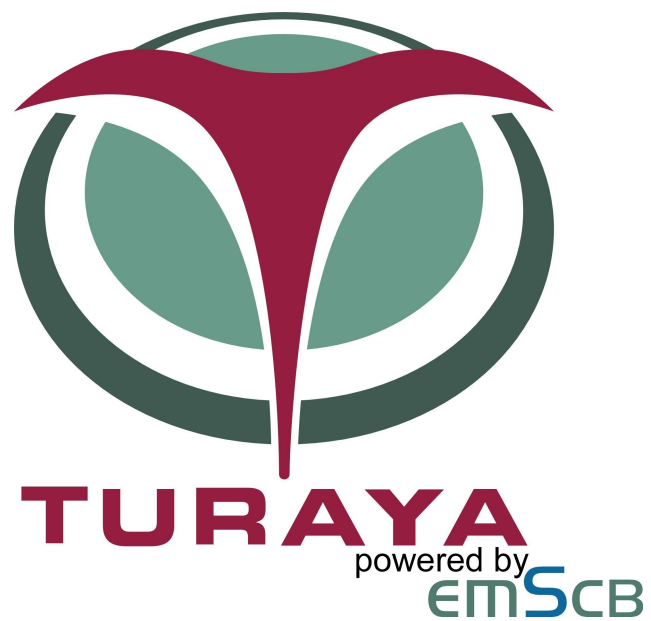


EMSCB:
European Multilaterally Secure Computing Base



DemoCD User Guide
for Turaya.Crypt / Turaya.VPN / Turaya.FairDRM

March 13, 2007

Contents

1	Overview	3
1.1	EMSCB	3
1.2	Trusted Computing	3
1.2.1	Authenticated Boot	3
1.2.2	Secured Cryptography	4
1.2.3	Trusted Platform Module	4
1.3	Turaya.VPN	4
1.4	Turaya.Crypt	5
1.5	Turaya.FairDRM	7
2	System Requirements	8
3	Starting the CD	9
4	Demo Applications	10
4.1	Trusted GUI	10
4.2	Turaya.VPN	11
4.3	Turaya.Crypt	13
4.3.1	Pre-configured encrypted device	13
4.3.2	Creating encrypted devices with TPM support	15
4.3.3	Command-line configuration tool	20
4.4	Turaya.FairDRM	23
4.4.1	Requirements	23
4.4.2	Certificate generation	23
4.4.3	Media Download	24
4.4.4	Media Playback	25
4.4.5	Media Transfer	26
A	Known Issues and Problems	29
B	Initializing the TPM	30

Chapter 1

Overview

1.1 EMSCB

European Multilaterally Secure Computing Base (EMSCB) aims at developing a trustworthy computing platform with open standards that solves many security problems of conventional platforms. The platform deploys

- hardware functionalities provided by Trusted Computing,
- a security kernel based on a microkernel and
- an efficient migration of existing operating systems.

In the sense of multilateral security, the EMSCB platform allows the enforcement of security policies of different parties, i.e., end-users as well as industry. Consequently, the platform enables the realization of various innovative business models while averting the potential risks of Trusted Computing platforms concerning privacy issues. The source code of the EMSCB platform will be published under an open source license.

The DemoCD at hand documents the current development status of the EMSCB project.

You can find the EMSCB webpage at <http://www.emscb.org>.

1.2 Trusted Computing

This section introduces Trusted Computing functionalities, namely the authenticated boot process and some cryptographic building blocks.

1.2.1 Authenticated Boot

During an authenticated boot process, each part of code which is executed is “measured” before execution, e.g., by calculating a cryptographic hash of the code. Trusted Computing hardware is responsible for the secure storage and provision of the measurement results. Upon completion of an authenticated boot process, these measurement results reflect the configuration of the hardware and software environment currently running. Trusted Computing technology, however, remains passive and does explicitly not prevent a certain computing

environment from being compromised during runtime. However, integrated cryptographic mechanisms enable the platform to verifiably report their actual state to local and remote parties.

1.2.2 Secured Cryptography

Trusted Computing hardware implements a set of cryptographic operations to ensure that malicious software cannot compromise cryptographic keys. Usually, key generation and decryption operations are done “on-chip”. Private and secret keys never leave the chip without being encrypted. To perform a decryption operation with a specific key, several types of authorization are possible. A distinctive feature of Trusted Computing hardware is the ability to not only use passwords as authorization but also integrity measurements. That is, only a platform running previously defined software or hardware components is authorized to use a certain key. Moreover, the property that a certain key is “bound” to a platform configuration can be certified by Trusted Computing hardware. This certification includes the integrity measurements, which authorize a platform to employ the key. A remote party can verify the certificate and validate the embedded integrity measurement against “known good” configurations before encrypting data with the certified key¹.

1.2.3 Trusted Platform Module

The base of Trusted Computing technology is the Trusted Platform Module (TPM), which is considered to be a tamper-resistant hardware device similar to a smart-card and is assumed to be securely bound to the computing platform. The TPM is primarily used as a root of trust for integrity measurement and reporting and to secure all critical cryptographic operations.

The Trusted Computing Group (TCG) publishes the TPM specifications. The TCG is the successor of the Trusted Computing Platform Alliance (TCPA), an initiative led by AMD, HP, IBM, Infineon, Intel, Lenovo, Microsoft, and Sun. TPMs are available, e.g., from Atmel, Broadcom, Infineon, Sinosun, STMicroelectronics, and Winbond. As depicted in [Figure 1.1](#), a TPM basically consists of an asymmetric cryptographic engine (RSA), a cryptographic hash function (SHA-1), platform configuration registers (PCRs), a true random number generator, a few bytes of volatile memory, some kilobytes non-volatile memory and sensors for tampering detection.

To initialize the TPM on your platform for the Turaya Demo applications, see [Appendix B](#).

1.3 Turaya.VPN

Turaya.VPN is a Virtual Private Network client which is running in parallel to, but completely isolated from, a legacy operating system, i.e., Linux in this case. Certificates and keys as well as cryptographic operations are thus protected from hostile access by Linux and any malicious software such as worms and viruses.

¹Mostly, the data to be encrypted is a secret symmetric key used for the encryption of larger amounts of data.

