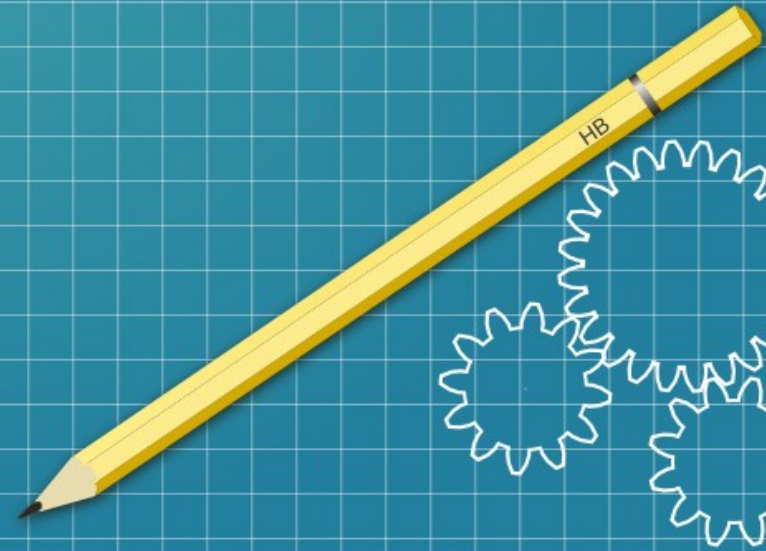
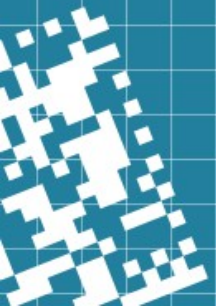


Criptografía e Cifrado

Exemplos Práticos

Cifrado de Carpetas e Discos



Introdución



Si queremos protexer a privacidade da nosa información non basta co control de acceso ao dispositivo, nin a autorización. Nin siquiera o borrado dos datos impide necesariamente que terceiras persoas podan acceder a eles (**Si queremos que ninguén poda acceder a información borrada é necesario un borrado seguro do dispositivo**)


O único xeito realmente efectivo de garantir a confidencialidade da información almacenada é o seu cifrado. Mediante o cifrado garantimos a confidencialidade incluso en caso de roubo dos dispositivos de almacenamento.

O punto débil é unicamente (aparte de posibles debilidades no sistema de cifrado) o usuario, que pode ser forzado a comunicar a chave para descifrar a información.

Existen multitude de sistemas que nos permiten cifrar a información. Algúns dos máis coñecidos son:

- **TrueCrypt:** Multiplataforma. No seu día un dos máis utilizados, pero ***está discontinuado e presenta vulnerabilidades nos seus sistemas de cifrado.***
- **VeraCrypt:** Multiplataforma. É o sucesor de TrueCrypt.
- **EFS (Encrypted Filesystem):** E o sistema nativo en Windows, permite o cifrado de arquivos e carpetas
- **BitLocker:** E o sistema de cifrado de dispositivos nativo de Windows
- **encFS:** Permite o cifrado de carpetas en Linux de xeito moi simple, pero presenta algunhas vulnerabilidades.
- **ecryptFS:** Permite cifrado de carpetas en Linux.
- **dm-crypt + LUKS (Linux Unified Keys Setup):** Permite o cifrado de dispositivo en Linux. Ten certo grao de compatibilidade con VeraCrypt.

Borrado Seguro



- Cando eliminamos información dun dispositivo mediante as ferramentas estándar do sistema de arquivos, simplemente estamos indicando de diferentes xeitos que os datos xa non se consideran válidos, pero permanecen.
- O xeito máis rápido de facer un borrado é a posta a 0 de todo o espazo (**dd if=/dev/zero of=/dev/dispositivo bs=4k**). Pero isto non asegura que os datos non sexan recuperables (https://en.wikipedia.org/wiki/Data_remanence), particularmente nos SSD e memorias flash as distintas capas de firmware que xestionan a escritura fan relativamente fácil a recuperación e non sempre nos podemos fiar das operacións **TRIM** que borran físicamente as celdas borradas.

