especialitzada en fer aquestes transformacions Base64<->Binari sense que calgui, doncs, utilitzar la suite OpenSSL. Concretament, es tracta de la comanda *base64* ; per codificar s'utilitzaria així: *base64 unfitxer.binari > unfitxerde.text* i per decodificar així: *base64 -d unfitxerde.text > unfitxer.binari*

*Desxifrar el contingut d'un fitxer amb un algoritme simètric:

openssl enc -d -XXX [-in /ruta/fitxer.enc] [-out /ruta/fitxer]

NOTA: Si no s'especifica cap fitxer d'entrada, l'entrada serà *stdin* (i aquesta s'acabarà pulsant CTRL+D).
NOTA: Si no s'especifica cap fitxer de sortida, la sortida serà *stdout*
NOTA: El valor "-XXX" representa el nom de l'algoritme simètric escollit (precedit per un guió). Aquest nom pot ser un de la llista mostrada per *openssl enc -list*, com per exemple "**aes-256-cbc**"
NOTA: Existeixen "àlies" de les comandes *openssl enc -d -XXX ...* amb la forma *openssl XXX -d ...*
NOTA: Si en el procés de xifrat es va emprar el paràmetre *-pbkdf2* o *-iter nnnn*, caldrà indicar-los igualment i obligatòriament en la comanda de desxifrat per a què aquest tingui éxit
NOTA: Aquesta comanda demana una "passphrase" interactivament que servirà per comprovar si la clau real utilitzada pel xifratge es correspon amb la generada en aquell moment a partir d'ella. Si el que es vol és indicar la "passphrase" directament a la pròpia comanda, cal indicar el paràmetre *-pass pass:hola* (suposant que la "passphrase" sigui "hola"). També es pot indicar així: *-pass env:VAR* si la variable d'entorn VAR especificada té com a valor la "passphrase" adient, o també així: *-pass file:/ruta/fitxer.txt* si el fitxer especificat (el qual podria estar al seu torn xifrat amb algun algoritme simètric...) té com a primera línia el valor de la "passphrase" adient, o també així: *-pass stdin* si la "passphrase" es rebrà a través d'una canonada. Per saber més opcions consulta l'apartat "Pass Phrase Options" de *man openssl*
NOTA: Si l'arxiu a desxifrar estigués en format ascii en comptes de en format binari, s'haurà afegir a la comanda anterior el paràmetre *-a* (per tal de decodificar primer aquest arxiu de Base64 a binari i llavors, un cop en format binari, procedir al desxifrat).

---

*Crear una clau privada de tipus RSA (anomenada "private.key"):

openssl genpkey -algorithm RSA [-pkeyopt rsa_keygen_bits:2048] [-XXX] [-out private.key]

NOTA: Si no s'especifica cap fitxer de sortida, la sortida serà *stdout*
NOTA: L'algoritme escollit s'indica amb el paràmetre *-algorithm*.
NOTA: El paràmetre *-pkeyopt* serveix per especificar les característiques tècniques de la clau a generar, que dependran de l'algoritme triat. En l'exemple anterior concret s'indica el seu tamany, amb l'opció *rsa_keygen_bits* (el qual per defecte ja seria de 2048, de fet). Un altre paràmetre que es podria també afegir és *-pkeyopt rsa_keygen_pubexp:65537* , que serveix per indicar el valor de l'exponent usat per calcular la clau RSA (tot i que normalment això no caldrà perquè el valor per defecte ja és suficient).
NOTA: Es pot afegir el paràmetre *-outform {PEM|DER}* per especificar el format de la clau. Per defecte és PEM (que és un format textual, al contari que DER, que és binari)