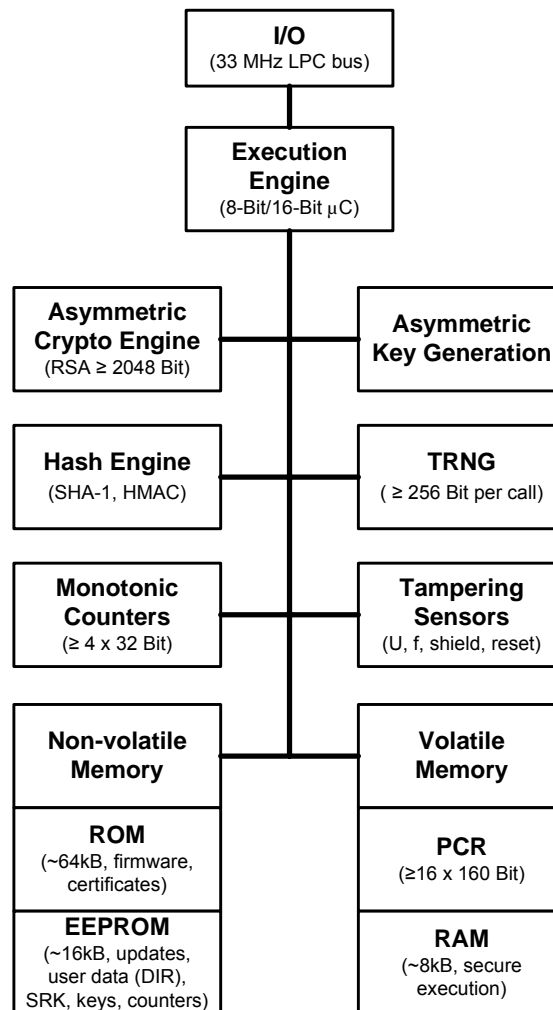


Figure 1.1: TPM chip version 1.2.

Turaya.VPN is based on EMSCB technology and is interoperable with standard VPN servers (IPsec) and fully transparent to the user. It integrates a firewalling component and network configuration via the Dynamic Host Configuration Protocol (DHCP).

We have set up a demonstration webserver which is part of a demonstration intranet accessible through the Turaya.VPN client. The network communication to this webserver is encrypted with a secret key residing in the VPN component.

For further information visit the EMSCB webpage at <http://www.emscb.org>.

1.4 Turaya.Crypt

Turaya.Crypt provides a secure device encryption for Linux. In contrast to the conventional device encryption mechanism provided by the Linux kernel, Turaya.Crypt is based on EMSCB technology to isolate security-critical key information and cryptographic operations from the

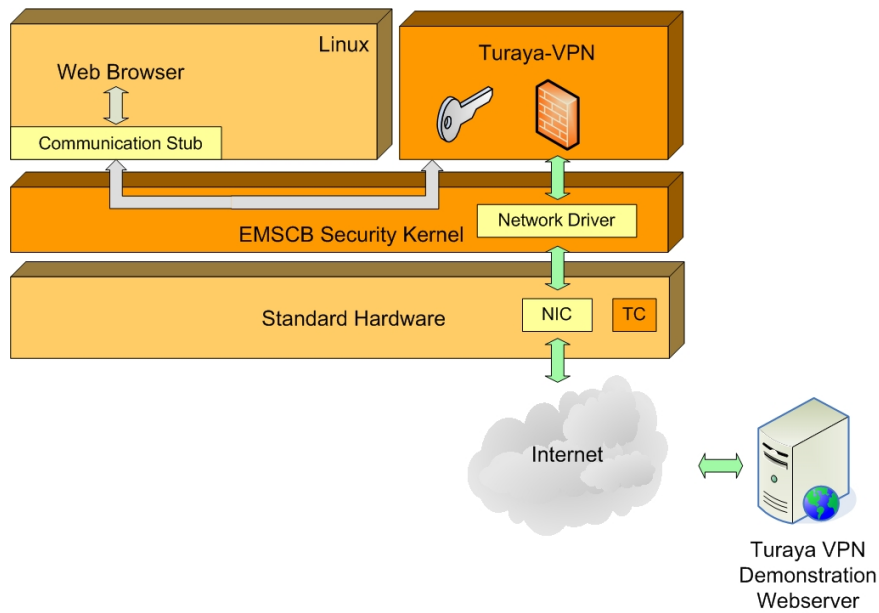


Figure 1.2: Architecture of Turaya.VPN

Linux operating system in order to prevent unauthorized access.

Turaya.Crypt handles security-critical information (e.g., keys) and cryptographic operations used by a wrapper module of the Linux device encryption mechanism. It acts as a decryption/encryption service running in parallel to the Linux operating system.

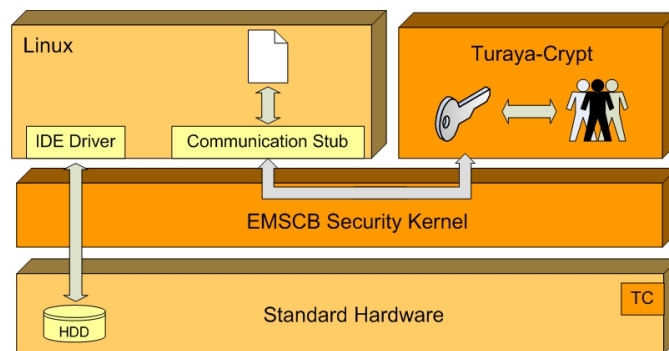


Figure 1.3: Architecture of Turaya.Crypt

Turaya.Crypt prevents any Linux application from accessing security-critical information and operations. Therefore, it clearly enhances the security of previous hard-disk encryption schemes and provides full usability while implementing the standard Linux encryption interface.

For further information visit the EMSCB webpage at <http://www.emscb.org>.

1.5 Turaya.FairDRM

Turaya.FairDRM demonstrates how a Digital Rights Management (DRM) system with a strong focus on “fair use” aspects as well as overall security and unforgeability of the system can be designed and implemented.

Turaya.FairDRM allows the content providers to make sure that the previously defined license conditions are met by the user platform (policy enforcement). On the other hand, users are able to backup their media or transfer it to another device or user (provided that the license that was negotiated between user and provider before allows this).

Turaya.FairDRM uses the TPM’s binding functionality to bind content (e.g., audio files) to the user’s platform. In order to perform the binding operation, the content server requires a certificate generated on the user’s platform. The certificate contains information about the user’s platform configuration (by using PCR values) and the public part of a cryptographic key used for the binding procedure (therefore called “binding key”). Because the private key can only be accessed by the TPM, the encrypted media content can only be decrypted on the user’s platform.

