

Chiffrement de données

sous GNU/Linux

Christophe SAHUT
csahut@nerim.net

RÉSIST - le 17 février 2003



Introduction

Qui

<http://csahut.nerim.net/resist/>

Plan

1- Cryptographie et GNU/Linux

2- Le chiffrement de données

3- Limites

4- Curiosités

1- Cryptographie et GNU/Linux

Vocabulaire :

- crypter ~ chiffrer
 - ◆ éviter d'employer "crypter"
- décrypter != déchiffrer
- cryptographie : ensemble des techniques de chiffrement de l'information
- cryptanalyse : ensemble des techniques de décryptage de l'information
- cryptologie : étude de la cryptographie et de la cryptanalyse

1- Cryptographie et GNU/Linux

Vocabulaire (on ne le dira jamais assez) :

- Linux = noyau (www.kernel.org)



- GNU/Linux = système d'exploitation : 2021 packages (www.gnu.org)



- www.CULTe.org : Club des Utilisateurs de Linux de Toulouse et des environs

1- Cryptographie et GNU/Linux

Pourquoi GNU/Linux ?

- Système stable, performant, complet et ouvert
- Sources disponibles
- De plus en plus répandu : gouvernements, ministères, écoles, entreprises...
- Même les poids lourds de l'informatique s'y mettent : IBM

1- Cryptographie et GNU/Linux

Les algorithmes de chiffrement :

- symétriques
- asymétriques
- de hachage

1- Cryptographie et GNU/Linux

Algorithmes symétriques

- Exemples : AES, DES, 3DES, IDEA, BLOWFISH, TWOFISH ...
- Longueur des clés : 56 bits (faible), 128 (correct), 256 (bon)

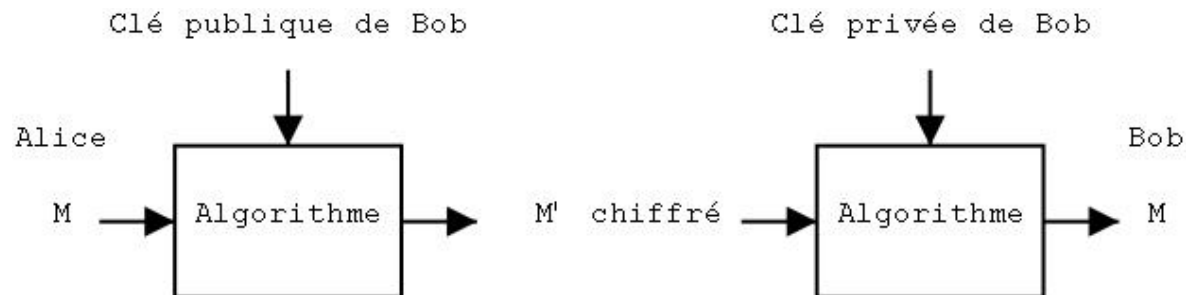


- Une seule clé K pour chiffrer et déchiffrer
- Nécessité au préalable de partager la clé K avec son interlocuteur

1- Cryptographie et GNU/Linux

Algorithmes asymétriques

- Exemples : RSA, El Gamal, GQ2, ECC ...
- Longueur des clés : 512 bits (faible), 1024 (correct), 2048 (bon)

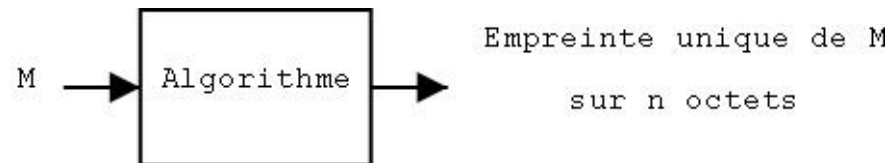


- Deux clés (complémentaires) pour chiffrer et déchiffrer
 - ♦ Clé publique
 - ♦ Clé privée
- Seule la clé privée peut déchiffrer des messages chiffrés par la clé publique

1- Cryptographie et GNU/Linux

Algorithmes de hachage

- Exemples : MD5, SHA-1, RIPE-MD-160...