O Departamento de Defensa de USA, establece o método de eliminación de datos en 3 pasos DoD DoD 5220.22-M:

- 1) Sobrescribir todo con 0 e verificar el disco
 - 2) Sobrescribir todo con 1 e verificar el disco
 - 3) Sobrescribir todo con datos aleatorios e verificar o disco.
- As controladoras de disco modernas teñen comandos de borrado, que poden ser apropiadas no caso dos SSD. En Linux podemos xestionalos mediante a utilidade **hdparm**:
 - Verificamos que o disco non esté bloqueado (**hdparm -l /dev/sdX** e nos aseguramos que aparece “not frozen”, si aparece “frozen” debemos reiniciar)
 - Debemos poñer unha password (**hdparm --user-master u --security-set-pass password /dev/sdX**)
 - Verificamos con **hdparm -l** que aparece “enabled” en lugar de “not enabled” na password e “Security level high”. Tamén veremos si que tipos de borrado soporta o dispositivo “SECURITY ERASE” / “ENHANCED SECURITY ERASE”.
 - SECURITY ERASE cambiará a chave interna de cifrado dos datos da controladora facendo que non sexan accesibles.
hdparm --user-master u --security-erase-enhanced password /dev/sdX
 - ENHANCED SECURITY ERASE ademais de cambiar a chave escribirá un patrón de datos no disco.
hdparm --user-master u --security-erase password /dev/sdX

Cifrado a Nivel de Archivo



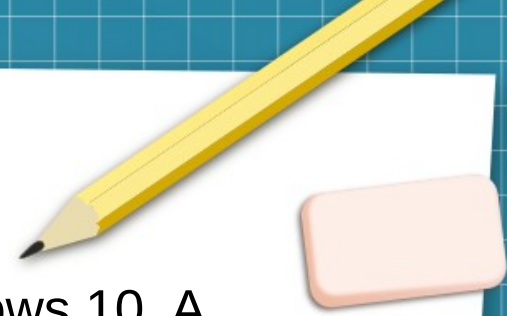
Mediante o cifrado a nivel de arquivo protexemos arquivos concretos cifrando o seu contido. Habitualmente os sistemas proporcionan métodos de cifrado / descifrado transparente para o seu propietario. Algúns dos máis habituais son:

- Windows
 - **EFS (Encrypted File System)**: incluída a partir da versión 3.0 de NTFS salvo nas versións Home. Cifra a partir dunha chave simétrica xerada automaticamente a partir da password do usuario.
- GNU/Linux
 - **ecryptFS**: Permite cifrar o contido dunha carpeta a partir dunha chave elexida polo usuario.
 - **encFS**: Moito máis simple, pero menos seguro que ecryptFS

O cifrado a nivel de arquivo permite que un posible atacante coñeza o tipo de sistema de arquivos instalado, a estrutura de carpetas, os usuarios dos sistemas... etc. En cambio, permite:

- Facer backup dos arquivos cifrados sen necesidade de descifralos
- Gran flexibilidade de xestión de sistemas de cifrado e chaves de cifrado. Cada usuario pode ter o seu propio cifrado.
- Permite redimensionar as particións ou sistema de arquivos con total facilidade
- Únicamente existe a sobrecarga de procesamento que implica cifrar e descifrar cando se accede a carpeta cifrada
- Podemos enviar a terceiras persoas o arquivo cifrado sin ter que descifralo e volvelo a cifrar

Windows EFS



EFS é o sistema de cifrado a nivel de arquivos propio de Windows 10. A diferencia de Bitlocker que é o subsistema de Windows que ofrece cifrado a nivel de bloques. É moi simple e utiliza a propia contrasinal do usuario para o cifrado. O veremos cun exemplo:

- 1) Crear na carpeta de Documentos unha nova carpeta “Privados”
- 2) Configurar o sistema para que os documentos almacenados en “Privados” se garden cifrados.
- 3) Gardar un documento de texto “confidencial.txt” co contido “hola mundo !”
- 4) Pechar sesión e iniciar sesión con outro usuario, por exemplo, administrador. Comprobar que non pode ver o contido dese ficheiro.

ecryptFS - I

- **ecryptFS** é un sistema de cifrado a nivel de arquivo, a diferenza de dm-crypt ou BitLocker, que son sistemas de cifrado a nivel de bloque.
- **ecryptFS** consiste nun sistema de arquivos criptográfico “apilado” enriba dun sistema de ficheiros físico. Cando escribimos nese sistema de ficheiros a información se almacena cifrada no disco físico, cando lemos, se descifra automaticamente a información almacenada. Polo tanto, se utilizan dous carpetas: unha carpeta oculta (*.nomedecarpeta*) que almacenará realmente os datos e a carpeta de acceso (*nomedecarpeta*) onde podemos acceder aos datos descifrados. Cando montamos o sistema cifrado, **ecryptfs** se encargará de cifrar os datos que escribamos en *nomedecarpeta* e gardalos en *.nomedecarpeta*. Cando leamos datos, **ecryptfs** se encarga de leelos de *.nomedecarpeta* e de descifralos.
- **Ecryptfs** garda nos ficheiros cifrados a información necesaria para poder ser descifrados en calquera sitio si coñecemos o algoritmo de cifrado e a password empregada.
- **ecryptFS** so se pode establecer sobre directorios baleiros ou con datos xa propios de **ecryptFS** (e dos que coñecemos os parámetros de cifrado e a password)
- Debemos ter en conta que os ficheiros “sparse” ocuparán cifrados toda a extensión, e incluso máis (os ficheiros ‘sparse’ son ficheiros nos que as partes que están sen escribir non ocupan disco)

O procedemento é mais complexo que o sistema EFS de Windows, pero moito máis flexible, aínda que existen scripts que nos facilitarán moito o traballo. Os pasos a seguir serían os seguintes:

1) Creamos a carpeta que queremos cifrar e a carpeta oculta correspondente. Recordemos que os datos se gardarán físicamente na carpeta oculta (realmente poden ser dúas carpetas calquera).

2) **mount -t ecryptfs carpeta_cifrada carpeta_de_acceso** – Isto nos amosará un asistente no que nos preguntará:

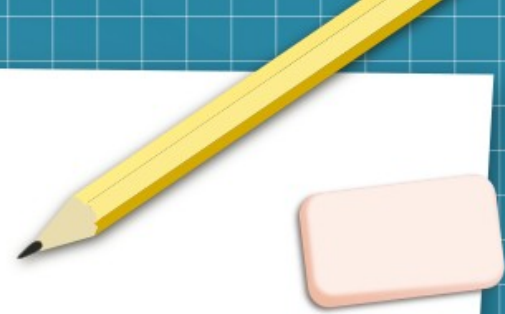
- 1) Chave de cifrado: Podemos elixir entre **tspi** que fará uso do módulo TPM si existe e está configurado, ou **passphrase**
- 2) O algoritmo de cifrado que queremos utilizar. O habitual e preferido é aes de 32 bits
- 3) Si queremos permitir ter ficheiros non cifrados dentro da carpeta cifrada (enable plaintext passthrough)
- 4) Si queremos cifrar tamén os nomes dos arquivos, non so o seu contido

O sistema montará a carpeta_cifrada enriba de carpeta_de_acceso. Todo o que escribamos en carpeta_de_acceso se cifrará e se almacenará na carpeta_cifrada. Cando leemos na carpeta_de_acceso se leerán os datos de carpeta_cifrada e se descifrarán.

