HowTo Debian Encrypted Root

Contents

[hide]

Practical Privacy - How-To: installazione di un server Debian GNU/Linux 5.0.1 (Lenny) con filesystem di root cifrato

Ultimo aggiornamento: 2009.08.15

Obiettivo

Questa guida descrive passo a passo il procedimento da seguire per installare Debian GNU/Linux 5.0.1 su un PC che fungera' da server di rete ed i cui contenuti su hard disk saranno completamente cifrati con loop-AES ed algoritmo AES256, ad eccezione di una partizione di boot minimale in chiaro.

All'avvio del computer sara' necessario inserire un'unica passphrase per decifrare i filesystem di sistema e permettere l'avvio del sistema operativo. La chiave di cifratura principale e' protetta con GPG e puo' risiedere nella partizione di boot o su una chiavetta USB (autenticazione multi-fattore), sono fornite le configurazioni per entrambe le soluzioni.

Dal momento che la macchina installata verra' usata come server, in /usr /var /tmp /srv saranno montati filesystem contenuti in volumi logici distinti cosi' da prevenire blocchi di sistema causati dall'esaurimento dello spazio disponibile ed imporre specifiche opzioni di sicurezza (nodev,noexec,nosuid,..) ai diversi mountpoint. Sia /tmp che la partizione di swap saranno cifrate con chiavi random ad ogni mount cosi' da reinizializzarle ad ogni avvio del server.

L'installazione di Debian GNU/Linux avviene inizialmente in un volume logico di supporto chiamato lvRescue: questa scelta permette di semplificare il procedimento e di disporre in ogni momento di un sistema operativo funzionante dal quale e' possibile correggere errori e ripristinare il buon funzionamento del sistema Debian GNU/Linux contenuto nell'area cifrata del disco.

Il layout finale delle unita' di memorizzazione e' il seguente:

Disco IDE /dev/hda				
Device	**Dimensione**	**Tipo (hex)/Filesystem**	**Mount point**	**Note**
/dev/hda1	255MB	Linux (83) ext2	/boot	
/dev/hda5	32,0GB	Linux LVM (8e)		
/dev/hda6	32,0GB	Linux LVM (8e)		
/dev/hda7	32,0GB	Linux LVM (8e)		
/dev/hda8	32,0GB	Linux LVM (8e)		
/dev/hda9	32,0GB	Linux LVM (8e)		
/dev/hda10	32,0GB	Linux LVM (8e)		
/dev/hda11	32,0GB	Linux LVM (8e)		
/dev/hda12	32,0GB	Linux LVM (8e)		

/dev/hda13	32,0GB	Linux LVM (8e)		
/dev/hda14	31,9GB	Linux LVM (8e)		
LVM				
Un unico Volume Group chiamato vgSystem composto da 10 Physical Volume /dev/hda[5-14]				
/dev/vgSystem/lvRoot	2 GB	ext3 (AES256, GPG encrypted key)	/	
/dev/vgSystem/lvSwap	2,23 GB	swap (AES256, random key)	swap	
/dev/vgSystem/lvUsr	5 GB	ext3 (AES256, cleartext key)	/usr	
/dev/vgSystem/lvVar	3 GB	ext3 (AES256, cleartext key)	/var	
/dev/vgSystem/lvTmp	4 GB	ext3 (AES256, random key)	/tmp	
/dev/vgSystem/lvRescue	3,72 GB	ext3		per installazione e recovery
Chiavetta USB /dev/sda				
Device	**Dimensione**	**Tipo (hex)/Filesystem**	**Mount point**	**Note**
/dev/sda1	100MB	Linux (83) ext2	/boot	

Partizionamento ed installazione Debian di supporto in lvRescue

- Scaricare l'ISO CD1 di Debian 5.0.1 da http://www.debian.org/, verificarne l'integrita' attraverso MD5 check e masterizzarla.
- Avviare il server e bootare dal CD di Debian.
- Scegliere il metodo di installazione expert testuale e proseguire fino alla voce Partition disks. Il disco di riferimento e' un IDE da 320GB e corrisponde a /dev/hda. Premere ALT+F2 per accedere alla shell e digitare:
  fdisk /dev/hda
  - Creare partizione primaria ext2 da 250MB per /boot con flag bootable on:
    n p 1 <return> +250M
    a 1
  - Creare partizione estesa in tutto lo spazio restante per contenere i volumi logici:
    n e 2 <return> <return>
  - Creare 10 partizioni logiche Phisical volume for LVM, per calcolarne le dimensioni in cilindri si puo' ad esempio prendere il numero di cilindri del disco visualizzato da fdisk (38913 per il disco di riferimento), sottrarre il numero di cilindro in cui finisce la prima partizione primaria (31) e dividere per 10: (38913-31)/10=3888. La suddivisione del disco in vari volumi fisici sara' utile nel caso in futuro insorga l'esigenza di creare altri VolumeGroup oltre a quello di sistema.
    Ripetere 9 volte:
    n l <return> +3888
    Infine:
    n l <return> <return>
    t 5 8e
    t 6 8e
    t 7 8e
    t 8 8e
    t 9 8e
    t 10 8e
    t 11 8e
    t 12 8e
    t 13 8e
    t 14 8e
  - Salvare la partition table con le modifiche apportate:
    w q
  - Verificare la partition table con il comando:

Command: fdisk -l /dev/hda

```
Disk /dev/hda: 320.0 GB, 320072933376 bytes
255 heads, 63 sectors/track, 38913 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1          31      248976   83  Linux
/dev/hda2              32       38913   312319665    5  Extended
/dev/hda5              32        3921    31246393+  8e  Linux LVM
/dev/hda6            3922        7811    31246393+  8e  Linux LVM
/dev/hda7            7812       11701    31246393+  8e  Linux LVM
/dev/hda8           11702       15591    31246393+  8e  Linux LVM
/dev/hda9           15592       19481    31246393+  8e  Linux LVM
/dev/hda10          19482       23371    31246393+  8e  Linux LVM
/dev/hda11          23372       27261    31246393+  8e  Linux LVM
/dev/hda12          27262       31151    31246393+  8e  Linux LVM
/dev/hda13          31152       35041    31246393+  8e  Linux LVM
/dev/hda14          35042       38913    31101808+  8e  Linux LVM
```

- Premere ALT+F1 per tornare all'installer, selezionare Partition disks e viene avviato il partitioner. Selezionare Configure the Logical Volume Manager, rispondere Yes alla domanda Keep current partition layout and configure LVM?.
  - Selezionare Create Volume Group, immettere il nome **vgSystem** e selezionare tutti i device da /dev/hda5 a /dev/hda14.
  - Selezionare Create Logical Volume, selezionare vgSystem come volume group e digitare **lvRescue** come nome per il nuovo volume logico. Qui verra' inizialmente installato il sistema Debian su una partizione in chiaro cosi' da semplificare il setup delle partizioni cifrate e mettere a disposizione un ambiente di disaster recovery. Una volta configurate le partizioni cifrate e migrato il sistema su di esse questa partizione potra' essere eliminata e lo spazio liberato aggiunto ad uno dei logical volume cifrati. Immettere 4GB come dimensione.
  - Selezionare Create Logical Volume, selezionare vgSystem come volume group e digitare **lvSwap** come nome per il nuovo volume logico. La partizione sara' usata sia dal sistema in chiaro che in quello cifrato come swap. Immettere come dimensione il doppio della RAM installata (nel riferimento 2.4GB).