- Le calcul d'un hachage est rapide
- Il est impossible de retrouver le message à partir d'un hachage
- Il est difficile de trouver deux messages ayant le même hachage
- Une entrée de longueur quelconque donne une sortie de longueur fixe

1- Cryptographie et GNU/Linux

Comparaison de rapidité

- Les algorithmes symétriques sont plus rapides que les algorithmes asymétriques
- Plus la clé est longue, plus le coût de chiffrement / déchiffrement est élevé
- On préfère les algorithmes symétriques pour chiffrer des données statiques
 - ◆ Essentiellement AES, Blowfish et Twofish
- Rapidité des principaux algorithmes symétriques (empirique, dépend de beaucoup de paramètres)
 - ◆ AES > Blowfish > RC5 > DES > IDEA > 3DES

1- Cryptographie et GNU/Linux

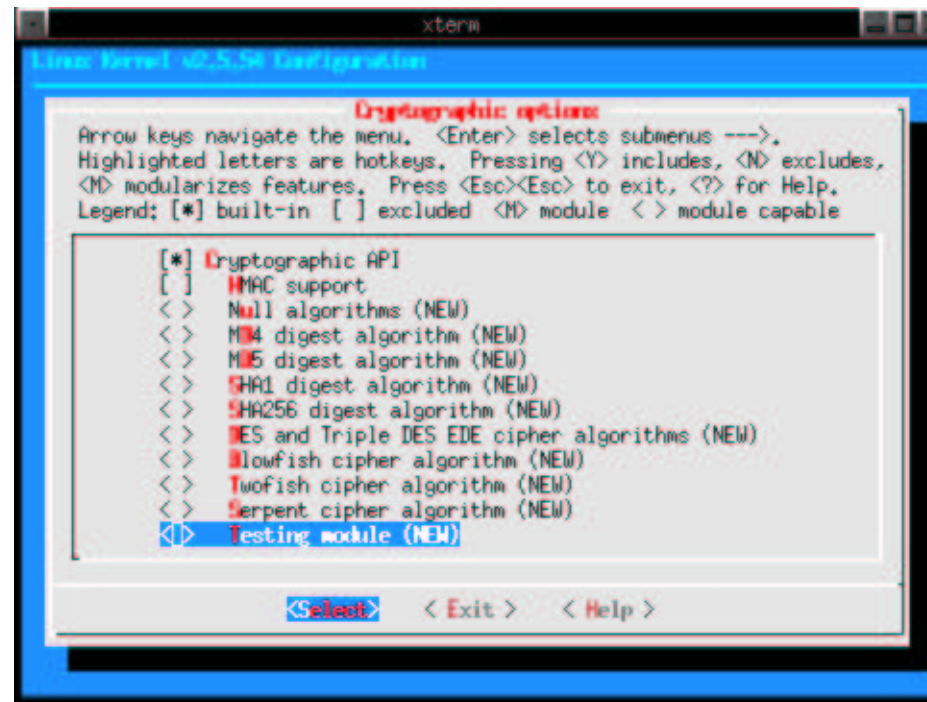
Cryptographie dans GNU/Linux

- Dans les noyaux 2.2.x et 2.4.x : AUCUNE
- Refus de Linus Torvalds d'intégrer de la crypto dans le noyau.
- Puis ...
 - ◆ levée des restrictions d'exportation de la crypto hors des USA
 - ◆ besoin d'un support ipsec dans le noyau (pour alléger freeswan entre autres)

1- Cryptographie et GNU/Linux

Cryptographie dans GNU/Linux

- Apparition de la crypto dans le noyau 2.5.45 et suivants



2- Le chiffrement de données

2- Le chiffrement de données

Plan :

- Niveau utilisateur
- Niveau systèmes de fichiers
- Niveau périphérique
- Swap

2- Le chiffrement de données

■ Niveau utilisateur



■ <http://www.gnupg.org>

2- Le chiffrement de données



■ Propriétés :

- ◆ Remplacement complet de PGP
- ◆ Distribué sous Gnu Public License
- ◆ N'utilise aucun algorithme breveté

- ◆ Permet l'utilisation d'RSA, ElGamal, DSA, AES, 3DES, Blowfish, Twofish, CAST5, MD5, SHA-1, RIPE-MD-160 et TIGER.