For further information visit the Turaya.FairDRM webpage at <http://fairdrm.emscb.org>.

Chapter 2

System Requirements

- 32-bit Intel Pentium III processor or better (IA32 586 compatible)
- 1024 MB RAM at minimum
- CD-ROM drive from which the system can be booted
- VESA 2.0 or higher compatible graphics card
- PS/2 or USB mouse
- PS/2 keyboard
- TPM 1.1b

Note: This DemoCD is a snapshot of work in progress and not a finished product. It has been tested on a limited amount of different configurations only and cannot be guaranteed to run on all systems that meet the mentioned requirements.

Chapter 3

Starting the CD

Insert the Turaya DemoCD into the CD-ROM drive of your computer and boot from CD. (You might need to change the boot order of your drives in the BIOS. Consult your computer manual in case you do not know how to do this.)

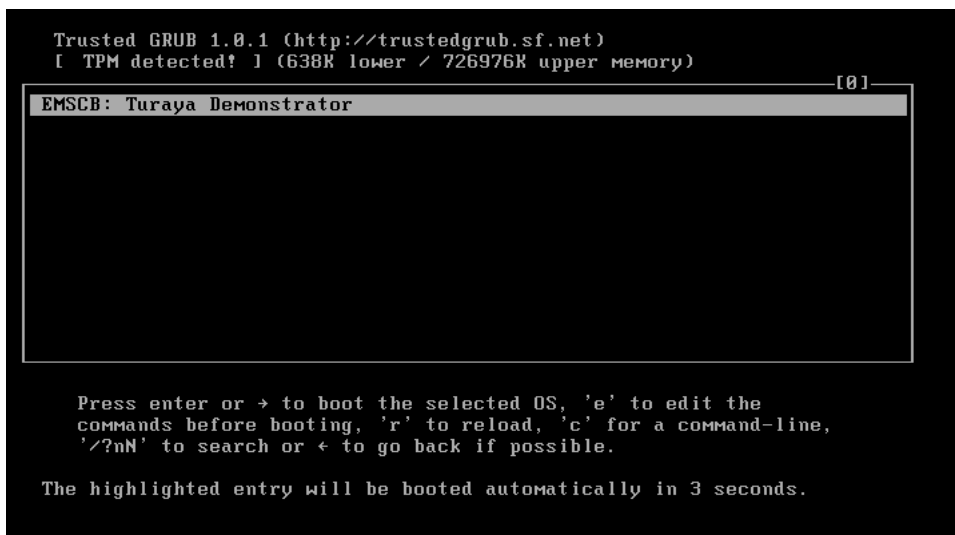


Figure 3.1: Screenshot of Trusted GRUB.

We use Trusted GRUB¹ as bootloader. If a Trusted Platform Module (TPM) is detected, it will be used to measure the loaded components. The measurement result will be stored within the TPM and can later be used to verify the system state.

¹<http://trustedgrub.sourceforge.net>

Chapter 4

Demo Applications

On this DemoCD, we present some applications using the Turaya platform technology. You will find a description of each demo application in the following sections. But first, we will give a short introduction of the graphical user interface (GUI) of Turaya and of this DemoCD.

4.1 Trusted GUI

After a few moments of loading, you will see the Control Center view of the μ GUI, the trusted GUI system of Turaya. The μ GUI Control Center displays a thumbnail view of each compartment currently running, see [Figure 4.1](#).



Figure 4.1: Screenshot of the Trusted GUI.

You can select a compartment by using the arrow keys. The currently selected compartment has a highlighted border on its thumbnail. Press [ENTER] to switch to that compartment, bringing it to fullscreen mode. You can always switch back to the Control Center by pressing the [F12] key at any time. The Control Center provides a help screen, which shows all of its key shortcuts.

The whole graphical display is controlled by the μ GUI. Applications are running in compartments, which have their own distinct virtual screens. Moreover, all user input (from keyboard and pointer devices) is controlled by the μ GUI and is passed to the compartment that currently has the focus. In this way, the μ GUI provides isolation at the (graphical) user interface level.

Note that there is always a status bar at the top of screen (see [Figure 4.2](#)). This status bar shows the name of the compartment being currently displayed as well as the total number of compartment screens. This status bar cannot be forged by any compartment because it is under control of the μ GUI. Thus, the status bar provides a mean of application authentication and can be used to establish a trusted path to security-critical applications. For example, we use this to indicate the authenticity of the password dialog for user login purposes.



Figure 4.2: Compartment identification in the status bar.

4.2 Turaya.VPN

Go to the first L4Linux compartment, and click on the icon called “Turaya.VPN Demo” (see [Figure 4.3](#)).



Figure 4.3: Start Turaya.VPN demo.

The Firefox browser will open and show a local welcome page (see [Figure 4.4](#)). Follow the link at the bottom to connect to the Turaya.VPN demonstration webserver. A connection to the Turaya Demonstration Virtual Private Network (VPN) will then be established.¹

If you have not yet authenticated yourself, the Turaya system will ask you to do so by entering a password. Press [F12] to escape from the Linux window and select the password dialog

¹Please note that it may take up to 1 minute until the DHCP-client integrated in the Turaya.VPN has gathered the necessary network information to access the internet.

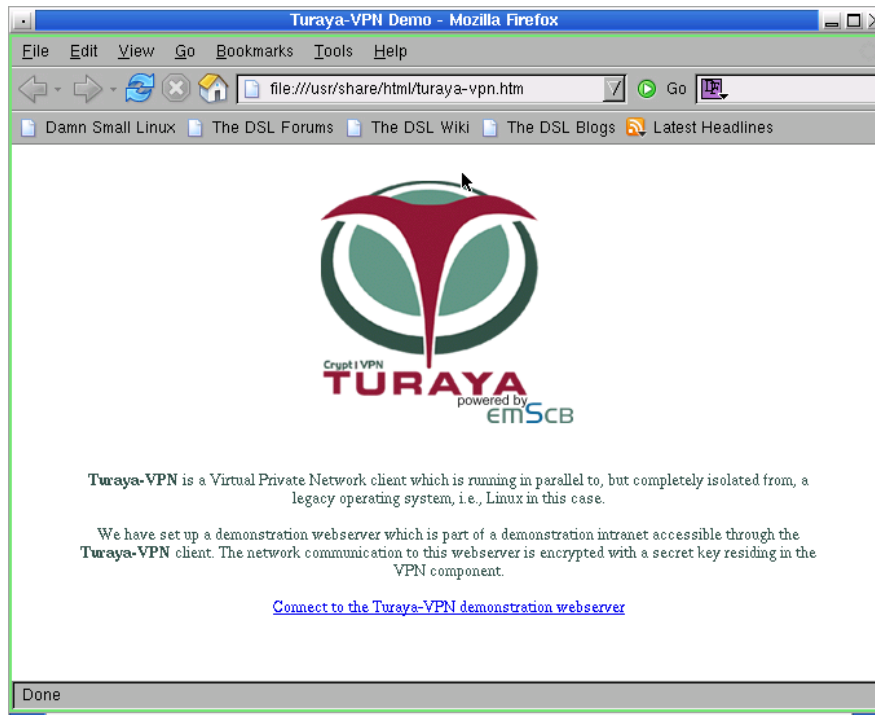


Figure 4.4: Welcome page of the Turaya.VPN demo.