- Completare la configurazione LVM selezionando Finish.
- Scorrere l'elenco partizioni.
  Selezionare /dev/hda1, specificare Use as: Ext2, Mount point /boot, Done setting up the partition.
  Selezionare vgSystem-lvRescue-#1, specificare Use as: Ext3, Mount point /, Done setting up the partition.
  Selezionare vgSystem-lvSwap-#1, specificare Use as: swap, Done setting up the partition.
  Selezionare Finish partitioning and write changes to disk, rispondere Yes a Write the changes to disks.

- Proseguire l'installazione di Debian selezionando Install the base system. Alla voce Software Selection spuntare solo la linea Standard system. Installare GRUB nell'MBR di /dev/hda. Riavviare.

Creazione del sistema Debian cifrato

- Accedere come root. Commentare nel file /etc/apt/sources.list la riga corrispondente al cdrom, quindi controllare la disponibilita' di nuove versioni dei package installati e procedere all'aggiornamento con **apt-get update ; apt-get upgrade**. Installare un server ssh e screen se si desidera continuare l'installazione da terminale remoto per comodita' **apt-get install openssh-server screen** (opzionale).

Configurazione e ricompilazione del kernel

- Per avviare il sistema cifrato e' necessario che nella configurazione del kernel i filesystem di **/boot** e **/** e il supporto ai dischi IDE siano compilati inclusi, ma non come moduli e che si rispettino i seguenti parametri:

```
CONFIG_MODULES=y
CONFIG_BLK_DEV_LOOP=n (CONFIG_BLK_DEV_LOOP=y or CONFIG_BLK_DEV_LOOP=m will *not* work.)
CONFIG_KMOD=y is recommended but not required
CONFIG_BLK_DEV_RAM=y
CONFIG_BLK_DEV_RAM_SIZE=4096
CONFIG_BLK_DEV_INITRD=y
CONFIG_MINIX_FS=y
CONFIG_PROC_FS=y
CONFIG_CRAMFS=n (or CONFIG_CRAMFS=m)
```

- Controllare nella configurazione del kernel installato **/boot/config-2.6.26-2-686** il valore dei parametri di cui sopra:

Command: grep -e "CONFIG_MODULES=" -e "CONFIG_BLK_DEV_LOOP=" -e "CONFIG_KMOD=" -e "CONFIG_BLK_DEV_RAM=" -e "CONFIG_BLK_DEV_RAM_SIZE=" -e "CONFIG_BLK_DEV_INITRD=" -e "CONFIG_MINIX_FS=" -e "CONFIG_PROC_FS=" -e "CONFIG_CRAMFS=" /boot/config-2.6.26-2-686

```
CONFIG_BLK_DEV_INITRD=y
CONFIG_MODULES=y
CONFIG_KMOD=y
CONFIG_BLK_DEV_LOOP=m
CONFIG_BLK_DEV_RAM=y
CONFIG_BLK_DEV_RAM_SIZE=8192
CONFIG_PROC_FS=y
CONFIG_CRAMFS=m
CONFIG_MINIX_FS=m
```

- Dal momento che i valori dei parametri non sono quelli desiderati, ricompiliamo il kernel impostandoli opportunamente:

```
apt-get install kernel-package libncurses5-dev fakeroot wget bzip2 build-essential
cd /usr/src/
wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.30.1.tar.bz2
wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.30.1.tar.bz2.sign
gpg --verbose --verify linux-2.6.30.1.tar.bz2.sign
gpg --keyserver pgp.mit.edu --recv-keys 517D0F0E
gpg --verify linux-2.6.30.1.tar.bz2.sign
tar jxvf linux-2.6.30.1.tar.bz2
ln -s linux-2.6.30.1.tar.bz2 linux
cd linux
make clean && make mrproper
cp /boot/config-2.6.26-2-686 ./.config
make menuconfig
```

- Selezionare Load an Alternate Configuration File e inserire **.config**. Modificare i parametri opportuni:

```
CONFIG_BLK_DEV_INITRD=y (->General setup->Initial RAM filesystem and RAM disk (initramfs/initrd) support)
CONFIG_MODULES=y (->Enable loadable module support)
CONFIG_BLK_DEV_LOOP=n (->Device drivers->Block devices->Loopback device support)
CONFIG_BLK_DEV_RAM_SIZE=4096 (->Device drivers->Block devices->Default RAM disk size (kbytes))
CONFIG_IDE=y (->Device drivers->ATA/ATAPI/MFM/RLL support)
CONFIG_BLK_DEV_DM=y (->Device drivers->Multiple devices driver support (RAID and LVM)->Device mapper support)
CONFIG_EXT2_FS=y (->File systems->Ext3 journalling file system support)
CONFIG_EXT3_FS=y (->File systems->Ext3 journalling file system support)
CONFIG_CRAMFS=n (->File systems->Miscellaneous filesystems->Compressed ROM file system support (cramfs))
CONFIG_MINIX_FS=y (->File systems->Miscellaneous filesystems->Minix file system support)
```

- Se si desidera tenere la chiave di cifratura di root su una chiave USB da montare all'avvio, e' necessario abilitare il supporto generico USB, usb-storage e SCSI nel kernel.
- Uscire salvando la nuova configurazione del kernel. Compilare il kernel:

```
make-kpkg clean
fakeroot make-kpkg --initrd --append-to-version=-crypto01 kernel_image kernel_headers
dpkg -i /usr/src/linux-image-2.6.30.1-crypto01_2.6.30.1-crypto01-10.00.Custom_i386.deb
dpkg -i /usr/src/linux-headers-2.6.30.1-crypto01_2.6.30.1-crypto01-10.00.Custom_i386.deb
```

- Controllare che **/boot/grub/menu.lst** sia stato correttamente aggiornato, quindi riavviare per verificare il funzionamento del nuovo

kernel.