- ◆ Multiplateforme (GNU/Linux, *BSD, beaucoup d'UNIX, windows)

2- Le chiffrement de données



- Autorisé en France (à partir de la version 1.07 avec OpenSSL 0.9.6d) depuis mai 2002
- Très utilisé pour les mails (chiffrement / signature)
- Possibilité de chiffrer des fichiers en local
 - ◆ chiffrement symétrique
 - ◆ chiffrement asymétrique
- Démo

2- Le chiffrement de données



■ Conclusion :

- ♦ difficile à gérer quand on a beaucoup de données à chiffrer
- ♦ des données peuvent être oubliées (déchiffrées) dans des fichiers temporaires
- ♦ idéal pour du chiffrement occasionnel

2- Le chiffrement de données

Niveau système de fichiers

- * CFS - <http://www.crypto.com/software/>
- TCFS - <http://www.tcfs.it>
- * Self Certifying FS - <http://www.fs.org/>
- ReiserFS v4 - <http://www.namesys.com/>
- Autres : StegFS, SecureFS ...

2- Le chiffrement de données

CFS : Cryptographic File System

- Créé par Matt Blaze en 1993
- Repose sur NFS
- Pas de modification du système de fichiers sous-jacent
- Utilisation de DES-CBC 64 bits

2- Le chiffrement de données

CFS : Cryptographic File System

■ Principe :

- ◆ Création de répertoires spéciaux : `cmkdir`
- ◆ "Attachement" de ces répertoires : `cattach`
- ◆ Ecriture de données dans le répertoire attaché
- ◆ "Détachement" des répertoires : `cdetach`
- ◆ Les données sont écrites chiffrées

■ Démo

2- Le chiffrement de données

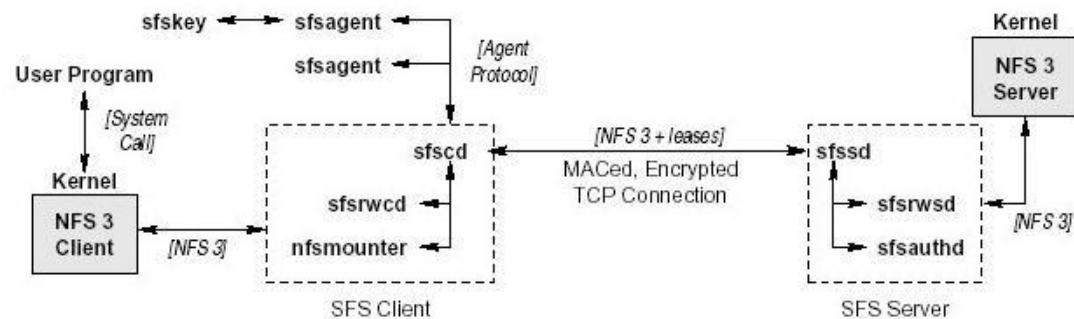
TCFS : Transparent Cryptographic File System

- Développé par un groupe de chercheurs de l'université de Salerno (Italie)
- Porté sur linux 2.0, 2.2, 2.4, NetBSD et OpenBSD
- Amélioration de CFS
 - ◆ Possibilité de partager des fichiers par groupe d'utilisateurs
 - ◆ Plusieurs modules pour ajouter des algorithmes de chiffrement

2- Le chiffrement de données

SFS : Self-Certifying File System

- système de fichiers chiffré basé sur NFS
- accès à des systèmes de fichiers distants de manière transparente



2- Le chiffrement de données

SFS : Self-Certifying File System

- Le serveur :

- ◆ génère un couple clé publique / clé privée
- ◆ exporte certains répertoires comme accessibles par SFS (à la NFS)

- Le client :

- ◆ génère un couple clé publique / clé privée
- ◆ s'enregistre auprès du serveur en lui fournissant sa clé publique

2- Le chiffrement de données

SFS : Self-Certifying File System

"usine à gaz" :

- utilisation de SHA-1 pour le hachage, ARC4 pour la confidentialité
- Blowfish pour le chiffrement des file handles de NFS
- eksblowfish pour le chiffrement des clés privées
- Rabin-Williams + SRP pour l'authentification

2- Le chiffrement de données

- Conclusion sur les systèmes de fichiers chiffrés :
 - ◆ lourd pour un utilisation locale
 - ◆ plus adapté pour des ressources partagées en réseau (NFS)
 - ◆ plus adapté en environnement multi-utilisateur

2- Le chiffrement de données

Niveau périphérique

- Le loopback
- kerneli - <http://www.kerneli.org>
- loop-aes - <http://loop-aes.sourceforge.net>

