Xifrat de discos/particions o carpetes (I)

Tot i que ja hem explicat en PDFs anteriors com xifrar un arxiu puntual (tant amb *openssl* com amb *GnuPG*), aquest procediment només és útil si es vol protegir un arxiu. essent impracticable si el que es pretén és emmagatzemar varis arxius en disc, ja que exigeix que cada vegada que es vulgui realitzar la modificació del contingut, fer un desxifrat i un xifrat manual. A la pràctica, el que es fa per evitar aquest inconvenient és xifrar la totalitat (o part) d'un sistema d'arxius i així aconseguir que tot l'inclòs en aquella zona estigui xifrat i, per tant, no pugui ser llegit (o escrit) llevat que es conegui la contrasenya. Per posar aquesta idea en pràctica hi ha tres estratègies.

1. Xifrar dispositius de blocs complets (normalment discos o, sobre tot, particions)
2. Xifrar amb un software determinat el contingut de carpetes concretes en un sist. de fitxers no xifrat
3. Fer servir les capacitats de xifrat del propi sistema d'arxius, si el que fem servir les té (com Ext4)

A nivell de disc/partició (dispositiu de blocs)

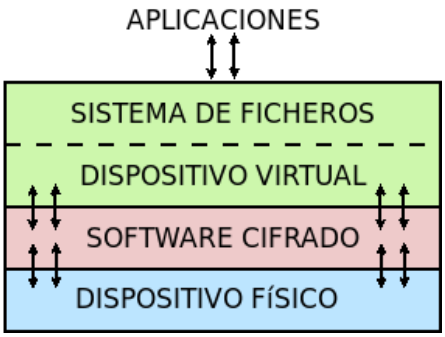
Els mètodes de xifratge de dispositius de bloc (discos, particions, fitxers "loop" o fins i tot fitxers regulars!!, etc) funcionen per sota de la capa del sistema de fitxers i així s'asseguren que tot el que s'hi escrigui directament al dispositiu de bloc estigui xifrat. Això vol dir que, mentre el dispositiu de bloc està desmuntat, tot el seu contingut sembla una gran quantitat de dades aleatòries, sense manera de determinar quin tipus de sistema de fitxers i dades conté. D'aquesta manera, qualsevol usuari maliciós que volgués, per exemple, accedir a aquestes dades des d'un sistema "Live", no podria. L'accés a les dades només es pot realitzar, un cop el dispositiu de bloc ha sigut formatat amb qualsevol sistema de fitxers (el que sigui en concret és irrellevant), a través d'un punt de muntatge (en una ubicació arbitrària), muntatge que necessita, per completar-se correctament, d'una contrasenya que haurà de proporcionar l'usuari en aquell moment per poder llavors desxifrar el contingut del dispositiu i oferir-lo de forma transparent.

> NOTA: Aquest funcionament implica, doncs, que per muntar/desxifrar un disc en arrencar el sistema, cal introduir manualment la contrasenya a la consola cada cop que aquest s´inicïi (hi ha altres mètodes per assegurar el muntatge que són no interactius, però el mètode de la contrasenya interactiva és el més habitual).

Hi ha diversos mètodes / tecnologies / programari disponible a Linux:

Veracrypt (https://www.veracrypt.fr) "Fork" del projecte (ja abandonat) Truecrypt. Proporciona interfície gràfica. Multiplataforma. No obstant, té una llicència particular que fa que no estigui directament disponible als repositoris de les distribucions més importants
Zulucrypt (http://mhogomchungu.github.io/zuluCrypt) : Compatible amb Veracrypt i Cryptsetup. Proporciona interfície gràfica. Només funciona en sistemes Linux
Cryptsetup (https://gitlab.com/cryptsetup/cryptsetup) : Compatible amb Veracrypt si és necessari. No proporciona interfície gràfica. Només funciona en sistemes Linux. És el sistema que estudiarem a continuació degut a ser el més comú i extés
Cryptmount (https://github.com/rwpenney/cryptmount) : Utilitza la mateixa tecnologia del kernel que Cryptsetup. No proporciona interfície gràfica Només funciona en sistemes Linux
Loop-AES (https://sourceforge.net/projects/loop-aes) : Especialitzat en xifrar arxius "loop".

Cryptsetup es basa en el mòdul del nucli "dm-crypt" (https://gitlab.com/cryptsetup/cryptsetup), que al mateix temps es basa en el mòdul del nucli "device-mapper" (https://gitlab.com/lvmteam/lvm2) . El mòdul "dm-crypt" és un subsistema de xifratge de disc transparent que el que fa és assignar un dispositiu de bloc físic a un dispositiu de bloc virtual: quan s'escriu al dispositiu virtual, cada bloc de dades es xifra i s'emmagatzema al dispositiu físic i quan es llegeix des del dispositiu virtual, cada bloc es desxifra en temps d'execució. És a dir, els blocs de dades s'escriuen xifrats al dispositiu d'emmagatzematge mentre que el dispositiu virtual és llegit com un dispositiu de blocs normals sense xifrar. D'altra banda, el mòdul "device-mapper" proporciona una manera genèrica de crear capes virtuals de dispositius de blocs entre el dispositiu físic i el sistema de fitxers. Per tant, la combinació "device-mapper+crypt" proporciona xifratge transparent de dispositius de blocs mitjançant l'API criptogràfica del kernel, tal com mostra la figura següent.