Installazione del modulo kernel loop-AES

- Installare i sorgenti di loop-AES (saranno disponibili in **/usr/src/loop-aes.tar.bz2**) e i package richiesti per compilarlo e installarlo via module-assistant:
  apt-get install debhelper gettext html2text intltool-debian po-debconf module-assistant loop-aes-utils
  cd /usr/src && wget http://http.us.debian.org/debian/pool/main/l/loop-aes/loop-aes-source_3.2g-1_all.deb
  dpkg -i loop-aes-source_3.2g-1_all.deb
- Compilare il modulo kernel loop-AES:
  m-a prepare
  m-a build loop-aes
  m-a install loop-aes
  modprobe loop && cd /usr/src/modules/loop-aes && make tests

Cifratura dello swap

- Cifrare lo swap:
  swapoff -a

Command: vim /etc/fstab

```
#Modificare la riga dello swap come segue:
/dev/mapper/vgSystem-lvSwap none            swap    sw,loop=/dev/loop0,encryption=AES256       0       0
```

dd if=/dev/zero of=/dev/mapper/vgSystem-lvSwap bs=64k conv=notrunc
mkswap /dev/mapper/vgSystem-lvSwap
swapon -a

Command: losetup -a

```
/dev/loop0: [000c]:1733 (/dev/mapper/vgSystem-lvSwap) offset=4096 encryption=AES256 multi-key-v3
```

Installazione in /boot dei programmi necessari all'avvio del sistema cifrato

- Compilare AESpipe statico:
  apt-get install dietlibc dietlibc-dev
  cd /usr/src
  wget http://loop-aes.sourceforge.net/aespipe/aespipe-v2.3e.tar.bz2
  wget http://loop-aes.sourceforge.net/aespipe/aespipe-v2.3e.tar.bz2.sign
  gpg --verify aespipe-v2.3e.tar.bz2.sign
  gpg --keyserver pgp.mit.edu --recv-key 3A220F51
  gpg --verify aespipe-v2.3e.tar.bz2.sign
  tar jxvf aespipe-v2.3e.tar.bz2
  cdaespipe-v2.3e
  CFLAGS="-O2" LDFLAGS="-static -s" ./configure
  make && make tests
- Compilare GnuPG statico e patching per aumentare la resistenza ai dictionary attack:
  cd /usr/src/ && wget ftp://ftp.gnupg.org/gcrypt/gnupg/gnupg-1.4.9.tar.bz2{,.sig}
  gpg --verify gnupg-1.4.9.tar.bz2.sig
  gpg --keyserver pgp.mit.edu --recv-key 1CE0C630
  gpg --verify gnupg-1.4.9.tar.bz2.sig
  tar jxvf gnupg-1.4.9.tar.bz2
  cd gnupg-1.4.9
  patch -p1 < /usr/src/modules/loop-aes/gnupg-1.4.9.diff
  CFLAGS="-O2" LDFLAGS="-static -s" ./configure --prefix=/usr --enable-static-rnd=linux
  make
  rm -f /usr/share/man/man1/{gpg,gpgv}.1.gz
  make install
  chown root:root /usr/bin/gpg
  chmod 4755 /usr/bin/gpg
- Copiare in **/bin** l'eseguibile di GnuPG perche' nella configurazione finale **/usr** non sara' ospitata sullo stesso filesystem montato in **/** e pertanto **/usr/bin/gnupg** non risulterebbe accessibile in fase di boot:
  mv /usr/bin/gpg /bin/gpg
  ln -s /bin/gpg /usr/bin/gpg
- Sospendere l'aggiornamento automatico di GnuPG (opzionale):
  echo gnupg hold | dpkg --set-selections
- Copiare in **/boot** il modulo loop-AES:
  mkdir -p /boot/modules/2.6.30.1-crypto01/
  cd /boot/ && ln -s modules/2.6.30.1-crypto01/ modules-2.6.30-1-crypto01
  cp /lib/modules/2.6.30.1-crypto01/updates/loop.ko /boot/modules/2.6.30.1-crypto01
- Compilare e copiare in **/boot** una shell:
  cd /usr/src/
  wget http://gondor.apana.org.au/~herbert/dash/files/dash-0.5.5.1.tar.gz
  tar zxvf dash-0.5.5.1.tar.gz
  cd dash-0.5.5.1
  CFLAGS="-O2" LDFLAGS="-static -s" ./configure --prefix=/usr
  make
  cp src/dash /boot
  cd /boot && ln -s dash sh
- Patchare e compilare **mount** per abilitare l'opzione "cleartextkey" che permette di montare partizioni cifrate specificando un file che contiene in chiaro la chiave. Sulla macchina di riferimento la versione del pacchetto **util-linux** che include **mount**, visualizzata eseguendo il comando **dpkg -l util-linux**, e' la 2.13.1.1-1 pertanto occorre scaricare una vecchia versione del pacchetto loop-AES

che contiene la corretta patch. Per la compilazione e' inoltre necessario installare i package per lo sviluppo delle librerie libuuid e libblkid.

```
cd /usrc/src/
wget http://www.kernel.org/pub/linux/utils/util-linux-ng/v2.13/util-linux-ng-2.13.1.tar.bz2
wget http://www.kernel.org/pub/linux/utils/util-linux-ng/v2.13/util-linux-ng-2.13.1.tar.bz2.sign
gpg --verify util-linux-ng-2.13.1.tar.bz2.sign
tar jxvf util-linux-ng-2.13.1.tar.bz2
wget http://loop-aes.sourceforge.net/loop-AES/loop-AES-v3.2c.tar.bz2
wget http://loop-aes.sourceforge.net/loop-AES/loop-AES-v3.2c.tar.bz2.sign
gpg --verify loop-AES-v3.2c.tar.bz2.sign
tar jxvf loop-AES-v3.2c.tar.bz2
apt-get install libblkid-dev
apt-get install uuid-dev
cd util-linux-ng-2.13.1
patch -p1 < ../loop-AES-v3.2c/util-linux-ng-2.13.1.diff
CFLAGS="-O2 -Wall" ./configure
make SUBDIRS="mount"
cp /bin/mount /bin/mount.bak
cp /bin/umount /bin/umount.bak
cp /sbin/swapon /sbin/swapon.bak
cp /sbin/swapoff /sbin/swapoff.bak
cp /sbin/losetup /sbin/losetup.bak
cd mount
install -m 4755 -o root mount umount /bin
install -m 755 losetup swapon /sbin
rm -f /sbin/swapoff && ( cd /sbin && ln -s swapon swapoff )
rm -f /usr/share/man/man8/{mount,umount,losetup,swapon,swapoff}.8.gz
install -m 644 mount.8 umount.8 losetup.8 /usr/share/man/man8
install -m 644 swapon.8 swapoff.8 /usr/share/man/man8
rm -f usr/share/man/man5/fstab.5.gz
install -m 644 fstab.5 /usr/share/man/man5
mandb
cd ../..
#echo mount hold | dpkg --set-selections
```