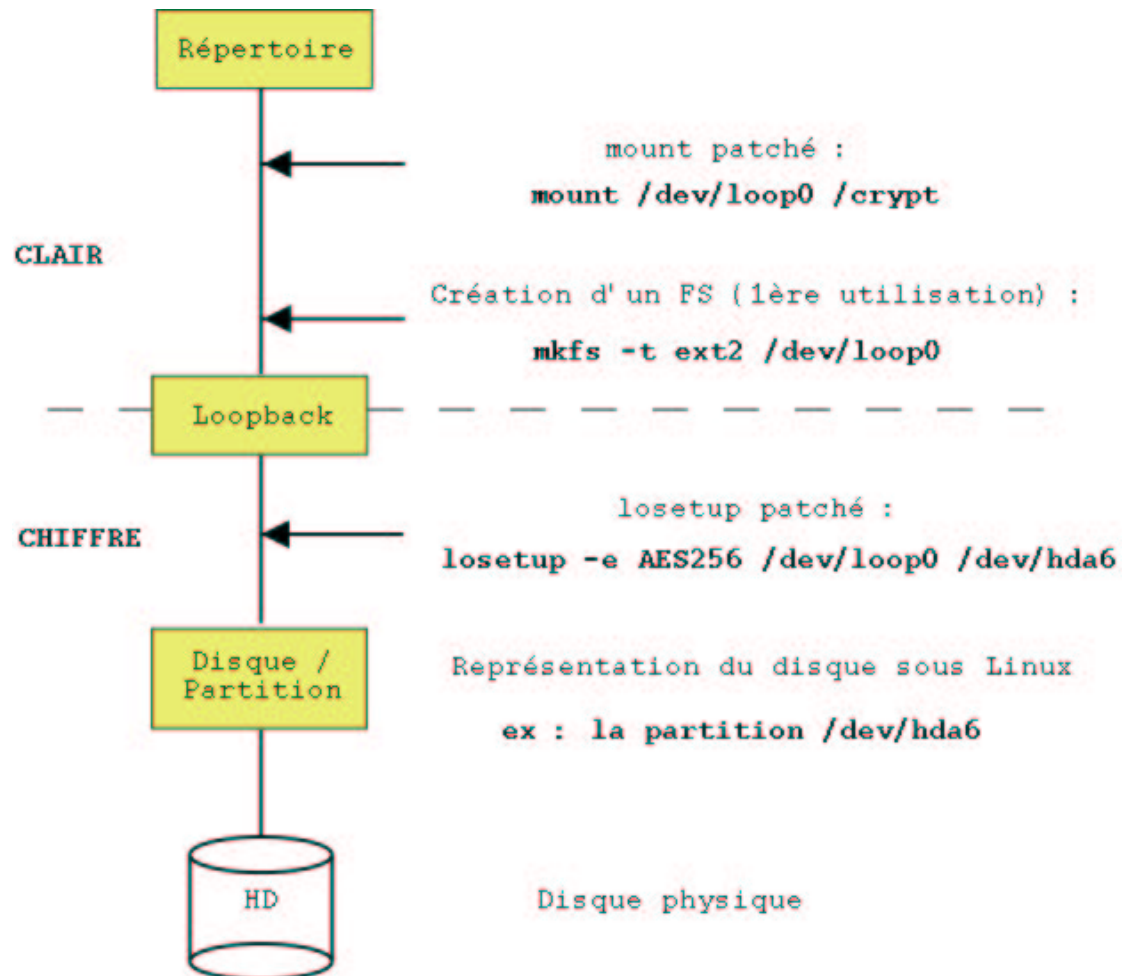
2- Le chiffrement de données

Le loopback

- Périphériques spéciaux sous Linux `/dev/loop[0-7]` qui permettent d'utiliser un fichier comme un périphérique.
- Permet de monter des fichiers contenant des systèmes de fichiers comme un disque quelconque.
- Interface utilisée pour chiffrer les données avant l'écriture sur le disque, et pour déchiffrer lors des lectures.

