# LUKS

Linux Unified Key Setup, est le standard associé au noyau Linux pour le chiffrement de disque.

# Les commandes LUKS

## Initialiser la partition

`sdX1` à remplacer par la partition à formater. Exemple sur une partition libre /dev/sdc1 :

```
  sudo cryptsetup luksFormat  /dev/sdX1
```

On invoque cette commande pour formater la partition au type LUKS (initialiser la partition LUKS et définir la clé initiale). Le chiffrement sera de type AES avec un algorithme de hachage SHA256. Vous allez taper votre phrase de déchiffrement qui va permettre de créer un conteneur standard chiffré à l'aide de votre phrase.

Le conteneur chiffré de manière standard va stocker la clé de chiffrement maître qui servira à ouvrir votre volume chiffré. Il est possible d'ajouter jusqu'à 8 clefs supplémentaires dans des "slots", qui déverrouillent l'accès à la clef maître.

## Ajouter une clé

Pour ajouter une clé d'accès au conteneur chiffré précédent :

```
sudo cryptsetup luksAddKey /dev/sdX1 /root/keyfile
```

## Supprimer une clé

Pour révoquer une clé contenue dans un "slot" :

```
sudo cryptsetup luksKillSlot /dev/sdX1 <numero_de_slot>
```

## Etat de la partition

Pour voir l'état du conteneur chiffré et les "slots" utilisés :

```
sudo cryptsetup luksDump </dev/sdX1>
```

## Montage manuel et formatage

Ouverture et formatage en ext3 de la partition chiffrée. L'appellation du volume est ici **home**[1)].

```
sudo cryptsetup luksOpen /dev/sdX1 <mapperName>

# a faire seulement la première fois :
sudo mkfs.ext4 /dev/mapper/<mapperName>
```

puis montage de la partition :

```
sudo mount -v /dev/mapper/<mapperName> </folder/location/>
```

puis démontage et fermeture du volume chiffré :

```
sudo umount /mnt
sudo cryptsetup luksClose home
```

# Mise en place de LUKS

> Afin de protéger au mieux vos données personnelles, il peut être nécessaire de chiffrer vos partitions utilisateur. En effet, si via le système il est impossible d'accéder aux fichiers qui ne vous appartiennent pas, un simple passage sur un livecd permet d'accéder à n'importe quel fichier de votre système. Le chiffrement de partition permet d'éviter ça. Ubuntu intègre en standard les outils nécessaires à une gestion simple de votre sécurité.

source : https://doc.ubuntu-fr.org/cryptsetup

source : https://www.val-r.fr/geek/os/linux/creer-et-utiliser-une-partition-chiffree-avec-luks-sous-linux/

La mise en place d'une partition chiffré se résume en 5 étapes :

- 1/ Installation & Préparation
- 3/ Création de la partition
- 4/ Chiffrement de la partition
- 5/ Utilisation de la partition chiffrée
  - Déverrouillage
  - Montage
  - Démontage
  - Verrouillage

# 1/ Installation & Préparation

Tout d'abord il est nécessaire d'installer cryptsetup.

```
sudo apt-get install cryptsetup
```

La commande `lsblk` permet de lister les disques et les partitions existantes :

```
NAME            MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINT
sda               8:0    0   20G  0 disk
├─sda1            8:1    0   19G  0 part  /
└─sda5            8:5    0  975M  0 part  [SWAP]
sdb□□□□ 8:16□0□10G□ 0 disk
```

```
└─sdb1          8:17   0   10G  0 part  /mnt/media
sdc             8:32   0 □50G  0 disk□□□□□# Disque a utiliser
```

Dans notre exemple ci-dessus nous avons 3 disques dur avec des partitions sur le serveur :

- Le disque **`sda`** de 20Go
  - Partition **`sda1`** qui monté à la racine du serveur
  - Partition **`sda5`** qui est du SWAP pour le serveur

- Le disque **`sdb`** de 10Go
  - Partition **`sdb1`** qui est monté sur le dossier media.

- Le disque **`sdc`** de 50Go sans partition existante

Nous allons utiliser le disque `sdc` pour créer la partition chiffrés.