- Copiare in **/boot** i binari necessari:

```
cp -p /bin/mkdir /boot/
cp -p /bin/mount /boot/
cp -p /bin/mknod /boot/
cp -p /bin/umount /boot/
cp -p /sbin/lvm /boot/
```

- Copiare in **/boot** le librerie richieste dai binari appena copiati, individuandole con il comando **ldd**:

```
cp -p /lib/libdevmapper.so.1.02.1 /boot/
cp -p /lib/libreadline.so.5 /boot/
cp -p /lib/libdl.so.2 /boot/
cp -p /lib/libc.so.6 /boot/
cp -p /lib/libselinux.so.1 /boot/
cp -p /lib/libncurses.so.5 /boot/
cp -p /lib/ld-linux.so.2 /boot/
cp -p /lib/libblkid.so.1 /boot/
cp -p /lib/libuuid.so.1 /boot/
```

Generazione initrd e configurazione script di avvio rootsetup

- E' necessario creare l'initrd che effettuera' le seguenti operazioni (tratto da 1):

```
1) mount some device as /lib, using configured file system type
2) if configured, run loadkeys from /lib to re-map keyboard
3) if configured, load loop kernel module from /lib/modules-SOMETHING/
4) run losetup from /lib, using many config parameters
5) mount the loop device at /new-root, using configured file system type
6) unmount /lib
7) call pivot_root system call to switch /new-root -> / and initrd minix root file system -> /initrd
8) run /sbin/init from encrypted root
```

Scelta A: chiave di cifratura della root nella partizione di /boot

- Per generare l'initrd bisogna creare il file di configurazione **build-initrd.conf** che si ottiene effettuando una copia di **build-initrd.sh**, eliminando da essa tutto cio' che segue la riga "### End of options" e impostando opportunamente le variabili:

```
cd /usr/src/modules/loop-aes && cp build-initrd.sh build-initrd.conf
vim build-initrd.conf
```

File: /usr/src/modules/loop-aes/build-initrd.conf

```
### All default-values can be altered via the configfile

# 1 = use devfs, 0 = use classic disk-based device names. If this is
# enabled (USEDEVFS=1) then setting USEPIVOT=1 is also required and kernel
# must be configured with CONFIG_DEVFS_FS=y CONFIG_DEVFS_MOUNT=y
USEDEVFS=0

# 0 = use old change_root, 1 = use pivot_root, 2 = use initramfs/switch_root
# See above header for root= and append= lilo.conf definitions.
```

```
# pivot_root is not available on 2.2 and older kernels.
# initramfs/switch_root is not available on kernels older than 2.6.13
USEPIVOT=2

# Unencrypted /boot partition. If devfs is enabled (USEDEVFS=1), this must
# be specified as genuine devfs name.
BOOTDEV=/dev/hda1

# /boot partition file system type
BOOTTYPE=ext2

# Encrypted root partition. If devfs is enabled (USEDEVFS=1), this must
# be specified as genuine devfs name.
CRYPTROOT=/dev/hda5

# root partition file system type
ROOTTYPE=ext3

# Encryption type (AES128 / AES192 / AES256) of root partition
CIPHERTYPE=AES256

# Optional password seed for root partition
# (this option is obsolete when gpg encrypted key file is used)
#PSEED="-S XXXXXX"

# Optional password iteration count for root partition
# (this option is obsolete when gpg encrypted key file is used)
#ITERCOUNTK="-C 100"

# This code is passed to cipher transfer function.
LOINIT="-I 0"

# 1 = use gpg key file to mount root partition, 0 = use normal key.
# If this is enabled (USEGPGKEY=1), file named rootkey.gpg or whatever
# GPGKEYFILE is set to must be manually copied to /boot (or to
# EXTERNALGPGDEV device if EXTERNALGPGFILES=1). If rootkey.gpg is not
# encrypted with symmetric cipher, pubring.gpg and secring.gpg must be
# manually copied to /boot (or to EXTERNALGPGDEV device if
# EXTERNALGPGFILES=1).
USEGPGKEY=1

# gpg key filename. Only used if USEGPGKEY=1
GPGKEYFILE=rootkey.gpg

# 1 = mount removable device EXTERNALGPGDEV that contains gpg key files
# 0 = don't mount
EXTERNALGPGFILES=0

# Device name that contains gpg key files. If devfs is
# enabled (USEDEVFS=1), this must be specified as genuine devfs name.
# Only used if EXTERNALGPGFILES=1
EXTERNALGPGDEV=

# Removable device EXTERNALGPGDEV file system type
# Only used if EXTERNALGPGFILES=1
EXTERNALGPGTYPE=

# 1 = use loop module, 0 = loop driver linked to kernel
USEMODULE=1

# 1 = stop after creating and copying initrd, 0 = also copy tools/libs
INITRDONLY=0

# Source root directory where files are copied from
SOURCEROOT=

# Destination root directory where files are written to.
# Normally this is empty, but if you run this script on some other root
# (i.e. Knoppix live CD), this must be configured to point to directory
# where your about-to-be-encrypted root partition is mounted. This script
# checks that an initrd directory exists there.
DESTINATIONROOT=

# dest-dir below dest-root
DESTINATIONPREFIX=/boot

# Name of created init ram-disk
INITRDGZNAME=initrd.gz

# Encrypted root loop device index (0 ... 7), 5 == /dev/loop5
# Device index must be one character even if max_loop is greater than 8
# _must_ match /etc/fstab entry:   /dev/loop5  /  ext2  defaults,xxxx  0  1
ROOTLOOPINDEX=1

# Temporary loop device index used in this script, 7 == /dev/loop7
```

```
TEMPLOOPINDEX=7

# Additional loop module parameters.
# Example: LOOPMODPARAMS="max_loop=8 lo_prealloc=125,5,200"
LOOPMODPARAMS=""

# 1 = set keyboard to UTF-8 mode, 0 = don't set
UTF8KEYBMODE=1

# 1 = load national keyboard layout, 0 = don't load
# You _must_ manually copy correct keyboard layout to /boot/default.kmap
# which must be in uncompressed form. (can not be .gz file)
LOADNATIONALKEYB=0

# Initial delay in seconds before /linuxrc attempts to mount /boot
# partition. Slow devices (USB-sticks) may need some delay.
INITIALDELAY=10

# Delay in seconds before /linuxrc attempts to mount partition containing
# external gpg key files. Slow devices (USB-sticks) may need some delay.
MOUNTDELAY=10

# 1 = prompt for BOOT-TOOLS media and ENTER press before mounting /boot
# 0 = normal case, don't prompt
TOOLSPROMPT=0

# 1 = use "rootsetup" program that executes losetup to initialize loop
# 0 = use normal "losetup" program directly to initialize loop
# If enabled, rootsetup program (+libs) _must_ be manually copied to /boot.
USEROOTSETUP=1

# 1 = use dietlibc to build /linuxrc. This permits passing parameters to init.
# 0 = use glibc to build /linuxrc. This prevents passing parameters to init
# and includes hacks that may be incompatible with some versions of glibc.
# The dietlibc can be found at http://www.fefe.de/dietlibc/
USEDIETLIBC=1

# C compiler used to compile /linuxrc program.
# 32bit x86 ubuntu-7.04 gcc-4.1.2 is known to miscompile /linuxrc. Affected
# users should install gcc-3.3 package, and change this to GCC=gcc-3.3
GCC=gcc

# 1 = load extra module, 0 = don't load
# If this is enabled, module must be manually copied to
# /boot/modules-KERNELRELEASE/ directory under name like foomatic.o
EXTRAMODULELOAD1=0
EXTRAMODULENAME1="foomatic"
EXTRAMODULEPARAMS1="frobnicator=123 fubar=abc"
# 1 = load extra module, 0 = don't load
EXTRAMODULELOAD2=0
EXTRAMODULENAME2=""
EXTRAMODULEPARAMS2=""
# 1 = load extra module, 0 = don't load
EXTRAMODULELOAD3=0
EXTRAMODULENAME3=""
EXTRAMODULEPARAMS3=""
# 1 = load extra module, 0 = don't load
EXTRAMODULELOAD4=0
EXTRAMODULENAME4=""
EXTRAMODULEPARAMS4=""
# 1 = load extra module, 0 = don't load
EXTRAMODULELOAD5=0
EXTRAMODULENAME5=""
EXTRAMODULEPARAMS5=""

### End of options
```