2- Le chiffrement de données

Schéma de principe :



2- Le chiffrement de données

Deux implémentations :

- kerneli
- loop-aes

ou HVR vs jari Ruusu sur nl.linuxcrypto.org :)

2- Le chiffrement de données

Kerneli

- ♦ Créé par Alexander Kjeldaas <astor@guardian.no>
- ♦ Maintenu par Herbert Valerio Riedel <hvr@ifs.tuwien.ac.at>
- ♦ Sous Gnu Public License
- ♦ Possibilité de patcher le noyau, ou de créer uniquement des modules

2- Le chiffrement de données

Kerneli (suite)

- CRYPTOTAPI `cryptoapi-x.x.x.tar.gz` :
 - ♦ Codes des algorithmes de chiffrement (3DES, AES, BLOWFISH, CAST5, RC5, RC6, SERPENT, TWOFISH etc..)
 - ♦ Codes des algorithmes de hachage (MD5, SHA1, SHA256/384/512, RIPE-MD160)
 - ♦ Le module `cryptoapi` : une api pour utiliser les algorithmes

2- Le chiffrement de données

Kerneli (suite)

- CRYPTOLOOP `cryptoloop-x.x.x.tar.gz` :
 - ◆ Le modules `cryptoloop` : permet de chiffrer le loopback
 - ◆ `Cryptoswap` : permet de chiffrer la swap
- PATCH-INT `patch-int-2.x.x.gz` :
 - ◆ Patch noyau avec `cryptoapi` + `cryptoloop`

2- Le chiffrement de données

LOOP-AES

- ♦ Créé par Dr Brian Gladman <brg@gladman.uk.net>
- ♦ Maintenu par Jari Ruusu <jari.ruusu@pp.inet.fi>
- ♦ Sous Gnu Public License
- ♦ Chiffrement en AES, BLOWFISH, SERPENT, TWOFISH
- ♦ Chiffrement de systèmes de fichiers et de partitions

2- Le chiffrement de données

LOOP-AES (suite)

- Plusieurs gadgets :)

- ◆ Possibilité d'utiliser une graine en plus du password (ralentir les attaques par dictionnaire)
- ◆ Possibilité d'utiliser des clés GPG
- ◆ Possibilité de chiffrer la racine / du système
- ◆ Possibilité de chiffrer des données sur CDROM ou tout autre media (via aespice)

2- Le chiffrement de données

LOOP-AES (suite)

- Utilisation d'une graine (seed) en plus du password

- ◆ Exemple de graine :

```
$ head -c 15 /dev/urandom | uuencode -m - | head -2 | tail -1  
pX+M259clcIW3rRy4Z2O
```

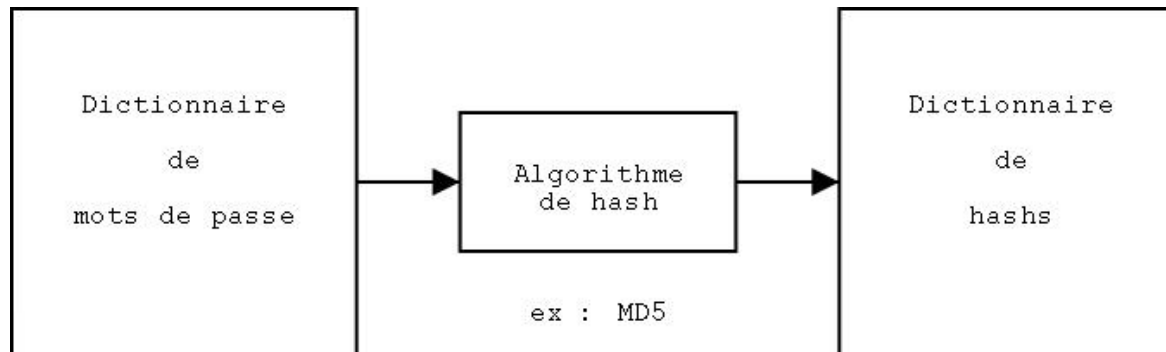
- On rajoute cette graine quand on monte le système de fichiers

```
# losetup -e AES256 -S pX+M259clcIW3rRy4Z2O /dev/loop0  
/dev/hda6
```

2- Le chiffrement de données

LOOP-AES (suite)