# 3/ Créer une partition

Pour créer la partition, nous allons tout simplement utiliser l'outil Linux `fdisk` :

```
sudo fdisk /dev/sdc

# n   ajouter une nouvelle partition
Commande (m pour l''aide) : n

Type de partition
   p   primaire (0 primaire, 0 étendue, 4 libre)
   e   étendue (conteneur pour partitions logiques)
Sélectionnez (p par défaut) : p
Numéro de partition (1-4, 1 par défaut) :
Premier secteur (2048-105456767, 2048 par défaut) :
Dernier secteur, +/-secteurs ou +/-taille{K,M,G,T,P} (2048-105456767, 105456767 par défaut) :

Une nouvelle partition 1 de type « Linux » et de taille 50,3 GiB a été créée.

# w   écrire la table sur le disque et quitter
Commande (m pour l''aide) : w

La table de partitions a été altérée.
Appel d''ioctl() pour relire la table de partitions.
Synchronisation des disques.
```

Si nous relançons la commande lsblk nous pouvons voir apparaître la nouvelle partition :

```
NAME           MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINT
sda              8:0    0   20G  0 disk
├─sda1           8:1    0   19G  0 part  /
└─sda5           8:5    0  975M  0 part  [SWAP]
sdb□□□□ 8:16□0□10G□ 0 disk
└─sdb1           8:17   0   10G  0 part  /mnt/media
sdc              8:32   0 □50G  0 disk
└─sdc1           8:33   0 □50G  0 part□□□□# Partition créée
```

# 4/ Chiffrement de la partition

Pour créer une partition LUKS il suffit de lancer la commande suivante :

```
cryptsetup --verbose luksFormat --verify-passphrase /dev/sdc1

WARNING!
========
Cette action écrasera définitivement les données sur /dev/sdc.

Are you sure? (Type 'yes' in capital letters): YES
Saisissez la phrase secrète pour /dev/sdc :
Vérifiez la phrase secrète :
Emplacement de clef 0 créé.
Opération réussie.
```

# 5/ Utilisation de la partition chiffrée

## Déverrouillage

Avant d'utiliser la nouvelle partition, il est nécessaire *d'ouvrir* la partition chiffrée ***(et il faudra le faire à chaque démarrage ou après la fermeture de la partition chiffrée)***.

```
cryptsetup -v luksOpen /dev/sdc1 maPartition
Saisissez la phrase secrète pour /dev/sdc1 :
Emplacement de clé 0 déverrouillé.
Opération réussie.
```

Pour vérifier que la partition a bien été déverrouiller utiliser la commande `lsblk` :

```
NAME           MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINT
sda              8:0    0   20G  0 disk
├─sda1           8:1    0   19G  0 part  /
└─sda5           8:5    0  975M  0 part  [SWAP]
sdb□□□□ 8:16□0□10G□ 0 disk
└─sdb1           8:17   0   10G  0 part  /mnt/media
sdc              8:32   0 □50G  0 disk
└─sdc1           8:33   0 □50G  0 part
  └─maPartition 254:1   0 □50G  0 crypt□□□□# Partition Déverrouillé
```

> Lors de la création d'une partition chiffré il est nécessaire de réaliser un formatage de la partition avant sont utilisation. Dans le cas contraire la partition sera corrompu.

Pour formater la partition déchiffré **(a faire seulement la première fois)** :

```
 sudo mkfs.ext4 /dev/mapper/maPartition
```

## Montage

Maintenant que la partition est déverrouillé, pour l'utiliser il faut la monter avec un dossier.

> **Attention !**
> A présent, pour accéder au contenu de la partition chiffrée /dev/sdX1, il **ne faut pas** utiliser /dev/sdX1 mais le *mapper* créé lors de l'ouverture de la partition chiffrée :
> **/dev/mapper/monNomDeVolume**
> (le nom du *mapper* a été spécifié lors de l'*ouverture* avec *cryptsetup luksOpen /dev/sdX1* ***monNomDeVolume***).

```
# Création du point de montage
mkdir /mnt/monDossier

# Montage du mapper
```

```
mount -v /dev/mapper/maPartition /mnt/monDossier
```

Maintenant vous pouvez utiliser votre partition comme vous le voulez via le dossier /mnt/monDossier.

## Démontage

Avant de verrouiller à nouveau la partition il est nécessaire d'enlever le point de montage.

```
# Démontage du mapper
umount -v /dev/mapper/monVolume

# ou
umount -v /mnt/monVolume
```

## Verrouillage

```
cryptsetup -v luksClose maPartition
```

# Déverrouillage Automatique Partition Principale

> Pour réaliser ce tuto il est nécessaire d'avoir au préalable créé la partition principale chiffrée avec LUKS.
>
> Voir Partitionnement des disques + Chiffrement (avancé)

## Le but de ce tuto est de pouvoir déverrouiller la partition principale d'un serveur avec une clé USB. La clé USB contiendra un fichier qui correspondra à une clé accepté par LUKS.

Pour ce faire nous allons modifier l'`initramfs`. Petite explication : L'initramfs est un système de fichiers temporaire utilisé pendant le processus de démarrage du système. L'initramfs contient les modules du noyau et les scripts nécessaires pour monter le système de fichiers racine réel.