cd /usr/src/modules/loop-aes/ && sh build-initrd.sh build-initrd.conf

- Creare lo script **/boot/rootsetup** e settarne i permessi con **chmod 755 /boot/rootsetup**:

File: /boot/rootsetup

```
#!/lib/sh
if [ "x$1" != "x-d" ] ; then
    /lib/mkdir -p /proc /dev/mapper
    /lib/mount -n -t proc proc /proc
    #/lib/mknod /dev/hda5 b  3  5
    /lib/mknod /dev/hda6 b  3  6
    /lib/mknod /dev/hda7 b  3  7
    /lib/mknod /dev/hda8 b  3  8
    /lib/mknod /dev/hda9 b  3  9
    /lib/mknod /dev/hda10 b  3  10
    /lib/mknod /dev/hda11 b  3  11
    /lib/mknod /dev/hda12 b  3  12
    /lib/mknod /dev/hda13 b  3  13
```

```
    /lib/mknod /dev/hda14 b  3  14
    /lib/lvm vgscan --ignorelockingfailure
    /lib/lvm vgchange -ay --ignorelockingfailure
    /lib/umount -n /proc
    /lib/losetup -e AES256 -K /lib/rootkey.gpg -G /lib /dev/loop1 /dev/mapper/vgSystem-lvRoot
    x=$?
    exit ${x}   # exit with return status of losetup
else
    /lib/losetup -d /dev/loop1
    x=$?
    exit ${x}   # exit with return status of losetup
fi
```

Scelta B: chiave di cifratura della root su chiavetta USB

- Per generare l'initrd bisogna creare il file di configurazione **build-initrd.conf** che si ottiene effettuando una copia di **build-initrd.sh**, eliminando da essa tutto cio' che segue la riga "### End of options" e impostando opportunamente le variabili:
  cd /usr/src/modules/loop-aes && cp build-initrd.sh build-initrd.conf
  vim build-initrd.conf