- Utilisation d'une graine (seed) en plus du password



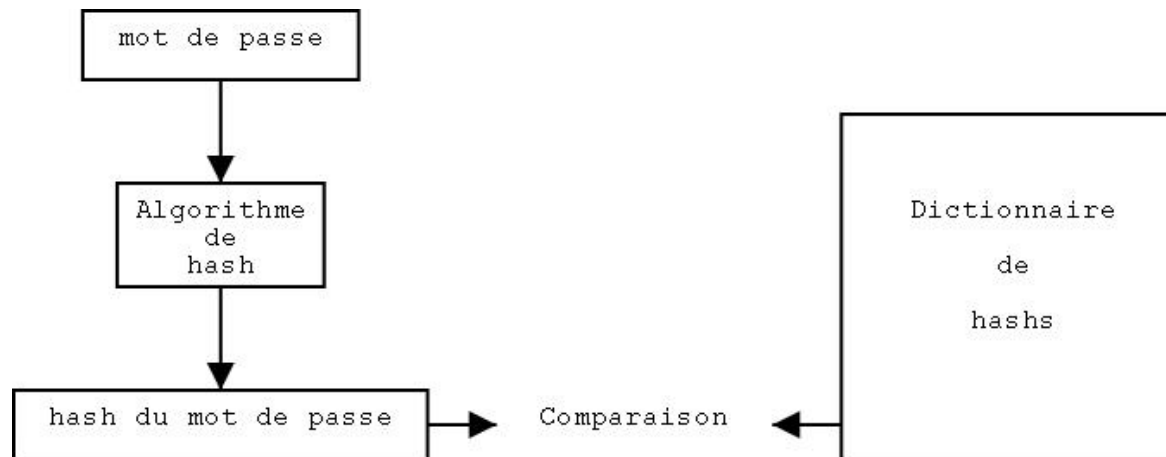
1ère étape classique avant une attaque "brute force" :

création d'un dictionnaire de hashes

2- Le chiffrement de données

LOOP-AES (suite)

- Utilisation d'une graine (seed) en plus du password

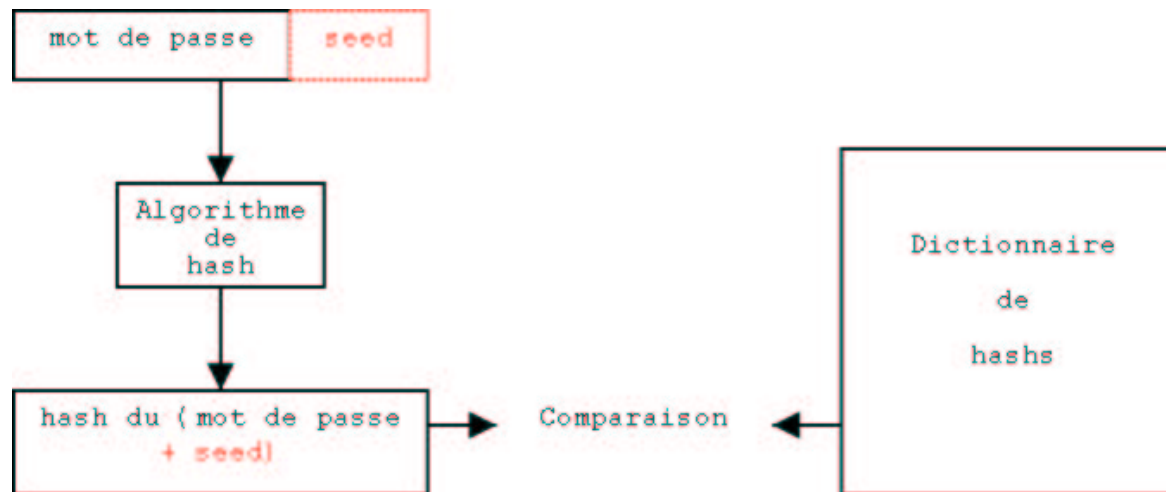


Utilisation sans graine

2- Le chiffrement de données

LOOP-AES (suite)

- Utilisation d'une graine (seed) en plus du password



Utilisation avec graine : besoin de reconstruire le dictionnaire de hash car peu de chance que la comparaison ne réussisse

2- Le chiffrement de données

LOOP-AES (suite)

- Possibilité d'utiliser des clés GPG stockées sur un media amovible (clé USB, etc...)
 - ♦ Création d'une paire de clés asymétriques
 - ♦ Chiffrement avec la clé publique d'une longue séquence aléatoire

```
$ head -c 45 /dev/random | uuencode -m - | head -2 | tail -1 | gpg  
--homedir /usbdongle -e -a -r "csahut" > /usbdongle/keyfile.gpg
```
- ♦ Besoin de la passphrase, des clés privée/publique, et du fichier keyfile.gpg pour monter la partition.