L'automatisation du déverrouillage d'une partition chiffrée au démarrage peut-être mis en place en suivant les étapes suivantes :

- 1/ Création d'un fichier de clé random
- 2/ Ajouter le fichier de clé à la partition LUKS
- 3/ Créer le script de déverrouillage
- 4/ Activer les modules attendu
- 5/ Recompiler l'init

# 1/ Création d'un fichier de clé

la commande suivante créera un fichier au contenu aléatoire d'une taille de 4096 bits (mieux qu'un mot de passe de 20/30 caractères....). Vous pouvez utiliser n'importe quel fichier comme fichier clé, mais je pense qu'un fichier de 4kb avec un contenu aléatoire convient bien.

```
sudo dd if=/dev/urandom of=/root/keyfile bs=1024 count=4
```

Comme vous pouvez le voir avec l'arrtibut **`of`** , le fichier sera généré dans le dossier **`/root`** avec le nom **`keyfile`**.

Avant d'utiliser la nouvelle clé, rendre le fichier clé accessible en lecture seule à root

```
sudo chmod 0400 /root/keyfile
```

> Cela rendra le fichier clé lisible uniquement par root. Si quelqu'un accède à ce fichier clé, vous avez de toute façon un problème plus important sur votre serveur.

> Une autre solution consiste à attribuer à root:root le droit d'accès au fichier clé souhaité et à le placer dans le dossier /root.

# 2/ Ajouter le fichier à LUKS

Les dispositifs LUKS/dm_crypt peuvent contenir jusqu'à 10 fichiers clés/mots de passe différents. Ainsi, en plus du mot de passe déjà configuré, nous allons ajouter ce fichier clé comme méthode d'autorisation supplémentaire.

```
sudo cryptsetup luksAddKey /dev/sdX /root/keyfile
```

Il vous sera d'abord demandé d'entrer un mot de passe (existant) pour déverrouiller le lecteur.

# 3/ Préparer le support USB

Tout d'abord nous allons identifier notre support USB pour le formater dans un format compatible avec un noyau linux .

Pour lister les disques nous allons utiliser la commande `fdisk -l` ce qui donnera :

```
...
Disque /dev/sdb : 29,3 GiB, 31457280000 octets, 61440000 secteurs
Disk model: PHILIPS USB
Unités : secteur de 1 × 512 = 512 octets
Taille de secteur (logique / physique) : 512 octets / 512 octets
taille d'E/S (minimale / optimale) : 512 octets / 512 octets
Type d'étiquette de disque : dos
Identifiant de disque : 0x128faffe

Périphérique Amorçage Début      Fin Secteurs Taille Id Type
/dev/sdb1    *           32 61439999 61439968  29,3G  b W95 FAT32
```

```
...
```

> Si le périphérique n'est pas considéré comme "Amorçage", ce n'est pas grave.

Ou avec la commande `lsblk -f` :

```
...
sdb
└─sdb1      vfat     FAT32 PHILIPS UFD 1872-8C67                              29,3G     0%
/media/usbkey/PHILIPS UFD
...
```

Notre clé USB doit être au format `ext4` qui est nativement compatible avec un noyau Linux contrairement au format `FAT32`.

## Formater la clé USB

La commande pour formater la clé USB est la suivante : `fidsk /dev/sdb`
Cette commande permet dans l'ordre d'exécution de :

- `d` Supprimer la partition 1
- `n` Créer une nouvelle partition de type primaire (Tout laisser par défaut)
- `w` Enregistrer et appliquer les modifications

```
root@server: fdisk /dev/sdb

Bienvenue dans fdisk (util-linux 2.37.2).
Les modifications resteront en mémoire jusqu'à écriture.
Soyez prudent avant d'utiliser la commande d'écriture.


Commande (m pour l'aide) : d
Partition 1 sélectionnée
La partition 1 a été supprimée.

Commande (m pour l'aide) : n
Type de partition
   p   primaire (0 primary, 0 extended, 4 free)
   e   étendue (conteneur pour partitions logiques)
Sélectionnez (p par défaut) :
```

```
Utilisation de la réponse p par défaut.
Numéro de partition (1-4, 1 par défaut) :
Premier secteur (2048-61439999, 2048 par défaut) :
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-61439999, 61439999 par défaut) :

Une nouvelle partition 1 de type « Linux » et de taille 29,3 GiB a été créée.

Commande (m pour l'aide) : w
La table de partitions a été altérée.
Appel d'ioctl() pour relire la table de partitions.
Synchronisation des disques
```

Maintenant que la nouvelle partition est créé on va lui mettre l'étiquette ext4, avec la commande `mkfs.ext4 /dev/sdb1` ce qui donne :

```
mke2fs 1.46.5 (30-Dec-2021)
/dev/sdb1 contient un système de fichiers vfat étiqueté « PHILIPS UFD »
Procéder malgré tout ? (o,N) o
En train de créer un système de fichiers avec 7679996 4k blocs et 1921360 i-noeuds.
UUID de système de fichiers=6eb7057f-b73a-4586-9112-06c4fe7bb191
Superblocs de secours stockés sur les blocs :
32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
4096000

Allocation des tables de groupe : complété
Écriture des tables d'i-noeuds : complété
Création du journal (32768 blocs) : complété
Écriture des superblocs et de l'information de comptabilité du système de
fichiers : complété
```