(see Figure 4.5). Now, enter the password.

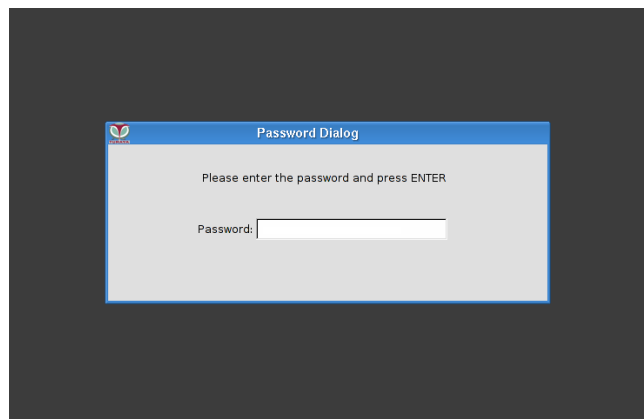


Figure 4.5: Password dialog of Turaya.

Password: **emscb**

Note that the password dialog will only appear if you have not yet authenticated yourself. If you have authenticated yourself already before (e.g., because you have started to use Turaya.Crypt or any other security-critical application), there will be no password dialog again because you have authenticated yourself for the whole Turaya system in a single sign-on

mechanism.

Technically speaking, the HTTP request to the private IP address 10.9.8.7 is routed through the EMSCB Security Kernel to the Turaya.VPN component. The Turaya.VPN component then identifies the IP address 10.9.8.7 as the destination address to the Demonstration VPN and encrypts the request. The responses back from the Turaya Demonstration VPN will be transparently decrypted and again handed over to the browser.

4.3 Turaya.Crypt

Let's take a look at the functions of Turaya.Crypt. On this DemoCD, Turaya.Crypt is executed in single-user multi-key mode, i.e., there is only one user who can use Turaya.Crypt at the same time but using different keys for different encrypted devices. All keys are generated at random and each key is identified by a unique name. When a new encrypted device is going to be created, a key has to be chosen and can be selected by its unique name, the key ID. All keys of Turaya.Crypt are encrypted themselves to protect against unauthorized usage. Before any key can be used, the user has to authenticate himself at the Turaya system.

4.3.1 Pre-configured encrypted device

This DemoCD contains a pre-configured encrypted virtual device, which can be used directly when running the DemoCD. Thus, there is no need to modify any hard-drive partition on your system. On the Linux desktop, you can find an icon called "Turaya.Crypt Device". When first started, this icon will display a lock symbol, which indicates that the device is encrypted and cannot be accessed (see [Figure 4.6](#)). If you click on the icon, the Firefox browser will start and open the corresponding directory. Since the device is not mounted yet, no containing data will be displayed.



Figure 4.6: At the beginning, the Turaya.Crypt Device is locked.

To access the encrypted device, right-click on the icon and select the mount option: If you have not yet authenticated yourself, the Turaya system will ask you to do so by entering a password. Press [F12] to escape from the Linux window and select the password dialog (see [Figure 4.5](#)). Now, enter the password.

Password: **emscb**



Figure 4.7: Mounting the Turaya.Crypt Device.

Note that the password dialog will only appear if you have not yet authenticated yourself. If you have authenticated yourself already before (e.g., because you have started to use the Turaya.VPN or any other security-critical application), there will be no password dialog again because you have authenticated yourself for the whole Turaya system in a single sign-on mechanism.

If you have authenticated yourself correctly, Turaya.Crypt will mount the device and the lock symbol will disappear (see [Figure 4.8](#)).



Figure 4.8: Turaya.Crypt Device has been mounted.

All data read from the device will be transparently decrypted, while all data written to the device will be transparently encrypted, respectively. Left-click on the icon and the Firefox browser shows you the content of the encrypted device. You should see some PDF documents stored in the device (similar to [Figure 4.9](#)). Click on a document link and a PDF viewer will start and display the document.

You can unmount the device again by right-clicking the icon and choosing the unmount option. The lock symbol will then appear and all data contained in the device will remain encrypted until you mount the device again.

The pre-configured device demonstrates the usage of Turaya.Crypt for a simple case. In the following, we describe how to create new encrypted devices and mount or unmount them manually. However, this is considered for advanced and more experienced users only.

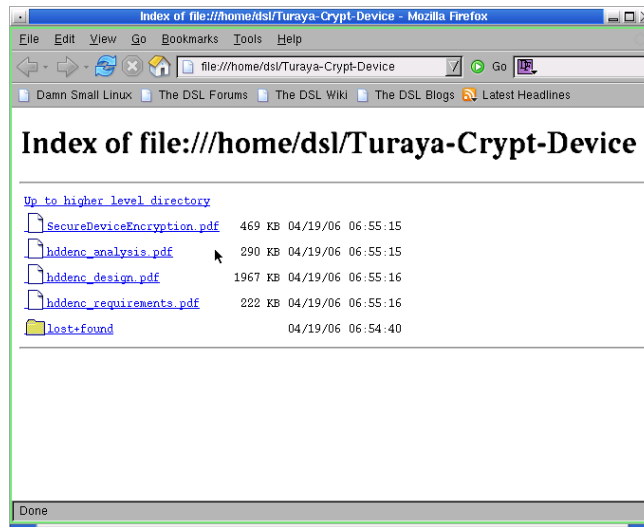


Figure 4.9: Contents of the mounted Turaya.Crypt Device.

4.3.2 Creating encrypted devices with TPM support

Besides using the pre-configured encrypted device, you can create new encrypted devices. However, we recommend to create only virtual devices, since this will not result in any modification of your system.