La comanda anterior no xifrarà la clau generada, a no ser que s'escrigui el valor "-XXX" representant el nom de l'algoritme simètric escollit per realitzar aquesta acció (nom que pot ser un -precedit per un guió- de la llista mostrada per *openssl enc -list*, com per exemple "**aes-256-cbc**", entre altres). Si, efectivament, s'indica aquest paràmetre, llavors es preguntarà una "passphrase" interactivament que serà utilitzada per xifrar la clau. No obstant, cal tenir present que el fet que la clau estigui xifrada implicarà que cada cop que la màquina hagi de fer-la servir (com per exemple, a l'hora de posar en marxa un servidor web segur HTTPS) haurà de demanar la "passphrase" associada per desxifrar la clau i llavors poder-la utilitzar. Per tant, l'inconvenient de xifrar la clau amb "passphrase" és que cada cop que aquesta clau es necessiti fer servir, s'haurà d'introduir la "passphrase" d'alguna manera, i si aquesta manera és interactiva, fins que una persona no la escrigui, el servidor en qüestió no s'iniciarà. Afortunadament, es pot indicar la "passphrase" d'altres formes no interactives (ja estudiades en paràgrafs anteriors) com per exemple mitjançant el paràmetre *-pass pass:hola* (estem suposant que el

"passphrase" és "hola") o mitjançant *-pass env:VAR* (si la variable d'entorn VAR especificada té com a valor la "passphrase" adient), o també així: *-pass file:/ruta/fitxer.txt* (si el fitxer especificat -el qual podria estar al seu torn xifrat amb algun algoritme simètric...- té com a primera línia el valor de la "passphrase" adient), o també així: *-pass stdin* (si la "passphrase" es rebrà a través d'una canonada)...per saber més opcions consulta l'apartat "Pass Phrase Options" de *man openssl* .

> NOTA: L'inconvenient de no incloure cap "passphrase" a la clau privada és que si algú la robés del disc dur de la nostra màquina (o d'alguna còpia de seguretat on la tinguéssim emmagatzemada) i se l'emportés al seu sistema, aquesta podria utilitzar-se com a impostora. Cal tenir en compte, però, que la "passphrase" no ens protegeix si estem fent servir la nostra clau privada en memòria (ja que allà roman desxifrada) i algú és capaç d'obtenir-la d'allà fent servir tècniques forenses (més si la nostra màquina es tracta d'un servidor en continu funcionament accessible des de l'exterior). Es pot evitar aquest risc fent servir dispositius hardware anomenats "Hardware Security Modules" (HSMs), els quals protegeixen les claus privades en cas d'un compromís del servidor però són molt cars, així que només són justificables en organitzacions amb uns requisits de seguretat estrictes. Per tant, les "passphrases" haurien de veure's només com un mecanisme per protegir les claus privades quan no estan en ús (és a dir, en sistemes de producció). En aquest context, també és important donar permisos restrictius a la clau privada generada de manera que només l'"usuari "root" (o l'usuari amb el què estigui funcionant el servei que la necessiti utilitzar) puguin llegir-la (via *chown root:root* i *chmod 400* o *chown root:www-data* i *chmod 440*, o similars)

*Crear una clau privada de tipus corba el·líptica P-256 (anomenada "private.key"):

openssl genpkey -algorithm EC -pkeyopt ec_paramgen_curve:prime256v1 [-XXX] [-out private.key]

> NOTA: Si no s'especifica cap fitxer de sortida, la sortida serà *stdout*
> NOTA: L'algoritme escollit s'indica amb el paràmetre *-algorithm*. En aquest cas, és "**E**l.liptic **C**urve", que no és un algoritme concret sinó un tipus d'algoritme matemàtic amb interessants característiques criptogràfiques; l'algoritme concret que es farà servir llavors s'indicarà amb el paràmetre *-pkeyopt* (veure següent nota)
> NOTA: El paràmetre *-pkeyopt* serveix per especificar les característiques tècniques de la clau a generar. Com a mínim (i és el que es fa en l'exemple anterior) s'haurà d'indicar el nom de la corba concreta a utilitzar, amb l'opció *ec_paramgen_curve* Aquest valor determinarà, entre altres, el tamany de la clau així com altres característiques. Per saber els possibles noms a indicar com a valor d'aquesta opció es pot executar la comanda *openssl ecparam -list_curves* , la qual en mostra la llista.
> NOTA: Es pot afegir el paràmetre *-outform {PEM|DER}* per especificar el format de la clau. Per defecte és PEM
> NOTA: Llegeix els paràgrafs relacionats amb la comanda anterior sobre els possibles algoritmes de xifrat que es poden indicar com a valor "-XXX" així com les implicacions d'afegir una "passphrase" (i les maneres de fer-ho).