Voila maintenant nous avons bien notre clé USB prête à l'emploi. Pour confirmer que la clé est au bon format on peut refaire la commande `lsblk -f` ainsi que la commande `fidisk -l` :

```
...
sdb
└─sdb1      ext4     1.0         6eb7057f-b73a-4586-9112-06c4fe7bb191
...
```

```
Disque /dev/sdb : 29,3 GiB, 31457280000 octets, 61440000 secteurs
Disk model: PHILIPS USB
Unités : secteur de 1 × 512 = 512 octets
```

```
Taille de secteur (logique / physique) : 512 octets / 512 octets
taille d'E/S (minimale / optimale) : 512 octets / 512 octets
Type d'étiquette de disque : dos
Identifiant de disque : 0x128faffe

Périphérique Amorçage Début      Fin Secteurs Taille Id Type
/dev/sdb1              2048 61439999 61437952  29,3G 83 Linux
```

### Transférer la clé de déchiffrage sur la clé usb

Avant de pouvoir transférer le fichier il faut monter la clé USB pour qu'elle soit accessible :

```
# Création du dossier de montage
mkdir /mnt/usbkey

# Montage de la clé USB
mount /dev/sdb1 / mnt/usbkey
```

Déplacer ou copier le fichier généré *(à l'étape 1)* vers la clé USB :

```
cp /root/keyfile /mnt/usbkey/.
```

# Créer le script à exécuter au démarrage

Pour ce faire nous allons ajouter un script qui sera utilisé par initramfs-tools pour construire l'image initramfs. Tous les scripts se trouvent dans l'emplacement `/etc/initramfs-tools/scripts/` . Dans ce répertoire, se trouvent les différentes étapes du processus de démarrage pour configurer et monter le système.

Nous allons nous intéresser au dossier `local-top` qui contient des scripts exécutés après les scripts dans `init-top`, mais avant les scripts dans `init-premount`. Ces scripts sont utilisés pour des tâches locales spécifiques avant le montage du système de fichiers racine.

Créer le fichier `unlock-luks-from-usb` dans le répertoire vu précédemment : `/etc/initramfs-tools/scripts/local-top/` ce qui donne :

```
nano /etc/initramfs-tools/scripts/local-top/unlock-luks-from-usb
```

Pour y mettre le contenu suivant :

```
#!/bin/sh

PREREQ=""

prereqs() {
    echo "$PREREQ"
}

case $1 in
prereqs)
    prereqs
    exit 0
    ;;
esac

# Variables
CRYPT_NAME="sda5_crypt"
USB_UUID="XXXXXXX-XXXX-XXXX-XXXX-XXXXXXXX" # Remplace par l'UUID exact de la partition de la clé USB
KEYFILE="/mnt/usbkey/keyfile"   # Emplacement (laisser /mnt/usbkey) + Nom du fichier sur la clé USB
HDD_UUID="XXXXXXX-XXXX-XXXX-XXXX-XXXXXXXX" # UUID exact de la partition LUKS a déchiffrer

# On attend que les périphériques USB soient prêts
sleep 3

# Création du point de montage
mkdir -p /mnt/usbkey

# En cas d'erreur pour aider au débuggage décommenter cette ligne
# ls -al /dev/disk/by-uuid

# Cherche le slot de la clé USB via l'UUID
for dev in /dev/disk/by-uuid/$USB_UUID; do
    echo "[unlock-luks-from-usb] Tentative de montage de $dev..."
    mount -t ext4 $dev /mnt/usbkey && break
done

# Si le fichier clé est présent, on tente le déchiffrement
if [ -f "$KEYFILE" ]; then
```

```
    echo "[unlock-luks-from-usb] Clé trouvée. Déverrouillage..."
    # Cherche le slot de la partition LUKS à partir de l'UUID
    for dev in /dev/disk/by-uuid/$HDD_UUID; do
        echo "[unlock-luks-from-usb] Récupération de la partition LUKS $dev..."
        echo "[cryptsetup] tentative de déverrouillage"
        cryptsetup luksOpen $dev "$CRYPT_NAME" --key-file "$KEYFILE" && exit 0
        echo "Échec du déverrouillage "
    done
else
    echo "[unlock-luks-from-usb] Fichier clé non trouvé sur la clé USB."
fi

# Si on arrive ici, on ne déverrouille pas, donc cryptsetup demandera le mot de passe
```

Maintenant il faut prendre soin de remplacer les `XXXXXXX-XXXX-XXXX-XXXX-XXXXXXXX` par les vrais UUID des variables.
Commençons par l'UUID de la clé USB, c'est très simple nous pouvons faire la commande blkid pour lister les UUIDs des périphériques de stockage présent sur le serveur :

```
/dev/sdb1: UUID="8fedfdda-5293-43ec-96ce-0902440234c6" BLOCK_SIZE="4096" TYPE="ext4"
PARTUUID="128faffe-01"
```

Il faut récupérer tout ce qui est entre guillemet de la variable `UUID` pour le mettre dans le script pour la variable `USB_UUID`.