If your computer system is equipped with a TPM, encrypted devices can be bound to your TPM. Therefore, you need to create a new encrypted device and use a new key. This key is then automatically sealed for your platform. So you cannot decrypt the device on a different computer since that system will have a different TPM or even none.

In the following, we describe how to create an encrypted virtual device and store the image on a USB memory stick so that you can use the virtual device after rebooting the DemoCD.

Creation

Click on the *Turaya.Crypt* icon (see [Figure 4.10](#)).

Select “Create a new encrypted file image” and click on the *Next* button (see [Figure 4.11](#)).

Choose the size of the image, default is 5 MB, and click on *Next* (see [Figure 4.12](#)).

Choose a file name and select the directory where the file should be stored. To store it on a USB memory stick, select `/mnt` in the *Directories* box.² Type in the file name, e.g. `secstore.img`, in the box under the label *Selection* (see [Figure 4.13](#)).

²Note that you have to plug in and mount the memory stick before.



Figure 4.10: Turaya.Crypt.

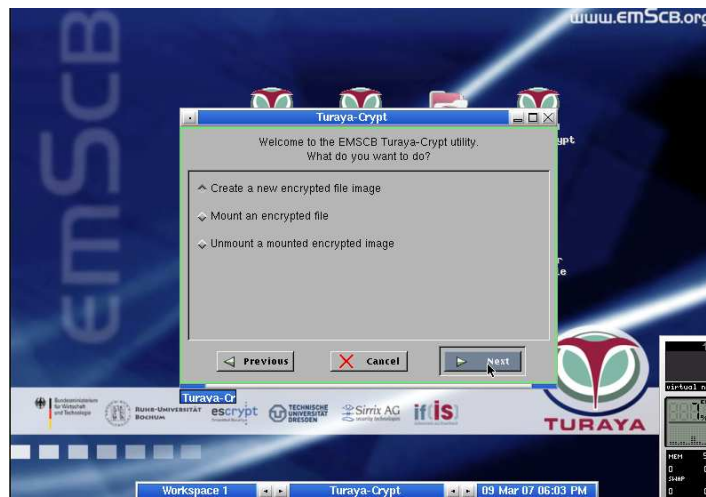


Figure 4.11: Turaya.Crypt: Create new encrypted file image.



Figure 4.12: Turaya.Crypt: Choose device image size.



Figure 4.13: Turaya.Crypt: Choose directory for storage.

Choose key name, e.g. “usbkey”, and type in the name in the dialog box when asked for (see [Figure 4.14](#)).

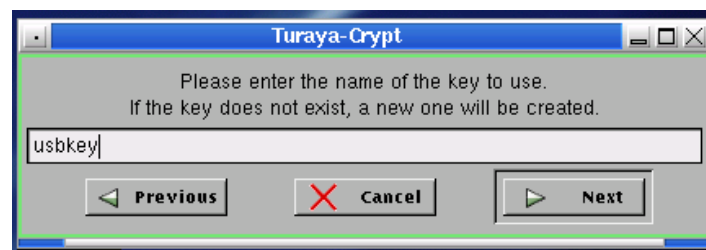


Figure 4.14: Turaya.Crypt: Enter key name.

When the dialog offers to mount the device, click on *yes* (see [Figure 4.15](#)).

Select a mountpoint in the following dialog box, e.g., `/home/ds1/usbstick` (see [Figure 4.16](#)).

Turaya.Crypt will ask for the key ID again. Enter the key name you have just chosen, e.g., “usbkey” (see [Figure 4.17](#)).

Now, you can save data files in the mounted (virtual) device. For example, use the text editor



Figure 4.15: Turaya.Crypt: Asking for mount operation.

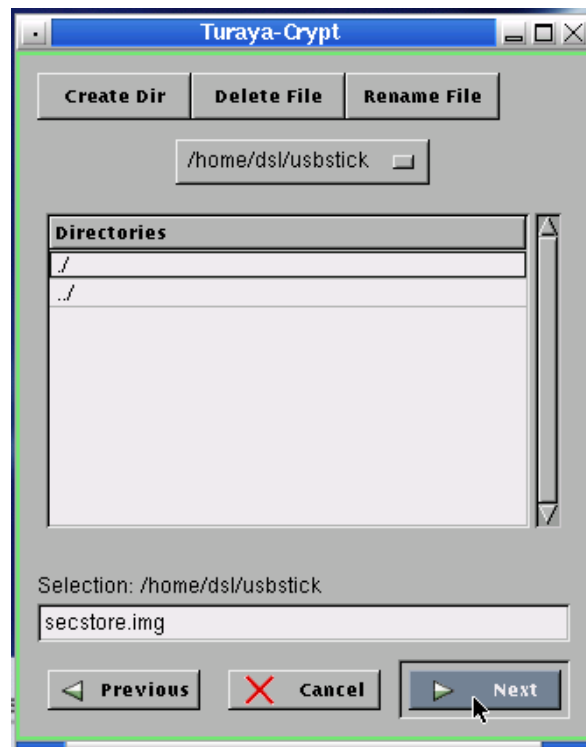


Figure 4.16: Turaya.Crypt: Select mountpoint to mount.

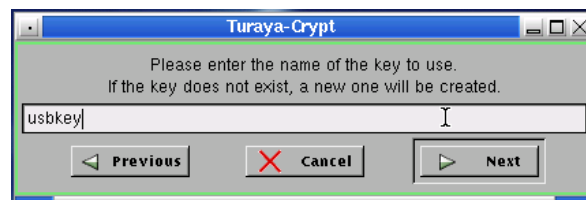


Figure 4.17: Turaya.Crypt: Enter key name again.

“Beaver” to create a text file and save it in the device. Therefore, right-click on the Linux desktop and choose *Apps / Editors / Beaver* (see Figure 4.18), create a new file and save it in the mountpoint directory of the encrypted device, e.g. `/home/dsl/usbstick/myfile.txt`.



Figure 4.18: Open the editor *Beaver* with right-click menu.

After you have finished your work, unmount the encrypted device to safely store all opened files and data. Therefore, click on the *Turaya.Crypt* icon, select “Unmount a mounted encrypted image” and click on *Next* (see Figure 4.19).



Figure 4.19: Turaya.Crypt: Unmount encrypted image.

Select the mountpoint of the encrypted image, e.g. `/home/dsl/usbstick`, and click on *Next* (see Figure 4.20).