Este procedemento é incómodo, xa que obriga a contestar numerosas preguntas, e sobre todo a introducir a password de cifrado cada vez que a montamos. Si non queremos responder a tanta pregunta, sempre podemos pasarlle as opcións ao comando mount como opcións: **mount -t ecryptfs carpeta_cifrada carpeta_de_acceso -o ecryptfs_ciphther=aes,ecryptfs_key_bytes=32** (Podemos montalo unha vez contestando as preguntas e observar os parámetros co comando mount.).

Tamén podemos engadir unha liña ao fstab que indique as opcións desexadas. Si queremos que o usuario poda montar e desmontar a carpeta cifrada a súa vontade non chega con especificar “user” no fstab. E necesario o uso dun script chamado **mount.ecryptfs_private** e **umount.ecryptfs_private** si queremos evitar a opción de “sudo”. Estes comandos non utilizan directamente a password de cifrado, se non que utilizan unha password para descifrala. O almacenar a password de cifrado no disco nos da a vantaxe de poder cambiar a password de acceso cando desexemos mediante o comando **ecryptfs-rewrap-passphrase**

ecryptFS - II



O procedemento para poder montar con **mount.ecryptfs_private** é o seguinte:

- **touch .ecryptfs/wrapped-passphrase** # Sitio estándar para gardar a password de cifrado codificada.
- **wrap=\$(head -c 48 /dev/random | base64)** # Obtemos un número de 48 díxitos aleatorios e o pasamos a base64. Será a password de cifrado
- **ecryptfs-wrap-passphrase .ecryptfs/wrapped-passphrase \$wrap abc123.** # Ciframos a password de cifrado coa contraseña que queramos. A podemos cambiar con **ecryptfs-unwrap-passphrase**, ou descifrala con **ecryptfs-unwrap-passphrase**
- **ecryptfs-insert-wrapped-passphrase-into-keyring .ecryptfs/wrapped-passphrase abc123.** # Insertamos no anel de chaves da memoria a password de cifrado (keyctl list @u). Xa non teremos que facelo máis si non cambia ata o proximo reinicio. Para insertar directamente a chave de cifrado poderíamos utilizar **ecryptfs_add_passphrase**
- **echo 5d7c563d8dd06fa1 > .ecryptfs/secret.sig** # Debemos gardar a signatura que nos indica o paso anterior neste ficheiro corresponde co parámetro de mount **ecryptfs_sig=**
- **echo \$HOME/.secret \$HOME/secret ecryptfs > .ecryptfs/secret.conf** # Isto é como un "fstab" para ecryptfs. Lista as montaxes permitidas para o usuario. Deben estar no fstab coa opción "user".
- **lsr/sbin/mount.ecryptfs_private secret** # Montamos
- **lsr/sbin/umount.ecryptfs_private secret** # Desmontamos

Todos estes pasos se poden facer de xeito "automático" sacrificando algunha cousa (correxible posteriormente) como que a carpeta cifrada debe chamarse **.Private** e a de acceso **Private** facendo uso do comando **ecryptfs-setup-private**. E posible automatizar todo aínda máis mediante o script **ecryptfs-migrate-home -u username** (e importante que o usuario inicie sesión antes de reiniciar, unha vez todo listo se pode borrar o backup de **/home/username.xxxxxxxxxxxxxx**).

Si queremos cambiar logo o nome da carpeta cifrada, bastará con renomear a carpeta **Private** co nome desexado e poñer o seu path completo e nome en **~/ecryptfs/Private.mnt**