Il nous manque plus que l'UUID de la partition LUKS à déchiffrer, pour ce faire, il est possible d'utiliser la commande précédente `blkid`. Pour reconnaître le bon UUID, c'est celui qui a le type `crypto_LUKS`, comme ci-deesous :

```
/dev/sda5: UUID="502cdf24-1935-4f4d-8262-9434ba606114" TYPE="crypto_LUKS" PARTUUID="485azer1-
05"
```

Pour résumer la partie variable de notre script ressemble à ceci :

```
# Variables
CRYPT_NAME="sda5_crypt"
USB_UUID="8fedfdda-5293-43ec-96ce-0902440234c6" # Remplace par l'UUID exact de la partition de
la clé USB
KEYFILE="/mnt/usbkey/keyfile"   # Emplacement (laisser /mnt/usbkey) + Nom du fichier sur la
clé USB
HDD_UUID="502cdf24-1935-4f4d-8262-9434ba606114" # UUID exact de la partition LUKS a déchiffrer
```

# Rendre la clé USB visible au démarrage

Même si on a formaté notre clé USB dans un format nativement géré par Linux. Il se peut que la clé USB ne soit visible au démarrage car des modules permettant d'identifier la clé USB seront manquant de l'initramfs.

> Pour qu'elle soit visible au démarrage sans rajouter la configuration ci-dessous, la clé USB doit être identifié comme "Amorçable" lorsque vous faîtes la commande `fdisk -l` .

Par exemple une clé USB identifié comme "Amorçable" ressemble à ceci :

```
Disque /dev/sdb : 29,3 GiB, 31457280000 octets, 61440000 secteurs
Disk model: PHILIPS USB
Unités : secteur de 1 × 512 = 512 octets
Taille de secteur (logique / physique) : 512 octets / 512 octets
taille d'E/S (minimale / optimale) : 512 octets / 512 octets
Type d'étiquette de disque : dos
Identifiant de disque : 0x128faffe

Périphérique Amorçage Début      Fin Secteurs Taille Id Type
/dev/sdb1           *  2048 61439999 61437952  29,3G 83 Linux


 # Il y a une petite étoile * en dessous d'Amorçage
```

Ce n'est pas grave si une clé USB n'est pas Amorçable il suffit de rajouter la configuration suivante dans le fichier `/etc/initramfs-tools/modules` :

```
# List of modules that you want to include in your initramfs.
# They will be loaded at boot time in the order below.
#
# Syntax:  module_name [args ...]
#
# You must run update-initramfs(8) to effect this change.
#
# Examples:
#
# raid1
# sd_mod
```

```
usb_storage
ehci_pci
ehci_hcd
ohci_hcd
uhci_hcd
xHCI_hcd
usbhid
sd_mod
ext4
```

## Explication des modules

| Module | Description | Utilisation |
|---|---|---|
| `usb_storage` | Module pour la prise en charge des périphériques de stockage USB. | Permet de monter et d'accéder aux périphériques de stockage USB, comme les clés USB et les disques durs externes. |
| `ehci_pci` | Module pour le contrôleur hôte EHCI (Enhanced Host Controller Interface) PCI. | Prend en charge les contrôleurs USB 2.0 sur les bus PCI. |
| `ehci_hcd` | Module pour le contrôleur hôte EHCI (Enhanced Host Controller Interface). | Prend en charge les contrôleurs USB 2.0. |
| `ohci_hcd` | Module pour le contrôleur hôte OHCI (Open Host Controller Interface). | Prend en charge les contrôleurs USB 1.1. |
| `uhci_hcd` | Module pour le contrôleur hôte UHCI (Universal Host Controller Interface). | Prend en charge les contrôleurs USB 1.1. |
| `xhci_hcd` | Module pour le contrôleur hôte xHCI (eXtensible Host Controller Interface). | Prend en charge les contrôleurs USB 3.0. |
| `usbhid` | Module pour la prise en charge des périphériques HID (Human Interface Device) USB. | Permet la prise en charge des périphériques d'interface humaine comme les claviers, les souris et les manettes de jeu USB. |
| `sd_mod` | Module pour la prise en charge des périphériques de stockage SD (Secure Digital). | Permet de monter et d'accéder aux cartes SD et aux périphériques de stockage SD. |
| `ext4` | Module pour le système de fichiers ext4. | Permet de monter et d'accéder aux systèmes de fichiers ext4, qui est le système de fichiers par défaut pour de nombreuses distributions Linux. |

# Mise à jours de l'initramfs

Dans le contexte de la configuration du chiffrement du disque, nous utilisons cette commande après avoir modifié les fichiers de configuration comme `/etc/initramfs-tools/modules` et `/etc/initramfs-tools/scripts/local-top/`. Cela garantit que les changements sont inclus dans l'initramfs et sont disponibles pendant le processus de démarrage.