A la pràctica, per implementar un dispositiu xifrat només cal que l'usuari especifiqui una "passphrase", la qual servirà per crear el dispositiu de bloc virtual a "/dev". A partir de llavors aquest nou dispositiu es podrà formatar amb un sistema de fitxers qualsevol com de costum (o apilar-lo amb un altre dispositiu "dm-crypt" en un volum RAID o LVM) i muntar-lo; tal com hem dit, les escriptures en aquest dispositiu es xifraran i les lectures es desxifraran.

> NOTA: El tipus de xifrat usat en aquest tipus de dispositius és simètric i l´algoritme concret sol ser predefinit, tot i que l´usuari podria triar-ne un altre de diferent si volgués (es pot consultar "/proc/crypto" per saber els modes i xifrats compatibles). En tot cas, quan estudiem les comandes concretes a fer servir en els propers paràgrafs, ja concretarem com es pot fer això. El que sí que cal tenir clar és que la "passphrase" indicada no és la clau real, sinó que s'utilitza (combinada amb una sal) per derivar internament, fent ús de "/dev/urandom" i a partir d'aquesta combinació "passphrase"+sal, la clau real binària (amb una longitud fixa donada per l'algoritme de xifratge utilitzat) fent-se servir per això una determinada funció de derivació de clau (KDF)
>
> NOTA: En realitat el procés de derivació de clau explicat a la nota anterior és una mica més complex: la "passphrase" s'usa per generar una clau, sí, però aquesta clau s'utilitza per xifrar/desxifrar una altra clau que s'emprarà, aquesta sí, per xifrar/desxifrar el disc. Això és necessari fer-ho així per tal de permetre canviar la "passphrase" en un moment donat sense haver de rexifrar tot el disc sencer; fent-ho així només caldrà rexifrar la clau xifradora del disc i prou. A més, aquest sistema també permet tenir més d'una "passphrase" -o sistemes diversos- per xifrar/desxifrar un determinat disc. En qualsevol cas, si es volgués canviar la clau xifradora/desxifradora real, amb Cryptsetup es pot fer però això implicarà rexifrar tot el disc

Cryptsetup es pot utilitzar en mode "plain" (és a dir, amb opcions "raw" -i perilloses!- executades directament) o en combinació amb el marc LUKS (https://gitlab.com/cryptsetup/cryptsetup també), que proporciona una manera genèrica, més senzilla i agradable de gestionar tots els aspectes del xifratge/desxifratge de dispositius de blocs (però menys flexible). Específicament, LUKS (dels seus segles en anglès, "Linux Unified Key Setup") és una especificació de xifratge de disc que defineix un format estàndard, independent de la plataforma, per utilitzar amb diverses eines. Això facilita la compatibilitat i la interoperabilitat entre els diferents programes. Els gestors d'arxius actuals, suporten el reconeixement automàtic de particions LUKS, sol·licitant la contrasenya del dispositiu xifrat quan aquest es connecta per tal de muntar la partició corresponent.

Plain Mode vs LUKS:

In plain mode, dm-crypt simply encrypts the device sector-by-sector, without adding any kind of headers or metadata. It's advised that beginners do not use this until they understand the risks.

Advantages of using Plain Mode are:
*It's more resistant to damage compared to LUKS: header can't be destroyed, resulting in a lost data set.
*Details of the encryption algorithm are obscured.
*It's not immediately obvious that the user is using encryption or dm-crypt.

Disadvantages of using Plain Mode are:
*If you enter the wrong password, no checks are performed and dm-crypt accepts any passphrase without complaining. Although you may notice something is wrong when your filesystem refuses to mount, there's still the potential of overwriting useful data.
*The same is true for opening your block device with different encryption settings. This might happen when you upgrade your distribution and the defaults change or when you open device on a different O.S
*There's no easy way of changing your password, the whole container will have to be decrypted with the old password and re-encrypted with the new secret.

Advantages of using the LUKS extension are:
*You can change your password without re-encrypting the whole block device.
*Multiple decryption keys are supported so you can share the container with others but without sharing your password (it's possible to handle multiple user keys with one master key). Keys can also be revoked.
*Encryption settings (cipher algoritm, key size, etc) are stored in a header at start of device (the metadata block). These settings protect you against accidentally re-encrypting with a different password or different cryptographic parameters.

Disadvantages of using LUKS are:
*Header is not encrypted so containers are easily recognizable. Encryption settings are also stored in clear. Keys are hashed and stored there; that's the reason it has to be secured with a passphrase.
*As we has said, header contains the keys which decrypts block device, so if that is overwritten there is no way to recover your data.