Si queremos facer a montaxe no inicio de modo automático, deberíamos utilizar como password de acceso (que non de cifrado) a nosa contrasinal de usuario e utilizar o módulo PAM (Plugin Authenticate Module) **pam_ecryptfs**. Na carpeta **.ecryptfs** deben existir os ficheiros **auto-mount** e **auto-umount** ademais de **wrapped-passphrase** (man **pam_ecryptfs**)

encFS



- EncFS é unha ferramenta de cifrado a nivel de arquivo moi simple que traballa mediante o driver **FUSE** (Filesystem in User Space), polo que *un usuario sin privilexios de administración pode crear carpetas cifradas con total facilidade*, en cambio teñen un rendimento menor que as ferramentas que traballan en espazo do Kernel.
- As diferencias fundamentais con ecryptFS son:
 - Funciona en espazo de usuario en lugar de en espazo de kernel.
 - Os datos de cifrado están centralizados (*.encfs6.xml*), non nas cabeceiras dos arquivos como en ecryptFS.
- O uso é moi simple: **encfs /ruta_completa/carpeta_cifrada /ruta_completa/carpeta_acceso**, para desmontar a carpeta utilizaremos a utilidade de fuse **fusermount -u /ruta_completa/carpeta_acceso**
- Mediante **encfsctl** podemos cambiar a contrasinal usada no cifrado (realmente é a contrasinal de cifrado para a chave que se utiliza realmente para cifrar/descifrar)
- Mediante **pam_mount** (*apt install libpam-mount*) podemos montar as carpetas automaticamente no inicio de sesión, sempre que utilizemos a nosa contrasinal como password da chave de cifrado. Deberemos especificar no ficheiro **/etc/security/pam_mount.conf.xml** algo así (*man pam_mount.conf*):

```
<volume user="usuario" fstype="fuse" path="encfs#/home/%(USER)/.ofye" mountpoint="/home/%(USER)/pd" options="nonempty" />
```


Cifrado de dispositivos



O cifrado de dispositivos ou cifrado a nivel de bloque é moito máis seguro que o cifrado a nivel de carpeta/arquivo xa que absolutamente todo o contido do disco é cifrado (incluíndo o sistema de arquivos e incluso a táboa de particións), deste xeito, un posible atacante nin siquiera poden coñecer o tipo de sistema de ficheiros utilizado, a estrutura de directorios ou o sistema operativo utilizado.

Algunhas desvantaxas son:

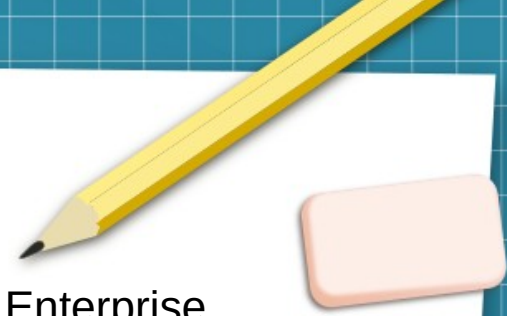
- O tamaño do sistema de ficheiros é difícil de alterar unha vez cifrado.
- É difícil cambiar o sistema de cifrado ou as chaves de cifrado
- Todo o disco está cifrado coa mesma chave
- Moitas aplicacións (como os backups incrementais) necesitan acceso aos datos descifrados
- Existe unha sobrecarga do proceso no acceso a calquera contido do disco, xa que todo está cifrado.
- Si queremos transmitir un ficheiro ou gardalo en outro sitio de xeito seguro é necesario cifralo de novo.

A perda da chave de cifrado supón a perda absoluta de toda a información almacenada, sendo practicamente imposible a súa recuperación.

Existen varias solucións que ofrecen cifrado, pero en Windows se ofrece nativamente **BitLocker**, mentres que en GNU/Linux o máis común é o uso de **cryptsetup + LUKS**.

Linux presenta unha gran vantaxe debido o xeito en que xestiona os dispositivos de bloques. Mediante o módulo **loop**, é capaz de tratar como un dispositivo de bloques un ficheiro plano, o que de facto fai innecesario o cifraxe a nivel de arquivo. Ademais combinando con sistemas de volumes (LVM) ofrece unha flexibilidade enorme.

