A startup key to protect my data 1.0

How to have a fully secure laptop

Step by step installation guide

Xavier Berger

A startup key to protect my data 1.0 How to have a fully secure laptop
Step by step installation guide
Edition 1.0

Author Xavier Berger berger.xavier@gmail.com

DISCLAIMER:

The information provided on this document comes without warranty of any kind, without even the implied warranty of merchantability or fitness for a particular purpose and is distributed AS IS.

Every effort has been made to provide the information as accurate as possible. The information may be incomplete, may contain errors or may have become out of date. The use of this information described herein is your responsibility, and to use it in your own environments do so at your own risk.

This document is a set of article explaining how to install, configure and use a fully secure Linux environment. This doc are applicable for laptop, desktop as well as for servers.

About the author

Xavier Berger is working as Solution Architect in a telecom company. He is a specialist in Linux and network deployment. Xavier enjoys hiking, geocaching, skiing and spending time with his family. His web site is: *http://xberger.free.fr*

[1] http://creativecommons.org/licenses/by-sa/3.0/

Preface

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts*[1] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keycaps and key combinations. For example:

> To see the contents of the file **`my_next_bestselling_novel`** in your current working directory, enter the **`cat my_next_bestselling_novel`** command at the shell prompt and press **`Enter`** to execute the command.

The above includes a file name, a shell command and a keycap, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from keycaps by the hyphen connecting each part of a key combination. For example:

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**+**`Alt`**+**`F2`** to switch to the first virtual terminal. Press **`Ctrl`**+**`Alt`**+**`F1`** to return to your X-Windows session.

The first paragraph highlights the particular keycap to press. The second highlights two key combinations (each a set of three keycaps with each set pressed simultaneously).

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **`mono-spaced bold`**. For example:

> File-related classes include **`filesystem`** for file systems, **`file`** for files, and **`dir`** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

> Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click

[1] https://fedorahosted.org/liberation-fonts/

> **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications** → **Accessories** → **Character Map** from the main menu bar. Next, choose **Search** → **Find…** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

Mono-spaced Bold Italic or ***Proportional Bold Italic***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type **ssh *username*@*domain.name*** at a shell prompt. If the remote machine is **example.com** and your username on that machine is john, type **ssh john@example.com**.
>
> The **mount -o remount *file-system*** command remounts the named file system. For example, to remount the **/home** file system, the command is **mount -o remount /home**.
>
> To see the version of a currently installed package, use the **rpm -q *package*** command. It will return a result as follows: ***package-version-release***.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

> Publican is a *DocBook* publishing system.

1.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **mono-spaced roman** and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svnbooks_tests  Desktop1
  downloads      images  notes  scripts  svgs
  stuff  svnbooks_tests  Desktop1  downloads      images  notes
```

Source-code listings are also set in **mono-spaced roman** but add syntax highlighting as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;
```

```
public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }
}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled 'Important' will not cause data loss but may cause irritation and frustration.

Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

2. Feedback

This document comes without warranty of any kind. But every effort has been made to provide the information as accurate as possible. I welcome emails from any readers with comments, suggestions, and corrections.

Introduction

In our modern society, computers become tools hosting a lot of private information. Losing these data or displaying these data to the public could have a big impact for the owner.

In this article will will then see how to:

1. **Prevent the computer to boot on the secured system without a startup USB key**

   The first barrier will consist to prevent the PC to boot when it is not into the hand of its owner. To achieve this goal, we will "split" the hardware in two pieces. When these two pieces are joined, the computer can be used (and boot). If not, the computer will not been able start. As many people have to purchase Windows with a laptop, will see how give an access to this OS when the key is not present. This could be useful if you want lend your PC to a friend to give him an access to the Internet.

2. **Prevent the data from an unwanted access**

   If a person can access to my disk he should not have access to the data. The file system and swap contain or can contain personal data. We will see how to encrypt the data at the level of the partition and keep our data in a safe place.

3. **Prevent data loss**

   To prevent data loss, doing regular backup is something mandatory but if the backup storage is located in the same building as the computer it may be robbed/destroyed as well. To prevent data loss, we need to externalize the data. The cloud is a good solution for such an externalization. It could be done in real time and doesn't require any discipline from the end user. To ensure the privacy of the data in such area, we will also encrypt the data before synchronizing it to the cloud. The data in the cloud are mirroring the current data of a computer, we will see into a second article how to setup a dedicated secured server acting as a NAS and allowing to go back in time.

4. **Use the created USB key as a toolbox by adding live distributions**

   As we have in our hands an USB key we will see how to add additional live distribution that could be useful for troubleshooting or rescue.

Audience

This article is designed for Linux advanced user or curious beginners. It has been written as an installation procedure and is explaining:

- how to install a custom operating system from scratch.
- how to modify the disk structure of a computer.
- how to modify the boot sequence.
- how to configure pam to simplify the access to the secured data.

Some action describe here may be risky and you may even lose your data in case of mistake. Before executing a command described in this article, you should understand what the command will do and what will be the impact on your system. The procedure described in this article has been tested several time and is currently in use into the computer of the writer of this article.

Installation

3.1. Prerequisite

To apply the procedure described below you will need the following items:

- A target computer (with or without an existing OS running)
- One USB key (min > 1GB, recommended > 4GB) which will be our startup key.
- Another USB key (min > 1GB, recommended > 4GB) or 2 blank CD-ROMs
- A computer with an operating system running (It could be the target)

3.2. Preparation installation media

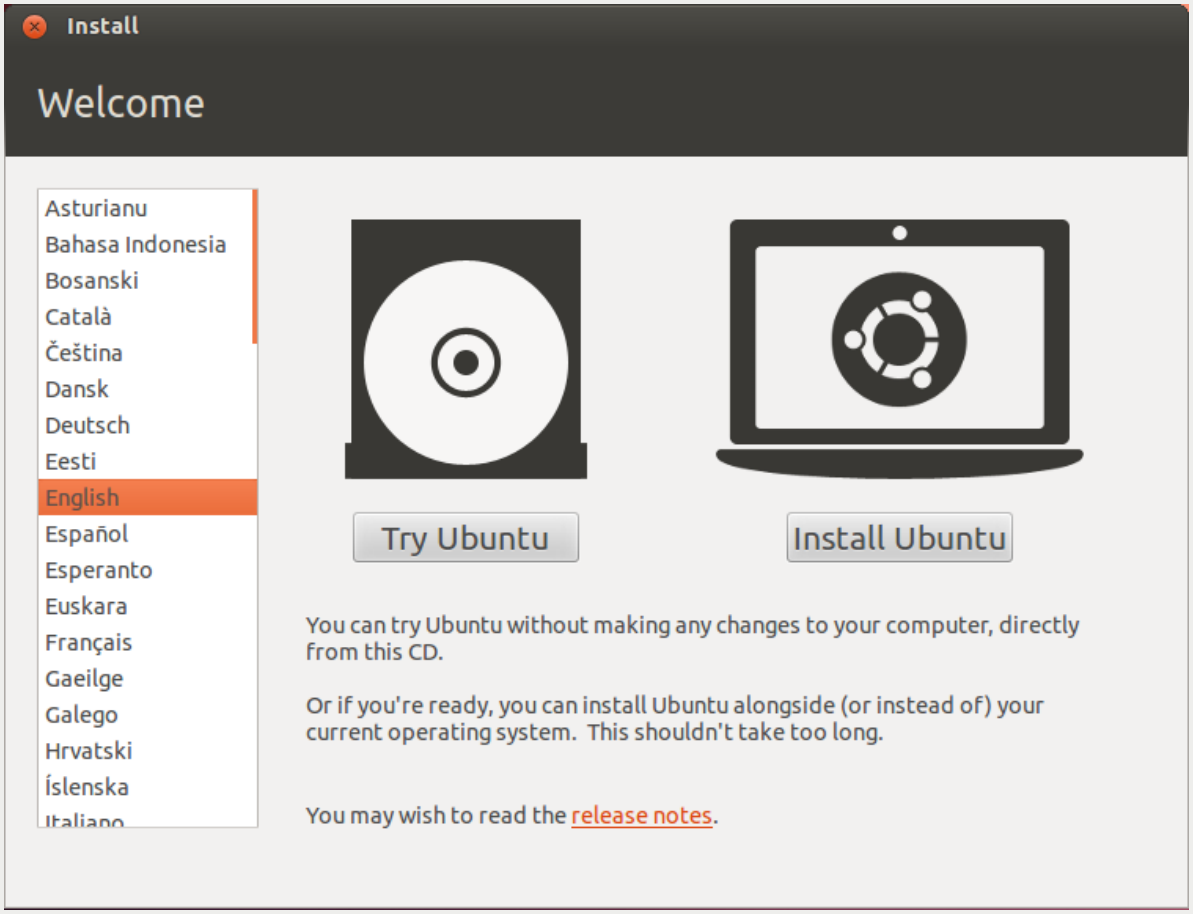
It is possible to install the secured PC using CD-ROMs or using USB keys if the target computer doesn't have CD-ROM reader. In the chapter we do explain the both solution. You should choose the one that match the best your requirements.

3.2.1. Install from CD-Rom

Download **`ubuntu-12.04.1-desktop-i386.iso`** from *http://releases.ubuntu.com/precise/* and burn it into the first CD-ROM.

Note

When you will boot on this media, you will see a welcome screen, select **`Try Ubuntu`** to boot into a live session.



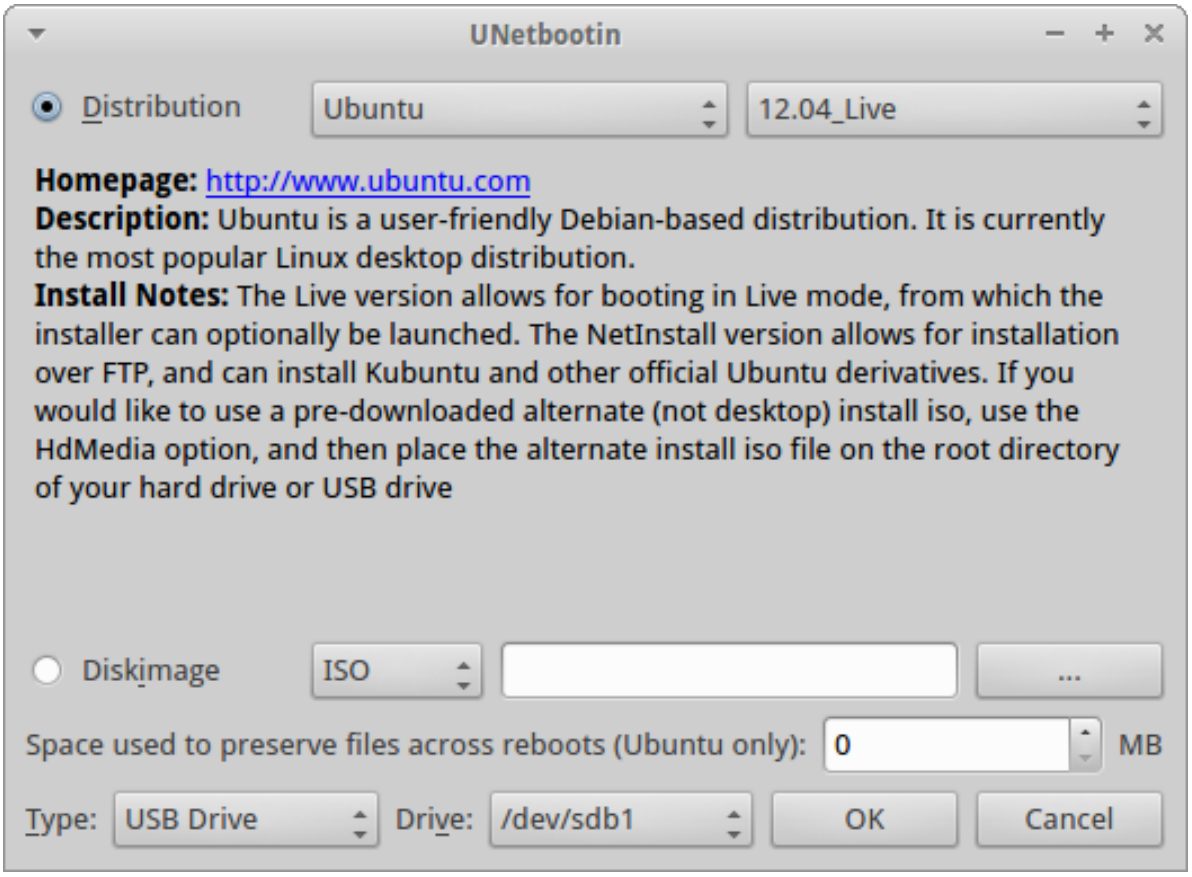
3.2.2. Install from USB Keys - part 1

First we need to install **UnetBootIn**. The latest version of **UnetBootIn** for Ubuntu 12.xx is available in launchpad *https://launchpad.net/ubuntu/quantal/i386/unetbootin/575-1*

Download the file **`unetbootin_575-1_i386.deb`** and install it with the following command:

```
sudo dpkg -i unetbootin_575-1_i386.deb
```

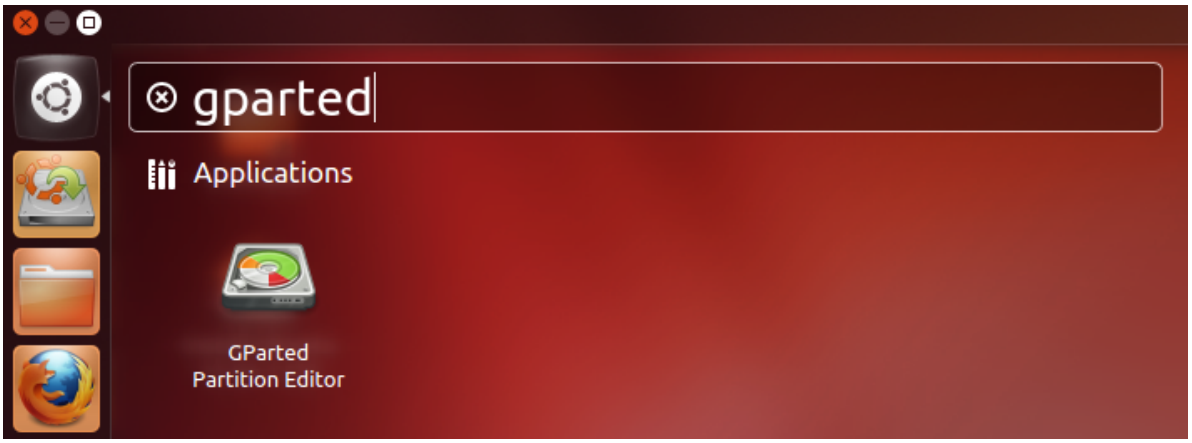
Start **UnetBootIn** and plug the second USB key. Then prepare a bootable media by delecting **`Ubuntu`** and **`12.04_Live`** into **Distribution** menu.



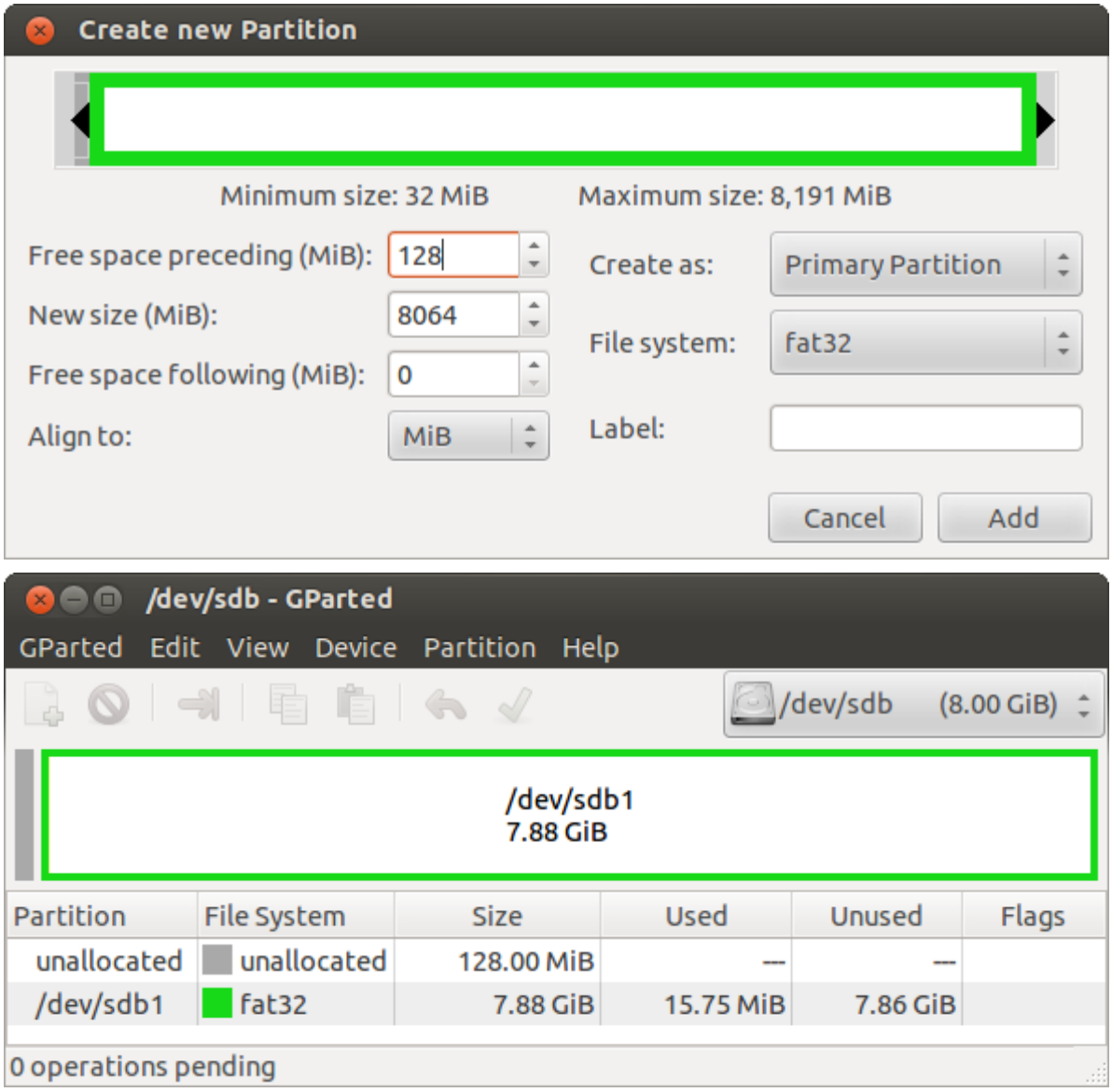
3.2.3. Preparation of the startup key

Boot on **Ubuntu 12.04.1 Desktop Live** media just created (from USB key or CD-ROM). Plug the USB key which will become our startup key.

Once the live system has booted, start **gparted** to prepare the USB key.



Create a FAT32 partition preceded by a free space of 128MB. (This space will be used later by the secured operating system.)



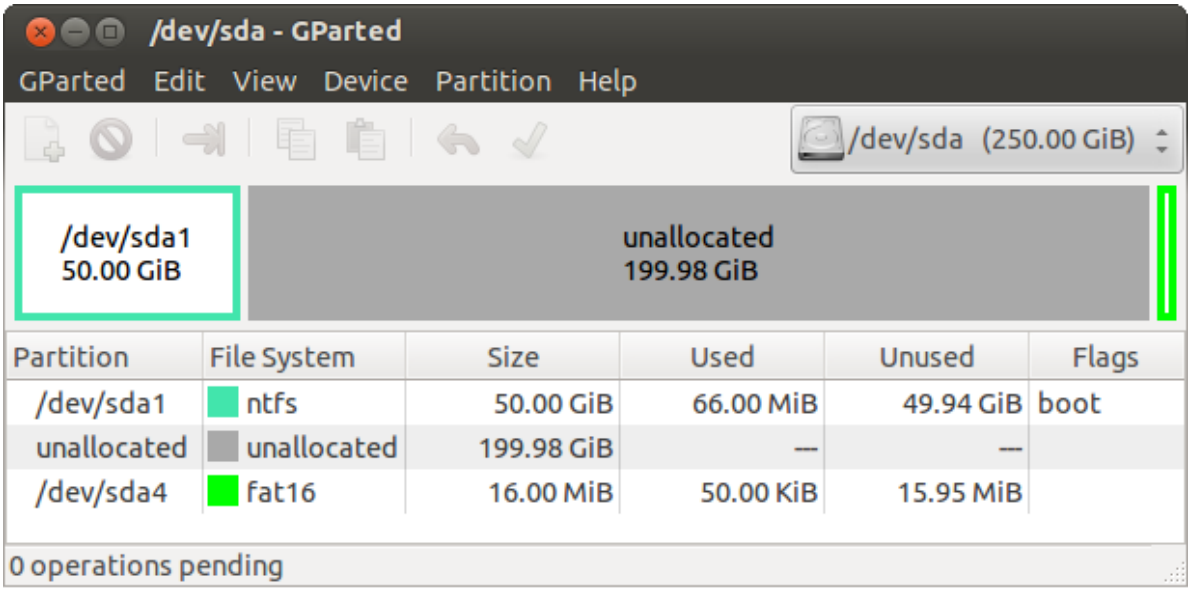
If your computer doesn't have an operating system installed, skip the next step of this chapter and continue with the next section.

You had to purchase Microsoft™ Windows™ with the laptop, we will keep it in the machine and let it available (if you would like to lend you PC to your friends).

First perform a backup of Windows™ using the tools provided by your manufacturer... you may need or will have to restore Windows™ once the partition resizing will be done.

Resize Windows™ partition to have space for the real operating system.

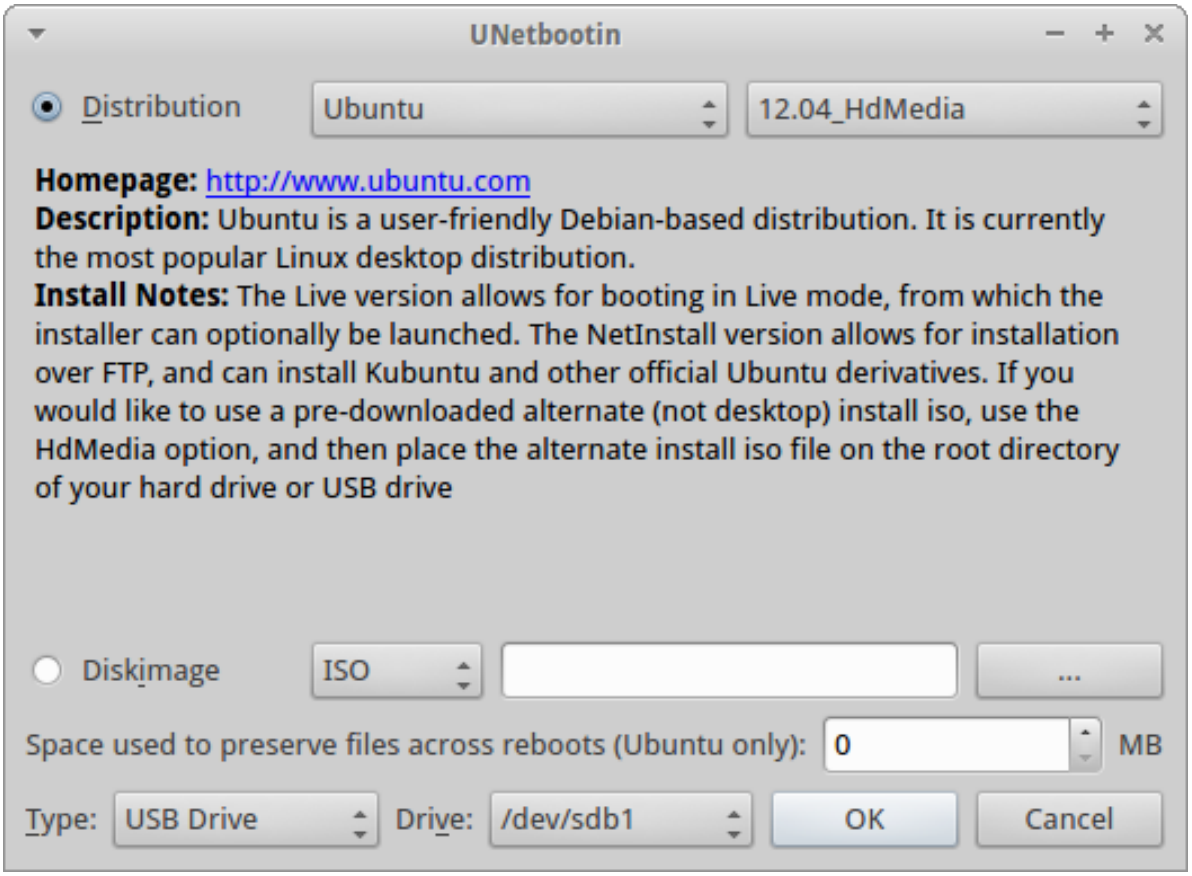
- Boot on the freshly created live media (USB stick or CD-ROM)
- Start **gparted**
- Select the disk of the computer **`/dev/sda`**
- Resize the windows partition to 50GB, it should be enough if time to time you need to use this OS
- Remove unneeded partition to create free space that will be use to install Linux



3.2.4. Installation from USB Keys - part 2

Download **`ubuntu-12.04.1-alternate-i386.iso`** from *http://releases.ubuntu.com/precise/* and copy it on the FAT32 partition of the unused usb stick.

Prepare a bootable USB key with **UnetBootIn** by selecting **`Ubuntu`** and **`12.04_HdMedia`**



3.3. Installation

During the installation we will configure the Operating System to encrypt the data stored in the disk. This encryption will ensure the security of the data. Longer is the key, better is the protection but longer is the time of encryption and decryption. In this article we choose the shortest proposed length for the key: *`AES 128bits`* to be fast and secure enough. A key of 128 bits give about 3,4 10^38 possibilities.

To understand how secure 128 bits keys are, you may read the analogy by Jon Callas at : *http://www.interesting-people.org/archives/interesting-people/200607/msg00058.html*[1]

“Imagine a computer that is the size of a grain of sand that can test keys against some encrypted data. Also imagine that it can test a key in the amount of time it takes light to cross it. Then consider a cluster of these computers, so many that if you covered the earth with them, they would cover the whole planet to the height of 1 meter. The cluster of computers would crack a 128-bit key on average in 1,000 years.”