This doesn't mean that plain mode is worse than LUKS, just that one mode of operation is simple and raw, that some experts might prefer, and requires more attention and care, while the other is a more user-friendly option, includes more bells and whistles and tries to protect beginners against some common mistakes.

***A la pràctica:**

A la pràctica, per gestionar un dispositiu de bloc amb Cryptsetup + LUKS, primer cal tenir instal·lat Cryptsetup (*sudo apt/dnf install cryptsetup*) i cal triar el dispositiu de bloc a encriptar (que pot ser un disc com "/dev/ sdb" o una partició com "/dev/sdb1", o un dispositiu "loop", etc). Com a exemple, a continuació farem servir un dispositiu "loop" (en aquest cas, "/dev/loop123") associat a un fitxer qualsevol (anomenat "disc1"), executant la següent ordre: *truncate -s 10G disc1 && sudo losetup /dev/loop123 disc1*

NOTA: En el moment d'executar la comanda *losetup* pot ser que /dev/loop123 estigui essent utilitzat ja; en aquests casos, va bé executar la comanda *losetup -f* per a obtenir el primer dispositiu "loop" disponible
NOTA: Altres formes de crear un fitxer és, per exemple, amb la comanda *fallocate -l 10G disc1* o també amb la comanda *dd if=/dev/zero of=disc1 bs=1024M count=10* (tot i que aquest fa ocupar realment l'espai de disc)
NOTA: Una altra possibilitat, en lloc d'utilitzar dispositius "loop", és utilitzar dispositius "brd" ("block RAM disks"), els quals representen dispositius de bloc el contingut del qual, però, es perdrà al següent reinici del sistema. Per utilitzar-los només cal carregar un mòdul del kernel específic executant la comanda *sudo modprobe brd rd_size=nº* (on el paràmetre "rd_size" indica la mida del dispositiu "brd" en KB; altres paràmetres opcionals poden ser *max_part=nº*, que indica el nombre màxim de particions que es permetran dins del dispositiu "brd" i *rd_nr=nº*, que indica el nombre de dispositius "brd" que es crearan) i, a partir d'aquí ja es podrà disposar com a mínim d'un dispositiu anomenat "/dev/ram0" (i/o "dev/ram1", "/dev/ram2", etc...segons el valor que s'hagi indicat amb el paràmetre *rd_nr)*, dispositiu que es podrà particionar (anomenant-se les particions "/dev/ram0p1", "/dev/ram0p2", etc), formatejar i muntar com qualsevol altre disc

A partir d'aquí, el que farem en els passos següents és encriptar el dispositiu sencer (és a dir, "/dev/loop123" en aquest cas). En aquest punt cal aclarir que els dispositius "loop" poden actuar com a discos durs sencers o bé com a particions, segons com els tractem. En general, per a allò que voldrem fer ens serà més còmode, senzill i pràctic tractar els dispositius "loop" com a particions independents (de l'estil "/dev/sdb2", "/dev/sdc3", etc) en comptes de discos (del estil "/dev/sdb", "/dev/sdc", etc), -veure nota següent per al perquè-

NOTA: Encara que tractant els dispositius "loop" com a discos durs complets aconseguiríem que totes les particions definides en ells passessin a estar automàticament encriptades també, la gestió del volum complet es complica força a l'hora de muntar-lo (sobretot si es vol fer de forma automàtica mitjançant "/etc/fstab", per la qual cosa, tal com hem dit, en aquest document optarem per tractar cada dispositiu "loop" com una partició individual (indicant, això sí, com s'hauria de tractar algun aspecte concret de manera diferent en el cas de tractar amb discos durs sencers en comptes de amb particions, si fos necessari)

1.-AIXÒ NOMÉS CAL FER-HO UNA VEGADA: Generar la capçalera (el "metadata block") per a aquest dispositiu triat, preguntant-se interactivament la "passphrase" que s'associarà a la clau criptogràfica binària real, la qual s'emmagatzemarà a la capçalera i que servirà per desencriptar el dispositiu cada vegada que es necessiti fer servir: *sudo cryptsetup luksFormat /dev/loop123*

2.-AIXÒ NOMÉS CAL FER-HO UNA VEGADA: "Mapejar" ("obrir") el dispositiu encriptat en el corresponent dispositiu virtual en clar (anomenat en aquest cas "/dev/mapper/pepe") per així poder assignar-li el sistema de fitxers que vulguem (Ext4 en aquest cas), tal com faríem en qualsevol disc "estàndard" per poder començar a treballar-hi. A l'hora d'obrir el dispositiu, es preguntarà la passphrase: *sudo cryptsetup open /dev/loop123 pepe && sudo mkfs.ext4 /dev/mapper/pepe*