Bitlocker



- **BitLocker Drive Encryption** so está dispoñible en Windows 10 Pro e Enterprise
- Windows 10 Home ten unha ferramenta máis simple denominada “Device Encryption” para plataformas con UEFI, TPM e “Modern Standby”
- O mellor resultado se consegue si se dispón dun módulo de cifrado hardware TPM (Trusted Platform Module)
- E posible usar BitLocker para o disco do sistema sin TPM habilitando o cifrado por software
 - **gpedit.msc** → **Plantillas Administrativas** → **Componentes de Windows** → **Cifrado de Unidad Bitlocker** → **Unidades del Sistema Operativo**
 - Requerir autenticación adicional al iniciar
 - Permitir BitLocker sin un TPM compatible (requiere contraseña o clave de inicio en una unidad flash USB)
- O cifrado sin TPM require dunha contrasinal, o cifrado con TPM non.
- O sistema debe utilizar NTFS

dm-crypt + LUKS

Consideracións iniciais



- dm-crypt crea un novo dispositivo (device-mapper) no que podemos leer en claro datos que se atopan cifrados nouro dispositivo e gardar datos cifrados.
- Si queremos cifrar un dispositivo de xeito seguro, o primeiro que deberíamos facer é un borrado seguro do mesmo, de modo que non ofrezan pistas sobre o cifrado e non facilitemos acceso a información que debería ser privada accidentalmente.
 - `cryptsetup open --type plain -d /dev/urandom /dev/<block-device> to_be_wiped`: Creamos un dispositivo cifrado `/dev/mapper/to_be_wiped` cunha chave aleatoria
 - `dd if=/dev/zero of=/dev/mapper/to_be_wiped status=progress bs=4M`: Escribimos ceros que serán almacenados cifrados en `/dev/block-device`
 - `cryptsetup close to_be_wiped`: Pechamos o dispositivo, o que eliminará o `/dev/mapper/to_be_wiped`
- Si non fixemos o paso anterior e xa temos o dispositivo cifrado habilitado, unha solución é crear un ficheiro ao azar que ocupe todo o disco: `dd if=/dev/random of=filename.txt status=progress bs=4M` e logo forzar o vaciado da caché (sync) e eliminar o ficheiro (`rm filename.txt`)
- Si queremos eliminar os datos dun dispositivo cifrado, basta con eliminar a cabeceira de cifrado. Sin eso é practicamente imposible recuperar o contido.
 - `cryptsetup erase device` – Eliminamos as claves de cifrado
 - `cryptsetup luksDump device` – Nos aseguramos que estan borradas
 - `wipefs -a device` – Eliminamos a cabeceira