Verification

1. Plug in USB stick on reboot
2. Mount existing Turaya.Crypt Device to check if this still works

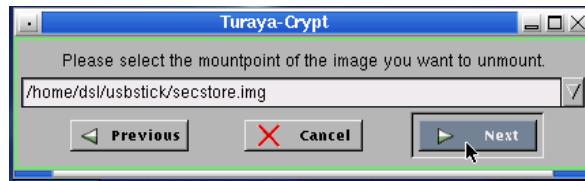


Figure 4.20: Turaya.Crypt: Select mountpoint to unmount.

3. Click on Turaya.Crypt icon
4. Choose “mount”
5. Select encrypted image file on USB stick (e.g. `secstore.img` on the stick mounted in `/mnt`)
6. Select mountpoint, e.g. `/home/dsl/usbstick`
7. Enter key name previously chosen on request, e.g. “usbkey”
8. Open the text file previously created and check its content.

4.3.3 Command-line configuration tool

Besides the pre-configured encrypted device and the graphical configuration tool, there is also a command-line interface to create, mount, and unmount encrypted devices. This tool is more powerful because you can also create encrypted partitions on your hard-drive instead of virtual devices as image file. However, we recommend to create only virtual devices, since this will not result in any modification of your system.³ You can use the “`turaya-crypt`” Linux command to create, mount and unmount encrypted devices manually. To enter the Linux command line mode, start a shell by clicking on the “ATerminal” icon. Remember that when you create or mount an encrypted device (image file or partition), the password dialog of Turaya will appear unless you have authenticated yourself once before.

Execute “`turaya-crypt`” for information about the usage of the Turaya.Crypt device encryption. “`turaya-crypt -h`” brings up a more detailed list:

```
usage: /bin/turaya-crypt <command>
available commands:
(-c | --create) destination [size] [keyid]
destination
    filename of the encrypted disc or device
    e.g. temp.img or /dev/hda2
size
    required if destination is a filename
    size of the resulting image in megabytes, minimum: 2
keyid
    parameter to specify the encryption key to be used
```

³Note that if you create an encrypted partition on your hard-drive, data previously stored in this partition will be overwritten. You should make a backup in order to prevent any data loss.

```

(-m | --mount) destination mountpoint [keyid]
    destination
        filename of the encrypted disc or device
    mountpoint
        the directory where the image file should be mounted
    keyid
        parameter to specify the encryption key to be used

(-u | --umount) mountpoint
    mountpoint
        the directory where the image file is mounted

(-h | --help)

(-v | --version)

--debug
    enable debug mode

```

Let's give an example. You might have some files that you want to store in a secure way.

At first, create a Turaya.Crypt *image file*. This is the space where your data will be stored. Let's assume that 2 megabytes of storage will be enough for your demands. Execute the command

```
turaya-crypt -c /home/dsl/storage.img 2 mykey1
```

where “-c” is the short form of “create”, “*storage.img*” is the name of the image file (that you can choose arbitrarily and that will be stored in your home directory “/home/dsl”), 2 is the size of the image file in megabytes, and *mykey1* is the ID of the key to be used.

Before the image file can be used to securely store data, it has to be *mounted*. This means that you have to specify a path in the file system that you want to use for direct access to the image file.

For this, create a new directory using the command

```
mkdir /home/dsl/secret
```

This directory will be the *mountpoint* of the image file – that is, the path that can be used for direct access to the image file as soon as it is mounted with the command

```
turaya-crypt -m /home/dsl/storage.img /home/dsl/secret mykey1
```

where “-m” is the short form of “mount”, “/home/dsl/storage.img” is the location of the image file, and “/home/dsl/secret” is the new path for direct access to the secret storage

(“the mountpoint of the image file”), and `mykey1` is again the ID of the key used at creation time of the image file.

Now you are ready to go! You can simply save your files that need to be stored securely to `/home/dsl/secret`. This directory can now be used just like any other directory, with the difference that all data in `/home/dsl/secret` will be automatically encrypted (when writing files) and decrypted (when reading files) by the `Turaya.Crypt` service.

When you are done, you can “unmount” the image file so that the path for direct access to the image file is no longer available (which also means that your secret data in the image file can no longer be accessed!). This is achieved by executing

```
turaya-crypt -u /home/dsl/secret
```

where `-u` is the short form of “unmount” and `/home/dsl/secret` is the mountpoint of the image file.

If you want to access your data (or store new data) in the image file again at a later time, you just have to mount the image file by executing `turaya-crypt -m /home/dsl/storage.img /home/dsl/secret` again and find your files in the directory `/home/dsl/secret`. Of course, this will only work if you use the same keyID that you used when you created the image file.

Note: Since this is a DemoCD with a live starting system that uses a ramdisk as filesystem, all files that you will save within the DemoCD filesystem will be lost when you reboot the system. To keep your files, i.e., “storage.img”, copy the file to a different computer system (e.g. using the Linux ssh command) or mount an existing physical hard disk partition (e.g. a USB memory stick) where you can copy your image file.

4.4 Turaya.FairDRM

This section describes the Turaya.FairDRM demonstrator. Please note that Turaya.FairDRM is still “work in progress” and thus has certain restrictions regarding usability aspects and implementation completeness.

4.4.1 Requirements

In order to use Turaya.FairDRM, the client platform needs to fulfill these requirements:

- TPM 1.1
- USB stick or network to transfer files among platforms

4.4.2 Certificate generation

Before you can download media files from the Turaya.FairDRM web server, you must generate a client certificate. This certificate contains information about your platform configuration and can therefore be used to bind content to your platform (so that the content can only be accessed by your platform).

1. To generate a client certificate, click on the “Retrieve Certificate” icon first (see [Figure 4.21](#)).

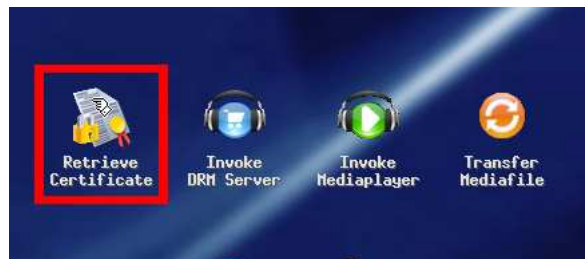


Figure 4.21: Turaya.FairDRM: Retrieve certificate.