NOTA: Noteu que s'ha donat format a tot el dispositiu "pepe" en global, no a particions individuals ("pepe1", "pepe2"...) En el cas, però, que volguéssim tenir-les xifrades individualment (totes elles estarian xifrades llavors amb la mateixa "passphrase", això sí), el procediment seria una mica més llarg, així:

sudo cryptsetup open /dev/loop123 pepe
Hem de crear les particions desitjades en "pepe" amb les subcomandes internes adients de *fdisk* (n, w...)
sudo fdisk /dev/mapper/pepe
Hem d'"avisar" al kernel per a què se n'adoni de l'existència de les particions recent creades ("pepe1"...)
sudo kpartx -a /dev/mapper/pepe
Podem veure que les particions s'han reconegut mitjançant *lsblk*
Donem format a les particions que siguin necessàries
sudo mkfs.ext4 /dev/mapper/pepe1

2BIS.-AIXÒ NOMÉS CAL FER-HO UNA VEGADA: Creem el punt de muntatge que s'assignarà al dispositiu virtual en clar (si aquest estigués particionat, haurem de crear llavors diversos punts de muntatge, cadascun per assignar-ho a les respectives particions existents a l'interior d'aquest disc virtual en clar): *mkdir /mnt/punt*

***.-**A continuació es mostren les comandes que s'hauran d'executar CADA COP que volguem muntar el dispositiu xifrat per poder treballar amb ell:

Si no s'ha executat encara: *sudo cryptsetup open /dev/loop123 pepe*
sudo mount /dev/mapper/pepe /mnt/punt
... A partir d'aquí treballarem en "/mnt/punt", etc com si fos una carpeta més qualsevol
sudo umount /mnt/punt
sudo cryptsetup close pepe

NOTA: En el cas de tenir un dispositiu xifrat amb particions, per poder muntar alguna d´elles primer hauríem d'obrir tot el dispositiu, tot seguit reconèixer les particions presents i llavors muntar la partició que volguem. Un cop acabada la feina, caldrà tancar la partició i seguidament tancar el dispositiu. És a dir, fer això:
sudo cryptsetup open /dev/loop123 pepe && sudo kpartx -a /dev/mapper/pepe
sudo mount /dev/mapper/pepe1 /mnt/part1
... A partir de aquí trabalarem en "/mnt/part1", etc como si fos una carpeta més qualsevol
sudo umount /mnt/part1
sudo cryptsetup close pepe1 && sudo cryptsetup close pepe

Es poden veure tots els detalls de la capçalera LUKS d´un dispositiu executant *sudo cryptsetup luksDump /dev/loop123* D'altra banda, es pot executar *sudo cryptsetup -v isLuks /dev/loop123* per comprovar si el dispositiu indicat té capçalera LUKS (i per tant, si és un dispositiu encriptat) i es pot executar *sudo cryptsetup status pepe* per veure si el dispositiu "/dev/mapper/pepe" està disponible (és a dir, es pot treballar amb ell perquè s'ha desencriptat -o no-)

NOTA: El xifratge predeterminat utilitzat depèn de la versió de *cryptsetup* emprada. Des de la versió 1.6.0, és "aes-xts-plain64", on "xts" és el mode d'encadenament que afecta l'aplicació del xifratge "aes" als blocs de dades consecutius ("xts" és una millora respecte al mode "cbc" utilitzat per versions anteriors). Els valors predeterminats són prou bons tret que tingueu motius rellevants per canviar-los; en general, els valors per defecte es poden consultar a les darreres línies de la sortida mostrada per l'argument *--help* de l'ordre *luksFormat*. Si vol canviar algú d'ells, es pot especificar l'argument corresponent a l'ordre *luksFormat*: *--hash nom_algoritme_hash* , *--cipher nom_algoritme_xifrat* i/o *--key-size num_bytes*

LUKS provides eight key slots, each of which can be used to store a password that can be used to access and decrypt your data (they can be viewed by executing *cryptsetup luksDump /dev/loop123* command). This allows several users to access the same device, each with their own password; if access must later be restricted to one of them, we will simply delete their password and that's all. This section will review the basic commands to edit, add, and delete these passwords.

To set up an additional passphrase (eventually in a specific free slot) that can unlock the container:
sudo cryptsetup luksAddKey [--key-slot 3] /dev/loop123 [/ruta/fitxer_passphrase]
NOTA: Instead of interactively key the passphrase to add, it's possible to specify it by pointing to a file containing it (of course, this file should be protected by restrictive read -and write!- permissions or, better, be erasen after being added to LUKS header). To use this file then in any *cryptsetup* command that needs it since then, the *--key-file* argument should be specified.
To remove a passphrase from a key slot (interactively or from slot nº3 specifically):
sudo cryptsetup luksRemoveKey /dev/loop123 o *cryptsetup luksKillSlot /dev/loop123 3*
To change a passphrase (asked interactively) by another (by default asked interactively too):
sudo cryptsetup luksChangeKey [--key-slot nº] /dev/loop123 [/ruta/fitxer_passphrase]
NOTA: Former passphrase, if binary, can be provided non-interactively by the *–key-file /ruta/fitxer_passphrase_antic* argument

Since the LUKS header is so important and losing it means losing your entire container, back it up:
sudo cryptsetup luksHeaderBackup /dev/loop123 --header-backup-file cabecera.bak
You can test the header-file without having to restoring it by using it to open the associated LUKS disc:
sudo cryptsetup open --header cabecera.bak /dev/loop123 pepe
But if you want to restore the valid header, to this:
sudo cryptsetup luksHeaderRestore /dev/loop123 --header-backup-file cabecera.bak