2- Le chiffrement de données

LOOP-AES : Conclusion

- Selon son concepteur, deux fois plus rapide que CryptoAPI
- Compatible avec la plupart des noyaux linux (-ac, -rh, -mdk, etc...)
- Meilleure gestion du chiffrement de la swap (scripts init propres à chaque distribution linux dans CryptoAPI, une seule ligne dans fstab avec loop-aes)
- Compatibilité des volumes chiffrés entre loop-aes et kerneli
- Loop-AES présent par défaut dans la Mandrake (> 8.2)
- Ecriture de l'HOWTO loop-AES en cours
- Démo

2- Le chiffrement de données

La swap

- Mémoire virtuelle, jouant le rôle d'une extension de la mémoire vive
- Risque :
 - ◆ passage dans la swap d'informations sensibles qui étaient dans la RAM
 - ◆ retrouver des données "étonnantes" : `strings /dev/hda5 |more`
- Solution :
 - ◆ Ne pas utiliser de swap et mettre beaucoup de RAM
 - ◆ Chiffrer les pages de swap

loop-aes et kerneli peuvent le faire

2- Le chiffrement de données

La swap (suite)

- Impact sur les performances négligeable
- Installation simple : /etc/fstab

```
/dev/hda5    none    swap    sw,loop=/dev/loop6,encryption=AES128 0 0
```

- Exemple d'utilité : gnupg n'a plus besoin d'être en suid root pour fixer ses pages mémoires (swap propre à la session)

3- Limites

3- Limites

Plan :

- Réflexion
- Faiblesses
- Performances

3- Limites

Réflexion

- A-t-on besoin de chiffrer ce que l'on chiffre ?
 - ◆ tout le disque ? (y-a-t'il un intérêt à chiffrer /bin ?)
 - ◆ données très importantes / confidentielles ? (probablement nécessaire / prudent de les chiffrer)

3- Limites

Réflexion (suite)

- Quelle est l'importance des données que je veux chiffrer ?
 - ◆ quel algorithme ?
 - ◆ quelle longueur de clé ?
 - ◆ pendant combien de temps ces données doivent être sûres ?

3- Limites

Faiblesses

- Essentiellement deux sources :
 - ♦ l'algorithme lui-même (existence de trappes ?) : peu probable
 - ♦ LE MOT DE PASSE : LA faiblesse du système
 - ♦ (l'utilisateur)

3- Limites

Faiblesse : le mot de passe

- Choix du mot de passe : azertyuiopmlkjhgfdsqwxcvbn ?
 - ◆ ce n'est pas parce que loop-aes exige 20 caractères qu'il faut mettre n'importe quoi.
- Sauvegarde : quelle(s) personne(s) possède(nt) le mot de passe ?
 - ◆ ex : bibliothèque suédoise
- Changement : le changement de mot de passe est relativement lourd
 - ◆ il faut tout déchiffrer et tout rechiffrer avec le nouveau mot de passe

3- Limites

Faiblesse : le mot de passe (suite)

- Le pire ennemi du mot de passe est le keylogger (après l'utilisateur :)
 - ◆ il existe de nombreuses variétés de keyloggers :
 - soft
 - lkm
 - matériel

3- Limites

Faiblesse : le mot de passe (suite)

- Quelques règles :
 - ◆ Vérifier que l'option "Secure Keyboard" est activée lorsque l'on tape la passphrase dans une xterm (interception par des X grabbers)
 - ◆ Vérifier les permissions des pty et ttys pour que personne ne puisse lire ce que vous tapez. (Utiliser plus généralement des "integrity checkers").
 - ◆ Via le réseau, toujours utiliser un tunnel chiffré.

3- Limites

Faiblesse : le mot de passe (suite)

■ Astuces :

- ◆ Copier/coller de caractères dans l'écran (long et fastidieux)
- ◆ Utilisation de gpggrid (cf partie "Curiosités")

3- Limites

Faiblesse : l'utilisateur

- Automount

- ◆ Montage de la partition dans les scripts init (le chiffrement ne sert plus à rien)

- Comportement incohérent

- ◆ Copie de données confidentielles dans une partition non chiffrée

Trop tard! Il y aura des traces

3- Limites

Performances

- Temps / CPU
 - ◆ un système de fichiers chiffré apporte une durée à chaque chiffrement / déchiffrement (écriture / lecture)
- Globalement, l'impact est négligeable (FS ou swap)