Boot on **Ubuntu 12.04.1 Alternate** media just created (USB key or CD-ROM) and follow the instruction bellow to install the system.

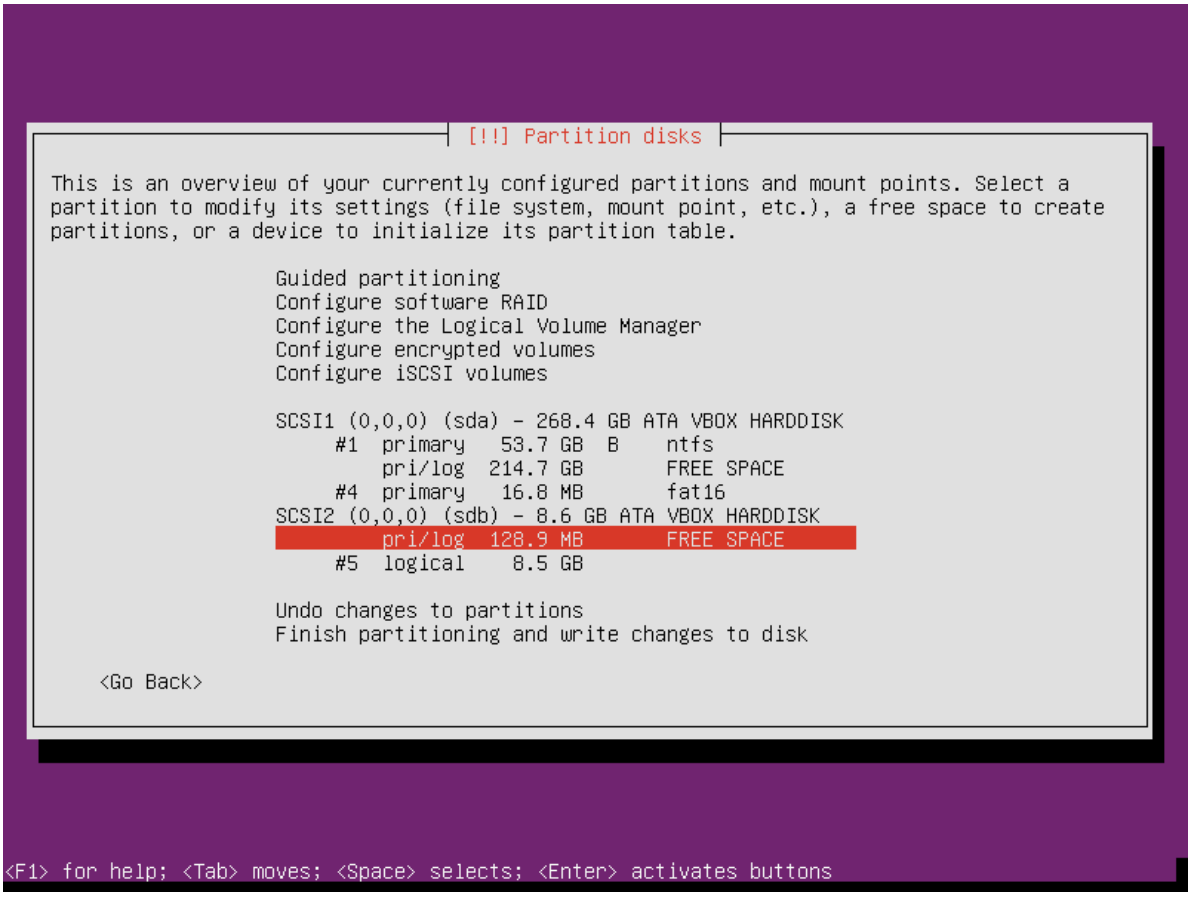
Note

The first screen of the installation process may differ depending on the media you choose for the installation. The procedure bellow has been written from the CD-ROM installation.

If you perform the installation from an USB key, the installation wizard will time to time invite you to umount **`/dev/sdb`**. **Do not umount it since this is our installation media.**

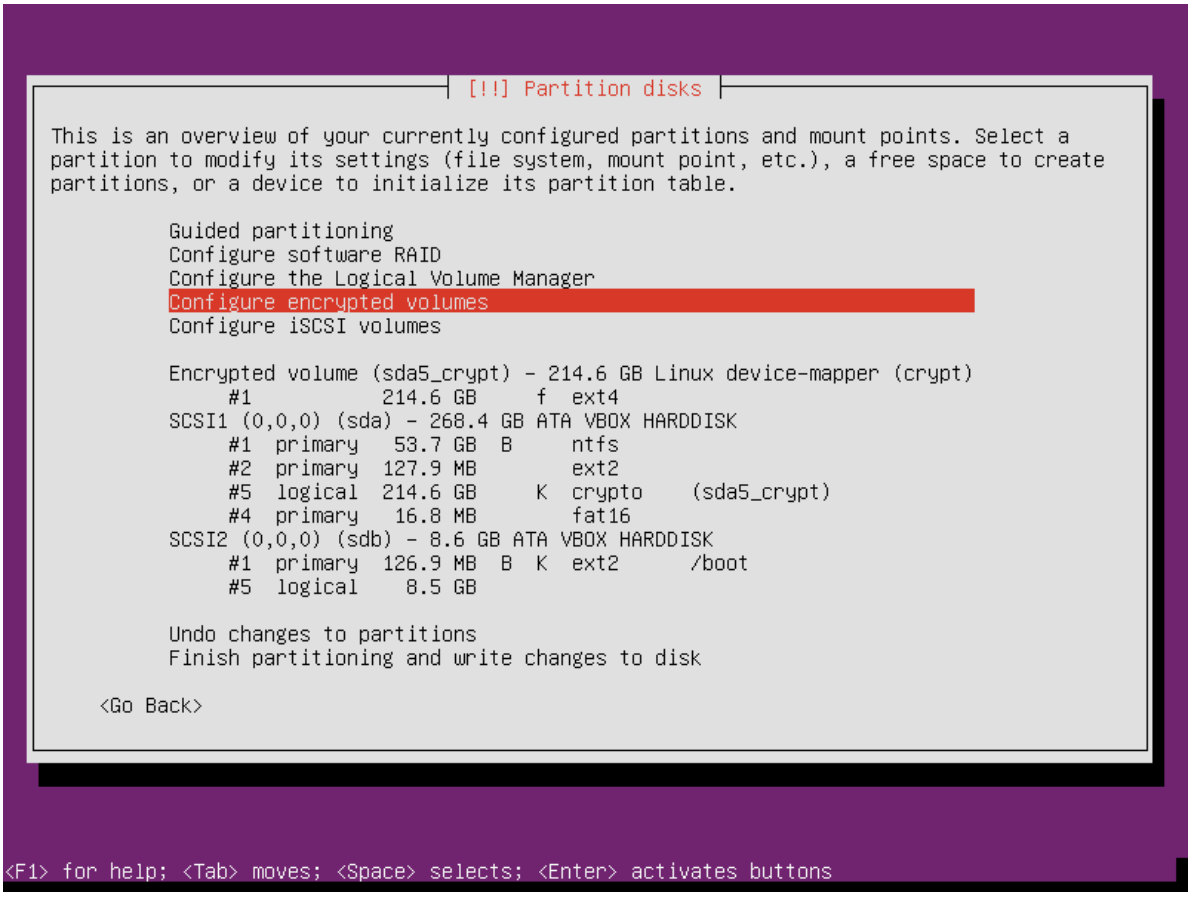
- Select the language to be used during the installation process: **`English`**.
- Select your location, configure locales, configure the keyboard
- Define hostname, user and password
- Do not chose to encrypt the home directory. We will encrypt all the partition.
- Set clock and timezone
- Partition disks: select a **`Manual`**

[1] http://www.interesting-people.org/archives/interesting-people/200607/msg00058.html



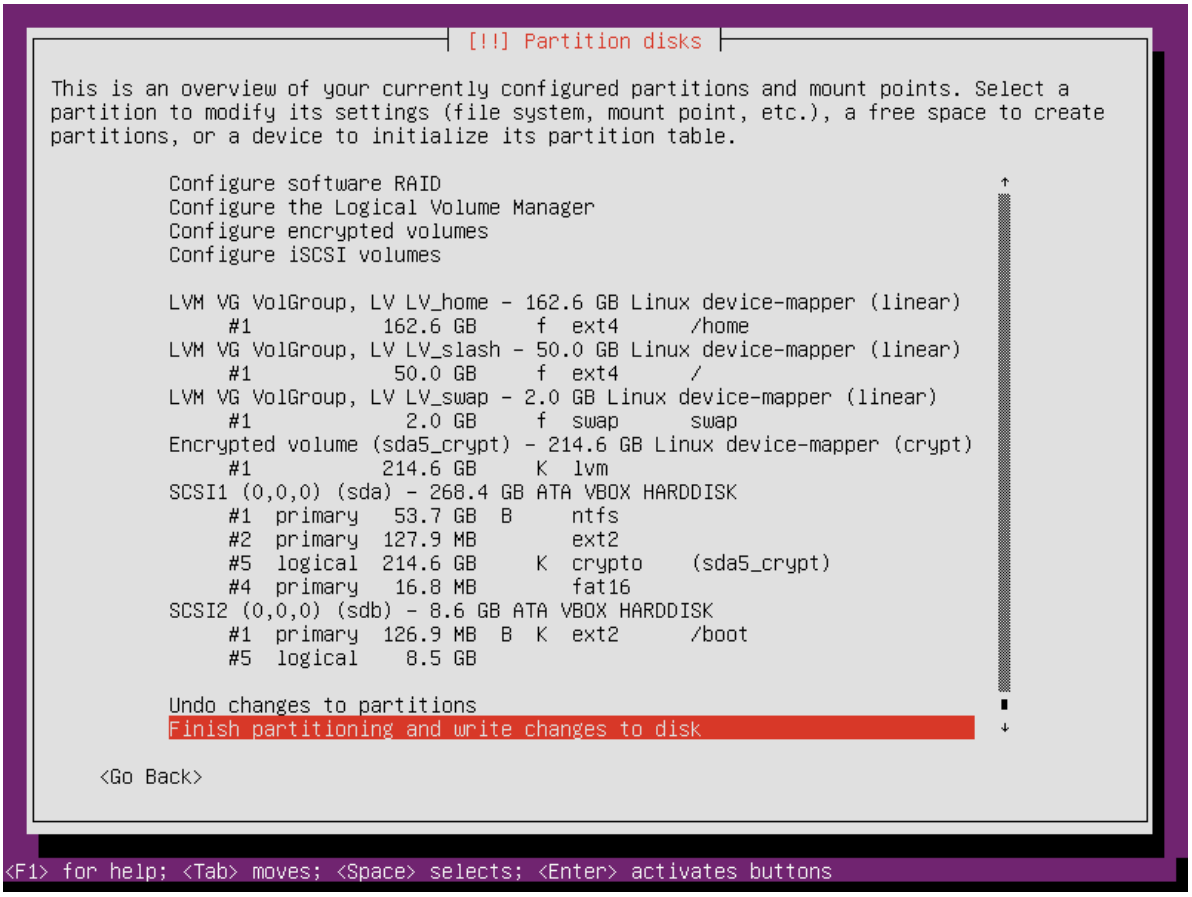
- **Create the /boot partition unencrypted**
  - Select the frees pace on sdb and press **`enter`**
  - Select: **`Create a new partition`**
  - Define the size: **`keep the proposed size`**
  - Type of the new partition: **`Primary`**
  - Use As: **`Ext2 file system`**
  - Mount point: **`/boot`**
  - Bootable flag: **`on`**
  - Select: **`Done setting up the partition`**
- **Create an Logical partition**
  - Select the free space on sda and press enter
  - Select **`Create a new partition`**
  - Define the size: **`128M`**
  - Type of the new partition: **`Primary`**
  - Location for new partition: **`Beginning`**
  - Use as: **`do not use`**

  - Select: Done setting up the partition
- **Create an Logical partition**
  - Select the frees pace on sda and press enter
  - Select **`Create a new partition`**
  - Define the size: **`keep the proposed size which should be the maximum space available`**
  - Type of the new partition: **`Logical`**
  - Use as: **`do not use`**
  - Select Done setting up the partition
- **Encrypt partition**
  - Select **`Configure encrypted volumes`**
  - Write the change to disk and configure encrypted volumes: **`Yes`**
  - Select **`Create Encrypted volumes`**
  - Select: **`[*] /dev/sda5`**
  - Key size: **`128`**
  - Done setting up the partition
  - Keep current partition layout and configure encrypted volume: **`Yes`**
  - Select **`Finish`**
  - Enter a passphrase twice



- **Create LVM group and volumes**
  - Select **`Configuring the Logical Volume Manager`**