NOTA: You can test a scenario where the LUKS header is accidentally overwritten by doing, for instance this: *sudo dd if=/dev/zero of=/dev/loop123 bs=16M count=1* (the LUKS header's size is roughly about 16M). Trying to open your device with *cryptsetup open* will now return an error until you don't restore the header

Tal com s'ha comentat als exemples anteriors, en lloc de teclejar interactivament la passphrase, per disposar de més automatització a l'hora d'accedir al dispositiu LUKS, es pot especificar aquesta en forma de fitxer. Per aconseguir això, cal fer el següent:

1.Creem el fitxer-clau (l'anomenarem ara "/root/clau") de, per exemple, 1MByte de números aleatoris:
sudo dd if=/dev/urandom of=/root/clau bs=1M count=1 && sudo chmod 400 /root/clau
NOTA: No cal que el fitxer-clau sigui binari; també podria contenir una simple cadena de caràcters (preferiblement ASCII); en aquest cas, però, no hauria de contenir cap salt de línia
2.A continuació creem el dispositiu LUKS amb la clau binària que està al fitxer anterior:
sudo cryptsetup luksFormat /dev/loop123 /root/clau (o *cryptsetup luksAddKey /dev/loop123 /root/clau*)
3.Per a obrir-lo, haurem d'indicar-la:
sudo cryptsetup open /dev/loop123 pepe --key-file=/root/clau

Però, ¿quin sentit té xifrar una partició si deixem la clau per desxifrar-la en un fitxer d'una altra partició sense xifrar? L'interessant d'això és comprovar que es pot desar la clau en un fitxer i es pot fer servir aquest per no haver d'escriure res interactivament. I això és útil, si emmagatzemem el fitxer-clau en un dispositiu extern (com un llapis USB) que procurarem retirar i portar-nos lluny de la màquina quan no la fem servir.

NOTA: En tot cas, és convenient amagar aquest arxiu perquè passi desapercebut si algú es fa amb el nostre llapis; respecte això, el més assenyat és guardar els 512 bytes de la clau en algun espai lliure del llapis aliena als sistemes d'arxius que hi pugui haver. Si el particionat és GPT (el més habitual actualment), podem utilitzar per exemple els darrers 512 bytes de l'espai que es reserva per definir particions, ja que és força improbable que al llapis haguem creat més de 124 particions. Concretament, en un disc GPT el primer sector simula ser un MBR tradicional (512B), el segon sector és la capçalera GPT (512B) i tot seguit hi ha espai per a 128 definicions de particions, cadascuna de les quals ocupa 128 bytes (16KiB). En conseqüència el començament del disc ocupa 17KiB o el que és el mateix 34 sectors, així que podem ocupar el sector 34 per emmagatzemar la nostra clau, amb l'únic cost que "només" podrem definir 124 particions, cosa que, certament, no sembla cap problema. Així doncs, suposant que el llapis USB es troba a "/dev/sdb", crearem una clau aleatòria de 512 bytes directament sobre el seu sector 34 així: *sudo dd if=/dev/urandom of=/dev/sdb bs=512 count=1 seek=33* Un cop creada, l'afegim a l'slot: *{ echo "secret" ; sudo dd if=/dev/sdb bs=512 count=1 skip=33; } | sudo cryptsetup luksAddKey /dev/sda6* - on "secret" és la contrasenya que vam introduir en crear el dispositiu xifrat i que ens servia per fer el muntatge interactiu. Afegida aquesta clau, podem provar si funciona de la manera següent: *sudo dd if=/dev/sdb bs=512 count=1 skip=33 | sudo cryptsetup open /dev/sda6 xifrat –key-file=-* Ja només falta que, en arrencar el sistema, aquest busqui el dispositiu, el munti i dugui a terme justament aquesta operació d'"obertura". Per fer-ho, el més fàcil és crear una regla Udev, que en detectar el dispositiu llanci un script, tal com aquesta: *SUBSYSTEMS=="usb", ACTION=="add", ATTRS{idVendor}=="abcd", ATTRS{idProduct}==" 1234", KERNEL=="sd?", SYMLINK+="usbkey", RUN+="/usr/local/bin/unlock.sh*" La regla Udev anterior identifica el dispositiu on hem desat la clau a través dels valors *idVendor* i *idProduct* del llapis (que es poden consultar fàcilment fent *lsusb*) A més, aprofitem per afegir un enllaç simbòlic "/dev/usbkey" que apunti al dispositiu (amb aquest nom podrem referir-nos al dispositiu dins de l'script "unlock.sh"):

```
#!/bin/sh
PART="/dev/sda6"
DEVICE="/dev/usbkey"
ENCVOL="xifrat"
MOUNTP="/srv"
{  until [ -b "$PART" ]; do sleep .5; done
   dd if="$DEVICE" bs=512 count=1 skip=33 | cryptsetup open "$PART" "$ENCVOL" --key-file=-
} &
```