File: /usr/src/modules/loop-aes/build-initrd.conf

```
### All default-values can be altered via the configfile

# 1 = use devfs, 0 = use classic disk-based device names. If this is
# enabled (USEDEVFS=1) then setting USEPIVOT=1 is also required and kernel
# must be configured with CONFIG_DEVFS_FS=y CONFIG_DEVFS_MOUNT=y
USEDEVFS=0

# 0 = use old change_root, 1 = use pivot_root, 2 = use initramfs/switch_root
# See above header for root= and append= lilo.conf definitions.
# pivot_root is not available on 2.2 and older kernels.
# initramfs/switch_root is not available on kernels older than 2.6.13
USEPIVOT=2

# Unencrypted /boot partition. If devfs is enabled (USEDEVFS=1), this must
# be specified as genuine devfs name.
BOOTDEV=/dev/hda1

# /boot partition file system type
BOOTTYPE=ext2

# Encrypted root partition. If devfs is enabled (USEDEVFS=1), this must
# be specified as genuine devfs name.
CRYPTROOT=/dev/hda5

# root partition file system type
ROOTTYPE=ext3

# Encryption type (AES128 / AES192 / AES256) of root partition
CIPHERTYPE=AES256

# Optional password seed for root partition
# (this option is obsolete when gpg encrypted key file is used)
#PSEED="-S XXXXXX"

# Optional password iteration count for root partition
# (this option is obsolete when gpg encrypted key file is used)
#ITERCOUNTK="-C 100"

# This code is passed to cipher transfer function.
LOINIT="-I 0"

# 1 = use gpg key file to mount root partition, 0 = use normal key.
# If this is enabled (USEGPGKEY=1), file named rootkey.gpg or whatever
# GPGKEYFILE is set to must be manually copied to /boot (or to
# EXTERNALGPGDEV device if EXTERNALGPGFILES=1). If rootkey.gpg is not
# encrypted with symmetric cipher, pubring.gpg and secring.gpg must be
# manually copied to /boot (or to EXTERNALGPGDEV device if
# EXTERNALGPGFILES=1).
USEGPGKEY=1

# gpg key filename. Only used if USEGPGKEY=1
GPGKEYFILE=rootkey.gpg

# 1 = mount removable device EXTERNALGPGDEV that contains gpg key files
# 0 = don't mount
EXTERNALGPGFILES=1

# Device name that contains gpg key files. If devfs is
# enabled (USEDEVFS=1), this must be specified as genuine devfs name.
# Only used if EXTERNALGPGFILES=1
EXTERNALGPGDEV=/dev/sda1

# Removable device EXTERNALGPGDEV file system type
# Only used if EXTERNALGPGFILES=1
```

```
EXTERNALGPGTYPE=ext2

# 1 = use loop module, 0 = loop driver linked to kernel
USEMODULE=1

# 1 = stop after creating and copying initrd, 0 = also copy tools/libs
INITRDONLY=0

# Source root directory where files are copied from
SOURCEROOT=

# Destination root directory where files are written to.
# Normally this is empty, but if you run this script on some other root
# (i.e. Knoppix live CD), this must be configured to point to directory
# where your about-to-be-encrypted root partition is mounted. This script
# checks that an initrd directory exists there.
DESTINATIONROOT=

# dest-dir below dest-root
DESTINATIONPREFIX=/boot

# Name of created init ram-disk
INITRDGZNAME=initrd.gz

# Encrypted root loop device index (0 ... 7), 5 == /dev/loop5
# Device index must be one character even if max_loop is greater than 8
# _must_ match /etc/fstab entry:   /dev/loop5  /  ext2  defaults,xxxx  0  1
ROOTLOOPINDEX=1

# Temporary loop device index used in this script, 7 == /dev/loop7
TEMPLOOPINDEX=7

# Additional loop module parameters.
# Example: LOOPMODPARAMS="max_loop=8 lo_prealloc=125,5,200"
LOOPMODPARAMS=""

# 1 = set keyboard to UTF-8 mode, 0 = don't set
UTF8KEYBMODE=1

# 1 = load national keyboard layout, 0 = don't load
# You _must_ manually copy correct keyboard layout to /boot/default.kmap
# which must be in uncompressed form. (can not be .gz file)
LOADNATIONALKEYB=0

# Initial delay in seconds before /linuxrc attempts to mount /boot
# partition. Slow devices (USB-sticks) may need some delay.
INITIALDELAY=10

# Delay in seconds before /linuxrc attempts to mount partition containing
# external gpg key files. Slow devices (USB-sticks) may need some delay.
MOUNTDELAY=10

# 1 = prompt for BOOT-TOOLS media and ENTER press before mounting /boot
# 0 = normal case, don't prompt
TOOLSPROMPT=0

# 1 = use "rootsetup" program that executes losetup to initialize loop
# 0 = use normal "losetup" program directly to initialize loop
# If enabled, rootsetup program (+libs) _must_ be manually copied to /boot.
USEROOTSETUP=1

# 1 = use dietlibc to build /linuxrc. This permits passing parameters to init.
# 0 = use glibc to build /linuxrc. This prevents passing parameters to init
# and includes hacks that may be incompatible with some versions of glibc.
# The dietlibc can be found at http://www.fefe.de/dietlibc/
USEDIETLIBC=1

# C compiler used to compile /linuxrc program.
# 32bit x86 ubuntu-7.04 gcc-4.1.2 is known to miscompile /linuxrc. Affected
# users should install gcc-3.3 package, and change this to GCC=gcc-3.3
GCC=gcc

# 1 = load extra module, 0 = don't load
# If this is enabled, module must be manually copied to
# /boot/modules-KERNELRELEASE/ directory under name like foomatic.o
EXTRAMODULELOAD1=0
EXTRAMODULENAME1="foomatic"
EXTRAMODULEPARAMS1="frobnicator=123 fubar=abc"
# 1 = load extra module, 0 = don't load
EXTRAMODULELOAD2=0
EXTRAMODULENAME2=""
EXTRAMODULEPARAMS2=""
# 1 = load extra module, 0 = don't load
EXTRAMODULELOAD3=0
EXTRAMODULENAME3=""
```

```
EXTRAMODULEPARAMS3=""
# 1 = load extra module, 0 = don't load
EXTRAMODULELOAD4=0
EXTRAMODULENAME4=""
EXTRAMODULEPARAMS4=""
# 1 = load extra module, 0 = don't load
EXTRAMODULELOAD5=0
EXTRAMODULENAME5=""
EXTRAMODULEPARAMS5=""

### End of options
```

cd /usr/src/modules/loop-aes/ && sh build-initrd.sh build-initrd.conf

- Il filesystem in /dev/sda1 che contiene la chiave di cifratura della root sara' montato in /mnt durante il boot, mentre i vari script e binari saranno accessibili da /lib come nel caso della scelta A.
- Creare lo script **/boot/rootsetup** e settarne i permessi con **chmod 755 /boot/rootsetup**:

File: /boot/rootsetup

```
#!/lib/sh
if [ "x$1" != "x-d" ] ; then
    /lib/mkdir -p /proc /dev/mapper
    /lib/mount -n -t proc proc /proc
    #/lib/mknod /dev/hda5 b  3  5
    /lib/mknod /dev/hda6 b  3  6
    /lib/mknod /dev/hda7 b  3  7
    /lib/mknod /dev/hda8 b  3  8
    /lib/mknod /dev/hda9 b  3  9
    /lib/mknod /dev/hda10 b  3  10
    /lib/mknod /dev/hda11 b  3  11
    /lib/mknod /dev/hda12 b  3  12
    /lib/mknod /dev/hda13 b  3  13
    /lib/mknod /dev/hda14 b  3  14
    /lib/lvm vgscan --ignorelockingfailure
    /lib/lvm vgchange -ay --ignorelockingfailure
    /lib/umount -n /proc
    /lib/losetup -e AES256 -K /mnt/rootkey.gpg -G /lib /dev/loop1 /dev/mapper/vgSystem-lvRoot
    x=$?
    exit ${x}   # exit with return status of losetup
else
    /lib/losetup -d /dev/loop1
    x=$?
    exit ${x}   # exit with return status of losetup
fi
```

Configurazione di GRUB

- Aggiungere in **/boot/grub/menu.lst** una entry per avviare il sistema cifrato:

Command: vim /boot/grub/menu.lst

```
title          Debian GNU/Linux, kernel 2.6.30.1-crypto01loop
root           (hd0,0)
kernel         /vmlinuz-2.6.30.1-crypto01
initrd         /initrd.gz
```

Command: grub

```
grub> device (hd0) /dev/hda

grub> root (hd0,0)
 Filesystem type is ext2fs, partition type 0xfd

grub> setup (hd0)
```

Creazione della root cifrata

Scelta A: chiave di cifratura della root nella partizione di /boot

- Creare la encryption key di **/** (il completamento dell'operazione puo' richiedere molto tempo, le attivita' che usano tastiera, mouse e dischi rigidi possono velocizzare molto il processo):

```
umask 077
head -c 3705 /dev/random | uuencode -m - | head -n 66 | tail -n 65 \
| gpg --symmetric -a --s2k-count 8388608 > /boot/rootkey.gpg
```

- Creare il volume logico e il filesystem di **/**:
  lvcreate -L 2G -n lvRoot vgSystem
  losetup -e AES256 -K /boot/rootkey.gpg /dev/loop1 /dev/mapper/vgSystem-lvRoot
  mkfs -t ext3 /dev/loop1
  losetup -d /dev/loop1