Pour mettre à jours l'initramfs il faut faire la commande :

```
update-initramfs -u -k all
```

- `-u` : Cette option signifie "update" (mettre à jour). Elle indique à la commande de mettre à jour l'initramfs existant plutôt que de créer un nouveau.
- `-k all` : Cette option spécifie pour quels noyaux l'initramfs doit être mis à jour. `-k all` signifie que l'initramfs doit être mis à jour pour tous les noyaux installés sur le système.

> Le système de démarrage est maintenant prêt, plus qu'a tester en redémarrant le serveur avec `reboot`.
> Si tout se passe bien le serveur doit démarrer normalement.
> Si le fichier n'est pas trouvé sur la clé, le mot de passe pour déverrouiller le disque dur sera demandé.

> Il est possible que quelque chose se passe mal dû a une mauvaise configuration. Dans ce cas nous allons nous retrouver dans le terminal d'initramfs. Si c'est le cas, il est possible de débloquer le serveur en suivant le tuto : Démarrer via le terminal initramfs

# Déverrouillage Automatique Autres Partition

> Pour réaliser ce tuto il est nécessaire d'avoir au préalable créé une partition chiffrée avec LUKS.

source : https://www.howtoforge.com/automatically-unlock-luks-encrypted-drives-with-a-keyfile

L'automatisation du déverrouillage d'une partition chiffrée au démarrage peut-être mis en place en suivant les étapes suivantes :

- 1/ Création d'un fichier de clé random
- 2/ Ajouter le fichier de clé à la partition LUKS
- 3/ Créer le mapper
- 4/ Monter la partition

# 1/ Création d'un fichier de clé random

la commande suivante créera un fichier au contenu aléatoire d'une taille de 4096 bits (mieux qu'un mot de passe de 20/30 caractères....). Vous pouvez utiliser n'importe quel fichier comme fichier clé, mais je pense qu'un fichier de 4kb avec un contenu aléatoire convient bien.

```
sudo dd if=/dev/urandom of=/root/keyfile bs=1024 count=4
```

Comme vous pouvez le voir avec l'arrtibut `of` , le fichier sera généré dans le dossier `/root` avec le nom `keyfile`.

Avant d'utiliser la nouvelle clé, rendre le fichier clé accessible en lecture seule à root

```
sudo chmod 0400 /root/keyfile
```

> Cela rendra le fichier clé lisible uniquement par root. Si quelqu'un accède à ce fichier clé, vous avez de toute façon un problème plus important sur votre serveur.

>

> Une autre solution consiste à attribuer à root:root le droit d'accès au fichier clé souhaité et à le placer dans le dossier /root.

# 2/ Ajouter le fichier à LUKS

Les dispositifs LUKS/dm_crypt peuvent contenir jusqu'à 10 fichiers clés/mots de passe différents. Ainsi, en plus du mot de passe déjà configuré, nous allons ajouter ce fichier clé comme méthode d'autorisation supplémentaire.

```
sudo cryptsetup luksAddKey /dev/sdX /root/keyfile
```

Il vous sera d'abord demandé d'entrer un mot de passe (existant) pour déverrouiller le lecteur.

# 3/ Création du Mapper

Les périphériques LUKS doivent créer un mappeur qui peut ensuite être référencé dans la fstab. Ouvrez /etc/crypttab :

```
sudo nano /etc/crypttab
```

et ajouter la ligne suivante dans le fichier :

```
# MapperNAme□partition□keyfile□□method
sdc_crypt      /dev/sdc1  /root/keyfile  luks

# OU
# MapperName□device UUID□□□□□□□□□□□□□keyfile□□  method
sdX_crypt      /dev/disk/by-uuid/247ad289-dbe5-4419-9965-e3cd30f0b080  /root/keyfile  luks

# OU encore
sdc1_crypt □   UUID=d4eca898-8155-4c4d-b1ed-48f696a6ad99 □□□  /root/keyfile    luks
```

sdX_crypt est le nom du mappeur qui est créé. Vous pouvez utiliser n'importe quel nom, par exemple "music" ou "movies" ou "sfdsfawe" ....

Sauvegardez et fermez le fichier.

Ce que nous avons fait ici, c'est dire que le fichier /root/keyfile sera utilisé au lieu du mot de passe pour déverrouiller le lecteur.