Una manera alternativa de crear una clau el·líptica equivalent és mitjançant la comanda *openssl ecparam -genkey -name prime256v1 -out private.key* però aquí no tenim la possibilitat de xifrar-la (és a dir, d'indicar-li una "passphrase"). Si volem fer-ho, caldria usar la combinació de comandes següents: *openssl ecparam -genkey -name prime256v1 | openssl ec [-XXX] -out private.key*

*Eliminar la passphrase d'una clau privada ja existent (útil per no haver d'escriure-la cada cop):

openssl pkey -in private.key -out newPrivate.key

> NOTA: Aquesta comanda pregunta la "passphrase" a eliminar interactivament. Per indicar-la directament en la comanda es pot afegir algun dels paràmetres següents (ja estudiats anteriorment): *-passin pass:hola* o *-passin env:VAR* o *-passin file:/ruta/fitxer.txt* o *-passin stdin*, entre altres. Per saber més opcions consulta l'apartat "Pass Phrase Options" de *man openssl*.
> NOTA: Es pot afegir el paràmetre *-inform {PEM|DER}* per especificar el format de la clau d'entrada (per defecte és PEM). D'altra banda, es pot afegir el paràmetre *-outform {PEM|DER}* si es vol especificar un format concret per la clau de sortida (per defecte és PEM).

Aquesta comanda també es pot fer servir pel contrari: per afegir una "passphrase" a una clau que no en tenia. Per fer això cal utilitzar els següents paràmetres (suposant que la "passphrase" és "hola"): *-XXX -passout pass:hola* on "XXX" representa el nom de l'algoritme simètric escollit (precedit per un guió), el qual pot ser un de la llista mostrada per *openssl enc -list*, com per exemple "aes-256-cbc", entre molts d'altres.

*Comprovar les característiques d'una clau privada:

openssl pkey -in private.key -check -noout

> NOTA: El paràmetre *-check* serveix simplement per mostrar el missatge "Key is valid" (o "Key is invalid")
> NOTA: El paràmetre *-noout* serveix per no mostrar el contingut de la clau en sí
> NOTA: Es pot afegir el paràmetre *-text* per obtenir més metainformació sobre la clau
> NOTA: Aquesta comanda demanarà interactivament una "passphrase" (si la clau privada ha sigut creada amb ella). Per indicar-la directament en la comanda es pot afegir algun dels paràmetres següents (ja estudiats anteriorment): *-passin pass:hola* o *-passin env:VAR* o *-passin file:/ruta/fitxer.txt* o *-passin stdin*, entre altres. Per saber més opcions consulta l'apartat "Pass Phrase Options" de *man openssl*.

En el cas de claus el.líptiques, la comanda equivalent és *openssl ec -in private.key -check -noout*

*Crear una clau pública (anomenada "public.key") a partir d'una clau privada (ja existent):

openssl pkey -in private.key -pubout -out public.key

En el cas de claus el.lípt., la comanda equivalent és *openssl ec -in private.key -pubout -out public.key*

---

*Xifrar el contingut d'un fitxer amb una clau pública (ja existent):

openssl pkeyutl -encrypt [-in /ruta/fitxer] [-out /ruta/fitxer.enc] -pubin -inkey public.key

> NOTA: Si no s'especifica cap fitxer d'entrada, l'entrada serà *stdin* (i aquesta s'acabarà pulsant CTRL+D).
> NOTA: Si no s'especifica cap fitxer de sortida, la sortida serà *stdout*
> NOTA: El paràmetre *-pubin* serveix per indicar que la clau especificada al paràmetre *-inkey* és pública (ja que per defecte *openssl* sempre la interpreta com a privada).