Scelta B: chiave di cifratura della root su chiavetta USB

- Se nel kernel e' stato correttamente incluso il supporto scsi e usb-storage, quando si connette la chiavetta USB essa viene vista

come device /dev/sda. Creare una partizione ext2 sulla chiavetta per contenere la chiave di cifratura della root:

fdisk /dev/sda
n p 1 <return> +100M
w q
mkfs.ext2 /dev/sda1
mkdir /tmp/usbmnt/ && mount -t ext2 /dev/sda1 /tmp/usbmnt/

- Creare la encryption key di **/** (il completamento dell'operazione puo' richiedere molto tempo, le attivita' che usano tastiera, mouse e dischi rigidi possono velocizzare molto il processo):

```
umask 077
head -c 3705 /dev/random | uuencode -m - | head -n 66 | tail -n 65 \
| gpg --symmetric -a --s2k-count 8388608 > /tmp/usbmnt/rootkey.gpg
```

- Creare il volume logico e il filesystem di **/**:

lvcreate -L 2G -n lvRoot vgSystem
losetup -e AES256 -K /tmp/usbmnt/rootkey.gpg /dev/loop1 /dev/mapper/vgSystem-lvRoot
mkfs -t ext3 /dev/loop1
losetup -d /dev/loop1

Copia dei file nella nuova root cifrata

- Copiare i file di sistema nella nuova root cifrata:

mount /dev/loop1 /mnt
mkdir /mnt/boot /mnt/proc /mnt/sys /mnt/mnt
cp -av /bin/ /cdrom /etc/ /initrd.img /lib/ /media/ /opt/ /root/ /selinux/ /usr/ /vmlinuz /dev/ /home/ /initrd.img.old /sbin/ /srv/ /tmp/ /var/ /vmlinuz.old /mnt/

- Modificare fstab nel fs cifrato:

cp /mnt/etc/fstab /mnt/etc/fstab.noloop
vim /mnt/etc/fstab

File: /mnt/etc/fstab

```
proc                        /proc          proc    defaults               0       0
/dev/loop1                  /              ext3    defaults,errors=remount-ro 0    1
#/dev/mapper/vgSystem-lvRescue  /           ext3    errors=remount-ro      0       1
/dev/hda1                   /boot          ext2    defaults               0       2
/dev/mapper/vgSystem-lvSwap     none           swap    sw,loop=/dev/loop0,encryption=AES256
#/dev/mapper/vgSystem-lvSwap none           swap    sw             0       0
#/dev/hdc                   /media/cdrom0   udf,iso9660 user,noauto        0       0
/dev/hdc                    /media/cdrom0   iso9660 user,noauto            0       0
/dev/fd0                    /media/floppy0  auto    rw,user,noauto         0       0
```

Creazione di /usr e /var cifrati

- Creare e inizializzare il volume logico per **/usr**:

lvcreate -L 5G -n lvUsr vgSystem
**head -c 15 /dev/urandom | uuencode -m - | head -n 2 | tail -n 1 **
| losetup -p 0 -e AES128 /dev/loop2 /dev/mapper/vgSystem-lvUsr
dd if=/dev/zero of=/dev/loop2 bs=4k conv=notrunc 2>/dev/null
losetup -d /dev/loop2

- Creare una directory nel filesystem di root cifrato per contenere le encryption key di **/usr** e **/var**:

mkdir /mnt/etc/crypto

- Creare la encryption key di **/usr** (il completamento dell'operazione puo' richiedere molto tempo, le attivita' che usano tastiera, mouse e dischi rigidi possono velocizzare molto il processo). A differenza della chiave di **/**, questa e' in chiaro anziche' cifrata via gpg:

umask 077
head -c 3705 /dev/random | uuencode -m - | head -n 66 | tail -n 65 > /mnt/etc/crypto/usrkey

- Creare il filesystem di **/usr** e copiarvi i contenuti:

losetup -e AES256 -P /mnt/etc/crypto/usrkey /dev/loop2 /dev/mapper/vgSystem-lvUsr
mkfs.ext3 /dev/loop2
mount -t ext3 /dev/loop2 /mnt/usr/
cd /usr/ && cp -av . /mnt/usr/

- Creare e inizializzare il volume logico per **/var**:

lvcreate -L 3G -n lvVar vgSystem
**head -c 15 /dev/urandom | uuencode -m - | head -n 2 | tail -n 1 **
| losetup -p 0 -e AES128 /dev/loop3 /dev/mapper/vgSystem-lvVar
dd if=/dev/zero of=/dev/loop3 bs=4k conv=notrunc 2>/dev/null
losetup -d /dev/loop3

- Creare la encryption key di **/var** (il completamento dell'operazione puo' richiedere molto tempo, le attivita' che usano tastiera, mouse e dischi rigidi possono velocizzare molto il processo). A differenza della chiave di **/**, questa e' in chiaro anziche' cifrata via gpg:

umask 077
head -c 3705 /dev/random | uuencode -m - | head -n 66 | tail -n 65 > /mnt/etc/crypto/varkey

- Creare il filesystem di **/var** e copiarvi i contenuti:

losetup -e AES256 -P /mnt/etc/crypto/varkey /dev/loop3 /dev/mapper/vgSystem-lvVar
mkfs.ext3 /dev/loop3
mount -t ext3 /dev/loop3 /mnt/var/
cd /var/ && cp -av . /mnt/var/

- Aggiungere ad fstab le nuove entry:

vim /mnt/etc/fstab

File: /mnt/etc/fstab

```
proc                        /proc          proc    defaults              0      0
loop1                       /              ext3    errors=remount-ro     0      1
/dev/hda1                   /boot          ext2    defaults              0      2
/dev/mapper/vgSystem-lvSwap     none           swap    sw,loop=/dev/loop0,encryption=AES256
/dev/mapper/vgSystem-lvUsr      /usr           ext3    nodev,loop=/dev/loop2,encryption=AES256,cleartextkey=/etc/crypto/usrkey       0      0
/dev/mapper/vgSystem-lvVar      /var           ext3    loop=/dev/loop3,encryption=AES256,cleartextkey=/etc/crypto/varkey       0      0
#/dev/mapper/vgSystem-lvSwap none           swap    sw              0      0
#/dev/hdc                    /media/cdrom0  udf,iso9660 user,noauto        0      0
/dev/hdc                    /media/cdrom0  iso9660 user,noauto           0      0
/dev/fd0                    /media/floppy0 auto    rw,user,noauto        0      0
```

Creazione di /tmp cifrata

- Creare il volume logico per **/tmp**:
  lvcreate -L 4G -n lvTmp vgSystem
- Aggiungere ad fstab la nuova entry, selezionando la modalita' a chiave random cosi' che ad ogni mount di **/tmp** il volume sia inizializzato con un nuovo filesystem ext2 cifrato con una chiave casuale:
  vim /mnt/etc/fstab