A "/etc/crypttab" no hi ha d'haver cap referència, ja que és l'script el que realitza l'operació de crear el dispositiu xifrat. A "/etc/fstab", sí que podem deixar la línia, però afegint l'opció *nofail*, perquè no falli el muntatge i aturi l'arrencada en cas que no es trobi el llapis (és a dir, així: */dev/mapper/xifrat /srv ext4 defaults,nofail 0 0*). En tot cas, aquesta estratègia només és vàlida si es xifra una partició de dades i no la partició del sistema perquè en aquest darrer cas, cal recórrer a una altra estratègia diferent basada en manipular la imatge initramfs

---

Un altre aspecte important a tenir en compte és com muntar automàticament una partició LUKS quan arrenqui el sistema. Per aconseguir això hem de treballar, a més de amb el fitxer "/etc/fstab", amb el fitxer "**/etc/crypttab**". Aquest últim el fa servir Cryptsetup a l'arrencada per realitzar el mapeig dels dispositius físics que s'especifiquin (és a dir, per fer l'equivalent a *cryptsetup open*). Cada línia representa un dispositiu i les quatre columnes que el descriuen signifiquen el següent:

*Nom del dispositiu mapejat sobre el dispositiu físic (és a dir, el que hi ha despŕes de "/dev/mapper")
*Ruta del dispositiu físic (o el seu UUID, indicant-lo així *UUID="xxxxx"*; aquest valor es pot conèixer fent *lsblk -o NAME,UUID*)
*Ruta (sota "/") del fitxer de clau binària (o bé "*none*" per indicar que la passphrase es teclejarà)
*Opcions per Cryptsetup (com ara el nombre d'intents per introduir la "passphrase", l'algoritme de xifratge simètric utilitzat -si aquest no és el per defecte-, la longitud de la clau -si aquesta no és la per defecte-, etc); aquí normalment només s'especificarà "*luks*" (veure *man crypttab* per més)

Un exemple de línia en aquest fitxer podria ser, suposant que xifrem la partició "sdb1", així:

pepa /dev/sdb1 none luks

D'altra banda, en el fitxer "/etc/fstab" farem referència a la partició dins del dispositiu "/dev/mapper/pepe" que volguem muntar, així:

/dev/mapper/pepa /mnt/hola ext4 noauto,x-systemd.automount 0 0

NOTA: Les opcions de muntatge "noauto,x-systemd.automount" serveixen per no realitzar el muntatge efectivament fins que l'usuari hi accedeixi per primera vegada, ja dins de la seva sessió, a la carpeta en qüestió. D'aquesta manera s'optimitzen recursos en el sentit que no es té muntat un dispositiu si encara no s'ha utilitzat. Aquest truc també és molt habitual a l'hora de voler muntar sistemes de fitxers remots (com ara NFS), i, de fet, s'hi sol afegir en aquest casos també l'opció *x-systemd.device-timeout=nºs* per indicar el número de segons que el sistema s'esperarà a fer el muntatge correctament abans de deixar-ho còrrer.
NOTA: Si volguéssim utilitzar en "/etc/fstab" l'UUID del dispositiu LUKS en lloc del nom del dispositiu mapejat, per saber aquest UUID haurem d'executar *cryptsetup luksUUID /dev/sdb*

Amb tot l'anterior, en arrencar el sistema, se'ns demanarà la "passpharse" d'accés al dispositiu LUKS

NOTA: Partitions presented using *kpartx* can't be listed in "/etc/fstab" because the effect of *kpartx* does not persist across a reboot. If long-term access is required then the *kpartx* command would be need to be scripted to run at boot time. This can be done, for example, by using a Udev rule to run automatically when the main device is created, so the corresponding devices for the partitions are created too.
NOTA: You can use "/etc/crypttab" for all filesystems and swap devices. However, if you want to encrypt your root partition, then your system's initrd will need cryptography support! You should refer to your distro's docs for more information about this because boot configurations vary. Anyway, the system's BIOS needs to read the boot partition, so that cannot be encrypted.

EXERCICIS (I):

1.-a) Assegura't de tenir una màquina virtual qualsevol amb un segon disc dur buit de 5GB (que pot representar, per exemple un llapis USB) i arrenca-la

> NOTA: Si no tens la possibilitat d'afegir cap disc dur a la màquina virtual, pots fer igualment aquest exercici emprant un dispositiu "loop". Concretament, caldrà que executis les següents comandes per preparar-lo (suposarem que fem servir el dispositiu "loop" /dev/loop0): *sudo fallocate -l 5G /root/elmeudisc && sudo losetup /dev/loop0 /root/elmeudisc*

b) Fes que aquest segon disc dur (o dispositiu "loop", el que hagis triat) estigui compost per una única partició de tipus LUKS Ext4. Per aconseguir-ho, executa les següents comandes (on estem suposant que el dispositiu s'anomena "/dev/loop0" però podria ser perfectament "/dev/sdb" o similar...o fins i tot una partició individual, com per exemple "/dev/sdb2" o similar):

sudo cryptsetup luksFormat /dev/loop0
sudo cryptsetup open /dev/loop0 elmeuusb
sudo mkfs.ext4 /dev/mapper/elmeuusb
sudo cryptsetup close elmeuusb