  - Write change to disk and configure LVM : **`Yes`**
  - Select: **`Create volume group`**
    - Volume group name: **`VolGroup`**
    - Device for new volume group: **`[*] /dev/mapper/sda5_crypt`**
    - Keep current partition layout an configure LVM: **`Yes`**
  - Select: **`Create logical volume`**
    - Volume group: **`VolGroup`**
    - Logical volume name: **`LV_slash`**
    - logical volume size: **`50GB`**
  - Select: **`Create logical volume`**
    - Volume group: **`VolGroup`**
    - Logical volume name: **`LV_swap`**
    - logical volume size: **`2GB`**
  - Select: **`Create logical volume`**

    - Volume group: **`VolGroup`**
    - Logical volume name: **`LV_home`**
    - Logical volume size: keep the proposed size
    - Select Finish
- **Configure the mounting points of LVM volumes**
  - Select LV_home --> #1
    - Use as : **`Ext4 journaling file system`**
    - Mount point: **`/home`**
    - Options: **`[*] noatime`** (--> we don't care the know the last time the file has been read)
    - Select **`Done setting up the partition`**
  - Select LV_slash --> #1
    - Use as : **`Ext4 journaling file system`**
    - Mount point: **`/`**
    - Options: **`[*] noatime`** (--> we don't care the know the last time the file has been read)
    - Select **`Done setting up the partition`**
  - Select LV_swap --> #1
    - Use as : **`swap area`**
    - Select **`Done setting up the partition`**



- **Complete the disk configuration**
  - Finish partitioning and write changes to disk
  - Write changes to disks
  - Install GRUB in **`/dev/sdb`** (the key will then be required to boot on the secured system)
- **Complete the installation**

Congratulation, you have now a system where your data are encrypted and needing an external USB key to start. As we didn't touch the MBR of the internal disk, the previous operating system should continue to start as it did previously. It is now required to boot on the usb key to access to the secured area. Doing so, you will see the boot screen asking the password required to decrypt the disk.

We will see in the next chapter how to make our secured computer as easy to use as an unencrypted and unsecured computer and will configure it to ensure the sustainability of our data.

Configuration

In this section we will see how to customize the system to improve its usability and to protect our data from lose.

Warning

Unless specified, the command bellow have to be executed as root.

4.1. Use label for boot partition to simplify the startup key generation

Using label for USB key will allow us to duplicate this key and ensure that the system will recognize the copy as the booting device

Define the label for **/dev/sdb2** as *BOOT* (**/dev/sdb2** is the **/boot** partition)