2. Then, choose a certificate filename, which defaults to “client.cert” (see [Figure 4.22](#)).

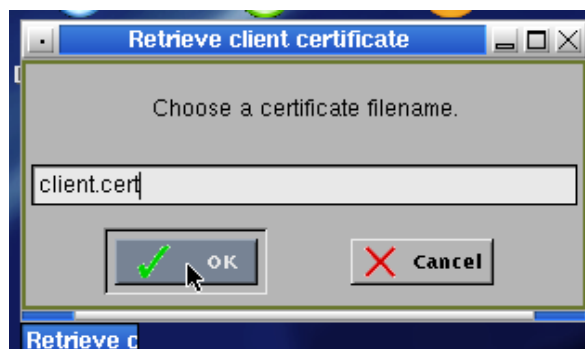


Figure 4.22: Turaya.FairDRM: Choose certificate filename.



Figure 4.23: Turaya.FairDRM: Certificate generation progress.

3. Wait for the TPM to generate a certificate. Depending on the TPM, this procedure can take one minute or more to finish (see [Figure 4.23](#)).
4. If the operation was successful, click ok to proceed (see [Figure 4.24](#)).

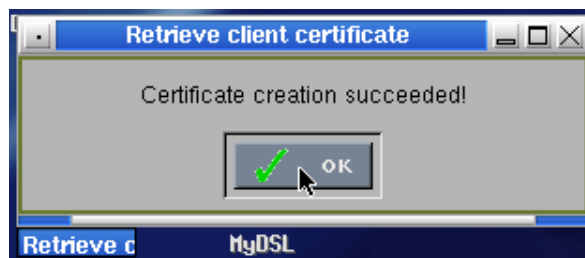


Figure 4.24: Turaya.FairDRM: Certificate generation success.

If everything worked fine, you now have your client certificate (in the `/home/dsl` directory) and can continue to download media files from the Turaya.FairDRM web server.

4.4.3 Media Download

To download a DRM-protected media file from the Turaya.FairDRM web server, follow these steps:

1. Click on the “Invoke DRM Server” icon (see [Figure 4.25](#)).



Figure 4.25: Turaya.FairDRM: Invoke DRM server.

2. Choose the client certificate file. Use the “Browse...” button to open a file requester and select the previously generated certificate (e.g. “client.cert”).
3. Select a media file from the list.
4. Choose the license conditions. There are three types of conditions:

- Playback Count: the number of times the media file can be played back.
 - Playback Period: the time interval during which the media file can be played back. Notice that the period starts at the moment of generation of the resulting DRM-protected media file.
 - Transfer Count: the number of times the media file can be transferred to another platform.
5. Click “Proceed” to let the DRM server generate the DRM-protected media file.
 6. Finally, click the “Click here” button to download the generated file to your platform.

The downloaded DRM-protected media file now resides on your platform. You can use the Mediaplayer to play it.

4.4.4 Media Playback

To start the playback of the downloaded DRM-protected media file, follow these steps:

1. Click on the “Invoke Mediaplayer” icon (see [Figure 4.26](#)).



Figure 4.26: Turaya.FairDRM: Invoke mediaplayer.

2. Choose the DRM-protected media file (see [Figure 4.27](#)).

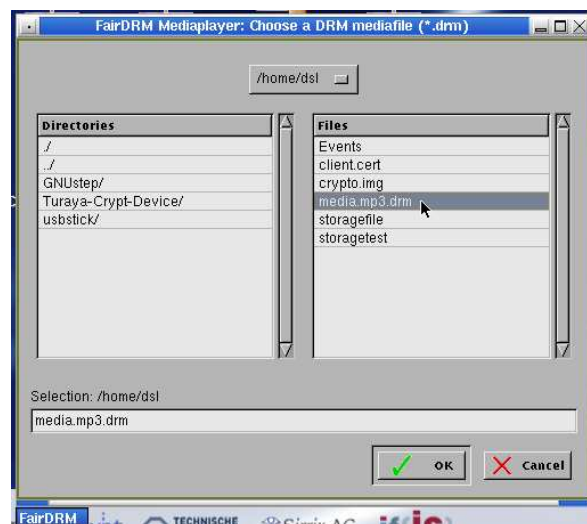


Figure 4.27: Turaya.FairDRM: Choose media file.

3. While the license is checked and the audio is played back, a progress window is displayed (Figure 4.28).



Figure 4.28: Turaya.FairDRM: Media playback progress.

4. If the license is met, the content is played back (see Figure 4.29).



Figure 4.29: Turaya.FairDRM: Media playback success.

4.4.5 Media Transfer

To transfer a DRM-protected media file to another platform, follow these steps:

1. First, generate a certificate on the destination platform (choose e.g. "destination.cert").
2. Copy the resulting certificate file to the source platform.
3. It is assumed that a DRM-protected media file resides on the source platform.
4. Click on the "Transfer Mediafile" icon on the source platform (see Figure 4.30).



Figure 4.30: Turaya.FairDRM: Start transfer of media file.

5. Then, choose the DRM-protected source media file you want to transfer. In this example, the file "media.mp3.drm" is chosen (see Figure 4.31).
6. Choose the destination certificate, i.e., the certificate created on the destination platform. In this example, the file is called "destination.cert" (see Figure 4.32).

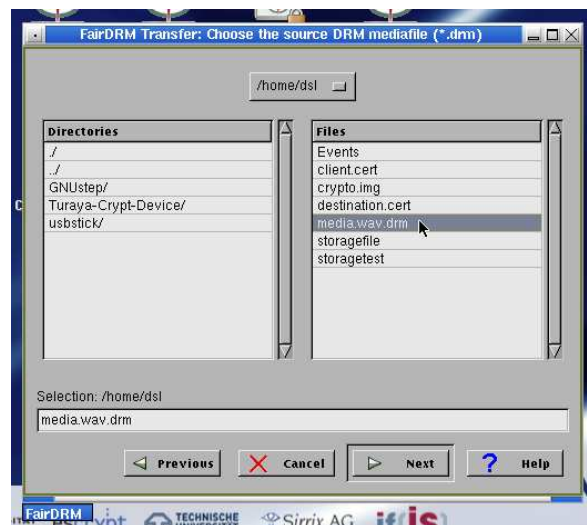


Figure 4.31: Turaya.FairDRM: Choose source DRM media file.