dm-crypt + LUKS

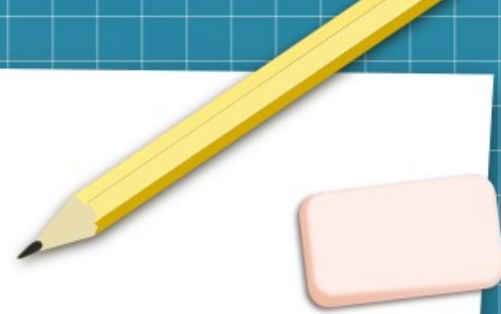
cifrado do dispositivo



- `cryptsetup` é o comando que permite configurar o cifrado mediante `dm-crypt` incluído no kernel. Permite xestionar as chaves de cifrado mediante `luks`, `luks1`, `luks2`, `plain`, `loopaes` ou `tcrypt` para compatibilidade con TrueCrypt / VeraCrypt.
- LUKS (Linux Unified Key Setup) permite almacenar toda a información necesaria para o cifrado/descifrado en disco permitindo ao usuario olvidarse da súa xestión e utilizar varias chaves diferentes para o acceso.
- Para cifrar un dispositivo con `dm-crypt` + LUKS (`luks2`):
 - **`cryptsetup luksFormat /dev/dispositivo`**: Crea a cabeceira LUKS cos datos sobre o cifrado. Por defecto será `--type luks2 --cipher aes-xts-plain64 --hash sha256 --iter-time 2000 --key-size 256 --pbkdf argon2id --use-urandom --verify-passphrase` (`cryptsetup --help`)
 - Podemos indicar o tamaño do sector desexado. Si o noso dispositivo é de Advanced Format (AF/4Kn) de 4k deberíamos indicar ademais `--sector-size 4096` como parámetro para mellorar o rendimento. Tamén podemos indicar ao final un nome de arquivo no que almacenaremos a chave de cifrado, de xeito que non sexa necesario escribila cada vez.
 - **`cryptsetup open /dev/dispositivo dispositivo-acceso`**: Creamos o dispositivo de acceso `/dev/mapper/dispositivo-acceso` que fará de capa de cifrado co dispositivo real `/dev/dispositivo`.
 - Si especificamos a chave nun arquivo, debemos engadir o parámetro `--key-file` arquivo
 - É posible engadir máis ficheiros de chaves mediante `cryptsetup luksAddkey` (si a chave de cifrado anterior está nun arquivo, debemos indicalo co parámetro `-d`) e examinar as chaves cargadas mediante `cryptsetup luksDump`
 - Mediante os comandos `luksErase` (eliminar todas as chaves) `luksRemoveKey` (eliminar unha chave) e `luksKillSlot` (eliminar un slot) podemos eliminar chaves de cifrado
 - **`cryptsetup close dispositivo-acceso`**: Eliminamos a capa de cifrado, e polo tanto `/dev/mapper/dispositivo-acceso`
- **`cryptsetup luksHeaderBackup /dev/device --header-backup-file /mnt/backup/file.img`**: Permite facer un backup da cabeceira LUKS
- **`cryptsetup -v --header /mnt/backup/file.img open /dev/device test`**: Permite usar a cabeceira LUKS para probar si é correcta
- **`cryptsetup luksHeaderRestore /dev/device --header-backup-file ./mnt/backup/file.img`**: Permite restaurar a cabeceira LUKS
- Para automontar os dispositivos `dm-crypt` podemos utilizar `pam_mount`

dm-crypt + LUKS

cifrando root e swap



- E posible cifrar a partición raíz do sistema e o ficheiro de swap.
- Particularmente é importante o cifrado da partición de swap, xa que pode almacenar o resto do contido da RAM do ordenador e revelar información confidencial.
- Aínda que é posible preparalo manualmente, o máis sinxelo é realizar a instalación xa sobre o sistema cifrado. Manualmente poderíamos utilizar **systemd-cryptsetup-generator**, que crea as unidades systemd necesarias a partir do `/etc/crypttab`
- O crypttab xerado é similar a este: **`vda5_crypt UUID=7cfcc7c2-383e-4535-8b31-bcc05b2ffd08 none luks[,discard]`**
- Para arrancar poñendo unha chave USB concreta:
 - Supoñamos que o dispositivo USB é **`USBDEV`**, e que usaremos a partición 1
 - Supoñamos que o dispositivo cifrado é **`C_DEV`** con UUID **`C_DEV_UUID`**, e que o dispositivo de acceso **`A_DEV`**
 - **`mount ${USBDEV}1 /root/usbkey-${A_DEV}`**
 - **`mount dd if=/dev/urandom of=/root/usbkey-${A_DEV}/${A_DEV} bs=1M count=4`**: Montamos o usb e creamos unha chave de acceso aleatoria
 - **`cryptsetup luksAddKey ${C_DEV} /root/usbkey-${A_DEV}/${A_DEV}`** :O engadimos a LUKS
 - **`umount ${USBDEV}1`**: Desmontamos a chave USB
 - **`blkid`**: Averiguamos o UUID de `USBDEV`1. Supoñamos que é `USB_UUID`)
 - Modificamos `/etc/crypttab` de xeito que quede:
`$(A_DEV) UUID=$(C_DEV_UUID) /dev/disk/by-uuid/${USB_UUID}:${A_DEV} luks,keysript=/lib/cryptsetup/scripts/passdev`



This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License. It makes use of the works of Mateus Machado Luna.