```
e2label /dev/sdb2 BOOT
```

Update **/etc/fstab** to use label instead UUID

- replace **UUID=(...)** by **LABEL=BOOT**

Edit **/etc/default/grub** and uncomment the following line to not use UUID in **grub**.

- **GRUB_DISABLE_LINUX_UUID=true**

Append the bold lines in the file **/usr/lib/grub/grub-mkconfig_lib** to configure **grub** to use label if available in a volume:

```
if label="`${grub_probe} --device ${device} --target=fs_label 2> /dev/null`" ; then
  echo "search --no-floppy --label ${label} --set root"
elif fs_uuid="`${grub_probe} --device ${device} --target=fs_uuid 2> /dev/null`" ; then
  echo "search --no-floppy --fs-uuid --set ${fs_uuid}"
fi
```

Upgrade **grub** configuration files with the following command

```
update-grub
```

4.2. Add a 'keyfile' on USB key to activate the automatic decryption

We will now configure the system to decrypt the partition based on a file stored into the startup key. The computer will then recognize the startup key and decrypt the partition without asking a password anymore.

Create the file **keyfile** in **/boot** and change its access rights by executing the following commands:

```
dd if=/dev/urandom of=/bootkeyfile bs=512 count=4 chmod 400 /boot/keyfile
```

Add the new key into as a valid key to decrypt the disk.

```
cryptsetup luksAddKey /dev/sda5 keyfile
```

Note

The contents of the file is important, not the filename.

We will now configure the system to us the created keyfile to automatically decrypt the disk at startup. Edit **`/etc/crypttab`** modify the line

- **`sda5_crypt UUID=(...) none luks`**

as follow:

- **`sda5_crypt UUID=(...) /dev/disk/by-label/BOOT:/keyfile luks,keyscript=/lib/cryptsetup/scripts/passdev`**

Finally, update the initramfs:

```
update-initramfs -uv
```

To remove auto decryption and reactivate passphrase only, modify **`/etc/crypttab`** in the reverse order. The file should look like something like that:

- **`sda5_crypt UUID=(...) none luks`**

Finally, update the initramfs:

```
update-initramfs -uv
```

4.3. Booting from the main disk instead of startup key

Some BIOS don't really like to always try to boot on an external USB drive. In such a situation we will create a screen displayed at boot that will allow to select which operating system to startup. We will use BURG which is have nice looking graphical interface. This solution will modify the MBR of the HDD you have then to pay a particular attention on the action proposed in this chapter.

First, create symbolic links **`vmlinuz`** and **`initrd`** to the latest kernel and initrd image

```
cd /boot *
ln -s vmlinuz-... vmlinuz
ln -s initrd.img-... initrd.img
```

Warning

After every kernel upgrade, you will have to update these links pointing to the latest kernel

Mount the **/boot** partition of HDD:

```
umount /boot mount /dev/sda2 /boot
```

Install BURG on the system:

```
apt-get install python-software-properties
add-apt-repository ppa:n-muench/burg
apt-get update
apt-get install burg
```

Configure BURG to run from the internal disk:

- Accept default parameters
- Select **/dev/sda**

Edit **/etc/burg/30_osprober** and append the keyword **exit** at the beginning of the file.

Edit **/etc/burg.d/10_linux** and append the keyword **exit** before the last while.

Update custom menu to boot on Linux or Windows™:

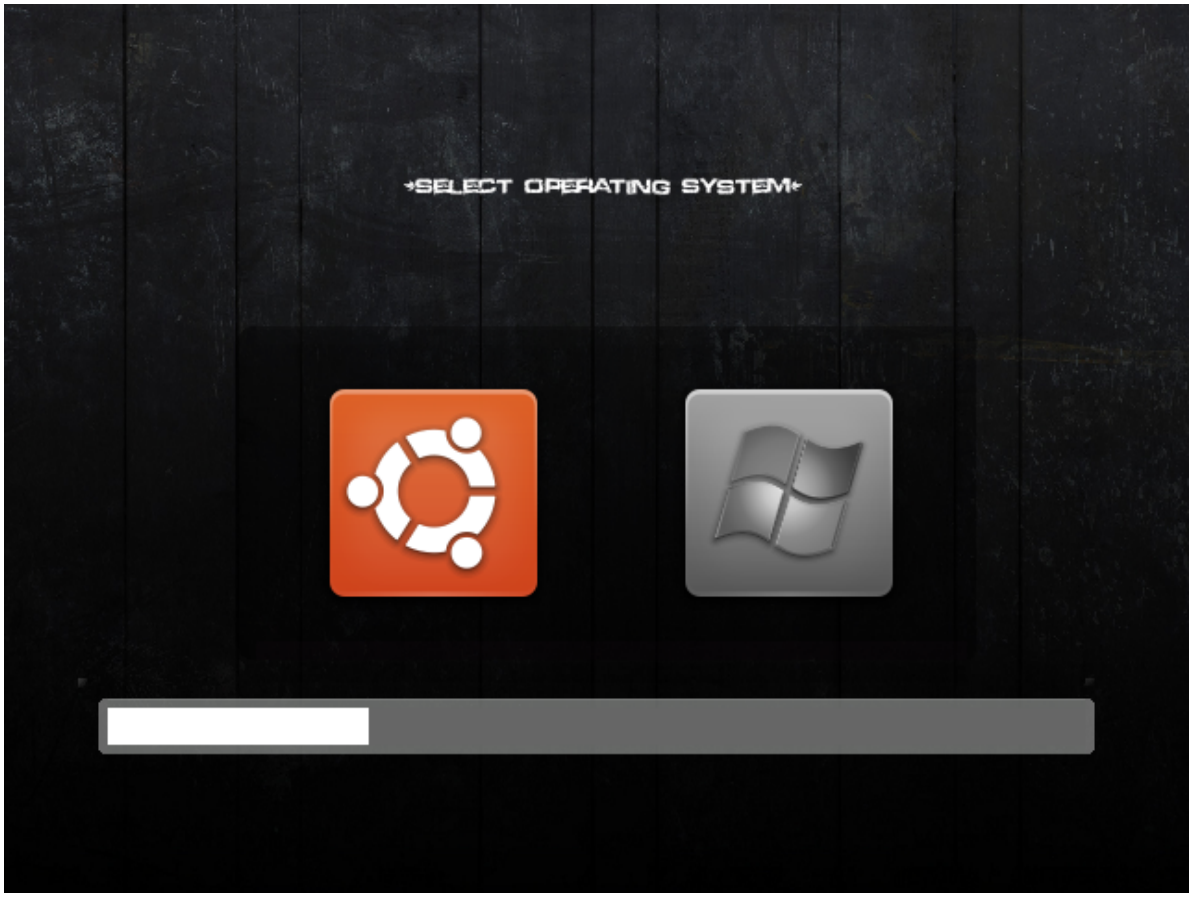
```
menuentry "Linux" --class ubuntu {  insmod ext2  set root=(hd1,2)  linux /vmlinuz quiet
 splash  initrd /initrd.img}menuentry "Windows" --class windows {  set root=(hd0,1)
 chainloader +1}
ubuntu {
 insmod ext2
 set root=(hd1,2)  linux /vmlinuz
 quiet splash
initrd
 /initrd.img}menuentry "Windows" --class
 windows {
 set root=(hd0,1)
```

Edit **/etc/default/burg** and uncomment **GRUB_DISABLE_LINUX_RECOVERY="true"**.

Apply the configuration:

```
update-burg
```

Here is the screen that will be displayed at startup:



Note

Note that this boot screen doesn't propose recovery mode.

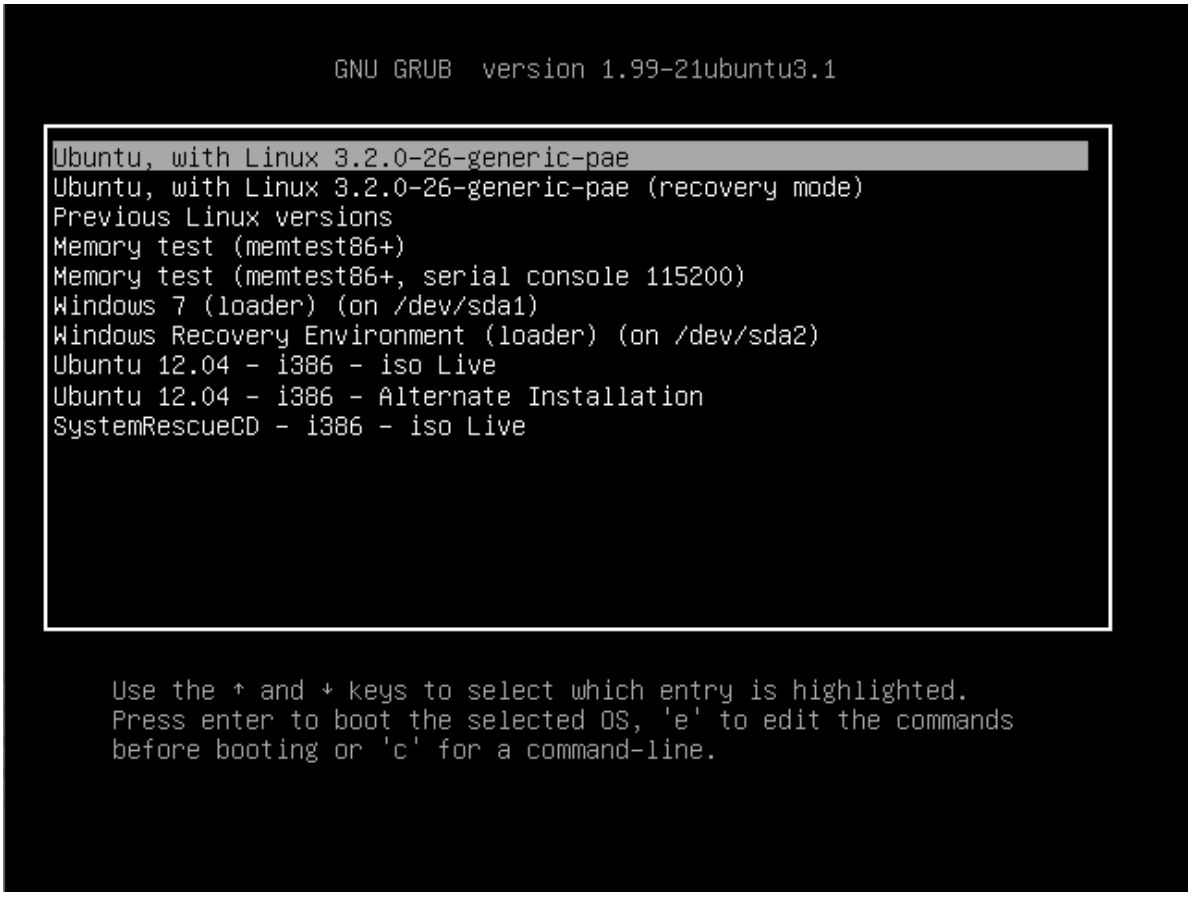
The theme use for this boot screen is available in this archive: *http://xberger.free.fr/downloads/fortune.tar.gz* [1]

Untar in **`/boot/burg/themes/`** and execute the command **`update-burg`** and reboot.

At next boot, hit **`t`** and select **`fortune`**.

The full startup menu is available on the USB key. You should select to boot on the USB key and hold **`shift`** to access to the following menu.

[1] http://xberger.free.fr/downloads/fortune.tar.gz



Note

This screenshot is showing additional boot. Ref bellow to see how to install live OS into the USB key.

4.4. Create a startup key from a working system

If you have a secured system running, it is easy to recreate an USB key with the following procedure:

Plug the new USB key and mount it in **`/media/usb`**:

Note

The key should be prepared with **`gparted`** as described in first chapter and have a the partition formatted as *`ext2`*.

```
mount /dev/sdc2 /media/usb
```

Copy the content of the original key to the startup key:

```
cp -a /boot/* /media/usb
```

Label the key to be a boot key:

```
e2label /dev/sdc2 BOOT
```

Install grub2 into the new disk:

```
grub-install --force --no-floppy --boot-directory=/media/usb --root-directory=/ /dev/sdc
```

Note

Each time the kernel is update, the second key will also have to be updated using this procedure

4.5. Backup of the startup key and store it is a safe location

The USB key you just create is now the only way you have to start your computer. It is mandatory to have a backup of it and to be able to recreate it.

Cleanup unused space of **`/boot`** partition:

```
dd if=/dev/zero of=/boot/todelete
rm /boot/todelete
```

Umount **`/boot`** partition:

```
umount /dev/sdb2
```

Backup the MBR of the USB key:

```
dd if=/dev/sdb of=startup.mbr bs=512 count=1
```

Backup the boot partition:

```
dd if=/dev/sdb2 of=startup.sdb2
```

Compress the backup:

```
tar cvjf startup.bkp.tar.bz2 startup.mbr startup.sdb2
```

Store the file **`startup.bkp.tar.bz2`** into a safe area.

Warning

This to perform an update of the startup key image after every kernel update.

4.6. Restore the startup key into another key

The following action should be performed on the computer freshly installed, from a live version of the OS or start from the startup key if installation is completed.

Plug the new target USB key and identify its device. Let's assume it is **/dev/sdc**.

Then follow these instructions to recreate the key from the backup:

```
dd if=startup.mbr of=/dev/sdc
mkfs.vfat /dev/sdc1
dd if=startup.sdb2 of=/dev/sdc2 fsck -y /dev/sdc2
mount /dev/sdc2 /media/usb grub-install --force --no-floppy --boot-directory=/media/usb --
root-directory=/ /dev/sdc
```

This command could be a little bit long. Executing the following command will let **dd** write a status of its progress:

```
kill -USR1 $(pidof dd)
```

4.7. Store data in a remote location to secure their availability

Our goal is to store data in a place that will ensure their availability even is the hardware is lost. The easiest solution is to use the service of cloud provided by one of the following company:

Online storages services

- 5GB up to 20GB Free - *https://one.ubuntu.com/*
- 5GB Free - *https://www.wuala.com/*[2]
- 2GB Free - *https://www.dropbox.com/*
- 2GB Free - *https://www.spideroak.com/*
- 5GB Free - *https://drive.google.com/* --> with grive: *https://github.com/Grive/grive* [3]

For Windows™ only, some additional space can be used to store non confidential data

- 5GB Free - *https://www.sugarsync.com/*
- 7GB Free - *https://skydrive.live.com/*

Offline backup service

- 25GB Free - *https://www.hubic.me/*

Online notebook

- 60MB/month Free - *https://www.evernote.com/*

[2] https://www.wuala.com/
[3] https://github.com/Grive/grive

This solution is not intending to replace a real solution of backup restore but has the advantage to be easy to setup and to be cheap. A dedicated article is explaining how to setup a real solution of backup: **A secure place to backup my data** .

4.8. Ensure the confidentiality of data stored into the cloud

The cloud is a private area provided by an external company... This sentence is not sounding right because it is mixing private and external company... So if we consider that this external area is not a fully private place we will have to add another layer of encyption to secure our data in the cloud. For this, we will use encfs and we will configure pam to automatically unlock the directory during the login process.

Install **encfs** and **fuse-utils** using the following command as root:

```
apt-get install encfs fuse-utils
sh -c "echo fuse >> /etc/modules"
modprobe fuse
adduser $USER fuse
```

We will configure the encryption of sensitive data into in a dedicated directory of **Ubuntu One**.

Install libpam-mount with the followinf command:

```
sudo apt-get install libpam-mount libpam-encfs
```

Execute the following command as standard user to create the secure area:

```
LC_ALL=C encfs /home/$USER/Ubuntu\ One/.encrypted /home/$USER/encrypted/
```

Let encfs create the directories and select the **`(p) pre-configured paranoia mode`** or just press **`enter`** to have a normal protection.

Enter the passphrase twice. The passphrase should be the same as the passphrase of the user account. this will allow pam-mount to automatically decrypt the encfs directory

Edit the file **`/etc/security/pam_mount.conf.xml`** look for the line **`<!-- Volume definitions -->`** Append the following lines just after by replacing **`<<user>>`** with your login

```
<volume user="<<user>>" fstype="fuse" path="encfs#/home/<<user>>/Ubuntu One/.encrypted"
 mountpoint="/home/<<user>>/encrypted" />
```

4.9. Passphrase management

LUKS

The LUKS encryption system can manage up to eight passphrases (in this article, we already used two).

Adding a password can be done with the following command:

```
cryptsetup luksAddKey /dev/sda5
```

To delete a passphrase :

```
cryptsetup luksKillSlot /dev/sda5 <the slot number to be deleted>
```

Encfs

encfs have only one passphrase. The passphrase can be changed with the following command:

```
encfsctl passwd ~/Ubuntu\ One/.encrypted/
```

Note

Remember that the passphrase of **encfs** and the passphrase of you account should be identical to allow **libpam_mount** to decrypt **encfs** directory at login.

4.10. Add live OS into the usb key

On the following section we will consider that the FAT32 partition of the startup key is mounted in **`/media/usb`**.

If this is not yet the case, execute the following command to do it:

```
mkdir /media/usb mount /dev/sdb1 /media/usb
```

Add ubuntu desktop on the usb stick

Create the directory **`/media/usb/iso`**:

```
mkdir /media/usb/iso
```

Download **`ubuntu-12.04.1-desktop-i386.iso`** from *http://releases.ubuntu.com/precise/* and copy it in **`/media/usb/iso`**.

Create the file **`/etc/grub.d/42_custom`** with the following content:

```
#!/bin/sh
exec tail -n +3 $0
# This file provides an easy way to add custom menu entries. Simply type the
# menu entries you want to add after this comment. Be careful not to change
# the 'exec tail' line above.
menuentry "Ubuntu 12.04.1 - i386 - iso Live" {
     set gfxpayload=800x600x16
     set root=(hd0,msdos1)
     set isofile="/iso/ubuntu-12.04.1-desktop-i386.iso"
     search --set -f $isofile
     loopback loop $isofile
     linux (loop)/casper/vmlinuz boot=casper iso-scan/filename=$isofile noeject noprompt
 splash -- locale=fr_FR console-setup/layoutcode=fr
     initrd (loop)/casper/initrd.lz
}
```

Upgrade **grub** configuration files with the following command:

```
update-grub
```

Add Unbuntu Alternate CD on the usb stick

Download **ubuntu-12.04.1-alternate-i386.iso** from *http://releases.ubuntu.com/precise/* and copy it in **/media/usb/iso**.

Create the file **/etc/grub.d/43_custom** with the following content:

```
#!/bin/shexec tail -n +3 $0# This file provides an easy way to add custom menu entries.
 Simply type the# menu entries you want to add after this comment. Be careful not to change#
 the 'exec tail' line above.menuentry "Ubuntu 12.04.1 - i386 - Alternate Installation" {
 set gfxpayload=800x600x16     set root=(hd0,msdos1)     search --set -f /iso/ubuntu-12.04.1-
alternate-i386.iso     linux /iso/vmlinuz noeject -- locale=fr_FR console-setup/layoutcode=fr
     initrd /iso/initrd.gz}
$0# This file provides an easy way to add custom menu entries. Simply type
the# menu entries you want to add after this comment. Be careful not to
change# the 'exec tail' line
above.menuentry "Ubuntu 12.04.1 - i386 - Alternate Installation"
{     set
gfxpayload=800x600x16     set
root=(hd0,msdos1)     search --set -f /iso/ubuntu-12.04.1-alternate-
i386.iso     linux /iso/vmlinuz noeject -- locale=fr_FR console-setup/
layoutcode=fr     initrd /iso/
initrd.gz
```

Upgrade **grub** configuration files with the following command:

```
update-grub
```

Add System Rescue CD on the usb stick

Download and copy it in **/media/usb/iso**.

Create the file **/etc/grub.d/44_custom** with the following content:

```
#!/bin/sh
exec tail -n +3 $0
# This file provides an easy way to add custom menu entries. Simply type the
# menu entries you want to add after this comment. Be careful not to change
# the 'exec tail' line above.
menuentry "SystemRescueCD - i386 - iso Live" {
     set gfxpayload=800x600x16
     insmod ext2
     insmod loopback
     insmod iso9660
     set root=(hd0,msdos1)
     set isofile="/iso/systemrescuecd-x86-2.8.1.iso"
     search --set -f $isofile
     loopback loop $isofile
     linux (loop)/isolinux/rescuecd isoloop=$isofile setkmap=fr
     initrd (loop)/isolinux/initram.igz
}
```

Upgrade **grub** configuration files with the following command:

```
update-grub
```

4.11. Online security

You are using Linux, this is a first good step for the on line security. If you intend to use your computer directly connected to the Internet you should at least start the firewall by executing the following command:

```
ufw enable
```

To go further into online security, refer to the document: **A configuration to preserve my data**.

4.12. Remove the key after startup

Warning

In this section it is proposed to modify the authentication process. An error could block the authentication of your computer. If you do so, start your computer in recovery mode and delete the modification you previously did. Be aware that the option to remove the key only belong to the graphical interface. We do consider that if you start a console, you can as well mount the **`/boot`** partition manually.

To not compromise the security of your PC you need to carry the USB key with you all the time even if the PC is still running and locked.

To make this easy we will configure the computer to automatically mount and umount the key in different occasions:

Umount the key:

- At startup to let you unplug the key and go before login
- When the computer is locked to let you unplug the key when you need to let your PC alone
- When you close your session

Mount the usb key:

- When you open a session
- When the session is unlocked

Umount key after boot

Add the following lines as the first active line of the file **`/etc/rc.local`**

```
umount /boot
/usr/bin/aplay /usr/share/sounds/purple/send.wav > /dev/null 2>&1
```

This will umount the USB key after boot and play a sound letting you know it is safe to remove the key.

Now, we need to authorize a standard user to mount and umount the /boot partition which is in the startup key. To do so, it is required to update the **`/boot`** description in the file **`/etc/fstab`** and append the option *`users`* as an option. After modification, the line should look like that:

```
LABEL=BOOT /boot ext2 noatime,users 0 2
```

To automatically mount and umount the key when the session is open or close we will use the capabilities given by PAM. **libpam-script** will allow us to execute a script when user open or close a session.

Install **libpam-script** with the following command:

```
apt-get install libpam-script
```

/usr/share/libpam-script/pam_script_ses_open is executed when the session is open and will mount the partition **/boot**. Create this script with the following content:

```
#!/bin/bash
if [[ "$PAM_USER" = "lightdm" ]] || ( mount | grep /boot > /dev/null 2>&1 );
then
exit 0
fi
if ( mount /boot > /dev/null 2>&1 ); then
/usr/bin/aplay /usr/share/sounds/purple/receive.wav > /dev/null 2>&1
fi
exit 0
```

This script mounts the **/boot** partition and play a sound letting you know that the key has been successfully reconnected.

Change the access right to make it executable:

```
chmod 755 /usr/share/libpam-script/pam_script_ses_open
```

/usr/share/libpam-script/pam_script_ses_close is executed when the session is closed and will dismount the partition **/boot**. Create this script with the following content:

```
#!/bin/bash
if [[ "$PAM_USER" = "lightdm" ]]; then
exit 0
fi
device=$(mount | grep /boot | cut -c -8)
if ( umount /boot > /dev/null 2>&1 ); then
umount $device* > /dev/null 2>&1
/usr/bin/aplay /usr/share/sounds/purple/send.wav > /dev/null 2>&1
fi
exit 0
```

This script umounts the **/boot** partition and all the partition of the USB key then play a sound letting you know that you can remove the USB key safely.

Change the access right to make it executable:

```
chmod 755 /usr/share/libpam-script/pam_script_ses_close
```

We need now to add **pam_script** into PAM session management. Modify the file **/etc/pam.d/lightdm** and append the **pam_script** line below just after the line *@include common-account* as below:

```
@include common-accountsession optional pam_script.so
mon-accountsession
```

We will now create a script managing the event of the screen-saver. The script below is applicable on **xscreensaver** which is the default screensaver of **xubuntu**. If your screensaver is different you can replace it by **xcreenserver** or, more difficult, you can update the script bellow. Create the file **/usr/local/bin/startup_key_manager.sh** with the following content:

```
#!/bin/bash
/usr/bin/xscreensaver-command -watch | while read line; do
if [ x"$(echo "$line" | grep 'LOCK')" != x ] ; then
/usr/share/libpam-script/pam_script_ses_close
fi
if [ x"$(echo "$line" | grep 'UNBLANK')" != x ] ; then
/usr/share/libpam-script/pam_script_ses_open
fi
done
```

Change the right to make it executable:

```
chmod 755 /usr/local/bin/startup_key_manager.sh
```

This script will monitor the screensaver and manage the **/boot** partition of the key accordingly.

This script should be added to start automatically when the session is open. As normal user, create the script **~/.config/autostart/startupKeyManager.desktop** with the following content:

```
[Desktop En
try]Encoding=UTF-8Version=0.9.4Type=ApplicationName=startupKeyManagerComment=startup key
 managerExec=/usr/local/bin/
startup_key_manager.shStartupNotify=falseTerminal=falseHidden=false
En
try]
Encoding=UTF-8
Version=0.9.4
Type=ApplicationName=startupKeyManagerComment=startup key
 managerExec=/usr/local/
bin/
startup_key_manager.sh
StartupNotify=false
```

The script will be activated when you will start another session.

Warning

When your are updating the kernel, be sure the session will stay open and the screen server stays deactivated during the upgrade.

4.13. Two factor authentication

Warning

In this section it is proposed to modify the authentication process for graphical as well as console login. An error could block the authentication of your computer. If you do so, start your computer in recovery mode and delete the modification you previously did.

We have now a system which is secured and easy to use but we can improve a little bit the security by adding a two factor authentication requiring the USB key to be plugged and the password to be cor-

rect before opening the session. With this two factor authentication you will be sure that, in an event of somebody knows your password, he will not be able to unlock your session when you are at the coffee corner with the USB key in your pocket.

To activate the two factor authentication we will use the pam module previously installed: pam-script. The logic would be to use pam_usb but this module has some inconvenient: It requires an action on every USB keys you would have and make the key replication more complex. We will then authenticate the USB key based on the keyfile present in it using pam_script.

Create the script **`/usr/share/libpam-script/pam_script_ses_auth`** dedicated to authenticate the USB key with the following content:

```
#!/bin/bash
mount /boot
result=1;
if ( sha1sum -c --status /usr/share/libpam-script/keycheck ); then
result=0
fi
umount /boot
exit $result
```

Change the access right to make it executable:

```
chmod 755 /usr/share/libpam-script/pam_script_auth
```

The sha1sum is used to validate the key. The keycheck file is created with the following commands:

```
sha1sum /boot/keyfile > /usr/share/libpam-script/keycheck
chmod 444 /usr/share/libpam-script/keycheck
```

We need now to add **pam_script** into the system authentication process just after the authentication by password. Modify the file **`/etc/pam.d/common-auth`** and add *`pam_script`* just after *`pam_deny`* as follow:

```
auth requisite pam_deny.soauth required pam_script.so
 pam_deny.soauth required
```

Troubleshooting

In some circumstances you may need to access to the data of the encrypted partition without booting the computer. Here are some method to do so.

5.1. Boot in recovery mode

Boot on the USB stick and select recovery mode.

Select **`root Drop to root shell prompt`**

Mount **`/`** with a read write access and mount **`/boot`** using the following commands

```
mount -oremount,rw /
mount /dev/sdb2 /boot
```

Note

As you are using you startup key to boot your PC, you are identified to be the owner of the machine. If you used a keyfile, you will be granted a root access without any password. Without the key such a startup is impossible and you will have to follow the instruction of the next chapter to access to your data.

5.2. Manually access to the partition

To access to the encrypted partition, boot on a live Operating System and follow the procedure below to mount and umount the disk.

Mount encrypted partition

```
modprobe dm-crypt
cryptsetup luksOpen /dev/sdb5 crypt Enter your passphrase
vgscan --mknodes
vgchange -ay
mkdir /mnt/crypt
mount /dev/VolGroup/LV_slash /mnt/crypt
```

Umount encrypted partition

```
umount /mnt/crypt
vgchange -an
cryptsetup luksClose crypt
```

To access to the encrypted partition from (initramfs) follow the tips procedure bellow

Mount encrypted partition

```
cryptsetup luksOpen /dev/sdb5 crypt
Enter your passphrase
mkdir /mnt/crypt
```

```
mount /dev/VolGroup/LV_slash /mnt/crypt
```

Umount encrypted partition

```
umount /mnt/crypt
cryptsetup luksClose crypt
```

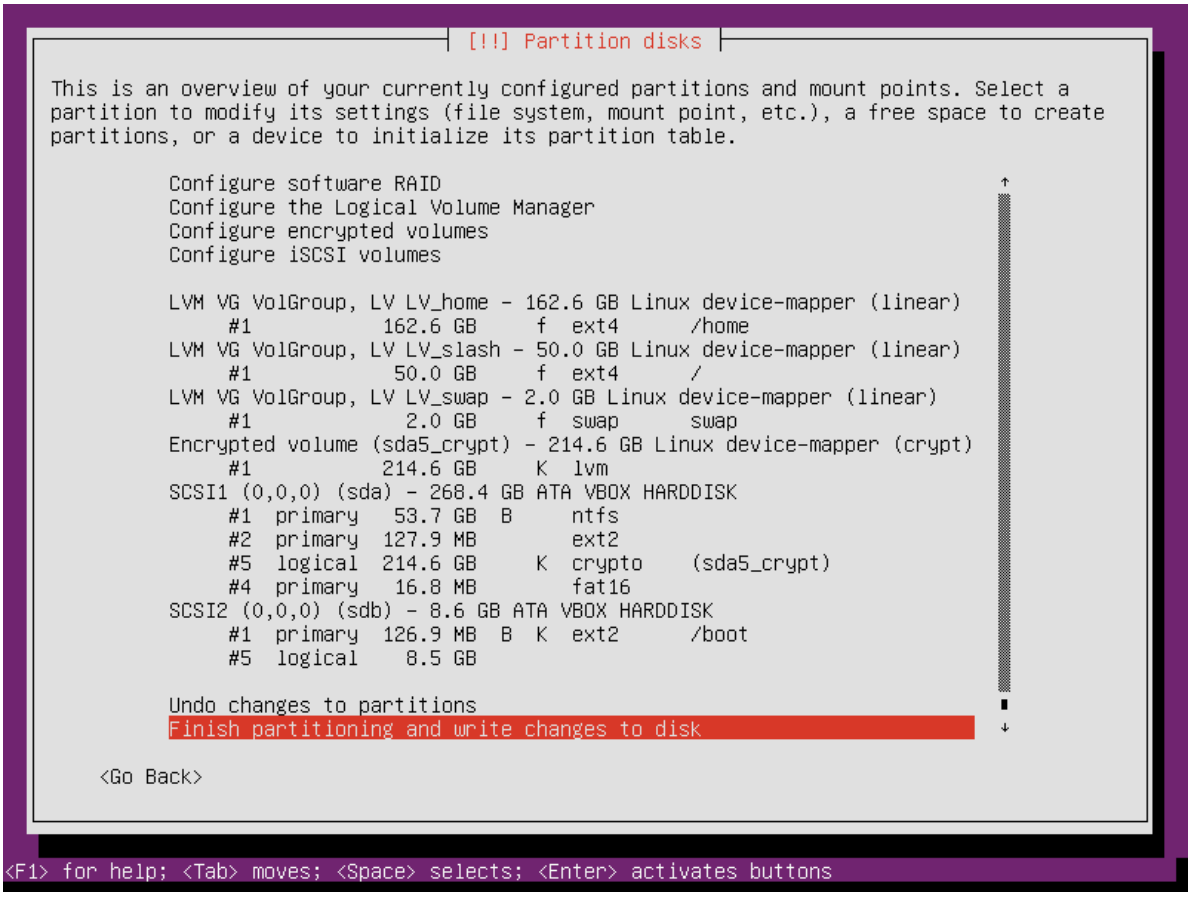
5.3. Reinstall the secure system and keep data in home directory

In case of a major issue you may have to reinstall your system from scratch.

Boot on **Ubuntu Alternate** image **`ubuntu-12.04.1-alternate-i386.iso`**

- Enter the name of the computer
- Enter the full name of the main user
- Enter the username for the account
- Choose a password and enter it twice
- Do not select to encrypt home directory (We will reused the encrypted partition already existing into the system)
- Set clock timezone
- Partition disks: **`Manual`**
- Select: **`Configure encrypted volume`**
  - Keep current partition layout and configure encrypted volumes: Yes
  - Activate existing encrypted volume
    - Enter the pass phrase

  You will see the LVM volumes in the disk partition description.
- Define mounting point as described in the previous chapter (Format the partition **`/`** and **`/boot`** but do not format the partition **`/home`**)



- Install the OS
- Reboot

After this installation **`/boot`** and **`/`** have been recreated from scratch. It is then required to reapply the configuration described in previous chapter. If you use a keyfile to unlock the secured partition, this file should be reinstalled in the **`/boot`** partition from the backup you did. If you previously saved the installed packages into a file **`installed-packages`** as described into **A secure place to backup my data**, it is possible to restore to reinstall them with the following commands:

```
apt-get update
apt-get upgrade
dpkg --set-selections < installed-packages
apt-get -u dselect-upgrade
```

To go further and improve the security and data integrity

Some action could improve the security of your computer. You can for instance remove Windows™ from your computer. In this case you computer will not boot at all without the startup key and will be unusable and without any evidence on what could hold the encrypted partition. Then you can add password to BIOS and avoid boot on usb and add password to burg and gurb to avoid startup command modification.

You can also use TrueCrypt with/without inner volume to secure confidential data.

Other article from the same author are also available:

- Read the article Security: A secure place to backup my data
- Read the article Security: A configuration to preserve my data

To go even further, you can apply the recommendation from the NSA:*http://www.nsa.gov/ia/_files/os/redhat/rhel5-guide-i731.pdf* [1]

[1] http://www.nsa.gov/ia/_files/os/redhat/rhel5-guide-i731.pdf

References

This article has been written based on information found in the Internet:

- https://help.ubuntu.com/community/EncryptedFilesystemHowto
- http://ubuntuforums.org/showthread.php?t=1549847
- http://ubuntuforums.org/showthread.php?t=1369019
- http://blog.stalkr.net/2012/05/usb-rescue-and-secure-boot-disk.html
- http://www.nsa.gov/ia/_files/os/redhat/rhel5-guide-i731.pdf
- http://askubuntu.com/questions/63594/mount-encrypted-volumes-from-command-line
- http://ubunteros.tuxfamily.org/spip.php?article204
- http://www.howtogeek.com/?post_type=post&p=70146
- http://www.sevenforums.com/tutorials/7412-guest-account-turn-off.html
- http://windows.microsoft.com/en-us/windows-vista/What-is-a-guest-account
- http://doc.ubuntu-fr.org/tutoriel/chiffrer_son_disque * http://doc.ubuntu-fr.org/cryptsetup
- http://doc.ubuntu-fr.org/encfs
- http://linuxconfig.org/linux-authentication-login-with-usb-device
- http://artisan.karma-lab.net/petite-introduction-a-pam
- http://www.psychocats.net/ubuntu/security
- https://code.google.com/p/cryptsetup/

Appendix A. Revision History

Revision 0-0 **Wed Sep 19 2012** **Xavier Berger** *berger.xavier@gmail.com*

Initial creation of book by publican

Index

F