# cmtab

Informations sur les systèmes de fichier gérés par cryptmount

/etc/cryptmount/cmtab contient les informations sur les systèmes gérés par cryptmount. Le format est flexible :

```
TARGET_NAME {
 dev=DEVICE # REQUIRED
 flags=FLAG,FLAG,...
 startsector=STARTSECTOR
 numsectors=NUMSECTORS
 loop=LOOPDEV
 dir=MOUNT_POINT
 fstype=TYPE # REQUIRED
 mountoptions=MOPT,MOPT,...
 fsckoptions=FOPT;FOPT;...
 supath=SUPATH
 bootaction=BOOTACTION
 cipher=CIPHER
 ivoffset=IVOFFSET
 keyformat=KEYMANAGER
 keyfile=KEYFILE # REQUIRED
 keyhash=KEYHASH
 keycipher=KEYCIPHER
 keymaxlen=KEYMAXLEN
 passwdretries=NUMATTEMPTS
}
```

Certains champs, tels que dev et fstype sont obligatoires. De nombreux champs ont des valeurs par défaut. Un champ contenant une valeur non-numérique peut contenir une référence à une variable d'environnement :

**$(HOME)** Répertoire personnel de l'utilisateur

**$(UID)** UID de l'utilisateur

**$(USERNAME)** Nom de l'utilisateur

**$(GID)** GID du groupe primaire de l'utilisateur

**$(GROUPNAME)** Groupe primaire de l'utilisateur

# Définitions de cibles

**dev** Dpfinis le nom du périphérique ou du fichier.

**flags** "user", "nouser", "fsck", "nofsck", "mkswap", "nomkswap", "trim", "notrim". Défaut : user,fsck,nomkswap,notrim

**startsector** Secteur de début du système de fichier dans le périphérique. défaut : 0

**numsectors** Donne la longueur totale du système de fichier en secteur. Défaut : -1

**loop** Permet de spécifier un périphérique loopback. Défaut : auto

**dir** Point de montage

**fstype** Type de système de fichier

**mountoptions** Options de montage utilisées par mount

**fsckoptions** options utilisées par fsck

**supath** Définis la variable d'environnement PATH en lançant des sous-processus en tant que root. Peut être nécessaire pour fsck et mount. Défaut : /sbin :/bin :/usr/sbin :/usr/bin

**bootaction** Action lors du démarrage du système (none, mount, swap ou prepare)

**cipher** Algorithme de chiffrement à utiliser. Défaut : aes-cbc-plain

**keyformat** schéma de gestion de clé utilisé pour interagir avec le keyfile. (libgcrypt, luks, openssl-compact, builtin, raw). Défaut : builtin

**keyfile** Nom du fichier contenant la clé à utiliser

**ivoffset** Offset ajouté au numéro de secteur utilisé pour construire le vecteur d'initialisation de l'algorithme de chiffrement. Défaut : 0

**keyhash** Algorithme de hashage à utiliser

**keycipher** Algorithme de chiffrement à utiliser pour sécuriser la clé de déchiffrement

**keymaxlen** Longueur en octet de la clé de déchiffrement

**passwdretries** Nombre de tentative du mot de passe

# Choix du keymanager

cryptmount supporte différentes manières de protéger l'accès à la clé associée avec chaque système de fichier chiffré. Pour la plupart des utilisateurs, le keymanager builtin fournis a bon niveau de sécurité et une bonne flexibilité. Les keymanager alternatifs offre un grand choix de schéma de hashage de mot de pass et de compatibilité avec d'autres outils de chiffrement.

**builtin** Ce keymanager utilise un keyfile séparé. Une fonction de dérivation de clé (PBKPF2) utilisant l'algorithme de hashage SHA1, avec chiffrement blowfish-cbc est utilisé pour protéger la clé.

**libgcrypt** Ce keymanager utilise un keyfile séparé. Une fonction de dérivation de clé (PBKPF2) est utilisé pour protéger la clé, avec un algorithme de hashage et de chiffrement supporté par la version installé de la librairie libgcrypt.