4- Curiosités

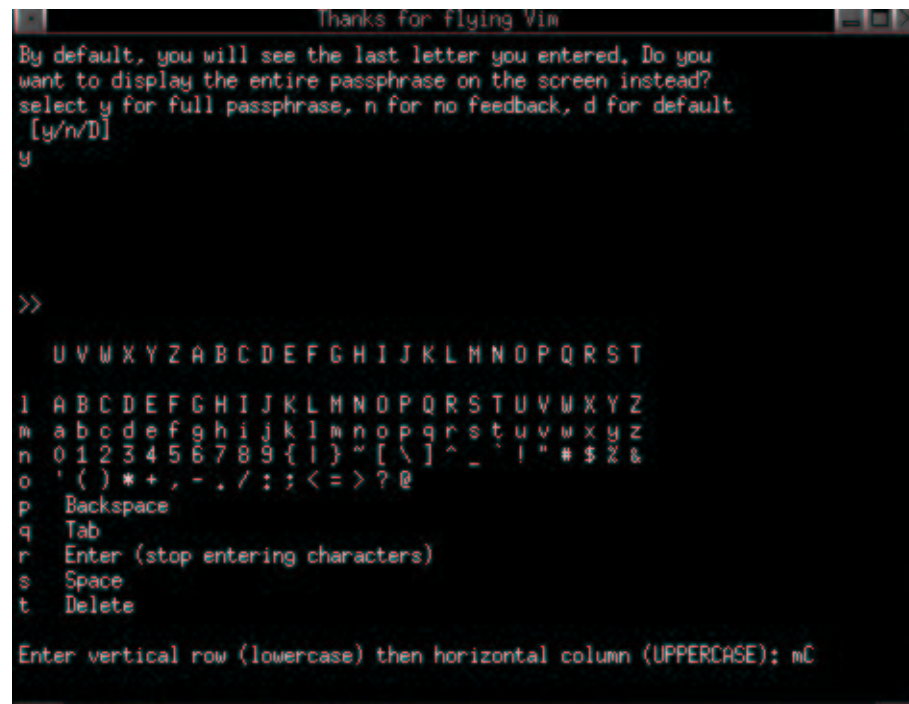
4- Curiosités

Plan :

- gpggrid
- losetup0.diff

4- Curiosités

- gpggrid de Tinfoil Hat Linux : <http://tinfoilhat.shmoo.com/>



```
Thanks for flying Vim
By default, you will see the last letter you entered. Do you
want to display the entire passphrase on the screen instead?
select y for full passphrase, n for no feedback, d for default
[y/n/D]
y

>>

  UVWXYZ ABCDEFGHIJKLMNOPQRST
l  ABCDEFGHIJKLMNOPQRSTUVWXYZ
m  abcdefghijklmnopqrstuvwxyz
n  0123456789{ | } ~ [ \ ] ^ _ ` ! " # $ % &
o  ' ( ) * + , - . / : ; < = > ? @
p  Backspace
q  Tab
r  Enter (stop entering characters)
s  Space
t  Delete

Enter vertical row (lowercase) then horizontal column (UPPERCASE): mC
```

Saisie de mot de passe façon "bataille navale"

4- Curiosités

losetup0.diff

- patch leurre pour loop-aes
- possibilité de changer le système de fichier monté selon le mot de passe

4- Curiosités

losetup0.diff (suite)

```
$ mount /dev/hda6 /crypt
```

```
Password :
```

```
Si mot_de_passe_1, on monte /dev/hda6
```

```
Si mot_de_passe_2, on monte  
/usr/X11R6/X11/lib/modules/libGLXcore.a (ou tout autre  
fichier/partition ou l'on a créé un système de fichier à l'intérieur)
```

```
et sans le signaler à l'utilisateur
```

- ◆ Démo

Pour se tenir informé :

Mailing lists / Newsgroups :

- linux-crypto@nl.linux.org
- sci.crypt
- fr.misc.cryptologie
- fr.comp.securite
- <http://www.counterpane.com/crypto-gram.html>

Conclusions

- Pour une utilisation personnelle sur ordinateur portable :
 - ◆ Chiffrement en loopback, éventuellement swap chiffrée
- Pour un utilisation en réseau (ressources partagées)
 - ◆ Utilisation de FS chiffrés à base de NFS (mieux adapté)
- Pour une utilisation occasionnelle :
 - ◆ GnuPG
- Pour la plupart des utilisations, l'AES 128 est un bon compromis (rapidité /fiabilité/légalité)

Questions ?

Contact:

- Email : csahut@nerim.net
- CV, transparents, etc.. : <http://csahut.nerim.net/resist/>
- ICQ : 148516209
- (IRC : CleeK sur irc.freenode.net)