Aquesta comanda, no obstant, no permet xifrar arxius més grans que el tamany de la clau pública emprada per una limitació de la pròpia criptografia asimètrica. On se sol fer servir és precisament per xifrar altres claus que existeixin també en forma de fitxers (els quals sí que són suficientment petits), normalment claus simètriques que no donen cap problema, aquestes no, en fer-les servir per xifrar arxius arbitràriament grans. La idea més comuna, per tant, és xifrar fitxers amb una clau simètrica (com ja hem vist anteriorment) que tindrà forma també de fitxer i tot seguit protegir-la amb una clau pública (gràcies a la comanda anterior) per tal de poder-la enviar, juntament amb els fitxers xifrats, al destinatari d'una manera segura; el destinatari llavors haurà de desxifrar primer la clau simètrica rebuda (amb la comanda que veurem a continuació, fent servir la seva pròpia clau privada) i, un cop fet això, podrà desxifrar amb ella tots el/s fitxer/s en qüestió.

*Desxifrar el contingut d'un fitxer amb una clau privada (ja existent):

openssl pkeyutl -decrypt [-in /ruta/fitxer.enc] [-out /ruta/fitxer] -inkey private.key

> NOTA: Si no s'especifica cap fitxer d'entrada, l'entrada serà *stdin* (i aquesta s'acabarà pulsant CTRL+D).
> NOTA: Si no s'especifica cap fitxer de sortida, la sortida serà *stdout*
> NOTA: Aquesta comanda demanarà interactivament una "passphrase" (si la clau privada ha sigut creada amb ella). Per indicar-la directament en la comanda es pot afegir algun dels paràmetres següents (ja estudiats anteriorment): *-passin pass:hola* o *-passin env:VAR* o *-passin file:/ruta/fitxer.txt* o *-passin stdin*, entre altres. Per saber més opcions consulta l'apartat "Pass Phrase Options" de *man openssl*.

---

*Signar el contingut d'un fitxer amb una clau privada (ja existent):

openssl pkeyutl -sign [-in /ruta/fitxer] [-out /ruta/fitxer.sig] -inkey private.key

NOTA: Veure totes les notes de la comanda anterior

La comanda anterior genera un fitxer de sortida que representa el fitxer d'entrada signat (que estarà sempre en format binari; si es volgués en format textual caldrà convertir-lo a Base64 en un pas posterior perquè aquesta comanda no incorpora el paràmetre *-a* per fer-ho en un sol cop). Només oferint aquest fitxer de sortida a l'altre extrem ja és suficient per a què aquest verifiqui la signatura i n'obtingui el contingut original (veure següent comanda)

*Verificar (i extreure) el contingut original d'un fitxer signat amb una clau pública (ja existent):

openssl pkeyutl -verifyrecover -in /ruta/fitxer.sig [-out /ruta/fitxer] -pubin -inkey public.key

NOTA: Si no s'especifica cap fitxer d'entrada, l'entrada serà *stdin* (i aquesta s'acabarà pulsant CTRL+D).
NOTA: Si no s'especifica cap fitxer de sortida, la sortida serà *stdout*

El fitxer signat ha d'estar en format binari per a què la comanda anterior funcioni; si aquest estigués codificat en Base64, abans d'executar la comanda anterior caldrà, doncs, decodificar-lo primer (perquè la comanda no incorpora el paràmetre *-a* per poder-ho fer-ho tot de cop)

*Verificar que el fitxer signat es correspongui a un determinat fitxer original:

openssl pkeyutl -verify -in /ruta/fitxer -sigfile /ruta/fitxer.sig -pubin -inkey public.key