> NOTA: Cryptsetup també és capaç de treballar directament amb fitxers (encarregant-se ell mateix d'associar-lo a un determinat dispositiu "loop"). És a dir, que en el cas de fer servir un fitxer en lloc d'un segon disc dur, no hagués calgut executar explícitament la comanda *losetup* ja que aquesta és executada internament en fer *sudo cryptsetup luksFormat unfitxer* i *sudo cryptsetup open unfitxer elmeuusb* De totes formes, a l'enunciat de l'exercici indicarem explícitament el dispositiu "loop" per recalcar que estem treballant sempre amb dispositius que hauran de ser particionats i formatejats

c) Comprova els detalls de la capçalera LUKS del dispositiu anterior executant *sudo cryptsetup luksDump /dev/loop0* Concretament, observa els valors i significat de les següents dades: "UUID", "cipher" i "offset" (en la secció "Data segments"), "Key" i "PBKDF" (en la secció "Keyslots"). D'altra banda, si has fet servir un dispositiu "loop", ¿què mostra la comanda *sudo file /root/elmeudisc* ?

d) Obre i munta la partició LUKS per poder treballar-hi amb ella. En concret, executa el següent...:

sudo mkdir /mnt/punt
sudo cryptsetup open /dev/loop0 elmeuusb
sudo mount /dev/mapper/elmeuusb /mnt/punt
sudo chown -R $USER:$USER /mnt/punt (opcionalment, per no treballar com a "root" en el punt de muntatge)

...i tot seguit copia dins de "/mnt/punt" un fitxer qualsevol (que consideris "ultrasecret"). Finalment, executa les següents comandes per "tancar" de nou la partició LUKS

sudo umount /mnt/punt
sudo cryptsetup close elmeuusb

> NOTA: Per curiositat, pots observar, abans de tancar la partició, com es mostra aquesta a la sortida de la comanda *lsblk -f*

e) Prova ara d'executar s*udo mount /dev/loop0 /mnt/punt* (canviant el dispositiu "loop" pel dispositiu de disc físic que estiguis utilitzant, si és el cas). ¿Què passa i per què?

2.-a) Afegeix una segona "passphrase" al dispositiu LUKS. Això ho pots aconseguir executant el següent:

sudo cryptsetup luksAddKey /dev/loop0

b) Afegeix una tercera "passphrase" al dispositiu LUKS, però aquest cop no interactivament sinó a partir del contingut d'un determinat fitxer. Això ho pots aconseguir executant el següent:

echo -n "mipassphase" | sudo tee /root/fitxer-clau.txt
sudo cryptsetup luksAddKey /dev/loop0 /root/fitxer-clau.txt

> NOTA: Podries fer *rm /root/fitxer-clau.txt* però llavors només podràs usar la "passphrase" interactivament,

c) Comprova amb *sudo cryptsetup luksDump /dev/loop0* que a la capçalera LUKS apareixen efectivament tres "KeySlots" ocupats (corresponents a les tres "passphrases" guardades al dispositiu LUKS: l'original, la introduïda a l'apartat a) d'aquest exercici i la introduïda a l'apartat b))

d) Torna a obrir i muntar la partició LUKS per veure el seu contingut desxifrat. Concretament, executa ara les següents comandes, on podràs indicar qualsevol de les tres "passphrases" guardades al dispositiu LUKS (l'original, la introduïda a l'apartat a) d'aquest exercici i la introduïda a l'apartat b))...:

sudo cryptsetup open /dev/loop0 elmeuusb
sudo mount /dev/mapper/elmeuusb /mnt/punt
ls /mnt/punt
sudo umount /mnt/punt
sudo cryptsetup close elmeuusb

e) Torna a obrir i muntar la partició LUKS per veure de nou el seu contingut desxifrat però ara fes-ho de forma no interactiva, fent servir el fitxer-clau creat a l'apartat b). Concretament, executa ara les següents comandes:

sudo cryptsetup open ***--key-file /root/fitxer-clau.txt*** */dev/loop0 elmeuusb*
sudo mount /dev/mapper/elmeuusb /mnt/punt
ls /mnt/punt
sudo umount /mnt/punt
sudo cryptsetup close elmeuusb

f) Esborra ara la "passphrase" introduïda a l'apartat a). Això ho pots aconseguir executant la comanda *sudo cryptsetup luksRemoveKey /dev/loop0* Comprova tot seguit amb *sudo cryptsetup luksDump /dev/loop0* de nou que ara a la capçalera LUKS només queda constància de dues "passphrases" (en dos "slots"): l'original i la introduïda a l'apartat b).