**luks** Ce keymanager fournis une compatibilité avec le format LUKS. Au lieu d'un fichier séparé, LUKS utilise un en-tête dans le système de fichier lui-même.

**openssl/openssl-compat** Ce keymanager utiliser un keymanager séparé qui est compatible avec le format utilisé par l'outil de chiffrement opennssl. Une fonction de dérivation de clé (PBKPF2) est utilisé pour protéger la clé, avec un algorithme de hashage et de chiffrement disponible.

**password** Ce keymanager dérive la clé directement depuis le mot de passe de l'utilisateur

**raw** Ce keymanager utilise un keyfile séparé où la clé accès est stocké directement et sans chiffrement. Ce keymanager est utile pour gérer les partitions swap chiffrés, où le keyfile peut être choisis avec /dev/random et la clé sera différente à chaque fois quelle est lue.

# Sécurité

Parce que cryptmount nécessite d'opérer avec des privilèges setuid, il est très important que son fichier de configuration soit sécurisé. Idéalement, /etc/cryptmount/cmtab devrait être géré seulement par l'administrateur système, et tous les keyfiles devraient lisibles par leur propriétaire. cryptmount effectue des vérifications de sécurité sur /etc/cryptmount/cmtab chaque fois qui est lancé, et va refuser d'opérer sauf si les conditions suivant sont rencontrées :

- cmtab doit être possédé par root
- cmtab doit être un fichier régulier
- cmtab ne doit pas être en écriture globalement
- Le répertoire contenant cmtab doit être possédé par root
- Le répertoire contenant cmtab ne doit pas être en écriture globalement
- Pour chaque cible dans @CM_SYSCONFDIR@/cmtab, tous les chemins doivent être absolus

# Exemples

L'exemple suivant de @CM_SYSCONFDIR@/cmtab consiste de 5 cibles, utilisant divers algorithmes de chiffrement et stockent leur système de fichier de différentes manières, incluant une cible représentant une partition swap chiffrée :

```
_DEFAULTS_ {
 passwdretries=3 # allow 3 password attempts by default
}

basic {
 dev=/home/secretiveuser/crypt.fs
 dir=/home/secretiveuser/crypt # where to mount
 loop=auto # find free loop-device
 fstype=ext3 mountoptions=default
 cipher=aes-cbc-plain # filesystem encryption
 keyfile=/home/secretiveuser/crypt.key
 keyformat=builtin
}

partition {
 dev=/dev/hdb62 # use whole disk partition
 dir=/mnt/crypt62
 fstype=ext3 mountoptions=nosuid,noexec
 cipher=serpent-cbc-plain
 keyfile=@CM_SYSCONFDIR@/crypt_hdb62.key
 keyformat=openssl # use OpenSSL key-encryption
 keyhash=md5 keycipher=bf-cbc # encryption of key file
}

subset {
 dev=/dev/hdb63
 startsector=512 numsectors=16384 # use subset of partition
 dir=/mnt/encrypted\ subset\ of\ hdb
 fstype=reiserfs mountoptions=defaults
 cipher=twofish-cbc-plain # filesystem encryption
 keyfile=@CM_SYSCONFDIR@/crypt_hdb63.key
 keyformat=libgcrypt
 keyhash=md5 keycipher=blowfish-cbc # encryption of key file
}

encswap { # encrypted swap partition
 bootaction=swap
 dev=/dev/disk/by-id/scsi-SATA_ST500_ABCDEFG-part37
 startsector=16896 numsectors=1024 # use subset of partition
 fstype=swap flags=mkswap cipher=twofish-cbc-plain
 keyfile=/dev/random keymaxlen=16 keyformat=raw
}

luks { # partition created by cryptsetup-luks
 dev=/dev/hdb63
 dir=/mnt/luks-partition-$(USERNAME)
 keyformat=luks
 keyfile=/dev/hdb63
 fstype=ext3
}
```