# 4/ Monter la partition

Maintenant, nous avons un périphérique déverrouillé (enfin, pas encore, mais lorsque le système sera démarré) et nous avons juste besoin de le monter. Ouvrez le fichier /etc/fstab :

```
sudo nano /etc/fstab
```

puis ajouter une nouvelle ligne comme ci-dessous :

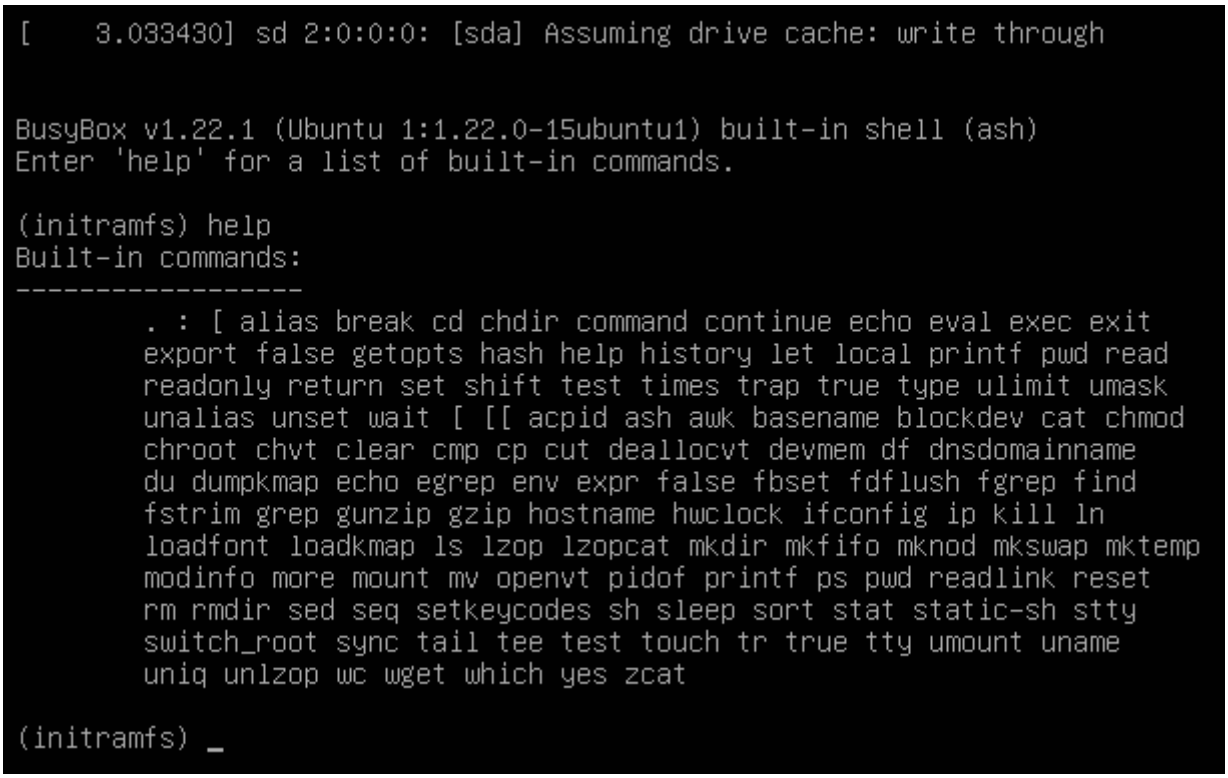
```
# MapperName□□  folder location□□
/dev/mapper/sdX_crypt  /media/sdX     ext3    defaults        0       2
```

Assurez-vous que le nom du mappeur que vous avez ajouté à l'étape 3 est correct. Assurez-vous également que le point de montage/dossier existe. Après l'avoir ajouté, enregistrez à nouveau le fichier et fermez-le.

# Démarrer via le terminal initramfs

> L'invite de commande `initramfs>` correspond à un shell de secours fourni par l'initramfs (Initial RAM Filesystem) lorsque le processus de démarrage normal échoue.

Dans notre cas, cela peut arriver si notre configuration est erroné, alors au démarrage le serveur proposera une interface semblable à ceci :




Pas de panique, ce tuto permet de débloquer cette situation afin de faire démarrer le serveur.

## Déverrouiller le disque principale

Tout d'abord, si on est dans l'invite de commande de l'initramfs, c'est que le disque principal n'a pas était déverrouillé via notre fichier clé présent dans la clé USB.

Donc la première étape est d'identifier le slot du disque principal ainsi que sa partition pour faire la commande `cryptsetup` pour le déverrouiller à la main.

