Encrypted Windows 10 & Ubuntu Dual boot

This document describes how to dual boot windows 10 and Ubuntu and enable "full system encryption" for both. Windows 10 will be encrypted using BitLocker and Ubuntu will be installed on an LUKS encrypted LVM partition. LUKS is implemented in Linux using dm-crypt, a disk encryption subsystem in Linux kernel. dm-cypt can be use to encrypt whole hard disks, partition or volume. The setup described here assumes windows is installed using UEFI mode.

Windows 10 installation

If no operating system is installed, windows 10 is first installed on a partition of desired size. BitLocker is then enabled.

Assuming windows 10 is already installed and BitLocker encryption enabled the windows partition can be shrunk using windows Disk Utilities. The resulting unallocated space is used to install Linux.

Linux installation

The setup described here involves setting up LVM on a LUKS encrypted partition. LVM allows all encrypted Linux partition (home, /, swap) to share a single key.

Boot into Ubuntu live cd/usb and choose to try Ubuntu.

Setting up a LUKS encrypted partition

The **/boot** partition for Linux is not encrypted. The boot partition of Windows encrypted using BitLocker are also not encrypted.

fdisk is going to be used to create **boot** and another partition housing the rest of Ubuntu installation.

Assuming that windows occupy partitions sda1 to sda5 (this is the likely setup for a UEFI installed Windows 10). Create the intended boot partition:

```
fdisk /dev/sda
```

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (6-9)
p
Partition number (6-9): 6
First cylinder (****-*****, default 1):<RETURN>
Using default value ****
Last cylinder or +size or +sizeM or +sizeK (**=***-***, default *****): +1G
w
```

Now create another partition also using fdisk that will hold the rest of Ubuntu. Assuming that this partition is /dev/sda7 use **cryptsetup** to encrypt the partition with LUKS.

```
sudo cryptsetup luksFormat --hash=sha512 --key-size=512 \
--cipher=aes-xts-plain64 --verify-passphrase /dev/sda7
```

Open the encrypted partition so that LVM logical partition can be set up. you will be asked for the passphrase. we have chosen to call this partition "CryptDisk"

```
sudo cryptsetup luksOpen /dev/sda7 CryptDisk
```

Now set up LVM logical partitions (home, /, swap). For this demonstration, root is 10GB, swap is 2GB and home is the rest of the space in the LVM partition. Creating LVM logical partition involves creating a physical volume using **pvcreate**, creating a volume group with **vgcreate** and then creating logical volumes belonging to a logical group using **lvcreate**.

```
sudo pvcreate /dev/mapper/CryptDisk
sudo vgcreate vg0 /dev/mapper/CryptDisk
sudo lvcreate -n swap -L 2G vg0
sudo lvcreate -n root -L 10G vg0
sudo lvcreate -n home -l +100%FREE vg0
```

Installing Ubuntu on LVM partition

Now you're ready to install. Click on "Install Ubuntu". When you get to the "Installation type" portion of the install, choose the "Something else" option. Then manually assign the /dev/mapper/vg0-* partitions as you would like to have the configured (eg mount /home on /dev/mapper/vg0-home). Don't forget to set /dev/sda6 as /boot. the /boot partition must not be encrypted. If it is, we won't be able to boot. Change the "Device for boot loader installation" to /dev/sda, and continue with installation. When installation is complete, don't reboot! Choose the option to "Continue Testing".

Re-installing grub and making Ubuntu aware of the encrypted partition

chroot into the installed Ubuntu system

```
sudo mount /dev/mapper/vg0-root /mnt
sudo mount /dev/sda6 /mnt/boot
sudo mount /dev/sda2 /mnt/boot/efi #find out the actual efi partition
# bind the dev, proc, and sys to the new root
sudo mount --bind /dev /mnt/dev
sudo mount --bind /proc /mnt/proc
sudo mount --bind /sys /mnt/sys
# you may need to create /etc/resolv.conf to specify DNS/nameserver server
sudo chroot /mnt
```

Find the uuid of the luks encrypted partition

```
blkid /dev/sda7
```

Add a new file /etc/crypttab with the content (where CryptDisk is just a name, I used the same here as in the luksOpen above). eg:

```
# <target name> <source device>     <key file>  <options>
CryptDisk UUID=f2ee83ef-a828-4a84-a150-2ffd781b495a none luks
```

Re-install grub and update initramfs

```
apt-get install --reinstall grub-efi-amd64
grub-install
update-initramfs -k all -c -v
update-grub
```

Installation in now complete. Given that the windows EFI partition has been edited, windows will ask for Bit Locker recovery key the first time you log into windows.

Often the computer will boot into windows and not offer an option to choose Ubuntu. When that happens the **bootmgr** option in windows need to be edited. This is done by booting into windows, opening the CMD shell as admin and setting the options using **bcdedit** as follows.

```
bcdedit /set "{bootmgr}" path \EFI\ubuntu\grubx64.efi
# dont quote {bootmgr} if you get an error running the command
```

LUKS recovery

It is important to back up the LUKS headers which can get damaged during kernel upgrade. If the headers are damaged and can not be restored it is not possible to recover encrypted data.

Backing up LUKS headers

LUKS headers can be exported to a file for same storage on another device as follows

```
cryptsetup luksHeaderBackup <device> --header-backup-file <file>
```

If the headers are damaged they can be restored as follows:

```
cryptsetup luksHeaderRestore <device> --header-backup-file <file>
```

Adding additional keys

LUKS has 8 key slots that allow having a total of 8 keys. The 7 keys can be used for backup.

To view which key slots are already filled:

```
cryptsetup luksDump /dev/sda7
```

A key slot can be filled with a passphrase or a file. For instance, we can generate a random file and use it as a recover key as follows:

```
dd bs=512 count=4 if=/dev/random of=/etc/mykeyfile #generate key file
cryptsetup luksAddKey /dev/sda7 /etc/mykeyfile #add keyfile
```

You will be asked to enter an already existing passphrase before adding a new key. The keyfile can then be stored in another secure location and used for recovery. To unlock the LUKS using a keyfile:

```
cryptsetup open /dev/sda7 <dm_name> --key-file /etc/mykeyfile
```