Si la comprovació és exitosa, aquesta comanda mostrarà per pantalla el missatge "Signature Verified Succefully"; si no mostrarà el missatge "Signature Verification Failure"

---

*Crear un "hash" (del contingut d'un fitxer o de *stdin*):

openssl dgst {-md5 | -sha512 | ... } [/ruta/fitxer]

NOTA: L'algoritme del "hash" s'indica després del paràmetre *dgst*. Pot ser un dels llistats a *openssl dgst -list*
NOTA: Si no s'especifica cap fitxer, l'entrada serà *stdin* (i aquesta s'acabarà pulsant CTRL+D).
NOTA: Es pot guardar la sortida en un fitxer amb el paràmetre *-out /ruta/hash.txt* D'aquesta manera, es podrà sempre tornar a crear el "hash" del fitxer en qualsevol moment i comparar-ho amb l'inicialment guardat en aquest fitxer per confirmar que no hi hagi hagut cap canvi
NOTA: Existeixen "àlies" de les comandes "digest"; així, *openssl md5 ...* és equivalent a *openssl dgst -md5 ...* , *openssl sha512 ...* és equivalent a *openssl dgst -sha512 ...*

A l'hora de protegir un "hash" d'eventuals modificacions (i així, garantir la seva pròpia integritat i autenticació, així com les del missatge associat corresponent) tenim la possibilitat de fer servir criptografia asimètrica (amb l'us de signatures) o bé criptografia simètrica (amb l'ús de HMACs).

* En el primer cas, cal signar/verificar el "hash" (amb, respectivament, una clau privada/clau pública ja existents) en el moment de crear-lo/comprovar-lo. Per fer la signatura, simplement caldrà afegir el paràmetre *-sign private.key* ; per fer la verificació, suposant que el hash signat al pas anterior es va guardar, gràcies al paràmetre *-out*, en un fitxer anomenat "hashsignat.txt", caldrà afegir en canvi els paràmetres *-verify public.key -signature /ruta/hashsignat.txt* (a més d'indicar al final la ruta del fitxer original)

* En el segon cas, cal xifrar/desxifrar el "hash" amb una determinada contrasenya en el moment de crear-lo/comprovar-lo. Per fer això, simplement caldrà afegir el paràmetre *-hmac "contrasenya"* a la comanda anterior

NOTA: Cal dir que també existeix una comanda de la suite OpenSSL específicament especialitzada en l'ús de HMAC anomenada *openssl mac*, de la qual es pot consultar el seu funcionament a la seva pàgina *man openssl-mac*

*Crear un "hash" amb una sal (d'una contrasenya per gravar-ho a l'arxiu "/etc/shadow", per exemple):

openssl passwd -6 [-salt unaSal] [contrasenya]

NOTA: El paràmetre *-6* indica que el hash serà de tipus SHA512 (el típic en Linux actualment). Altres números indiquen altres algoritmes
NOTA: Si no s'especifica cap sal, se'n generarà una aleatòria de 16 bytes.
NOTA: Si no s'especifica cap contrasenya, s'agafarà de *stdin* (i aquesta s'acabarà pulsant CTRL+D). També es podria indicar, si estigués escrita dins un fitxer (amb o sense salt de línia al final, això és irrellevant), amb el paràmetre *-in /ruta/fitxer*
NOTA: No existeix cap opció *-out* per guardar el resultat en un fitxer: sempre sortirà per la *stdout*

---

*Generar un valor pseudoaleatori:

openssl rand [-out /ruta/fitxer] nºbytes

NOTA: Si no s'especifica cap fitxer de sortida, la sortida serà *stdout*
NOTA: El paràmetre *-hex* fa que la sortida es mostri en format hexadecimal
NOTA: El paràmetre *-base64* fa que la sortida es mostri en codificació Base64.

Aquesta comanda pot ser útil per generar claus en forma de fitxers directament sense haver de fer servir cap funció KDF (per llavors indicar-les amb el paràmetre *-pass file:/ruta/fitxer* o similar quan s'escaigui)