Pour se faire la commande `blkid` fonctionne et va donner une résultat ressemblant à :

```
/dev/sdb1: UUID="868906c1-d1db-4431-aa76-44d7babb6798" BLOCK_SIZE="4096" TYPE="ext4"
PARTUUID="XXXXXXX-XX"
/dev/sda5: UUID="f6d7xxx-xxx-xxx-xxx-xxxx" TYPE="crypto_LUKS" PARTUUID="456azee1-05"
/dev/sda1: UUID="f831bxxxx-xxx-xxx-xxx-xxxxx" BLOCK_SIZE="1024" TYPE="ext2"
PARTUUID="456azee1-01"
```

Nous voyons bien notre clé USB (sdb1) ainsi que le disque dur (sda1) avec la partition chiffré (sda5). Nous pouvons remarquer que la partition UUID de notre HDD principal est la même pour sda5 et sda1.

Maintenant que nous connaissons les slots de nos différents périphériques nous avons 2 possibilités pour nous débloquer de cette situation :

- Option 1 : Monter la clé USB et utiliser le fichier clé pour déverrouiller la partition LUKS. Puis monter le système à la main.
  - Cela permettrai de tester que tout se passe bien durant la procédure de démarrage dans l'initramfs.
- Option 2 : Déverrouiller la partition LUKS avec le mot de passe puis de monter le système à la main
  - Si vous souhaitez directement modifier des fichiers dans votre systèmes (le fstab, modules, scripts, etc..)

## Option 1 - Monter la clé USB et utiliser le fichier de la clé

Pour cette étape nous commençons par créer le dossier qui servira de point de montage. Dans le système de l'initramfs le dossier "mnt" n'existe pas.

```
mkdir /mnt
```

Maintenant on va monter la clé USB en utilisant la même commande utilisé dans le fichier de montage automatique :

```
mount -t ext4 /dev/sdb1 /mnt
```

> Techniquement si le montage ne fonctionne pas avec cette commande c'est que le problème vient probablement de cette étape.

Afin de vérifier que tout c'est bien passé vous êtes sensé voir votre fichier dans le dossier mnt.

```
ls /mnt
> keyfile
```

Si on a bien le fichier on va pouvoir utiliser la deuxième commande utilisé dans le fichier de montage automatique :

```
cryptsetup luksOpen /dev/sda5 sda5_crypt --key-file /mnt/keyfile
```

> Techniquement si le déverrouillage ne fonctionne pas avec cette commande c'est que le problème vient probablement de cette étape.

Maintenant que la partition LUKS est déverrouillé nous pouvons passer à l'étape de montage du système Linux.

### Option 2 - Déverrouiller la partition LUKS avec le mot de passe

## Monter le système Linux

Pour réaliser cette étape, le déverrouillage de la partition LUKS est nécessaire. Le déblocage de la partition LUKS doit faire apparaître des mapper :

```
ls /dev/mapper
> control  sda5_crypt  sirius--vg-root  sirius--vg-swap_1
```

> Vérifier que les mappers existent bien

Tout d'abord si vous avez réaliser l'option 1 vous devez démonter la clé USB :

```
umount /mnt
# OU
umount /dev/sdb1
```

Maintenant nous allons réutiliser le dossier mnt pour monter le système dedans :

```
mount -t ext4 /dev/mapper/sirius--vg-root /mnt
```

Puis monter les dossiers nécessaires pour que le vrai système fonctionne :

```
mount -t proc /proc /mnt/proc
mount -t sysfs /sys /mnt/sys
mount --bind /dev /mnt/dev
```

et ainsi faire basculer la racine dans ce nouveau point de montage :

```
chroot /mnt
```

Nous sommes maintenant dans notre vrai systèmes. Mais ce n'est pas encore finis !

Pour retravailler le initramfs il faut monter également le `boot` qui se trouve dans une autre partition non chiffré.

La commande `lsblk -f` doit lister les périphériques ainsi que leurs partitions. Parmi la liste des partitions de notre disque principal nous devons en avoir une qui corresponds au `boot`.

```
sdb

├─sdb1                 ext2        1.0           978c28c7-0e7b-4172-ac8c-411d91a045f9
309M    27%
├─sdb2

└─sdb5                 crypto_LUKS 2             a27bc3e3-1d00-42bf-a312-
d77c19132baf
  └─sda5_crypt         LVM2_member LVM2 001      4da22ec6-4b19-4c5d-8080-
e5eeed0ec682
    ├─sirius--vg-root  ext4        1.0           935e9b50-e913-4af5-9e44-c68caf7393fc
102,8G     6% /
    └─sirius--vg-swap_1 swap       1             afa4e557-7b48-4f8a-96ec-
391a3c663d87                [SWAP]
```

Dans notre cas c'est la partition `sdb1`.

Donc on fait la commande suivante pour monter le boot dans notre système : `mount /dev/sda1 /boot`

Puis vérifier que tout le montage c'est bien passé avec la commande `ls /boot` le dossier ne doit pas être vide.