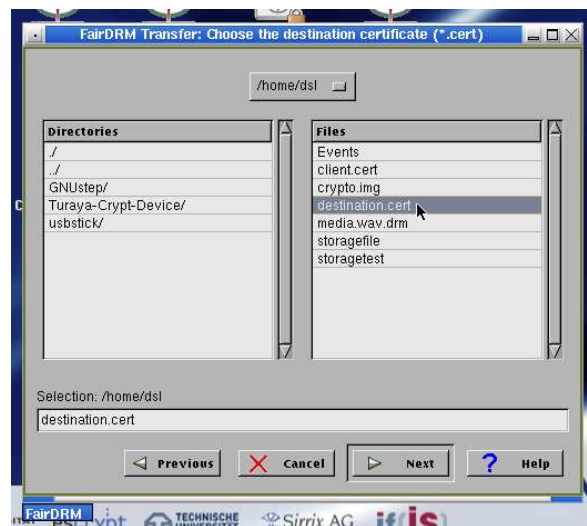


Figure 4.32: Turaya.FairDRM: Choose destination certificate.

7. Choose the destination media file name. For example, you can choose “destination.mp3.drm” (see [Figure 4.33](#)).
8. Please wait while the transfer is in progress. This can take several seconds, depending on the type of TPM installed in your platform (see [Figure 4.34](#)).
9. If the transfer was successful, click OK to proceed (see [Figure 4.35](#)).
10. Finally, copy the destination DRM-protected media file to the destination platform. If the license is met, the content can be played back on the destination platform.

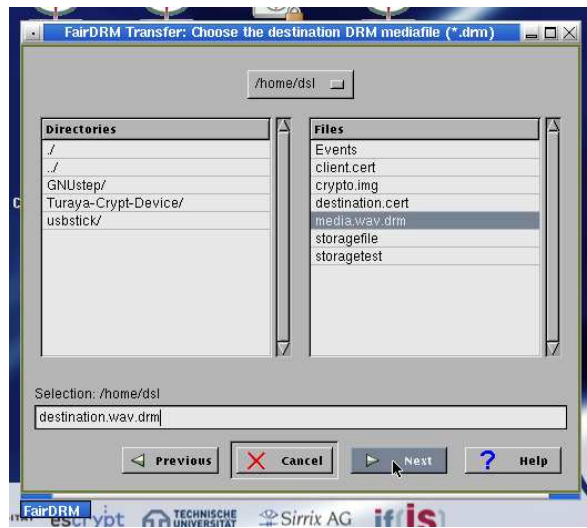


Figure 4.33: Turaya.FairDRM: Choose destination DRM media filename.

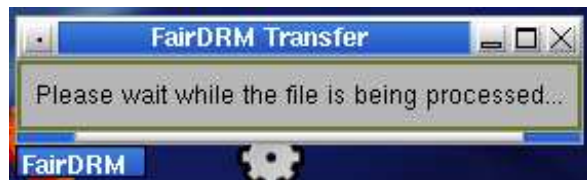


Figure 4.34: Turaya.FairDRM: Transfer progress.

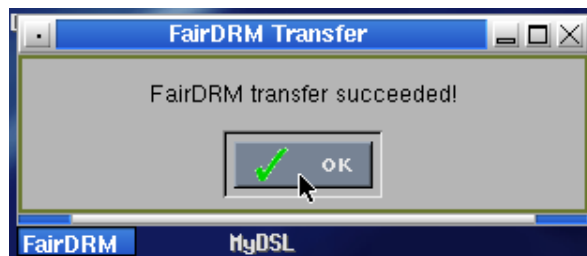


Figure 4.35: Turaya.FairDRM: Media transfer success.

Appendix A

Known Issues and Problems

This DemoCD is a snapshot of work in progress and not a finished product. Known problems include:

- When using the Legacy Linux compartment for the first time, the mouse has to be moved a bit and then stopped. Only after this, the mouse pointer reacts normally.
- USB mouse on IBM ThinkPad may not work properly unless you touch the TrackPoint or the TouchPad once.
- Currently, we support the German keyboard layout only.
- A USB keyboard can only be used within DSL Linux, but not within the μ GUI.
- To safely unmount a USB memory stick, shutdown the User Linux, i.e., press **Ctrl + Alt + Del** twice.

Appendix B

Initializing the TPM

The Turaya DemoCD contains a TPM initialization tool for detection and proper initialization of a TPM. To use it, you should first enable the TPM in the BIOS configuration and clear any eventually existing TPM owner password. (Note that if you used your TPM before and already have TPM-protected data on your computer, you should not clear your owner password! Otherwise, you won't be able to access that data anymore!) After that, Turaya.TPM-Initialization is able to detect and initialize common 1.1 TPMs.

To initialize a TPM version 1.1, switch to the Turaya.TPM-Initialization compartment, skip the detection screen (Figure B.1) by pressing Enter and confirm the initialization by pressing Enter again (Figure B.2).

After Turaya has properly initialized the TPM, a success screen will pop up (Figure B.3). Otherwise, an informative error message will be shown. In this case please consult our website <http://www.emscb.org> on how to get support.

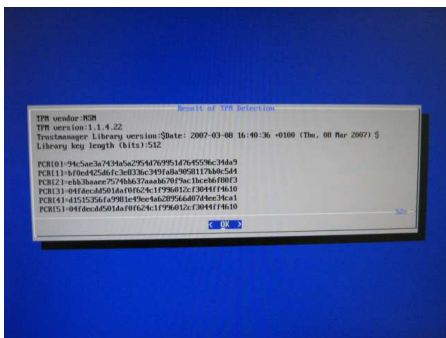


Figure B.1: TPM Detection

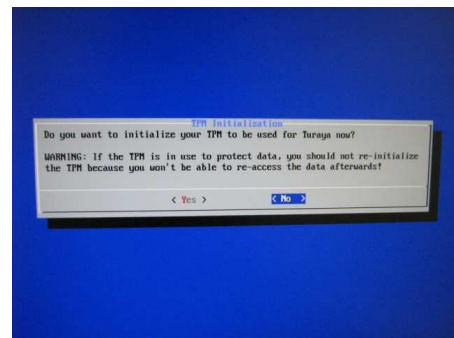


Figure B.2: Confirm Initialization

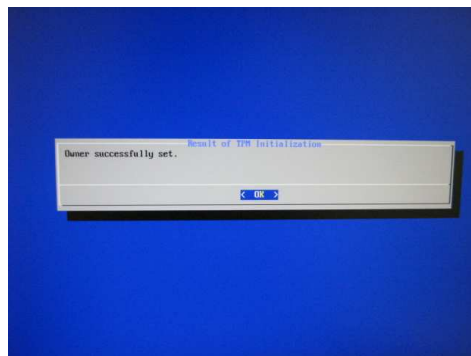


Figure B.3: Success Screen