3.-a) Fes una còpia de seguretat de la capçalera LUKS del dispostiu que has fet servir en els exercicis anteriors (en el nostre cas hem suposat que és "/dev/loop0"). Això ho pots aconseguir executant la comanda *sudo cryptsetup luksHeaderBackup /dev/loop0 --header-backup-file cabecera.bak* ¿Quina és la sortida de la comanda *file cabecera.bak* i per què?

b) Ara "destrossa" la capçalera LUKS (que ocupa uns 16MB aproximadament) en el dispositiu per tal d'impedir el seu muntatge. Això ho pots fer així: *sudo dd if=/dev/zero of=/dev/loop0 bs=16M count=1*

c) Comprova, efectivament, que la comanda *sudo cryptsetup open /dev/loop0 elmeuusb* dona error.

d) Restaura la capçalera LUKS en el dispositiu. Això ho pots aconseguir executant la comanda *sudo cryptsetup luksHeaderRestore /dev/loop0 --header-backup-file cabecera.bak*

e) Comprova que ara la comanda *sudo cryptsetup open /dev/loop0 elmeuusb* ja no dona error. Tot seguit, tanca el dispositiu de nou

> NOTA: En el moment de crear el dispositiu LUKS amb *luksFormat*, es pot indicar que la capçalera no es guardi en el propi dispositiu LUKS sinó en un fitxer a banda (el qual es podrà emmagatzemar llavors en qualsevol lloc); això es pot fer indicant el paràmetre *--header cabecera.luks* a la comanda *luksFormat* D'aquesta manera però, el dispositiu LUKS no serà possible ser obert si no es té acceś a aquest fitxer (fent servir, per això, el paràmetre *--header cabecera.luks* a la comanda *open* cada cop).

4.-a) Aconsegueix que el dispositiu LUKS utilitzat als exercicis anteriors s'intenti muntar automàticament en cada arranc del sistema però que per aconseguir això, demani interactivament la "passphrase". És a dir, fes el següent:

*Afegeix la línia següent a l'arxiu "/etc/crypttab":
elmeuusb /root/elmeudisc none luks

NOTA: A la línia anterior estem tenint en compte, com es comentava en una nota anterior, que si, en lloc d'un fitxer de dispositiu "/dev/xxx", com a valor del segon camp de l'arxiu "/etc/crypttab" s'indica la ruta d'un fitxer regular, Cryptsetup automàticament reconeixerà aquest com a dispositiu "loop" (i, per tant, no cal fer res més al respecte)

*Afegeix les línies següents a l'arxiu "/etc/fstab":
/dev/mapper/elmeuusb /mnt/punt ext4 noauto,x-systemd.automount 0 0

aII) Comprova, un cop reiniciada la màquina, que durant el nou arranc se't preguntarà la "passphrase" necessària per muntar i tenir visible el contingut ubicat dins de "/mnt/punt". Introdueix-la correctament per comprovar finalment, un cop hagi finalitzat l'arranc per complet i hagis iniciat sessió, que, efectivament, el punt de muntatge està muntat.

b) Repeteix l'apartat a) anterior però ara fent que el muntatge durant l'arranc del disc xifrat sigui automàtic gràcies a l'ús d'un fitxer-clau (com el generat a l'exercici 2). Per aconseguir això només cal que modifiquis el valor indicat en negreta de l'arxiu "/etc/crypttab" respecte l'apartat a) anterior, així:

*elmeuusb /root/elmeudisc **/root/fitxer-clau.txt** luks*

bII) Comprova, un cop reiniciada la màquina i iniciada la teva sessió de nou, que ara tens muntat i visible el contingut ubicat dins de "/mnt/punt" sense haver fet res

Algú podria pensar que desxifrar automàticament un disc xifrat en arrencar el sistema mitjançant l'ús d'un fitxer-clau elimina l'avantatge de tenir precisament un disc xifrat, que és que un malfactor pugui accedir a les nostres dades directament. Això és cert, efectivament, en el supòsit que, per exemple, el malfactor ens robés la màquina sencera (un portàtil, per exemple). Però l'ús de fitxers-clau no se sol emprar en aquests supòsits, sinó en màquines ubicades en centre de dades o entorns segurs on no sigui tan fàcil extreure una màquina sencera però sí, en canvi, un disc dur individual. En aquests casos, el que se sol fer, precisament, és no guardar el fitxer-clau en el mateix disc dur (com hem fet a l'exercici anterior) sinó emmagatzemar-ho en un altre lloc, de forma que si el malfactor vol utilitzar aquest disc dur robat, veurà que no el podrà desxifrar perquè el fitxer-clau no estarà disponible. Respecte quins poden ser aquests altres magatzems on allotjar els fitxers-clau dels dispositius LUKS, ens trobem amb diferents possibilitats, com ara l'ús del chip TPM (mitjançant *systemd-cryptenroll* o similar) o bé solucions en xarxa com ara la combinació Clevis/Tang (https://github.com/latchset), útil per LANs, o bé solucions a nivell d'Internet (genèricament anomenades "Key Management Service") i proporcionades per diferents empreses com ara Amazon, Google o Microsoft ; d'algunes d'aquestes solucions en parlarem properament.

TO BE CONTINUED