File: /mnt/etc/fstab

```
proc                        /proc          proc    defaults              0      0
loop1                       /              ext3    errors=remount-ro     0      1
/dev/hda1                   /boot          ext2    defaults              0      2
/dev/mapper/vgSystem-lvSwap     none           swap    sw,loop=/dev/loop0,encryption=AES256
/dev/mapper/vgSystem-lvUsr      /usr           ext3    nodev,loop=/dev/loop2,encryption=AES256,cleartextkey=/etc/crypto/usrkey       0      0
/dev/mapper/vgSystem-lvVar      /var           ext3    loop=/dev/loop3,encryption=AES256,cleartextkey=/etc/crypto/varkey       0      0
/dev/mapper/vgSystem-lvTmp      /tmp           ext2    nodev,noexec,nosuid,loop=/dev/loop4,encryption=AES256,phash=random/1777          0      0
#/dev/mapper/vgSystem-lvSwap none           swap    sw              0      0
#/dev/hdc                    /media/cdrom0  udf,iso9660 user,noauto        0      0
/dev/hdc                    /media/cdrom0  iso9660 user,noauto           0      0
/dev/fd0                    /media/floppy0 auto    rw,user,noauto        0      0
```

Personalizzazioni aggiuntive

- Seguono alcune configurazioni tipiche che possono essere adattate in base alle proprie specifiche esigenze.

Creazione di /srv cifrata con chiave in chiaro

- E' desiderabile creare un filesystem cifrato in un volume logico dedicato dove ospitare i dati dei servizi (apache, samba,..) eseguiti automaticamente all'avvio del server. Il filesystem e' montato automaticamente all'avvio del sistema per cui la sua chiave di cifratura risiede in chiaro in **/etc/crypto**:
  lvcreate -L50G -nlvSrv vgSystem
  **head -c 15 /dev/urandom | uuencode -m - | head -n 2 | tail -n 1 **
  | losetup -p 0 -e AES256 /dev/loop5 /dev/mapper/vgSystem-lvSrv
  dd if=/dev/zero of=/dev/loop5 bs=4k conv=notrunc 2>/dev/null
  losetup -d /dev/loop5
  head -c 3705 /dev/random | uuencode -m - | head -n 66 | tail -n 65 > /etc/crypto/srvkey
  chmod 400 /etc/crypto/srvkey
- Aggiungere in **/etc/fstab**:
  mount /srv:

Command: vim /etc/fstab

```
/dev/mapper/vgSystem-lvStorage /storage ext3 noexec,nosuid,loop=/dev/loop5,encryption=AES256,cleartextkey=/etc/crypto/srvkey 0 0
```

- Creare il filesystem e montarlo:
  losetup -F /dev/loop5
  mkfs.ext3 /dev/loop5
  losetup -d /dev/loop5
  mount /srv

Creazione di /data cifrata con chiave protetta da GPG

- Nel caso si vogliano ospitare sul server dati importanti ai quali e' necessario accedere solo saltuariamente, e' preferibile memorizzarli in un filesystem separato, cifrato con chiave protetta da GPG e non montato automaticamente. E' buona regola infatti tenere montati solo i filesystem contenenti dati a cui e' effettivamente necessario accedere frequentemente cosi' da minimizzare il rischio di corruzione dei file nel caso si sbagli a digitare un comando o vi sia un malfunzionamento di qualche programma. Inoltre questo mitiga anche il rischio rappresentato da un attaccante che abbia accesso fisico al server o che vi ottenga accesso non autorizzato via rete: potra' solo consultare i file dei filesystem montati in quel momento, mentre quelli all'interno di volumi cifrati con chiave protetta da GPG non saranno immediatamente rivelati.
- Per creare un filesystem in un volume logico dedicato cifrato con chiave protetta da GPG, si puo' ad esempio procedere come segue:
  lvcreate -L10G -nlvData vgSystem
  **head -c 15 /dev/urandom | uuencode -m - | head -n 2 | tail -n 1 **
  | losetup -p 0 -e AES256 /dev/loop6 /dev/mapper/vgSystem-lvData
  dd if=/dev/zero of=/dev/loop6 bs=4k conv=notrunc 2>/dev/null
  losetup -d /dev/loop6
  **head -c 3705 /dev/random | uuencode -m - | head -n 66 | tail -n 65 **
  | gpg --s2k-count 8388608 --symmetric -a >/etc/crypto/datakey.gpg
  chmod 400 /etc/crypto/datakey.gpg
- Aggiungere in **/etc/fstab**:

Command: vim /etc/fstab

```
/dev/mapper/vgSystem-lvData /data ext3 nodev,noexec,nosuid,noauto,loop=/dev/loop6,encryption=AES256,gpgkey=/etc/crypto/datakey.gpg 0 0
```

- Creare il filesystem e montarlo:
  losetup -F /dev/loop6
  mkfs.ext3 /dev/loop6
  losetup -d /dev/loop6
  mkdir /data
  mount /data

Rimozione di lvRescue

- Per rimuovere lvRescue:
  lvremove /dev/vgSystem/lvRescue
- Oppure per rinominarlo ed espanderne il filesystem:
  lvrename /dev/vgSystem/lvRescue /dev/vgSystem/lvNotEncrypted
  lvextend -L10G /dev/vgSystem/lvNotEncrypted
  fsck.ext3 /dev/vgSystem/lvNotEncrypted
  tune2fs -O ^has_journal /dev/vgSystem/lvNotEncrypted
  fsck.ext2 -f /dev/vgSystem/lvNotEncrypted
  resize2fs /dev/vgSystem/lvNotEncrypted 10G
  tune2fs -j /dev/vgSystem/lvNotEncrypted
  mount /dev/vgSystem/lvNotEncrypted /mnt

Risorse utili

- Documentazione
  - Scegliere una passphrase forte
  - LVM e loop-AES How-To
  - Funzionamento di initrd
  - File necessari in /boot per attivare LVM
  - Avviare un sistema con root cifrata loop-AES da una chiavetta USB senza tabella delle partizioni presente sul disco fisso -- paranoia estrema!
  - Montare una partizione cifrata con il file della chiave in chiaro
  - Cambiare la passphrase di una chiave senza scrivere in chiaro la chiave stessa su disco
  - Compilazione del kernel su Debian Etch
- Software
  - Archivio sorgenti kernel Linux
  - Archivio pacchetto Debian sorgenti loop-AES
  - Archivio sorgenti AESpipe
  - Archivio sorgenti GnuPG
  - Archivio sorgenti dash
  - Archivio sorgenti util-linux-ng
  - Archivio sorgenti loop-AES